성균관대학교
SUNG KYUN KWAN UNIVERSITY

Smart Mirror
For Health Care

2022.05.15

Introduction to Software Engineering

Team 13

조장 조재훈
조원 박민서
조원 백송현
조원 설채은
조원 이재혁
조원 정민석
조원 Vincent Pan

목차

LIST OF FIGURES

LIST OF TABLES

1. Preface

이 장에서는 이 문서의 독자층, 목차, 목적 등을 명시하고있다

1.1. Readership

이 문서는 피부 건강 관리를 위한 스마트 미러의 주요 개발 인원인 2022 년 1 학기 소프트웨어공학개론 13 조를 주요 독자층으로 두고 있다. 추가적으로 해당 수업의 교수, 조교, 다른 조원들이 독자가 될 수 있다.

1.2. Scope

이 문서는 피부 진단을 위한 스마트 미러 및 보조 어플리케이션의 소프트웨어 공학과 소프트웨어 품질 공학에 사용될 예정이다.

1.3. Objective

본 소프트웨어 디자인 명세서(SDS; Software Design Specification)는 스마트 미러의 피부 진단 기능을 구현하기 위해 소프트웨어 요구사항 명세서(SRS; Software Requirements Specification)에서 기술한 소프트웨어 전체적인 디자인과 기능에 대해 다루고 있다. 또한 세부적으로 나누어 어플리케이션 단의 프론트엔드, 서버단의 백엔드, 피부 진단 결과 처리를 위한 AI 로 나누어 각각의 세부적인 구조와 디자인, 동작 구조에 대해 기술하고있다. 또한 다양한 다이어그램을 이용해 파트별 기능을 구체화 하고있다. 위 명세서는 개발자, 디자이너, 고객, 소프트웨어 테스터들이 주요 대상이며, 경우에 따라 늘어날 수 있다.

1.4. Document Structure

1 장에서는 문서 전반적인 구조와 독자층에 대해 기술한다.

2 장에서는 문서에 등장하는 다이어그램에 대한 간략한 설명과 예시, 작성을 위해 이용한 도구들에 대해 기술한다.

3 장에서는 스마트 미러 시스템 전반적인 구조에 대해 기술한다.

4 장에서는 스마트 미러 시스템의 프론트엔드 (스마트 미러 및 모바일 어플리케이션) 파트의 구조에 대해 기술한다.

5 장에서는 스마트 미러 시스템의 백엔드파트의 구조에 대해 기술한다.

6 장에서는 스마트 미러 시스템의 AI(피부 진단 및 학습)파트의 구조에 대해 기술한다.

7 장에서는 스마트 미러 시스템의 테스트 계획에 대해 기술한다.

8 장에서는 스마트 미러 시스템 개발에 이용할 툴, 가정들과 제약사항에 대해 기술한다.

9 장에서는 문서 기록 및 서식에 대해 기술한다.

2. Introduction

소프트웨어공학개론 13 조의 피부 건강 관리를 위한 스마트 미러는 코로나 19 의 유행으로 마스크의 착용이 일상화된 현 상황을 배경으로 하고있다. 이에 따라 증가한 피부 트러블을 개개인이 집에서도 편하게, 객관적으로 진단해 질병을 조기에 발견하고 치료할 수 있도록 보조하는 시스템을 개발하고자 한다. 사용자는 진단 히스토리를 조회해보고, 이전에 앓은 질병의 기록을 추가함으로써 진단 결과에 정확도를 높이도록 기여할 수 있다.

이 문서는 구현(개발)에 필요한 디자인에 초점을 두고 작성이 되었다. 스마트 미러 시스템의 세부 기능은 소프트웨어 요구사항 명세서에서 확인할 수 있다.

2.1. Objectives

위의 절에서는 이 문서에 사용되는 다이어그램, 기타 문서화 도구들에 대해 명시하고, 참고 문헌에 대해 정리해 기술함을 목적으로 한다.

2.2. Applied Diagrams

위의 절에서는 이 문서에 사용된 Diagram 에 대해 기술한다.

2.2.1. Class Diagram

Class Diagram 은 UML 의 구조 다이어그램으로, 클래스 내부 구성 요소 및 클래스 간의 관계를 도식화하여 시스템의 구조를 더욱 직관적으로 보여주는 다이어그램이다. Class Diagram 은 시스템을 구성하는 클래스 이름과 속성, 메서드, 관계로 구성되어있다. 클래스의 속성은 클래스의 구성 변수들이 해당되고, 클래스의 메서드는 클래스가 지닌 기능(함수)들이 해당된다. 클래스간의 관계는 상속, 의존성 등이 있다.

2.2.2. Sequence Diagram

Sequence Diagram 은 특정 행동이 어떠한 순서로 어떤 객체와 어떻게 상호작용 하는지 표현하는 행위 다이어그램이다. Sequence Diagram 을 통해 Use Case 를 디테일하게 알 수 있고, 시나리오를 파악하기에 용이하다. Sequence Diagram 은 Actor 와 기타 객체, 시간을 나타내는 Lifeline, 시나리오를 나타내는 message 로 구성되어있다.

2.2.3. Activity Diagram

Activity Diagram 은 객체의 상태가 아닌 처리 로직이나 조건에 따른 처리 흐름을 순서에 따라 정의한 다이어그램이다. 처리동작에 해당되는 Activity 와 그 결과, 혹은 입력값에 해당되는 Data 로 구성되어 있으며, 화살표를 이용해 과정을 표현한다.

2.2.4. State Diagram

State Diagram 은 사건이나 시간에 따라 시스템 내의 객체들의 변화를 잡아내 그 상태를 나타내는 다이어그램이다. State Diagram 은 시작점과 종료점, 시스템의 상태와 그 상태에 취하는 행동에 대한 간략한 설명, 이벤트를 나타내는 화살표로 구성되어있다.

2.2.5. Context Diagram

Context Diagram 은 전체적인 시스템의 데이터 흐름을 표현하는 다이어그램이다. 따라서 시스템과 사용자 간의 데이터의 흐름과 같은 큰 흐름들이 주로 표현된다. Context Diagram 은 일반적으로 정가운데에 시스템이 표시되고, 그 주위로 사용자와 기타 요소들의 관계를 표현하는 식으로 구성된다.

2.3. Applied Tools

위의 절에서는 소프트웨어 디자인 명세서 작성에 이용한 도구들에 대해 기술한다.

2.3.1. Microsoft Word

Microsoft Word(이하 Word)는 보편적으로 사용되는 문서 편집 프로그램으로, 13 조는 Word 를 활용해 소프트웨어 요구사항 명세서와 소프트웨어 디자인 명세서를 작성하였다. 또한 Word 의 기능을 활용해 페이지 및 도표, 다이어그램 정리를 간편하게 할 수 있었다.

2.3.2. Draw.io

Draw.io 는 플로우 차트를 작성하는 온라인 툴로, 소프트웨어 요구사항 명세서와 소프트웨어 디자인 명세서에 첨부된 UML 을 작성하는 데에 이용하였다.

2.4. Project Scope

스마트 미러를 통해 지속적으로 사용자의 사진을 전달받아 딥러닝 모델을 활용해 학습하고 분류함으로써 피부 상태를 진단하고, 그 결과의 정확도를 향상시킬 수 있다. 또한 사용자가 본인의 피부 상태에 대한 정보 및 이전의 피부 질환 이력을 제공함으로써 결과의 정확도를 높이고, 조금 더 각 사용자에게 맞는 결과를 내보낼 수 있다. 또한, 추후에 피부 진단 뿐만 아닌

표정을 통해 사용자의 감정을 분석하는 기능을 추가해 외적 건강 뿐만 아니라 내적(심적) 건강 관리에 도움을 줄 수 있다.

2.5. References

"2021spring_41class_team7." GitHub, 10 Mar. 2022, github.com/skkuse/2021spring_41class_team7/blob/main/doc/Team7_SDS.pdf. Accessed 11 May 2022.

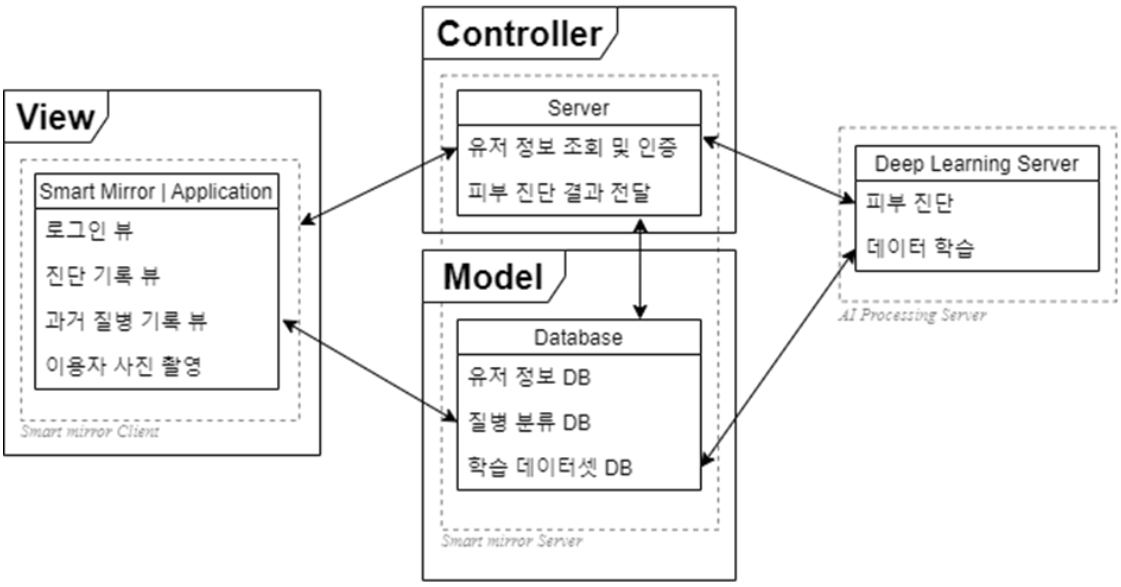
3. Overall Architecture

3.1. Objectives

이 장에서는 피부 건강 관리를 위한 스마트 미러 시스템의 전체적인 구조에 대해 기술한다.

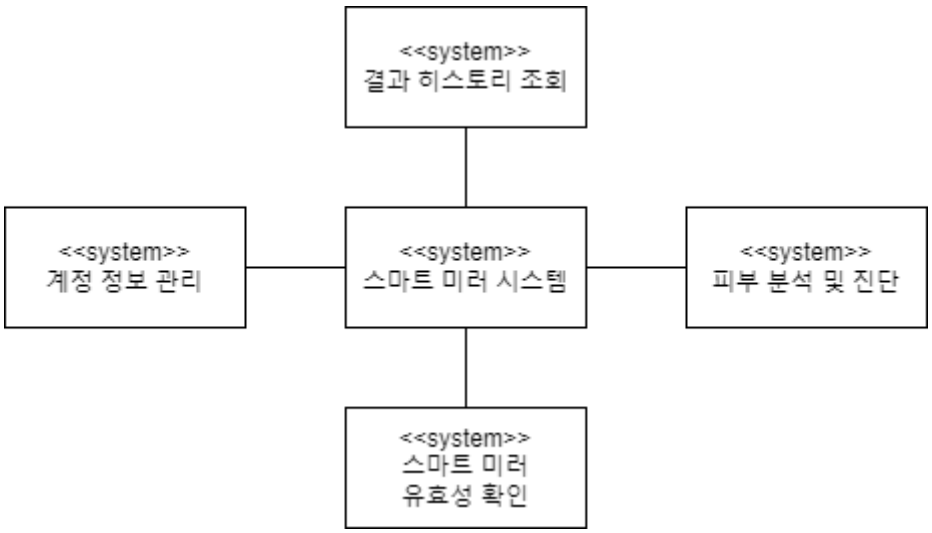
3.2. System Organization

스마트 미러 시스템은 카메라가 장착된 스마트 미러와, 이를 통해 얻은 사용자의 사진을 분석하는 AI 처리 서버, 그리고 스마트 미러와 AI 처리 서버 사이에서 상태를 처리하고, 데이터를 관리하는 메인 서버로 구성되어있다. 따라서 전체적인 구조는 MVC 패턴을 활용해 구성하였다. View 에 해당하는 스마트 미러와 모바일 어플리케이션 쪽에서 데이터 전송 및 결과 요청을 메인 서버에 요청하면, Controller 에 해당하는 메인 서버는 데이터를 받아 데이터베이스에 저장하거나 조회하고, AI 처리 서버에 요청을 보낸다. AI 처리 서버에서 완료되면 메인 서버에 처리가 완료되었음을 알리고, Controller 에게 해당 결과를 반환하라고 요청한다. 이 요청에 의해 결과가 스마트 미러 및 모바일 어플리케이션에 나타난다.



