Smart Dressing Table

Software Design Specification

2022.05.12

Team 4

Team Leader	박재성
Team Member	이영신
Team Member	최정훈
Team Member	안영태
Team Member	김학산
Team Member	임재원
~~Team Member~~	~~전준혁~~

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Preface

본 장에서는 Smart Dressing Table (이하 SDT)에 대한 Software Design Specification의 독자 정보, 범위, 문서의 목적, 시스템의 전체적인 구조에 대해서 다룰 것이다.

1.1 Readership

본 Software Design Specification는 10장으로 구성되며 각 장은 세부적인 분류 기준에 의거하여 몇몇의 소제목으로써 세분화되어 제공된다. 본 문서의 주 독자는 2022학년도 1학기 소프트웨어공학개론 강의를 수강 중인 학생들과 교수자이며, 향후 해당 시스템이 출시되어 상용화될 경우, SDT를 구매하여 이용할 소비자와 본 시스템으로 자회사의 서비스를 판촉할 법인, 그리고 해당 서비스와 관련한 다수의 이해관계자 등을 부가적인 독자로 상정한다.

1.2 Scope

본 Software Design Specification는 화장품의 보관이라는 기능과 더불어 사용자의 피부를 분석하여 개인의 피부 타입에 맞는 화장품을 추천하는 기능을 제공하는 SDT 시스템의 Software Engineering 과정과 Software Quality Engineering 전 과정에 걸쳐 기준점으로 사용될 것이다.

1.3 Objective

본 Software Design Specification의 주된 목적은 SDT의 제작 과정에 있어서의 기술적인 명세를 제공하는 것이다. 해당 명세서에서는 SDT의 구현을 위해 채택된 설계적, 디자인적인 결정 사항들에 대해 명시할 것이며, 시스템을 바라보는 다양한 관점들에 입각하여 SDT의 총체적인 구조를 나타낼 것이다. 또한, 본 Design Specification에서는 이전에 작성하였던 Software Requirement Specification 상에서 다루었던 SDT 시스템 내 존재하는 모듈들에 대해서 구조와 디자인적으로 구체화한 자료를 제공할 것이다. 그리고 본 프로젝트의 개발팀에서 각 모듈을 구현하는 과정을 나타내는 use cases에 대한 정보를 sequential, activity diagram을 통해 시각적으로 제공할 것이다. 본 명세서의 주된 독자는 SDT 서비스와 관련한 이해관계자와 시스템을 제작에 참여하는 개발자, 디자이너, 테스트 관계자들로 상정하며 이외에도 프로젝트의 진척 정도와 서비스의 출시 여부에 따라 추후에 독자층이 늘어날 수 있음을 적시한다.

1.4 Structures

- Preface : 해당 장에서는 독자들에 관한 구절과 본 명세서의 범주, 시스템의 목적, 그리고 명세서의 목적과 구조에 대해서 다룬다.
- Introduction : 해당 장에서는 본 명세서를 작성하는 과정에서 사용된 도구들과 다이어그램들, 그리고 참고문헌에 대한 설명을 수록하며 이와 더불어 프로젝트의 목적을 기술한다.
- System Architecture : 해당 장에서는 context diagram, sequential diagram, use case diagram 등을 이용하여 시스템의 전체적인 구조에 대한 설명을 제공한다.
- System Architecture – Frontend : 해당 장에서는 SDT 시스템 내 Frontend 구조를 다이어그램을 통해 나타낸다.
- System Architecture – Backend : 해당 장에서는 SDT 시스템 내 Backend 구조를 다이어그램을 통해 나타낸다.
- Protocol Design : 해당 장에서는 SDT 시스템 내 여러 컴포넌트들 간의 통신을 매개하기 위한 프로토콜에 대한 설명이 수록된다.
- Database Design : 해당 장에서는 SDT내 데이터베이스의 구조와 시스템 내 데이터의 흐름에 관한 정보를 제공한다,
- Testing Plan : 해당 장에서는 시스템의 테스트에 대한 실행 계획에 대해 다룬다.
- Developing plan : 해당 장에서는 시스템의 개발 과정에서 사용될 개발 도구들에 대해 기술한다, 또한 본 장에서는 개발 과정에 있어서의 제약 사항과 가정, 의존성에 대한 설명이 수록된다.
- Supporting information : 해당 장에서는 Software Design Specification 정보와 문서 작성 이력 정보와 같이 본문의 내용들을 보충하기 위한 부가적인 자료들이 수록된다.

2. Introduction

본 프로젝트의 주된 목적은 사용자들에게 개인 맞춤형 화장품을 구매하고 구매한 제품을 효율적으로 사용할 수 있는 환경을 제공하는 것이다. 이와 같은 목표를 달성하기 위해서 기존의 화장대가 가지고 있던 보관의 기능에 더하여 SDT는 사용자가 화장대를 통해 화장품의 구매부터 사용, 보관까지의 전 과정에 걸쳐 개인화된 서비스를 향유할 수 있도록 한다. 우선 기존의 화장대에서 제공하는 보관의 기능을 강화하기 위해 SDT는 화장대 디바이스 내에 온도와 습도 감지 센서를 탑재하여 사용자에게 화장대 내 환경에 대한 정보를 실시간으로 통지하며, 이와 더불어 SDT는 화장대에서 보관하고 있는 화장품에 대한 정보를 데이터베이스에 보관하면서 특정 화장품의 기대 효과나 제품의 사용 빈도, 구매 주기 등을 분석하는 기능을 제공한다.

또한, 사용자는 스마트 거울 형태의 디스플레이를 통해 자신의 현재 모습을 확인할 수 있으며, 이에 더 나아가 탑재된 센서를 이용하여 피부의 유분, 수분, ph, 퍼스널 컬러 등의 피부 상태를 본 시스템 내에서 진단할 수 있다. 그리고 진단된 자료를 바탕으로 화장품이 추천되어 사용자로 하여금 자신의 피부 타입에 맞는 화장품을 별다른 노력 없이 구매할 수 있도록 한다. 또한. 구매 과정에서 키워드와 필터링을 통해 보다 정밀하게 화장품을 추천해주는 화장품 검색 기능을 제공한다. 본 Software Design Specification에서는 SDT의 제작 및 구현에 사용될 다수의 디자인 사항들에 대한 세부적인 묘사가 수록될 것이다. 이때 묘사된 디자인 사항들은 앞선 Software Requirement Specification에 언급된 요구사항들에 기반하여 고안되었다.

2.1 Objectives

이어질 장에서는 SDT 프로젝트의 디자인 과정에서 사용된 다이어그램들과 작업 도구들에 관한 설명이 수록될 것이다.

2.2 Applied Diagrams

2.2.1 UML

UML(Unified Modeling Language)은 시스템의 문서화, 구체화, 설계 과정에서 사용되는 다이어그램들을 표준화한 모델링 언어이다. 또한 UML는 규모가 크고 복잡한 시스템을 성공적으로 디자인한 선례들의 집합체라고도 일컬을 수 있다. UML을 사용함으로써, 소프트웨어 개발자는 시스템을 정의하고 구현하는 과정에서 UML을 통해 작업을 촉진시킬 수 있을 뿐더러, 프로젝트 팀 전체의 소통을 강화할 수 있으며, 소프트웨어의 잠재적 디자인을 미리 탐색해볼 수 있고, 이전에 제시된 디자인에 대한 유효성을 검사할 수 있다.

2.2.2 Use case Diagram

Use case Diagram은 use case의 관점에서 시스템의 기능적 요구사항 (functional requirements)를 묘사하는 방법이다. 해당 다이어그램은 시스템의 의도된 기능 (use case)와 이를 둘러싼 환경 (actor) 간의 관계를 표현하는 모델로, 이때의 관계는 서로 다른 use case들 간의 관계 뿐만 아니라 use case와 actor 간의 관계도 포함된다. Use case Diagram은 시스템의 범주를 나타내거나 시스템의 기능을 시각적으로 나타내며, 사용자가 시스템에서 무엇을 원하는지, 그리고 시스템이 이러한 니즈를 어떻게 사용자에게 전달하는지에 대해 묘사한다,

2.2.3 Sequence Diagram

Sequence Diagram은 객체들의 조합을 통시적 흐름에 따라 나타내는 모델링 기법이다. 이 다이어그램은 객체들이 특정한 use case의 시나리오에서 어떻게 상호작용하는지를 나타내며 각 객체들은 직사각형 모양으로 표현되고, 이러한 객체들의 생애는 lifeline이라는 개념을 통해 점선으로 표현되며, 객체들 사이의 메시지는 화살표 모양으로 그려진다. 소프트웨어 엔지니어는 Sequence Diagram을 이용하여 새로운 시스템이나 기존의 비즈니스 프로세스를 나타낼 수 있다. 이때 Sequence Diagram은 시스템 내에서 특정한 업무가 여러 객체 혹은 컴포넌트 사이에서 어떻게 전달되고 처리되는지에 대한 시각적인 도움 자료로써 활용된다.

2.2.4 Class Diagram

Class Diagram은 UML 상의 여러 클래스, 메소드, 어트리뷰트들 간의 관계와 의존성에 대하여 묘사한 정적인 구조 다이어그램(Static Structural Diagram)이다. Class Diagram에서의 클래스는 공통의 특성을 가진 그룹으로 정렬된다. Class diagram은 모든 형태의 객체 지향 프로그래밍(Object Oriented Programming))에 적용될 수 있으며, OOP의 패러다임이 진화함에 따라 본 다이어그램의 형태도 지속적으로 변화하고 있다.

2.2.5 Context Diagram

Context Diagram은 특정 시스템이 다른 시스템과 프로세스들로 이루어진 환경에서 어떠한 방식으로 존재하는지를 나타내는 다이어그램이다. 본 다이어그램은 엔티티들 간의 상호작용을 나타내면서 시스템, 시스템 내부의 컴포넌트, 시스템 주변 환경 사이의 경계를 정의한다. Context Diagram은 시스템의 요구사항과 제약 사항을 고안하는 과정에서 시스템 외부의 요인들과 이벤트들에 대한 주의를 환기시킴으로써 활용된다. 따라서 Context Diagram은 프로젝트 초창기에 프로젝트의 범주를 정의하는 작업에서 협의점을 찾아내기 위해 주로 사용되며, Software Design Specification에 뿐만 아니라 Software Requirement Specification에도 포함되는 모델링 기법이다.

Context Diagram은 모든 이해관계자들에게 그 의미가 정확히 해석되어야 하므로 제작 시 자연어로 작성되어야 한다.

2.2.6 Entity Relationship Diagram

Entity Relationship Diagram (ERD)는 구조화된 데이터에 대한 표현 양식으로, 시스템 내에서 여러 개체(Entity)들이 상호 간에 어떻게 관계를 맺고 있는지를 시각적으로 표현한 다이어그램이다. 또한 ERD에서는 다양한 엔티티, 어트리뷰트, 관계들 간의 연결성을 명시적으로 나타내기 위해 정사각형, 마름모, 타원, 연결선과 같은 미리 정의된 상징 체계를 바탕으로 다이어그램이 제작된다. ERD는 소프트웨어 엔지니어링이나 비즈니스 정보 시스템 분야 등에서 관계형 데이터베이스를 디자인하거나 디버깅하는 과정에 주로 사용되어, 데이터베이스의 생성, 관리, 유지보수의 품질을 높이기 위한 수단으로 활용된다.

2.3 Applied Tools

2.3.1 Microsoft PowerPoint

PowerPoint는 마이크로소프트 사에서 개발한 프레젠테이션 소프트웨어이다. 본 소프트웨어는 오피스 오픈 XML을 기반으로 구현되었으며 확장자로 .pptx, .ppt를 사용한다. 파워포인트에서는 서식 파일을 사용하거나 처음부터 슬라이드 형 프레젠테이션을 만들고, 해당 프레젠테이션 상에 이미지, 도형, 텍스트 등의 부가 자료를 추가하고, 슬라이드 간 전환, 애니메이션 효과를 부여할 수 있는 기능을 제공한다. 또한 프레젠테이션은 마이크로소프트 사가 운영하는 Onedrive에 저장하여 컴퓨터, 스마트폰, 태블릿 등의 플랫폼에서 동일하게 접근할 수 있다.

2.3.2 Miro

Miro는 온라인 기반 협업 플랫폼으로서, 협업에 참여하는 사람들이 공유된 디지털 화이트보드 상에 프로세스 플로우, 칸반 보드, 타임라인 등의 다양한 템플릿과 스티키 노트, 텍스트, 도형 등의 드로잉 도구들을 이용하여 효율적으로 공동 작업을 할 수 있는 환경을 제공한다. 화이트보드에는 시간과 장소에 구애받지 않고 의견을 기입할 수 있고, 수정된 내용은 실시간으로 모니터링되어 팀 구성원이 게시판에서 활동하는 내용을 지속적으로 확인할 수 있다, 또한 Miro는 Dropbox, Google Suite, JIRA와 같은 기성 서비스와 통합되더라도 일관적으로 동작할 수 있으므로 기존의 서비스들을 원래의 워크플로우와 달라지는 점 없이 제공할 수 있다.

2.3.3 diagrams.net

diagrams.net은 HTML5과 자바스크립트로서 개발된 오픈 소스 크로스 플랫폼 그래프 제작 소프트웨어이다. 본 소프트웨어는 UML 다이어그램을 비롯하여, 와이어프레임, 순서도, 네트워크 다이어그램 등을 무료로 별도의 설치 없이 제작할 수 있다. 제작한 그래프는 로컬 환경뿐만 아니라 깃헙, 구글 드라이브 등의 클라우드에도 저장할 수 있고, 웹 애플리케이션으로 존재하기 때문에 온라인 가입, 로그인 절차가 필요없이 로컬 PC의 하드 드라이브에서 그래프를 저장하고 드라이브 상의 그래프를 열람할 수 있다.

2.3.4 Microsoft Word

Word는 마이크로소프트 사에서 개발한 워드 프로세서이다. 본 소프트웨어는 오피스 오픈 XML을 기반으로 구현되었으며 확장자로 .dox, .docx를 사용한다. Word는 여러 가지 테마, 그리고 이미지, 도형 등을 사용하여 문서를 만들 수 있도록 지원하며 또한 문서 서식 또한 사용자가 직접 지정할 수 있게 한다. 또한 문서를 수정한 이력을 지속적으로 모니터링하여 예기치 않은 문서 유실을 대비할 수 있으며, 편집기와 같은 문서 교정 도구를 내장하고 있어 오탈자를 수정하는 기능 또한 가지고 있다. Word는 앞서 언급한 글쓰기 도구의 압도적인 기술적 우수성과 전 세계 약 90%가 넘는 절대적인 점유율을 바탕으로 업계 표준 워드 프로세서로 일컬어진다.

2.4 Project Scope

SDT는 화장품의 구매부터 보관까지의 전 과정에 걸쳐 사용자에게 개인 맞춤형 서비스를 제공하고자 하는 스마트 디바이스이다. 이러한 목적과 부합하게, SDT 프로젝트의 범위는 화장품과 관련한 사용자의 "데이터를 수집" 하는 작업에서부터, 수집된 "데이터에서 의미를 분석"하는 과정, 사용자의 피부 타입과 니즈에 맞는 화장품을 "예측하고 및 추천"하는 작업까지로 구획이 나뉘어진다. 우선 데이터 수집 과정에서 사용자의 피부 상태를 자동화된 방법으로 수집하기 위하여 디바이스에 부착된 피부측정기기를 이용하여 사용자의 피부의 수분, 산성도 등의 정보를 기기에 내장된 센서를 이용하여 얻어내고 또한, 현재 시중에서 제공되는 화장품에 대한 데이터를 수집하여 화장품 데이터베이스를 구축하고, 이를 통해 사용자들이 원하는 제품을 검색할 수 있도록 한다.

또한 수집된 데이터는 인공지능(Artificial Intelligence, AI) 기술을 사용하여 의미 있는 정보로 분석되어 사용자들에게 그들의 피부 상태에 대한 진단으로서 제공된다. 더 나아가 SDT의 사용자는 현재의 피부 타입을 넘어서 향후 이어질 소비 과정에서 자신에게 가장 적합한 화장품을 추천해주는 서비스나 화장품의 구매 주기를 예측하는 서비스 등을 이용할 수 있는데, 이러한 서비스들은 본 시스템에서 고안된 AI 기술 기반의 연산 기법을 데이터에 적용하면서 구현된다.

2.5 Reference

본 명세서의 독자는 다음의 자료들을 참고자료로서 활용할 수 있다.

- "proposal_team04". Team4. SKKU. Last Modified : Mar.27.2022. https://github.com/SKKU-SWE2022-spring-team4/2022spring_42class_team4/blob/main/proposal_team04.pdf
- Ian Sommervile. "Software Engineering 10th Edition". Pearson Higher Education. 2015

3. System Architecture – Overall

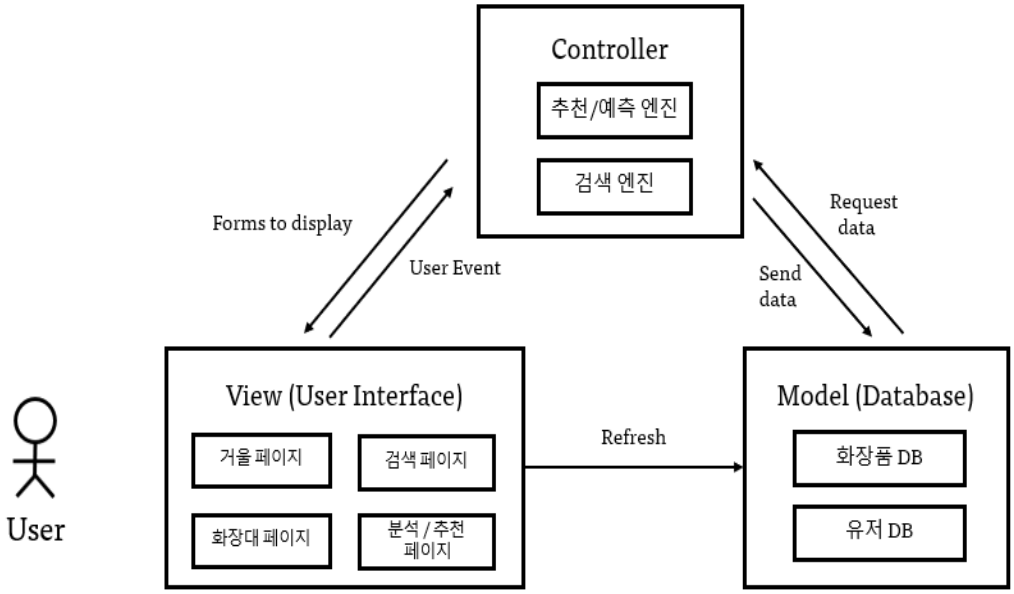
3.1 Objectives

본 장에서는 백엔드와 프론트엔드 디자인을 포함한 SDT 시스템의 전반적인 구조에 대한 묘사가 수록될 것이다.

3.2 System Organization

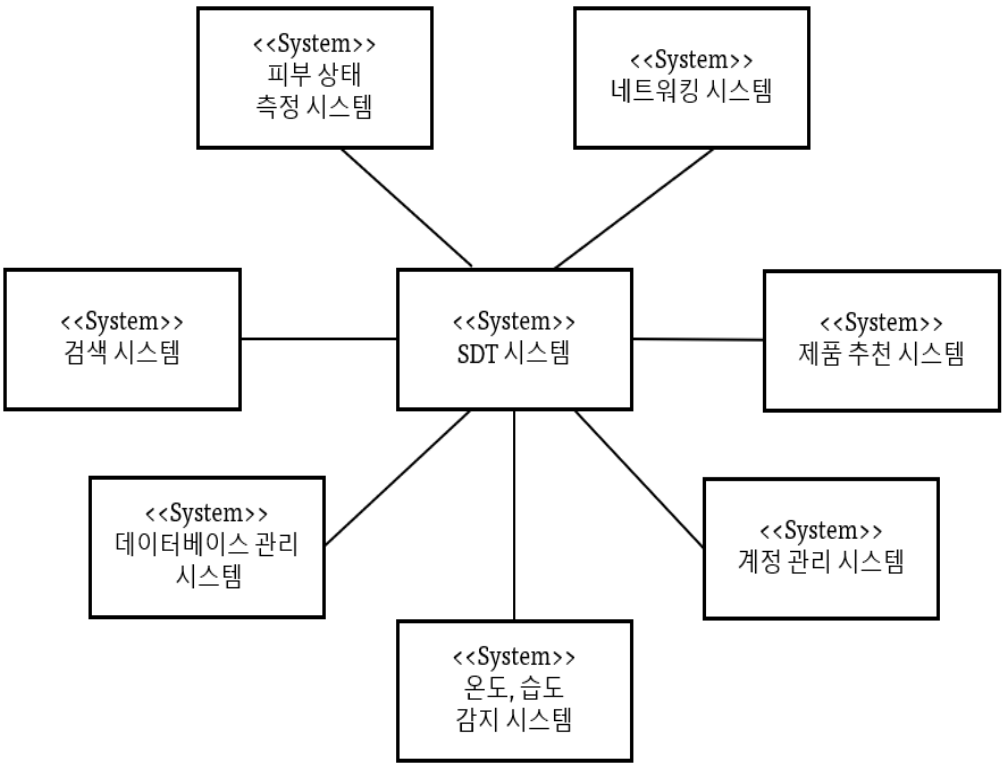
SDT는 서버-클라이언트 모델로 이루어진 시스템이다. SDT의 사용자(클라이언트)는 피부 측정, 분석 등의 서비스를 이용하기 위하여 서버에 접근하여 서비스를 요청하고, 클라이언트와 분리된 환경에서 서버는 데이터를 추출, 가공하는 작업을 서버 컴퓨터 내에서 진행하여 클라이언트에게 요청받은 서비스를 제공한다. 서버-클라이언트 모델을 구현하기 위해서 SDT는 MVC(Model-View-Controller)이라는 구조적인 패턴을 채택한다.

MVC 패턴 안에서 세 가지의 독립적인 컴포넌트는 서로 유기적으로 상호작용하며 SDT의 클라이언트와 서버 간 서비스 전달을 매개한다. 이때 컴포넌트들 간의 데이터 통신은 HTTP 프로토콜에 따라 이루어지며, 전달되는 데이터의 형태는 JSON이다. 우선, 시스템의 데이터를 관리하는 역할을 수행하는 Model은 SDT 서비스 구현을 위해 수집되는 사용자, 화장품 관련 데이터들과 해당 데이터를 조작하기 위한 오퍼레이션을 관리한다. 다음으로 유저의 인터페이스를 담당하는 컴포넌트인 View는 시스템의 데이터가 유저에게 표현되는 방식을 정의하고 관리한다. SDT는 사용자가 화장대 디바이스와 스마트폰 두 가지 방식으로 시스템에 접근할 수 있으므로, 두 가지 디바이스는 고유의 User Interface (View)를 가지고 있는 동시에 두 가지의 뷰는 검색 페이지, 화장대 페이지, 분석/추천 페이지, 거울 페이지의 공통 User Interface를 갖는다. 마지막으로 Controller는 View와 Model 사이의 상호작용을 매개하는 컴포넌트로서, 사용자가 View 상에서 검색 쿼리를 입력하거나, 화장품 추천 기능을 선택하였을 때, Controller는 해당 User Event를 해석하여 필요한 데이터를 Model에 요청하고, 이에 대한 응답으로 받은 데이터를 View에서 요구하는 형식, 내용에 맞게 가공하여 View에 전달한다.



