



Smart Home Pet Care

Software Design Specification

2022. 05. 15.

Introduction to Software Engineering 42
Team 2

Team Leader	차현묵
Team Member	곽재원
Team Member	김준식
Team Member	양승찬
Team Member	유새하
Team Member	이경돈
Team Member	최하늘

	오류! 책갈피가 정의되어 있지 않습니다.
1. Preface	
1.1. Readership	13
1.2. Scope	13
1.3. Objective	13
1.4. Document Structure	14
2. Introduction	14
2.1. Objectives	15
2.2 Applied Diagrams	15
2.2.1. UML	15
2.2.2. Use Case Diagram	15
2.2.3. Class Diagram	15
2.2.4. Sequence Diagram	16
2.2.5. Context Diagram	16
2.2.6. Entity Relationship Diagram	16
2.3. Applied tools	17
2.3.1. Microsoft Word	17
2.3.2. Google Docs	17
2.3.3. Draw.io	17
2.3.4. Figma	17
2.4 Project Scope	18
2.5 References	18
3. System Architecture - Overall	18
3.1. Objectives	18
3.2. System Organization	18

Smart Home Pet Care	Design Specification
---------------------	----------------------

3.2.1. Context Diagram	19
3.2.2. Sequence Diagram	20
3.2.3. Use Case Diagram	25
4. System Architecture - Frontend	26
4.1. Objectives	26
4.2. Subcomponents	26
4.2.1. 계정 관리	26
4.2.1.1. Attributes	26
4.2.1.2. Methods	26
4.2.1.3. Class Diagram	26
4.2.1.4. Sequence Diagram	27
4.2.2. IoT 디바이스 등록 및 설정	28
4.2.2.1. Attributes	28
4.2.2.2. Methods	29
4.2.2.3. Class Diagram	29
4.2.2.4. Sequence Diagram	30
4.2.3. 반려동물 이상 행동 앱 푸시 알림	31
4.2.3.1. Attributes	31
4.2.3.2. Methods	31
4.2.3.3. Class Diagram	32
4.2.3.4. Sequence Diagram	32
4.2.4. 반려동물 실시간 위치 파악	33
4.2.4.1. Attributes	33
4.2.4.2. Methods	33
4.2.4.3. Class Diagram	33
4.2.4.4. Sequence Diagram	34

Smart Home Pet Care	Design Specification
---------------------	----------------------

4.2.5. 반려동물 행동 로그	35
4.2.5.1. Attributes	35
4.2.5.2. Methods	35
4.2.5.3. Class Diagram	36
4.2.5.4. Sequence Diagram	36
4.2.6 반려동물 건강상태 획득/분석	37
4.2.6.1. Attributes	37
4.2.6.2. Methods	38
4.2.6.3. Class Diagram	38
4.2.6.4. Sequence Diagram	39
4.2.7. 반려동물 맞춤 케어 시스템 - IoT 디바이스 상태 확인	40
4.2.7.1. Attributes	40
4.2.7.2. Methods	41
4.2.7.3. Class Diagram	41
4.2.7.4. Sequence Diagram	42
4.2.7. 반려동물 맞춤 케어 시스템 설정	42
4.2.7.1. Attributes	42
4.2.7.2. Methods	42
4.2.7.3. Class Diagram	43
4.2.7.4. Sequence Diagram	43
5. System Architecture - Backend	44
5.1. Objectives	45
5.2. Subcomponents	45
5.2.1. 계정 관리	45
5.2.1.1. Class Diagram	45
5.2.1.2. Sequence Diagram	45

5.2.2. IoT 디바이스 등록 및 설정	46
5.2.2.1. Class Diagram	46
5.2.2.2. Sequence Diagram	47
5.2.3. 반려동물 이상 행동 앱 푸시 알림	48
5.2.3.1. Class Diagram	48
5.2.3.2. Sequence Diagram	49
5.2.4. 반려동물 실시간 위치 파악	49
5.2.4.1. Class Diagram	49
5.2.4.2. Sequence Diagram	49
5.2.5. 반려동물 행동 로그	50
5.2.5.1. Class Diagram	51
5.2.5.2. Sequence Diagram	51
5.2.6 반려동물 건강상태 획득/분석	52
5.2.6.1. Class Diagram	52
5.2.6.2. Sequence Diagram	53
5.2.7. 반려동물 맞춤 케어 시스템 설정	53
5.2.7.1. Class Diagram	53
5.2.7.2. Sequence Diagram	54
6. Protocol Design	55
6.1. Objectives	55
6.2. 전달 형식	55
6.2.1. HTTP	55
6.2.2. OAuth	55
6.2.3. JSON	56
6.3. Authentication	56
6.3.1. 사용자 인증	56

6.4. IoT 디바이스 등록 및 설정	57
6.4.1. 연결 가능 IoT 디바이스 목록 조회	57
6.4.2. 신규 IoT 디바이스 등록 및 설정	58
6.5. 반려동물 실시간 위치 파악	59
6.5.1. IoT 디바이스 위치 조회	59
6.5.2. 디바이스 위치 이동	60
6.6. 반려동물 행동 로그	61
6.6.1. 반려동물 행동 로그 조회	61
6.7. 반려동물 건강체크	62
6.7.1. 반려동물 건강 지표 조회	62
6.8. 반려동물 맞춤 케어	63
6.8.1. IoT 디바이스 상태 조회	63
6.8.2. IoT 디바이스 작동 설정	64
7. Database Design	65
7.1. Objectives	65
7.2. ER Diagram	65
7.2.1. Entities	66
7.2.1.1. User	66
7.2.1.2. Pet	67
7.2.1.3. Device	67
7.2.1.4. DeviceFunctions	68
7.2.1.5. DeviceTimeline	69
7.2.1.6. Log	69
7.2.1.7. HealthProfile	70
7.2.1.8. Stress/Oxygen/Pulse/Temperature	70
7.3. Relational Schema	71

Smart Home Pet Care	Design Specification
7.4. SQL DDL	71
7.4.1. User	71
7.4.2. Pet	72
7.4.3. Device	72
7.4.4. DeviceFunctions	72
7.4.5. DeviceTimeline	73
7.4.6. Log	73
7.4.7. HealthProfile	74
7.4.8. Stress/Oxygen/Pulse/Temperature	74
8. Testing Plan	75
8.1. Objectives	75
8.2. Testing Policy	75
8.2.1. Development Testing	75
8.2.1.1. Unit Testing	71
8.2.1.2. Component Testing	76
8.2.1.3. System Testing	76
8.2.2. Release Testing	77
8.2.2.1. Requirements-based testing	78
8.2.2.2. Scenario testing	78
8.2.2.3. Performance testing	78
8.2.3. User Testing	79
8.2.4. Testing Case	79
9. Development Plan	80
9.1. Objectives	80
9.2. Frontend Environment	80
9.2.1. Figma	80

Smart Home Pet Care	Design Specification
----------------------------	-----------------------------

9.2.3. Flutter	80
9.3. Backend Environment	81
9.3.1. AWS	81
9.3.2. Node.js	82
9.3.3. MariaDB	82
9.4. Constraints	83
9.5. Assumptions and Dependencies	83
10. Supporting Information	83
10.1. Software Design Specification	84
10.2. Document History	84

표 목차

표 1. 요청 - 사용자 인증	56
표 2. 응답 - 사용자 인증	56
표 3. 요청 - 연결 가능 IoT 디바이스 목록 조회	57
표 4. 응답 - 연결 가능 IoT 디바이스 목록 조회	57
표 5. 요청 - 신규 IoT 디바이스 등록 및 설정	58
표 6. 응답 - 신규 IoT 디바이스 등록 및 설정	58
표 7. 요청 - IoT 디바이스 위치 조회	59
표 8. 응답 - IoT 디바이스 상태 조회	59
표 9. 요청 - 디바이스 위치 이동	60
표 10. 응답 - 디바이스 위치 이동	60
표 11. 요청 - 반려동물 행동 로그 조회	61
표 12. 응답 - 반려동물 행동 로그 조회	61
표 13. 요청 - 반려동물 건강 지표 조회	62
표 14. 응답 - 반려동물 건강 지표 조회	62
표 15. 요청 - IoT 디바이스 상태 조회	63
표 16. 응답 - IoT 디바이스 상태 조회	63
표 17. 요청 - IoT 디바이스 작동 설정	64
표 18. 응답 - IoT 디바이스 작동 설정	64
표 19. SDS Document History	84

그림 목차

그림 1. Context Diagram	19
그림 2. Sequence Diagram - 계정관리	19
그림 3. Sequence Diagram - 이상행동 Push 알림	20
그림 4. Sequence Diagram - 반려동물 위치 정보	20
그림 5. Sequence Diagram - 반려동물 로그 조회	21
그림 6. Sequence Diagram - IoT 디바이스 관리	22
그림 7. Sequence Diagram - 반려동물 건강상태 관리	23
그림 8. Use Case Diagram	24
그림 9. Class Diagram - 계정 관리 Frontend	26
그림 10. Sequence Diagram - 계정 관리 Frontend	27
그림 11. Class Diagram - IoT 디바이스 등록 및 설정 FrontEnd	29
그림 12. Sequence Diagram - IoT 디바이스 등록 및 설정 FrontEnd	30
그림 13. Class Diagram - 반려동물 이상행동 앱푸시 알림 Frontend	31
그림 14. Sequence Diagram - 반려동물 이상행동 앱푸시 알림 Frontend	32
그림 15. Class Diagram - 반려동물 실시간 위치 파악 FrontEnd	33
그림 16 Sequence Diagram - 반려동물 실시간 위치 파악 FrontEnd	34
그림 17. Class Diagram - 반려동물 행동 로그 Front-End	35
그림 18. Sequence Diagram - 반려동물 행동 로그 Frontend	36
그림 19. Class Diagram - 반려동물 건강상태 획득/분석 FrontEnd	39
그림 20. Sequence Diagram - 반려동물 건강상태 획득/분석 FrontEnd	40
그림 21. Class Diagram - 반려동물 맞춤 케어 시스템 - IoT 디바이스 상태 확인 Frontend	41
그림 22. Sequence Diagram - 반려동물 맞춤 케어 시스템 - IoT 디바이스 상태 확인 Frontend	42
그림 23. Class Diagram - 반려동물 맞춤 케어 시스템 설정 Frontend	43
그림 24. Sequence Diagram - 반려동물 맞춤 케어 시스템 설정 Frontend	44
그림 25. Class Diagram - 계정 관리 Backend	45

그림 26. Sequence Diagram - 계정 관리 Backend	46
그림 27. Class Diagram - IoT 디바이스 등록 및 설정 BackEnd	47
그림 28. Sequence Diagram - IoT 디바이스 등록 및 설정 BackEnd	48
그림 29. Class Diagram - 반려동물 이상행동 앱푸시 알림 Backend	48
그림 30. Sequence Diagram - 반려동물 이상행동 앱푸시 알림 Backend	49
그림 31. Class Diagram - 반려동물 실시간 위치파악 Backend	49
그림 32. sequence Diagram - 반려동물 실시간 위치파악 Backend	50
그림 33. Class Diagram - 반려동물 행동 로그 Backend	51
그림 34. Sequence Diagram - 반려동물 행동 로그 Backend	52
그림 35. Class Diagram - 반려동물 건강상태 획득/분석 Backend	52
그림 36. Sequence Diagram - 반려동물 건강상태 획득/분석 Backend	53
그림 37. Class Diagram - 반려동물 맞춤 케어 시스템 설정 Backend	54
그림 38. Sequence Diagram - 반려동물 맞춤 케어 시스템 설정 Backend	55
그림 39. ER Diagram	66
그림 40. ER Diagram - User	66
그림 41. ER Diagram - Pet	67
그림 42. ER Diagram - Device	67
그림 43. ER Diagram - DeviceFunctions	68
그림 44. ER Diagram - DeviceTimeline	69
그림 45. ER Diagram - Log	69
그림 46. ER Diagram - HealthProfile	70
그림 47. ER Diagram - Stress/Oxygen/Pulse/Temperature	70
그림 48. Relational Schema	71
그림 49. 반려동물 맞춤 케어 HealthProfile 클래스	76
그림 50. 반려동물 로그 조회 Sequence Diagram	77
그림 51. Figma 로고	80

그림 52. Flutter 로고	81
그림 53. AWS 로고	81
그림 54. NodeJS 로고	82
그림 55. MariaDB 로고	82

1. Preface

이 챕터에서는 본 디자인 명세서의 독자층, 범주, 본 문서의 목적 그리고 본 디자인 명세서의 구조에 대해서 기술한다.

1.1. Readership

본 디자인 명세서의 내용은 총 10 개의 챕터와 각 챕터를 이루고 있는 서브 섹션으로 구성되었다. 각각의 챕터는 시스템의 전체적인 구조, frontend 시스템 구조, backend 시스템 구조, 프로토콜, 데이터베이스 디자인, 테스트 계획, 개발 계획의 내용을 다룬다. 본 디자인 명세서의 chapter에 대한 자세한 내용은 1.4 Document Structure에 기술되어 있다. 본 디자인 명세서는 2 팀의 프로젝트 진행과정에 활용될 것을 전제로 작성되었고 그에 따라 2 팀을 주요한 독자로 상정하여 작성하였다. 또한 본 디자인 명세서는 소프트웨어 공학개론의 교수, 조교, 학생들이 열람한다는 전제하에 작성하였다.

1.2. Scope

본 디자인 명세서는 반려동물을 위한 원격 케어 시스템인 Smart Home Pet Care의 개발 과정에서 소프트웨어 공학(Software Engineering)과 소프트웨어 품질 공학(Software Quality Engineering)에 사용된다.

1.3. Objective

본 디자인 명세서의 목적은 반려동물 원격 케어 시스템인 Smart Home Pet Care를 개발하기 위해 필요한 기술적 디자인 요소를 기술하는데 있다. 본 문서는 Smart Home Pet Care의 개발 과정에 사용된 소프트웨어 아키텍쳐(Software Architecture)와 소프트웨어 디자인 결정요소(Software Design Decisions)가 기술되어 있다. 또한 본 문서에서는 전체적인 시스템 아키텍처를 다양한 관점에서 바라본 조망을 묘사하고 있다. 추가로 본 문서에서는, 요구사항 명세서에서 다룬 모듈들의 디자인과 구조를 더욱 구체화 하였으며 use case 들을 sequential diagram과 class diagram으로 기술하여 개발에 직접적으로 활용할 수 있도록 하였다. 본 문서의 독자층은 본 문서의 이해관계자, 본 시스템의 개발자, 시스템 디자이너, 시스템 테스터를 상정하고 있으나, 상황에 따라 독자층은 더욱 확대될 수 있다.

1.4. Document Structure

1. Preface: 이 챕터에서는 본 문서의 독자층, 본 문서의 범주, Smart Home Pet Care 시스템의 목적, 본 문서의 구조에 대해서 기술한다.
2. Introduction: 이 챕터에서는 본 프로젝트에서 활용된 diagram 과 tool 에 대해서 기술하고, 프로젝트의 목적에 대해 기술한다. 또한 본 명세서의 작성에서 참고한 참고문헌을 언급한다.
3. Overall System Architecture: 이 챕터에서는 시스템의 전체적인 아키텍처를 context diagram, sequence diagram 그리고 use case diagram 을 활용해서 묘사한다.
4. System Architecture – Frontend: 이 챕터에서는 frontend 시스템 아키텍처를 class diagram 과 use case diagram 을 활용해서 묘사한다.
5. System Architecture – Backend: 이 챕터에서는 backend 시스템 아키텍처를 class diagram 과 sequence diagram 을 활용해서 묘사한다.
6. Protocol Design: 이 챕터에서는 클라이언트와 서버 사이의 통신에 사용되는 프로토콜에 대해서 기술한다.
7. Database Design: 이 챕터에서는 데이터 베이스의 구조를 ER diagram 과 SQL DDL 을 이용해서 묘사한다.
8. Testing Plan: 이 챕터에서는 본 시스템의 테스트 계획을 기술한다.
9. Development Plan: 이 챕터에서는 본 시스템의 개발에 사용된 tool, 본 시스템의 제약사항과 가정, 시스템의 의존성에 대해 다룬다.
10. Supporting Information: 이 챕터에서는 본 명세서의 수정 기록을 기술한다.

2. Introduction

Smart Home Pet Care 의 목적은 반려동물을 기르는 사용자가 원격으로도 집안의 IoT 기기를 제어하고 이를 통해 반려동물을 원격으로 돌볼 수 있도록 하는 것이다. Smart Home Pet Care 를 통해 사용자는 집안의 IoT 기기를 원격으로 제어하여 반려동물에게 먹이를 주거나 놀이 로봇을 통해 놀아주면서 상호작용할 수 있다. 또한 청소 로봇과 공기 관리 기기를 통해 반려동물이 생활하기 좋은 환경을

지속적으로 조성할 수 있다. 반려동물에 장착된 센서를 활용해 반려동물의 생체신호를 지속적으로 수집하여 반려동물의 건강상태를 지속적으로 확인할 수 있다. 사용자는 IoT 기기의 상태와 반려동물의 상태를 한눈에 확인하면서 원격으로 반려동물을 지속적으로 케어할 수 있다. 본 디자인 명세서에는 본 시스템의 개발을 위해 사용된 디자인 구조들이 묘사되어 있다. 묘사된 디자인 구조들은 이전에 작성한 요구사항 명세서에서 명시된 요구사항에 따라 설계되었다.

2.1. Objectives

이 챕터에서는 본 프로젝트에서 활용된 diagram 과 tool 에 대해서 기술한다.

2.2 Applied Diagrams

2.2.1. UML

UML(Unified Modeling Language)는 요구사항 분석, 시스템 디자인 그리고 시스템 구현과 같은 시스템 개발 과정에서 개발자들 간의 의사소통을 용이하게 하기 위해 사용되는 표준화된 모델링 언어이다. UML 이 가진 좋은 표현력과 논리적 표기법은 개발자들 간의 소통을 용이하기 해준다. 또한 UML 을 사용하면 시스템의 구조를 모델링하는 과정에서 빠진 부분이나 상충되는 부분을 찾기 쉽다. 그 분 아니라 UML 은 프로젝트의 규모와 상관없이 적용될 수 있다. UML 은 use case diagram 과 class diagram 을 이용해서 묘사할 수 있는 객체지향 소프트웨어의 개발의 분석과 디자인을 용이하게 해준다.

2.2.2. Use Case Diagram

Use Case Diagram 은 시스템이 제공하는 기능적 단위를 묘사할 수 있다. Use Case Diagram 의 주요 목적은 기능적 요구사항을 시각화 하는데 있다. 여기서 기능적 요구사항은 다른 use case 와의 관계, 다른 시스템 과의 관계를 포함한다. Use Case Diagram 은 시스템의 사용자를 묘사하는 Actor, 시스템의 주요 기능을 나타내는 Scope, 사용자가 실제 시스템을 사용할 때 맞이하게 되는 상황인 Use Case 로 이루어져 있다.

2.2.3. Class Diagram

Class Diagram 은 시스템 내에서 객체가 다른 객체와 이루는 관계를 보여준다. Class Diagram 은 시스템의 정적인 구조를 보여준다고 할 수 있다. Class Diagram 은 개발자들이 객체 지향 프로그램을 구현하는 과정에서 사용하는 class 의 구조를 보여준다. Class Diagram 에서 class 은 3 행 1 열의 구조를

보여주는데, 1 행에서는 class 의 이름, 2 행에서는 class 의 attribute, 3 행에서는 class 의 method 를 나타낸다.

2.2.4. Sequence Diagram

Sequence Diagram 은 use case 또는 use case 의 일부분에 대해서 구체적인 흐름을 묘사한다. Sequence diagram 은 객체간의 관계를 묘사하고, 객체가 다른 객체를 호출하는 과정이 구체적으로 묘사된다. Sequence Diagram 을 세로축으로 보면, 객체의 함수 호출/객체가 받은 메시지를 시간순으로 확인할 수 있다. 또한 Sequence Diagram 을 가로축으로 보게 될 경우 함수를 호출하거나 메시지를 전송하는 객체를 확인할 수 있다. Sequence Diagram 에서 각 객체는 diagram 의 위 쪽에 박스로 묘사되어 있다.

2.2.5. Context Diagram

Context Diagram 은 시스템의 경계와 시스템의 환경, 시스템이 상호작용하는 객체를 묘사한다. Context Diagram 은 시스템을 표현하는 가장 높은 레벨의 diagram 으로 시스템의 정적인 구조를 보여준다. Context Diagram 에서는 구체적인 내부적 구조는 묘사하지 않고 중앙 시스템과 시스템의 주변 환경, 시스템과 상호작용하는 객체를 묘사한다. Context Diagram 는 전체적인 시스템 요구사항과 제약사항을 정의하는 과정에서 시스템이 고려해야 할 외부적인 요인에 초점을 맞추는 것을 목표로 한다. Context Diagram 은 모든 이해관계자들이 읽어야 하므로, 전문적인 용어가 아닌 이해하기 쉬운 용어로 쓰여질 필요가 있다.

2.2.6. Entity Relationship Diagram

Entity Relationship Diagram 은 각 객체를 이루고 있는 attribute 와 객체간의 관계를 묘사하는 diagram 이다. Entity Relationship Diagram 은 각 객체를 잇고 있는 네트워크의 형태로 나타내게 된다. 각 객체는 직사각형, 객체가 가지고 있는 attribute 는 타원형으로 묘사된다. 객체와 객체가 가지고 있는 attribute 는 실선으로 연결되는 것으로 묘사된다. 또한 객체 간의 관계는 마름모와 화살표를 통해

묘사된다.

2.3. Applied tools

2.3.1. Microsoft Word

본 툴은 2 팀에서 프로젝트의 요구사항 명세서와 디자인 명세서를 작성하는데 사용되었다. 모든 팀원들이 공통적으로 사용하고 있는 문서 작업 툴일 뿐 아니라, 문법과 맞춤법 교정들의 여러 편의 기능을 제공한다.

2.3.2. Google Docs

본 툴은 팀 2 의 회의기록 작성 및 요구사항 명세서와 디자인 명세서의 취합에 사용되었다. Google Docs 는 웹 기반의 서비스로 네트워크에 연결되어 있다면 팀원들의 공동 작업을 지원하고 링크의 공유를 통해 문서를 쉽게 공유할 수 있다. 또한 만들어진 파일을 Microsoft word 의 파일 형식인 docx 로의 저장을 지원하는 등, Microsoft word 와도 높은 호환성을 가지고 있다.

2.3.3. Draw.io

본 툴은 플로우 차트를 작성하는 툴로, 본 문서에서 use case diagram, class diagram, sequence diagram 을 작성하는데 사용되었다. Draw.io 는 여러 diagram 을 위한 템플릿을 제공할 뿐 아니라 웹 기반의 서비스로 네트워크에 접속되어 있다면 어디서든 작업이 가능하다. 또한 만들어진 diagram 을 사진 파일로 저장하여 사용할 수 있다.

2.3.4. Figma

본 툴은 애플리케이션의 프로토타입을 만드는 툴로, 요구 사항 명세서에서 user interface 를 디자인하는데 사용되었다. Figma 를 통해 만들어진 프로토타입을 통해 사용자가 어떤 서비스 경험을 할 지 구체적으로 알 수 있다. Figma 또한 웹 기반의 서비스로 네트워크에 접속되어 있다면 어디서든 작업이 가능하고 팀원들의 공동 작업을 지원한다.