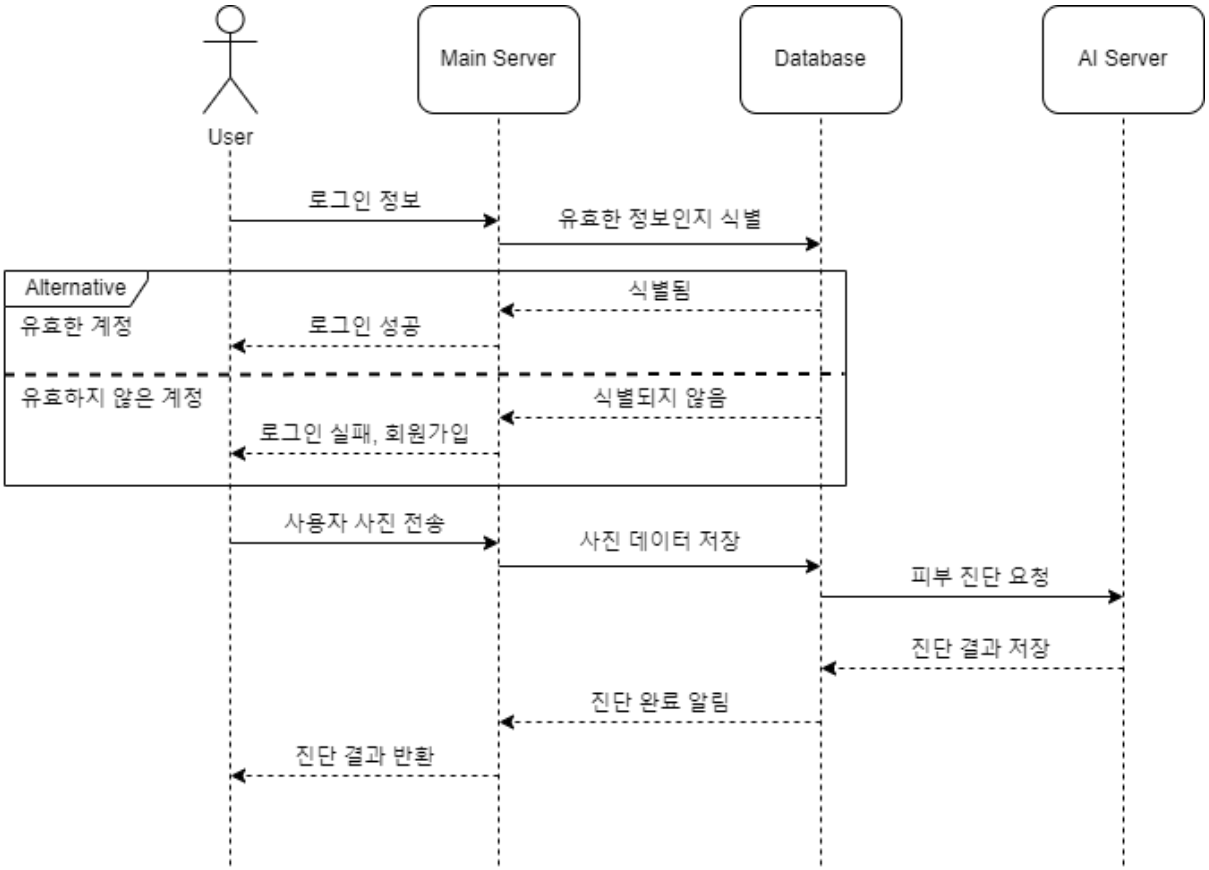
[Figure 1] Overall System Architecture

3.2.1. Context Diagram



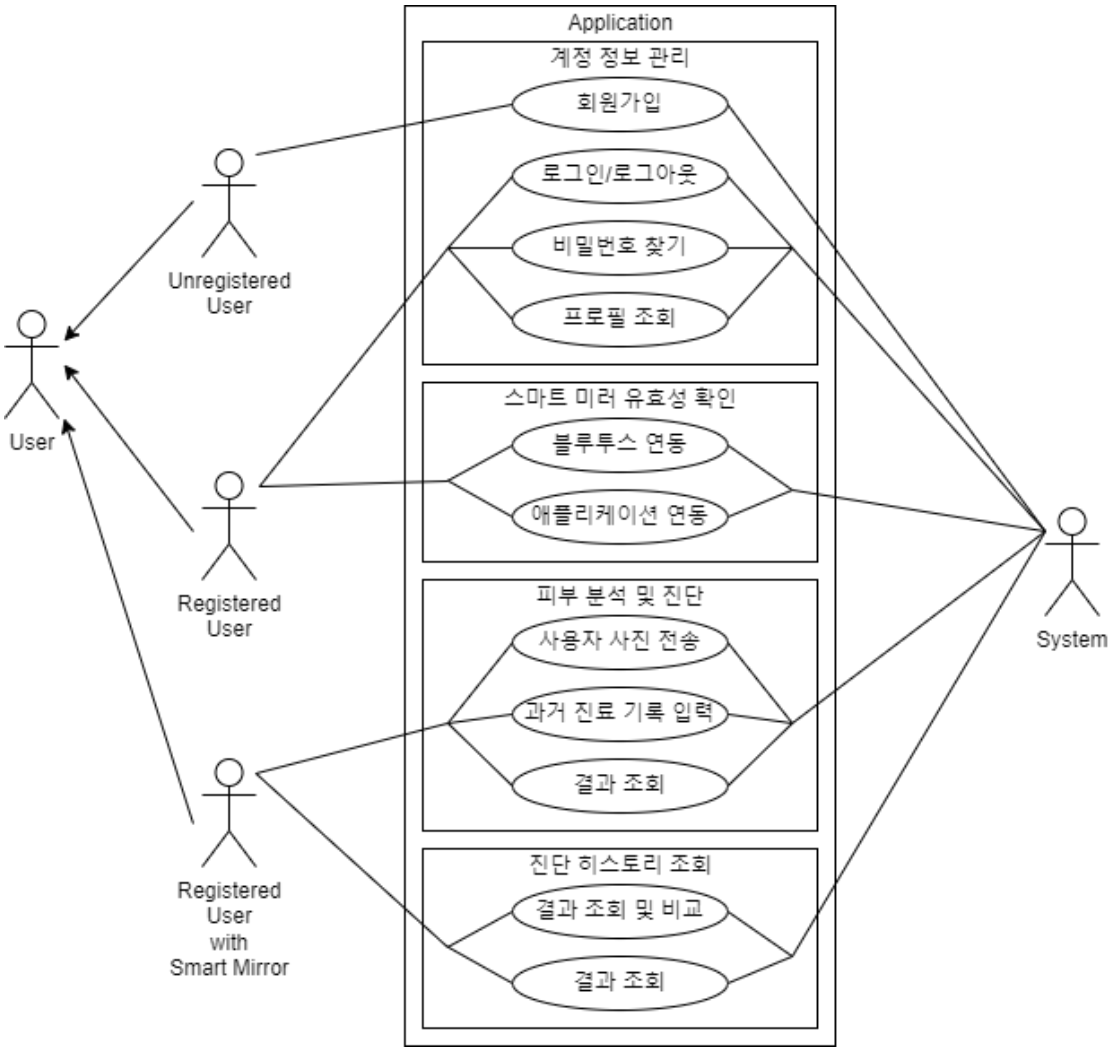
[Figure 2] Context Diagram

3.2.2. Sequence Diagram



[Figure 3] Sequence Diagram

3.2.3. Use Case Diagram



[Figure 4] Use Case Diagram

4. System Architecture – Frontend

4.1 Objectives

System architecture에서 사용자 인터페이스에 해당하는 Frontend 시스템을 이루는 컴포넌트들의 구성을 class diagram과 sequence diagram으로 도식화 및 설명한다. 각 subcomponent의 attribute와 method, component들 간의 관계에 대해 설명한다.

4.2 Subcomponents – Mobile Application

4.2.1 프로필(Profile)

프로필 컴포넌트는 사용자의 계정 및 프로필 정보를 관리한다. 로그인중인 사용자 계정의 프로필정보를 보여주고, 계정 및 프로필 정보를 수정할 수 있다.

4.2.1.1 Attributes

프로필 컴포넌트가 가지는 attribute는 다음과 같다.

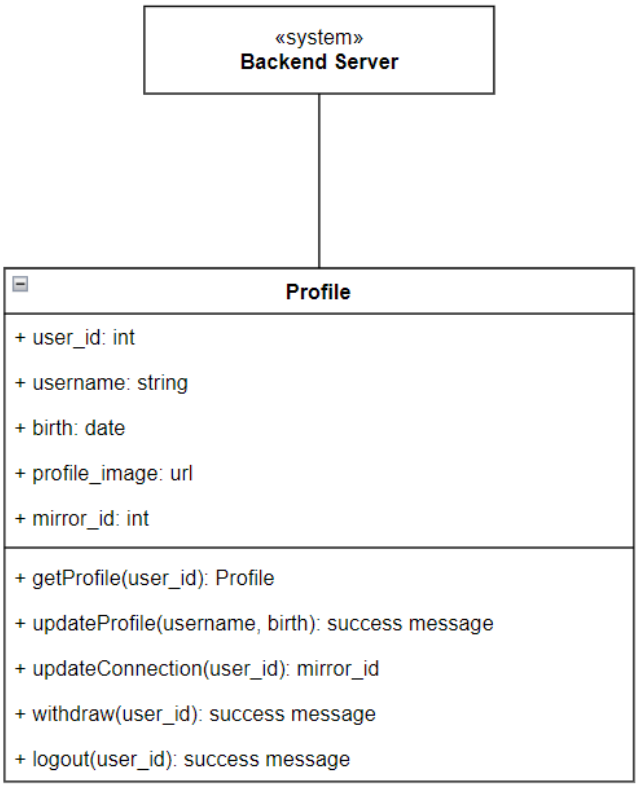
- user_id: 로그인중인 사용자 계정 ID
- username: 사용자의 본명
- birth: 사용자의 생년월일
- profile_image: 사용자의 프로필 이미지
- mirror_id: 사용자 계정과 연결된 스마트 미러의 고유 ID

4.2.1.2 Methods

프로필 컴포넌트가 가지는 methods는 다음과 같다.

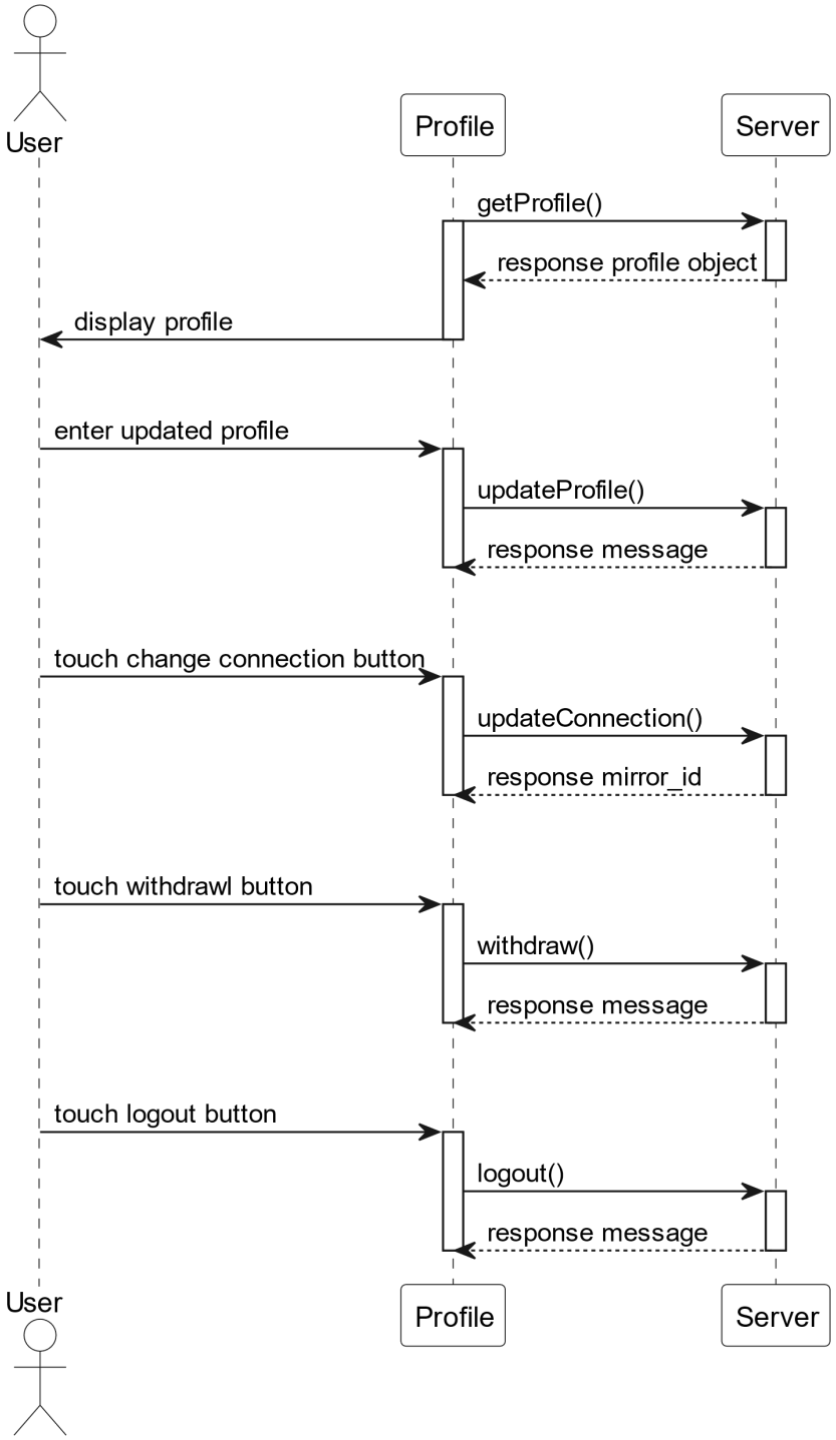
- GetProfile(): 로그인 중인 사용자의 프로필 정보를 가져온다.
- UpdateProfile(): 로그인 중인 사용자 프로필 정보 수정을 요청한다.
- UpdateConnection(): 로그인 중인 사용자 계정과 스마트 미러의 연동 정보 수정을 요청한다.
- Withdraw(): 로그인 중인 사용자 계정을 탈퇴한다.
- Logout(): 로그인 중인 사용자 계정에서 로그아웃한다.

4.2.1.3 Class Diagram



[Figure 5] Application Profile Class Diagram

4.2.1.4 Sequence Diagram



[Figure 6] Application Profile Sequence Diagram

4.2.2 피부 진단 기록(AnalysisRecord)

피부 진단 기록 컴포넌트는 사용자가 스마트 미러를 통해 분석한 피부 진단 결과 정보를 관리한다. 로그인 중인 계정으로 등록된 진단 기록의 목록과 특정 기록의 세부사항을 사용자에게 보여준다.

4.2.2.1 Attributes

피부 진단 기록 컴포넌트가 가지는 attribute는 다음과 같다.

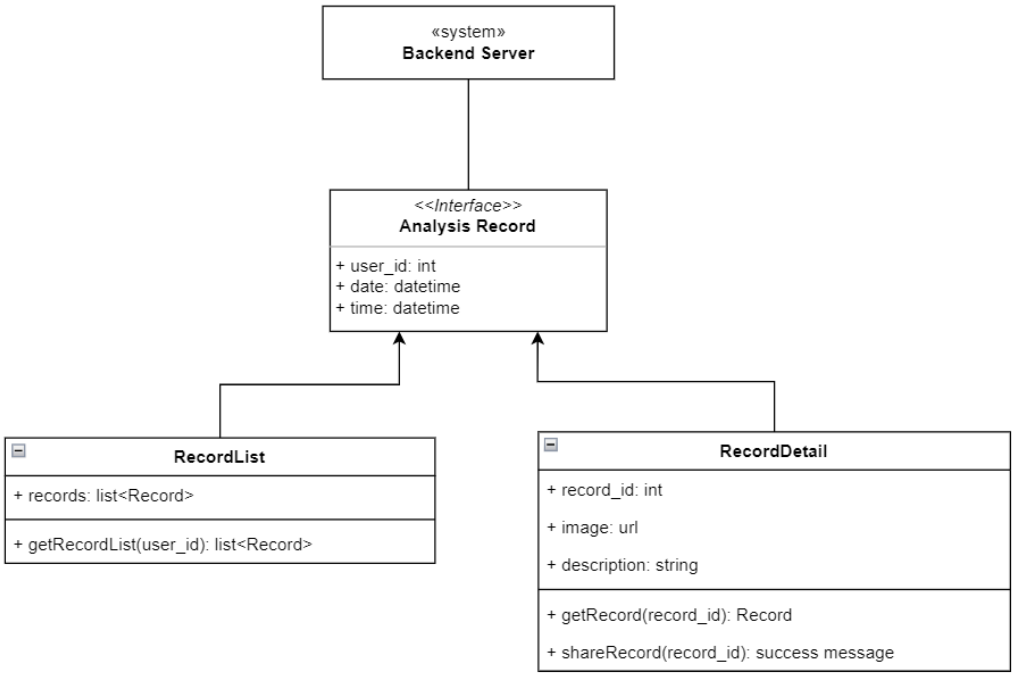
- user_id: 로그인 중인 사용자 계정 ID
- date: 피부 진단 기록 리스트(캘린더)에서 선택된 날짜
- time: 피부 진단 기록 리스트의 타임피커에서 선택된 시간
- records: 피부 진단 기록 리스트
- record_id: 세부사항을 보기위해 선택된 피부 진단 기록의 고유 ID
- image: 세부사항 진단 기록의 이미지
- description: 세부사항 진단 기록의 분석 결과 설명

4.2.2.2 Methods

피부 진단 기록 컴포넌트가 가지는 method는 다음과 같다.

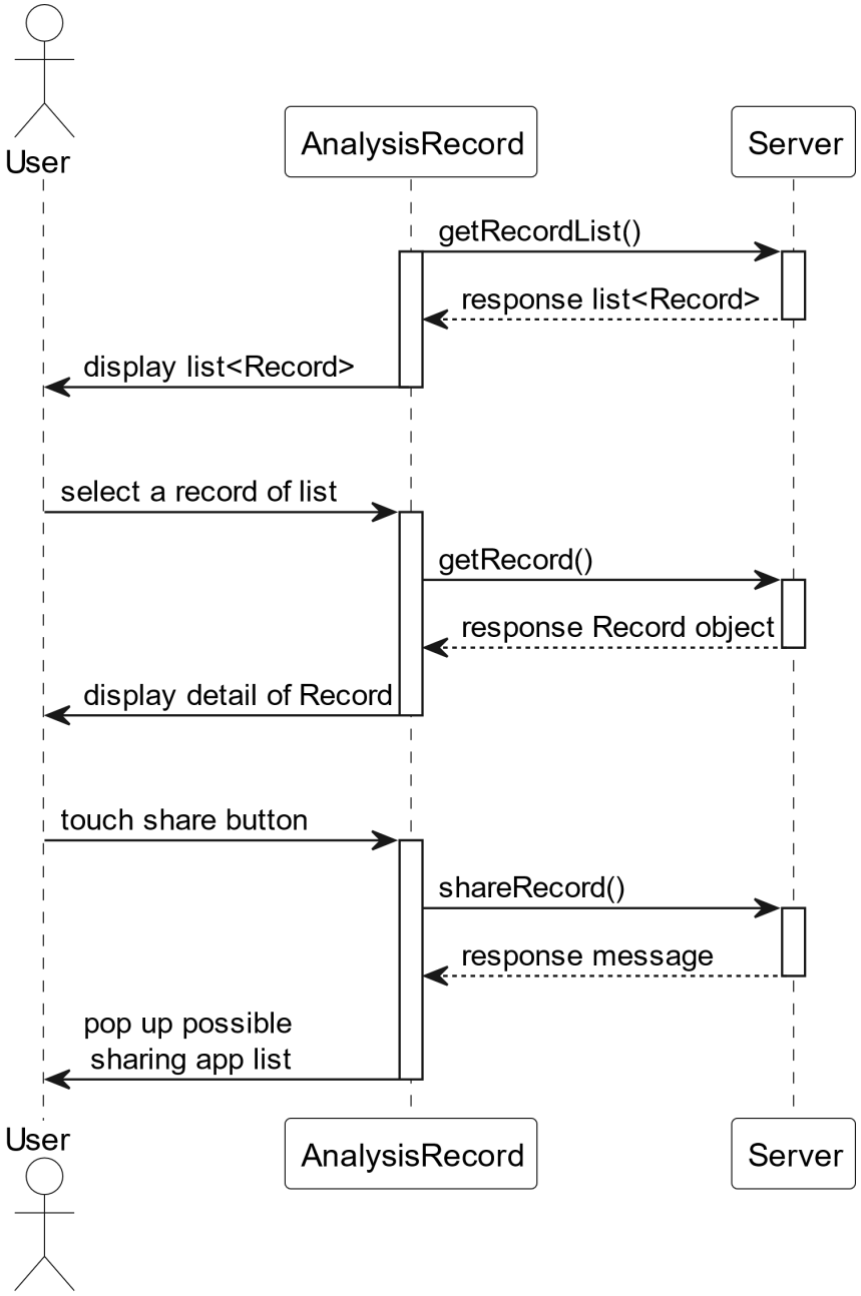
- GetRecordList(): 로그인 중인 계정으로 등록된 진단 기록의 목록을 가져온다.
- GetRecord(): 선택된 날짜와 시간에 해당하는 진단 기록의 세부 사항을 가져온다.
- ShareRecord(): 현재 확인하고 있는 진단 기록을 이미지로 기기에 저장 또는 외부로 공유를 요청한다.