[Figure 1] System Architecture

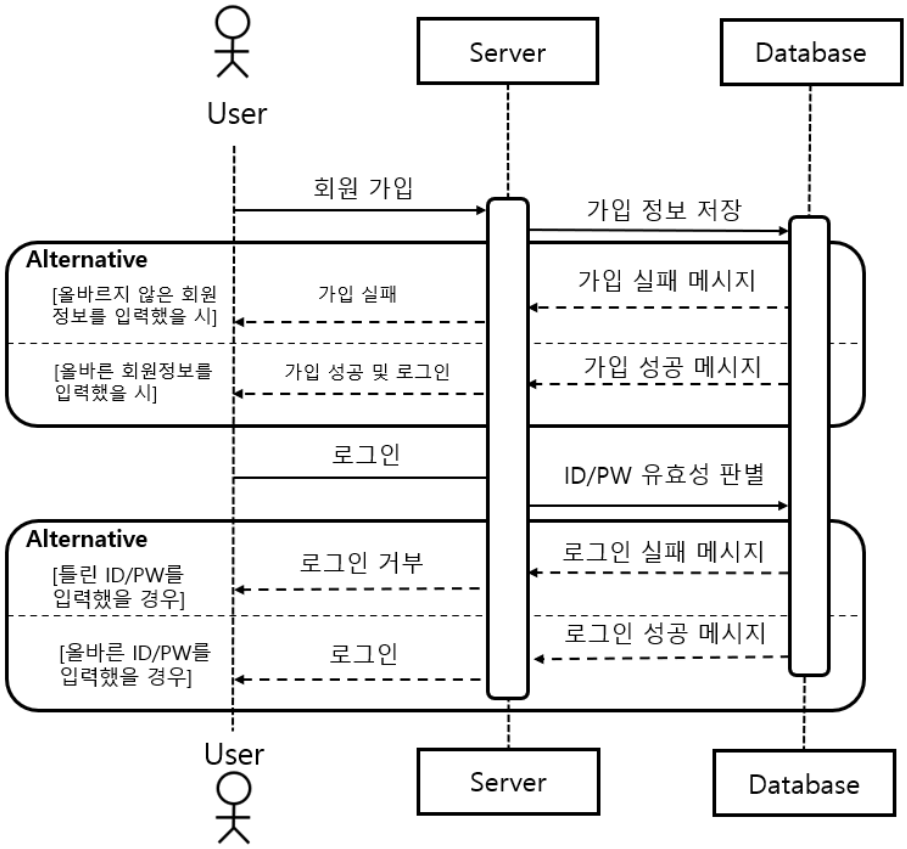
3.2.1 Context Diagram



[Figure 2] Context Diagram

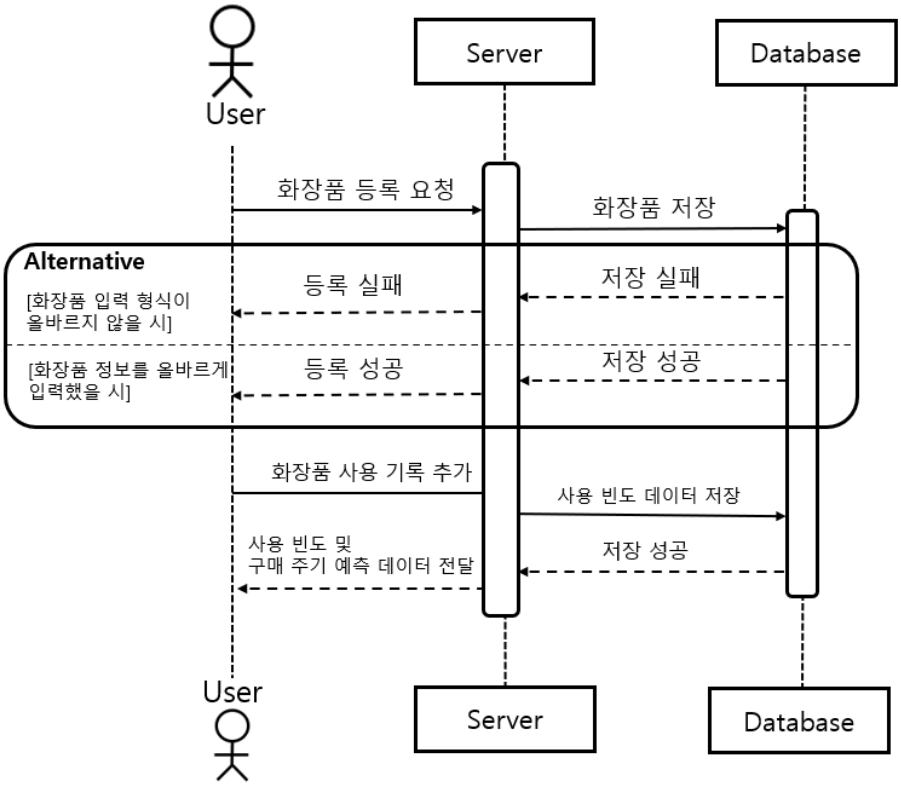
3.2.2 Sequence Diagram

3.2.2.1 Register & Login



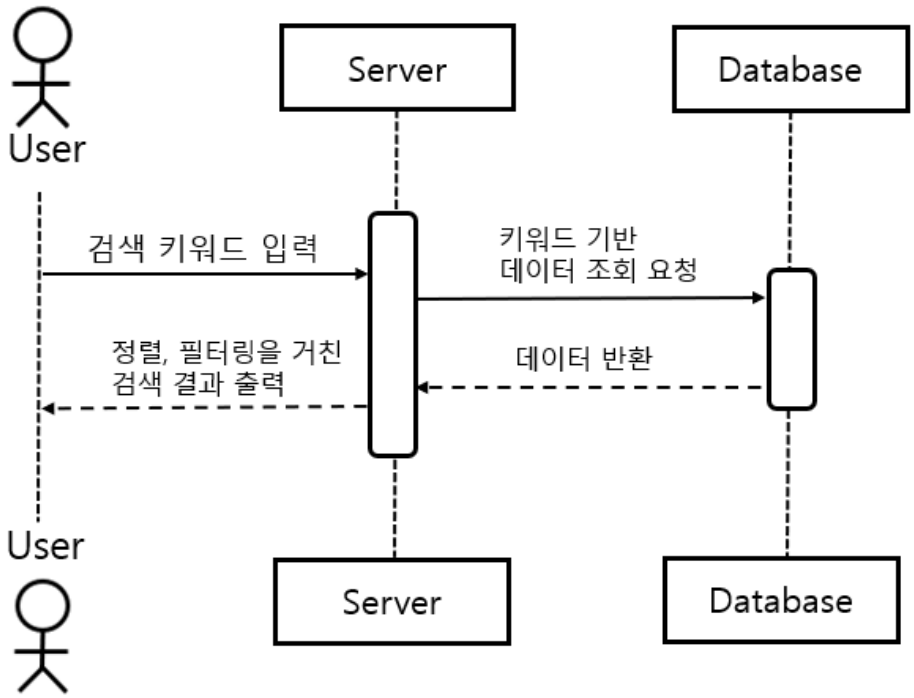
[Figure 3] Sequence Diagram for Register & Login

3.2.2.2 Add Cosmetics



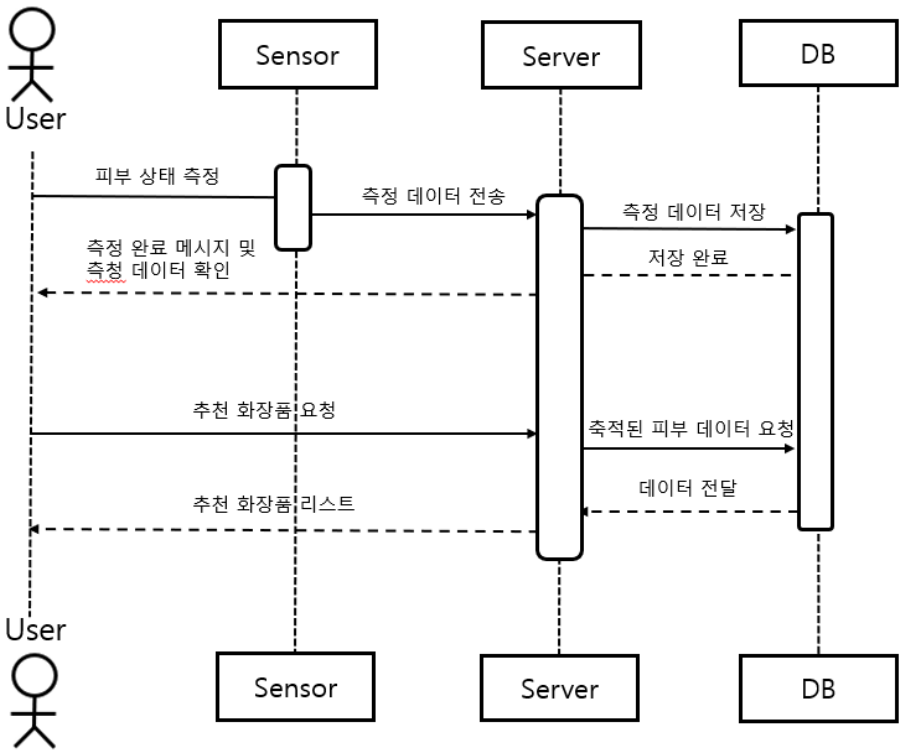
[Figure 4] Sequence Diagram for Add Cosmetics

3.2.2.3 Search Cosmetics



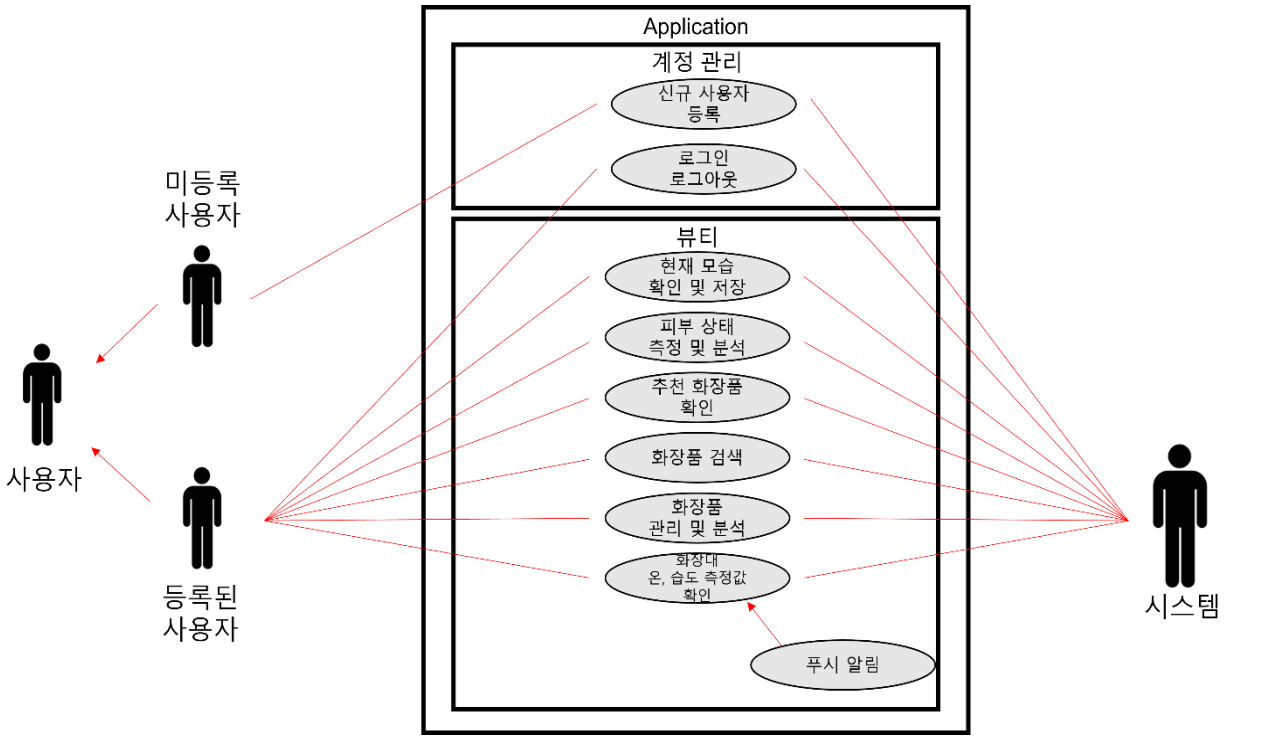
[Figure 5] Sequence Diagram for searching cosmetics

3.2.2.4 Cosmetics Recommendations



[Figure 6] Sequence Diagram for Cosmetics Recommendations

3.2.2 Use Case Diagram



[Figure 7] Use Case Diagram

4. System Architecture – Frontend

4.1 Objectives

System Architecture에서 사용자 인터페이스에 해당하는 Frontend 시스템을 이루는 Component 들의 구성을 Class Diagram과 Sequence Diagram으로 도식화 및 설명한다.

4.2 Subcomponents

4.2.1 Mirror Display

4.2.1.1 Attributes

해당 Mirror Display가 가진 attribute는 다음과 같다.

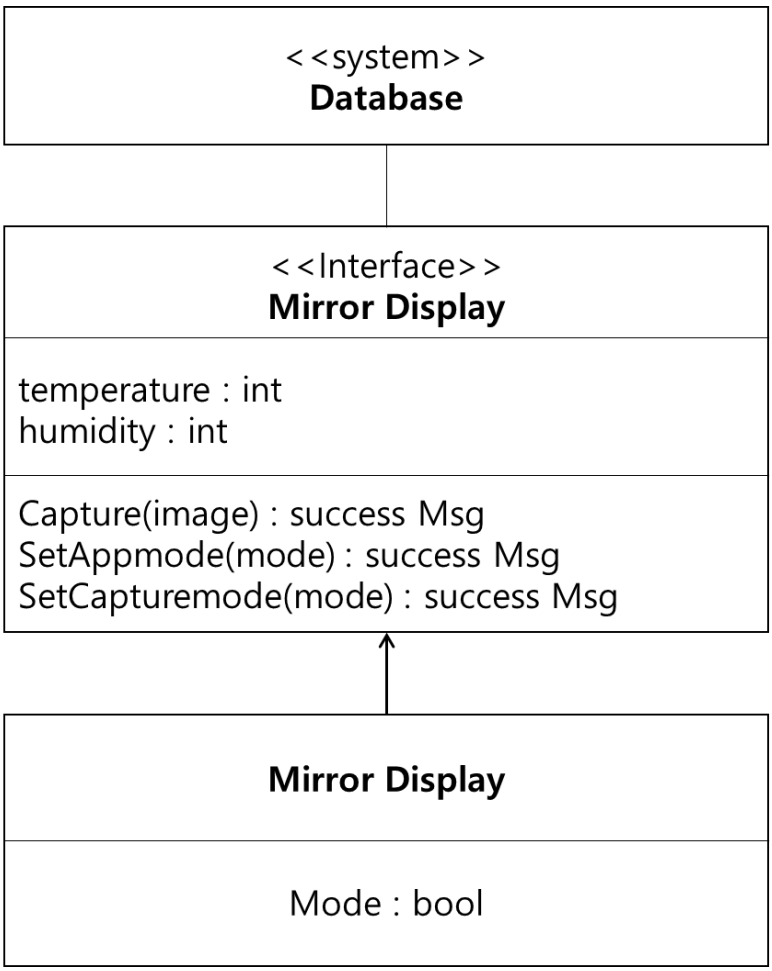
- Temperature : 온도를 값을 저장
- Humidity : 습도를 값을 저장

4.2.1.2 Methods

해당 Mirror Display가 가진 Method는 다음과 같다.

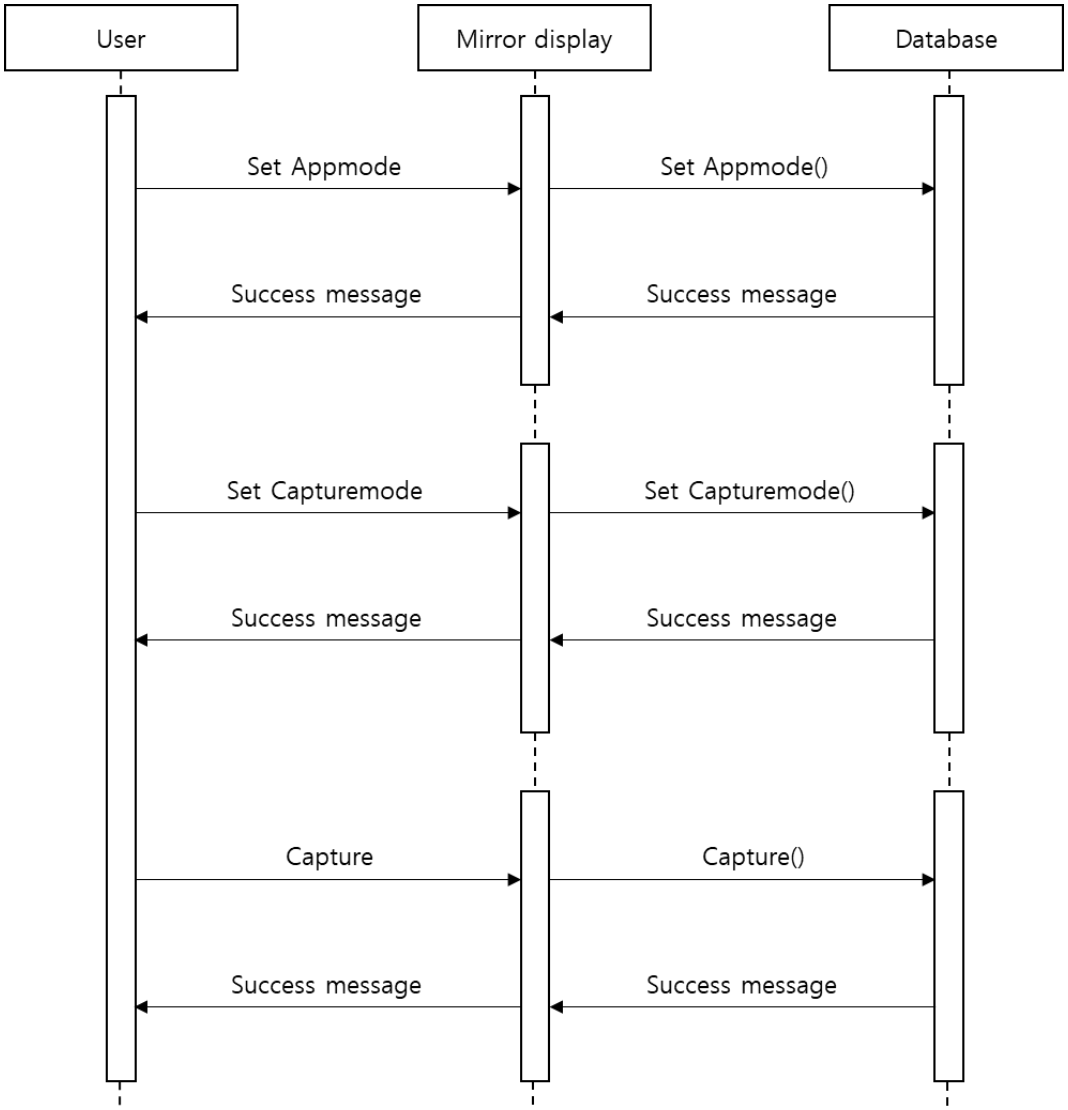
- Capture()
- SetAppmode()
- SetCapturemode()

4.2.1.3 Class Diagram



[Figure 8] Class Diagram – Mirror Display

4.2.1.4 Sequence Diagram



[Figure 9] Sequence Diagram – Mirror Display

4.2.2 로그인 및 회원가입

4.2.2.1 Attributes

해당 profile이 가진 attribute는 다음과 같다.

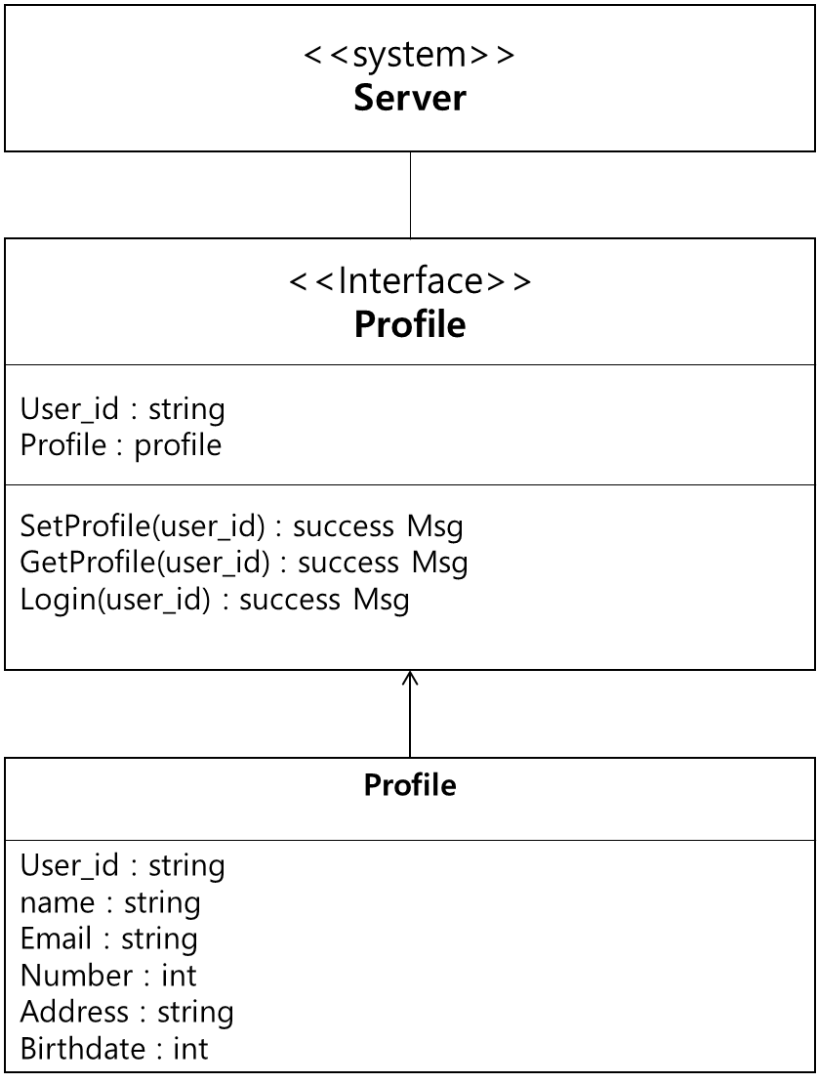
- User_id : 로그인시 사용되는 사용자의 id 이다.
- Name : 사용자의 본명이다.
- email : 사용자의 email 이다.
- number : 사용자의 연락처이다.
- address : 사용자의 주소이다.
- birthdate : 사용자의 생년월일이다.