2.4 Project Scope

1인 가구의 증가와 더불어 반려동물을 돌보는 가구의 증가로 반려동물을 혼자 집에 둘 수 밖에 없는 사람의 수도 늘어가고 있다. 본 프로젝트는 반려동물을 혼자 집에 두는 상황이 발생하는 사용자를 위해 원격 반려동물 케어를 제공하는 것을 목적으로 하고 있다. 사용자의 IoT 기기에 대한 요청이 IoT 기기에 전달되어야 하므로 이를 위한 서버가 필요하다. 또한 반려동물의 상태와 이상행동 로그, IoT 기기의 설정, 상태를 저장할 데이터 베이스가 필요할 것이다. 그리고 사용자와 IoT 기기와의 인터페이스 역할을 하는 어플리케이션의 개발이 필요하다. 이 외에 지속적인 업데이트를 통해 사용자의 새로운 요구사항이 발생할 경우 팀 2는 지속적으로 해당 요구사항을 반영해가며 사용자가 반려동물 케어라는 목표를 달성할 수 있도록 할 것이다.

2.5 References

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

-Team 12, 2021 Spring, Software Design Document, SKKU

3. System Architecture - Overall

3.1. Objectives

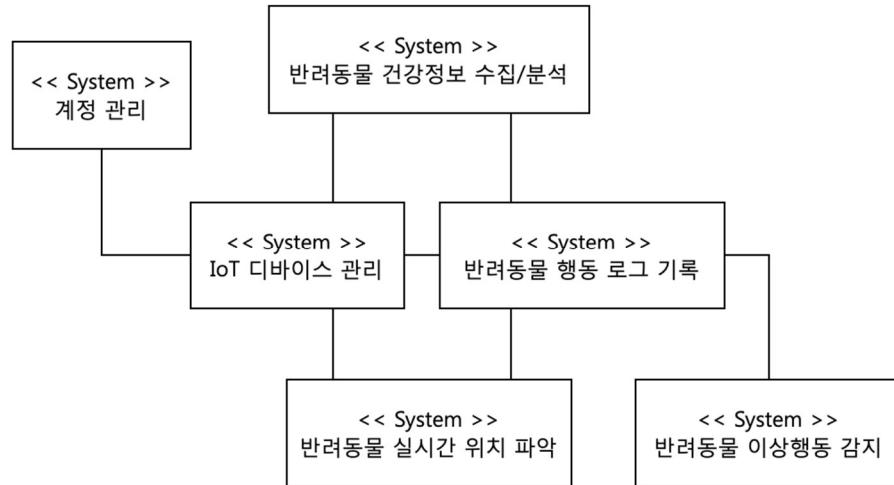
이 챕터에서는 frontend 와 backend 디자인을 포함한 전체적인 시스템의 아키텍처가 묘사되어 있다.

3.2. System Organization

Smart Home Pet Care 는 MVC (Model-View-Controller) 디자인 패턴으로 설명할 수 있다. 시스템의 frontend 는 사용자가 보는 View 부분을 담당한다. View 에는 계정 페이지, 펫 케어 페이지, 모니터링 페이지, 건강 체크 페이지가 있다. 시스템의 backend 에서 사용자의 사용자의 상호작용을 통해 만들어진 데이터 또는 요청은 controller 로 전달되어 처리된다. Frontend 와 backend 간의 통신은 JSON 포맷을 활용한 HTTP 프로토콜로 진행된다. Controller 는 사용자로부터 받은 데이터나 요청을 처리하여 이를 Model 에 저장한다. 그리고 결과는 다시 View 를 통해 사용자에게 전달된다. 예를 들어 사용자가 펫 케어 페이지라는 View 를 통해 펫 케어 IoT 라는 Controller 로 IoT 기기를 끄도록 요청하면, 펫 케어 IoT Controller 는 요청을 처리하여 IoT 기기를 끄고, IoT 기기의 현재 상태를 Model 에 해당하는 데이터베이스에 저장한다. 그리고 IoT 기기가 꺼졌다는 결과는 다시 사용자의 펫 케어 페이지라는 View 로 전달되어 사용자에게 전달되게 된다.

3.2.1. Context Diagram

그림 1. Context Diagram



3.2.2. Sequence Diagram

그림 2. Sequence Diagram - 계정관리

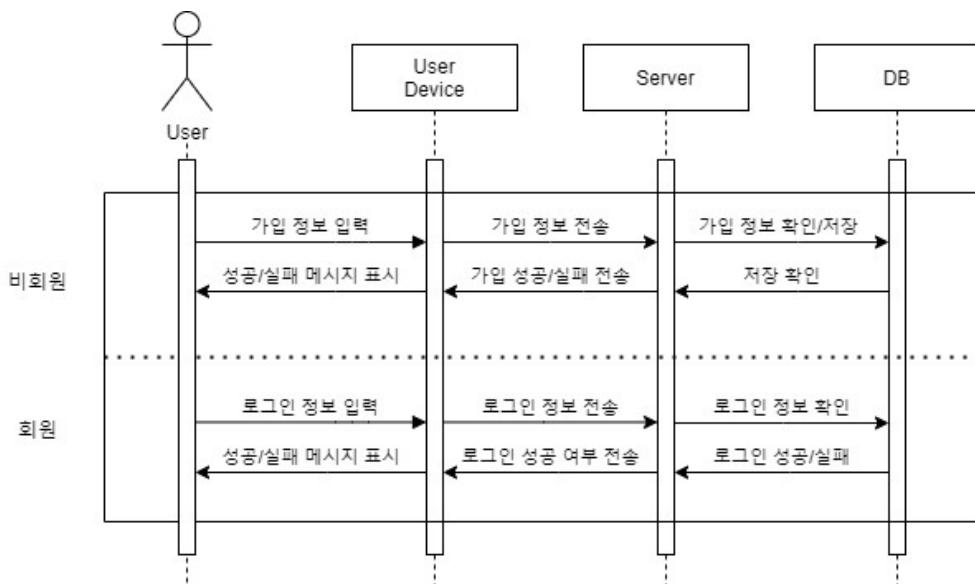


그림 3. Sequence Diagram - 이상행동 Push 알림

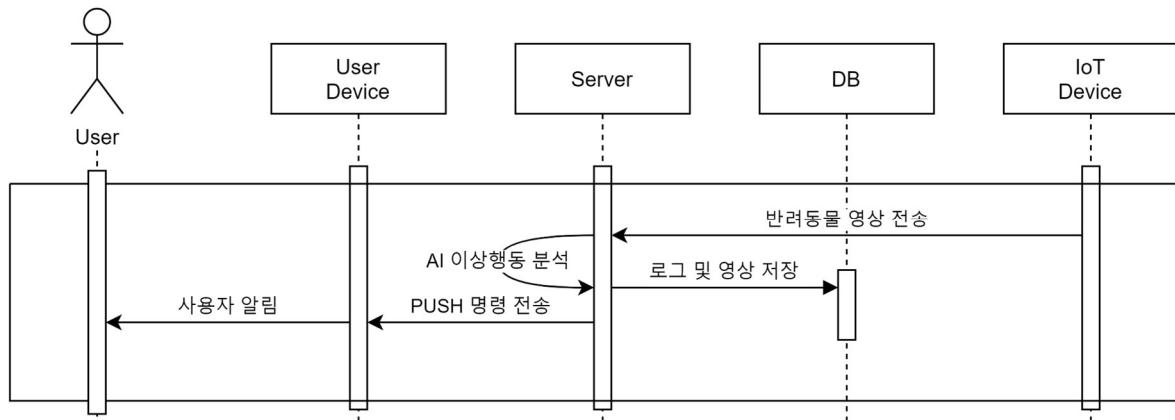


그림 4. Sequence Diagram - 반려동물 위치 정보

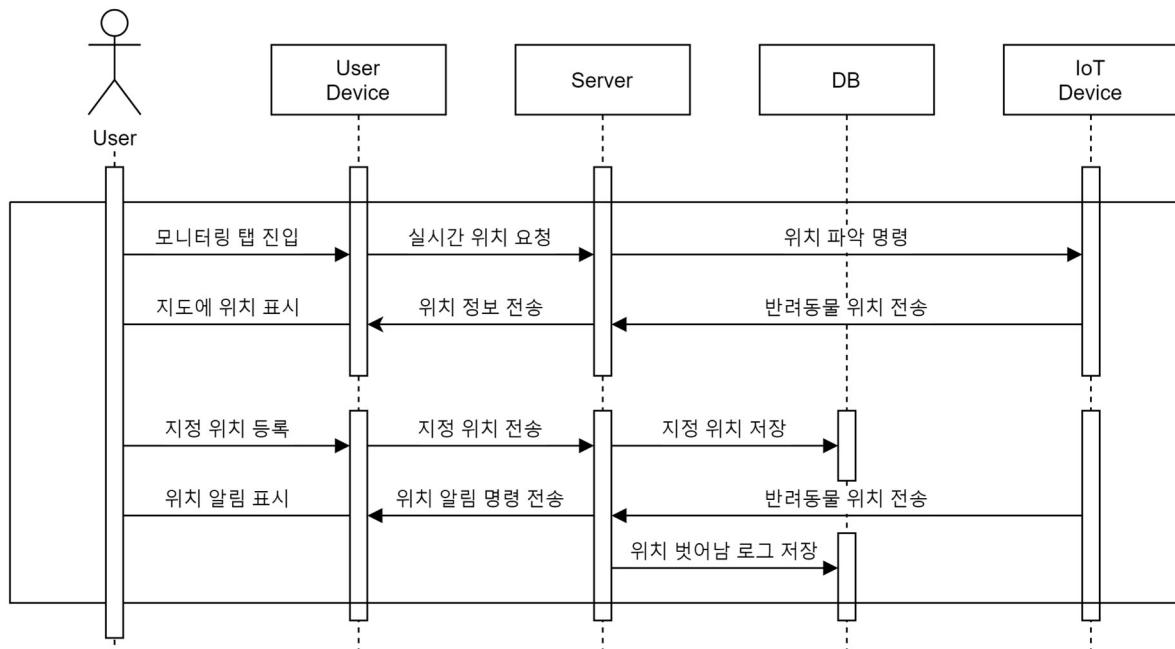


그림 5. Sequence Diagram - 반려동물 로그 조회

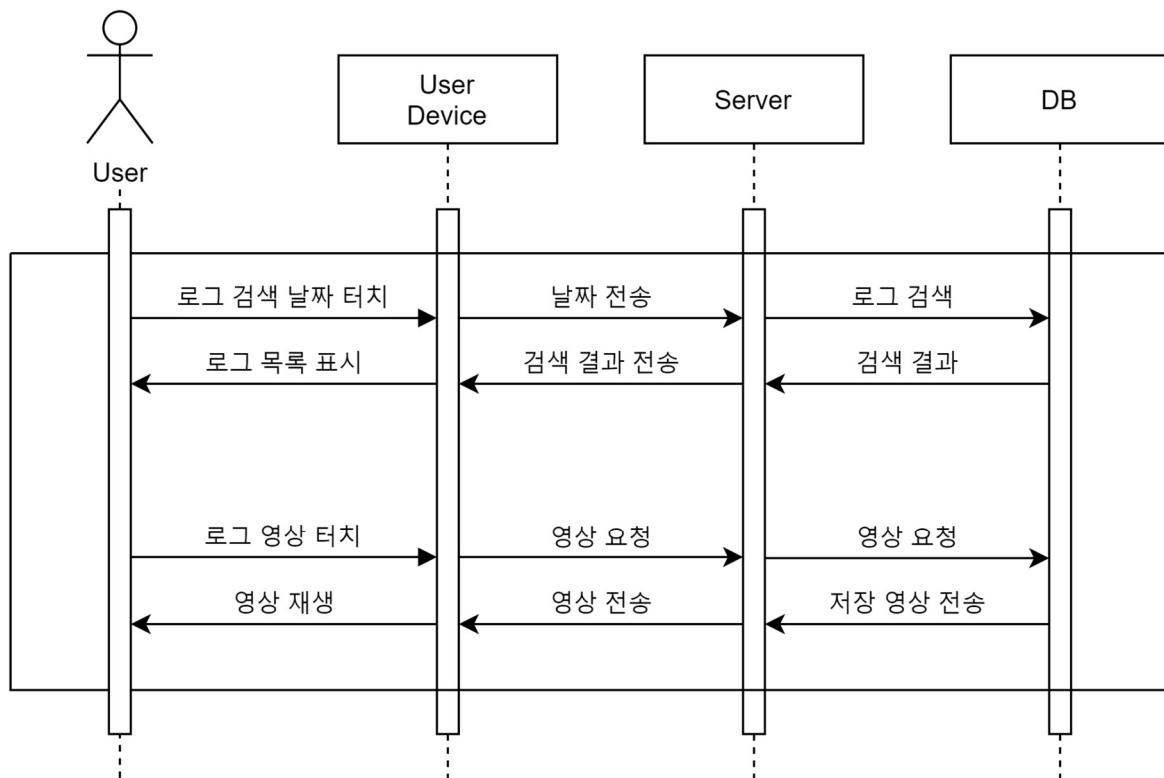


그림 6. Sequence Diagram - IoT 디바이스 관리

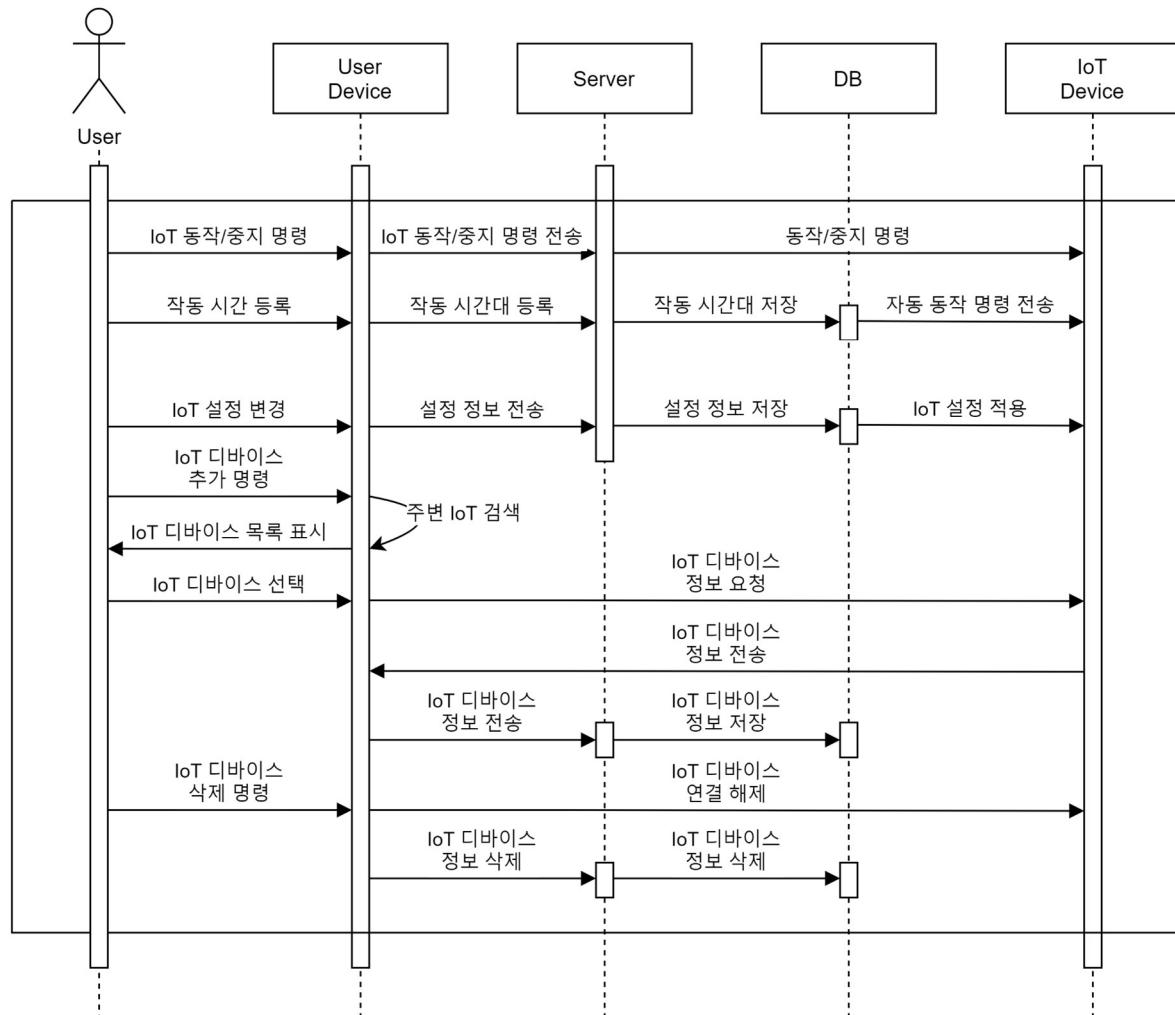
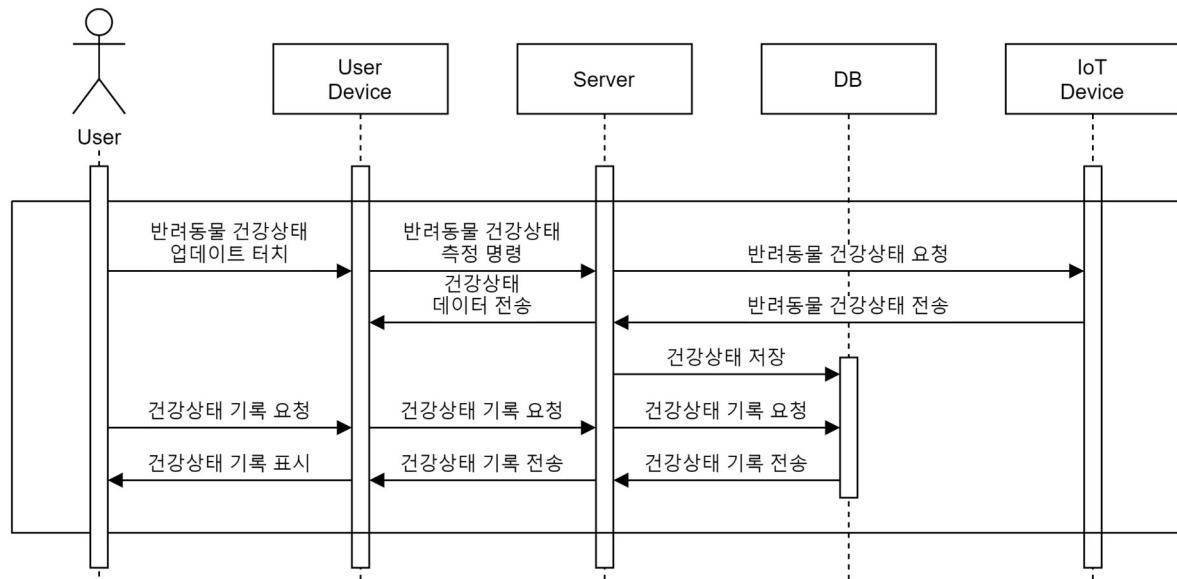
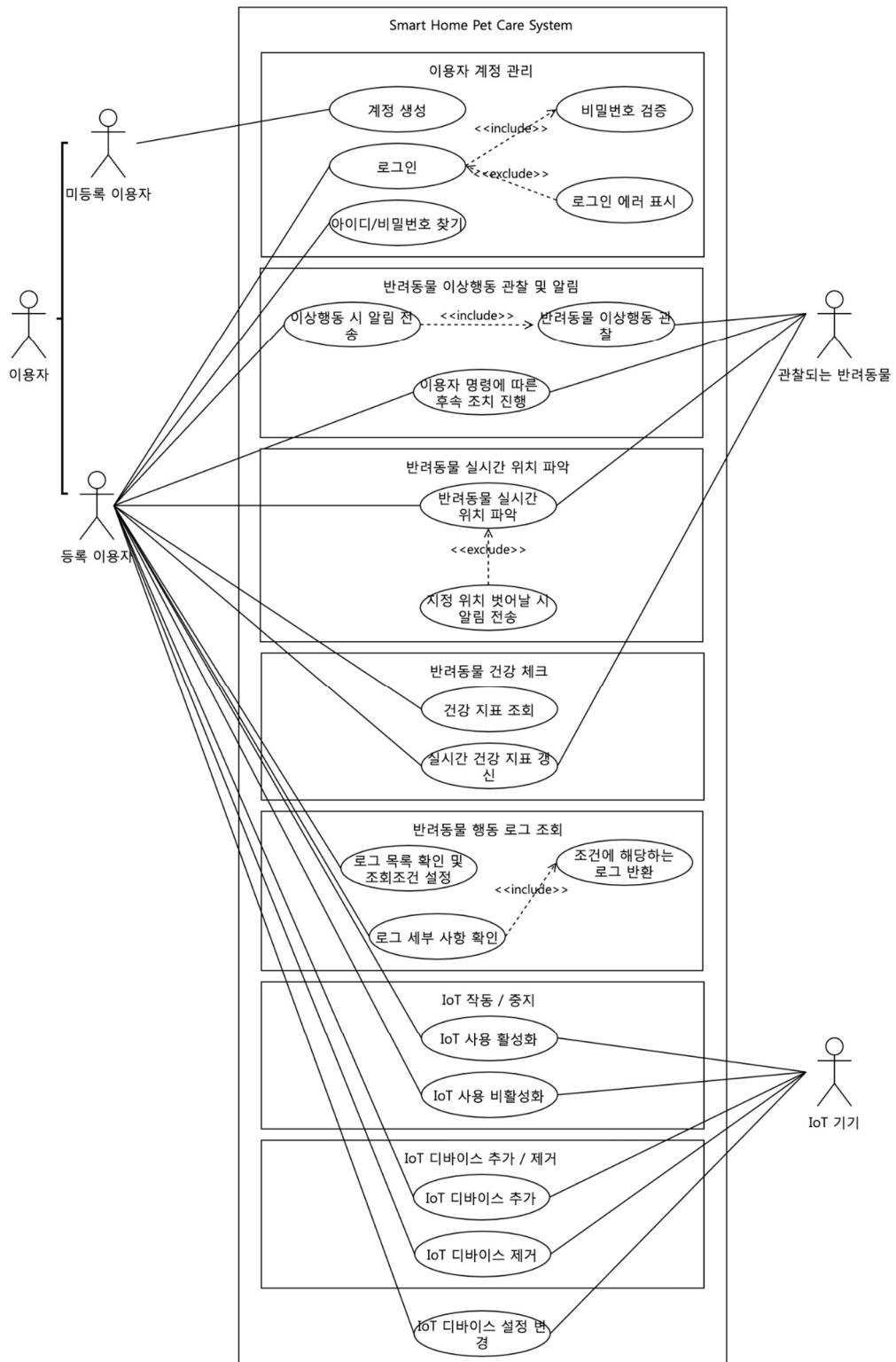


그림 7. Sequence Diagram - 반려동물 건강상태 관리



3.2.3. Use Case Diagram

그림 8. Use Case Diagram



4. System Architecture - Frontend

4.1. Objectives

해당 부분은 System 의 Front-end 부분의 구조에 대해 기술한다. 해당 부분은 각각의 Subcomponents 와 관련된 Attributes, Methods 를 포함한다.

4.2. Subcomponents

4.2.1. 계정 관리

계정 관리는 어플리케이션 실행 직후 등록된 ID/PW 를 통해 로그인하거나, 신규 이용자의 경우 회원가입을 진행할 수 있으며, 등록된 계정이 있으나 분실한 경우 찾을 수 있는 기능들이 구현되어 있는 class 이다.

4.2.1.1. Attributes

다음은 계정 관리 Class 의 Attributes 이다.