4.2.2.3 Class Diagram



[Figure 7] Application Analysis Record Class Diagram

4.2.2.4 Sequence Diagram



[Figure 8] Application Analysis Record Sequence Diagram

4.2.3 피부 진단 기록 비교(Comparison)

피부 진단 기록 비교 컴포넌트는 사용자가 선택한 2개의 피부 진단 기록에 대한 비교 정보를 관리한다.

4.2.3.1 Attributes

피부 진단 기록 비교 컴포넌트가 가지는 attribute는 다음과 같다.

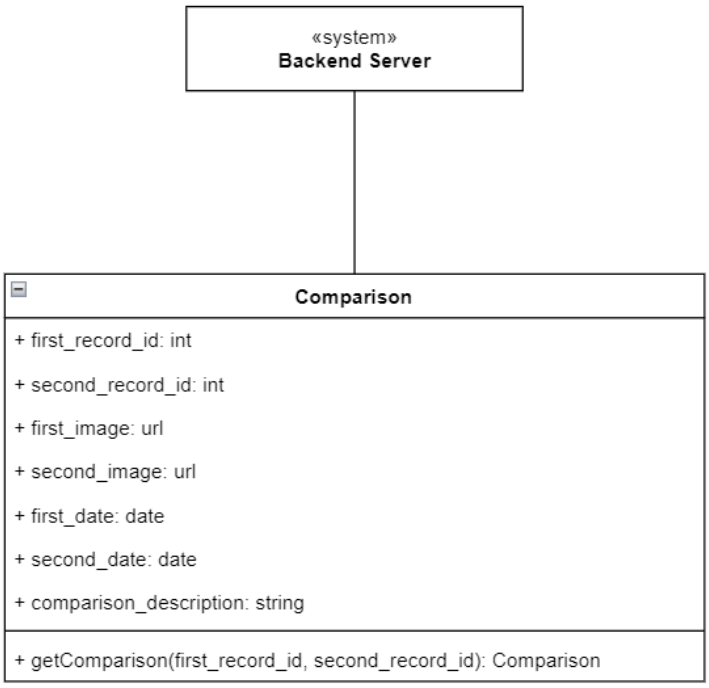
- first_record_id: 사용자가 첫번째로 선택한 진단 기록의 고유 ID
- second_record_id: 사용자가 두번째로 선택한 진단 기록의 고유 ID
- first_image: 첫번째 진단 기록의 이미지
- second_image: 두번째 진단 기록의 이미지
- first_date: 첫번째 진단 기록의 기록 날짜
- second_date: 두번째 진단 기록의 기록 날짜
- comparison_description: 2개의 진단 기록 비교 세부 사항

4.2.3.2 Methods

피부 진단 기록 비교 컴포넌트가 가지는 method는 다음과 같다.

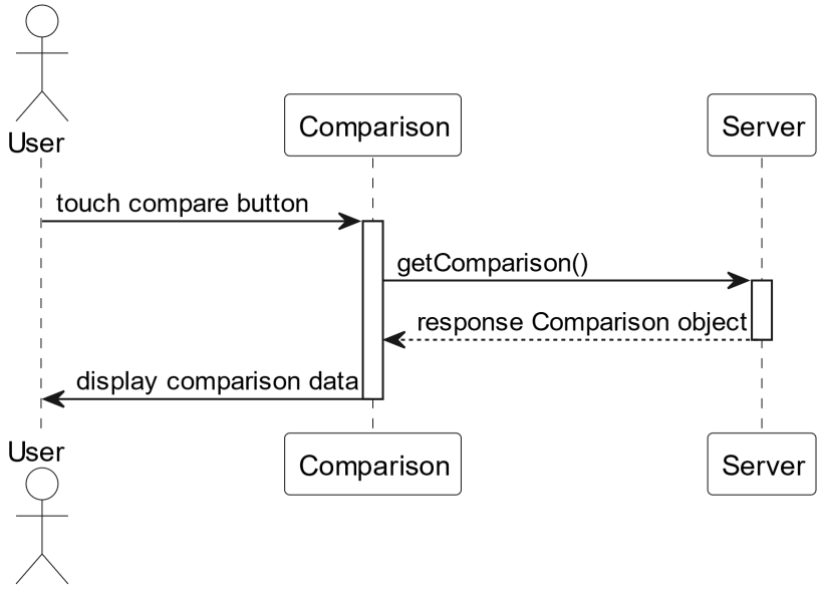
- GetComparison(): 선택한 2개의 진단 기록의 비교 세부 사항을 가져온다.

4.2.3.3 Class Diagram



[Figure 9] Application Analysis Record Comparison Class Diagram

4.2.3.4 Sequence Diagram



[Figure 10] Application Analysis Record Comparison Sequence Diagram

4.2.4 피부 상태 기록(StateRecord)

피부 상태 기록 컴포넌트는 사용자가 직접 입력하는 피부 상태 정보를 관리한다.

4.2.4.1 Attributes

피부 상태 기록 컴포넌트가 가지는 attribute는 다음과 같다.

- date: 피부 상태 정보를 저장할 날짜
- image: 사용자가 업로드한 피부 상태와 관련된 이미지
- description: 사용자가 입력한 피부 상태 세부사항

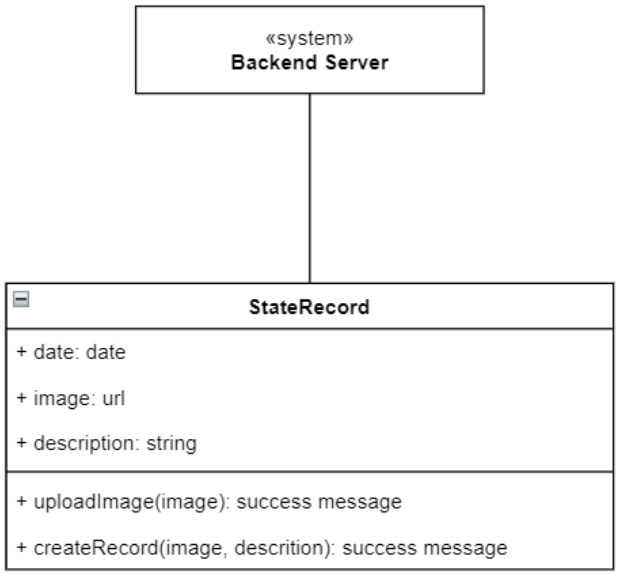
4.2.4.2 Methods

피부 상태 기록 컴포넌트가 가지는 method는 다음과 같다.

- UploadImage(): 기기에서 피부 상태 데이터로 저장할 이미지를 업로드한다.

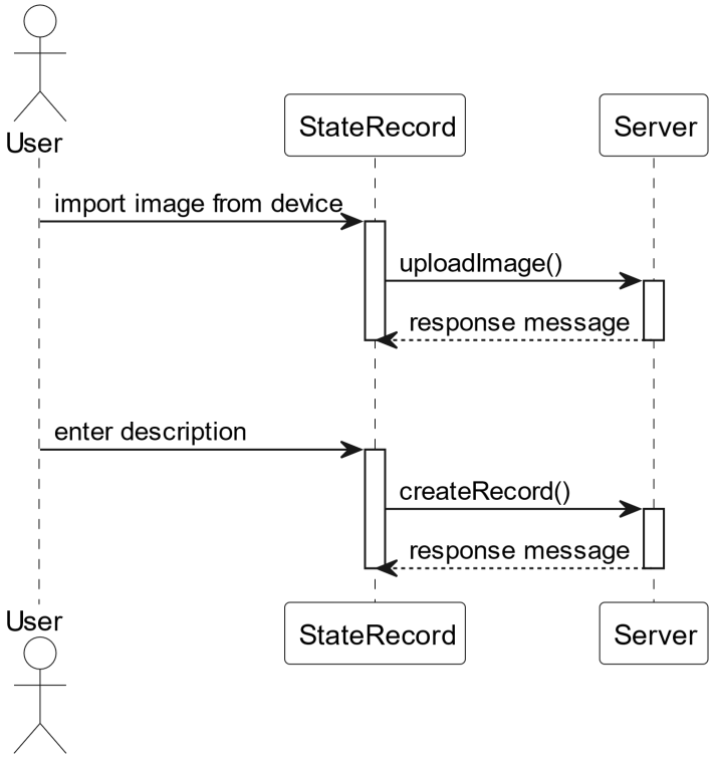
- CreateRecord(): 모든 데이터를 입력한 뒤 피부 상태 기록 데이터를 저장한다.

4.2.4.3 Class Diagram



[Figure 11] Application State Record Class Diagram

4.2.4.4 Sequence Diagram



[Figure 12] Application State Record Sequence Diagram

4.3 Subcomponents – Smart Mirror

4.3.1 피부 진단(Analysis)

스마트 미러에 내장되어있는 카메라를 통해 피부 상태를 측정하고 서버로 피부 상태 분석 및 진단을 요청한다.

4.3.1.1 Attributes

피부 진단 컴포넌트가 가지는 attribute는 다음과 같다.

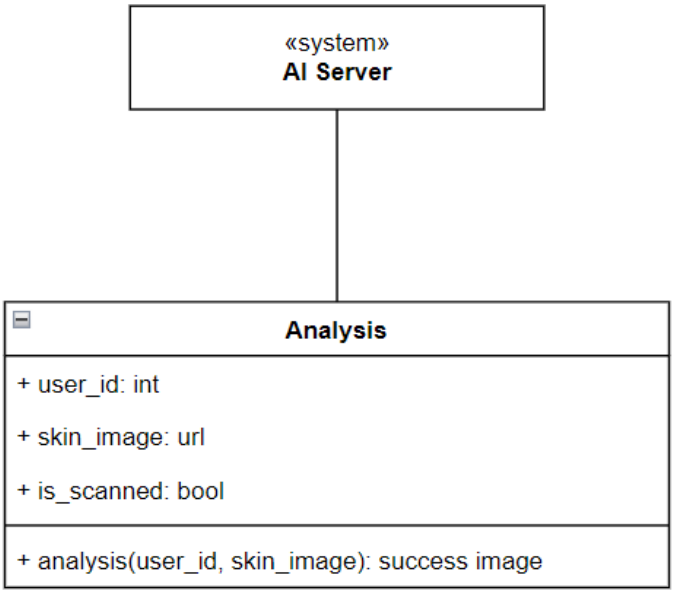
- user_id: 로그인 중인 사용자 계정 ID
- skin_image: 카메라를 통해 360도 촬영한 피부 이미지
- is_scanned: 촬영된 피부 이미지의 방향, 조명이 적절한지 나타내는 플래그

4.3.1.2 Methods

피부 진단 컴포넌트가 가지는 method는 다음과 같다.

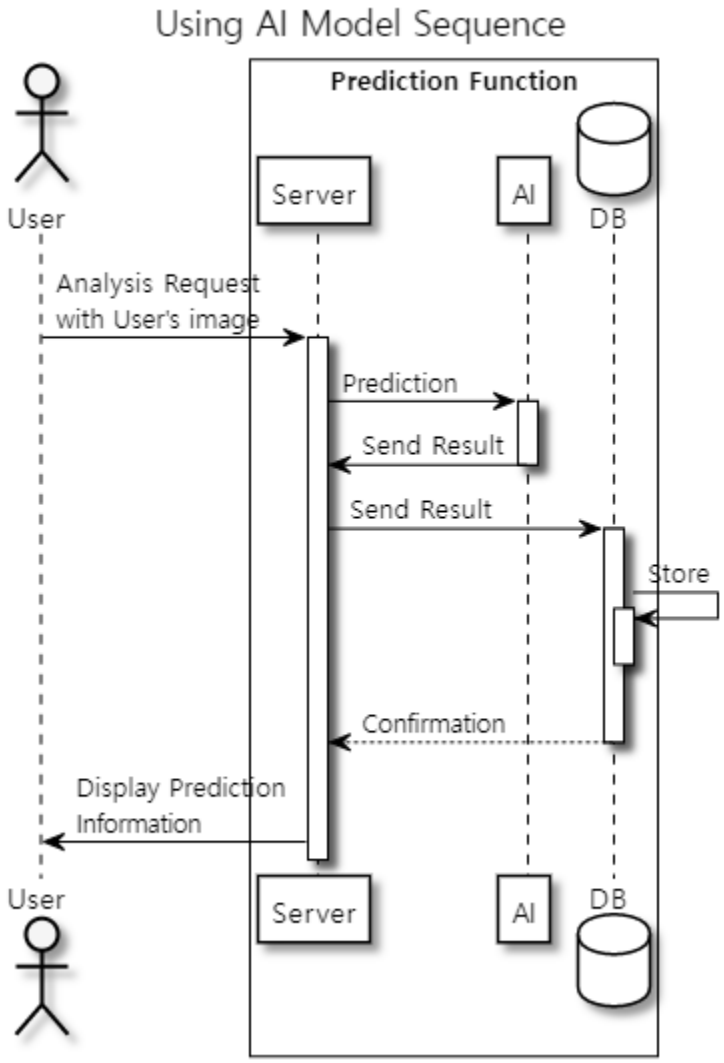
- Analysis(): 촬영된 피부 이미지의 피부 상태 진단 및 분석을 서버에 요청한다.

4.3.1.3 Class Diagram



[Figure 13] Smart Mirror Analysis Class Diagram

4.3.1.4 Sequence Diagram



[Figure 14] Smart Mirror Analysis Sequence Diagram

4.3.2 피부 진단 결과 (Result)

카메라를 통해 측정한 피부 상태에 대한 분석 및 진단 결과를 사용자에게 보여준다.

4.3.2.1 Attributes

피부 진단 결과 컴포넌트가 가지는 attribute는 다음과 같다.

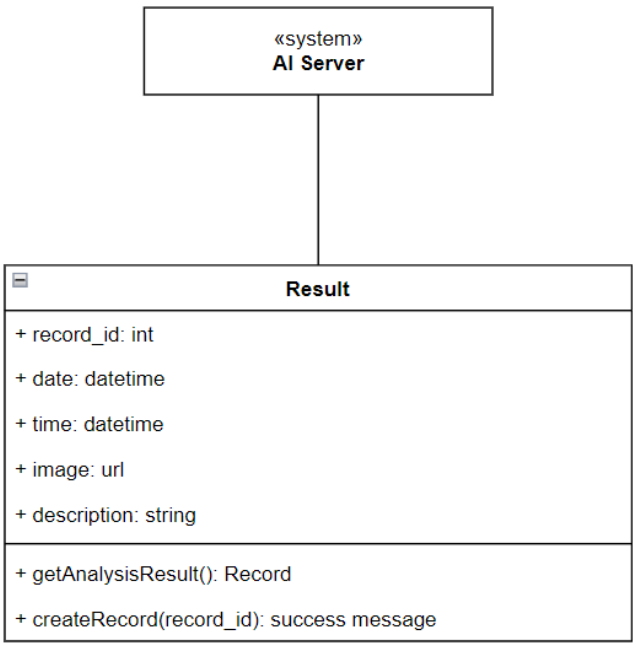
- record_id: 저장된 피부 진단 결과 데이터의 고유 ID
- date: 피부 진단 결과가 저장된 날짜
- time: 피부 진단 결과가 저장된 시간
- image: 피부 진단 결과의 측정 피부 이미지
- description: 피부 진단 결과 세부 사항

4.3.2.2 Methods

피부 진단 결과 컴포넌트가 가지는 method는 다음과 같다.

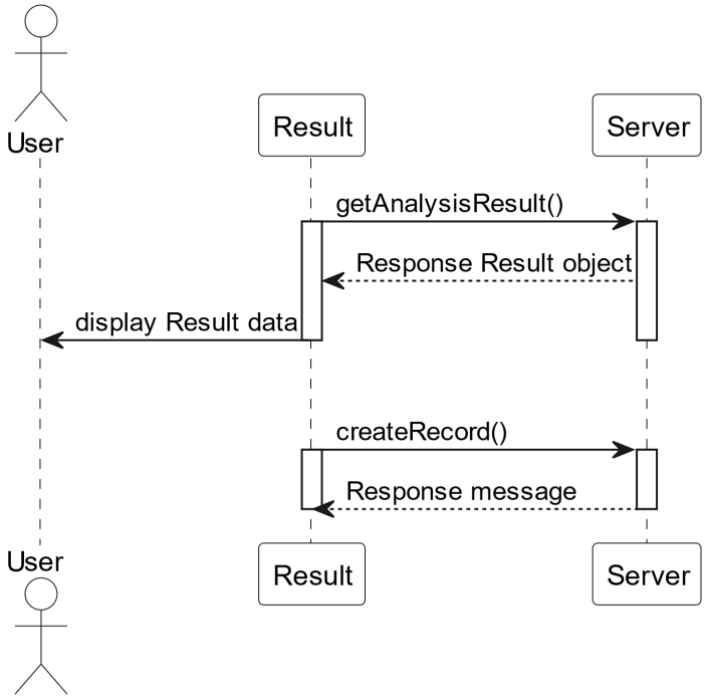
- GetAnalysisResult(): 피부 진단 결과 데이터를 가져온다.
- CreateRecord(): 확인한 진단 결과 데이터를 서버에 저장한다.