4.2.2.2 Methods

해당 profile 이 가진 Method는 다음과 같다.

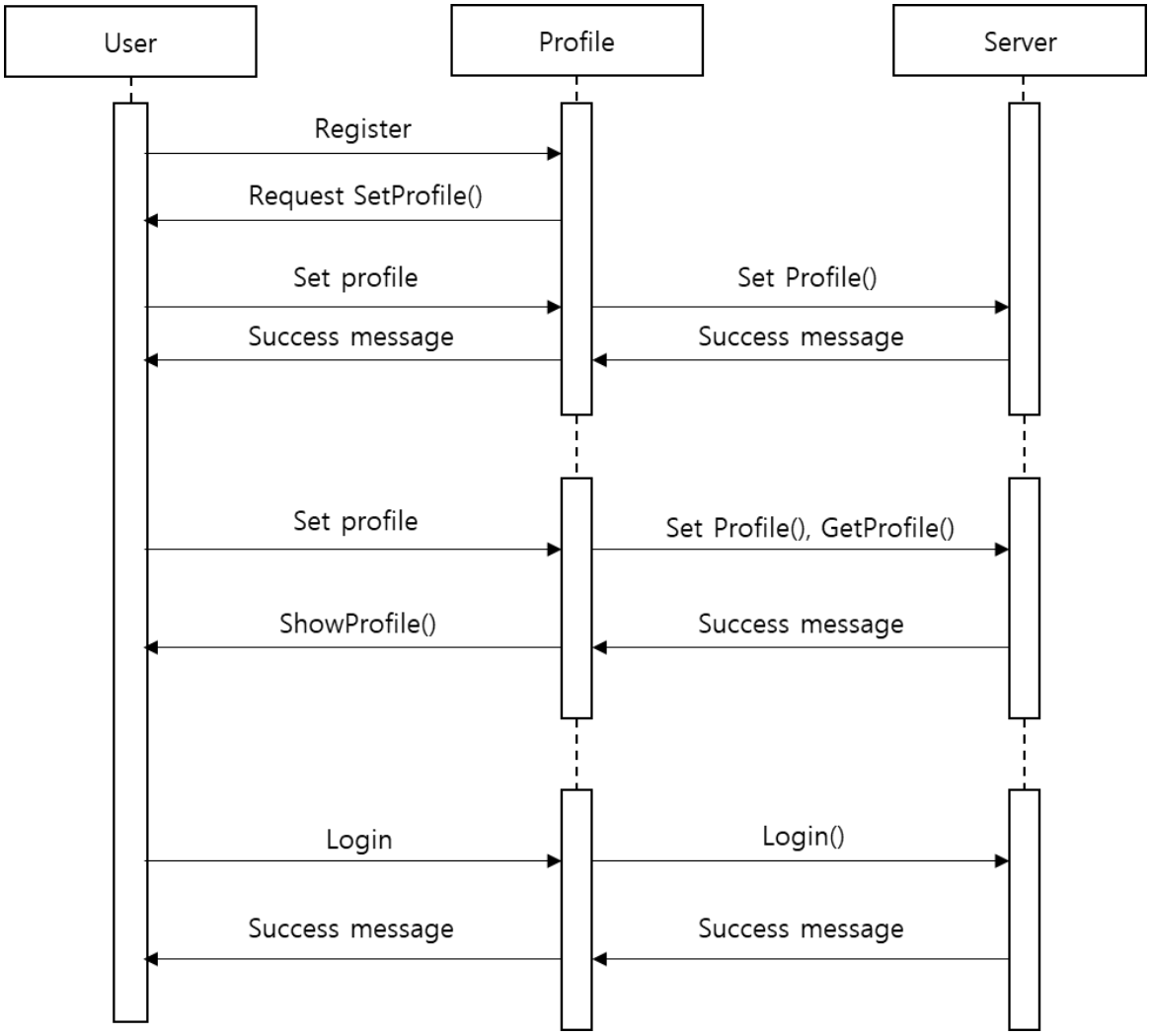
- SetProfile()
- GetProfile()
- Login()

4.2.2.3 Class diagram



[Figure 10] Class Diagram – Login & Register

4.2.2.4 Sequence diagram



[Figure 11] Sequence Diagram – Login & Register

4.2.3 장치 찾기

4.2.3.1 Attributes

해당 device가 가진 Attribute는 다음과 같다.

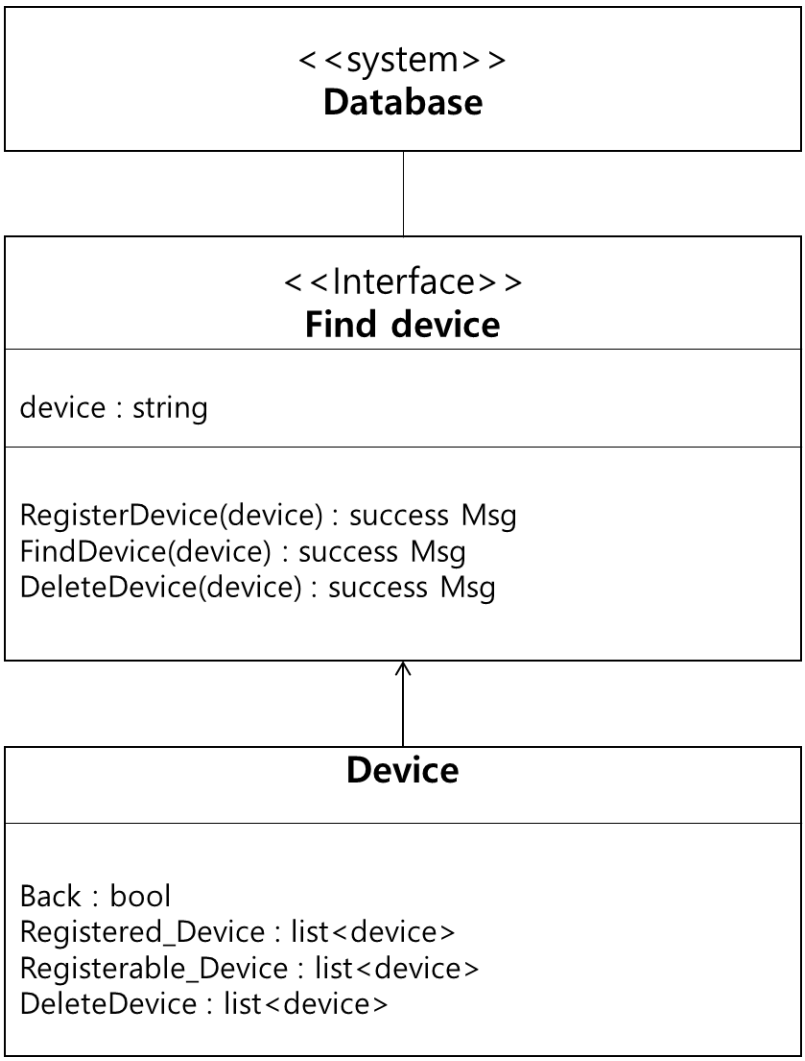
- Back : 뒤로가기가 눌렸는지 확인한다.
- Registered_Device : 등록된 Device 목록이다.
- Registerable_Device : 등록할 수 있는 Device 목록이다.
- DeleteDevice : 지우는 Device 목록이다.

4.2.3.2 Methods

해당 Device가 가진 Method는 다음과 같다.

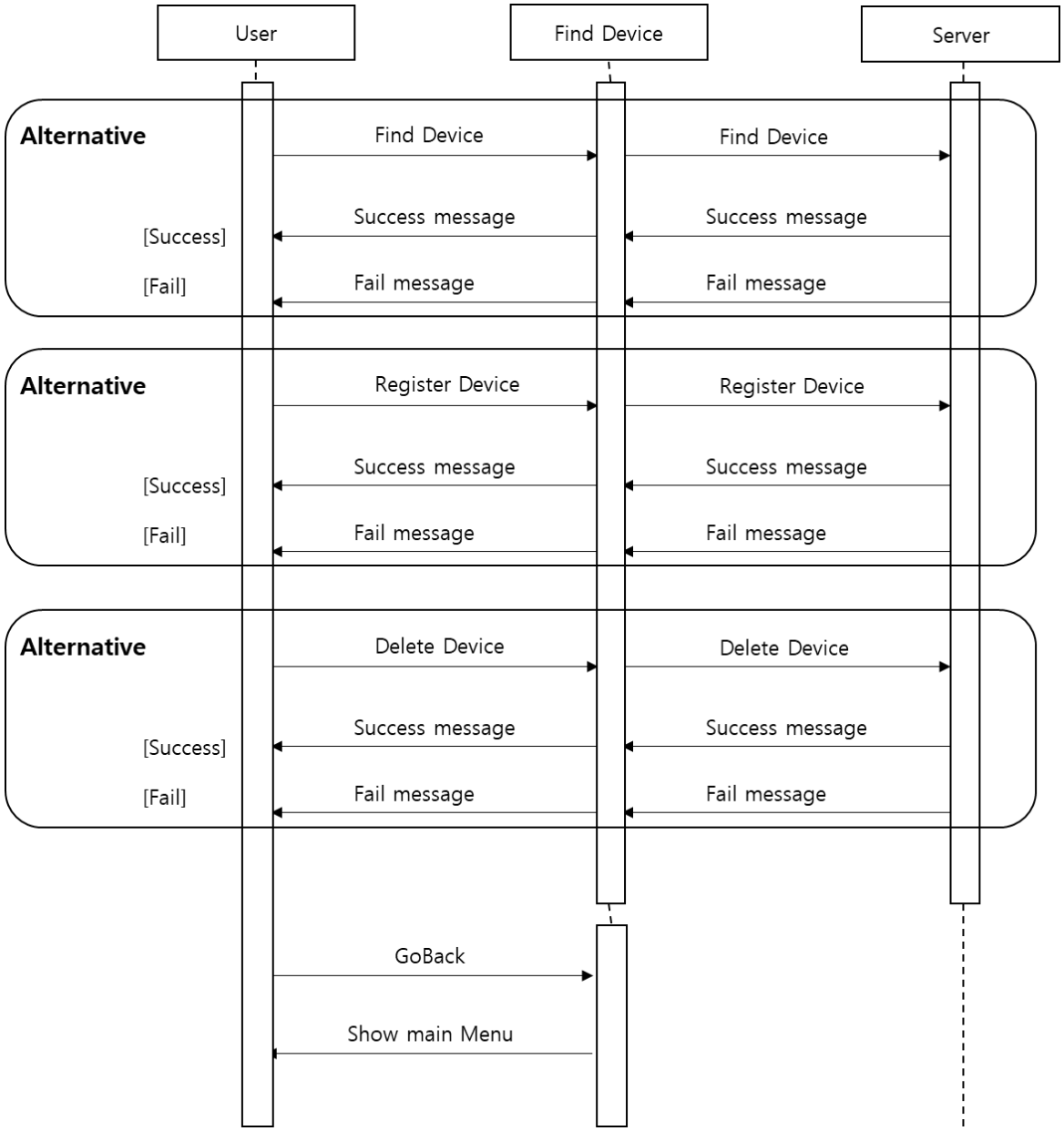
- RegisterDevice()
- FindDevice()
- DeleteDevice()

4.2.3.3. Class diagram



[Figure 12] Class Diagram – Find Devices

4.2.3.4. Sequence diagram



[Figure 13] Sequence Diagram – Find Devices

4.2.4. 메인 화면

4.2.4.1. Attributes

해당 Main Menu가 가지는 Attribute는 다음과 같다.

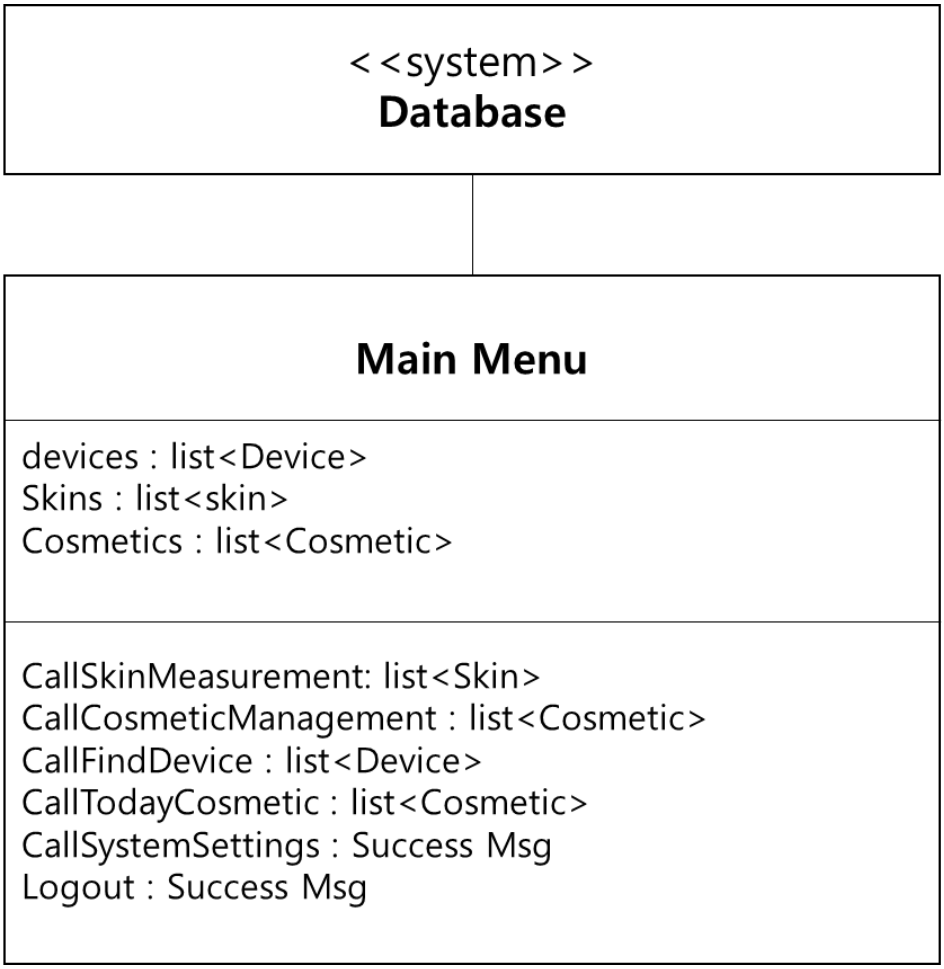
- SkinMeasurement button : 피부측정 페이지로 이동하는 버튼
- CosmeticManagement button : 화장품 관리 페이지로 이동하는 버튼
- FindDevice button : 장치 찾기 페이지로 이동하는 버튼
- TodayCosmetic button : 오늘의 화장품 페이지로 이동하는 버튼
- SystemSettings button : 설정으로 이동하는 버튼
- Logout button : 로그아웃 하는 버튼

4.2.4.2 Methods

해당 Main Menu가 가지는 method는 다음과 같다.

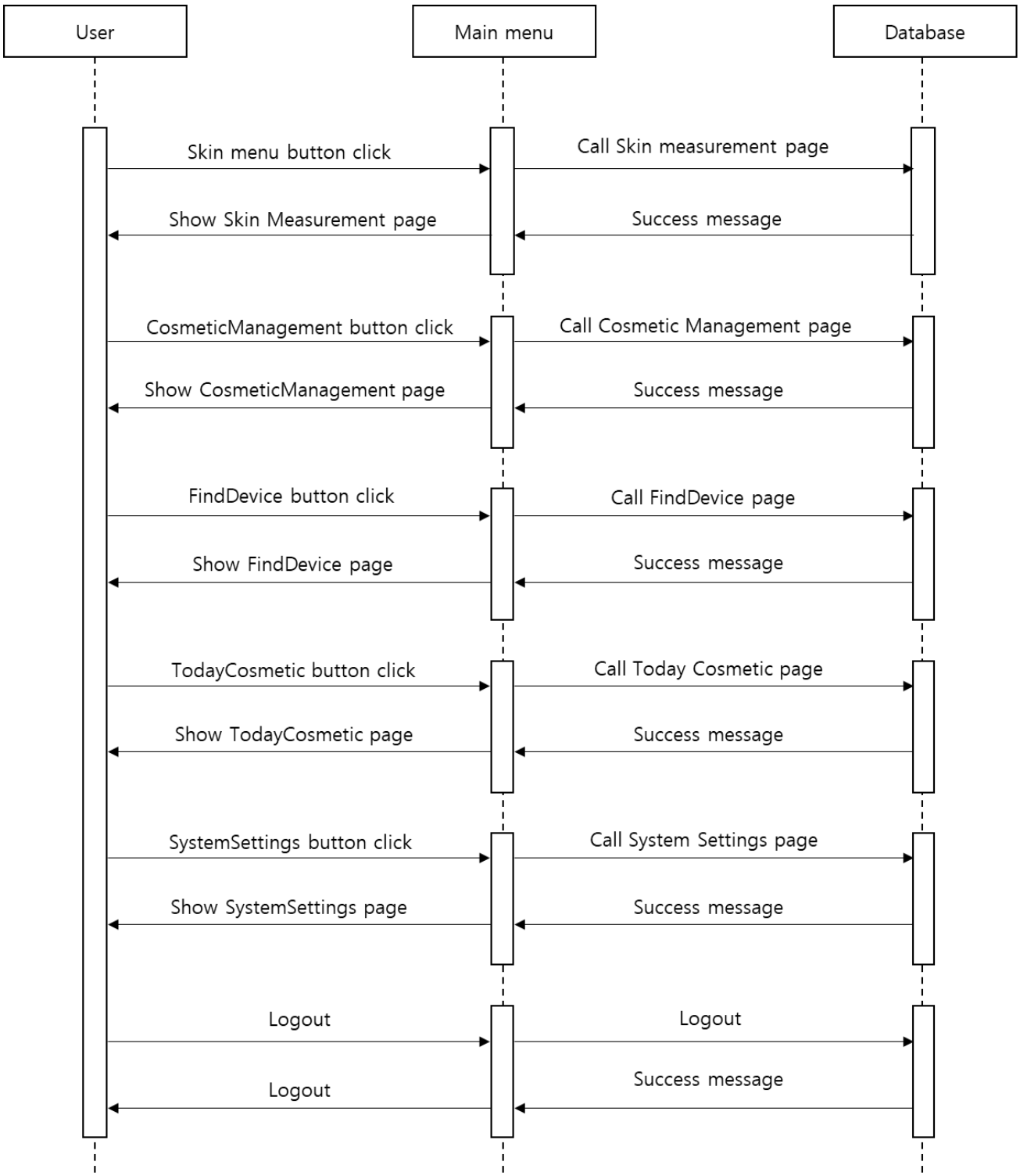
- CallSkinMeasurement
- CallCosmeticManagement
- CallFindDevice
- CallTodayCosmetic
- CallSystemSettings
- Logout

4.2.4.3. Class diagram



[Figure 14] Class Diagram – Main Menu

4.2.4.4 Sequence diagram



[Figure 15] Sequence Diagram – Main Menu

4.2.5. 피부 상태 표시

4.2.5.1. Attributes

Skin Measurement가 가지는 Attribute는 다음과 같다.

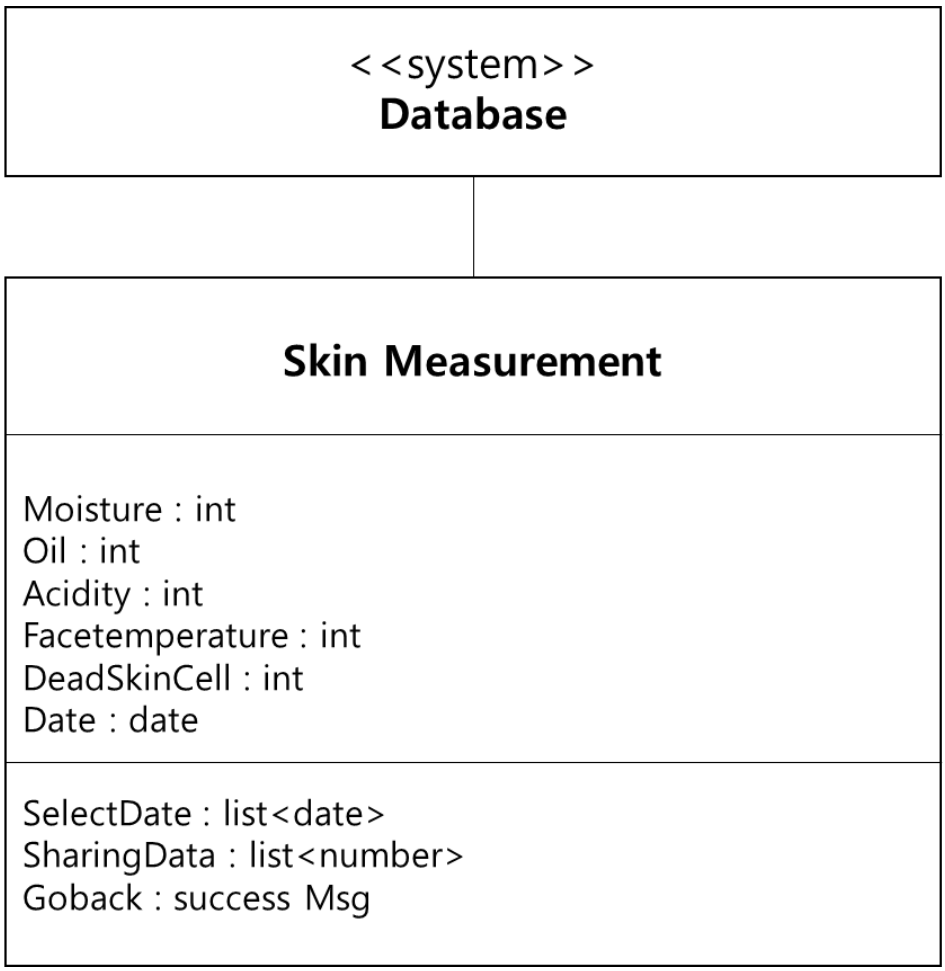
- Moisture : 사용자의 수분
- Oil : 사용자 얼굴의 유분
- Acidity : 사용자 얼굴의 산성도
- Facetemperature : 사용자 얼굴의 온도
- DeadSkinCell : 사용자의 각질
- Date : 사용자가 설정한 날짜

4.2.5.2. Methods

해당 Skin Measurement 가 가지는 Method는 다음과 같다.