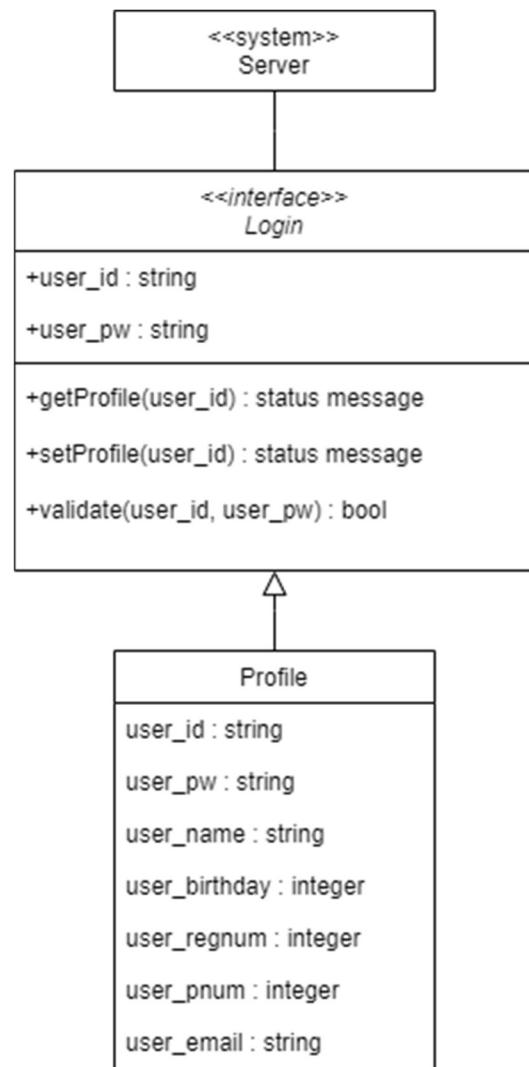
- **user_id** : 이메일 형식으로 된 사용자의 id 이다.
- **user_pw** : 사용자의 비밀번호이다.
- **user_name** : 회원가입시 기입해야하는 본인의 실명이다.
- **user_birthday** : 사용자의 생년월일이다.
- **user_regnnum** : 사용자의 주민등록번호이다.
- **user_pnum** : 사용자의 개인 휴대전화 번호이다.
- **user_email** : ID/PW 분실시 본인인증을 위한 이메일 주소이다.

4.2.1.2. Methods

- **getProfile()** : 로그인 시도시 서버와 통신하는 method 이다.
- **setProfile()** : 신규 사용자의 회원등록시 사용될 method 이다.
- **findingAccount()** : ID/PW 분실시 사용될 method 이다.

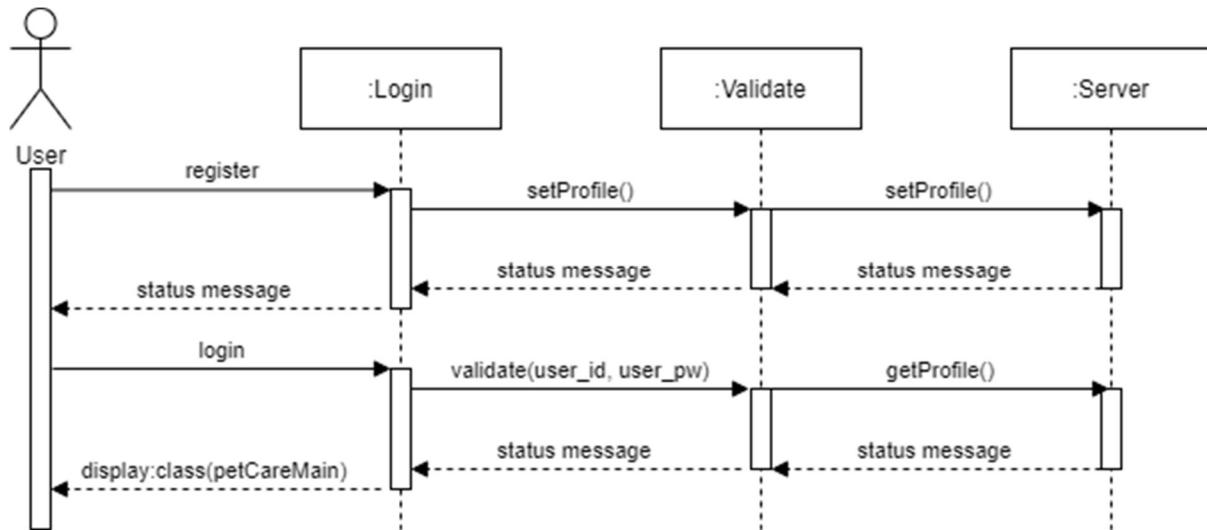
4.2.1.3. Class Diagram

그림 9. Class Diagram - 계정 관리 Frontend



4.2.1.4. Sequence Diagram

그림 10. Sequence Diagram - 계정 관리 Frontend



4.2.2. IoT 디바이스 등록 및 설정

IoT 디바이스 등록 및 설정은 펫 케어 시스템에서 사용할 IoT 디바이스들을 관리하고, 사용할 작업들을 설정하는 기능들을 수행하는 Class 이다.

Front-End 부에서는 주로 사용자가 설정을 조작할 수 있도록 사용자 디바이스에 관련된 내용을 표시하는 것을 목표로 한다.

4.2.2.1. Attributes

IoT 디바이스 등록 및 설정 Class 의 Attributes 는 다음과 같다.

- **devices_available** : 현재 연결 가능한 IoT 디바이스들의 목록을 저장하는 리스트이다. 해당 리스트는 연결된 디바이스를 포함해, 현재 동일 네트워크에 존재하는 모든 연결 가능한 디바이스들을 포함한다.
- **devices_connected** : 현재 펫케어 시스템과 연결된 IoT 디바이스들에 대한 리스트이다.

다음은 **devices_available** 객체의 Attributes 이다.

- **devices_id** : IoT 디바이스의 고유 번호를 나타낸다.
- **user_id**: IoT 디바이스에 등록된 사용자를 표기한다.
- **devices_type** : IoT 디바이스의 종류를 나타낸다.
- **devices_name** : IoT 디바이스의 사용자 설정 이름을 나타낸다.

다음은 **devices_connected** 객체의 Attributes 이다.

- **devices_id** : IoT 디바이스의 고유 번호를 나타낸다.
- **user_id**: IoT 디바이스에 등록된 사용자를 표기한다.
- **devices_type** : IoT 디바이스의 종류를 나타낸다.
- **devices_name** : IoT 디바이스의 사용자 설정 이름을 나타낸다.
- **devices_functions** : IoT 디바이스에서 사용 가능한 기능들을에 대한 리스트이다.

다음은 **devices_functions** 객체의 Attributes 이다. 해당 Attributes 는 상위 Attributes 의 **devices_id** 정보에 의해 사용가능한 기능이 규정된다.

- **function_moving** : 움직임이 가능한 IoT 디바이스의 경우 움직임을 수행할지의 여부를 선택한다.
- **function_camera** : 카메라가 부착된 IoT 디바이스의 경우 카메라를 사용할지의 여부를 선택한다.
- **function_feed** : 사료 지급 기능이 내장된 IoT 디바이스의 경우 이를 사용할지의 여부를 선택한다.
- **function_play** : 놀이 기능이 내장된 IoT 디바이스의 경우 이를 사용할지의 여부를 선택한다.
- **function_location** : GPS 가 탑재된 IoT 디바이스의 경우 위치 정보를 사용할지의 여부를 선택한다.
- **function_healthcare** : 건강 측정 센서가 탑재된 IoT 디바이스의 경우 이를 사용할지의 여부를 선택한다.

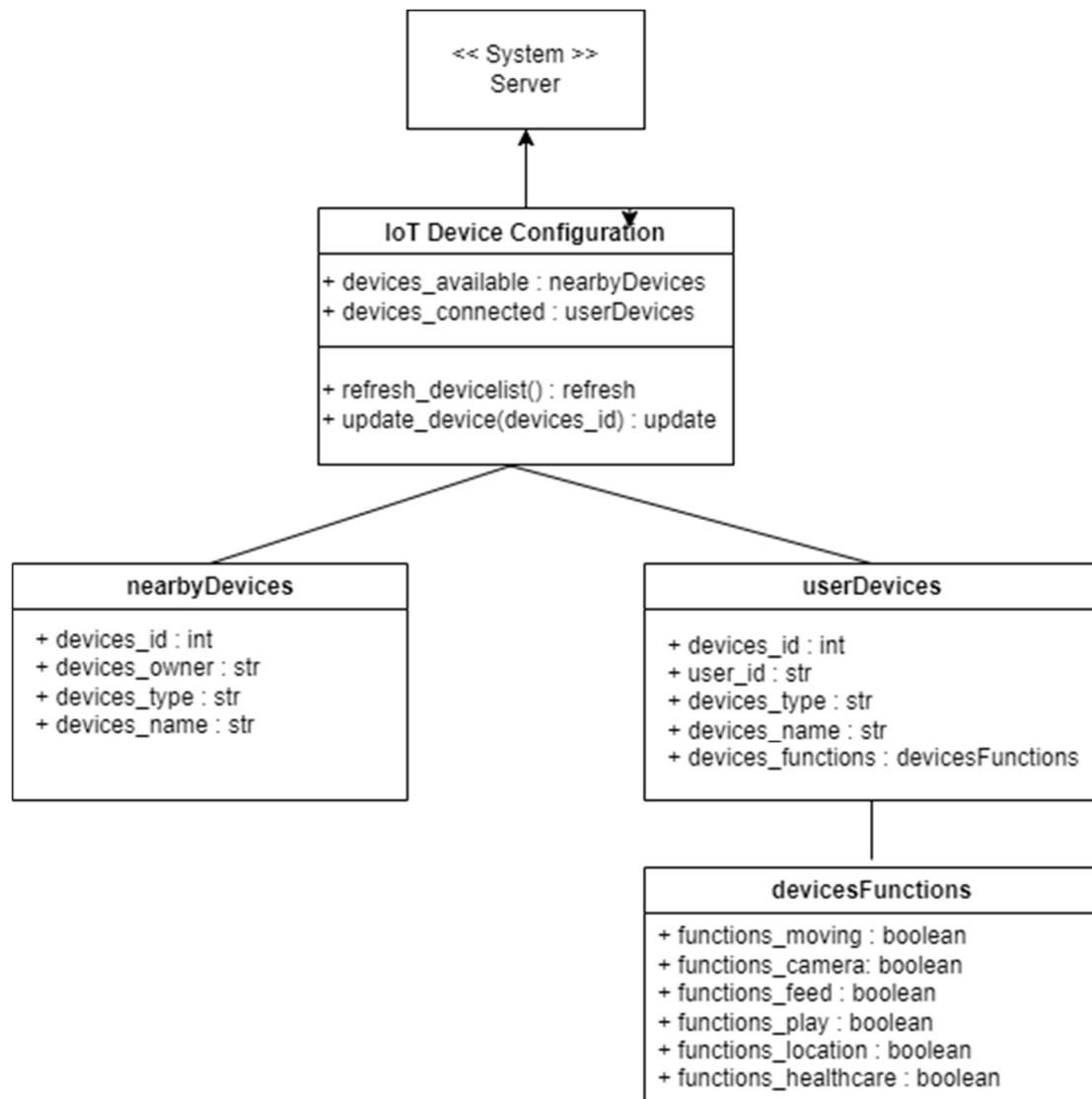
4.2.2.2. Methods

IoT 디바이스 등록 및 설정 Class 에서 사용되는 Methods 는 다음과 같다.

- **refresh_devicelist()** : 동일 네트워크에 연결된 다른 IoT 디바이스의 탐색을 수행하기 위해 디바이스 목록을 갱신하도록 요청한다.
- **update_device(devices_id)** : 사용자가 IoT 디바이스의 설정을 변경하면 (새로운 기기 연결, 기존 기기 연결 해제, 기능 설정/해제) 이를 반영하도록 요청한다.

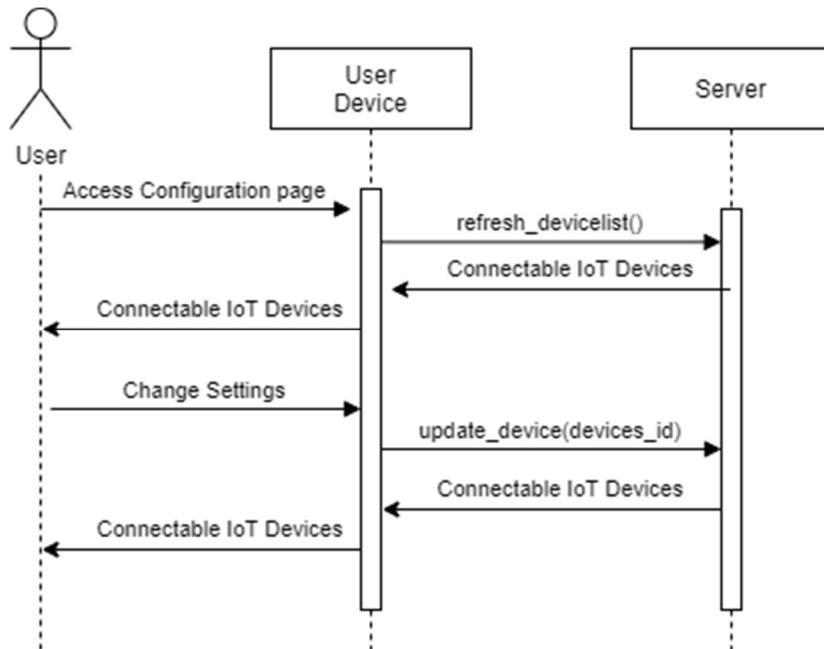
4.2.2.3. Class Diagram

그림 11. Class Diagram - IoT 디바이스 등록 및 설정 FrontEnd



4.2.2.4. Sequence Diagram

그림 12. Sequence Diagram - IoT 디바이스 등록 및 설정 FrontEnd



4.2.3. 반려동물 이상 행동 앱 푸시 알림

반려동물 이상행동 앱푸시 알림은 실시간으로 기록되는 반려동물 행동로그가 이상행동에 속한다고 판단되면 앱푸시 알림의 형태로 사용자에게 알리는 데에 사용되는 Class 이다.

4.2.3.1. Attributes

반려동물 이상행동 앱 푸시 알림 Class 의 Attributes 는 다음과 같다.

- **abnormal_behavior** : 해당 반려동물 행동 로그가 이상행동으로 분류되는지 여부를 나타낸다.
- **behavior** : 반려동물 행동 로그이다.
- **push_noti** : 앱푸시 알림을 생성하는 데에 필요한 정보를 담은 객체이다.

다음은 **push_noti** 객체의 Attributes 이다.

- **id** : 객체 식별자
- **time** : 서버에서의 알림 전송 시각
- **push_msg** : 앱푸시 알림 메세지 내용

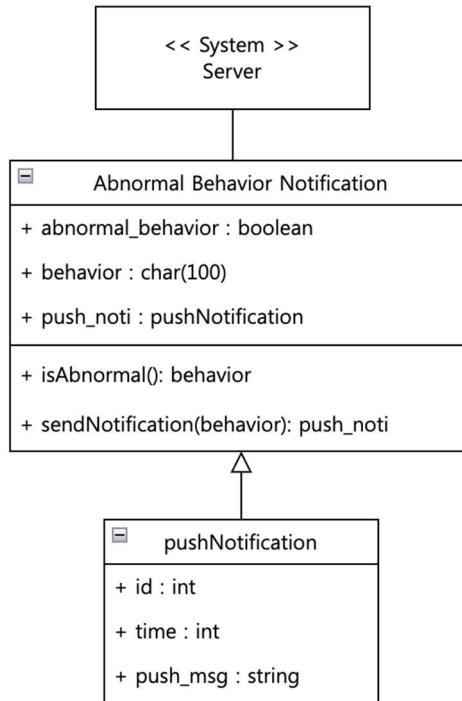
4.2.3.2. Methods

반려동물 이상행동 앱 푸시 알림 Class 에서 사용되는 Methods 는 다음과 같다.

- **isAbnormal()** : 해당 반려동물 행동 로그가 이상행동으로 분류되었는지 판단한다.
- **sendNotification(behavior)** : 이상행동으로 판단된 행동 로그 내용을 앱푸시 알림으로 사용자에게 전송한다.

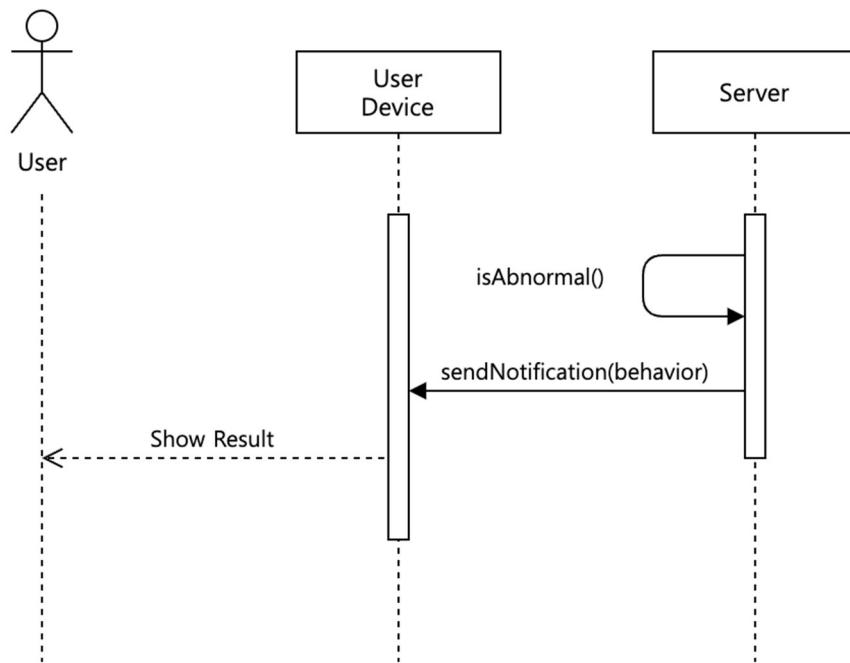
4.2.3.3. Class Diagram

그림 13. Class Diagram - 반려동물 이상행동 앱푸시 알림 Frontend



4.2.3.4. Sequence Diagram

그림 14. Sequence Diagram - 반려동물 이상행동 앱푸시 알림 Frontend



4.2.4. 반려동물 실시간 위치 파악

4.2.4.1. Attributes

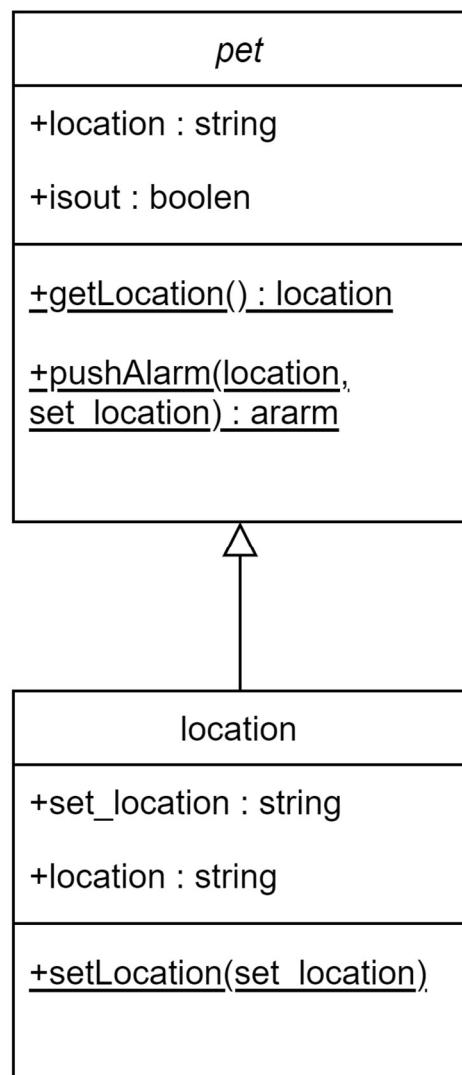
- **location** : 현재 반려동물의 위치이다
- **set_location** : 사용자가 등록해둔 반려동물의 허용 위치 범위이다
- **isOut** : 반려동물이 허용 위치 범위를 벗어났는지 확인하는 값이다.

4.2.4.2. Methods

- **getLocation()**:반려동물의 현재 위치를 얻는다
- **setLocation(set_location)**: 반려 동물의 허용 위치를 세팅한다
- **pushAlarm(location, set_location)**: 반려 동물의 현재 위치와 세팅 위치를 이용해 위치를 벗어나면 알람을 생성한다.

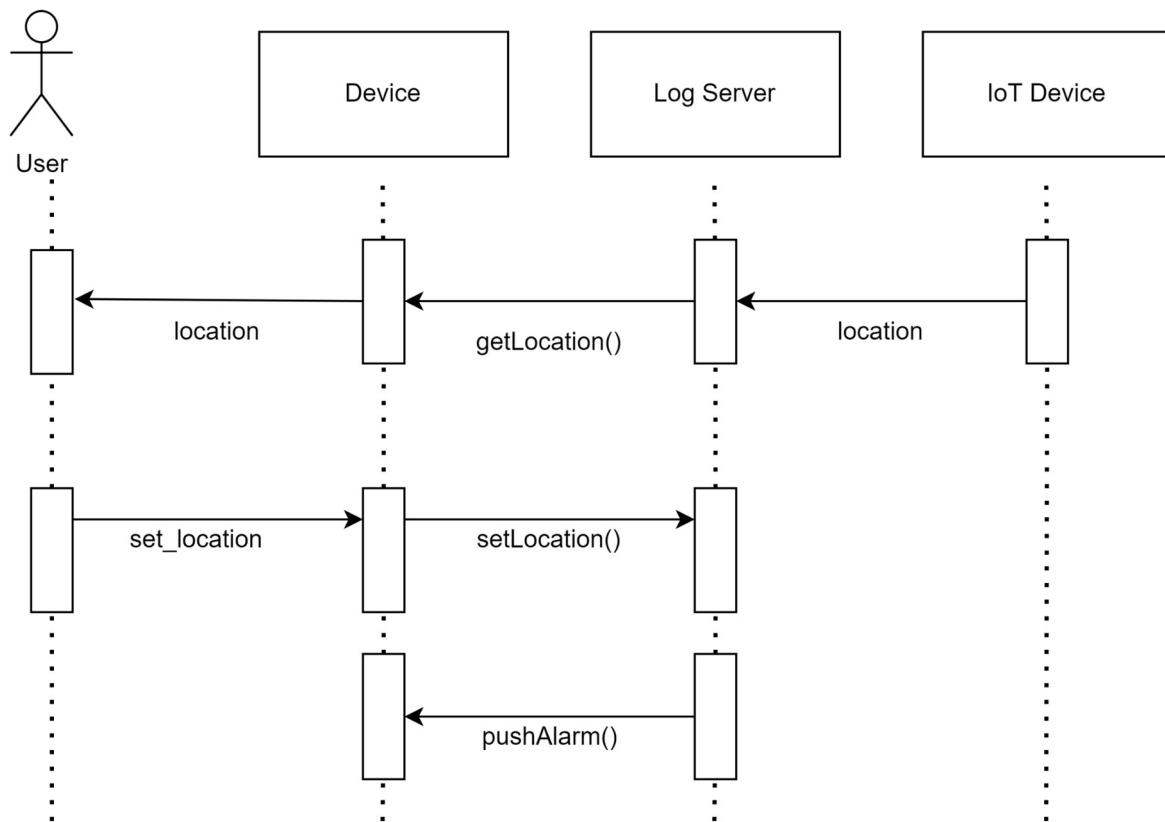
4.2.4.3. Class Diagram

그림 15. Class Diagram - 반려동물 실시간 위치 파악 FrontEnd



4.2.4.4. Sequence Diagram

그림 16 Sequence Diagram - 반려동물 실시간 위치 파악 FrontEnd



4.2.5. 반려동물 행동 로그

4.2.5.1. Attributes

반려동물 행동 로그 Object 의 Attributes 는 다음과 같다.

- **log_id**: 로그의 고유 id 이다.
- **user_id**: 로그 소유자의 id 이다.
- **pet_id**: 로그 발생 반려동물 id 이다.
- **device_id**: 로그가 발생한 IoT 디바이스의 id 이다.
- **timestamp**: 로그 발생시간의 타임스탬프이다.
- **message**: 로그의 세부 정보이다.