4.3.2.3 Class Diagram



[Figure 15] Smart Mirror Analysis Result Class Diagram

4.3.2.4 Sequence Diagram



[Figure 16] Smart Mirror Analysis Result Sequence Diagram

4.4 Protocols used in Frontend part

모바일 어플리케이션 및 스마트 미러의 Frontend 시스템은 Backend 서버, AI 시스템과 HTTP 통신을 한다. Request 및 Response는 모두 JSON 포맷을 사용한다.

4.4.1 프로필

4.4.1.1 GetProfile

- Request

[Table 1] Get profile request

Attribute	Detail
URI	/Api/profile/get
Method	Get

Parameter	User_id	User id
Header	Authorization	User authentication

- Response

[Table 2] Get profile response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
	HTTP 404 Not foud	
Success Response Body	User_id	User id
	Username	User name
	Birth	User birth
	Profile_image	User profile image url
	Mirror_id	Connected mirror id
Failure Response Body	Message	Message "Error"

4.4.1.2 UpdateProfile

- Request

[Table 3] Update profile request

Attribute	Detail

URI	/Api/profile/update	
Method	Update	
Body	username	Updated username
	birth	Updated birth
Header	Authorization	User authentication

- Response

[Table 4] Update profile response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad request	
	HTTP 404 Not found	
Success Response Body	Message	Message "Update success"
Failure Response Body	Message	Message "Error"

4.4.1.3 UpdateConnection

- Request

[Table 5] Update connection request

Attribute	Detail
URI	/Api/profile/connection

Method	Update	
Body	User_id	User id
Header	Authorization	User authentication

- Response

[Table 6] Update connection response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad request	
	HTTP 404 Not found	
Success Response Body	Mirror_id	Updated connected mirror id
Failure Response Body	Message	Message "Error"

4.4.1.4 Withdraw

- Request

[Table 7] Withdraw request

Attribute	Detail	
URI	/Api/profile/withdraw	
Method	Post	
Body	User_id	User id

Header	Authorization	User authentication

- Response

[Table 8] Withdraw response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad request	
	HTTP 404 Not found	
Success Response Body	Message	Message "Withdraw success"
Failure Response Body	Message	Message "Error"

4.4.1.5 Logout

- Request

[Table 9] Logout request

Attribute	Detail	
URI	/Api/profile/logout	
Method	Post	
Body	User_id	User id
Header	Authorization	User authentication

- Response

[Table 10] Logout response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad request	
	HTTP 404 Not found	
Success Response Body	Message	Message "Logout success"
Failure Response Body	Message	Message "Error"

4.4.2 피부 진단 기록

4.4.2.1 GetRecordList

- Request

[Table 11] Get record list request

Attribute	Detail	
URI	/Api/record/get_list	
Method	Get	
Parameter	User_id	User id
Header	Authorization	User authentication

- Response

[Table 12] Get record list response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad request	
	HTTP 404 Not found	
Success Response Body	Record list	Record list of the user
Failure Response Body	Message	Message "Error"

4.4.2.2 GetRecord

- Request

[Table 13] Get record request

Attribute	Detail	
URI	/Api/record/get	
Method	Get	
Parameter	Record_id	Record id
Header	Authorization	User authentication

- Response

[Table 14] Get record response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad request	
	HTTP 404 Not foud	
Success Response Body	Record_id	Record id
	date	Recorded date
	time	Recorded time
	image	Record image url
	description	Record analysis description text
Failure Response Body	Message	Message "Error"

4.4.2.3 ShareRecord

- Request

[Table 15] Share record request

Attribute	Detail	
URI	/Api/record/share	
Method	Post	
Body	Record_id	Record id
Header	Authorization	User authentication

- Response

[Table 16] Share record response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad request	
	HTTP 404 Not found	
Success Response Body	Message	Message "Success"
Failure Response Body	Message	Message "Error"

4.4.3 피부 진단 기록 비교

4.4.3.1 GetComparison

- Request

[Table 17] Get comparison request

Attribute	Detail	
URI	/Api/comparison/get	
Method	Get	
Parameter	Firste_record_id	First selected record

		id
	Second_record_id	Second selected record id
Header	Authorization	User authentication

- Response

[Table 18] Get comparison response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad reqeust	
	HTTP 404 Not found	
Success Response Body	First_record_id	First record id
	Second_record_id	Second record id
	First_image	First record image url
	Second_image	Second record image url
	First_date	First record recorded date
	Second_date	Second record recorded date

	Comparison_description	Compared anaysis text
Failure Response Body	Message	Message "Error"

4.4.4 피부 상태 기록

4.4.4.1 UploadImage

- Request

[Table 19] Upload image request

Attribute	Detail	
URI	/Api/state/upload_image	
Method	Post	
Body	Image	User upload skin state image
Header	Authorization	User authentication

- Response

[Table 20] Upload image response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 Bad request

	HTTP 404 Not found	
Success Response Body	Message	Message "Success"
Failure Response Body	Message	Message "Error"

4.4.4.2 CreateRecord

- Request

[Table 21] Create record request

Attribute	Detail	
URI	/Api/state/create	
Method	Post	
Body	image	Uploaded skin image url
	description	Description of skin state
Header	Authorization	User authentication

- Response

[Table 22] Create record response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 Bad request
	HTTP 404 Not found

Success Response Body	Message	Message "Success"
Failure Response Body	Message	Message "Error"

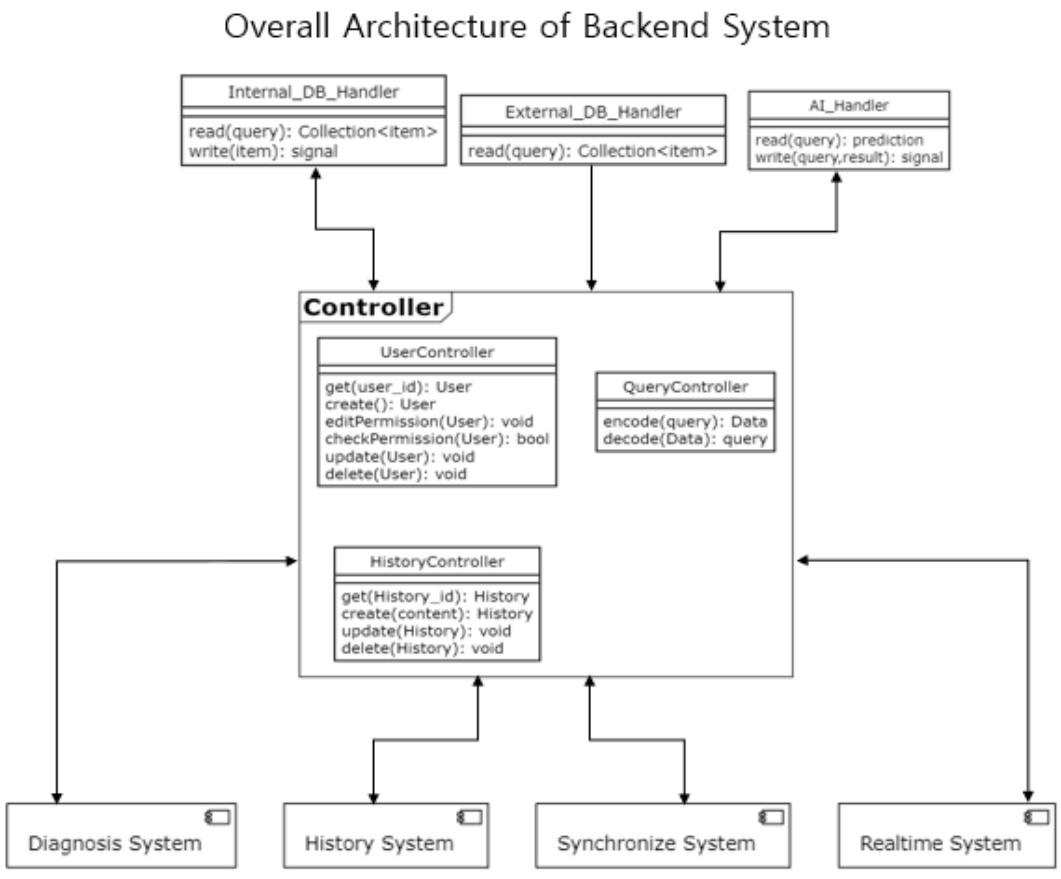
5. System Architecture – Backend

5.1. Objectives

이 섹션은 모바일 어플리케이션과 AI 시스템과의 상호작용을 토대로 백엔드의 구조를 나타낸다.

5.2. Overall Architecture

백엔드 시스템의 전반적인 아키텍처는 아래 그림과 같다. 백엔드 시스템은 Node.js를 통해 구현되며 System, Handler, 그리고 Controller로 구성된다.



[Figure 17] Class Diagram of Overall Backend Architecture

5.2.1. System

각 System은 세부적인 기능을 처리하는 역할을 하며 Realtime System, Diagnosis System, History System, Synchronize System이 있다.

5.2.2. Handler

Handler는 Internal DB Handler와 External DB Handler, AI Handler가 있다.

- Internal DB Handler: 내부 데이터베이스에 있는 History 데이터나 질병 정보 데이터를 읽고 쓸 수 있도록 하는 인터페이스 역할을 한다.

- External DB Handler: 외부 데이터베이스로부터 날짜와 시간, 날씨 데이터를 읽고 쓸 수 있도록 하는 인터페이스 역할을 한다.

- AI Handler: AI로 데이터를 보내서 진단 결과를 받거나 AI모델을 학습시킬 수 있도록 하는 인터페이스 역할을 한다.

5.2.3. Controller

Controller는 System과 Handler사이에 있으며 User Controller, Query Controller, History Controller가 있다.

5.3. Subcomponents - Backend

5.3.1. Realtime System

Realtime System은 실시간 날짜 정보, 날씨와 같이 내부 데이터베이스에서 가져올 수 없는 데이터들을 외부 데이터베이스를 통해 가져와서 고객들이 해당 정보를 확인할 수 있게 한다.

5.3.1.1. Attributes

Realtime system이 가지는 attribute는 다음과 같다.

- date: 현재 날짜

- time: 현재 시간

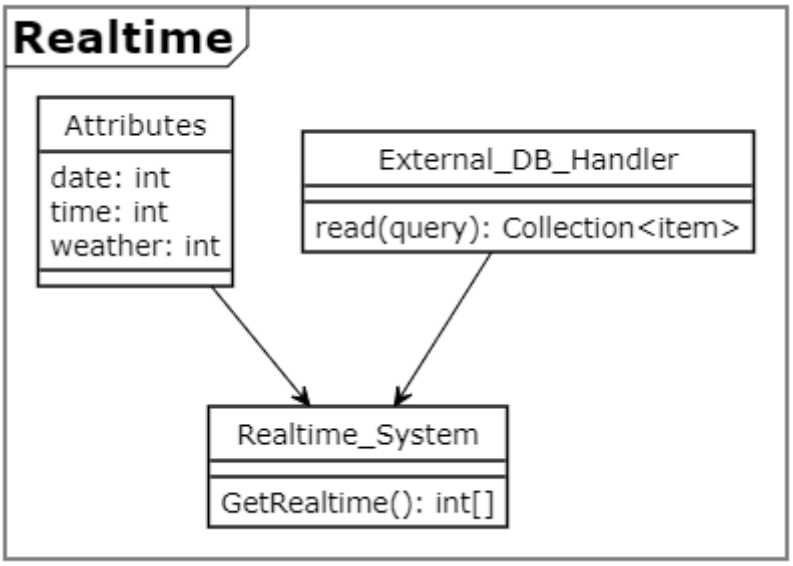
- weather: 현재 날씨

5.3.1.2. Methods

Realtime system이 가지는 method는 다음과 같다.

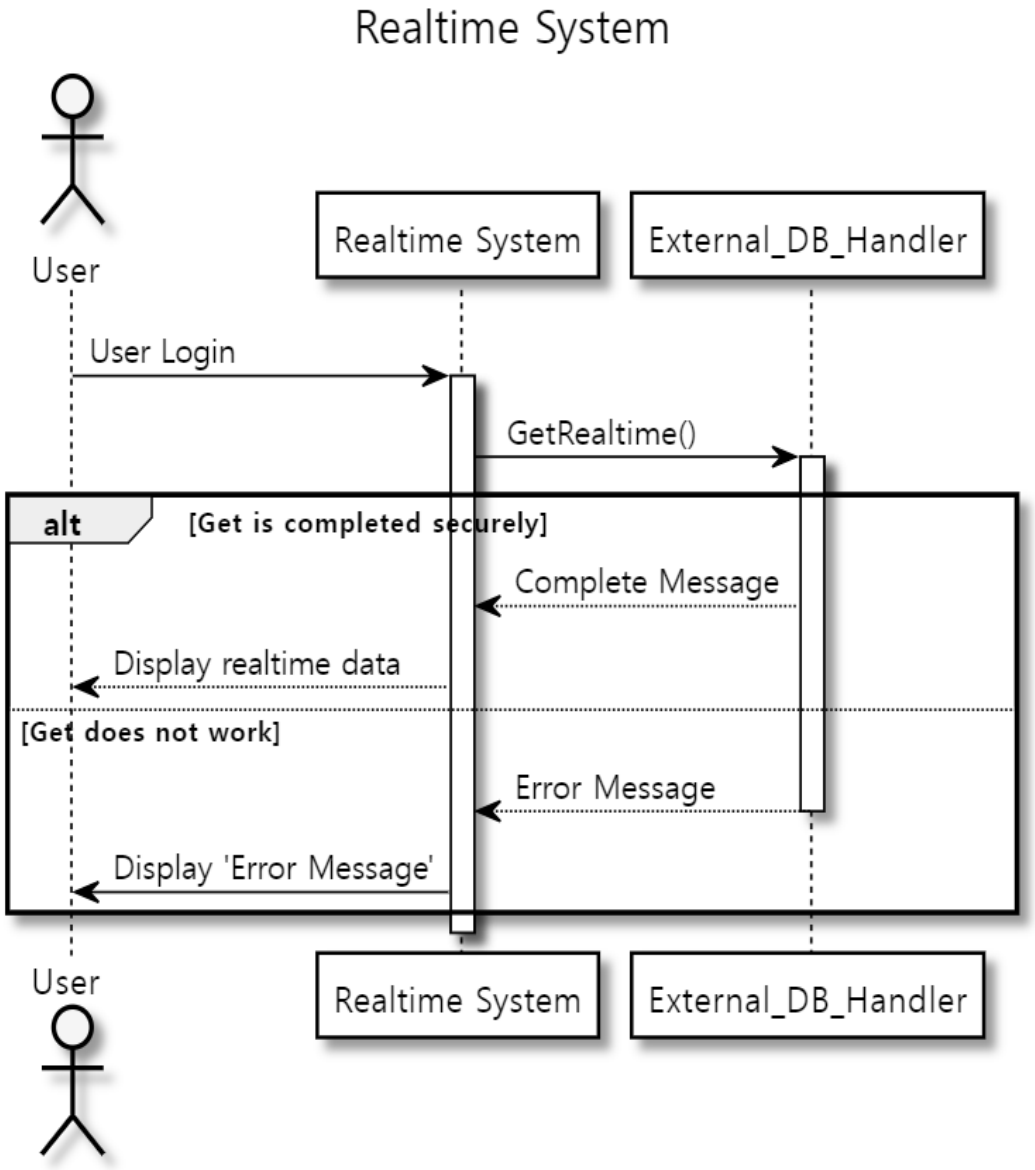
- GetRealtime(): 현재 날짜, 시간, 날씨에 해당되는 정보를 외부 데이터베이스에서 가져온다.

5.3.1.3. Class Diagram



[Figure 18] Realtime System Class Diagram

5.3.1.4. Sequence Diagram



[Figure 19] Register System Sequence Diagram

5.3.2. Diagnosis System

Diagnosis System은 계정 고유 번호를 확인해서 해당 유저 정보에만 접근할 수 있도록 한다. 그리고 프론트엔드로부터 이미지 데이터를 입력받는다. 해당 데이터를 AI 시스템으로 전송해서 피부 진단을 실행하게 하고 그 결과

를 이미지와 함께 프론트엔드로 전달한다. 추후에 들어올 실제 진단 결과를 위해서 이미지 데이터는 서버에 저장하고 데이터베이스에는 이미지 데이터의 경로를 저장한다.

5.3.2.1. Attributes

Diagnosis system이 가지는 attribute는 다음과 같다.