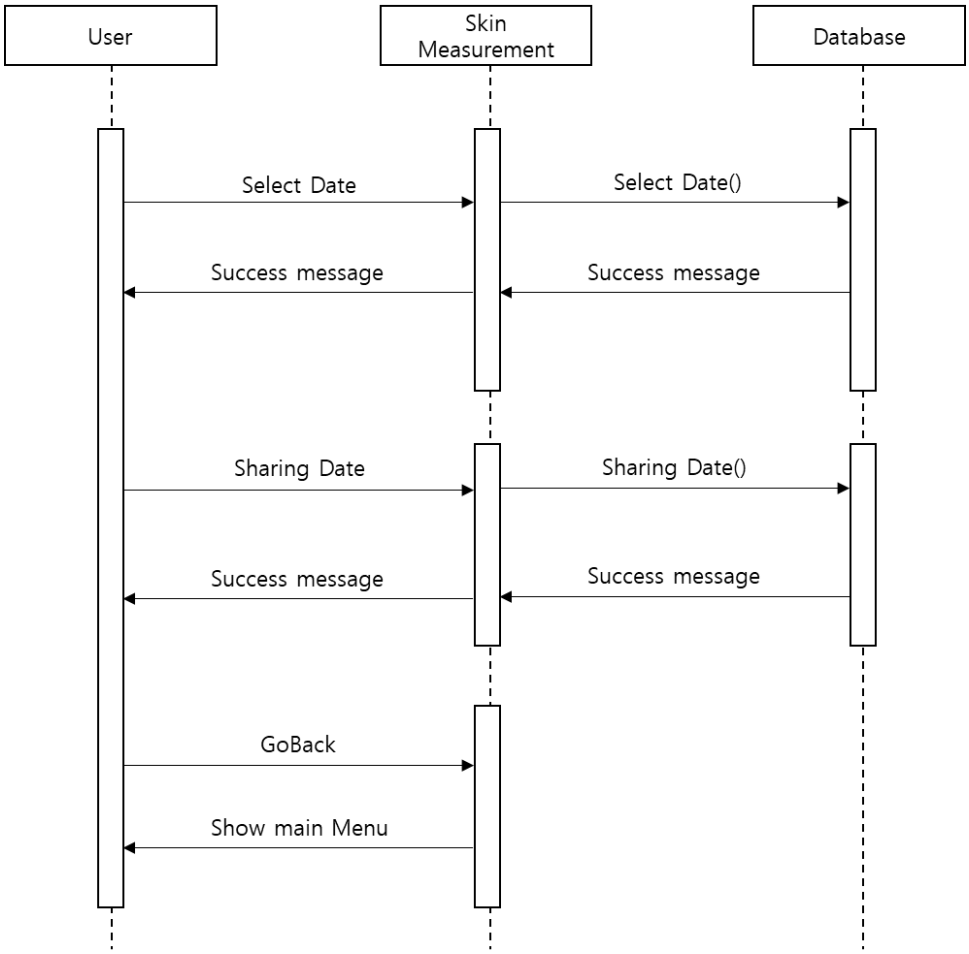
- SelecDate()
- SharingData()
- Goback()

4.2.5.3. Class diagram



[Figure 16] Class Diagram – Skin Measurement

4.2.5.4 Sequence diagram



[Figure 17] Sequence Diagram – Skin Measurement

4.2.6. 화장품 관리

4.2.6.1. Attributes

Cosmetic Manager Singleton 인스턴스가 가지는 attribute는 다음과 같다.

- ownedCosm_id: 화장품 ID
- ownedCosm_name: 화장품 이름
- openDate: 개봉 일자
- expDate: 유통 기한
- cosmInfo: 권장 보관 온도, 습도

Cosmetic 객체가 가지는 attribute는 다음과 같다.

- ownedCosm_name: 화장품 이름
- openDate: 개봉 일자
- expDate: 유통 기한

4.2.6.2. Methods

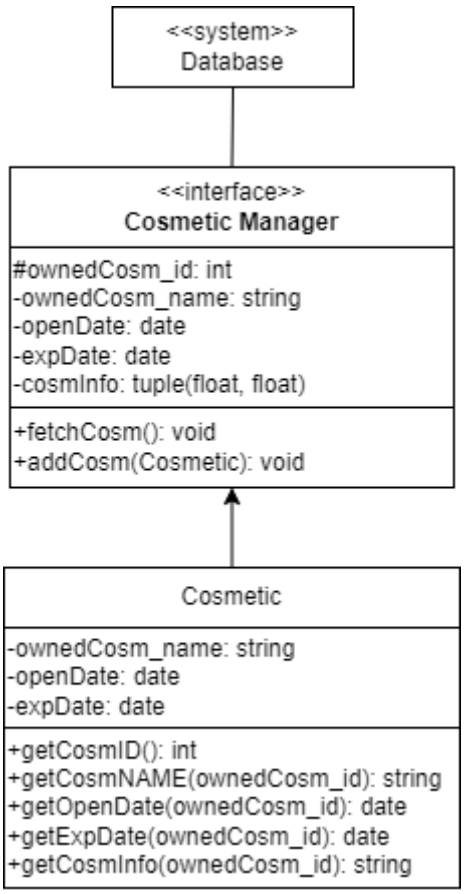
Cosmetic Manager class가 가지는 method는 다음과 같다.

- fetchCosm()
- addCosm(Cosmetic)

Cosmetic class가 가지는 method는 다음과 같다.

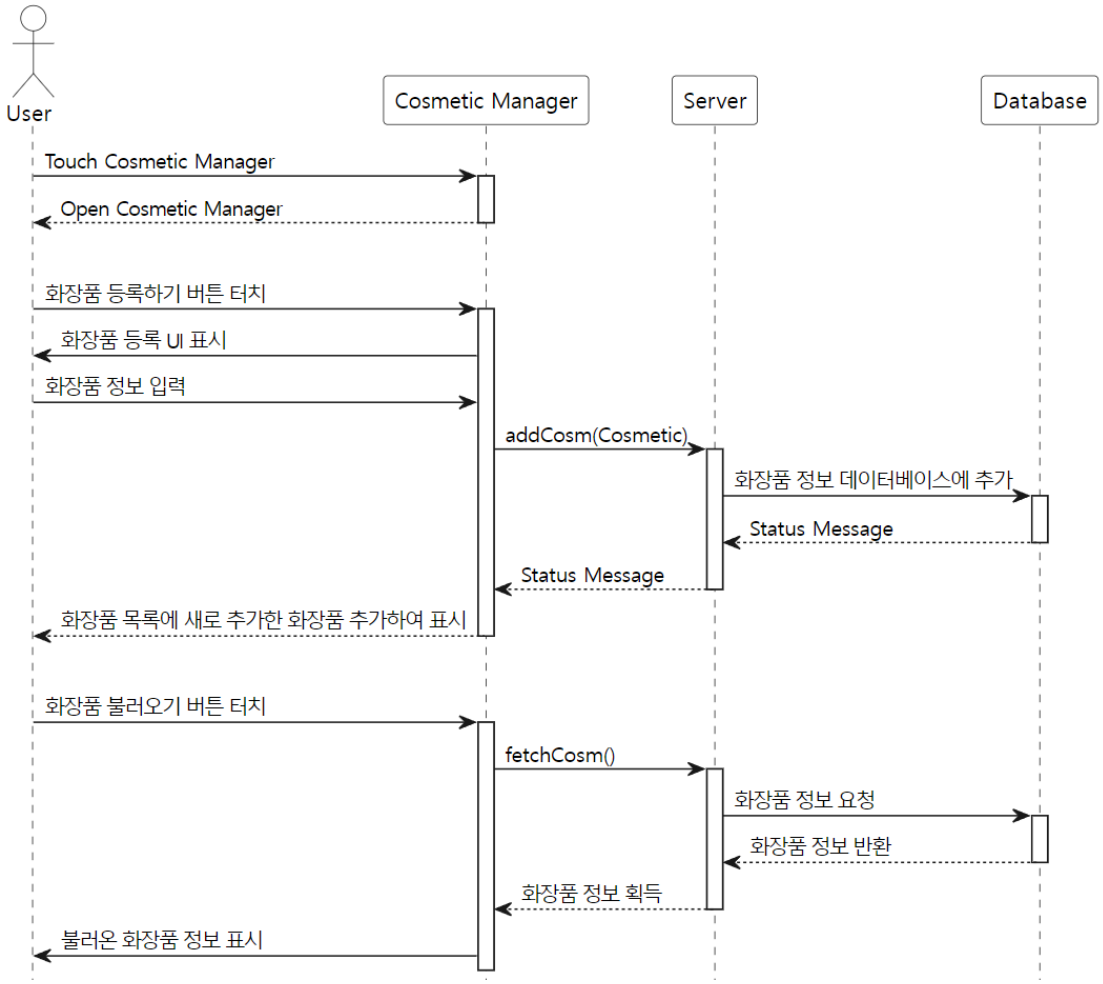
- getCosmID()
- getCosmNAME(int)
- getOpenDate(int)
- getExpDate(int)
- getCosmInfo(int)

4.2.6.3 Class Diagram



[Figure 18] Class Diagram – Cosmetic Manager

4.2.6.4 Sequence Diagram



[Figure 19] Sequence Diagram – Cosmetic Manager

4.2.7. 오늘의 화장품

4.2.7.1 Attributes

COTD(Cosmetic of the Day) 객체가 가지는 attribute는 다음과 같다.

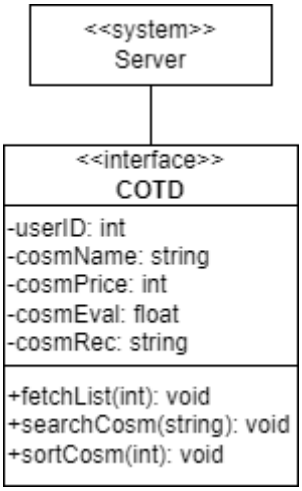
- userID: 유저 아이디
- cosmName: 화장품 이름
- cosmPrice: 화장품 가격
- cosmEval: 피부적합도
- cosmRec: 추천 사유

4.2.7.2 Methods

COTD(Cosmetic of the Day) class가 가지는 method는 다음과 같다.

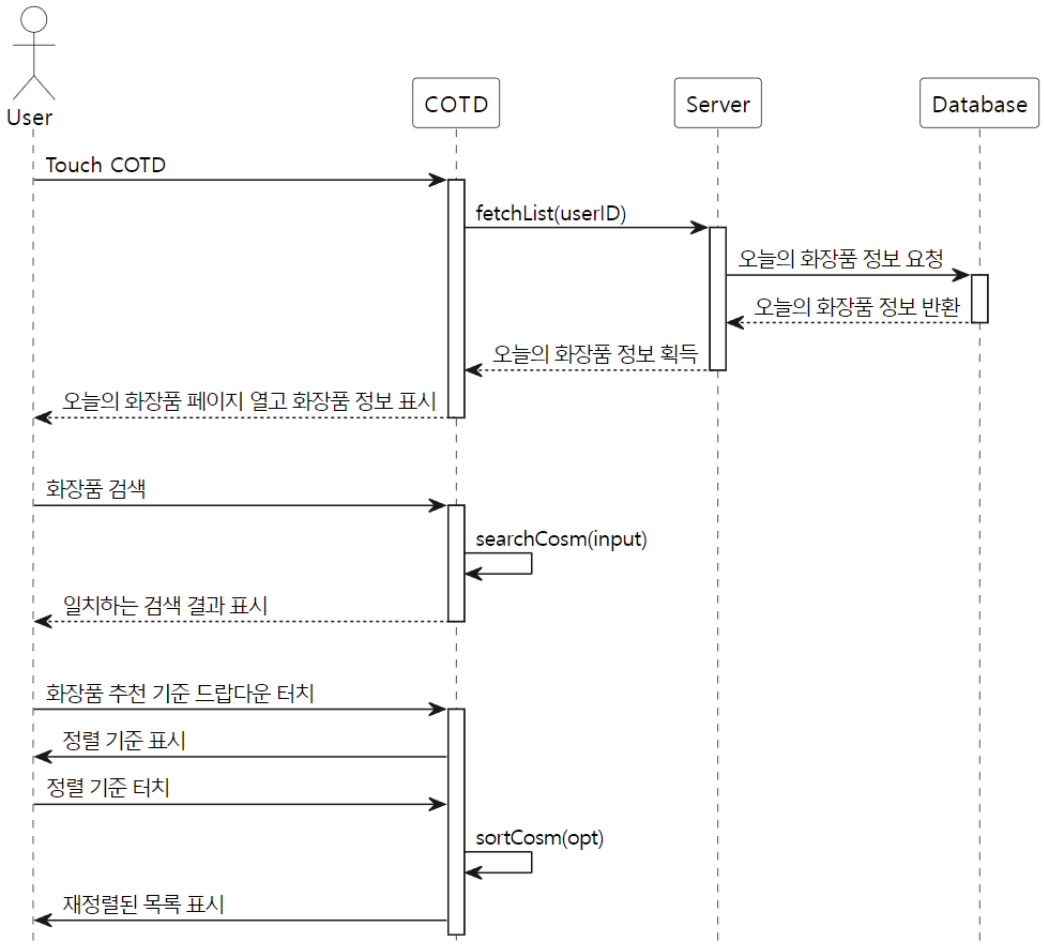
- fetchList(int)
- searchCosm(string)
- sortCosm(int)

4.2.7.3 Class Diagram



[Figure 20] Class Diagram – Cosmetic of the Day

4.2.7.4 Sequence Diagram



[Figure 21] Sequence Diagram – Cosmetic of the Day

4.2.8 가상 메이크업

4.2.8.1 Attributes

가상 메이크업 옵션 객체가 가지는 attribute는 다음과 같다.

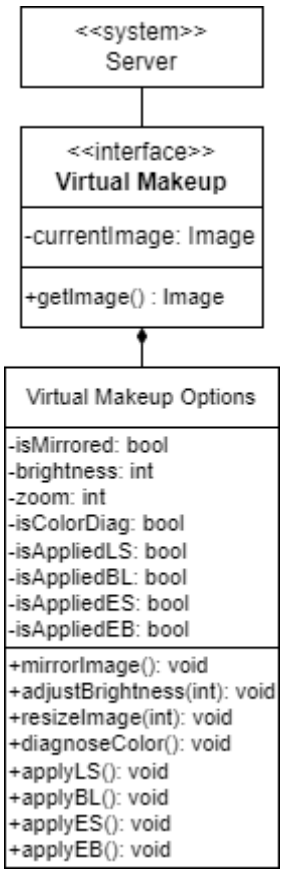
- currentImage(Image): 현재 선택된 이미지
- isMirrored(bool): 좌우반전 여부
- brightness(int): 이미지의 밝기를 나타내는 수치
- zoom(int): 이미지가 얼마나 확대되었는지 나타내는 수치
- isColorDiag(bool): 퍼스널 컬러 진단
- isAppliedLS(bool): 립스틱 적용 여부
- isAppliedBL(bool): 블러셔 적용 여부
- isAppliedES(bool): 아이섀도 적용 여부
- isAppliedEB(bool): 아이브로우 적용 여부

4.2.8.2 Methods

가상 메이크업 옵션 class가 가지는 method는 다음과 같다.

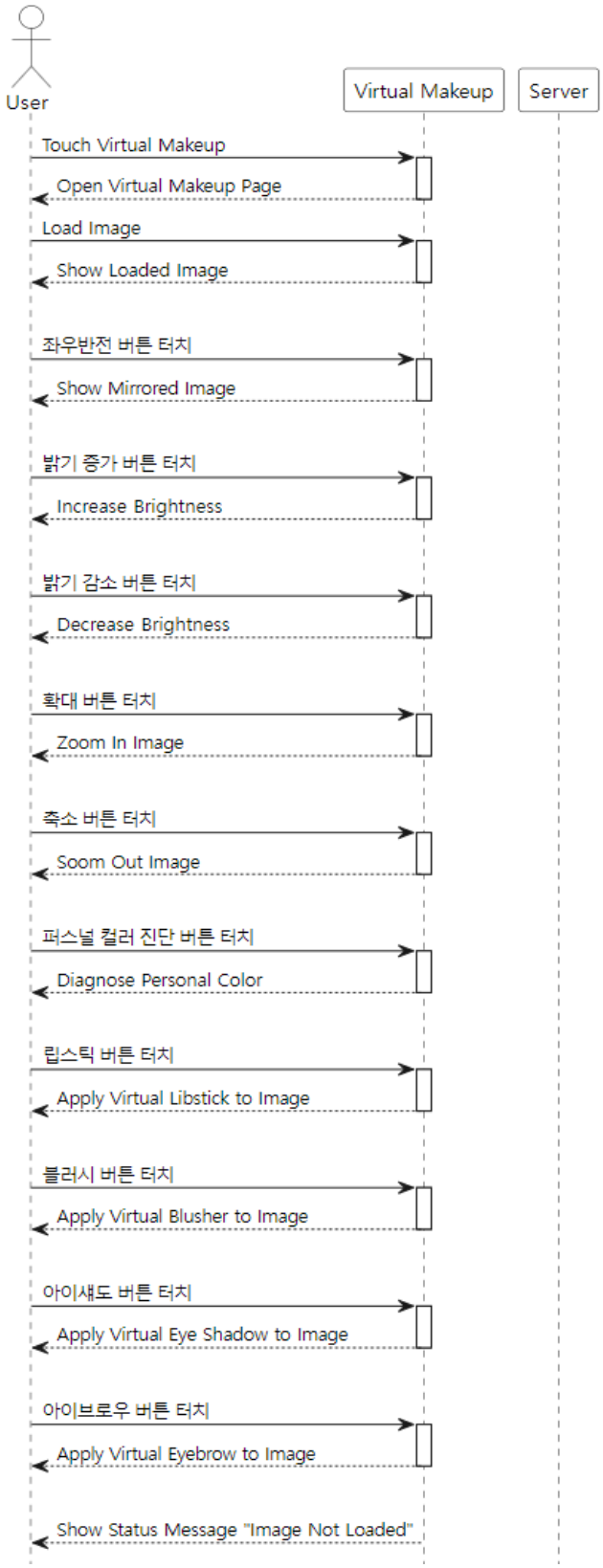
- getImage()
- mirrorImage()
- adjustBrightness()
- resizeImage()
- diagnoseColor()
- applyLS()
- applyBL()
- applyES()
- applyEB()

4.2.8.3 Class Diagram



[Figure 22] Class Diagram – Virtual Makeup

4.2.8.4 Sequence Diagram



[Figure 23] Sequence Diagram – Virtual Makeup

4.2.9 설정 페이지

4.2.9.1 Attributes

설정 페이지 Singleton 인스턴스가 가지고 있는 attribute는 다음과 같다.

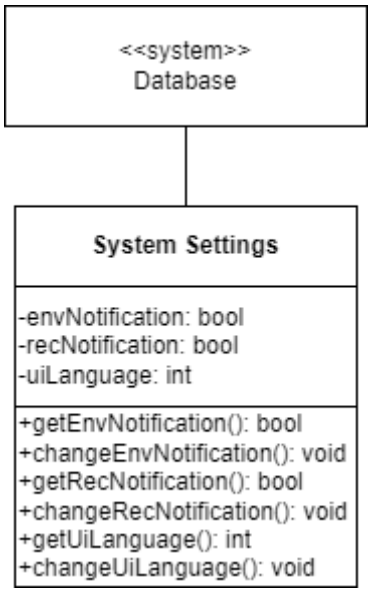
- envNotification: 화장대 온습도 경고 푸쉬 알림 허용 여부
- recNotification: 화장품 추천 푸쉬 알림 허용 여부
- uiLanguage: 사용자 인터페이스 언어

4.2.9.2 Methods

설정 페이지 Class가 가지고 있는 method는 다음과 같다.