4.2.5.2. Methods

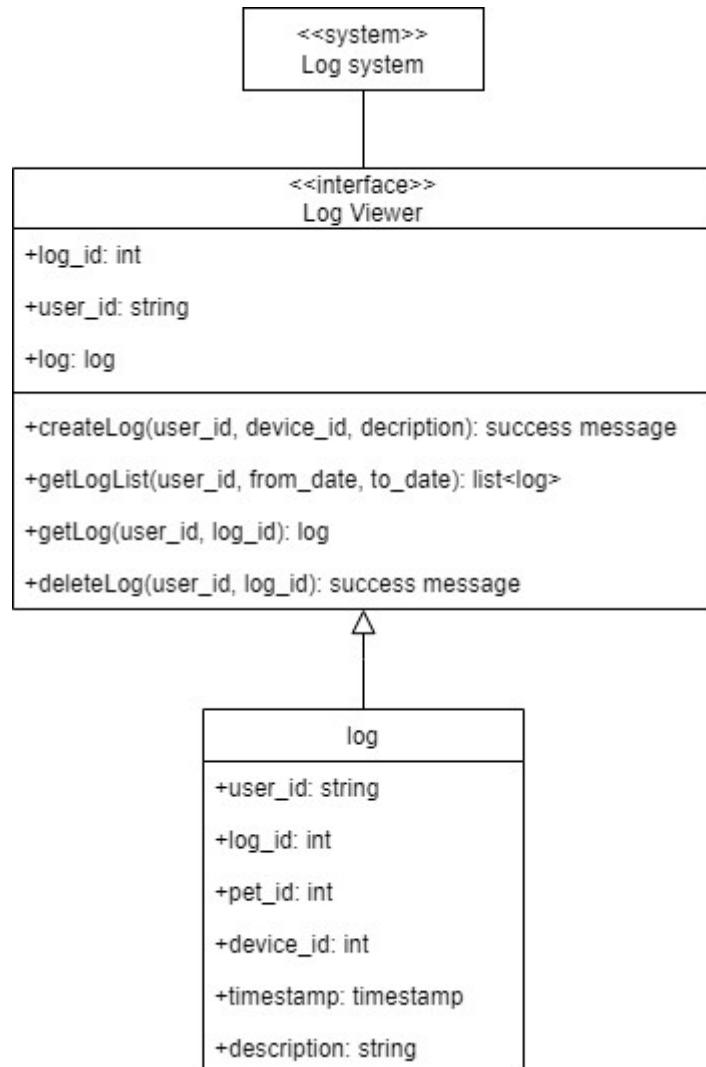
반려동물 행동 로그 Object 의 Method 는 다음과 같다.

- **createLog(user_id, device_id, description)**: 로그를 생성하는 메소드이다.

- **getLog(user_id, log_id)**: 로그 조회 메소드이다.
- **getLogList(user_id, from_date, to_date)**: 기간별 로그조회 메소드이다.
- **deleteLog(user_id, log_id)**: 로그 삭제 메소드이다.

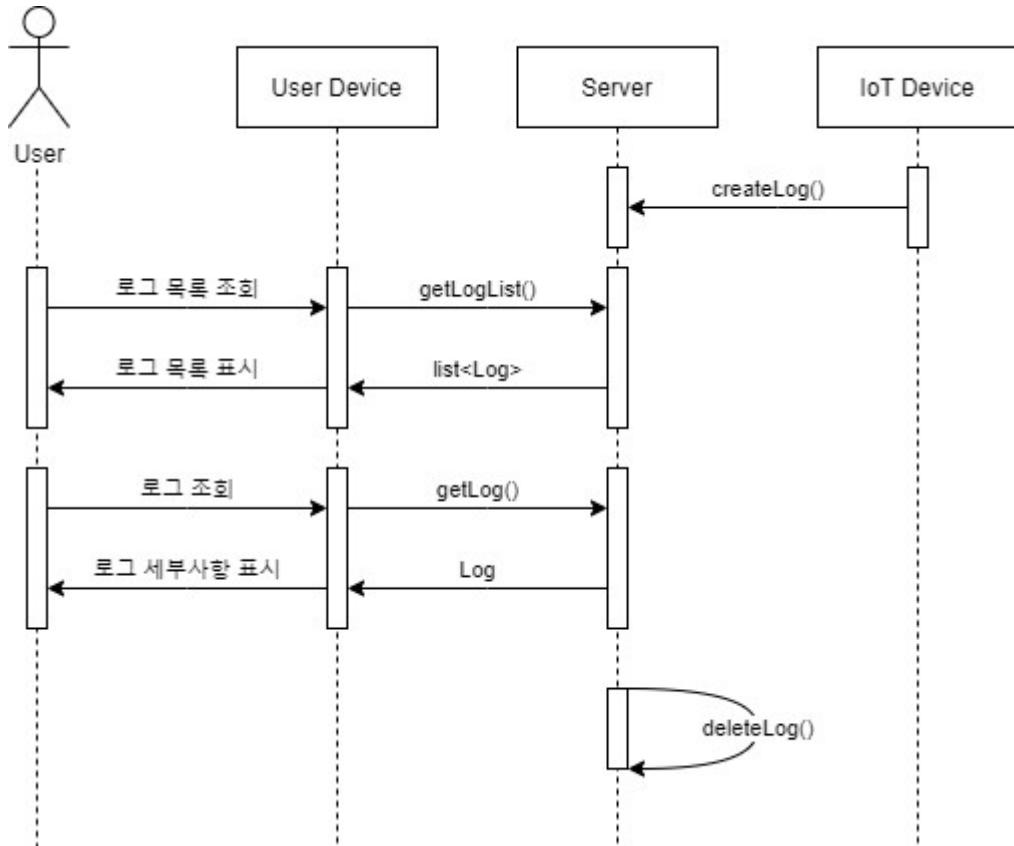
4.2.5.3. Class Diagram

그림 17. Class Diagram - 반려동물 행동 로그 Front-End



4.2.5.4. Sequence Diagram

그림 18. Sequence Diagram - 반려동물 행동 로그 Frontend



4.2.6 반려동물 건강상태 획득/분석

반려동물 건강상태 획득 및 분석은 반려동물의 건강 프로필 페이지를 통해 지정된 반려동물의 건강 지표를 실시간으로 확인하는데 사용되는 Class 이다.

4.2.6.1. Attributes

다음은 반려동물 건강상태 획득/분석 Class 의 Attributes 이다

- **user_id** : 사용자의 아이디를 나타내며, 여기서 사용자는 서비스를 이용하는 반려동물의 양육자를 의미한다.
- **pet_id** : 반려동물의 아이디를 나타내며, 사용자가 보유하고 있는 반려동물을 구분하는 고유값을 의미 한다.
- **health_profile** : 반려동물의 건강지표들의 정보를 표시하는 객체이다.

다음은 **health_profile** 객체의 Attributes 이다.

- **pulse** : 반려동물의 맥박수 값과 상태를 나타내는 객체이다.
- **oxygen** : 반려동물의 산소포화도 값과 상태를 나타내는 객체이다.

- **temperature** : 반려동물의 체온 값과 상태를 나타내는 객체이다.
- **stress** : 반려동물의 스트레스 지수값과 상태를 나타내는 객체이다.

다음은 **건강지표** 객체들의 Attributes 이다.

- **value** : 반려동물로부터 가장 최근에 측정된 맥박수/산소포화도/체온/스트레스 값을 나타낸다.
- **status** : 측정한 맥박수/산소포화도/체온/스트레스 값을 통해 진단하는 반려동물의 건강 상태 값을 나타낸다.
- **updated** : 반려동물로부터 언제 맥박수/산소포화도/체온/스트레스가 가장 최근에 측정되었는지를 나타내는 날짜, 시간값이다.

4.2.6.2. Methods

다음은 반려동물 건강상태 획득/분석 Class 의 Method 들 이다.

- **getHealthProfile()** - 반려동물의 건강정보 프로필 값을 조회하는 메소드
- **updateHealthProfile()** - 반려동물의 건강정보 프로필 값을 업데이트 하는 메소드

다음은 **health_profile** 객체의 메소드 들이다.

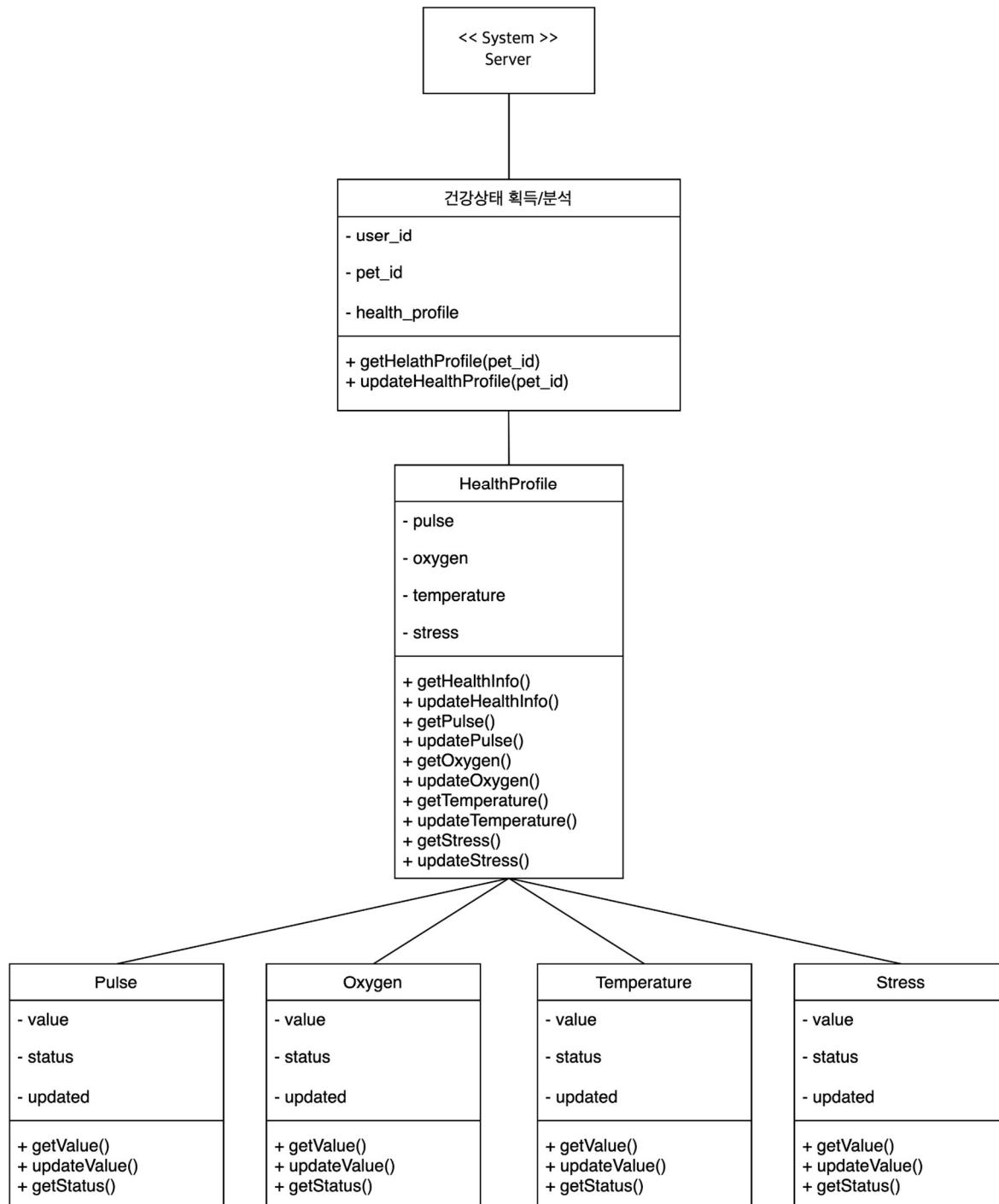
- **getHealthInfo()** - 모든 건강 정보들을 조회하는 메소드
- **updateHelathInfo()** - 모든 건강 정보들을 업데이트 하는 메소드
- **getPulse()** - 맥박수 정보를 조회하는 메소드
- **updatePulse()** - 맥박수 정보를 업데이트 하는 메소드
- **getOxygen()** - 산소포화도 정보를 조회하는 메소드
- **updateOxygen()** - 산소포화도 정보를 업데이트 하는 메소드
- **getTemperature()** - 체온 정보를 조회하는 메소드
- **updateTemperature()** - 체온 정보를 업데이트 하는 메소드
- **getStress()** - 스트레스 지수값을 조회하는 메소드
- **updateStress()** - 스트레스 지수값을 업데이트 하는 메소드

다음은 건강 지표 객체들의 관련 메소드 들이다.

- **getValue()** - 해당 건강 지표의 값을 조회하는 메소드
- **updateValue()** - 해당 건강 지표의 값을 갱신하는 메소드
- **getStatus()** - 해당 건강 지표의 진단 상태를 조회하는 메소드

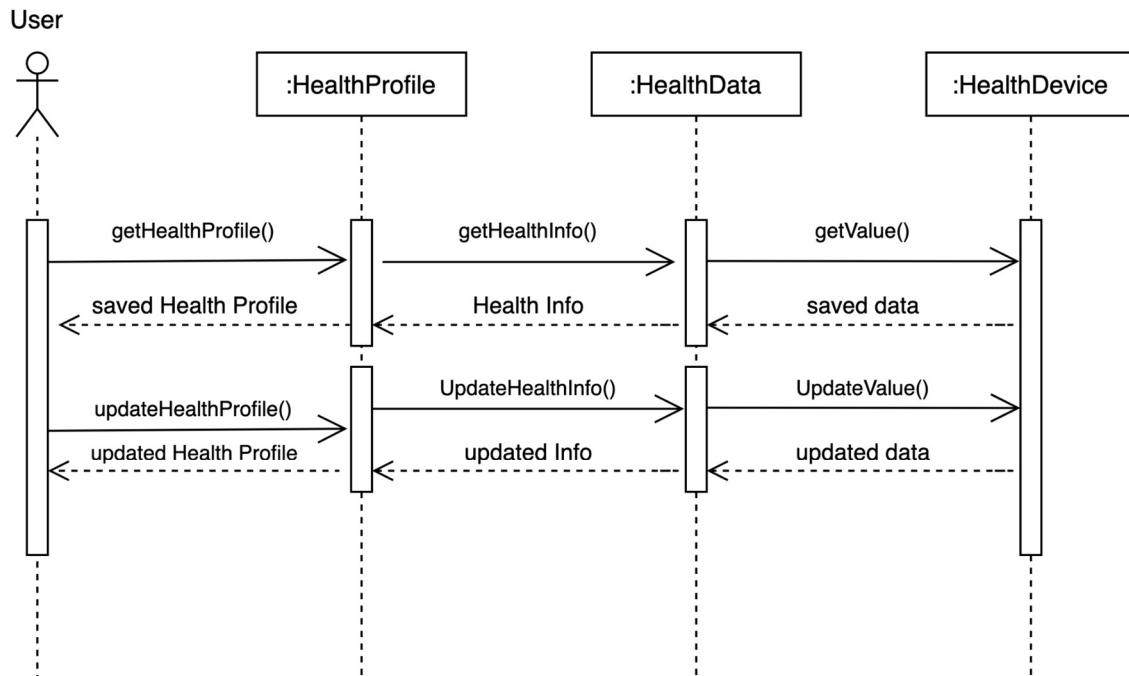
4.2.6.3. Class Diagram

그림 19. Class Diagram - 반려동물 건강상태 획득/분석 FrontEnd



4.2.6.4. Sequence Diagram

그림 20. Sequence Diagram - 반려동물 건강상태 획득/분석 FrontEnd



4.2.7. 반려동물 맞춤 케어 시스템 - IoT 디바이스 상태 확인

4.2.7.1. Attributes

해당 Object 가 가진 attribute 는 다음과 같다.

- **IoT_Devices:** 사용자가 등록한 IoT 디바이스들의 리스트이다.
- **IoT_Devices_Timelines:** 사용자가 등록한 IoT 디바이스들의 작동 시간대 리스트이다.
- **IoT_Devices_Workings:** 사용자가 등록한 IoT 디바이스들의 작동 여부를 저장한 리스트이다.

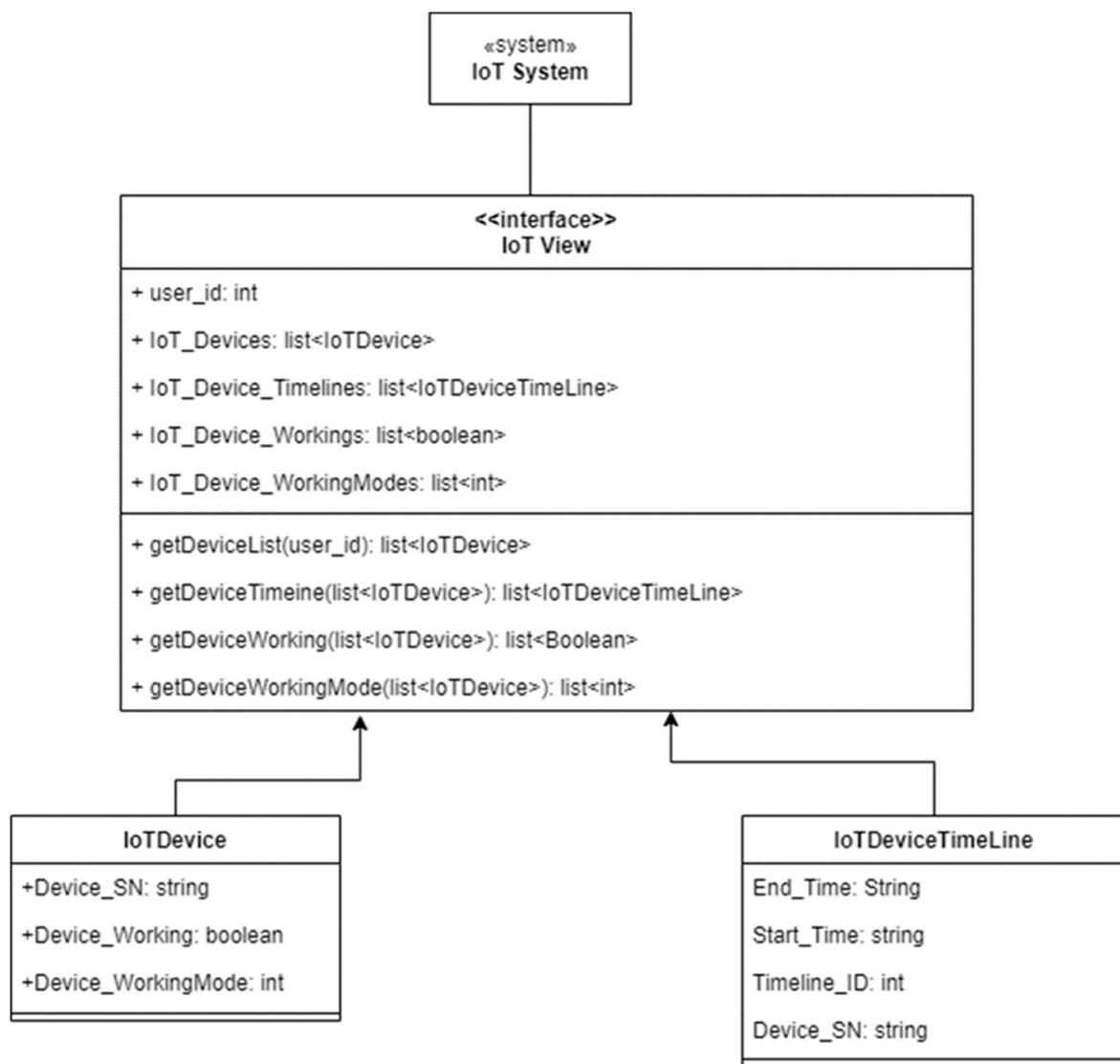
4.2.7.2. Methods

해당 Object 가 가진 method 는 다음과 같다.

- **getDeviceList()**
- **getDeviceTimeline()**
- **getDeviceWorking()**
- **getDeviceWorkingMode()**

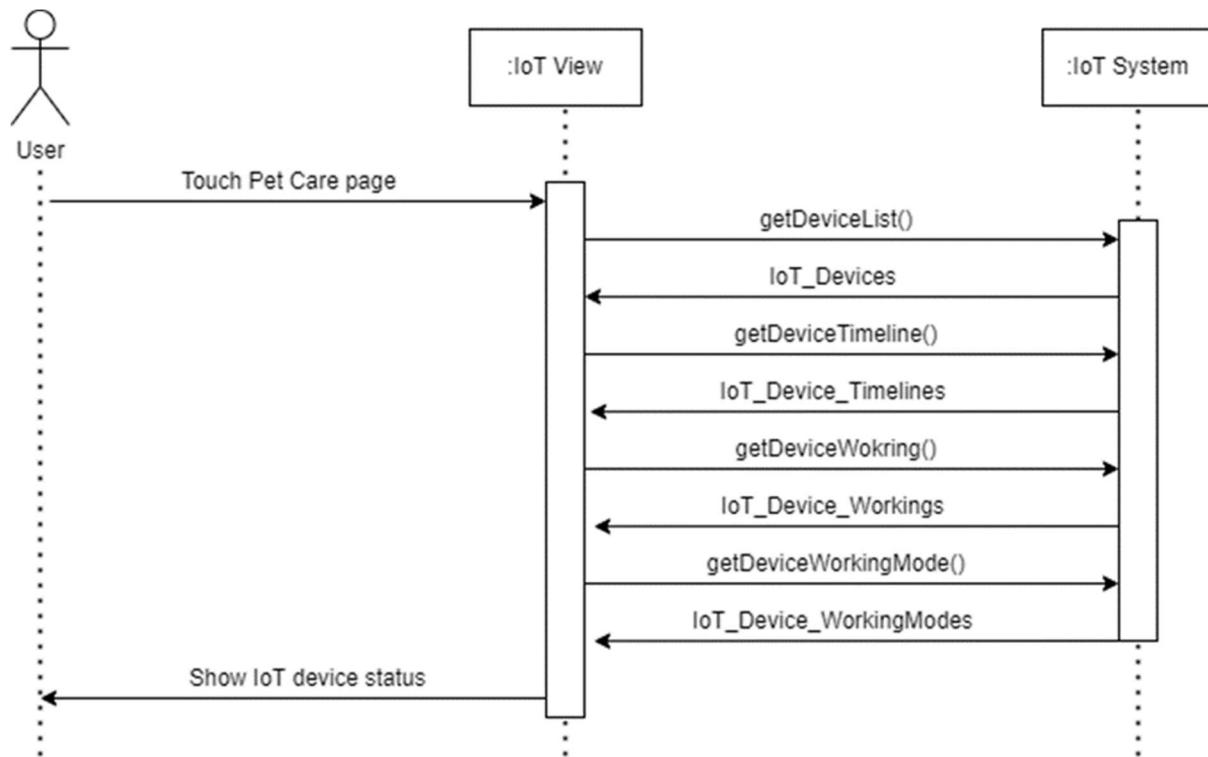
4.2.7.3. Class Diagram

그림 21. Class Diagram - 반려동물 맞춤 케어 시스템 - IoT 디바이스 상태 확인 Frontend



4.2.7.4. Sequence Diagram

그림 22. Sequence Diagram - 반려동물 맞춤 케어 시스템 - IoT 디바이스 상태 확인 Frontend



4.2.7. 반려동물 맞춤 케어 시스템 설정

4.2.7.1. Attributes

해당 Object 가 가진 attribute 는 다음과 같다.