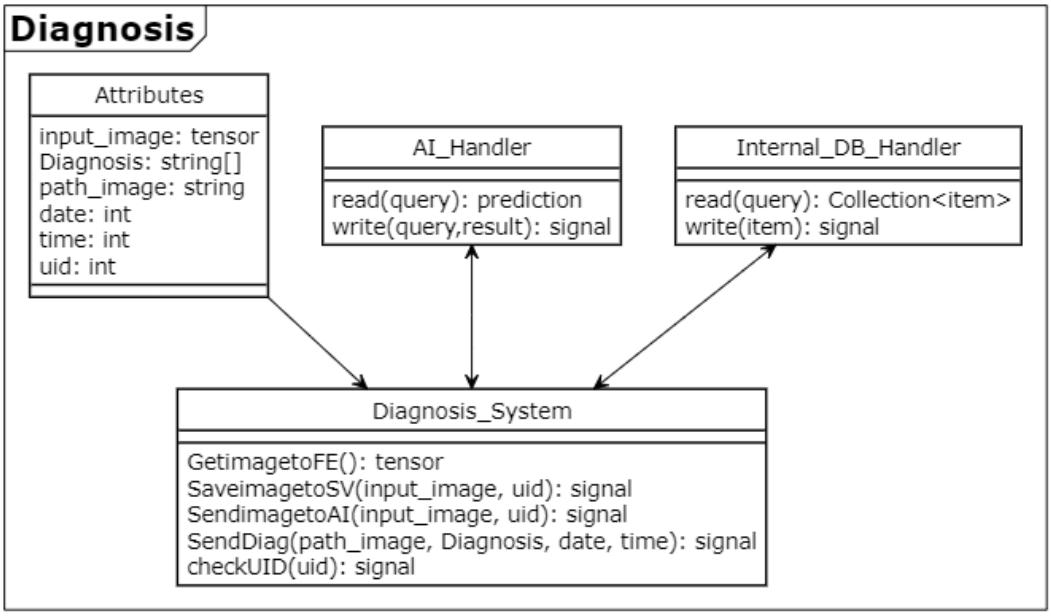
- input_image: 입력받은 이미지 데이터
- Diagnosis: 입력받은 이미지 데이터를 AI에 넣어서 나온 피부 진단 결과 데이터
- path_image: 입력받은 이미지 데이터를 저장한 서버의 경로
- date: 현재 날짜
- time: 현재 시간
- uid: 사용자 계정의 고유 번호

5.3.2.2. Methods

Diagnosis system이 가지는 method는 다음과 같다.

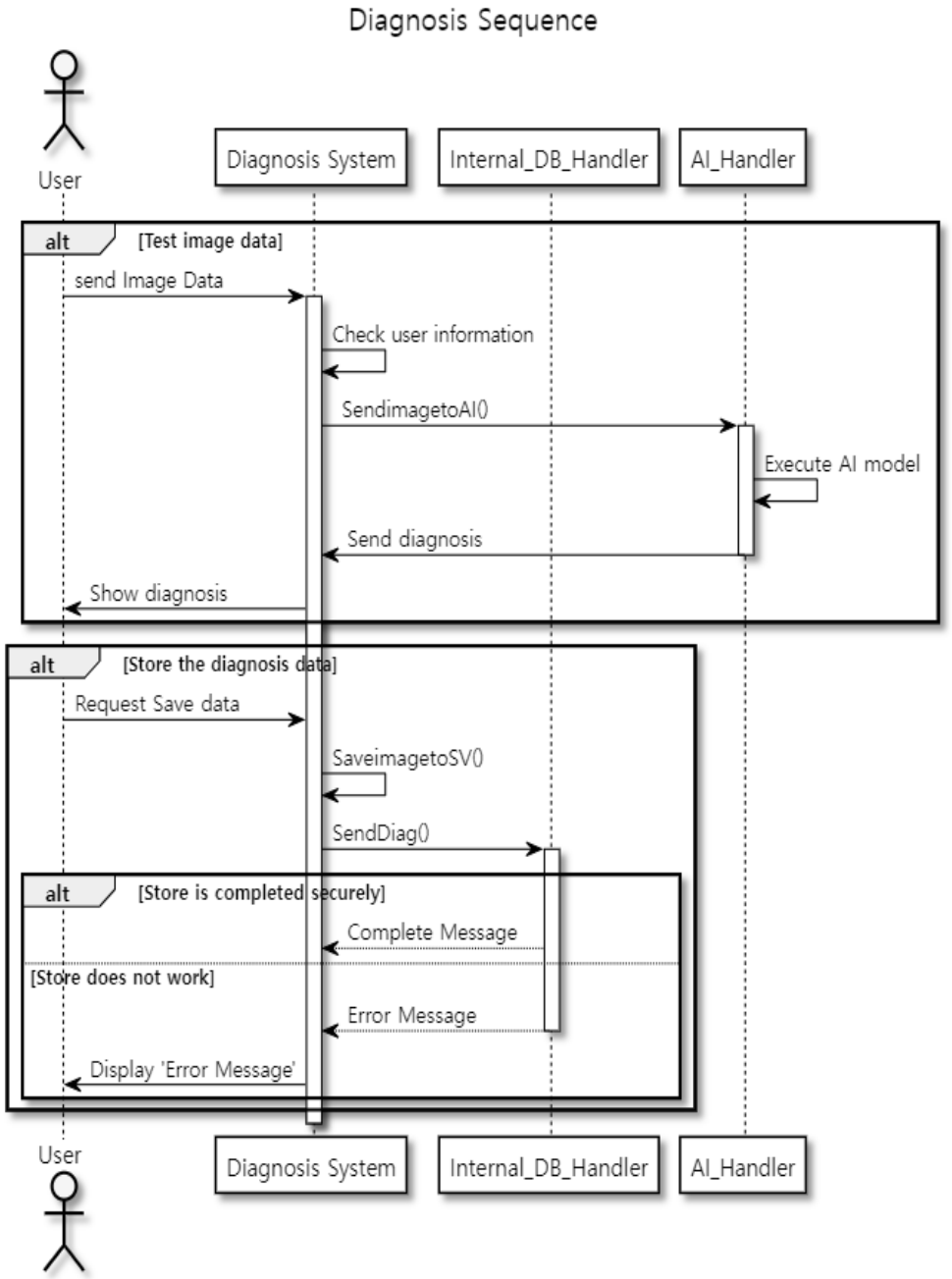
- GetimagetoFE(): 프론트엔드에서 이미지를 가져온다.
- SaveimagetoSV(): 이미지를 서버에 저장하고 저장된 서버의 경로를 가져온다.
- SendimagetoAI(): 이미지를 AI 시스템에게 전달하고 AI 시스템에서 진단 데이터를 가져온다.
- SendDiag(): 이미지 데이터의 경로와 진단 데이터를 데이터베이스에 저장한다.
- checkUID(): 계정의 고유 정보를 확인한다.

5.3.2.3. Class Diagram



[Figure 20] Diagnosis System Class Diagram

5.3.2.4. Sequence Diagram



[Figure 21] Diagnosis System Sequence Diagram

5.3.3. History System

History System은 과거 데이터를 데이터베이스에서 가져와서 사용자가 볼

수 있도록 한다. 또한 과거에 AI 시스템이 진단한 이미지에 대해 의사가 진단한 결과를 넣어주어서 AI모델이 해당 데이터를 학습할 수 있도록 한다.

5.3.3.1. Attributes

History system이 가지는 attribute는 다음과 같다.

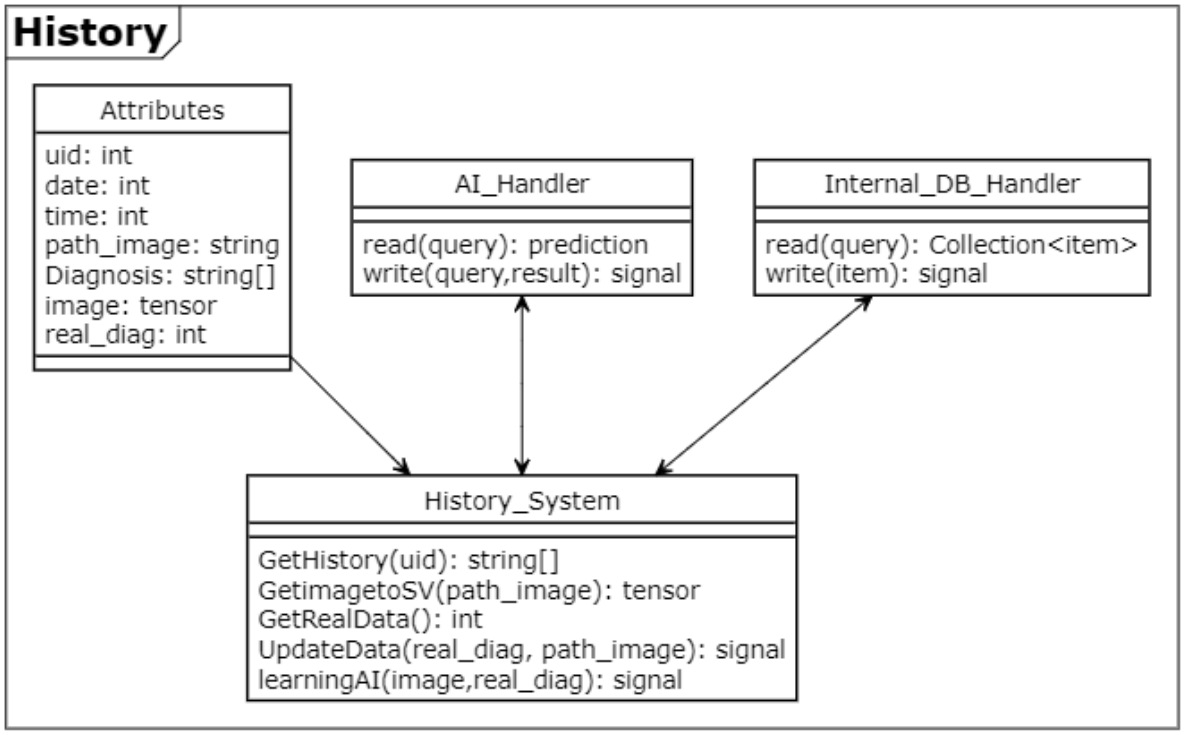
- uid: 사용자 계정의 고유 번호
- date: 현재 날짜
- time: 현재 시간
- path_image: 이미지 데이터가 저장된 위치 경로
- Diagnosis: 특정 이미지 데이터에 대해 AI모델이 예측한 진단 데이터
- image: 특정 이미지 데이터
- real_diag: 실제 의사가 진단한 정보 데이터

5.3.3.2. Methods

History system이 가지는 method는 다음과 같다.

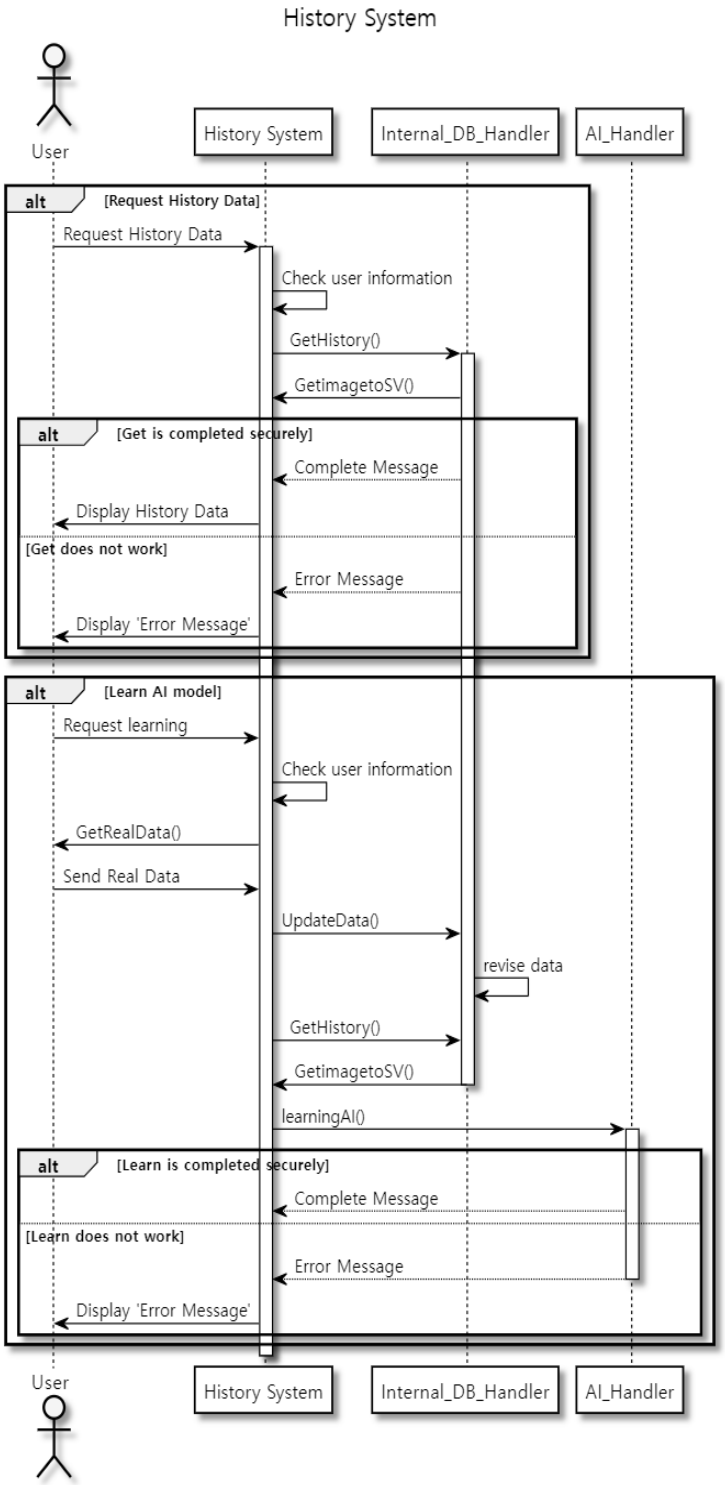
- GetHistory(): 사용자 계정의 고유 번호를 통해 과거 스마트 미러의 진단 기록에 대한 데이터를 불러온다.
- GetimagetoSV(): 과거 이미지 데이터를 가져온다.
- GetRealData(): 과거 스마트 미러의 진단 기록이 있는 이미지에 대해 의사가 진단한 데이터를 프론트엔드로부터 불러온다.
- UpdateData(): 의사 진단 데이터를 이용해서 과거 데이터를 최신화한다.
- learningAI(): 과거 스마트 미러의 진단 기록이 있는 이미지와 의사가 진단한 데이터를 AI 시스템에 전달하여 AI 시스템이 해당 데이터로 학습하도록 한다.

5.3.3.3. Class Diagram



[Figure 22] History System Class Diagram

5.3.3.4. Sequence Diagram



[Figure 23] History System Sequence Diagram

5.3.4. Synchronize System

Synchronize System은 모바일 어플리케이션이 스마트 미러와 같은 정보를 볼 수 있도록 동기화 해주는 시스템이다. 로그인된 계정의 고유 번호를 통해 서버와 데이터베이스에 접근해서 현재 데이터 정보를 모바일 기기에 전달해 준다.

5.3.4.1. Attributes

- uid: 사용자 계정의 고유 번호

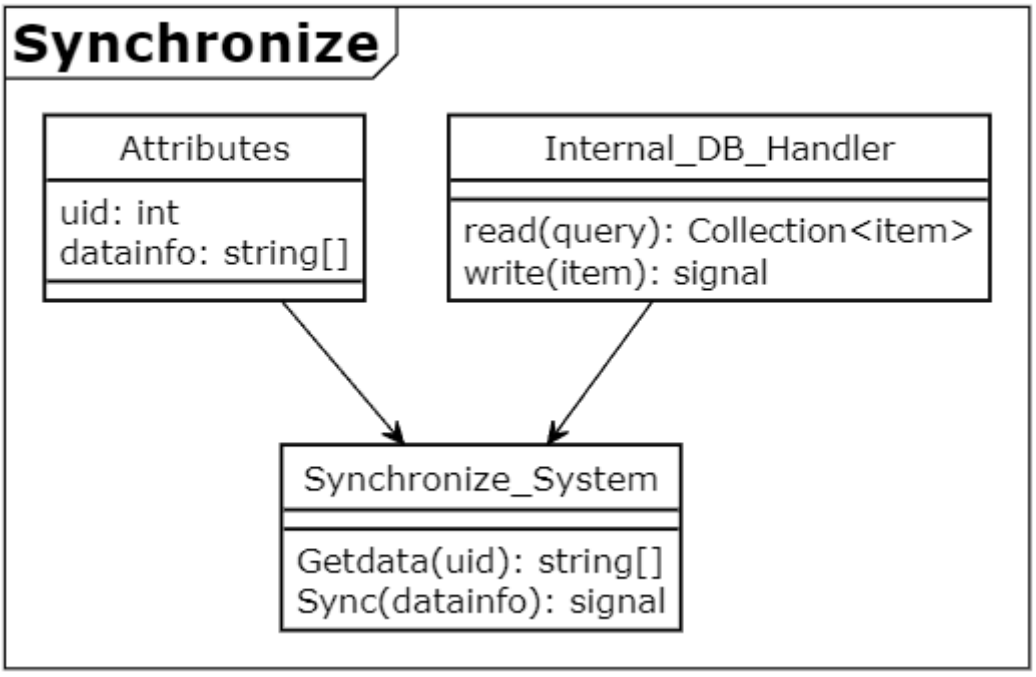
- datainfo: 사용자의 데이터 정보

5.3.4.2. Methods

- Getdata(): uid를 통해 해당 사용자의 데이터에 접근해서 데이터 정보를 불러온다.