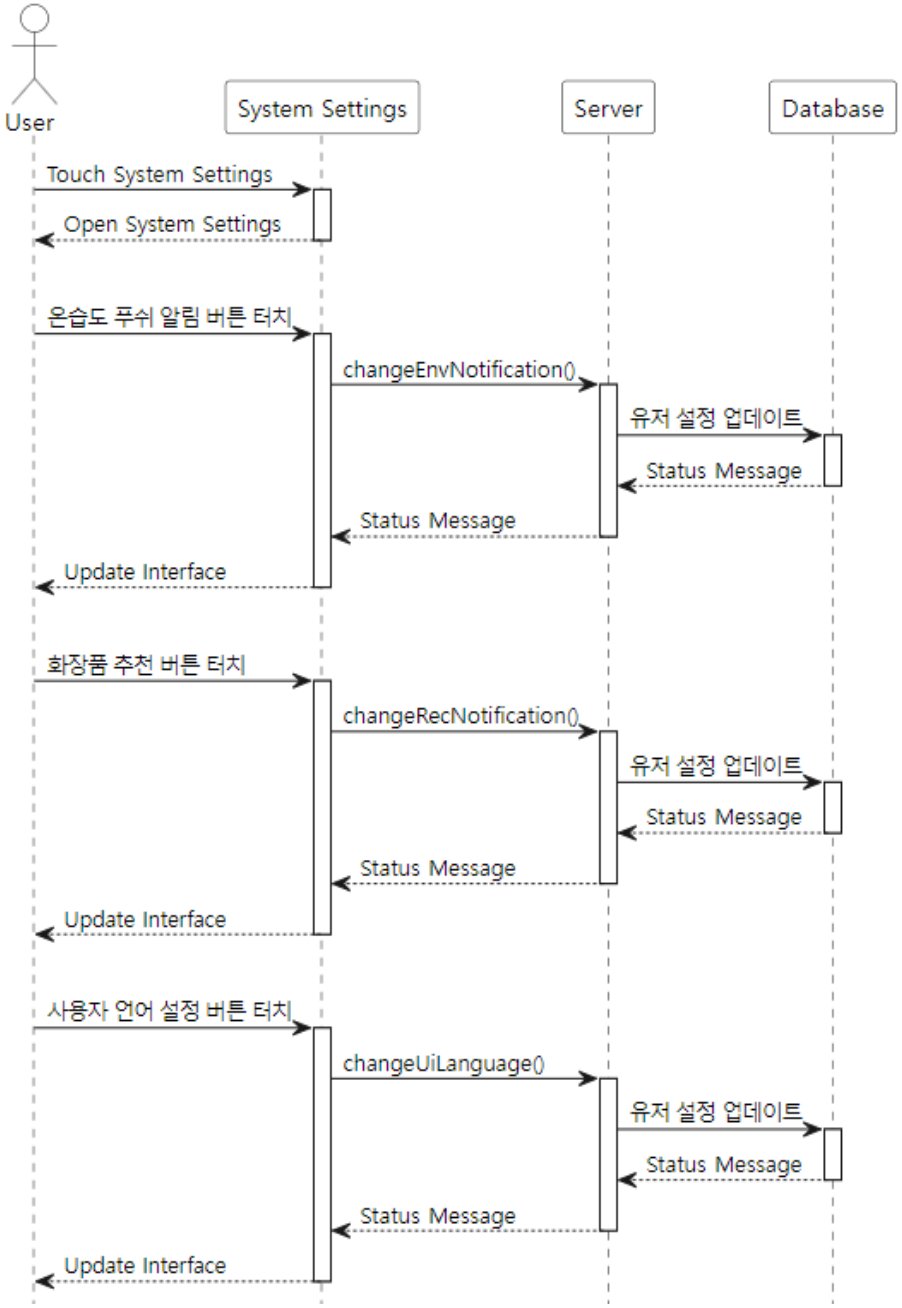
- getEnvNotification()
- changeEnvNotification()
- getRecNotification()
- changeRecNotification()
- getUiLanguage()
- changeUiLanguage()

4.2.9.3 Class Diagram



[Figure 24] Class Diagram – System Settings

4.2.9.4 Sequence Diagram



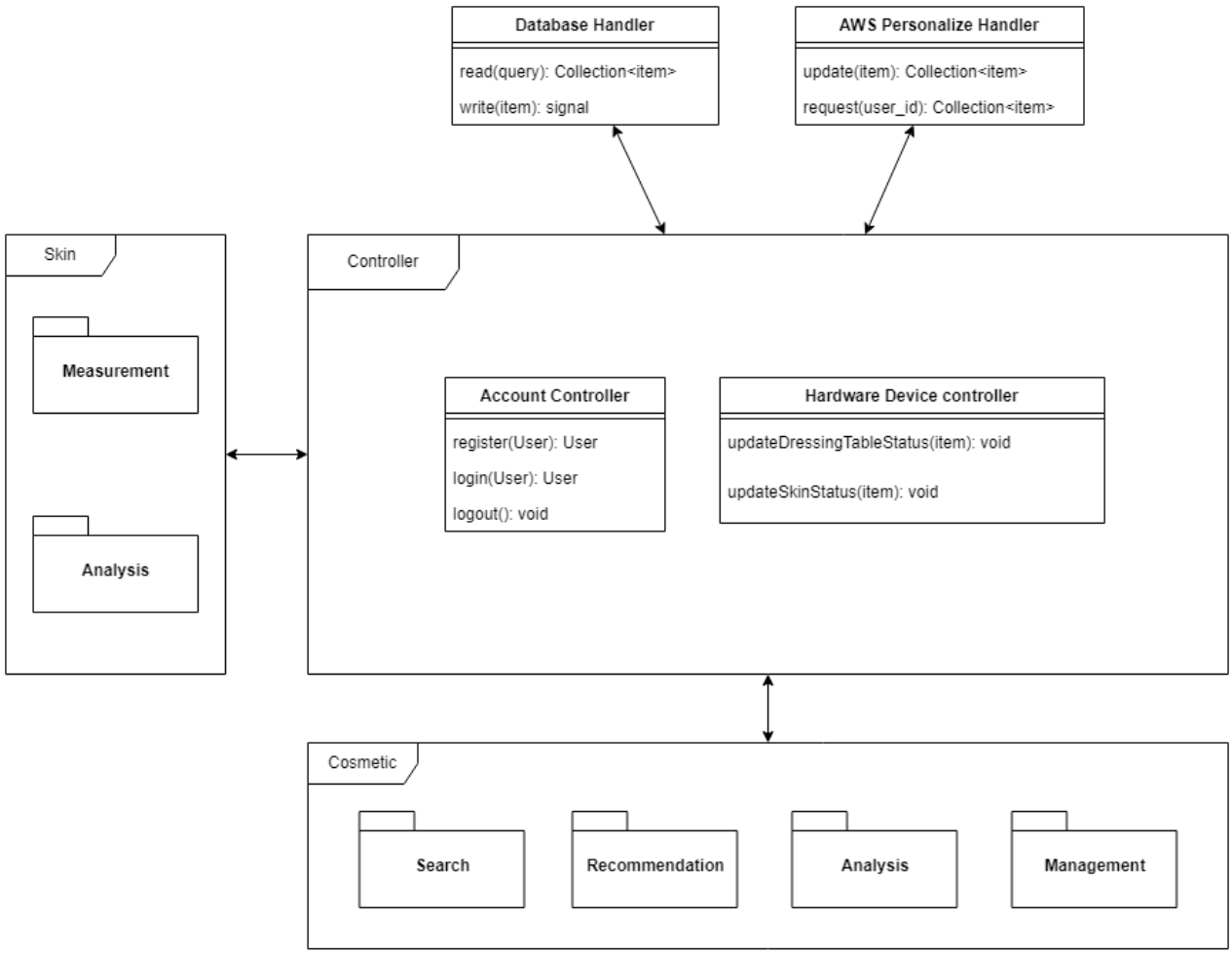
[Figure 25] Sequence Diagram – System Settings

5. System Architecture – Backend

5.1. Objectives

이 장에서는 backend 시스템의 구조와 기능에 대해서 설명한다

5.2. Overall Architecture



[Figure 26] Overall Architecture

5.2.1. System

각 시스템은 특화된 도메인으로 구분된다. Skin Measurement System(피부 측정 시스템), Skin Analysis(피부 상태 분석 시스템), Cosmetic Search System(화장품 검색 시스템), Cosmetic Recommendation System(화장품 추천 시스템), Cosmetic Analysis System(화장품 분석 시스템), Cosmetic Management System(화장품 관리 시스템)이 있다.

5.2.2. Handler

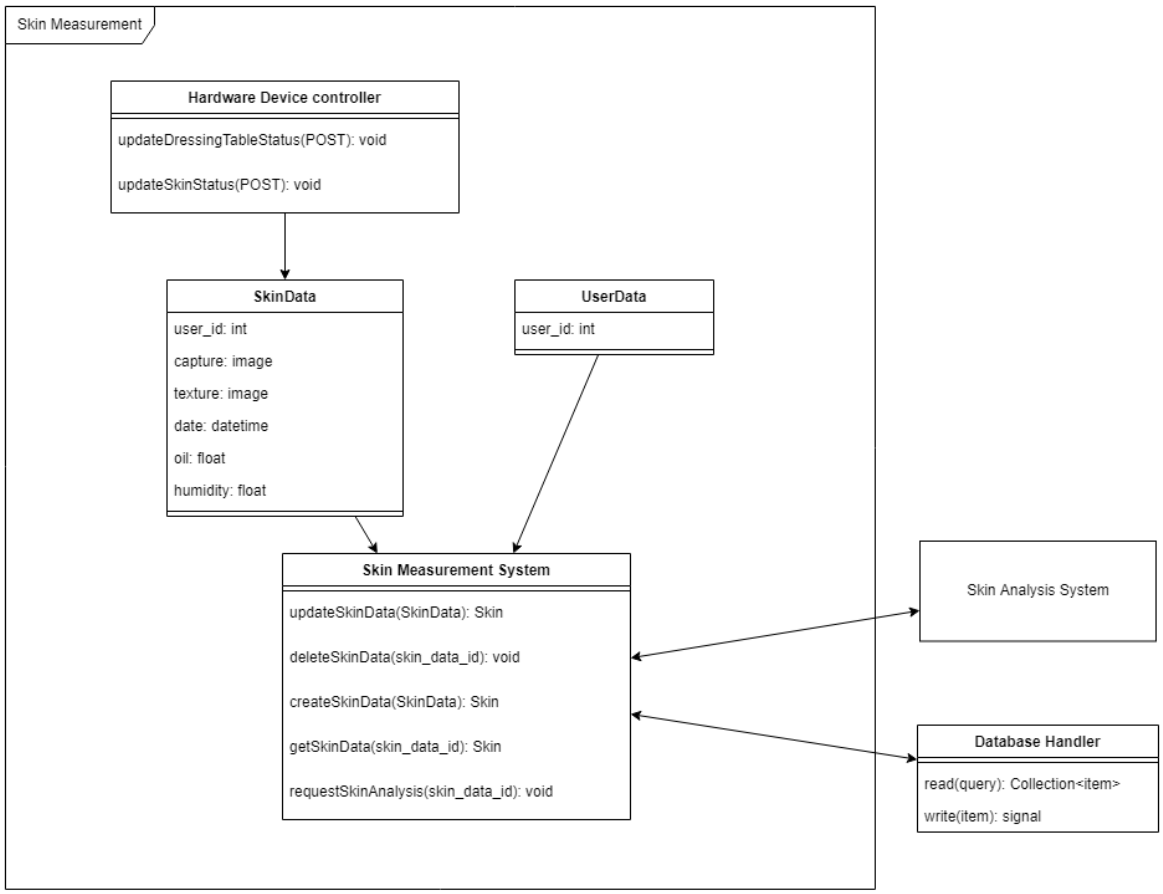
Handler는 Database Handler와 AWS Personalize Handler가 있다. Database Handler는 데이터베이스에서 데이터를 읽고 쓸 수 있도록 하는 인터페이스이고, AWS Personalize Handler는 완전 관리형 사용자 맞춤 추천 시스템인 AWS Personalize와 통신하여 추천 결과를 업데이트하고 요청하는 인터페이스이다.

5.2.3. Controller

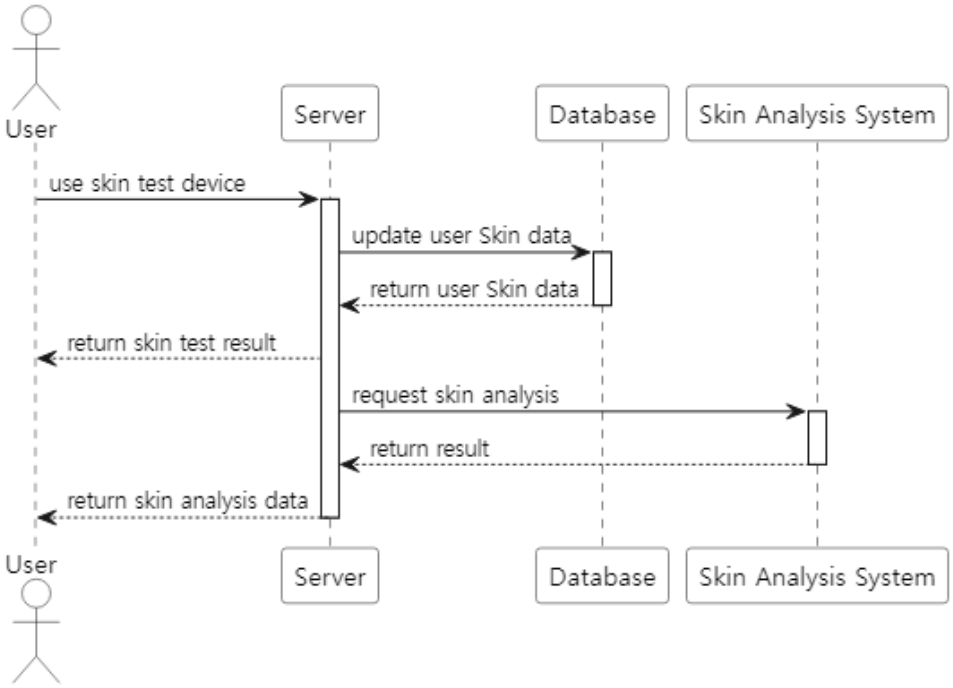
전체 시스템에서 공용으로 사용되는 Controller로는 사용자 계정을 관리하는 Account Controller와 화장대 및 피부 측정 장치의 데이터를 관리하는 Hardware Device Controller가 있다