- **IoT_Device**: 설정을 바꾸고자 하는 IoT 디바이스의 시리얼 번호
- **IoT_Device_Timelines**: IoT 디바이스에 등록된 작동 시간대 리스트
- **IoT_Device_Working**: IoT 디바이스의 작동 여부
- **IoT_Device_WorkingMode**: IoT 디바이스의 작동 모드(오토 케어모드, 지정 시간대 작동 모드)

4.2.7.2. Methods

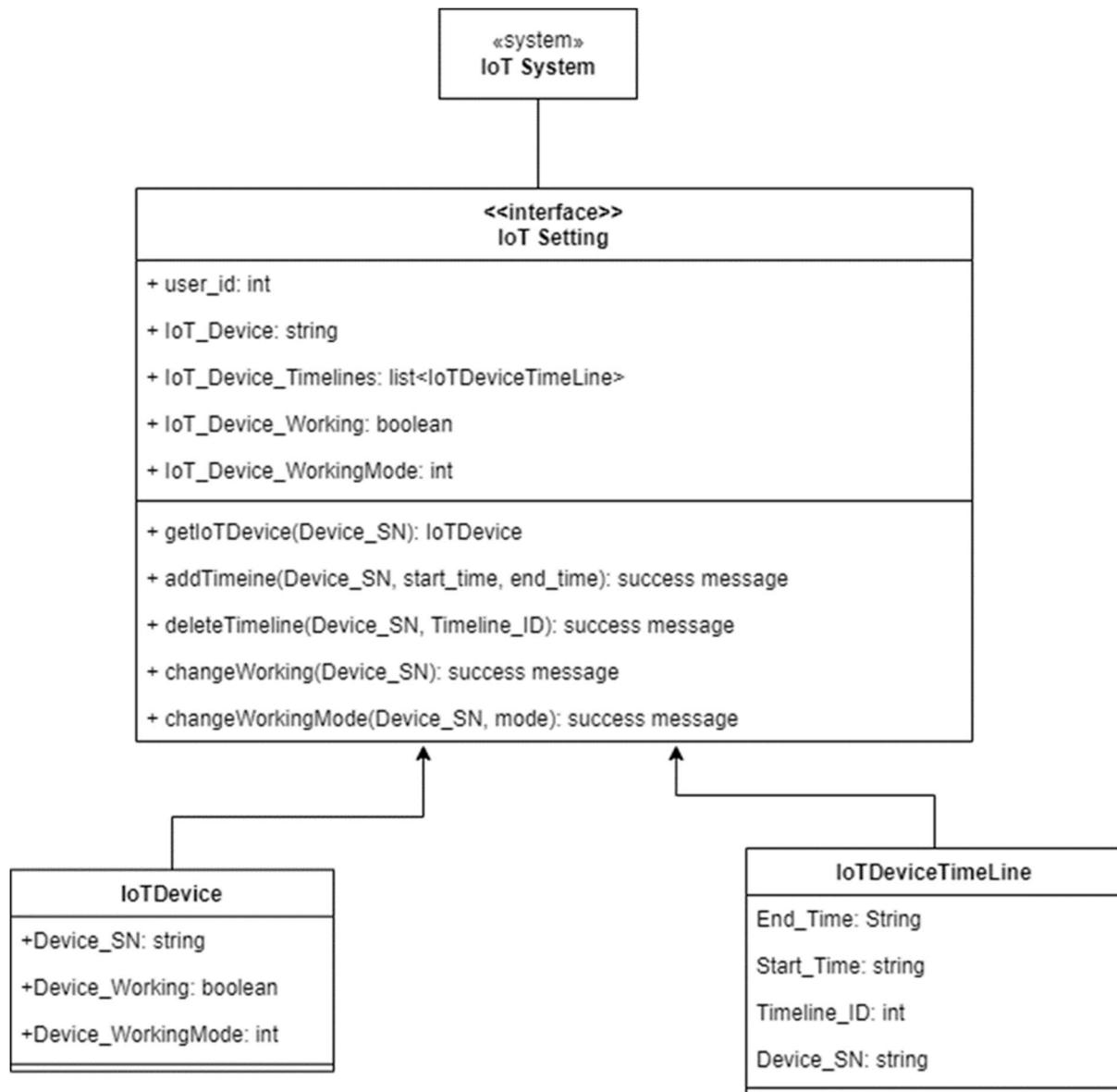
해당 Object 가 가진 method 는 다음과 같다.

- **addTimeline()**
- **deleteTimeline()**
- **changeWorking()**

● **changeWorkingMode()**

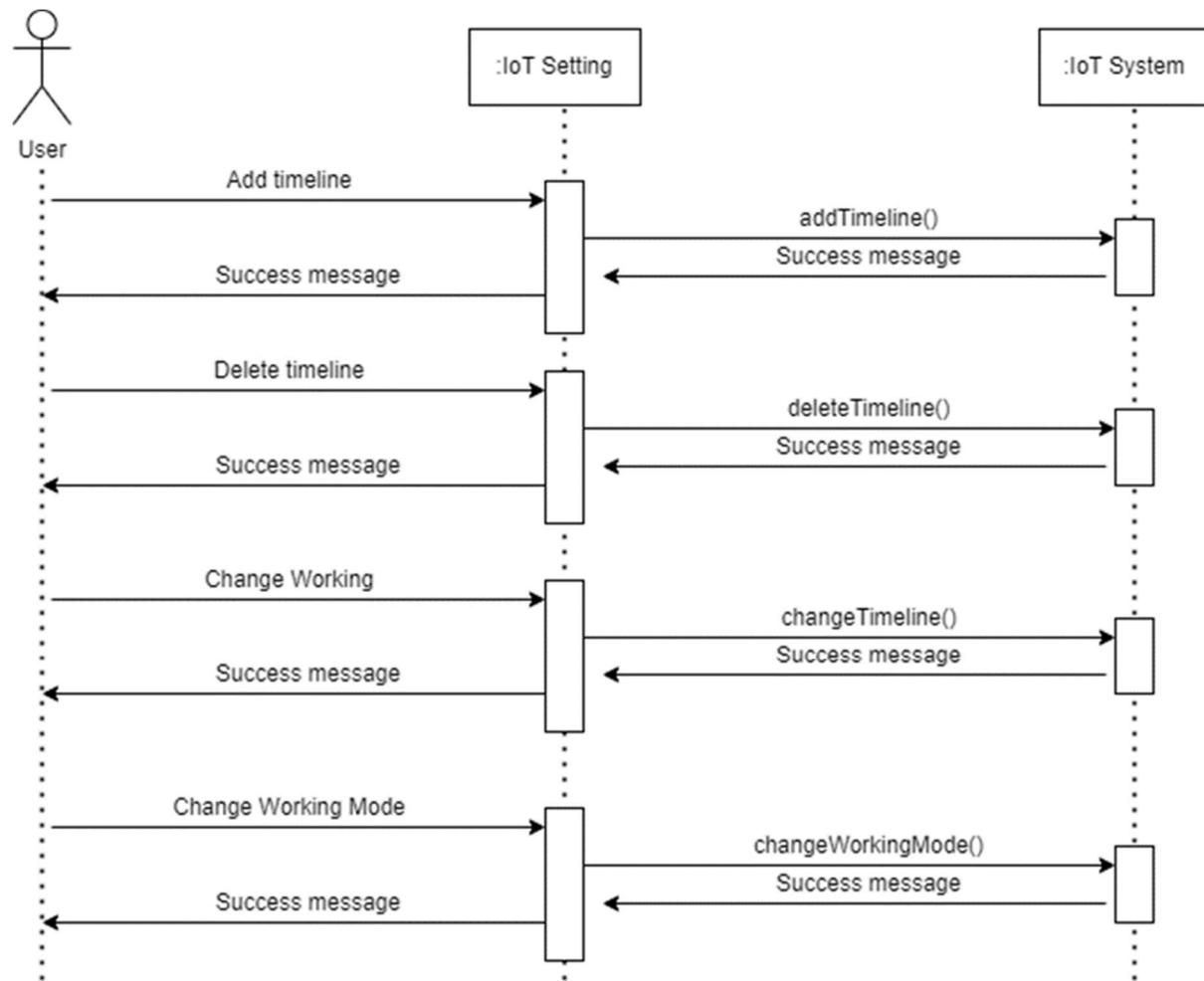
4.2.7.3. Class Diagram

그림 23. Class Diagram - 반려동물 맞춤 케어 시스템 설정 Frontend



4.2.7.4. Sequence Diagram

그림 24. Sequence Diagram - 반려동물 맞춤 케어 시스템 설정 Frontend



5. System Architecture - Backend

5.1. Objectives

해당 부분은 위에서 기술된 Front-End 부에 이어서, 시스템의 Back-End 부에 대해서 기술한다. 이 부분에서는 주로 Front-end 작업을 수행하기 위해, Back-end에서 수행해야 하는 작업을 위주로 기술했다.

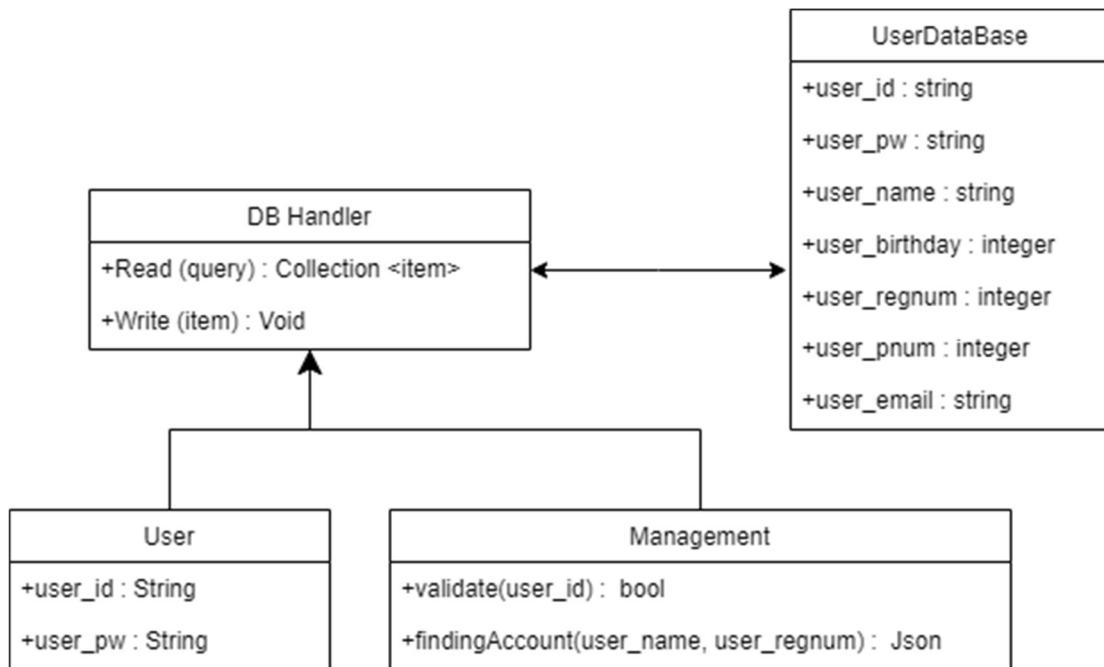
5.2. Subcomponents

5.2.1. 계정 관리

로그인 또는 회원가입시 BackEnd 부에서 유저 정보들이 등록되어있는 DataBase 로 유저 정보를 보내면, DataBase에서 정보를 대조하여 status message 혹은 해당 정보를 FrontEnd 부로 전달한다.

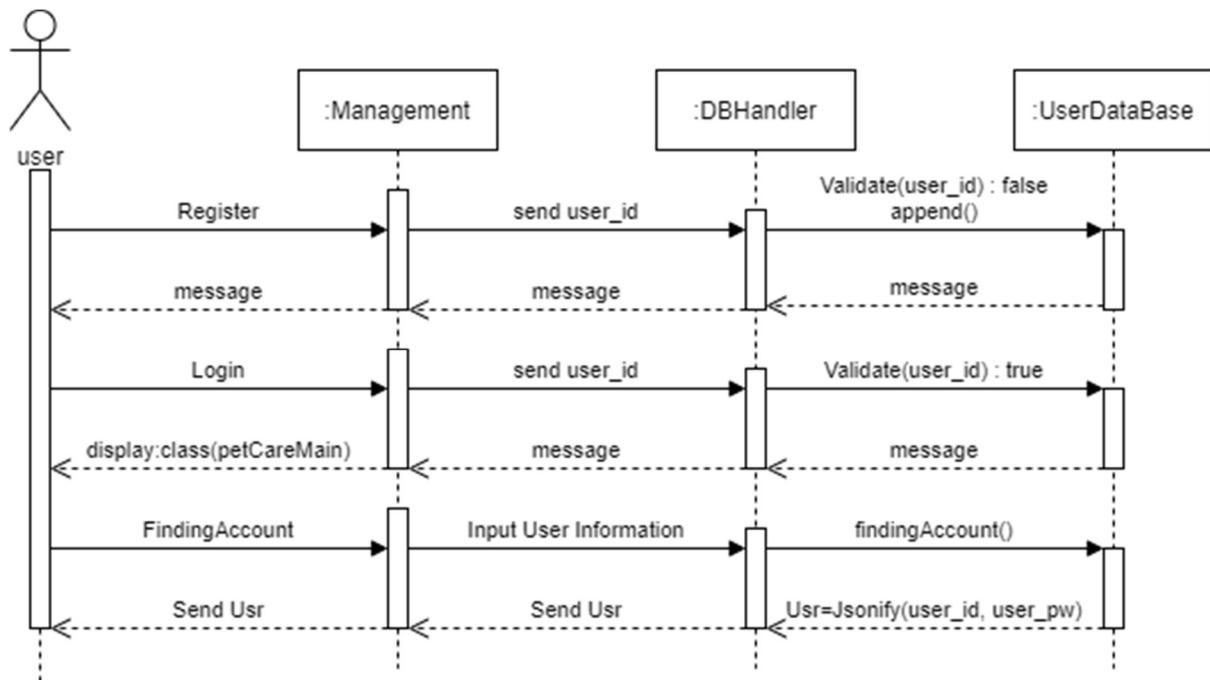
5.2.1.1. Class Diagram

그림 25. Class Diagram - 계정 관리 Backend



5.2.1.2. Sequence Diagram

그림 26. Sequence Diagram - 계정 관리 Backend



5.2.2. IoT 디바이스 등록 및 설정

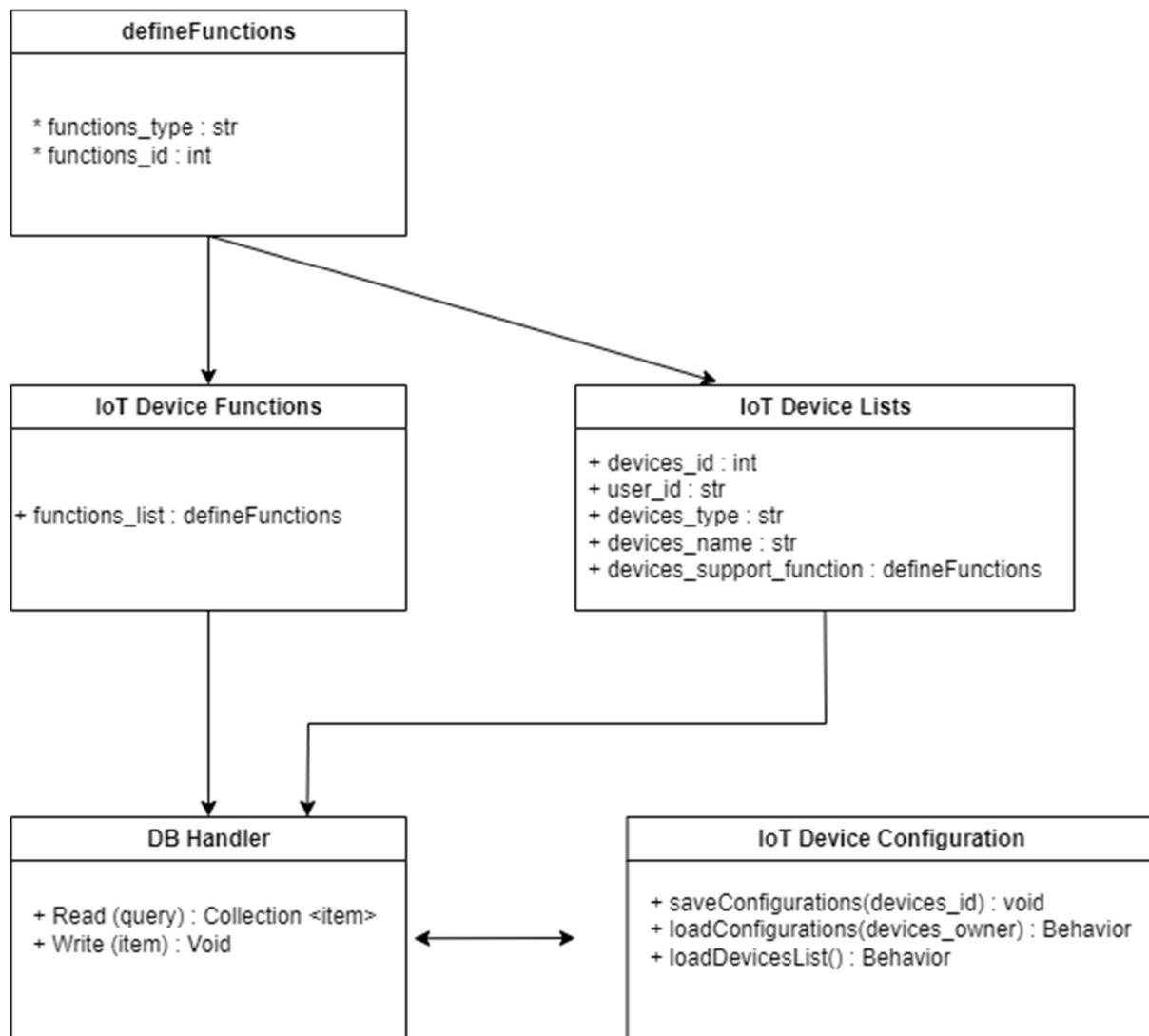
반려동물 이상행동 앱푸시 알림은 실시간으로 기록되는 반려동물 행동로그가 이상행동에 속한다고 판단되면 앱푸시 알림의 형태로 사용자에게 알리는 데에 사용되는 Class 이다.

FrontEnd 부에서는 사용자에게 현재 연결 가능한 디바이스의 목록을 보여주고, 설정 가능한 디바이스나 기능들을 보여주었다면,

BackEnd 부에서는 현재 설정된 정보를 FrontEnd 로 넘겨주거나, FrontEnd에서 갱신된 값을 DB에 저장하는 작업을 수행하게 된다.

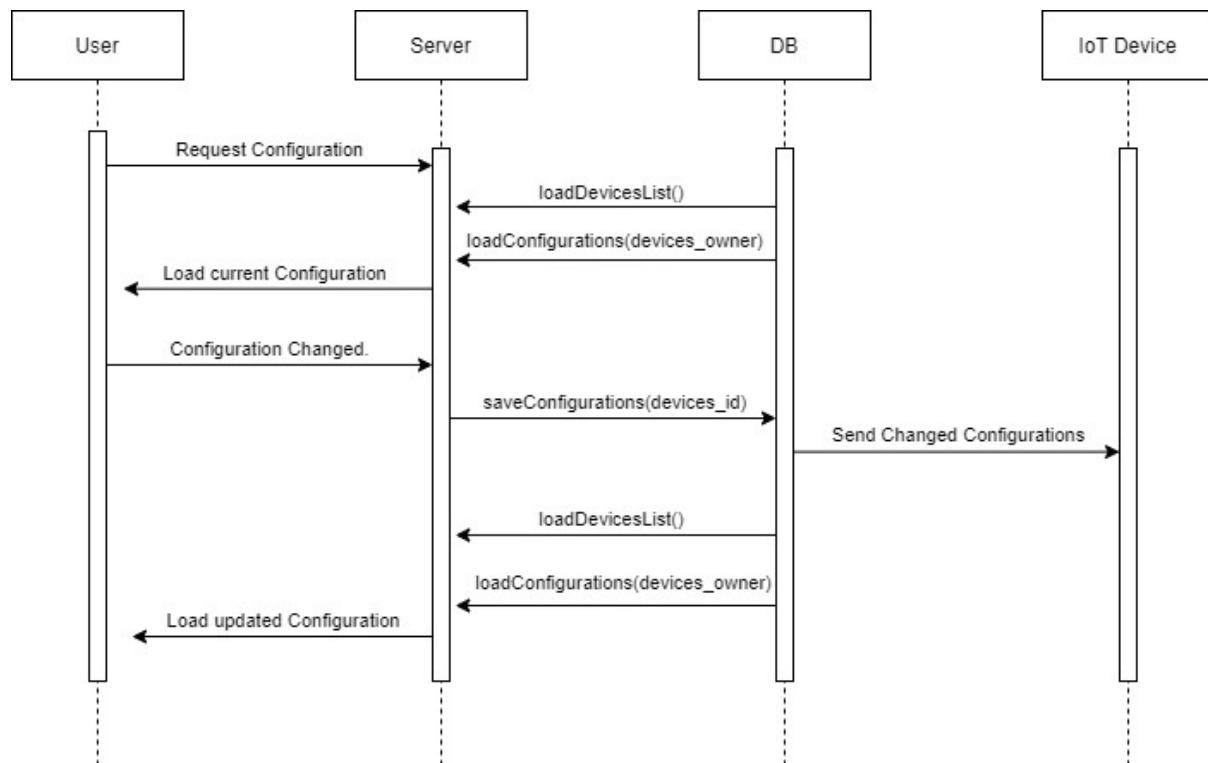
5.2.2.1. Class Diagram

그림 27. Class Diagram - IoT 디바이스 등록 및 설정 BackEnd



5.2.2.2. Sequence Diagram

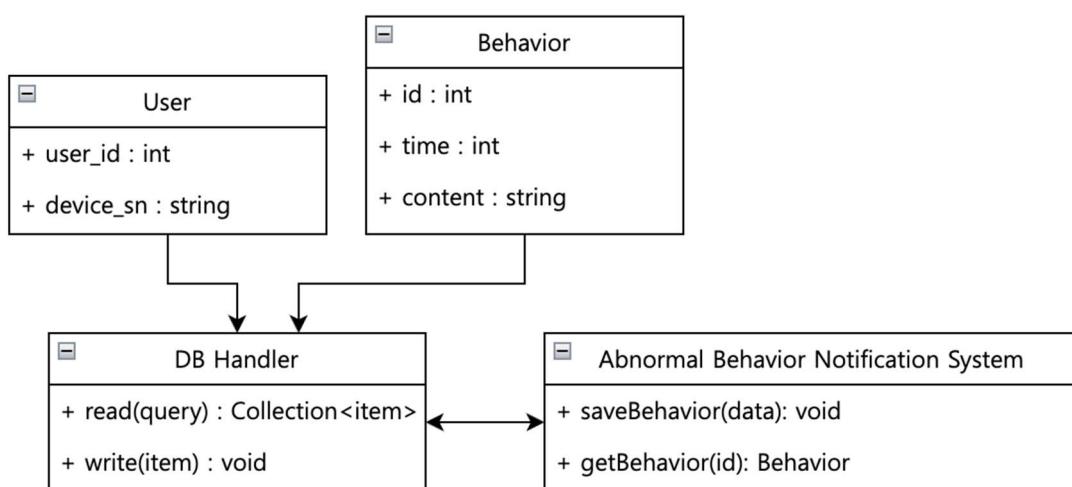
그림 28. Sequence Diagram - IoT 디바이스 등록 및 설정 BackEnd