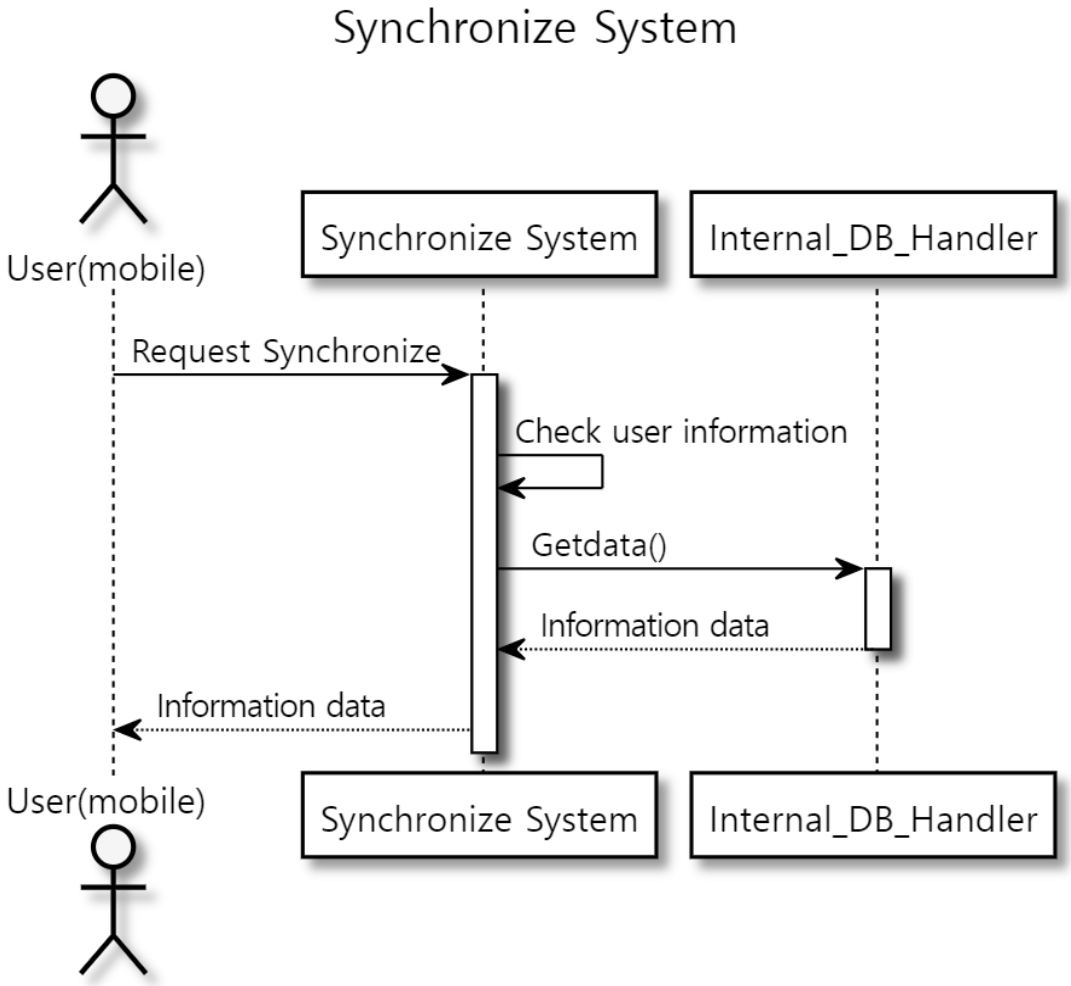
- Sync(): 불러온 데이터 정보를 모바일 어플리케이션에 전달한다.

5.3.4.3. Class Diagram



[Figure 24] Synchronize System Class Diagram

5.3.4.4. Sequence Diagram



[Figure 25] Synchronize System Sequence Diagram

5.4. Protocols used in Backend

스마트 미러의 백엔드 시스템은 모바일 어플리케이션, 스마트 미러의 프론트엔드 그리고 AI 시스템과 HTTP 통신을 한다. Request 및 Response는 모두 JSON 형식을 사용한다.

5.4.1. Realtime System

5.4.1.1. GetRealtime

- Request

[Table 23] GetRealtime request

Attribute	Detail	
URI	/Realtime/GetRealtime	
Method	GET	
Header	Authorization	User Authentication

- Response

[Table 24] GetRealtime response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	Message	Message "Logout Success"

Failure Response Body	Message	Message "Error"

5.4.2. Diagnosis System

5.4.2.1. GetimagetoFE

- Request

Table 25 GetimagetoFE request

Attribute	Detail	
URI	/Diagnosis/GetimagetoFE	
Method	GET	
Header	Authorization	User Authentication

- Response

[Table 26] GetimagetoFE response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request, overlap)
	HTTP 403 (Forbidden)
	HTTP 404 (Not found)

Success Response Body	Image	input image
Failure Response Body	Message	Message "Error"

5.4.2.2. SaveimagetoSV

- Request

[Table 27] SaveimagetoSV request

Attribute	Detail	
URI	/Diagnosis/SaveimagetoSV	
Method	GET	
Parameter	input_image	input image
	uid	user id serial number
Header	Authorization	User Authentication

- Response

[Table 28] SaveimagetoSV response

Attribute	Detail
Success Code	HTTP 200 OK

Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	path_image	path for image saved
	Message	Message "Success"
Failure Response Body	Message	Message "Error"

5.4.2.3. SendimagetoAI

- Request

[Table 29] SendimagetoAI request

Attribute	Detail	
URI	/Diagnosis/SendimagetoAI	
Method	GET	
Parameter	input_image	input image
	uid	user id serial number
Header	Authorization	User Authentication

- Response

[Table 30] SendimagetoAI response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	Diagnosis	Diagnosis information about image
	Message	Message "Success"
Failure Response Body	Message	Message "Error"

5.4.2.4. SendDiag

- Request

[Table 31] SendDiag request

Attribute	Detail
URI	/Diagnosis/SendDiag

Method	GET	
Parameter	path_image	path for image saved
	Diagnosis	Diagnosis information about image
	date	current date
	time	current time
Header	Authorization	User Authentication

- Response

[Table 32] SendDiag response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	Message	Message "Success"

Failure Response Body	Message	Message "Error"

5.4.2.5. checkUID

- Request

[Table 33] checkUID request

Attribute	Detail	
URI	/Diagnosis/checkUID	
Method	GET	
Parameter	uid	user id serial number
Header	Authorization	User Authentication

- Response

[Table 34] checkUID response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request, overlap)
	HTTP 403 (Forbidden)
	HTTP 404 (Not found)

Success Response Body	Message	Message "Success"
Failure Response Body	Message	Message "Error"

5.4.3. History System

5.4.3.1. GetHistory

- Request

[Table 35] GetHistory request

Attribute	Detail	
URI	/History/GetHistory	
Method	GET	
Parameter	uid	user id serial number
Header	Authorization	User Authentication

- Response

[Table 36] GetHistory response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request, overlap)

	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	date	current date
	time	current time
	image	image used to diagnosis
	Diagnosis	Diagnosis information
Failure Response Body	Message	Message "Error"

5.4.3.2. GetimagetoSV

- Request

[Table 37] GetimagetoSV request

Attribute	Detail	
URI	/History/GetimagetoSV	
Method	GET	
Parameter	path_image	path for image saved
Header	Authorization	User Authentication

- Response

[Table 38] GetimagetoSV response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	image	image used to diagnosis
Failure Response Body	Message	Message "Error"

5.4.3.3. GetRealData

- Request

[Table 39] GetRealData request

Attribute	Detail
URI	/History/GetRealData
Method	GET

Header	Authorization	User Authentication

- Response

[Table 40] GetRealData response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	real_diag	real diagnosis information
Failure Response Body	Message	Message "Error"

5.4.3.4. UpdateData

- Request

[Table 41] UpdateData request

Attribute	Detail
URI	/History/Updatedata

Method	GET	
Parameter	path_image	image used to diagnosis
	real_diag	real diagnosis information
Header	Authorization	User Authentication

- Response

[Table 42] UpdateData response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	Message	Message "Success"
Failure Response Body	Message	Message "Error"

5.4.3.5. learningAI

- Request

[Table 43] learningAI request

Attribute	Detail	
URI	/History/learningAI	
Method	GET	
Parameter	image	image used to diagnosis
	real_diag	real diagnosis information
Header	Authorization	User Authentication

- Response

[Table 44] learningAI response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request, overlap)
	HTTP 403 (Forbidden)
	HTTP 404 (Not found)

Success Response Body	Message	Message "Success"
Failure Response Body	Message	Message "Error"

5.4.4. Synchronize System

5.4.4.1. Getdata

- Request

[Table 45] Getdata request

Attribute	Detail	
URI	/Synchronize/Getdata	
Method	GET	
Parameter	uid	user id serial number
Header	Authorization	User Authentication

- Response

[Table 46] Getdata response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request, overlap)

	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	datainfo	user database information
Failure Response Body	Message	Message "Error"

5.4.4.2. Sync

- Request

[Table 47] Sync request

Attribute	Detail	
URI	/Synchronize/Sync	
Method	GET	
Parameter	datainfo	user database information
Header	Authorization	User Authentication

- Response

[Table 48] Sync response

Attribute	Detail

Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success Response Body	Message	Message "Success"
Failure Response Body	Message	Message "Error"

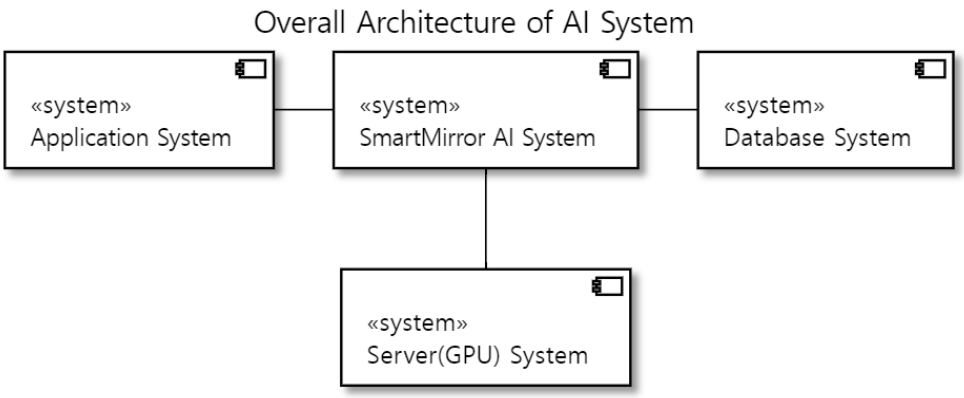
6. System Architecture - AI

6.1. Objectives

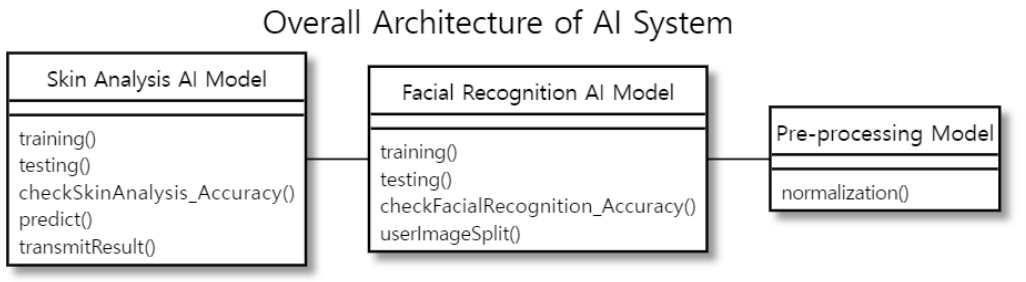
이 섹션은 모바일 어플리케이션과 백엔드와의 상호작용을 토대로 AI의 구조를 나타낸다.

6.2. Overall Architecture

AI 시스템의 전반적인 아키텍처는 아래 그림과 같다. 스마트 미러 시스템에 AI 서비스를 부착하기 전에 Open-Source Data를 이용하여 학습을 시켜 사용한다. 이후에 사용자의 모바일 어플리케이션에서 이미지 데이터를 수집하고, 프론트엔드 로부터 AI 학습 요청을 수신하여 서버의 GPU를 이용하여 AI 학습을 진행한다. 이 과정에서 데이터베이스에 저장된 스마트 미러 딥러닝 버전을 확인하고, 서버에서 학습된 결과값을 데이터베이스에 저장한 후, 프론트엔드로 결과값을 전달하여 피부 분석 결과를 UI로 나타낸다. 사용자가 병원에서 실제 진단을 받은 후, 진단 결과를 입력하면 이 데이터를 활용하여 AI 모델을 재 학습시켜 유저 맞춤형 AI 모델을 만들어 낸다.



[Figure 26] Context Model of Overall AI Architecture



[Figure 27] Class Diagram of Overall AI Architecture

6.3. Subcomponents

AI 학습 및 피부 분석 과정에서 사용되는 시스템들의 상세한 정보에 대하여 서술한다. 이 때, 학습 단계와 진단 단계 외에도 AI를 이용하는 과정에서 사용된 서버와 프로그램 버전 및 GPU 정보는 다음과 같다.

[Table 49] AI System에 사용된 제품 정보

Name	Description
GPU	NVIDIA GeForce RTX 3060
Python	3.7.11 version, conda 4.10.3 version
Library	PyTorch 1.9.1 version, Numpy 1.21.2 version
CUDA	CUDA Toolkit 11.1.0

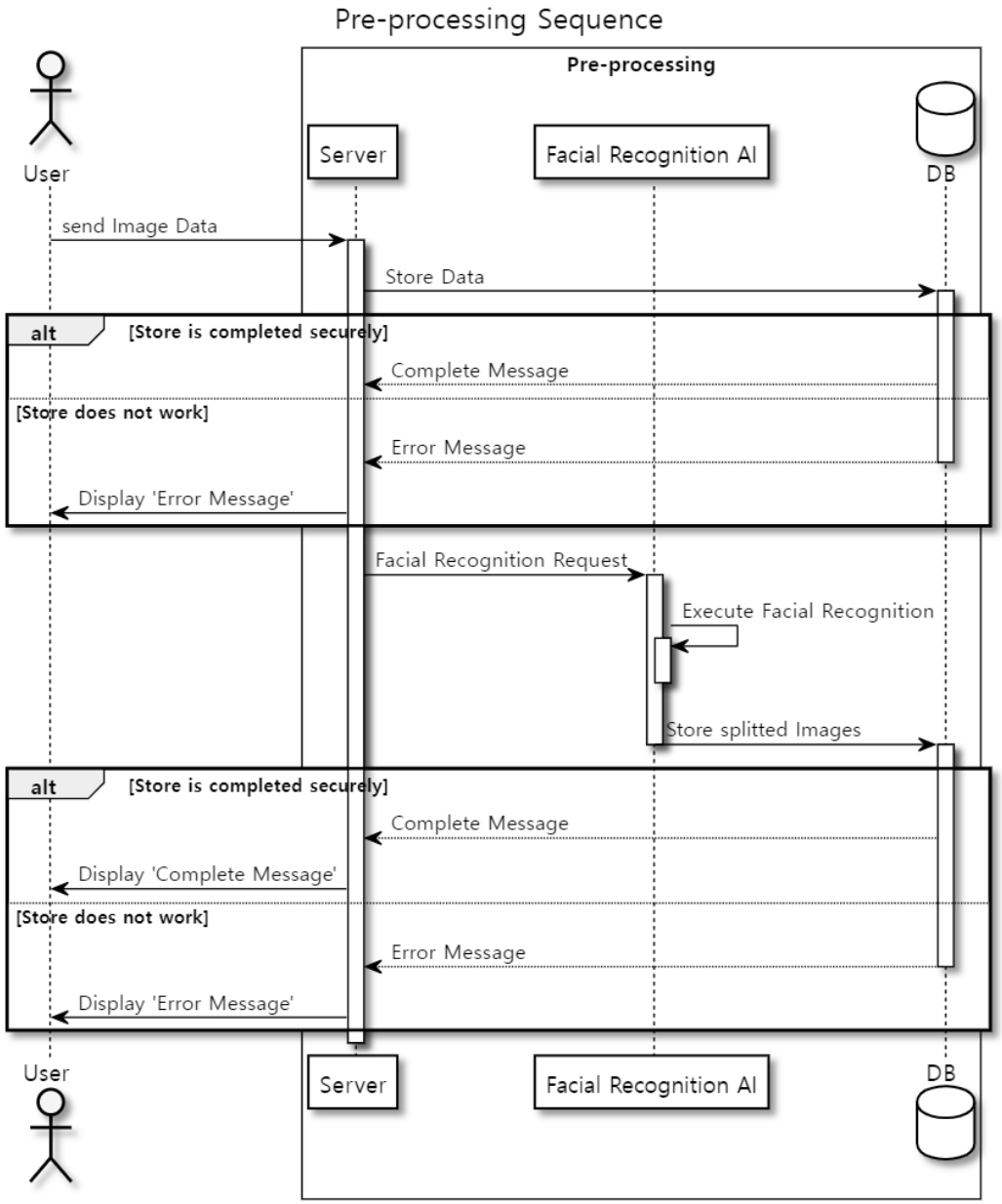
6.3.1. Pre-processing System

Pre-processing System은 데이터베이스로부터 전달받은 오픈 소스 데이터 혹은 사용자의 실제 이미지를 AI 모델의 input으로 넣기 위하여 적절하게 변환하는 과정이다. 이 때, input 이미지의 크기를 일정하게 맞추고 이미지에서 Facial