5.3. Subcomponents

5.3.1. Skin Measurement System

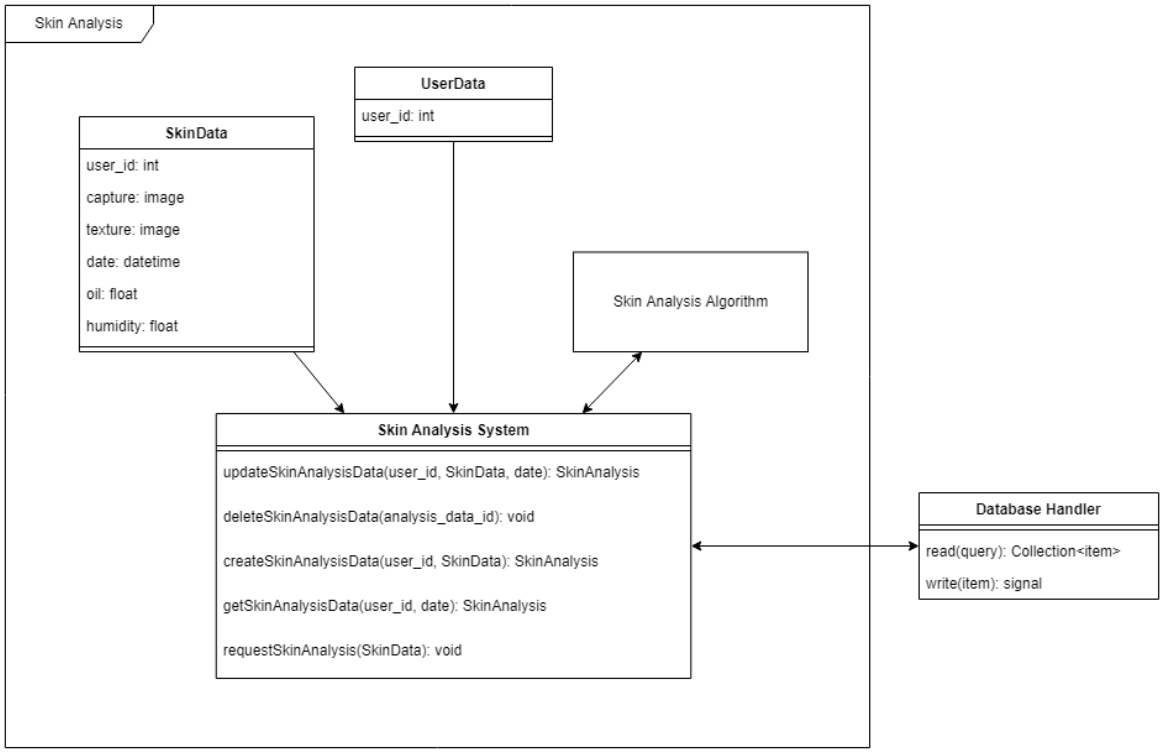


[Figure 27] Class Diagram - Skin Measurement System

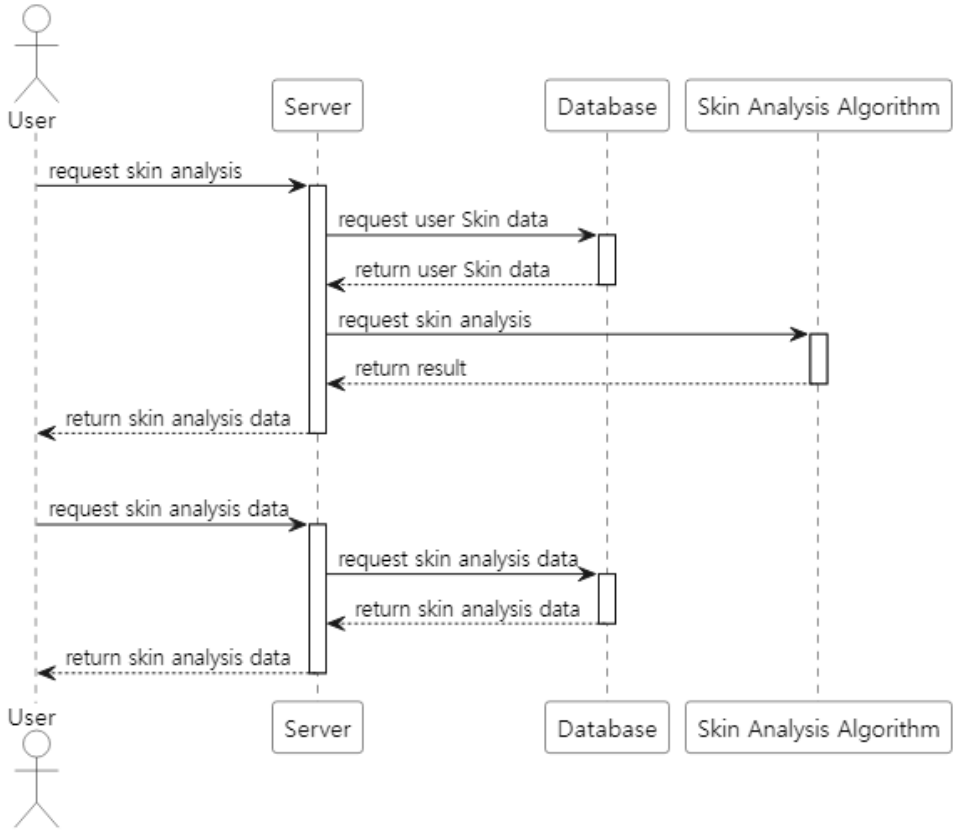


[Figure 28] Sequence Diagram - Skin Measurement System

5.3.2. Skin Analysis System

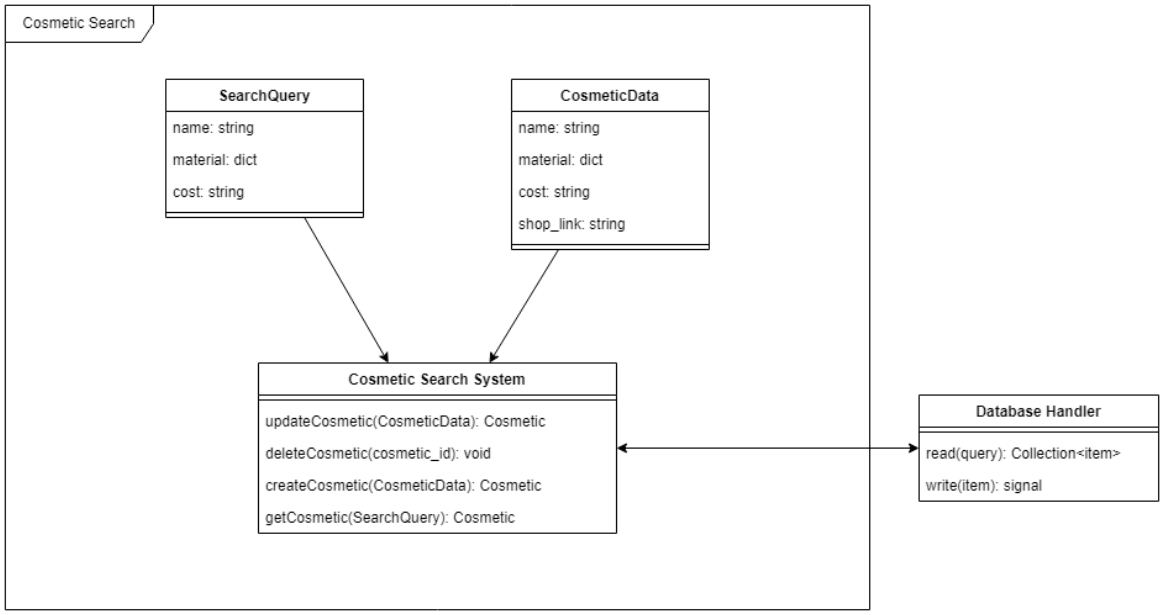


[Figure 29] Class Diagram - Skin Analysis System

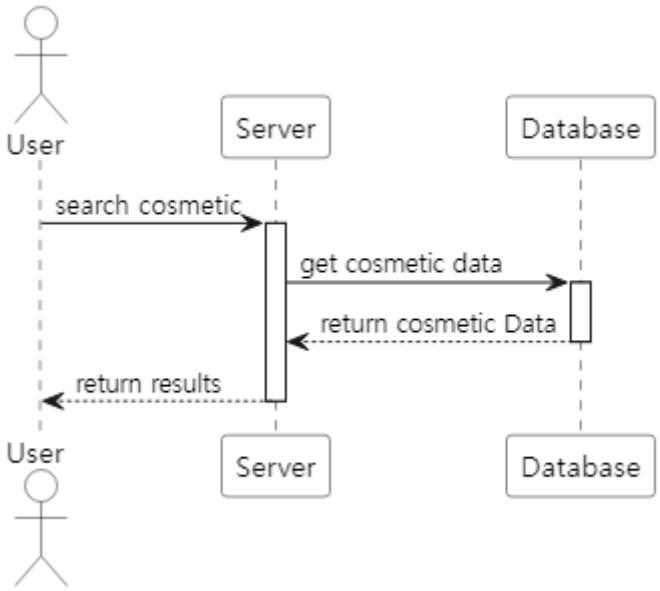


[Figure 30] Sequence Diagram - Skin Analysis System

5.3.3. Cosmetic Search System

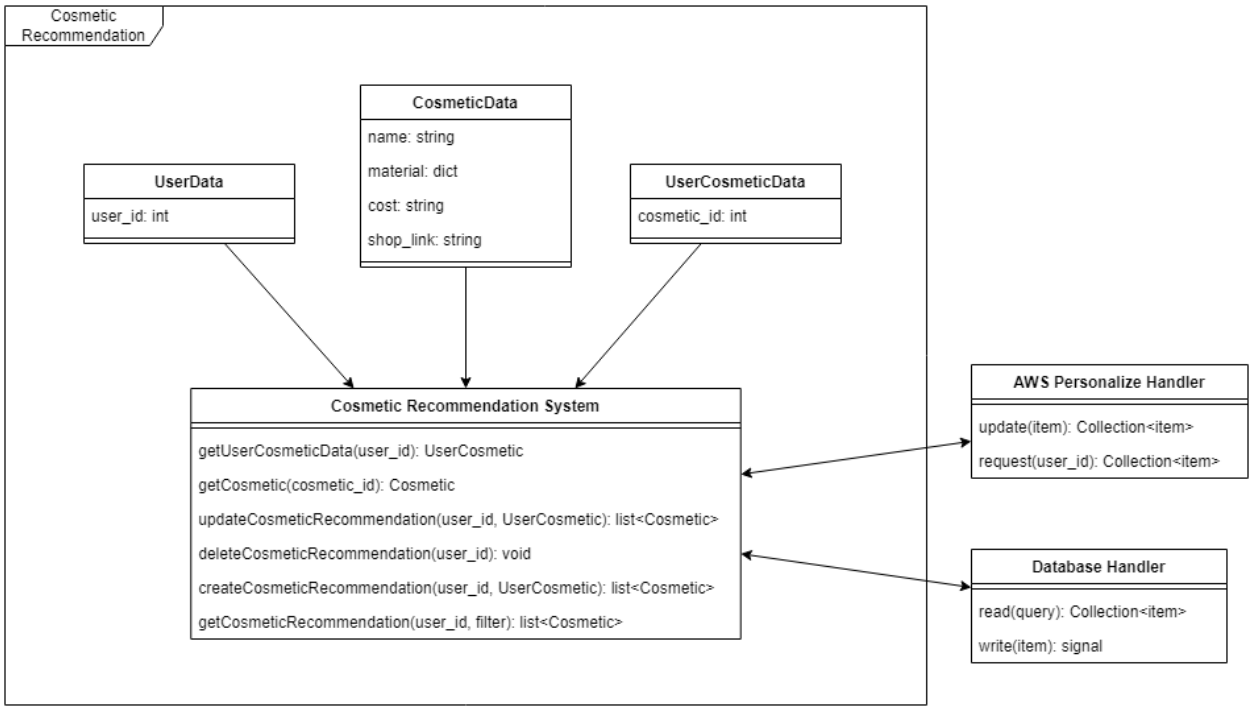


[Figure 31] Class Diagram – Cosmetic Search System

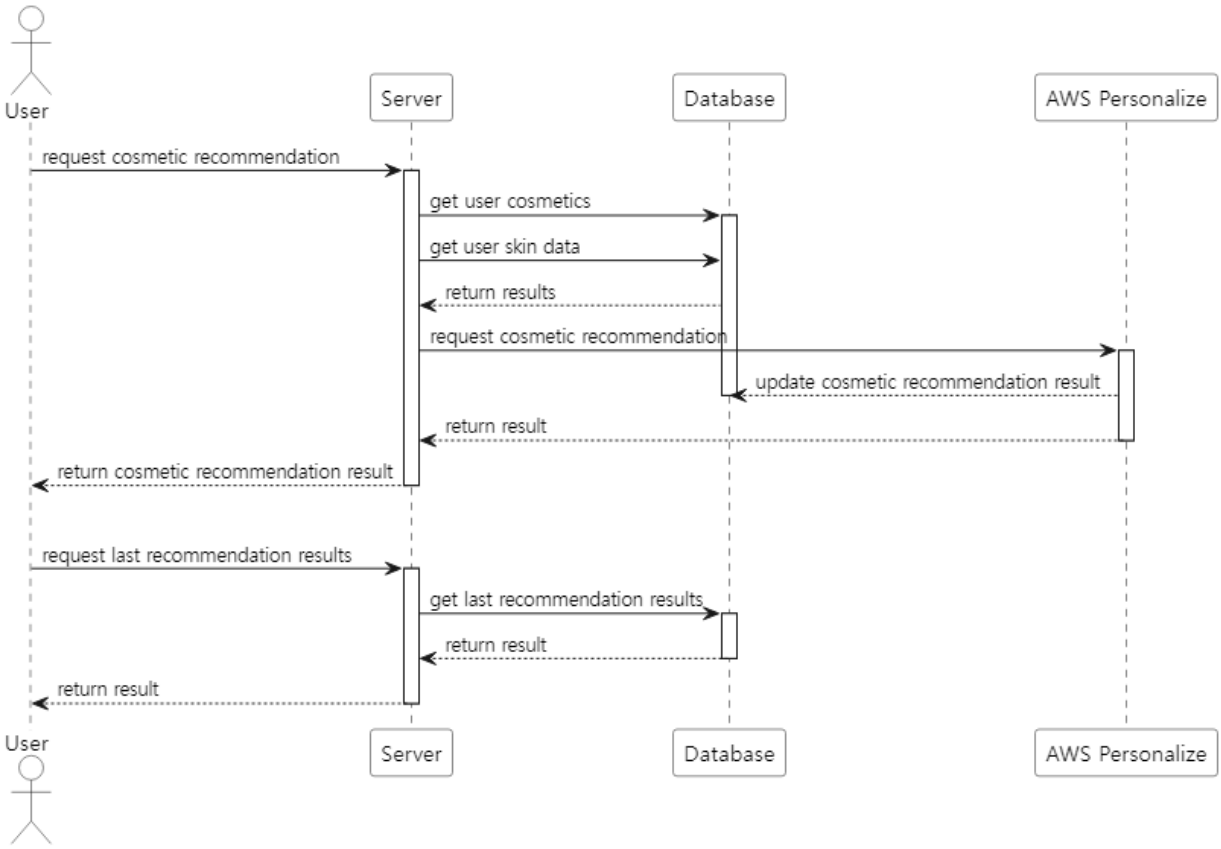


[Figure 32] Sequence Diagram – Cosmetic Search System

5.3.4. Cosmetic Recommendation System

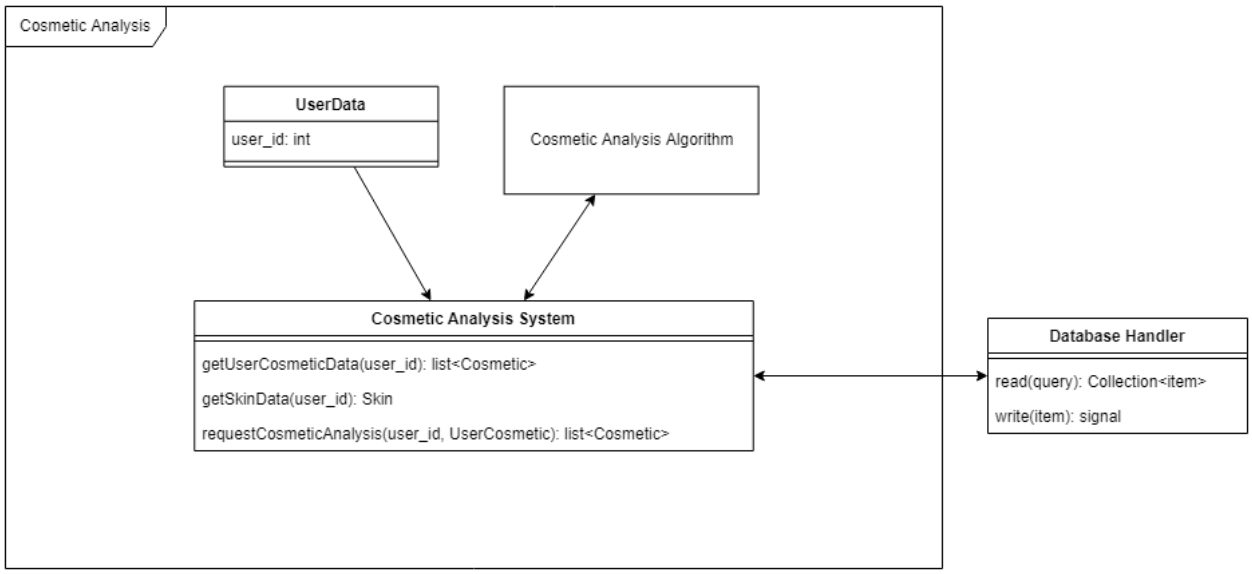


[Figure 33] Class Diagram – Cosmetic Recommendation System

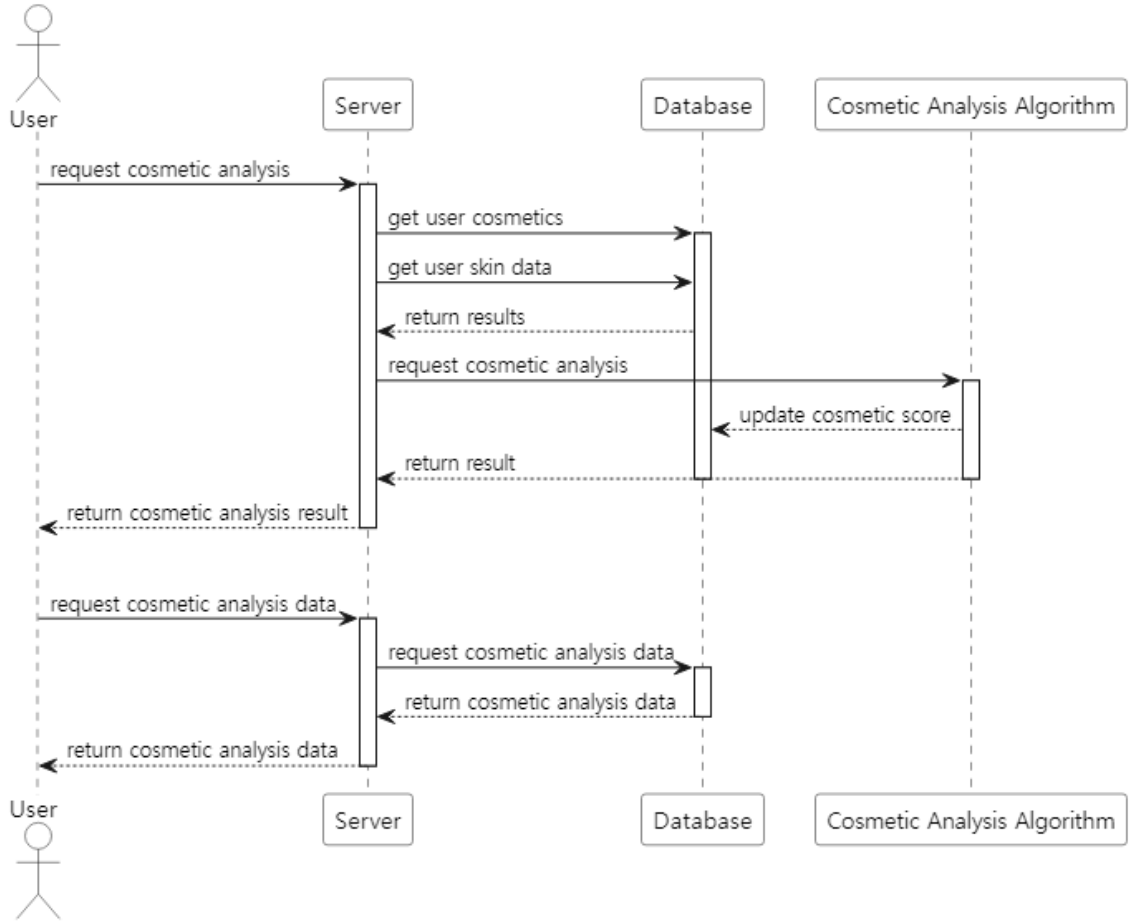


[Figure 34] Sequence Diagram – Cosmetic Recommendation System

5.3.5. Cosmetic Analysis System

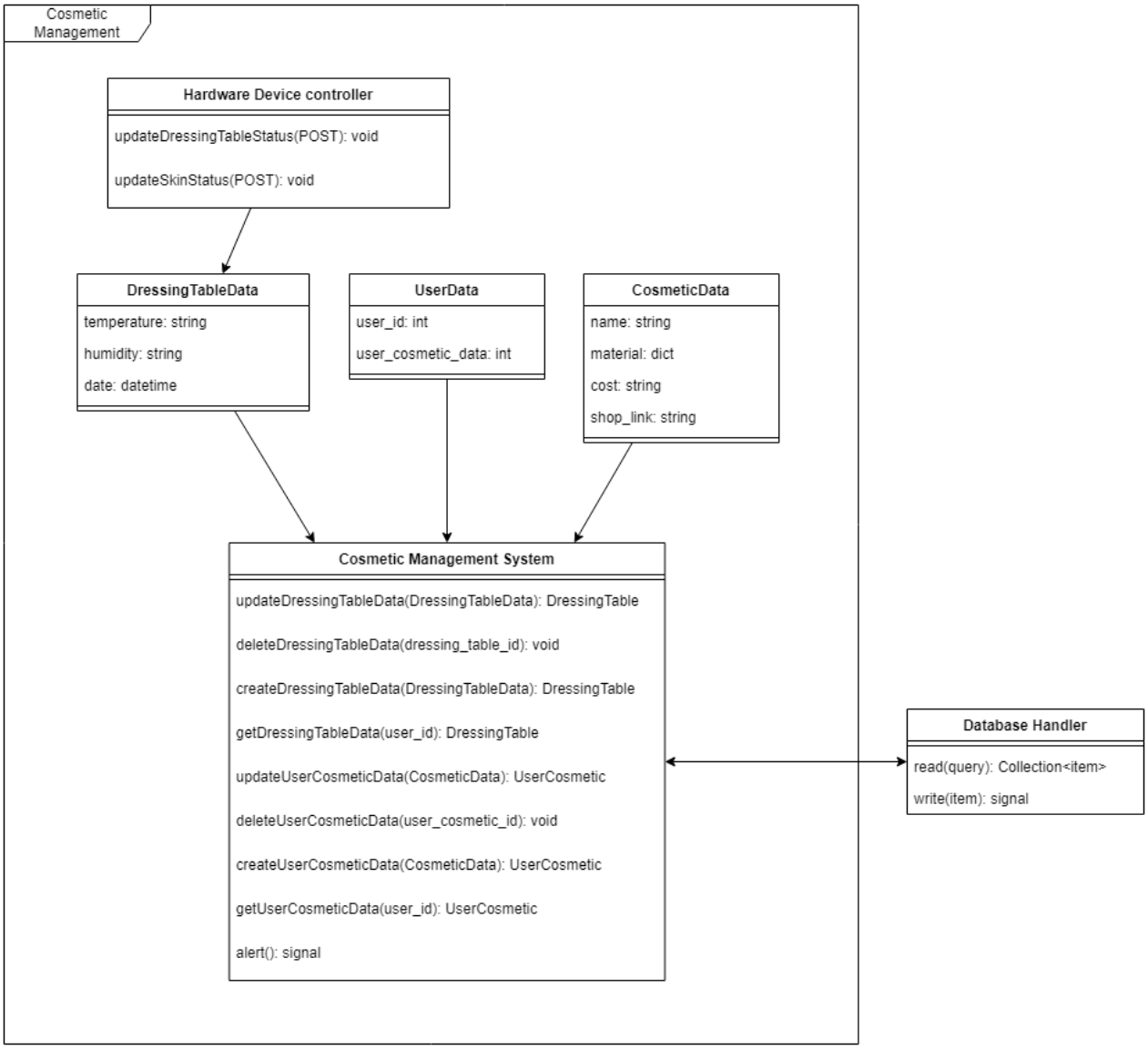


[Figure 35] Class Diagram – Cosmetic Analysis System

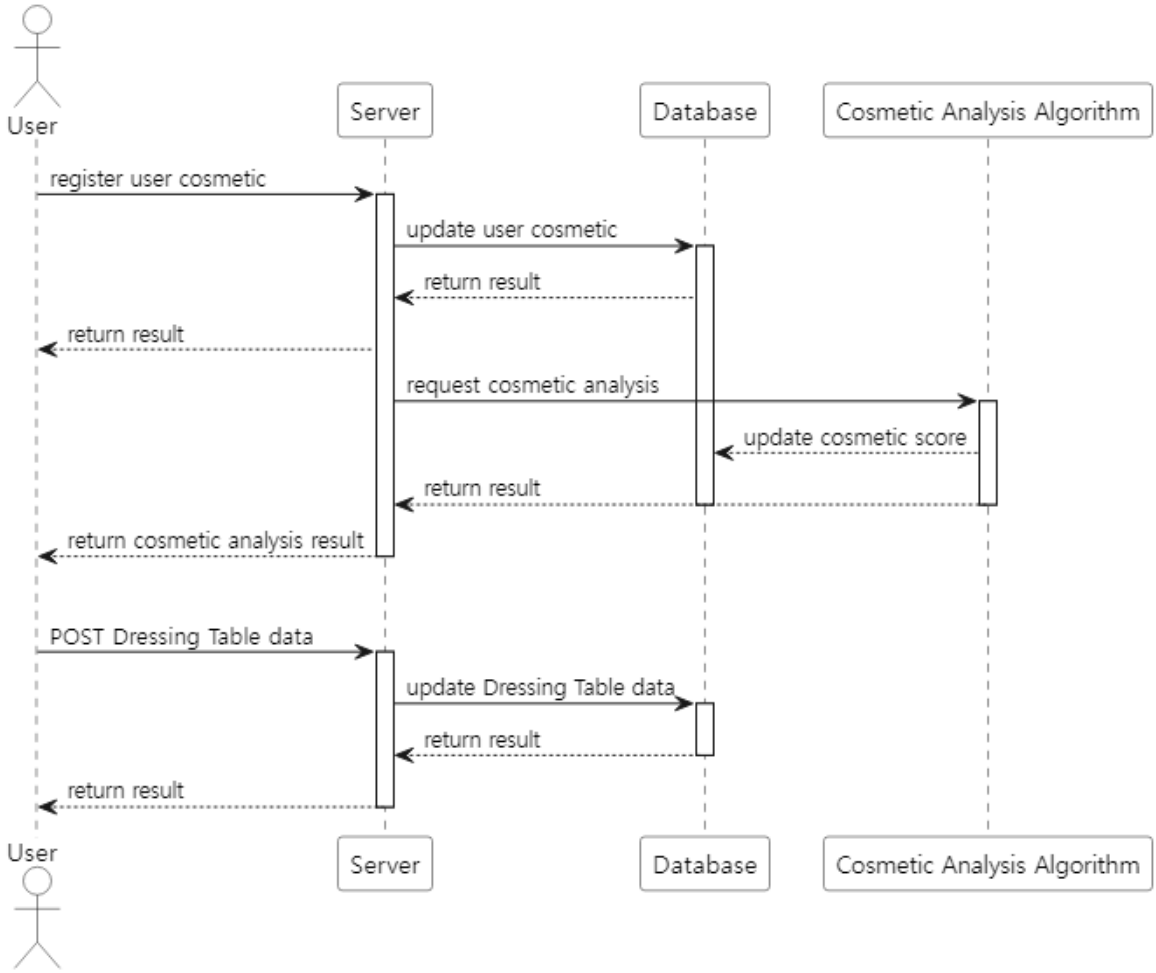


[Figure 36] Sequence Diagram – Cosmetic Analysis System

5.3.6. Cosmetic Management System



[Figure 37] Class Diagram – Cosmetic Management System



[Figure 38] Sequence Diagram – Cosmetic Management System

6. Protocol Design

6.1. Objective

이 장에서는 System Architecture 안에서 Front-End, Back-End 간의 통신 방식과 그것이 전달될 때 어떠한 형식을 가지는지에 대해 설명하고, Application과 Server 간의 통신에 중점을 둔다.

6.2. HTTP

HTTP는 웹 상에서 웹 서버 및 웹 브라우저 간의 데이터 전송을 위한 Application Layer Protocol 이다. Hyper Text Transfer Protocol의 약어이며, 크게 Request 및 Response로 이루어진다. HTTP는 Transaction 중심의 비 연결성 프로토콜이고 이는 종단간 연결이 없는 것(Connectionless)과 이전의 상태를 유지하지 않는다(Stateless)는 특징이 있다. 본 프로젝트에서는 아래에서 언급할 JSON 형태로 데이터를 보낸다.

6.3. JSON

프로그래머 및 개발자, IT 전문가들이 다양한 언어로 작성된 데이터 구조를 다양한 언어와 플랫폼에서 인식할 수 있어야 한다. JSON은 사람이 읽기 가능한 텍스트이고 간단성과 코딩이 덜 필요함과 동시에 빨리 처리할 수 있다. JavaScript Object Notation의 약어이며, serializable value나 key-value 쌍으로 이루어진 데이터 오브젝트를 전달한다. 본 프로젝트에서는 전송할 주요 데이터 양식을 JSON을 사용한다.

6.4. Protocol Design Detail

6.4.1. 회원가입

Attribute	Detail
Method	POST
URI	/registration
Request Body	ID : 아이디 PW : 비밀번호 Name : 사용자 이름 NickName : 사용자가 지정한 닉네임 EmailAddr : 이메일 BirthDate : 생년월일

JSON (Request)	{
	"registration":
	{
	"ID" : "johnny",
	"PW" : "12ac23!dd22",
	"Name" : "John",
	"NickName" : "JohnnyWalker",
	"EmailAddr" : "XXXX@XXXX.com"
	"BirthDate" : "19960402"
	}
	}

Response Body	Status : 회원가입 상태 코드 "RegisterSuccess" "RegisterFailed"

JSON (Response)	{
	"Status" : "RegisterSuccess"
	}

[Table 1] Register

6.4.2. 로그인

Attribute	Detail
Method	POST
URI	/login
Request Body	ID : 아이디 PW : 패스워드

JSON (Request)	{
	"login":
	{
	"ID" : "johnny",
	"PW" : "12ac23!dd22"
	}
	}

Response Body	Status : 로그인 상태 코드 "LoginSuccess" "LoginFailed" LoginKey : 로그인 시 사용자에게 부여하는 고유한 key

JSON (Response)	{
	"Status" : "LoginSuccess"
	"LoginKey" : "18422bca4a18542b"
	}

[Table 2] Login

6.4.3. 로그아웃

Attribute	Detail
Method	DEL
URI	/logout
Request Body	ID : 아이디 LoginKey : 로그인 시 사용자에게 부여하는 고유한 key

JSON (Request)	{
	"logout":
	{
	"ID" : "johnny",
	"LoginKey" : "18422bca4a18542b"
	}
	}

Response Body	Status : 로그아웃 상태 코드 "LogoutSuccess" "LogoutFailed"

JSON (Response)	{
	"Status" : "LogoutSuccess"
	}

[Table 3] Logout

6.4.4. 사용자 피부 상태 측정 및 분석

Attribute	Detail
Method	POST
URI	/skinmeasure
Request Body	Moisture : 피부 수분 Oil : 피부 유분 Acidity : 피부 산성도 Temperature : 피부 온도 DeadSkincell : 각질 정도 Redness: 피부 붉기 Date : 측정 일자 (피부 온도 제외 모든 범위 "0"(낮음) ~ "10"(높음))

JSON (Request)	{
	"skinmeasure":
	{
	"Moisture" : "9",
	"Oil" : "7",
	"Acidity" : "4",
	"Temperature" : "36.3",
	"DeadSkincell" : "5",
	"Redness" : "2",
	"Date" : "20220501"
	}
	}

Response Body	Status : 피부 분석 상태 코드 "MeasureSuccess" "MeasureFailed"

JSON (Response)	{
	"Status" : "MeasureSuccess"
	}

[Table 4] Skin Measurement & Analysis

6.4.5. 추천 화장품 확인

Attribute	Detail
Method	GET
URI	/recomcomes
Request Body	ID : 아이디 ListNum : 불러올 화장품 목록

JSON (Request)	{
	"recomcomes":
	{
	"ID" : "johnny",
	"ListNum" : "5"
	}
	}

Response Body	Status : 추천 상태 코드 "RecomSuccess" "RecomFailed" Comess : 화장품 정보 (이름, URL, 사진 ..)

JSON (Response)	{
	"Status" : "RecomSuccess",
	"Comess" : ["comes" :
	{
	"comesname" : "Flower",
	"comesurl" : "xxxxx@xxxx.com"
	"comespic" : "xxxx/xxxxxxx/xxxx.png"
	...
	}, ...
	]
	}

[Table 5] Cosmetics Recommendation

6.4.6. 화장품 검색

Attribute	Detail
Method	POST
URI	/searchcomes
Request Body	Keyword : 검색 키워드

JSON (Request)	{
	"searchcomes":
	{
	"Keyword" : "moisture"
	}
	}

Response Body	Status : 검색 상태 코드 "SearchSuccess" "SearchFailed" Comess : 화장품 정보 (이름, URL, 사진 ..)