5.2.3. 반려동물 이상 행동 앱 푸시 알림

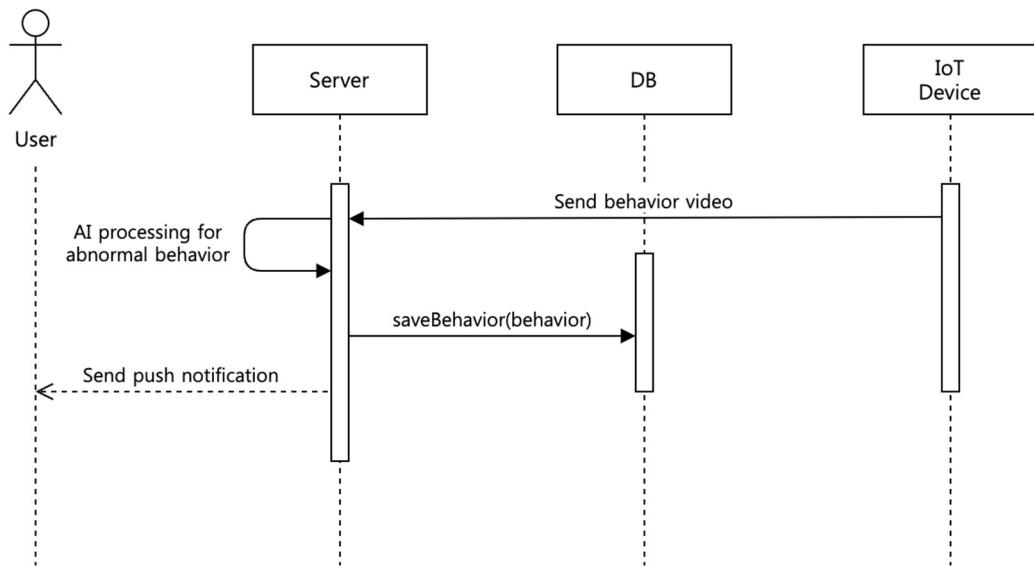
5.2.3.1. Class Diagram

그림 29. Class Diagram - 반려동물 이상행동 앱푸시 알림 Backend



5.2.3.2. Sequence Diagram

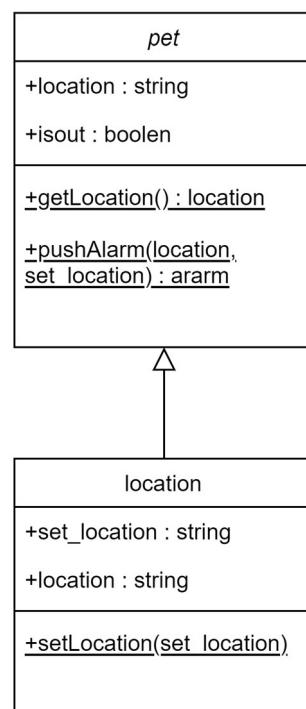
그림 30. Sequence Diagram - 반려동물 이상행동 앱푸시 알림 Backend



5.2.4. 반려동물 실시간 위치 파악

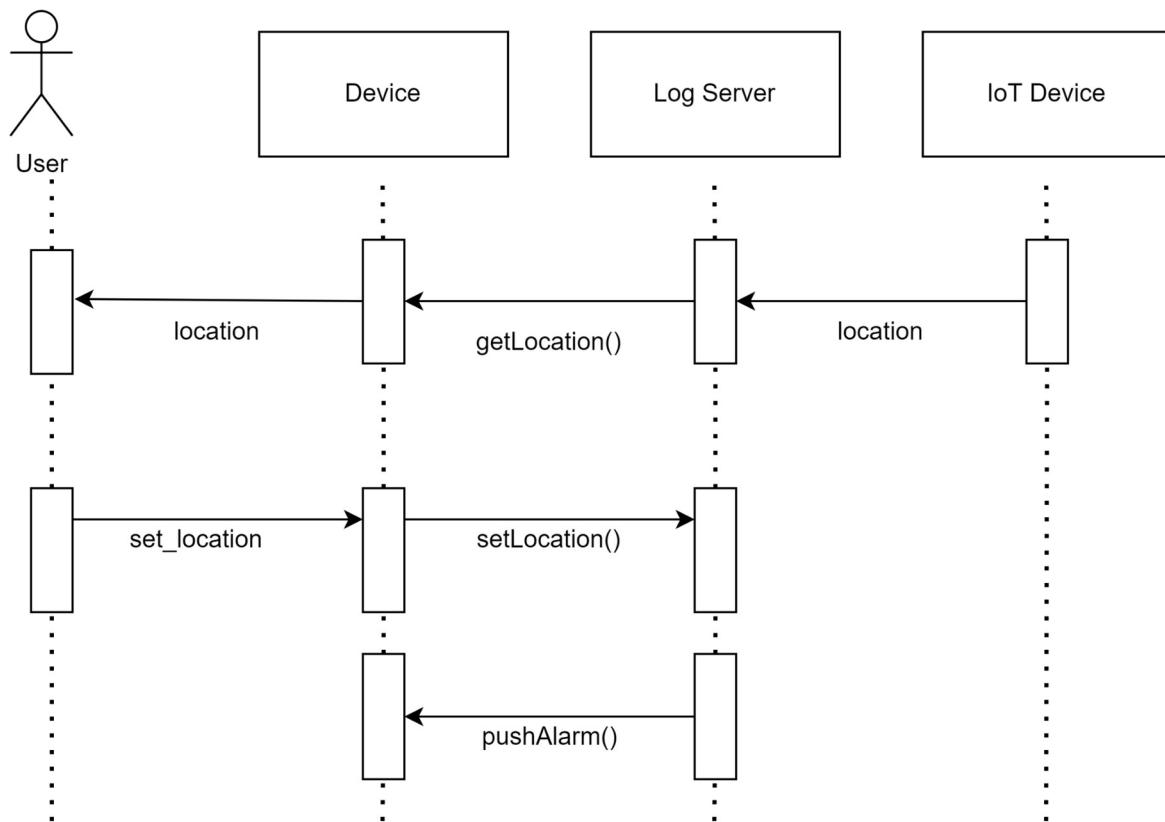
5.2.4.1. Class Diagram

그림 31. Class Diagram - 반려동물 실시간 위치파악 Backend



5.2.4.2. Sequence Diagram

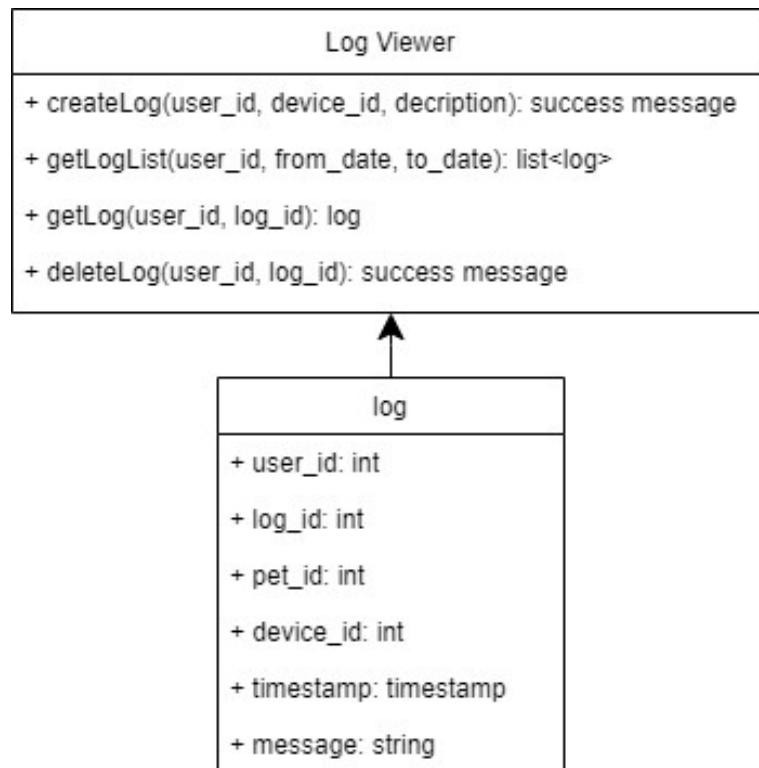
그림 32. sequence Diagram - 반려동물 실시간 위치파악 Backend



5.2.5. 반려동물 행동 로그

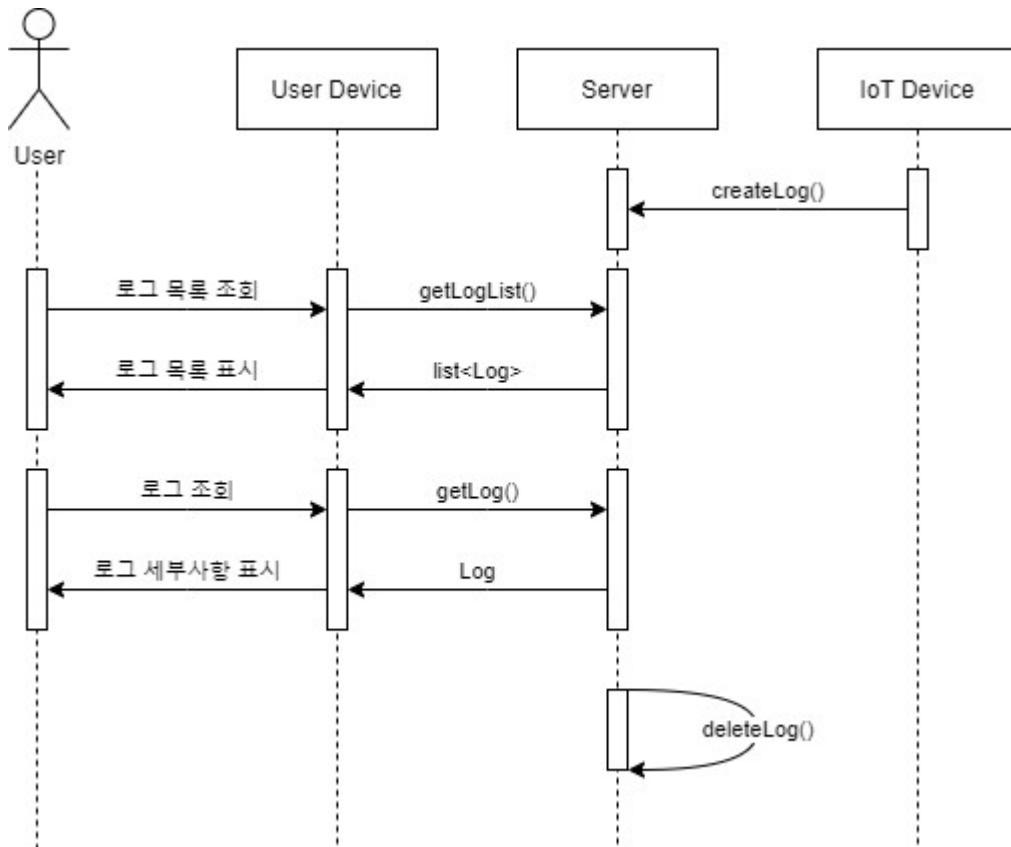
5.2.5.1. Class Diagram

그림 33. Class Diagram - 반려동물 행동 로그 Backend



5.2.5.2. Sequence Diagram

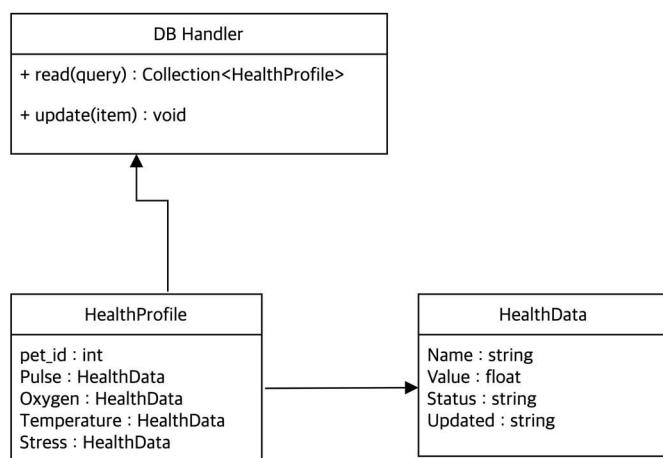
그림 34. Sequence Diagram - 반려동물 행동 로그 Backend



5.2.6 반려동물 건강상태 획득/분석

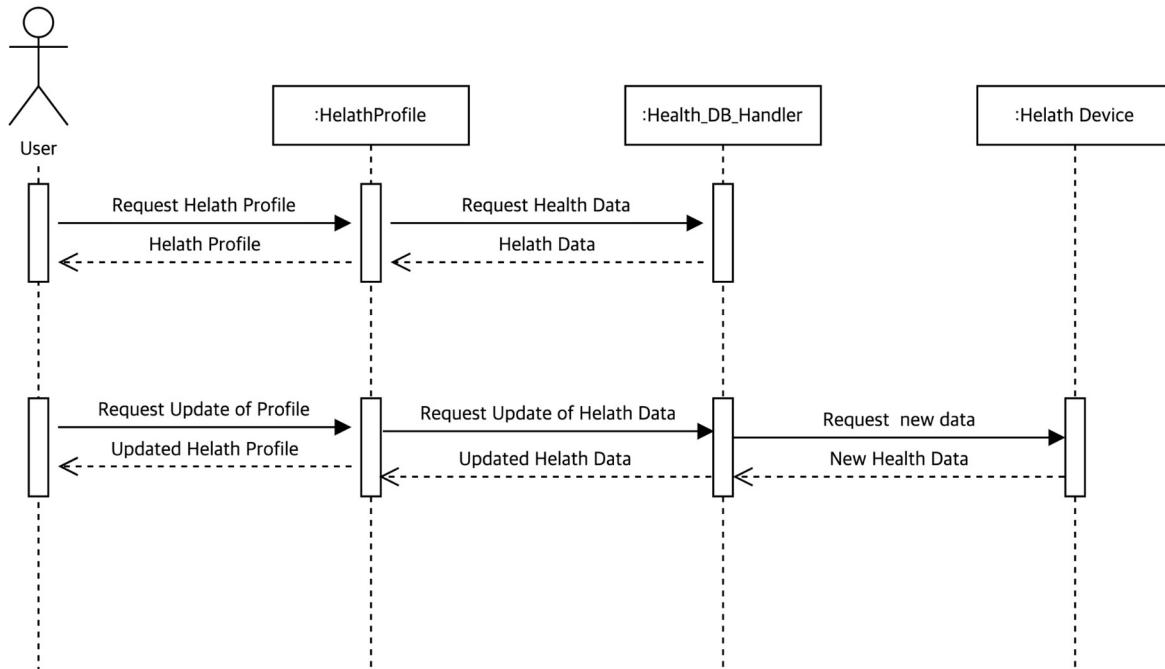
5.2.6.1. Class Diagram

그림 35. Class Diagram - 반려동물 건강상태 획득/분석 Backend



5.2.6.2. Sequence Diagram

그림 36. Sequence Diagram - 반려동물 건강상태 획득/분석 Backend

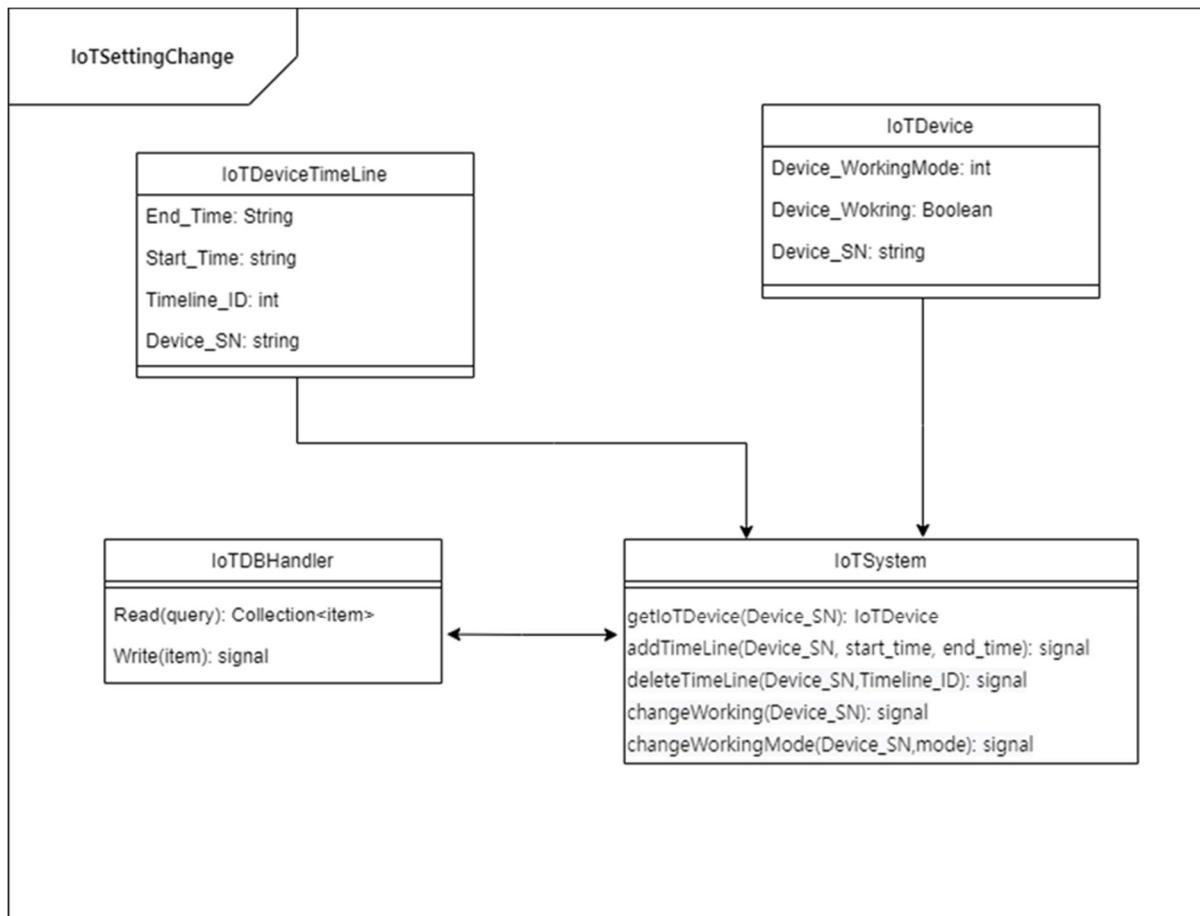


5.2.7. 반려동물 맞춤 케어 시스템 설정

IoTSystem은 사용자로부터 IoT 디바이스의 설정 변경 요구를 받고, 이를 데이터베이스에 저장하는 역할을 수행한다. 사용자가 IoT 디바이스의 작동 시간대를 추가하면 addTimeLine() 함수가 호출되고 추가한 시간대는 IoTHandler 를 통해 데이터베이스에 저장된다. 사용자는 IoT 디바이스의 작동 시간대를 삭제하는 경우 deleteTimeLine() 함수를 호출해서 IoTHandler 를 통해 삭제 대상 시간대가 삭제된다. 또한 사용자는 현재 IoT 디바이스가 작동/중지 상태일 때, 디바이스의 상태를 즉각적으로 중지/작동 상태로 만들 수 있다. 이 경우 changeWorking()이 호출되고 IoT 디바이스의 작동상태를 중지/작동 상태로 전환한다. 마지막으로 사용자는 IoT 디바이스의 작동 모드를 오토 케어 모드, 시간대별 작동 모드로 설정할 수 있다. 사용자가 디바이스의 작동모드를 변경하는 경우 changeWorkingMode()가 호출되고 IoTHandler 를 통해 데이터베이스에 저장된다.

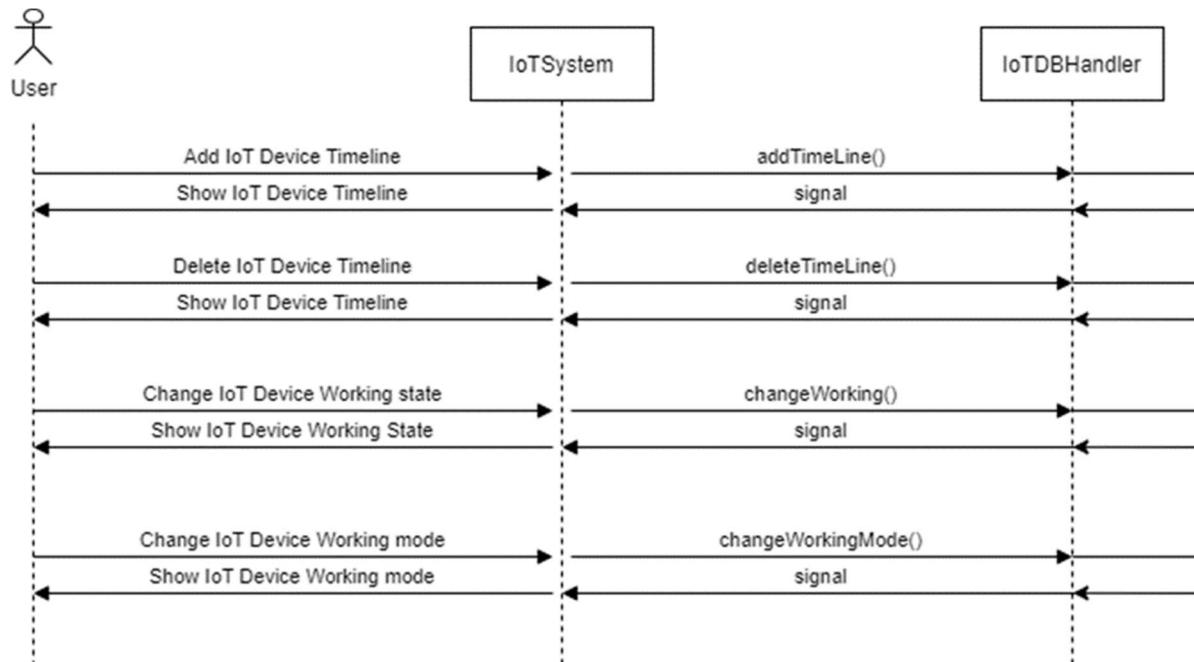
5.2.7.1. Class Diagram

그림 37. Class Diagram - 반려동물 맞춤 케어 시스템 설정 Backend



5.2.7.2. Sequence Diagram

그림 38. Sequence Diagram - 반려동물 맞춤 케어 시스템 설정 Backend



6. Protocol Design

6.1. Objectives

이 장에서는 하위 시스템 간의 상호 작용에 사용되는 프로토콜과 그 프로토콜의 전체 구조를 다룬다. 또한 애플리케이션과 서버 간의 통신과 각 인터페이스의 정의 방법을 설명한다.

6.2. 전달 형식

6.2.1. HTTP

HTTP는 "Hypertext Transfer Protocol"의 약자로, 웹 상에서 웹 서버 및 브라우저 상호 간의 데이터 전송을 위한 응용 계층 프로토콜이다. 초기에는 WWW 상의 하이パーテ스트 형태의 문서를 전달하는데 주로 이용되었지만 현재는 이미지, 비디오, 음성 등 거의 모든 형식의 데이터를 전송 가능하다. 요청 및 응답 메시지로 대응되는 구조이며, 클라이언트/서버 간에 HTTP 메세지를 주고 받으며 통신한다.

6.2.2. OAuth

OAuth는 인증을 위한 개방형 표준 프로토콜로, 인터넷 사용자들이 비밀번호를 제공하지 않고 다른 웹사이트 상의 자신들의 정보에 대해 웹사이트나 애플리케이션의 접근 권한을 부여할 수 있는 공통적인 수단으로서 사용된다. 사용자들이 Facebook이나 트위터 같은 인터넷 서비스의 기능을 다른 애플리케이션(데스크톱, 웹, 모바일 등)에서도 사용할 수 있게 하는 등 다양한 플랫폼 환경에서 권한 부여와 접근 위임을 위한 산업 표준 프로토콜이다. OAuth의 작동 방식은 크게 인증(Authentication)과 권한(Authorization)을 획득하는 것으로 볼 수 있다. 인증은 시스템 접근을 확인하는 절차로 로그인의 개념이라고 할 수 있으며, 권한은 행위의 권리를 검증하는 절차이다.