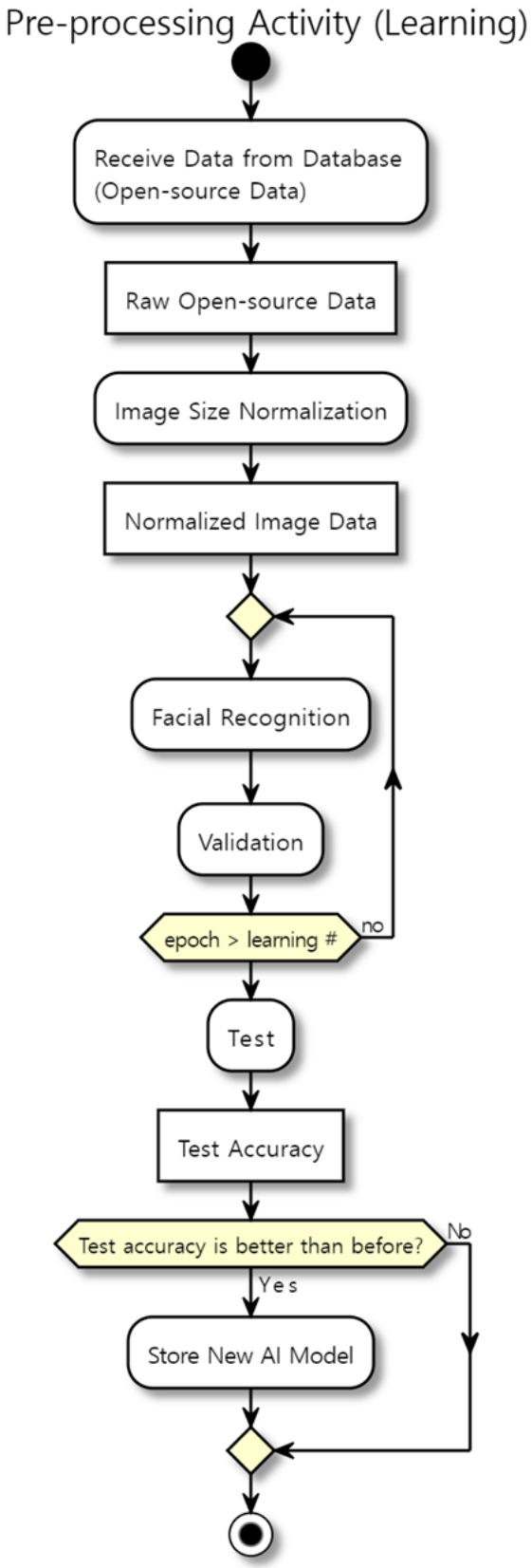
Recognition을 수행한다. 이를 토대로 얼굴만을 탐색하여 이미지를 자르고 이를 토대로 AI 모델에 전달한다. Pre-processing의 Facial Recognition 또한 AI Model을 이용하여 진행한다. 그렇기에 이 과정에서 오픈 소스 데이터를 이용하여 AI 모델을 학습시키며, 이에 대한 모델 정보는 아래 표에 정리하였다.

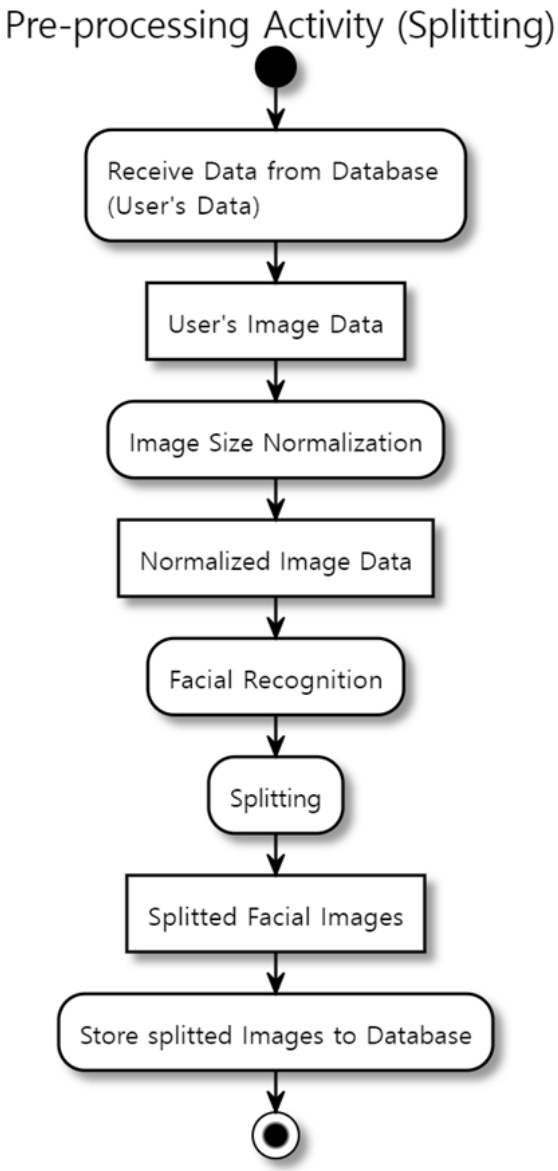
[Table 50] Facial Recognition AI Model에 사용된 제품 정보

Name	Description
GPU	NVIDIA GeForce RTX 3060
Python	3.7.11 version, conda 4.10.3 version
Library	PyTorch 1.9.1 version, Numpy 1.21.2 version
CUDA	CUDA Toolkit 11.1.0
Model	AInnoFace, SRN



[Figure 28] Pre-processing Sequence Diagram

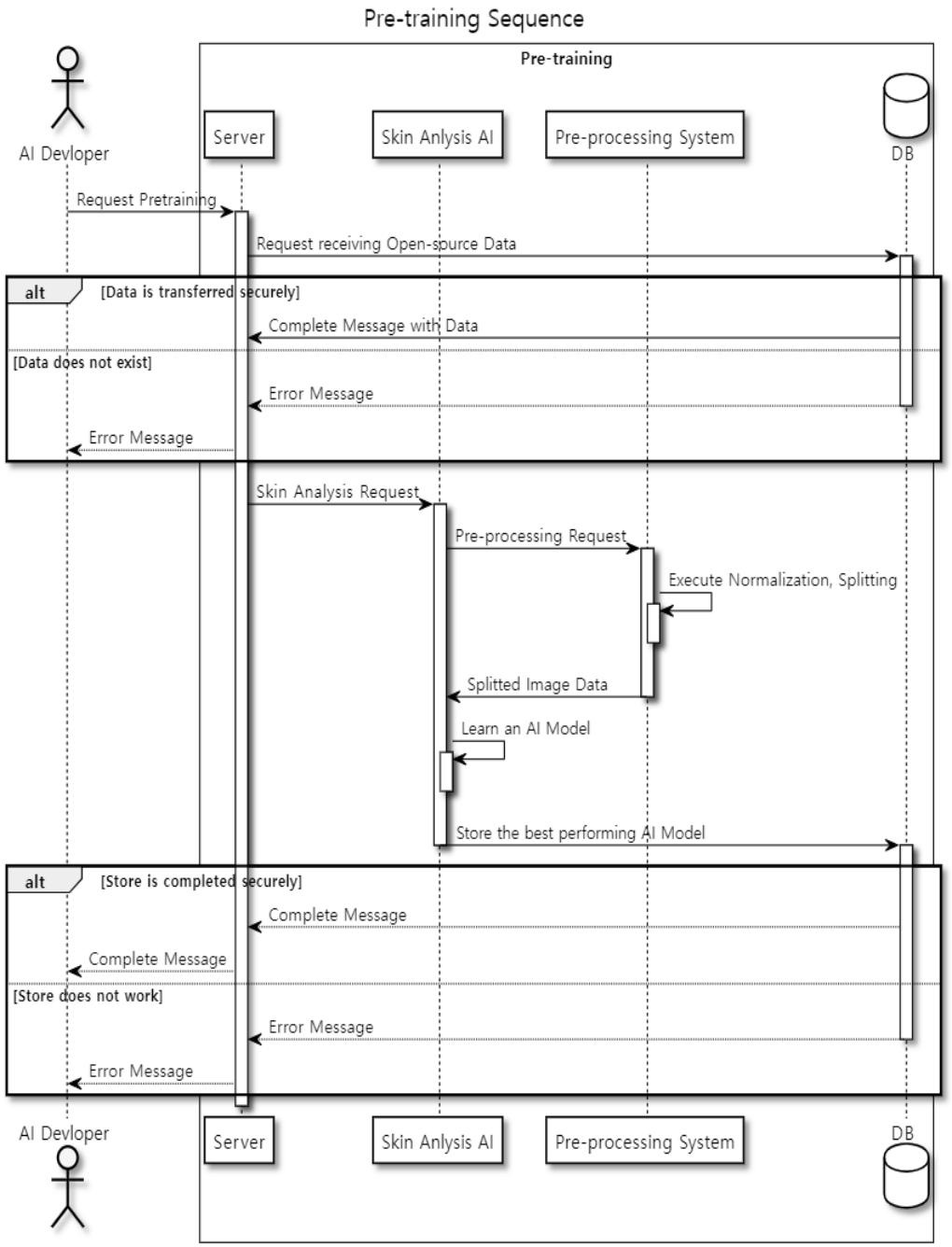

Pre-processing Activity (Learning)
Receive Data from Database (Open-source Data)
Raw Open-source Data
Image Size Normalization
Normalized Image Data
Facial Recognition
Validation
epoch > learning #
no
Test
Test Accuracy
Test accuracy is better than before?
No
Yes
Store New AI Model


Pre-processing Activity (Splitting)
Receive Data from Database (User's Data)
User's Image Data
Image Size Normalization
Normalized Image Data
Facial Recognition
Splitting
Splitted Facial Images
Store splitted Images to Database

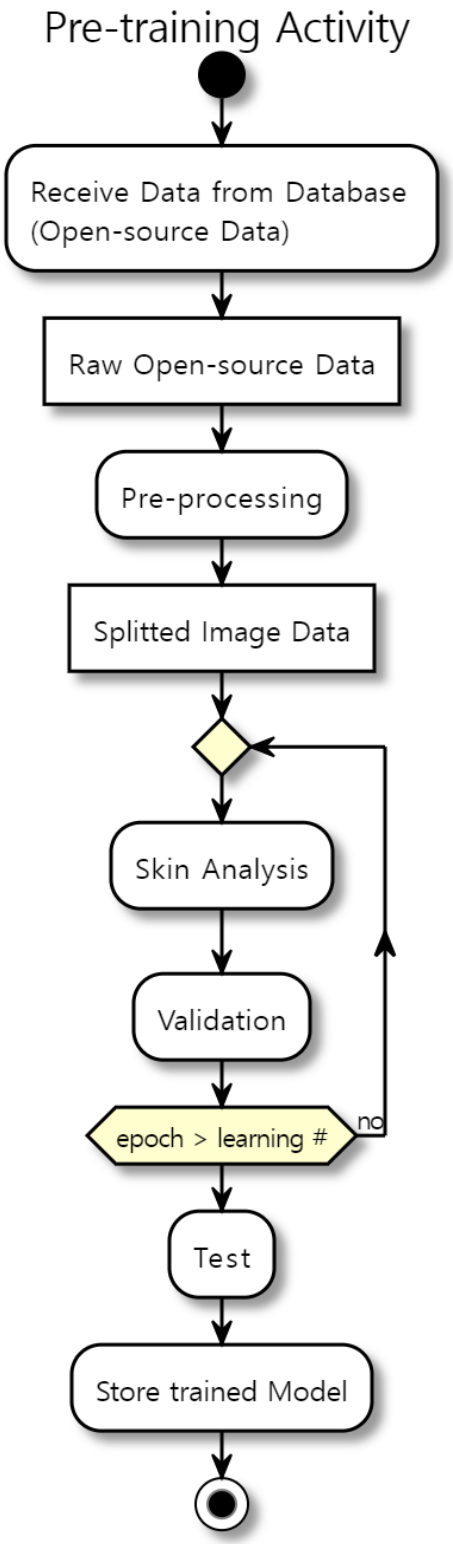
[Figure 29] Pre-processing Activity Diagram

6.3.2. Pre-training System

Pre-training System은 데이터베이스로부터 오픈 소스 데이터를 제공받아 학습을 하는 과정이다. 이 때, 데이터를 AI 모델에 적합한 형태로 변환하기 위해 pre-processing 시스템과 상호작용을 한다. 이 시스템에서는 모바일 어플리케이션으로부터 어떠한 input도 받지 않으며, 스마트 미러 시스템을 제공하기 전 AI 모델을 학습시킨다.



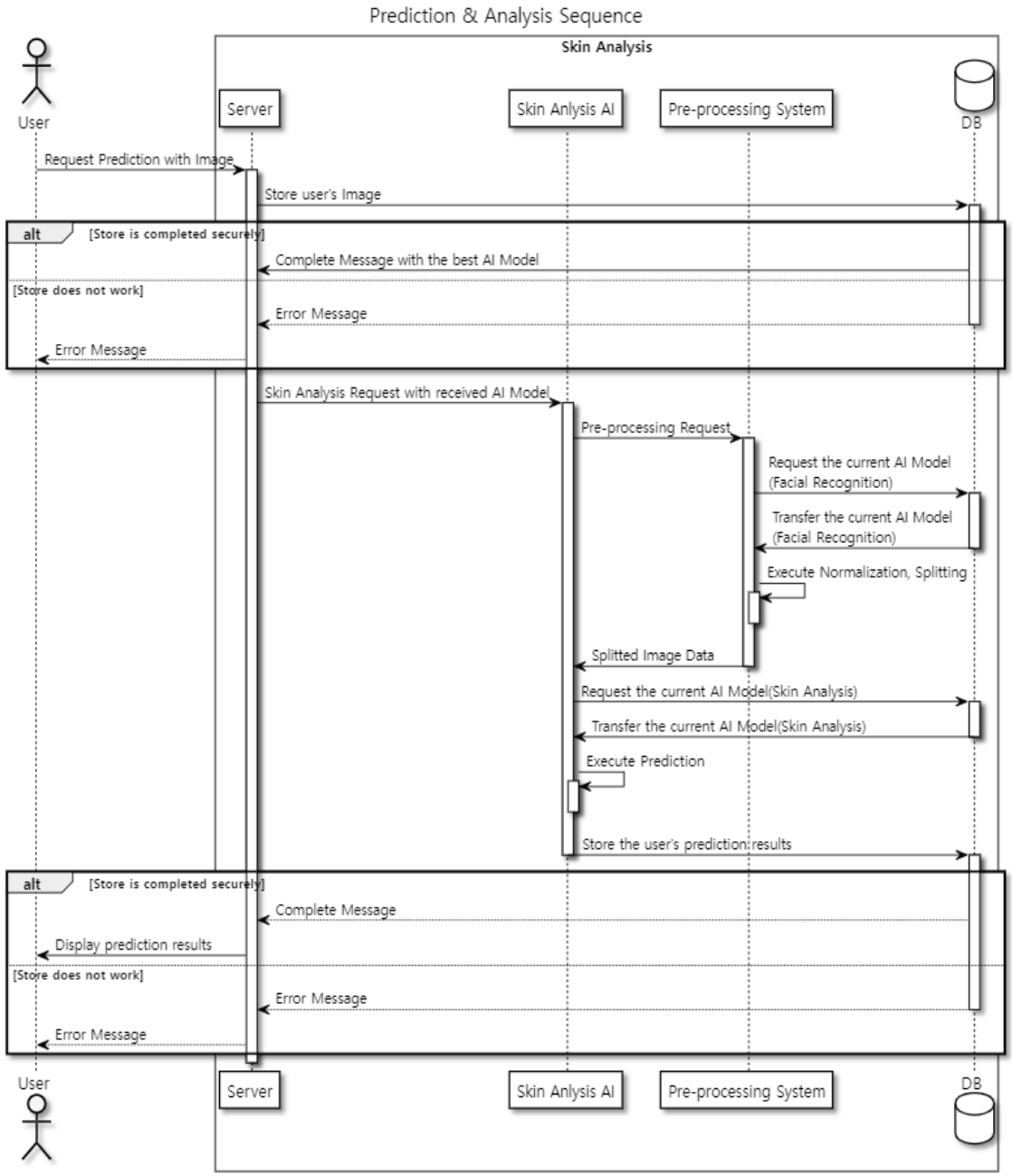
[Figure 30] Pre-training Sequence Diagram