JSON (Response)	{
	"Status" : "SearchSuccess",
	"comess" : ["comes" :
	{
	"comesname" : "Flower",
	"comesurl" : "xxxxx@xxxx.com"
	"comespic" : "xxxx/xxxxxxx/xxxx.png"
	...
	}, ...
	]
	}

[Table 6] Search Cosmetics

6.4.7. 화장품 관리 및 분석 확인

Attribute	Detail
Method	GET
URI	/managecomes
Request Body	ID : 아이디 MyComes : 유저의 화장품 목록 (화장품 이름, 사용기간)

JSON (Request)	{
	"managecomes":
	{
	"ID" : "johnny",
	"Mycomes" : ["comes" :
	{
	"comesname" : "Flower",
	"useofperiod" : "2"
	}, ...
	]
	}
	}

Response Body	Status : 관리 상태 코드 "ManageSuccess" "ManageFailed" UseFreq : 화장품 사용 빈도 (일주일 기준) PurchaseCycle : 화장품 구매 주기 (달 기준)

JSON (Response)	{
	"Status" : "RecomSuccess",
	"UseFreq" : "9",
	"PurchaseCycle" : "4"
	}

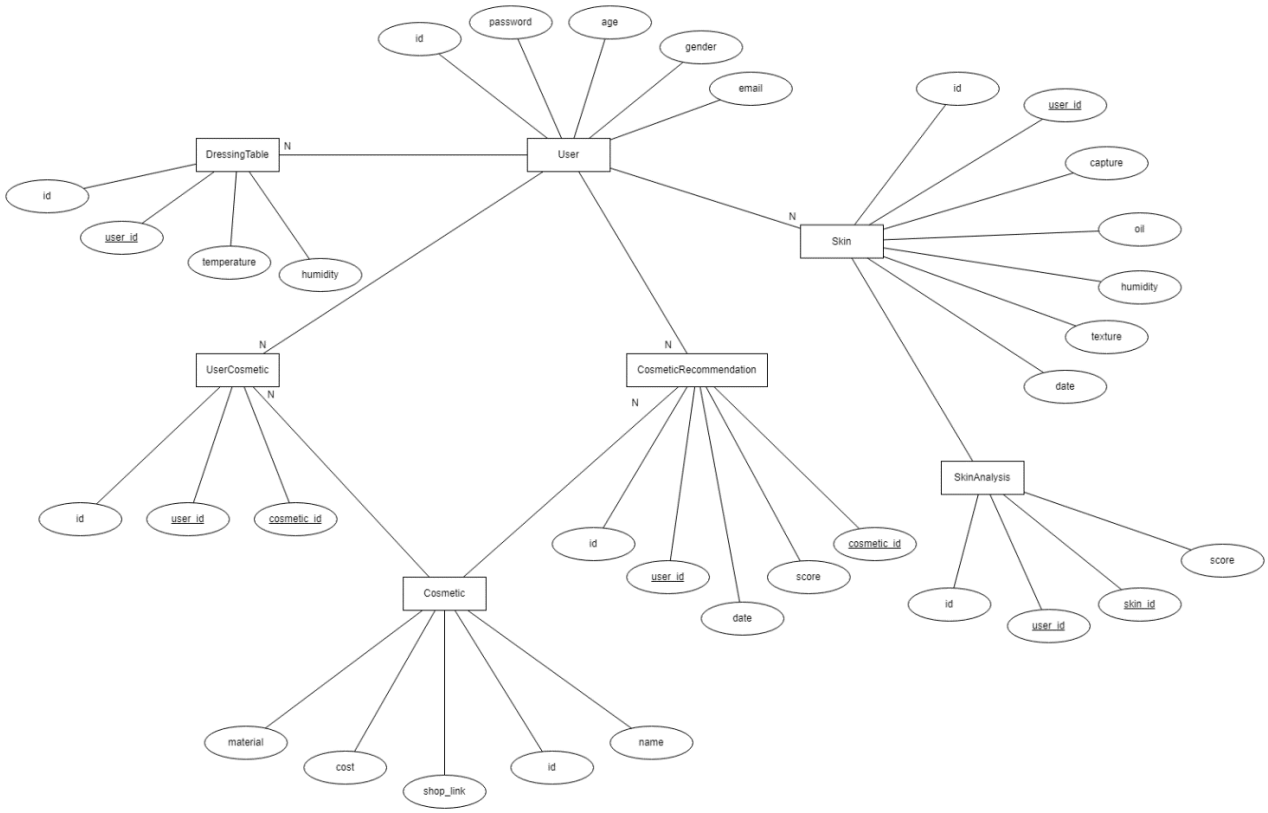
[Table 7] Cosmetics Management & View Analysis

7. Database Design

7.1. Objectives

이 장에서는 데이터베이스의 구성에 대해 설명한다.

7.2. E-R Diagram

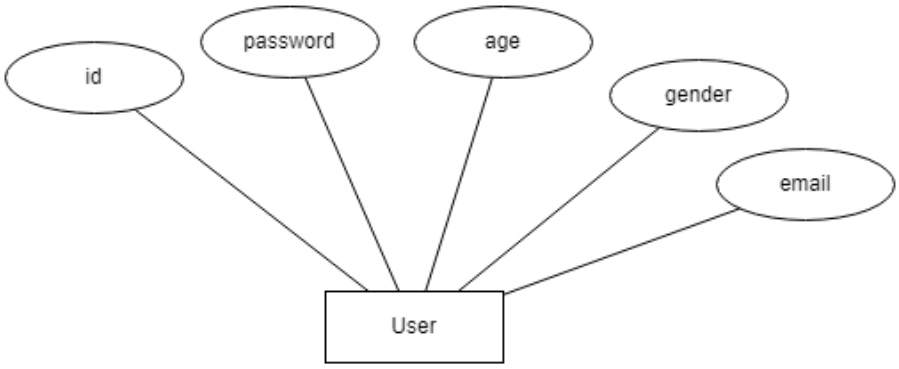


[Figure 39] ER Diagram

7.2.1. Entities

시스템은 총 7개의 entity로 구성되어 있다(User, Skin, SkinAnalysis, Cosmetic, UserCosmetic, CosmeticRecommendation, Dressing Table). 각 entity는 integer 형식의 primary key인 id필드를 가진다.

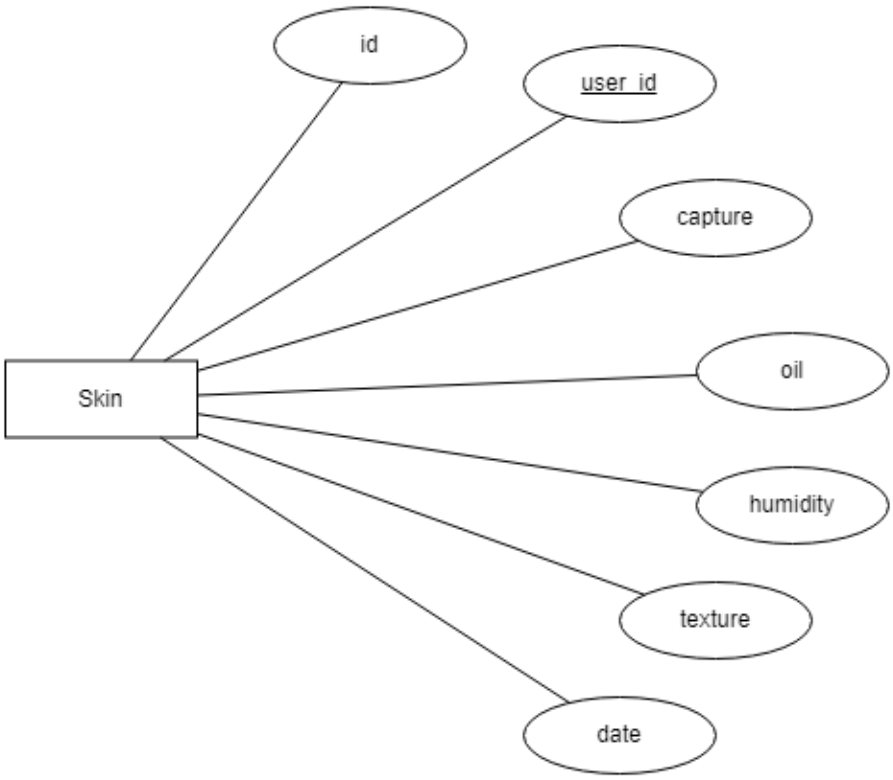
7.2.1.1. User



[Figure 40] ER Diagram - User

User entity는 사용자 정보를 포함한다. Password, age, gender, email등의 필드가 있으며 email은 계정 이름으로 사용된다. Password는 SHA-256 hash algorithm을 사용한 양방향 암호화를 적용하여 데이터베이스에 저장한다.

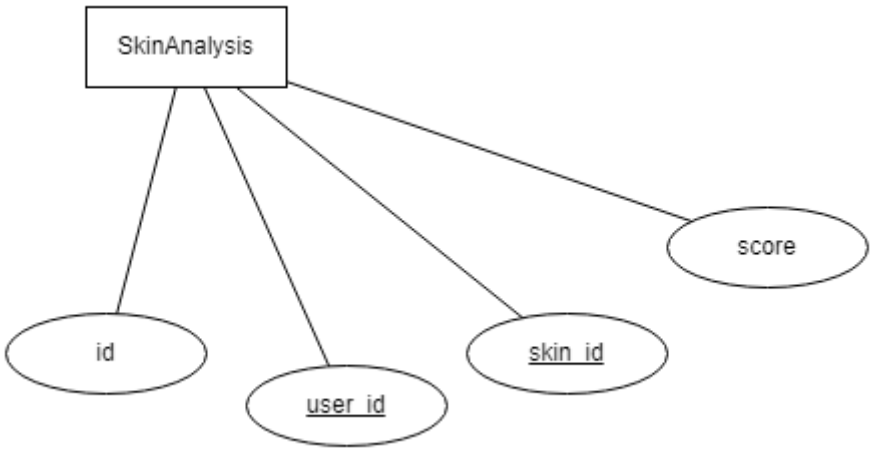
7.2.1.2. Skin



[Figure 41] ER Diagram - Skin

Skin entity는 사용자의 피부 상태 지표를 데이터로 저장한다. Date attribute를 통해 언제 측정했는지 파악이 가능하며 전체 이미지를 저장하는 capture attribute와 피부 결 레이어 이미지를 저장하는 texture attribute로 피부 상태를 육안으로 확인할 수 있다.

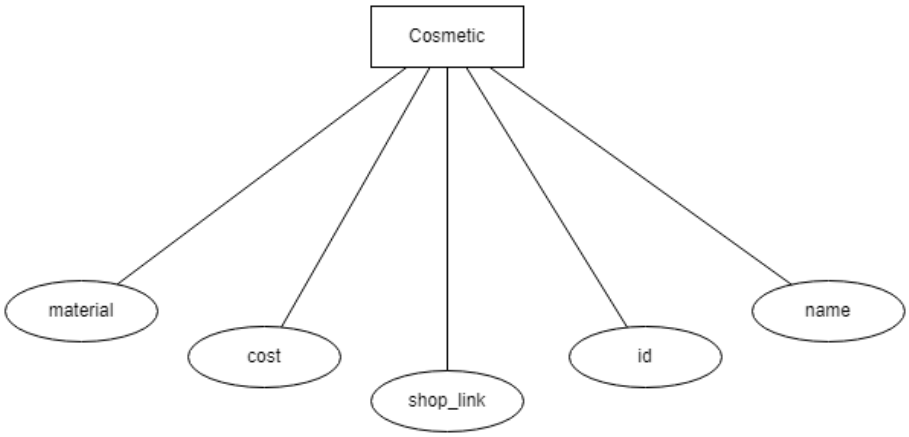
7.2.1.3. Skin Analysis



[Figure 42] ER Diagram – Skin Analysis

SkinAnalysis는 Skin data 분석 알고리즘에 따라 처리된 데이터를 저장한다. Score는 JSON형식의 attirbute 로 Skin entity의 각 요소에 대한 상세 분석 결과와 점수를 저장한다. 사용자와 skin data와의 join을 고려하여 Foreign key로 skin_id와 user_id를 포함한다.

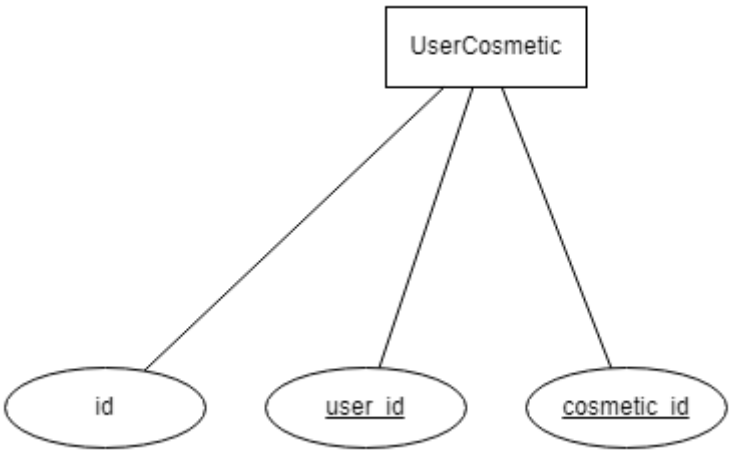
7.2.1.4. Cosmetic



[Figure 43] ER Diagram – Cosmetic

Cosmetic entity는 화장품 정보를 저장한다. Name, material, cost, shop_link등의 정보가 포함되며 shop_link는 해당 화장품을 판매하는 공식 사이트의 주소를 저장하는 필드로, 사용자가 화장품에 대한 자세한 정보를 쉽게 얻을 수 있도록 돕는다.

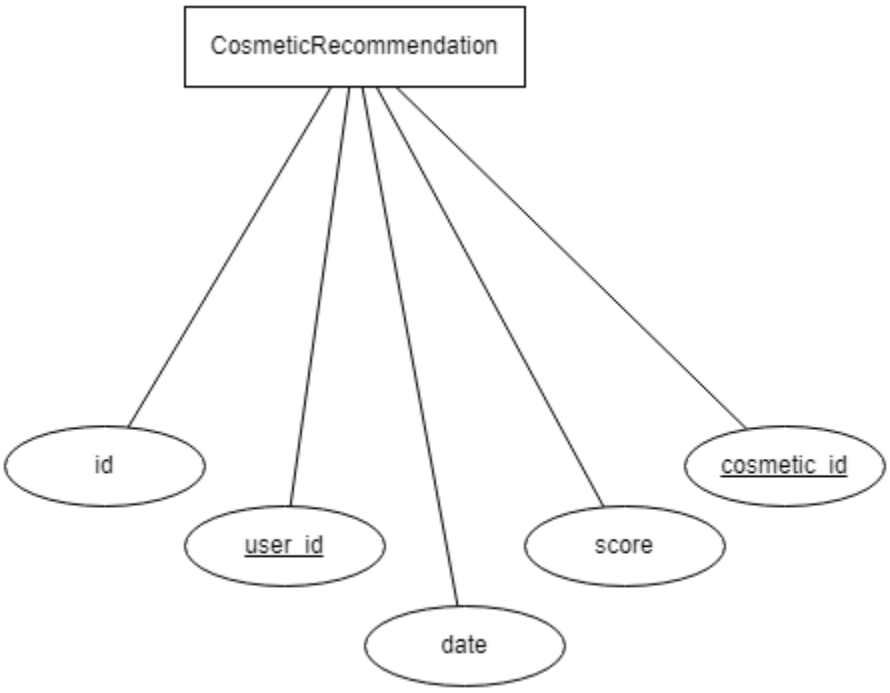
7.2.1.5. UserCosmetic



[Figure 44] ER Diagram – User Cosmetic

UserCosmetic entity는 사용자가 사용하는 화장품 목록을 저장하기 위한 entity이다. User와 Cosmetic entity와의 연관성을 유지하기 위해 cosmetic_id와 user_id를 foreign key로 가진다.

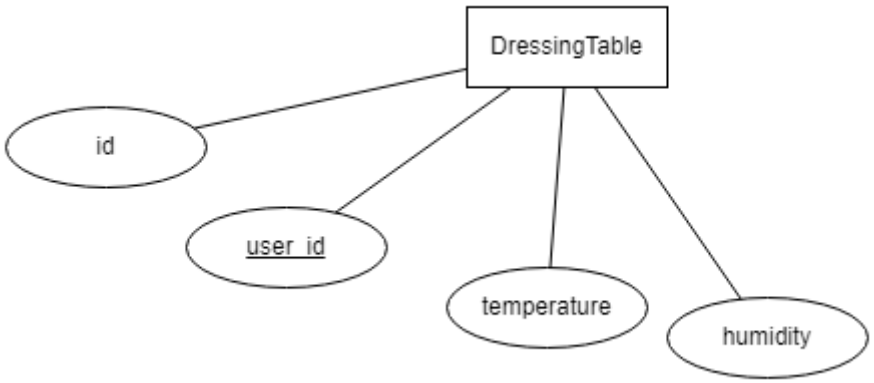
7.2.1.6. CosmeticRecommendation



[Figure 45] ER Diagram – Cosmetic Recommendation

CosmeticRecommendation entity는 AWS Personalize를 통해 반환된 사용자 개인 맞춤형 추천 결과를 저장하기 위한 entity이다. Date로 해당 추천 알고리즘을 사용한 날짜를 같이 저장한다. Score attirbute는 JSON형식으로 추천 결과에 포함된 각 요소별 가중치를 저장하여 사용자가 더 자세한 정보를 얻을 수 있도록 한다.

7.2.1.7. DressingTable



[Figure 46] ER Diagram – Dressing Table

DressingTable entity는 화장대 상태 정보를 저장하기 위한 entity이다. 스마트 화장대에 포함된 Iot기기와 연결되어 화장품을 보관하는 공간의 온도 및 습도 상태를 지속적으로 저장한다.

8. Testing Plan

8.1 Objectives

Testing Plan 단락에서는 소프트웨어 개발 및 배포 단계에서 필요한 여러 테스트에 대해 기술한다. 테스트의 단계는 컴포넌트 테스트, 시스템 통합 테스트, 소비자 테스트로 이루어진다. 이를 통해 프로그램의 각 모듈의 유효성 여부를 판단하며, 모듈을 통합하여 정상 범위 내에서 작동하는지 확인한다. 마지막으로 시스템 테스트와 소비자 테스트를 통하여 Non-functional requirement를 충족하였는지 검증한다. 각 과정에서 잠재된 오류를 발견하며 수정한다. 그리고 요구사항에 부합하는지 확인한다.

8.2 Testing Policy

8.2 Testing Policy 단락에서는 Development Testing과 Release Testing, User Testing 단계를 나누어 각 단계에서 충족되어야 하는 조건을 기술한다.

8.2.1 Development Testing

8.2.1.1 Unit Testing

Unit Testing 단계에서는 Component 내의 Class 또는 함수, Data의 유효성을 검증한다. Class를 검증할 때는 해당 Class가 디자인 계획에 부합하는 상속관계를 따랐는지, Class의 데이터가 목적에 맞는 타입으로 선언되었는지, Class와 Class내 member 변수와 함수의 접근 지정자가 디자인 계획과 부합하는지, Class각 OOP 원칙에 따라 구현되었는지 검증한다. 함수의 경우 함수의 매개변수의 타입이 유효한지, 함수의 매개변수가 유효 범위를 벗어나는 경우 어떻게 처리할지 구현되어 있는지, 각 함수의 동작이 계획에 따라 동작하는지 검증한다. 데이터의 경우 각 데이터가 유효한 범위와 타입을 보장하는지, 그렇지 않은 경우 어떻게 예외 처리를 하는지에 대해 구현되었음을 검증한다.

System Testing과 달리 Unit Testing의 경우, 각 Unit에 필요한 시간 자원은 선형적이기 때문에 가능한 모든 경우에 대해서 테스트한다. 그렇기 때문에 Unit Testing은 Google Test라는 Software testing framework를 이용하여 unit test를 진행한다. 이렇게 개발 단계에서 Unit Testing을 위한 Automatic Testing 환경을 구축하면, 추후에 Class나 함수 등을 수정했을 때 자동으로 Testing을 하도록 하여, 이를 대처하기 위해 들어가는 시간적 그리고 금전적 자원을 절약할 수 있다. 또한 자동으로 Unit Testing을 함으로써 해당 Unit이 정상 작동함을 온전히 확인할 수 있다.

8.2.1.2 Component Testing

프로그램의 컴포넌트들을 통합하기 전에 각 컴포넌트들을 개별 단위로 검증한다. Component는 전체 시스템에서 특정 기능을 수행하는 작은 기능적 단위다. Component Testing은 각 기능 구현을 담당하는 개발 팀 단위로 이루어진다. 각 Component들은 기술된 사항에 따라 입력과 출력이 유효한 결과를 보이는지 검증한다. 이를 통하여 해당 Component의 Interface가 유효한 범위의 데이터를 입력과 출력으로 가지며, 의도된 동작을 수행함을 확인한다. 이를 통해 해당 Component에 오류 여부를 검증하여, 다른 Component의 기능에 악영향을 주지 않음을 확인한다.

8.2.1.3 System Testing

System Testing은 개별 Component들에 대한 Testing을 마친 후, 개별 Component들을 하나의 System으로 통합(Integration)하여 나온 시스템에 대하여 진행한다. System Testing에서 각 Component들이 의도대로 통합되며 서로 오류 없이 통신하는지 확인한다. System Testing에는 성능(Performance), 신뢰성(Reliability), 보안(Security)이 포함된다.

개별 Component들을 테스트할 때는 예상된 시간 내에 실행되었다고 해도, System Testing 단계에서 소요되는 시간은 개별 Component들의 수행 시간의 총합이 아니다. System의 각 기능들의 응답 시간은 Requirement에서 명시된 시간을 준수해야 한다.

System Testing에서 검증해야 하는 다른 요소는 각 기능이 정확한 동작을 하였는지에 대한 신뢰성(Reliability) 검증이다. 만약 시스템이 내부적 요인 혹은 외부적 요인에 인하여 적절한 동작에 실패했을 경우, 시스템이 적절한 방식을 통해 복원 및 재기능을 하는지 검증해야 한다.