6.2.3. JSON

JSON은 JavaScript Object Notation의 약자로, 자바스크립트 객체 문법으로 구조화된 문자 기반의 데이터 포맷이다. 경량의 데이터 교환 형식으로 데이터를 저장하거나 전송할 때 XML을 대체하여 널리 사용되고 있다.

6.3. Authentication

6.3.1. 사용자 인증

표 1. 요청 - 사용자 인증

Attribute	Detail	
Method	OAuth	
Request Body	Email	사용자 계정 이메일
	Password	사용자 계정 비밀번호
	Request token	OAuth 용 토큰

표 2. 응답 - 사용자 인증

Attribute	Detail	
Success Code	HTTP 200 OK	

Failure Code	HTTP 400 Bad Request	
Success response body	Access Token	사용자 인증 토큰
	Message	통신 성공 메시지
Failure response body	Message	통신 실패 메시지

6.4. IoT 디바이스 등록 및 설정

6.4.1. 연결 가능 IoT 디바이스 목록 조회

표 3. 요청 - 연결 가능 IoT 디바이스 목록 조회

Attribute	Detail	
URI	/user/:id/IoT/	
Method	GET	
Header	Authorization	사용자 인증
Request Body	User	사용자 계정 정보
	Location	사용자 현 위치

표 4. 응답 - 연결 가능 IoT 디바이스 목록 조회

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
Success response body	Access Token	사용자 인증 토큰
	Nearby Devices	현 위치 기반 연결 가능한 디바이스 목록

	Message	통신 성공 메시지
Failure response body	Message	통신 실패 메시지

6.4.2. 신규 IoT 디바이스 등록 및 설정

표 5. 요청 - 신규 IoT 디바이스 등록 및 설정

Attribute	Detail	
URI	/device/:id/	
Method	POST	
Header	Authorization	사용자 인증
Request Body	User	사용자 계정 정보
	User Device	등록할 IoT 디바이스 정보

표 6. 응답 - 신규 IoT 디바이스 등록 및 설정

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
Success response body	Access Token	사용자 인증 토큰
	Message	통신 성공 메시지
Failure response body	Message	통신 실패 메시지

6.5. 반려동물 실시간 위치 파악

6.5.1. IoT 디바이스 위치 조회

표 7. 요청 - IoT 디바이스 위치 조회

Attribute	Detail	
URI	/device/:id/:deviceSN/location	
Method	GET	
Header	Authorization	사용자 인증
Request Body	User	사용자 계정 정보
	Device SN	조회할 디바이스 일련번호

표 8. 응답 - IoT 디바이스 상태 조회

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
Success response body	Access Token	사용자 인증 토큰
	Device Location	디바이스 위치 정보
	Message	통신 성공 메시지
Failure response body	Message	통신 실패 메시지

6.5.2. 디바이스 위치 이동

표 9. 요청 - 디바이스 위치 이동

Attribute	Detail	
URI	/device/:id/:deviceSN/control	
Method	POST	
Header	Authorization	사용자 인증
Request Body	User	사용자 계정 정보
	Device SN	조회할 디바이스 범위(전체/개별아이디)
	Device Destination	디바이스 희망 목적지 정보(지도 내 좌표)

표 10. 응답 - 디바이스 위치 이동

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
Success response body	Access Token	사용자 인증 토큰
	Device SN	조종한 디바이스 일련번호
	Device Location	(이동 후) 디바이스 위치 정보
	Message	통신 성공 메시지
Failure response body	Message	통신 실패 메시지

6.6. 반려동물 행동 로그

6.6.1. 반려동물 행동 로그 조회

표 11. 요청 - 반려동물 행동 로그 조회

Attribute	Detail	
URI	/log/:id/	
Method	GET	
Header	Authorization	사용자 인증
Request Body	User	사용자 계정 정보
	Period	조회할 기간 정보

표 12. 응답 - 반려동물 행동 로그 조회

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
Success response body	Access Token	사용자 인증 토큰
	Log List	해당 기간 반려동물 행동 로그 리스트
	Message	통신 성공 메시지
Failure response body	Message	통신 실패 메시지

6.7. 반려동물 건강체크

6.7.1. 반려동물 건강 지표 조회

표 13. 요청 - 반려동물 건강 지표 조회

Attribute	Detail	
URI	/healthCheck/:id/:petId	
Method	GET	
Header	Authorization	사용자 인증
Request Body	User	사용자 계정 정보
	Pet id	반려동물 식별자

표 14. 응답 - 반려동물 건강 지표 조회

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
Success response body	Access Token	사용자 인증 토큰
	Health Info	반려동물 최신 건강 지표 및 측정 시각
	Message	통신 성공 메시지
Failure response body	Message	통신 실패 메시지

6.8. 반려동물 맞춤 케어

6.8.1. IoT 디바이스 상태 조회

표 15. 요청 - IoT 디바이스 상태 조회

Attribute	Detail	
URI	/device/:id/:deviceSN	
Method	GET	
Header	Authorization	사용자 인증
Request Body	User	사용자 계정 정보
	Device Range/SN	조회할 디바이스 범위(전체/개별아이디)

표 16. 응답 - IoT 디바이스 상태 조회

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
Success response body	Access Token	사용자 인증 토큰
	IoT Device Working	디바이스 상태 정보(작동 여부)
	Message	통신 성공 메시지
Failure response body	Message	통신 실패 메시지

6.8.2. IoT 디바이스 작동 설정

표 17. 요청 - IoT 디바이스 작동 설정

Attribute	Detail	
URI	/device/:id/:deviceSN	
Method	POST	
Header	Authorization	사용자 인증
Request Body	User	사용자 계정 정보
	Device SN	설정할 IoT 디바이스 일련번호
	Device Timeline	IoT 디바이스 작동 시간대 정보
	Device Working Mode	IoT 디바이스 작동 모드 정보 (오토 케어모드/지정 시간대 작동모드)

표 18. 응답 - IoT 디바이스 작동 설정

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
Success response body	Access Token	사용자 인증 토큰
	Device SN	설정된 IoT 디바이스 일련번호
	Message	통신 성공 메시지
Failure response body	Message	통신 실패 메시지

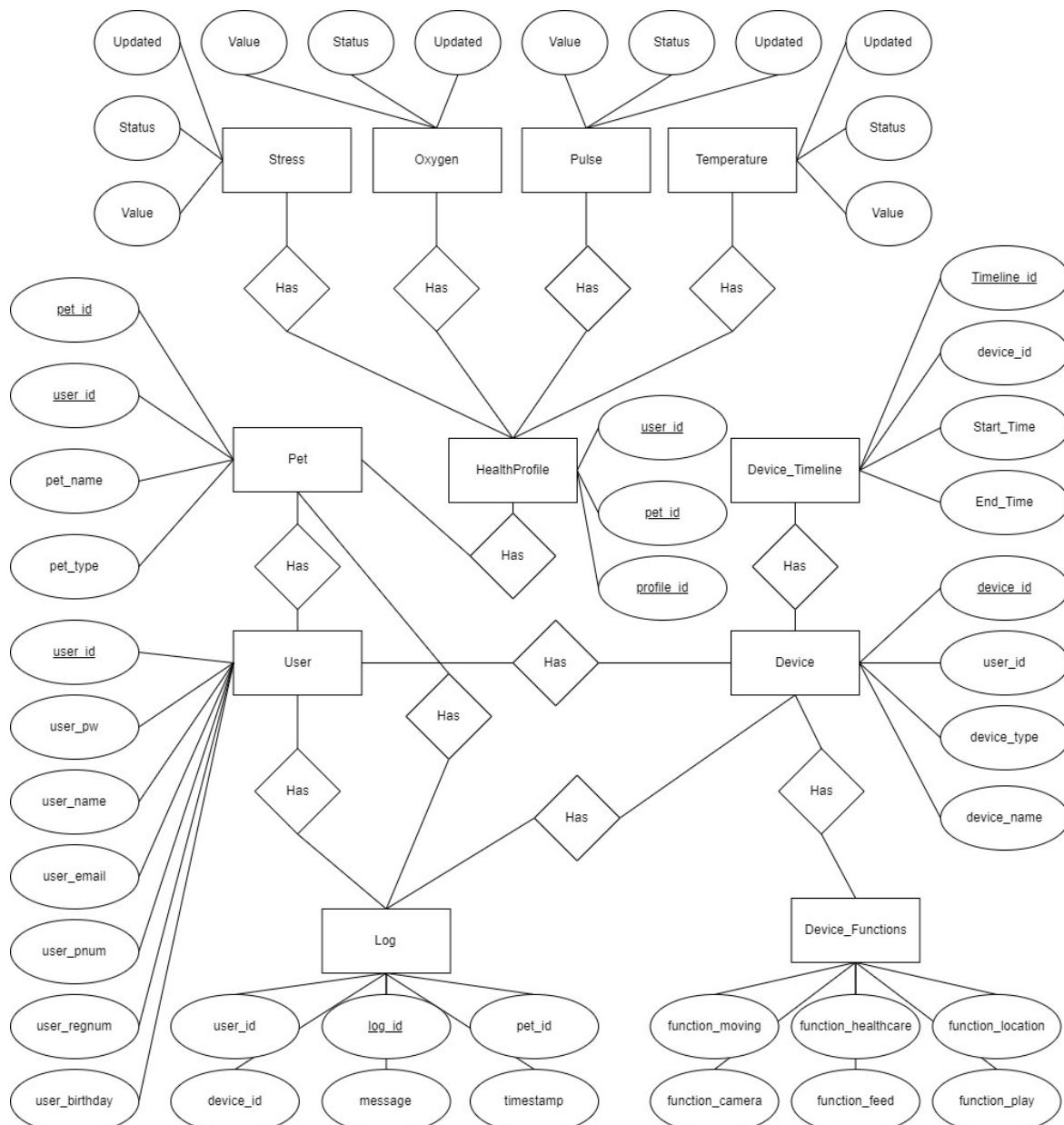
7. Database Design

7.1. Objectives

이 장에서는 본 프로그램에서 사용되는 데이터베이스의 구성과 그 상세를 다룬다. 데이터베이스의 Entity-Relation Diagram 과 각 Entity, Relation Schema 와 SQL DDL 을 다룬다.

7.2. ER Diagram

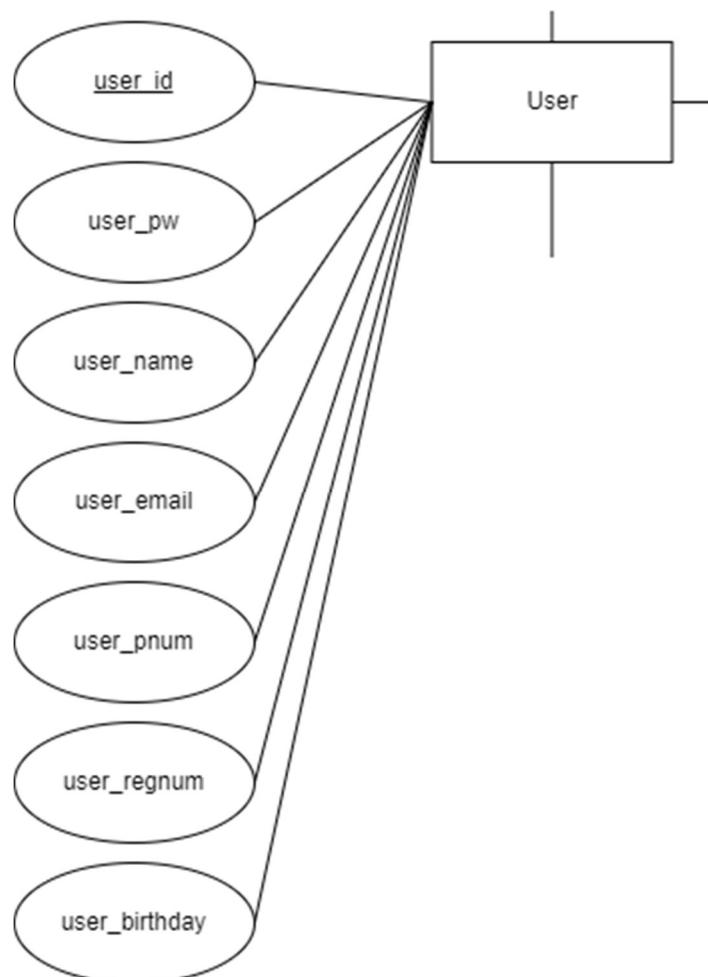
그림 39. ER Diagram



7.2.1. Entities

7.2.1.1. User

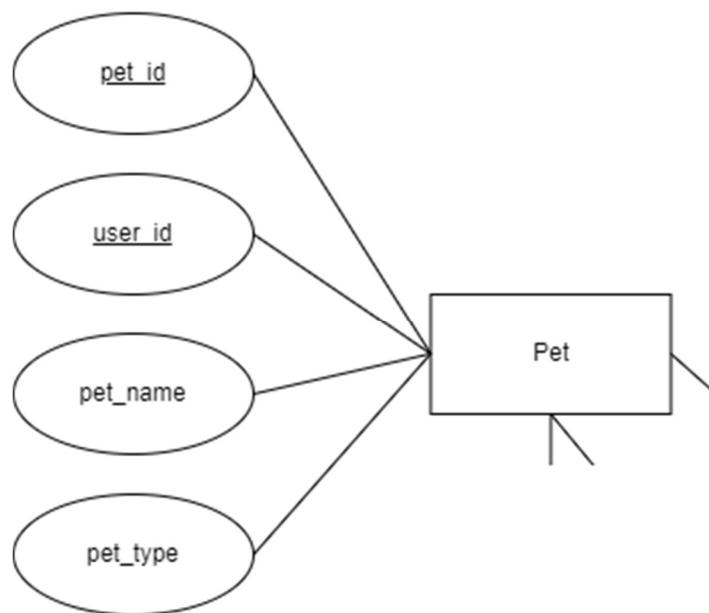
그림 40. ER Diagram - User



이 엔티티는 유저에 관련된 정보를 표현한다. 아이디, 패스워드, 이름, 이메일, 휴대폰 번호, 주민등록 번호, 생년월일을 포함한다. 이중 user_id 가 기본키이며 중복될 수 없다. Device, Pet, Log, HealthProfile 엔티티와 1:N 관계를 가진다.

7.2.1.2. Pet

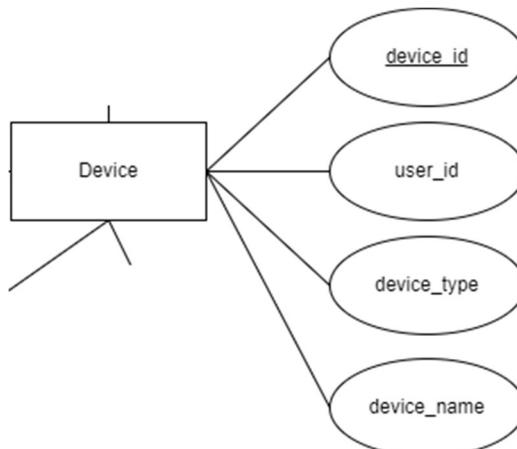
그림 41. ER Diagram - Pet



이 엔티티는 반려동물에 대한 정보를 표현한다. 반려동물 고유 id 를 pet_id 로 부여하며 기본키기본키로 사용한다. 반려동물의 이름, 종류를 저장한다. user_id 를 외래키로 가지며 HealthProfile 엔티티와는 1:1 관계, Log 엔티티와는 1:N 관계를 가진다.

7.2.1.3. Device

그림 42. ER Diagram - Device

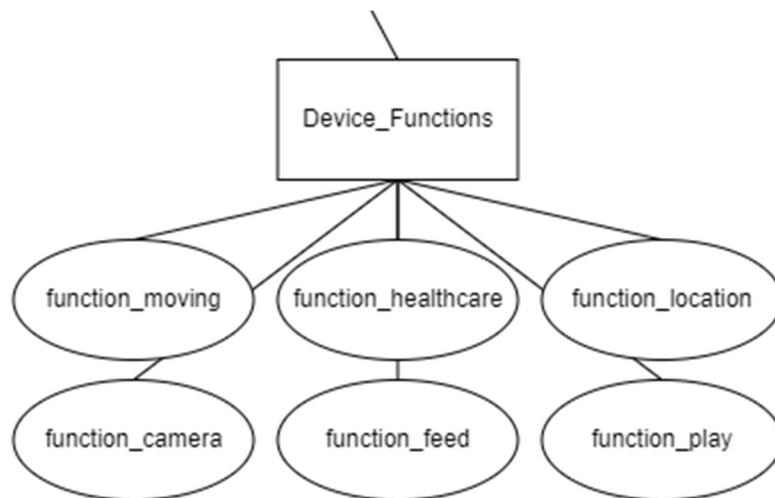


이 엔티티는 IoT 디바이스의 정보를 표현한다. 각 IoT 디바이스를 device_id 라는 기본키를 사용해

구분하며, user_id 는 외래키이다. IoT 디바이스의 종류, 이름등을 저장한다. DeviceFunctions, DeviceTimeline 과 1:1 관계를 가진다.

7.2.1.4. DeviceFunctions

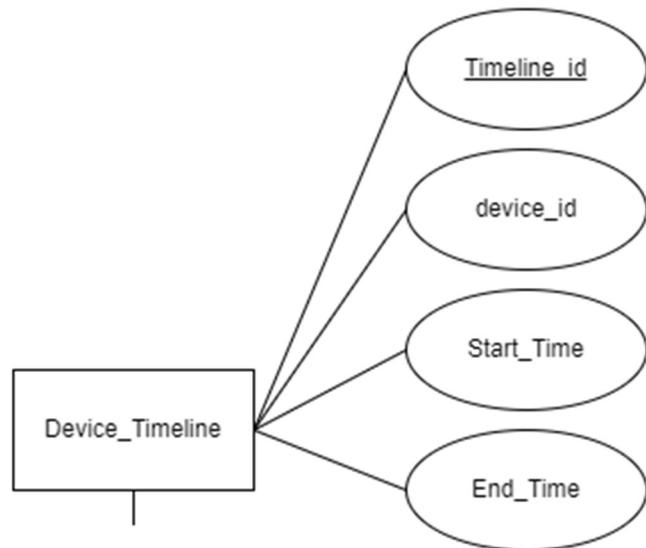
그림 43. ER Diagram - DeviceFunctions



이 엔티티는 IoT 디바이스가 수행하는 기능에 대한 정보를 표현하고 있다. IoT 디바이스의 이동, 건강상태, 위치, 카메라, 먹이 급여, 놀이 기능의 여부를 boolean 으로 표현한다. device 와 1:1 관계를 가진다.

7.2.1.5. DeviceTimeline

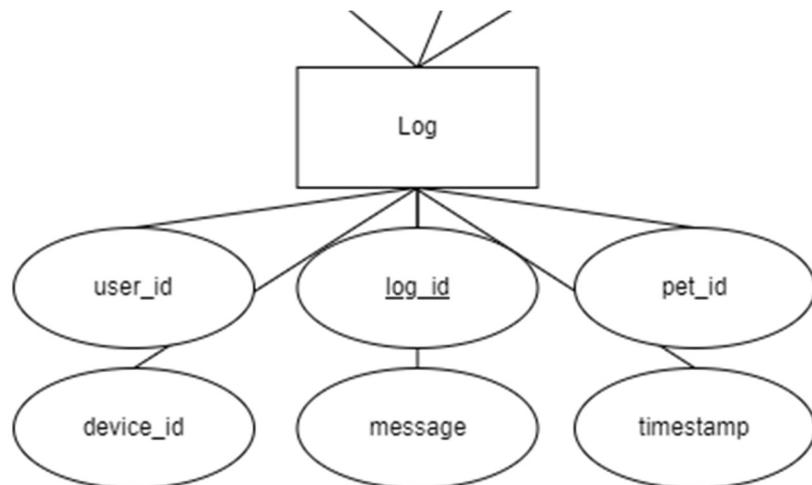
그림 44. ER Diagram - DeviceTimeline



이 엔티티는 IoT 디바이스가 작동 스케줄에 대한 정보를 표현하고 있다. timeline_id 를 기본기본키로 가진다. Start_Time 과 End_Time 을 통해 IoT 디바이스의 작동 시작 시간과 정지 시간을 저장한다. Device 엔티티와 1:1 관계를 가진다.

7.2.1.6. Log

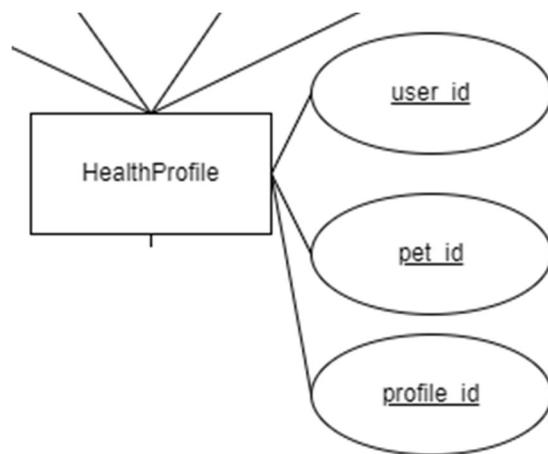
그림 45. ER Diagram - Log



이 엔티티는 IoT 디바이스에서 감지된 반려동물의 행동 로그를 표현한다. log_id 를 기본키로 가진다. user_id, pet_id, device_id 는 각각 User, Pet, Device 엔티티의 외래키이다. message, timestamp 로 로그 세부 내용과 로그 발생 시간을 저장한다.

7.2.1.7 HealthProfile

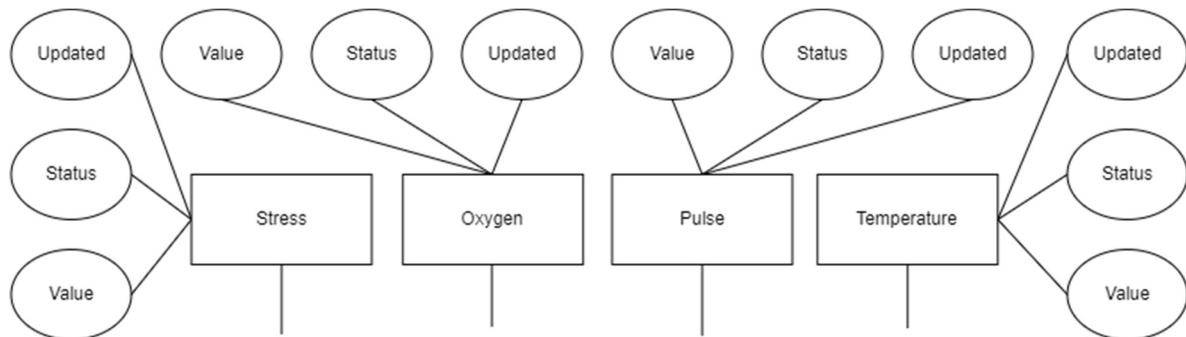
그림 46. ER Diagram - HealthProfile



이 엔티티는 각 반려동물의 건강상태를 표현한다. profile_id 를 기본키로 가진다. user_id 와 pet_id 의 외래 키를 통해 User 엔티티와는 1:N, Pet 엔티티와 1:1 관계를 가진다. Stress/Oxygen/Pulse /Temperature 엔티티와 1:1 관계를 가진다.

7.2.1.8. Stress/Oxygen/Pulse/Temperature

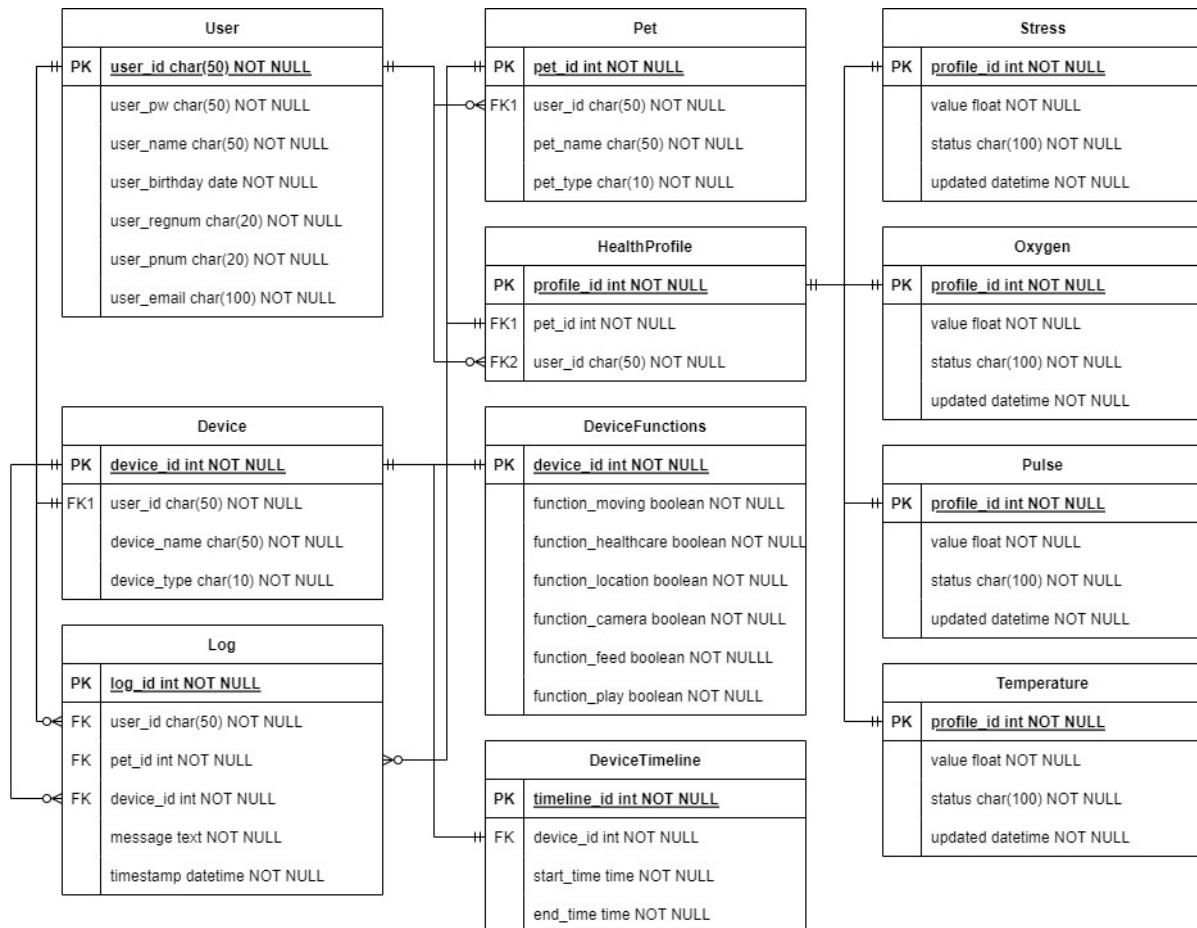
그림 47. ER Diagram - Stress/Oxygen/Pulse/Temperature



이 엔티티는 각 반려 동물의 HealthProfile 의 세부 정보를 표현한다. 4 가지 엔티티 모두 value, status, update 를 가진다. 각각 HealthProfile 과 1:1 관계를 가진다.

7.3. Relational Schema

그림 48. Relational Schema



7.4. SQL DDL

7.4.1. User

```

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`)
)
  
```

7.4.2. Pet

```
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`))
```

7.4.3. Device

```
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`))
```

7.4.4. DeviceFunctions

```
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`))
```

7.4.5. DeviceTimeline

```
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`))
```

7.4.6. Log

```
CREATE TABLE IF NOT EXISTS `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`)
```

7.4.7. HealthProfile

```
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`)
)
```

7.4.8. Stress/Oxygen/Pulse/Temperature

```
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/Oxygen/Pulse/Temperature 모두 동일한 DDL이다.

8. Testing Plan

8.1. Objectives

이번 장에서는 반려동물 펫케어 시스템 소프트웨어에 대한 테스트 기획 및 정책을 소개하고자 한다. 여기서 테스팅은 프로그램이 설계된대로 잘 작동되는지 혹은 치명적인 결함이 있는지를 배포 혹은 실사용 전에 확인하는 중요한 작업에 해당한다. 이번 프로젝트의 테스트 정책은 크게 세 가지로 나뉘는데 첫 번째는 Development testing, 두 번째는 Release testing이고 마지막으로 User testing이다.

8.2. Testing Policy

8.2.1. Development Testing

Development Testing은 시스템 상의 버그나 에러를 찾기 위해 개발 과정에서 시행되는 테스트로서 시스템 개발자와 심지어 디자이너까지 모두 참여하게 된다. Development Testing은 3 가지 단계들로 구분된다. 첫번째 단계는 Unit Testing으로 Unit 단위의 프로그램이나 객체 클래스들을 테스트하는 과정으로 해당 객체나 메소드의 기능성을 테스트하는 것에 목적을 둔다. 두 번째는 Component Testing으로 여러 Unit 들이 통합된 형태인 Component 들에 대하여 테스팅을 진행하게 되며 컴포넌트의 인터페이스와 기능을 테스트 하는 것을 중점으로 둔다. 마지막은 System testing으로서, 시스템의 일부 혹은 모든 컴포넌트들을 통합시켜 테스팅 하는 작업이며, 이는 컴포넌트 간의 상호작용을 테스트 하는데 목적으로 한다. 이러한 점을 바탕으로 본 시스템에서는 Development Testing 단계에서 다음의 테스팅을 진행할 것이다.

8.2.1.1. Unit Testing

그림 49. 반려동물 맞춤 케어 HealthProfile 클래스

HealthProfile
- pulse
- oxygen
- temperature
- stress
+ getHealthInfo() + updateHealthInfo() + getPulse() + updatePulse() + getOxygen() + updateOxygen() + getTemperature() + updateTemperature() + getStress() + updateStress()

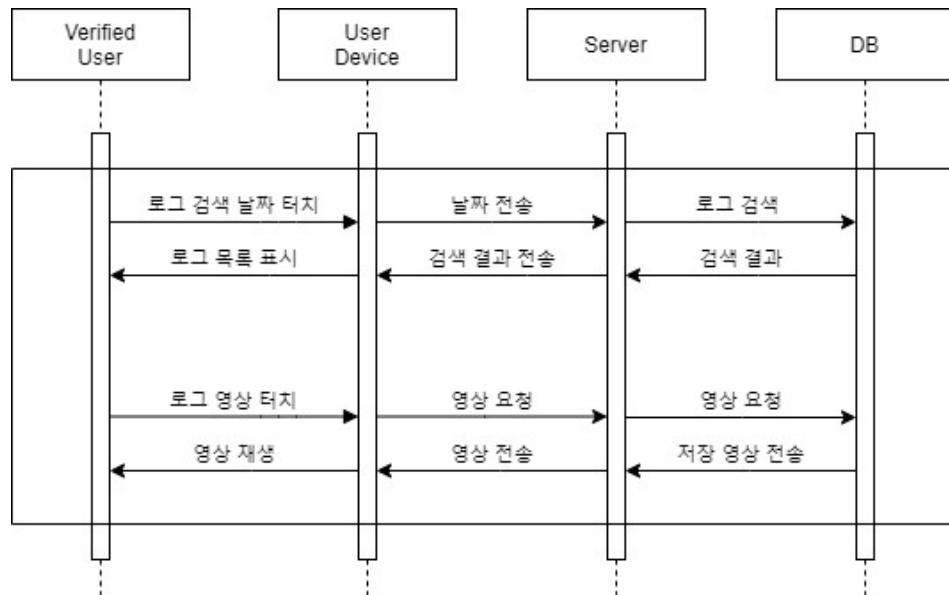
Unit Testing에서는 펫 케어 컴포넌트를 구성하는 각각의 클래스들에 대해 테스트를 진행할 예정이다. 예를 들어, 반려동물 맞춤 케어의 Unit에 해당하는 HealthProfile 클래스는 관찰 대상인 반려동물의 맥박수, 산소포화도, 체온과 스트레스 지수 등의 건강정보를 얻어 프로필을 구성하는 기능을 맡고 있다. 이때 Unit Test를 통해 해당 클래스의 건강 정보를 메소드인 getHelathInfo의 실행 과정에서 오류가 발생하지는 않는지 혹은 건강 정보를 얻어오는 과정에서 성능상의 이슈는 없는지 확인하게 될 것이다.

8.2.1.2. Component Testing

Component Testing은 Component를 구성하는 Unit들의 Unit Testing은 완료되었다는 가정 하에 진행되는 Testing이며, 객체(클래스)들 간의 상호작용이 발생하는 상황에 대해서 테스트를 진행하는 단계이다. 즉, Component Testing은 주로 해당되는 컴포넌트가 설계된 Interface에 따라 잘 작동하는지 테스트 하는 단계로 볼 수 있다. 스마트 펫 케어 시스템을 구성하는 각 컴포넌트들에 대하여 SRS에서 기획 및 설계하였던 User Interface와 Software Interface를 대로 잘 작동하는지를 테스트하게 될 것이다. Component Testing을 통해서 Unit Testing을 통해서는 확인할 수 없었던 인터페이스 상 오류를 해결하게 될 것으로 기대된다.

8.2.1.3. System Testing

그림 50. 반려동물 로그 조회 Sequence Diagram



System Testing은 컴포넌트들을 통합시켜 하나의 시스템 혹은 그 일부를 구성하고 이를 테스트 해보는 단계이다. 이 단계에서는 컴포넌트 끼리 서로 잘 상호작용 하는지, 서로가 잘 호환되는지 등을 확인하게 된다. System Testing은 Component Testing과 일부 겹치는 경향이 있지만, 독자적으로 각자 개발되거나 재사용 하는 컴포넌트 들이 새로운 하나의 컴포넌트로 통합되는 상황을 테스트 해본다는 점에서 큰 차이점이 있다.

위 그림 처럼, 반려동물 로그 조회를 하는 기능을 예로 들면, 로그 조회 시 다음의 과정들을 거치게 된다.

로그 검색 날짜 터치 => 날짜 전송 => 로그 검색 => 검색 결과 반환 => 검색 결과 전송 => 로그 목록 표시

위 과정들에서 컴포넌트들 간의 상호작용이 발생하는 만큼, 이러한 작업들이 무리없이 잘 진행되는지 혹은 컴포넌트 간 호환은 잘 되는지를 확인함으로써 System Testing 과정을 진행할 수 있다.

8.2.2. Release Testing

Release Testing은 시스템을 외부에 배포하기 위해 진행되는 테스트로서 해당 테스트를 통해 시스템에게 기능, 성능, 의존성 등의 요구사항들이 제대로 구현되었는지 확인하고 실제 사용 중에

에러나 오류가 발생하지 않을 것임을 보장해야 한다. 이러한 테스트는 시스템의 사용자로 하여금 현재 개발된 시스템이 바로 사용할 수 있을 만큼 잘 만들어졌음을 보여주기 위해 시행되는 테스트 과정이다. Release Testing은 크게 3 가지로 나누어진다.

8.2.2.1. Requirements-based testing

Requirements-based testing은 요구사항을 기반으로 이에 대한 테스트 케이스를 작성하여 개발된 시스템이 요구사항들을 만족하고 있는지를 확인하는 테스트 과정이다. 반려동물 맞춤케어나 반려동물 건강정보 체크와 같은 시스템 상에서의 요구사항들을 바탕으로 테스트 케이스를 작성하여 해당되는 요구사항이 개발된 시스템에서 잘 만족되고 있는지를 확인하게 될 것이다. 이 단계에서는 시스템 상에서 에러가 있는지를 확인하기보다는 요구사항이 잘 만족되고 있는지를 확인하는데에 중점을 두게 된다.

8.2.2.2. Scenario testing

Scenario testing은 시나리오를 작성하여 시스템의 테스트 케이스를 적절히 만들어낼 수 있도록 하는 테스트 방법이다. 여기서 시나리오란 시스템이 어떻게 사용될지에 대한 이야기를 의미하며 비록 가상의 이야기라고 할지라도 현실적이어야하고 실제로 시스템의 사용자들과 관련되어질 수 있도록 작성되어야한다. 해당 시스템에서는 반려동물 보유자가 스마트홈 펫 케어 시스템을 사용한다는 상황에서 나올 법한 시나리오들을 작성한 뒤 이러한 시나리오들을 개발된 시스템에 돌려보는 과정에 해당된다.

작성된 시나리오를 기반으로 시스템을 실행 시킨 뒤 이에 대한 output들을 확인하는데 이번 테스팅에서는 요구사항 만족 여부 뿐만 아니라 성능 문제 같은 시스템 상의 문제가 발생하는지를 같이 확인해야 한다. 또한 Scenario testing의 경우에는 여러 요구사항들이 잘 반영되었는지 하나의 시나리오를 통해서 확인하는 것이 가능하다.

8.2.2.3. Performance testing

Performance testing은 시스템 요구사항들이 만족되고 있는지 확인할 뿐만 아니라 시스템 상에서의 결함이나 문제들을 확인하는 것들이 관련되어 있다. 성능적인 요구사항이 잘 반영되고 있는지 확인하려면 Operational profile을 작성해야 한다. 여기서 Operational profile이란 시스템의 여러 기능 테스트들을 혼합하되 실제 시스템에서 특정 기능들이 차지하는 비율에 최대한 맞게 맞추도록 한 것을 의미한다. 스마트 홈 펫 케어 시스템에 적용한다면, 이용자 계정 관리, 이상행동 관찰 알림, 실시간 위치파악, 건강체크, 행동 로그 조회, IoT 디바이스 제어 기능들의 비율에 맞춰 테스트들을 구성하는 것이 이에 해당할 것이다. 한편, Operational profile만으로는 시스템 결함을 제대로 확인하는데에는 한계가 있어 서 이를 해소하기 위해서는 Stress Testing을 진행할 수 있다. Stress Testing은 시스템

설계상 처리가능한 Input 수에 근접하거나 혹은 그 이상으로 Input 을 제공하여 시스템이 얼마나 견고한지를 판단할 수 있다. 스마트 홈 펫 케어 시스템에서는 등록가능한 반려동물 수를 테스트하거나 등록할 수 있는 IoT 디바이스들의 개수를 테스트하는데 이용될 수 있을 것이다.

8.2.3. User Testing

User Testing 은 시스템 사용자나 고객이 시스템에 대한 Input 을 제공하고 이들의 결과들에 대해 피드백을 제공해주는 형태의 테스트 방법이다. User Testing 의 경우는 개발자들이 따라하는 것이 불가능 한데 이는 사용자가 직접 사용하는 환경을 개발자가 따라 하는 것이 불가능하기 때문이다. User Testing 은 크게 세가지로 나뉜다.

첫번째인 Alpha Testing 은 소프트웨어 사용자 그룹과 개발자들이 면밀하게 같이 소프트웨어의 초기 테스트를 진행하는 방식이다. 즉 유저들이 개발자들의 입장에서는 파악되지 않았던 소프트웨어의 문제점들을 확인할 수 있는 테스트 이다.

두번째인 Beta testing 은 소프트웨어를 많은 사용자들이 사용할 수 있게끔 배포하는 방식을 통해 소프트웨어 상에서 발생할 수 있는 문제점들을 파악하기 위한 테스트이다. 특히나 다양한 스마트폰 환경에서 사용될 수 있는 스마트홈 펫 케어 시스템의 경우에는 많은 사용자들이 미리 사용할 수 있도록 Beta testing 을 진행하는게 중요한데 개발자들이 일일이 다양한 사용자들의 환경 속에서 소프트웨어를 실행시켜보는 것은 거의 불가능하기 때문이다.

마지막인 Acceptance testing 은 소프트웨어 고객들이 사용자 환경상에서 최종 배포될 수 있는지의 여부를 확인하는 테스트 이다. 이는 Testing 의 최종 단계에 해당된다.

8.2.4. Testing Case

스마트 홈 펫 케어 시스템을 테스트 하는 목적은 "요구 사항대로 사용될 수 있는지", "실제 사용과정에서 사용자의 보안상 문제가 발생하지는 않는지", "실제 사용자의 환경 속에서 시스템이 원하는 성능을 내는지" 를 확인하기 위함이다. 첫 번째의 경우에는 사용자 시나리오를 기반으로 테스트를 진행하여 원하는 결과를 발생하는지 확인할 수 있는 테스트 케이스를 선정할 것이다. 두 번째의 경우에는 시스템의 보안상 약점을 파고드는 환경이나 시나리오를 구성한 뒤 시스템을 테스트 해봄으로써 확인해 볼 수 있는 항목이다. 마지막의 경우에는 다양한 사용자 환경 속에서 다양한 사이즈의 input 으로 시스템을 실행시켜 봄으로써 원하는 성능이 나오는지를 확인할 수 있을 것이다.

9. Development Plan

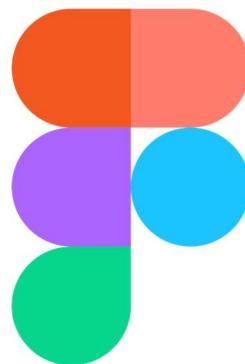
9.1. Objectives

이번 챕터에서는 시스템 개발에서 사용될 프로그램이나 모듈 혹은 프레임워크에 대한 내용을 기술할 것이다.

9.2. Frontend Environment

9.2.1. Figma

그림 51. Figma 로고



피그마는 웹 기반의 UI/UX 디자인 및 프로토타이핑 툴로서 누구나 쉽게 인터페이스를 디자인하는데에 유용한 점을 가지고 있다. 어도비 일러스트레이터의 벡터 이미지 제작 기능과 어도비 Xd UI/UX 디자인 툴의 장점들을 모아놓은 툴이라고 평가된다. 특히나 디자이너와 개발자간의 협업에 특화되어 링크 소유자가 온라인 상에서 실시간으로 프로토타입을 확인할 수 있고, 개발자가 참조할 수 있는 개발 툴바가 있어서 따로 가이드 라인 없이도 빠른 작업이 가능하다.

9.2.3. Flutter

그림 52. Flutter 로고



구글에서 출시한 모바일/웹/데스크톱 크로스 플랫폼 GUI SDK로서 하나의 코드를 기반으로 iOS나 안드로이드와 같은 서로 다른 모바일 환경 상에서도 원활하게 작동할 수 있는 앱 제작에 최적화된 어플리케이션 프레임워크이다. 프로그래밍 언어로는 Dart가 사용되며 크로스플랫폼 환경에서도 네이티브한 성능을 낼 수 있으며, 소스 코드 수정 후 에뮬레이터나 기기에 변경된 UI와 로직이 바로 반영이 되는 Hot Reloading 기능이 제공되어 개발상에서도 이점이 존재한다.

9.3. Backend Environment

9.3.1. AWS

그림 53. AWS 로고



아마존 EC2는 아마존 클라우드 컴퓨팅 플랫폼의 서비스로 유저들로 하여금 컴퓨터들을 과금을 통해 빌릴 수 있도록 하여 사용자의 서버 어플리케이션을 호스팅할 수 있도록 할 수 있다. AWS

EC2 에서는 다양한 기능들을 제공하는데 대표적으로 사용자의 트래픽에 따라 서버 인스턴스의 규모를 늘리고 줄일 수 있는 Auto Scaling 기능을 제공한다. 트래픽이 몰릴 경우 서버 인스턴스를 자동으로 늘려주어 늘어난 트래픽에 대응할 수 있고, 트래픽이 줄어들 경우에는 서버 인스턴스를 줄여주는 기능을 제공해준다.

9.3.2. Node.js

그림 54. NodeJS 로고



NodeJS 는 오픈소스 Javascript 엔진인 크롬 V8 에 비동기 이벤트 처리 라이브러리인 libuv 가 결합된 플랫폼으로 Javascript 를 통해 서버를 구축할 수 있는 코드를 실행할 수 있게 하는 런타임 환경이다. NodeJS 는 논블로킹 I/O 와 단일 스레드 이벤트 루프를 통해 고성능의 비동기 이벤트 처리 성능을 보이며 내장 HTTP 서버 라이브러리를 포함하고 있어 웹 서버에서 아파치 등의 별도의 소프트웨어 없이도 동작하는 것이 가능하다.

9.3.3. MariaDB

그림 55. MariaDB 로고



MariaDB 는 오픈 소스의 관계형 데이터베이스 관리 시스템으로 MySQL 과 동일한 소스 코드를 기반으로 하며 오라클 소유의 불확실한 MySQL 의 라이선스 상태에 반발하여 만들어졌다. MySQL 과 높은 호환성을 유지함과 동시에 MySQL 의 API 와 명령에 정확히 매칭되는 반면 MySQL 에 비해서는 어플리케이션 부분 속도에서 약 4~5 천배 정도 빠르며, 최고 성능 면에서 70%의 향상을 보인다고 한다.

9.4. Constraints

스마트 홈 펫 케어 시스템은 이전에 작성된 요구사항 명세서 및 현재 문서에서 상세화 된 아키텍쳐 디자인에 따라 구현될 예정이다. 이 외의 세부적인 구현 방향성은 아래의 제약사항에 따라 결정되고 일부 내용은 테스트 결과에 의해 혹은 사용자 요구 사항 변경에 의해 변경될 수 있다.

- Android 9.0 (API 26) 이상의 OS 버전에서 동작 가능
- IOS 14.0 이상의 OS 버전에서 동작 가능
- Android 11.0 에서 어플리케이션 테스트
- IOS 15.0 에서 어플리케이션 테스트
- 향후 관리를 위해 주석 사용 / 코드 최적화 작업

9.5. Assumptions and Dependencies

펫 케어 어플리케이션은 최소 Android 9.0 (API 26) 이상의 OS 버전, IOS 14.0 이상의 OS 버전을 사용하는 휴대 기기에서 사용되어야 한다. IOT 기기는 서버에 연결된 상태여야 한다.

10. Supporting Information

10.1. Software Design Specification

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

10.2. Document History

표 19. SDS Document History

Date	Version	Description	Writer
2022/05/07	0.1	문서작업 시작	전체
2022/05/10	0.1.1	Front-End / Back-End 반려동물 건강정보 Class, Sequence Diagram 작성	양승찬
2022/05/10	0.1.2	Front-End / Back-End IoT 디바이스 등록 및 설정 Class, Sequence Diagram 작성	김준식
2022/05/10	0.1.3	Front-End / Back-End 반려동물 맞춤 케어시스템 Class, Sequence Diagram 작성	차현묵
2022/05/10	0.1.4	Front-End / Back-End 반려동물 행동 로그 Class, Sequence Diagram 작성	곽재원
2022/05/10	0.1.5	Front-End / Back-End 이상행동 앱푸시 알림 Class, Sequence Diagram 작성	최하늘
2022/05/11	0.2	Protocol Design 작성	최하늘

2022/05/11	0.2.1	Database Design 작성	곽재원
2022/05/12	0.2.2	Account Management Front-End / Back-End Class, Sequence Diagram 작성	이경돈
2022/05/12	0.2.3	Preface, Introduction, System Architecture- overall 작성	차현묵
2022/05/13	0.2.4	Front-End / Back-End IoT 디바이스 등록 및 설정 일부 변수명 및 Class Diagram 수정	김준식
2022/05/13	0.2.5	Testing & Development Plan 작성	양승찬
2022/05/13	0.2.6	System Architecture - Frontend/Backend Objectives 작성 및 Backend 의 Overall Architecture 제거 개요 번호 수정	김준식
2022/05/13	0.2.7	반려동물 실시간 위치 파악 작성	유새하
2022/05/13	0.2.8	Database Design 수정	곽재원
2022/05/13	0.2.9	반려동물 위치파악 추가	유새하
2022/05/15	1.0.0	표/그림 목차 추가	양승찬
2022/05/15	1.1.0	오타 수정 및 서식 통일	전체