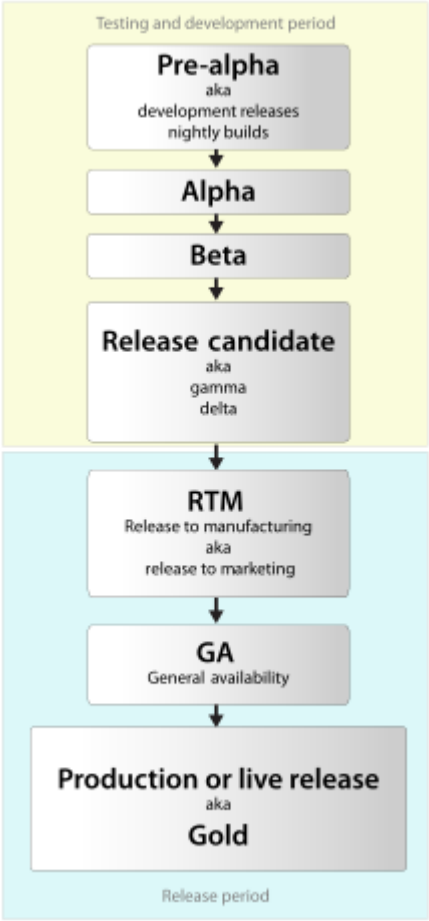
System이 외부의 공격을 막고, 접근 권한이 없는 자가 데이터에 접근하는 사고를 방지해야 한다. 더욱이나 Start Table은 유저 정보를 이용하여 동작하기 때문에 보안(Security)가 중요하다. Component Testing 단계에서 개별 Component에서 보안 문제가 발생하지 않아도 System Testing 단계에서 발생할 수 있다. 그래서 Ostorlab이나 Appvigil 같은 취약점 진단 프로그램을 이용하여 프로그램의 보안 문제가 없는지 검증한다.

8.2.2 Release Testing

Development 환경에서 프로그램이 문제없이 동작하였다고 해도, 이는 프로그램이 실제 환경에 Release된 후에도 정상 작동함을 보장하지 않는다. 비록 프로그램 내부적으로는 문제가 없다고 해도, 실제 프로그램이 실행되는 환경 또는 배포 방법에 의한 문제로 인해 Release 된 프로그램에

문제가 발생할 수 있다. 그렇기 때문에 소프트웨어를 소비자들에게 실제로 release하기 전에 반드시 Release Testing을 거쳐야 한다.

소프트웨어 생명 주기(Software Release Life Cycle)에 의하면 Release Testing 단계 전을 Pre-alpha 단계라고 통칭한다. 이후 첫 Release Testing인 Alpha 단계에서는 실제 환경에 프로그램을 Release 하여 Testing을 진행한다. 이 때 Alpha Testing은 개발팀 혹은 회사 내부적으로 진행한다.



[Figure 47] Testing

8.2.3 User Testing

소프트웨어 생명 주기 중 Beta 단계에서는 사용자를 대표하는 사람들을 대상으로 Testing을 진행한다. Beta Testing 중에 발견된 버그 혹은 제시된 의견을 반영하여 Release Candidates를 선발 및 개발 후 출시 단계로 넘어간다.

8.3 Testing Case

8.1 Objective와 8.2 Testing Policy에 의거하여 프로그램의 performance, reliability, security를 확인하고, 버그 발생과 기능의 정상 작동 여부를 확인하기 위하여 다음과 같은 Testing Case를 생성 후 Testing 한다.

8.3.1 Unit / Component Testing

8.3.1.1 Embedded System과 센서

Component	입력	동작	출력 / 유효성 검증
온/습도 측정	온/습도 측정 장치의 32bit 정수 값	입력으로 들어온 32bit 정수를 실수 값으로 변환하여 return	32bit 실수 값인지 확인
피부 상태 측정	피부 상태에 대한 패킷	해당 패킷을 Decode하여 서버로 전송	서버에 전송한 패킷의 내용이 진단한 내용과 부합하는지
카메라	카메라 센서 데이터	사용자의 얼굴 이미지를 Embedded System을 통하여 서버로 전송	서버에 전송되는 패킷에 담긴 사용자 얼굴 이미지 확인
스마트 거울 좌표 변환	압력 좌표 데이터	압력 좌표 데이터를 거울의 위치 데이터(cm)로 변환	압력 좌표 데이터와 cm 데이터가 동일한지 검증
스마트 거울 디스플레이	스마트 거울에 표시해야 하는 데이터 파일	데이터 파일을 스마트 거울에 표시해야 하는 이미지 파일로 변환 후, 해당 이미지 파일의 컬러 값과 불투명도 값에 따라서 거울 좌표에 표시해줘야 하는 불빛의 강도 값으로 변환	입력과 동일한 정보가 화면에 표시되는지 확인

[Table 8] Embedded System Unit / Component Testing

8.3.1.2 스마트폰 어플리케이션

Component	입력	동작	출력 / 유효성 검증
사용자 로그인	ID / password	사용자 정보 데이터 베이스와 입력된 ID/password를 비교하여 로그인 승인	정확한 입력: 메인 화면 부정확한 입력: 로그인 실패 메시지
회원 가입	사용자 정보	입력된 사용자 정보를 검증하여 데이터 베이스에 저장	입력된 사용자 정보가 데이터 베이스에 정확히 저장되었는지 확인
장치 찾기	장치 찾기 버튼 클릭 이벤트	장치 찾기 페이지를 불러온다.	장치 찾기 페이지
사용자 얼굴 불러오기	사용자 얼굴 불러오기	갤러리로부터 사용자의 얼굴 이미지를 불러온다.	화면에 사용자의 얼굴 이미지가 성공적으로 불러오는지 확인
얼굴 이미지 조작	버튼 클릭 이벤트	해당 버튼 이벤트에 따라 얼굴 이미지에 적용한다.	좌우반전, 밝기, 줌, 립스틱, 블러셔, 아이새도, 아이브로우는 육안으로 확인. 퍼스널 컬러 진단은 서버의 데이터 베이스와 동일한 결과가 나오는지 비교.
피부 상태 분석	사용자의 얼굴 이미지와 데이터 베이스로부터의 진단 데이터	사용자의 얼굴 이미지와 진단 데이터를 화면에 표시한다.	사용자의 얼굴과 진단 데이터가 표시된 화면
화장품 추천	검색 키워드 및 필터 설정	각 키워드와 필터에 적합한 화장품 목록 데이터 베이스로부터 로딩	화장품 목록
검사 결과 공유	공유 버튼 클릭 이벤트	해당 검사 결과를 공유하도록 SNS와 연결	SNS로 검사 결과가 공유되는지 확인
화장품 등록	화장품 정보 입력과 등록 버튼 클릭 이벤트	입력된 화장품 정보를 데이터 베이스에 저장	데이터 베이스에 해당 정보가 저장 되었는지 확인
사용중인 화장품 관리 및 분석	데이터 베이스로부터 받은 화장품 데이터	화장품 데이터를 화면에 표시한다.	현재 보유한 화장품 정보와 진단 데이터가 화면에 올바르게 표시되었는지 확

			인
설정	버튼 클릭 이벤트	해당 버튼 이벤트에 따라서 스마트폰의 설정 데이터를 변경하고 서버의 데이터와 동기화한다.	클릭 이벤트가 발생한 설정 값이 메모리에서 제대로 변경되었는지 확인. 그리고 서버로 동기화 요청을 송신하였는지 확인.
푸쉬 알림	Event queue로부터 나온 event	BroadcastReceiver를 통해서 해당 event를 포착하여 발동된다. 이 때 사용자의 설정 값을 읽은 후 푸쉬 허용 여부에 따라서 푸쉬 알림을 띄운다.	온도와 습도에 대한 event를 인위로 발생 시켰을 때 푸쉬에 대한 설정 값에 따라서 푸쉬가 화면이 표시되는지 확인

[Table 9] Smart Phone Application Unit / Component Testing

8.3.1.3 Server

Component	입력	동작	출력 / 유효성 검사
데이터 베이스	화장품 혹은 사용자 정보 추가 요청	새로 입력된 정보의 유효성을 검증 후 데이터 베이스에 추가	1: 추가에 성공 0: 추가에 실패
	화장품 혹은 사용자 정보 삭제 요청	해당 데이터의 Dependency를 확인 후 삭제 가능하면 삭제	1: 삭제에 성공 0: 삭제에 실패
분석 알고리즘	분석 알고리즘 함수 call, 사용자 ID	사용자 DB와 화장품 DB, 피부상태 DB로부터 분석을 위한 데이터를 불러온다. 그 후 AI 기술을 이용하여 사용자의 피부 상태 분석 및 추천 화장품을 위한 embedded 값을 추론하여 DB에 저장한다.	분석된 사용자의 피부 상태와 화장품 추천이 label 값과 얼마나 유사한지 확인

[Table 10] Server Unit / Component Testing

8.3.2 System Testing

개별 Unit/Component 들이 모두 Testing을 통과하면 모둔 Component들을 통합하여 각각의 프로그램으로 build후 설치하여 System을 구축한다. 그 후 System의 성능(performance), 신뢰성(Reliability), 보안(Security)를 확인하기 위해서 다음을 Testing한다.

약 만 대 이상의 단말 장치에서 동시에 서버로 요청을 보냈을 때 서버가 2초 내에 응답을 하는지, 그리고 해당 응답이 정확한 정보를 보내는지 확인한다. 각 요청을 위의 Component Testing Case와 동일하며, 여러가지 요청을 동시에 보낸다. 그리고 취약성 탐지 프로그램을 적용해보고, 전문가를 고용하여 System의 보안상 취약점을 없는지 검증한다.

8.2.3 Release Testing

Constraint에 명시된 기준에 부합하는 단말기와 실제 AWS 서버에 설치했을 때 성능과 신뢰성, 보안을 보장하는지 검증한다. Testing Case는 위 System Testing과 동일하다.

8.2.4 User Testing

선발된 사용자들에게 위의 프로그램을 체험하게 한 후 다음과 같은 설문을 받는다. (1점: 매우 불만족 ~ 5점: 매우 만족)

- Embedded 기기와 스마트폰 앱의 반응 속도는 적절했는가?
- 가입과 탈퇴가 용이했는가?
- 화장품 추천이 적절했는가?
- 화장품 관리 기능이 신뢰할 만했는가?
- 어플리케이션의 UI/UX가 디자인적으로 적절했는가?
- 설정 기능이 충분했는가?
- 가상 메이크업 기능이 자연스러웠는가?
- 피부 상태 진단이 적절했는가?
- 장치 등록이 원활했는가?
- 센서로 측정한 데이터가 정확하다고 보는가?
- 추가했으며 하는 기능이나 개선되어야 하는 점을 자유롭게 적어주세요.

9. Development Plan

9.1 Objectives

이 장에서는 소프트웨어 개발에 사용되는 여러 기술 및 API, SDK, 개발 환경이 기술된다.

9.2 Front-end Environment

9.2.1 Flutter

[Figure 48] Flutter

Smart Table Dressing의 스마트폰 어플리케이션은 게임 어플보다 긴 응답속도도 허용한다. 그렇기 때문에 개발 과정만 아니라 추후 Evolution 단계에서 수정이 일어난 경우 개발 및 배포의 용이성이 중요하게 고려되어야 한다. Flutter는 안드로이드와 iOS와 모두 호환이 되는 크로스 플랫폼 프레임워크다. Flutter를 이용하여 개발할 경우 각 하드웨어와 운영체제에 맞게 수정하는 비용을 줄일 수 있다. 그렇기 때문에 Flutter을 통해 스마트폰 어플리케이션의 UI 개발에 Flutter을 이용한다.

9.2.2 Android Studio

[Figure 49] Android Studio

안드로이드 스튜디오는 안드로이드 OS 기반 어플리케이션을 개발할 때 이용되는 IDE(Integrated Development Environment)다. 안드로이드 스튜디오에서 지원하는 언어에는 Java, Kotlin, C++ 이 있지만, Flutter Plugin을 통해서 Flutter 개발도 지원한다. 안드로이드 스튜디오는 상업적 이용에 대해서도 요금을 청구하지 않기 때문에 개발 비용 절감에도 도움이 된다.

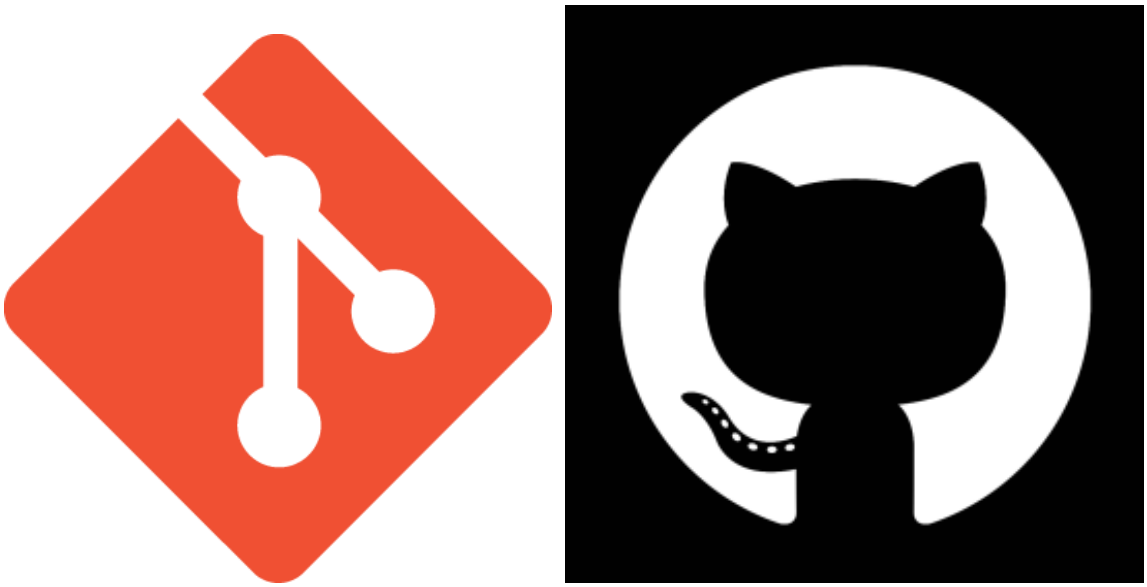
9.2.3 Adobe Photoshop / Adobe Illustrator

[Figure 50] Adobe Photoshop & Adobe Illustrator

스마트폰 앱의 UI/UX에 사용되는 아이콘 제작을 위해 Vector 기반으로 구동되는 Adobe Photoshop과 Adobe Illustrator 이용한다.

9.3 Back-end Environment

9.3.1 Git / Github



[Figure 51] Git & Github

소프트웨어의 버전 관리를 위하여 git과 github를 이용한다.

9.3.2 AWS EC2 (Server)

[Figure 52] AWS

AWS(Amazon Web Services)는 아마존에서 제공하는 클라우드 컴퓨팅 플랫폼이다. 이는 유저에게 가상 머신을 제공하여 사용량에 따라서 요금을 부과한다. 이를 통해 서버리스 환경을 구축한다. 소프트웨어 혹은 어플리케이션을 AWS의 EC2로 제공함으로써 사용자의 증감에 따라 필요한 서버 자원을 이용하게 된다. 이에 따라서 사용자 수에 따른 Scalability 문제를 해결할 수 있다. 사용 서비스를 이용하여 Scalability 문제에 필요한 자원을 절약할 수 있다.

9.3.3 postgreSQL

[Figure 53] PostgreSQL

PostgreSQL은 오픈 소스이며, 객체-관계형 데이터베이스 시스템(ORDBMS)다. PostgreSQL은 다른 관계형 데이터 베이스 시스템과 달리 연산자, 복합 자료형, 집계함수, 자료형 변환자, 확장 기능 등 다양한 데이터베이스 객체를 사용자가 임의로 만들 수 있는 기능을 제공한다. PortgreSQL은 클라이언트- 서버 모델로 이루어져 있다. 서버는 데이터 베이스 파일들을 관리하고, 클라이언트는 애플리케이션으로부터 들어오는 연결을 수용하며, 클라이언트를 대신하여 데이터 베이스 액션을 수행한다. 서버는 다중 클라이언트 연결을 처리하는데, 서버는 클라이언트의 연결 요청이 오면 각 커넥션에 대한 새로운 프로세스를 fork 한다. 클라이언트는 기존 서버와 간섭 없이 새로 생성된 서버 프로세스와 통신한다. PostgreSQL은 관계형 DBMS의 기본적인 기능인 트랜잭션 기능과, ACID(Atomicity, Consistency, Isolation, Durability)를 지원한다.

PostgreSQL은 라이선스에 대한 비용 문제가 없는 BSD 라이선스다. 그렇기 때문에 프로젝트에 비용 절감에 도움이 된다는 장점이 있다. 오랜 시간 검증되어 안정적이며 높은 신뢰성을 보인다.

9.3.4 Amazon Personalize

[Figure 54] Amazon Personalize

아마존 펄스널라이즈(Amazon Personalize)는 아마존에서 제공하는 완전 관리형 기계 학습 서비스다. 아마존 펄스널라이즈는 데이터를 처리, 기능, 식별, 최상의 알고리즘 탐색, 모델 훈련, 최적화 및 호스팅 등 전에 기계 학습 서비스 과정의 파이프라인을 관리해준다. 이를 통해 별도의 기계학습 프로그램 구축 없이, 사용자 데이터에 따라 SDT의 화장품 및 피부 관리법 추천 시스템을 구축한다.

9.4 Constraints

SDT 개발에 있어 본 문서에서 작성된 내용을 준수하여야 한다. 이 외에 본 문서에서 명시되지 않은 개별 Unit 과 Component 내의 구현 방식에 대해서는 개별 Developer 또는 팀의 자율에 따른다. 단, 개발 과정에 있어서 다음과 같은 원칙을 준수하여야 한다.

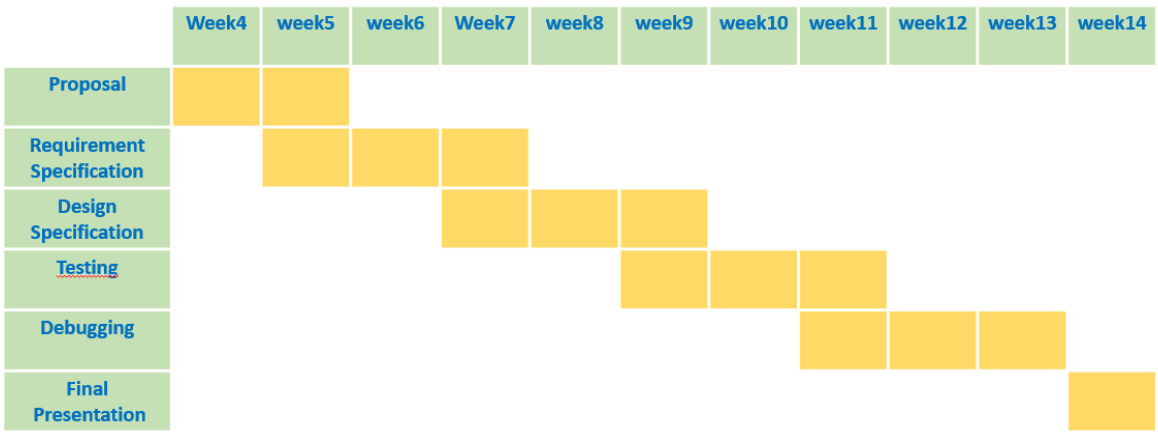
- 사용자의 회원정보 및 데이터 수집에 있어서 반드시 개인정보보호법에 따라 약관을 명시 및 동의를 구하여야 한다.
- 불필요한 사용자의 정보는 수집하지 않아야 한다.
- 오픈소스를 이용할 때에는 라이선스 조사 및 결정권자의 승인 후에 소프트웨어에 포함시킨다.
- 구현해야 하는 기능/Component 중 이미 상용화된 프로그램이 있다면 개발 시간 단축을 위해 가급적 기존의 프로그램을 이용한다. 이 때 해당 프로그램의 구매 여부는 회계 담장자에 청구 후 승인이 나오면 구매를 진행한다.
- 사용자의 개인정보를 위하여 권한에 대한 명세를 반드시 지키며, 만약 명세서에서 벗어나는 기능 또는 Component를 추가해야 할 때는 보안팀과 협의 후 진행한다.
- 사용자 정보는 Embedded System 혹은 스마트폰의 어플리케이션이 아니라 가급적 서버에 저장한여, Edge 단말기의 성능 저하를 방지한다.
- Edge 단말기의 특성상 연결이 불안정 할 수 있기 때문에 Transaction 실패 시 복원 및 복구하는 대안이 구비되어야 한다.
- 유저의 사용성을 위하여 불필요한 동기화 함수는 자제한다.
- 스마트폰 어플리케이션에 새로운 API나 SDK 도입 시 안드로이드와 iOS에서 모두 호환이 되는 후보군 중 선택한다.

9.5 Assumptions and Dependencies

본 문서의 SDT 시스템은 리눅스와 오픈 소스, 그리고 최소 Android 6.0 (API 23), iOS 15 이상의 버전을 기반으로 설계 및 구현할 것을 가정하여 작성되었다. 서버의 경우 AWS에 제공하는 운영체제 중 하나인 리눅스 운영체제만 고려했으며, 클라이언트의 경우 안드로이드 운영체제와 iOS만 고려하였다.

- Server
    - OS: AWS Lambda, AWS API Gateway
    - DBMS: PostgreSQL 14+
- Embedded System
    - OS: Ubuntu Core 20
    - Board: Raspberry Pi 4 Model B
    - 온, 습도 측정기: DHT22
    - 피부 측정기: 지니 스킨
- 스마트폰 OS
    - 안드로이드 6.0 이상
    - iOS 15 이상
- 스마트 폰과 Embedded system은 블루통신이 가능해야 한다.

9.6 Gantt Chart



[Figure 55] Gantt Chart

10. Supporting Information

10.1 서식

본 SDS(software design specification)은 IEEE Recommendation (IEEE Recommendation Practice for Software Requirements Specification, IEEE-std-830) 서식을 기반으로 제작되었다.

10.2 Document History

날짜	버전	비고	작성자
2022.05.09	0.1	문서 목차 및 개괄 작성	모두
2022.05.09	0.2	1. preface 작성	이영신
2022.05.09	0.3	4.1 Objective 작성	최정훈
2022.05.09	0.4	10. Supporting Information 작성	박재성
2022.05.09	0.5	4.2.5 피부 상태 표시 작성	임재원
2022.05.09	0.6	2. Introduction 작성	이영신
2022.05.10	0.7	4.2.1 ~4.2.3 작성	최정훈
2022.05.10	0.8	4.2.6 ~ 4.2.7 작성	임재원
2022.05.10	0.9	4.2.4 메인 화면 작성	최정훈
2022.05.10	1	5. System Architecture – backend 작성	김학산
2022.05.10	1.1	6. protocol design 작성	안영태
2022.05.10	1.2	4.2.8 ~ 4.2.9 작성	임재원
2022.05.10	1.3	7. Database design 작성	김학산
2022.05.11	1.4	3. System Architecture – Overall 작성	이영신
2022.05.11	1.5	8. Testing Plan 작성	박재성
2022.05.12	1.6	9. Development Plan 작성	박재성
2022.05.12	1.7	1. preface, 2. Introduction 수정	이영신
2022.05.12	1.8	4.2.1 ~ 4.2.5 수정	최정훈
2022.05.12	1.9	4.2.5~4.2.9 수정	임재원
2022.05.13	2	4.2.5 메인 화면 수정	임재원
2022.05.13	2.1	2. Introduction 수정	이영신
2022.05.13	2.2	3. System Architecture – Overall 수정	이영신
2022.05.14	2.3	8 ~ 9 수정	박재성
2022.05.14	2.4	6. protocol design 수정	안영태

[Table 11] 문서 작성 이력