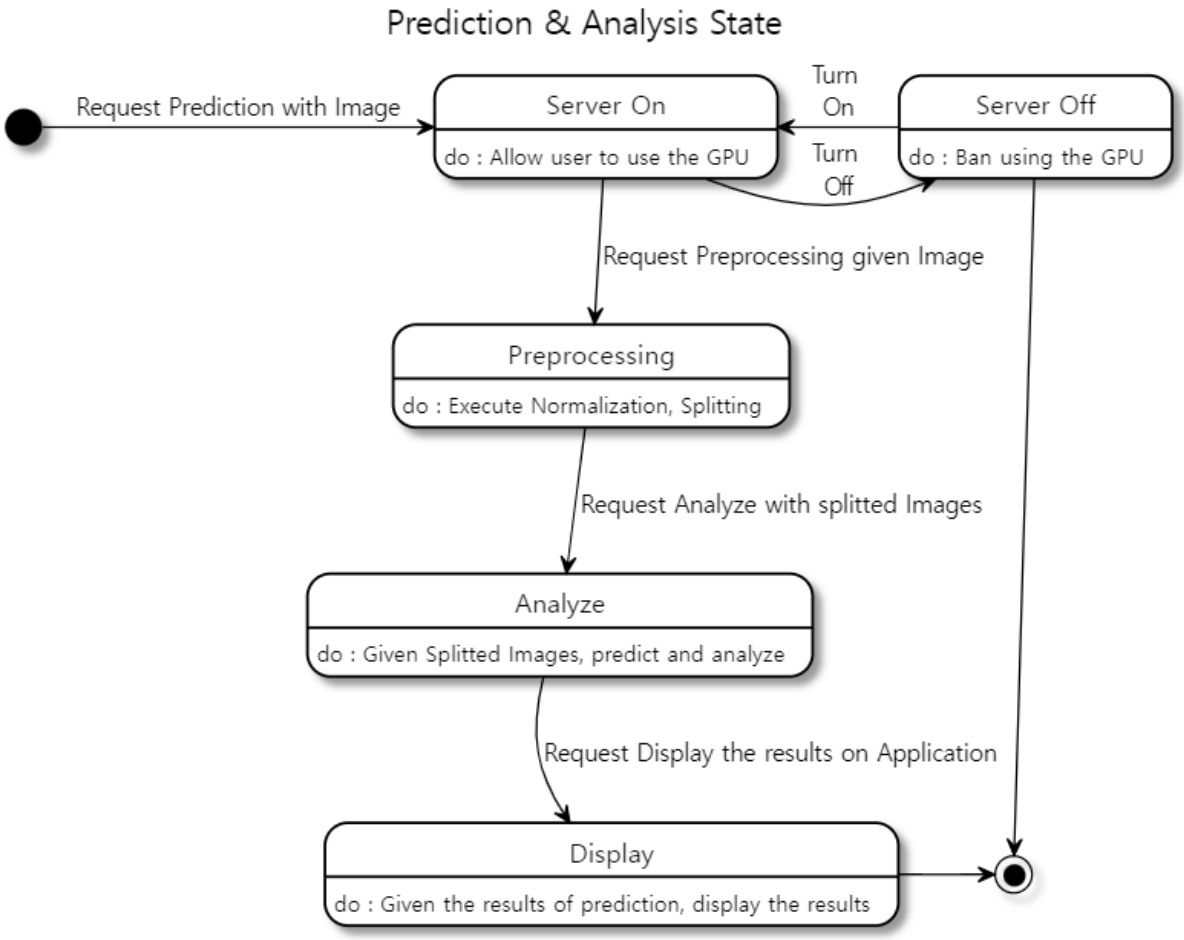
[Figure 31] Pre-training Activity Diagram

6.3.3. Prediction and Skin analysis using Users' data System

모바일 어플리케이션으로부터 전달받은 사용자의 이미지 데이터를 pre-processing을 통해 AI에 적합한 형태로 변환한 후, 피부 분석을 진행하는 과정이다. 모바일 어플리케이션과의 통신은 HTTP로 진행하였다.



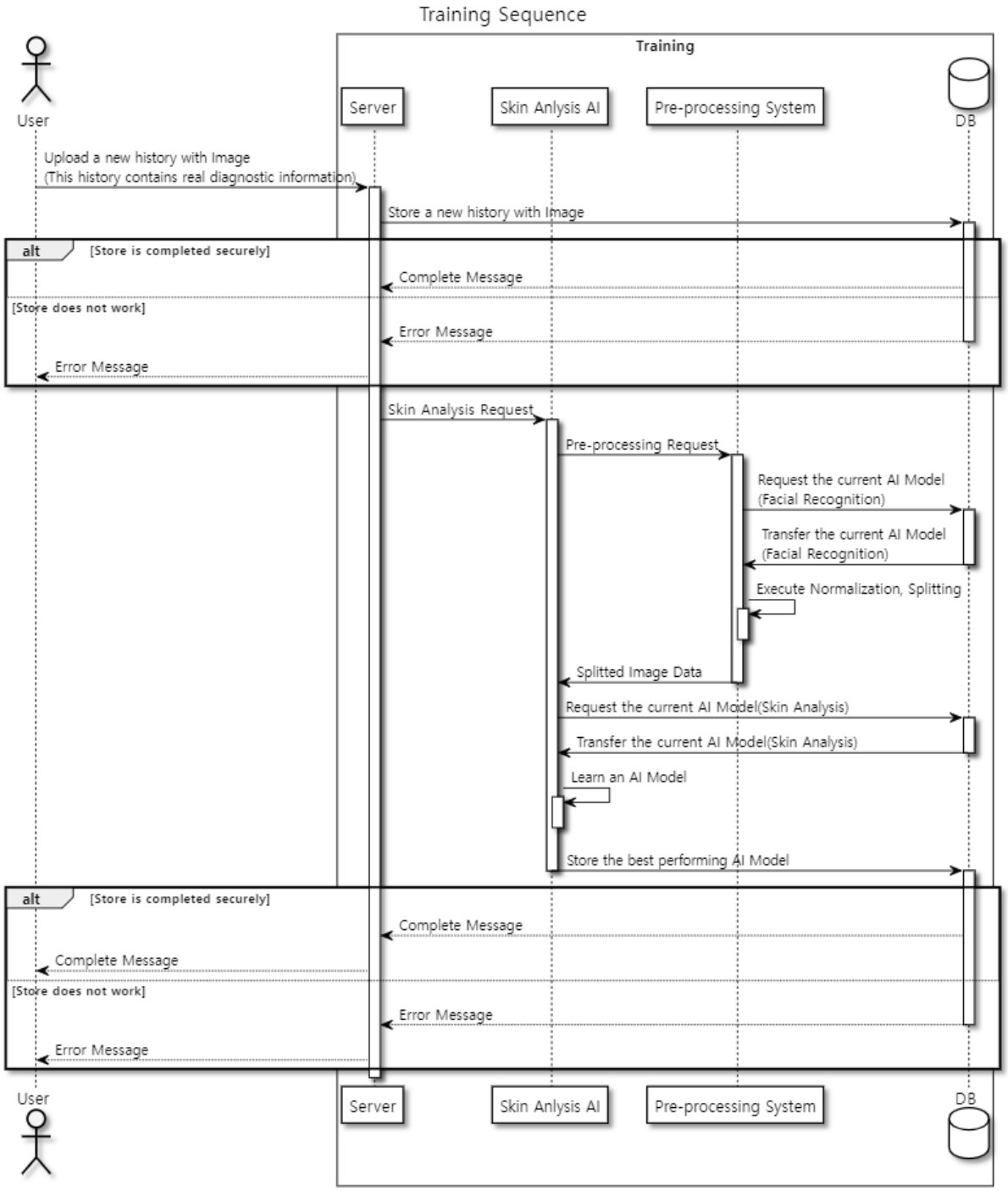
[Figure 32] Prediction & Analysis Sequence Diagram



[Figure 33] Prediction & Analysis State Diagram

6.3.4. Training System using Users' data System

스마트 미러 사용자가 병원에서 진단받은 결과를 모바일 어플리케이션에 입력하여 저장할 경우, 이 결과값과 이미지를 토대로 유저에게 맞춤형 AI 모델을 만들기 위해 추가 학습을 진행한다.



[Figure 34] Training Sequence Diagram

6.4. Protocols used in AI part

AI System은 모바일 어플리케이션과 서버와의 Request를 토대로 실행되는 것이며, 이 과정에서 JSON 포맷으로 통신한다. 그러므로 AI System이 전달받는 User의 Request는 전부 JSON 포맷이며, AI System이 User 측에 전달하는 결과 메시지들도 전부 JSON 포맷으로 구성된다. 하지만 AI System에서 결과를 전달하는 부분은 전부 CSV 포맷으로 이루어져 있으며 데이터베이스로부터 이미지 데이터를 받고 전달할 때는 이미지가 저장된 URL을 주고 받는 것을 원칙으로 한다. User가 전달하는 Request와 그에 따른 Response는 그림에 나타낸 메소드를 기준으로 정리하였다.

6.4.1. Skin Analysis AI Model

6.4.1.1 training()

- Request

[Table 51] training() request

Attribute	Detail	
URI	/skinAI/training	
Method	GET	
Parameter	Type	Training type (Pretraining, user-training)
	Data	Dataset used in training phase
Header	Authorization	Developer Authentication

- Response

[Table 52] training() response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Message	Message: "Training will be start"
Failure response body	Message	Message: "Access failure"

6.4.1.2 testing()

- Request

[Table 53] testing() request

Attribute	Detail
URI	/skinAI/testing
Method	GET

Parameter	Data	Dataset used in testing phase
Header	Authorization	Developer Authentication

- Response

[Table 54] testing() response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Result	The information of the results of testing (accuracy, elapsed ti me,...)
	Message	Message: "Access Success"
Failure response body	Message	Message: "Access failure"

6.4.1.3 checkSkinAnalysis_Accuracy()

- Request

[Table 55] checkSkinAnalysis_Accuracy() request

Attribute	Detail	
URI	/skinAI/checkAccuracy	
Method	GET	
Header	Authorization	Developer Authentication

- Response

[Table 56] checkSkinAnalysis_Accuracy() response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Accuracy	The best accuracy of AI Model

	Message	Message: "Access Success"
Failure response body	Message	Message: "Access failure"

6.4.1.4 predict()

- Request

[Table 57] predict() request

Attribute	Detail	
URI	/skinAI/userid/predict	
Method	GET	
Parameter	Data	User's Image used in prediction phase
Header	Authorization	User Authentication

- Response

[Table 58] predict() response

Attribute	Detail
Success Code	HTTP 200 OK

Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Result	The result of prediction
	ResultID	The UUID of User's prediction result
Success response body	Message	Message: "Access Success"
Failure response body	Message	Message: "Access failure"

6.4.1.4 transmitResult()

- Request

[Table 59] transmitResult() request

Attribute	Detail	
URI	/skinAI/userid/transmit	
Method	POST	
Parameter	Data	User's Image used in prediction phase

	ResultID	The UUID of User's prediction result
Header	Authorization	User Authentication

- Response

[Table 60] transmitResult() response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Message: "Access Success"	
Failure response body	Message	Message: "Access failure"

6.4.2. Facial Recognition AI Model

6.4.2.1 training()

- Request

[Table 61] training() request

Attribute	Detail	
URI	/faceAI/training	
Method	GET	
Parameter	Type	Training type (Pretraining, user-training)
	Data	Dataset used in training phase
Header	Authorization	Developer Authentication

- Response

[Table 62] training() response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Message	Message: "Training will be start"

Failure response body	Message	Message: "Access failure"

6.4.2.2 testing()

- Request

[Table 63] testing() request

Attribute	Detail	
URI	/faceAI/testing	
Method	GET	
Parameter	Data	Dataset used in testing phase
Header	Authorization	Developer Authentication

- Response

[Table 64] testing() response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request, overlap)
	HTTP 403 (Forbidden)

	HTTP 404 (Not found)	
Success response body	Result	The information of the results of testing (accuracy, elapsed time,...)
	Message	Message: "Access Success"
Failure response body	Message	Message: "Access failure"

6.4.2.3 checkFacialRecognition_Accuracy()

- Request

[Table 65] checkFacialRecognition_Accuracy() request

Attribute	Detail	
URI	/faceAI/checkAccuracy	
Method	GET	
Header	Authorization	Developer Authentication

- Response

[Table 66] checkFacialRecognition_Accuracy() response

Attribute	Detail

Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	Accuracy	The best accuracy of AI Model
	Message	Message: "Access Success"
Failure response body	Message	Message: "Access failure"

6.4.2.4 userImageSplit()

- Request

[Table 67] userImageSplit() request

Attribute	Detail	
URI	/faceAI/userid/userImageSplit	
Method	GET	
Parameter	Data	User's Image used in prediction phase

Header	Authorization	User Authentication

- Response

[Table 68] predict() response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	ResultImg	The splitted Images
	Message	Message: "Access Success"
Failure response body	Message	Message: "Access failure"

6.4.3. Pre-processing Model

6.4.3.1 normalization()

- Request

[Table 69] normalization() request

Attribute	Detail	
URI	/preprocessing	
Method	GET	
Parameter	Data	Image data
Header	Authorization	Developer Authentication

- Response

[Table 70] normalization() response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request, overlap)	
	HTTP 403 (Forbidden)	
	HTTP 404 (Not found)	
Success response body	ResultImg	The normalized Images

	Message	Message: "Access Success"
Failure response body	Message	Message: "Access failure"

7. Testing Plan

7.1. Objectives

이 장에서는 개발, 배포, 사용자에 따른 시스템의 테스트 계획에 대해 기술한다. 이는 시스템에 존재하는 에러와 결함을 찾아내고, 소비자들에게 배포함에 있어 안정성과 무결점을 보장할 수 있다.

7.2. Testing Policy

7.2.1. Development Test

Development Test는 소프트웨어 개발 과정에서 생길 수 있는 여러 위험 요소들을 줄이고, 시간과 비용을 절약하기위해 진행되는 테스트이다. Development Testing 단계에서는 컴포넌트 간의 충돌이 발생할 수 있으며, 충분히 테스트되지 않았기 때문에 소프트웨어가 불안정할 수 있다. 따라서 코드 분석, 데이터 플로우 분석, 동료 평가, unit testing이 진행된다. 이 과정에서 성능, 신뢰도, 보안을 중점으로 확인하며 개선해나갈 것이다.

7.2.1.1. Performance

스마트 미러 시스템에서 핵심적인 기능은 사용자의 사진을 촬영하고 피부 진단 결과를 보여주는 것이고, 서버와의 통신 및 AI 처리 서버의 성능적인 결함으로 인해 데이터의 처리 및 분석이 너무 지연되면 안 된다.

통신상태에 문제가 없다는 가정하에, 사용자의 사진을 서버에 업로드 하는 것은 최소 10Mbps의 속도를 지원해야 하며, 10초 내에 AI 처리 서버에서 분석하여 그 결과를 보내야한다. 또한 프로그램이 실행되면 5초 내에 메인 화면에 진입해야 하고, 로그인 과정은 2초 내에 완료되어야 하며, 스마트 미러와 모바일 기기의 연동이 1초 내에 완료되어야 한다.

7.2.1.2. Reliability

시스템이 실패 없이 안전하게 동작하기 위해, 시스템의 서브 컴포넌트들이

처음에 잘 동작하고 서로 잘 연결되어야한다. 따라서 13조는 unit developing stage 에서부터 테스트를 진행할 것이고, 반복하여 확인하며 시스템에 잘 결합되는지 확인할 것이다.

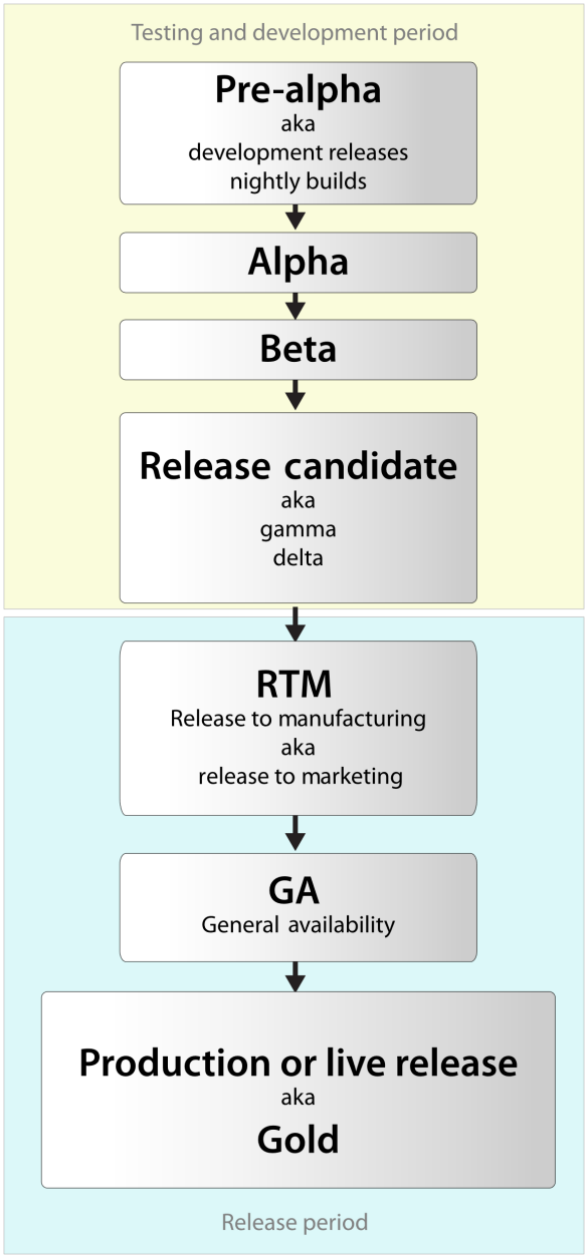
7.2.1.3. Security

사용자의 개인 정보를 보호하는 것은 개발자 차원에서 당연히 안전장치를 마련해야한다. 사용자들의 계정 정보 및 개인 정보들 역시 중요하지만, 스마트 미러 시스템에서 수집하는 사용자의 사진 데이터와 의료 데이터에 해당되는 진단 결과 데이터 역시 치밀하게 관리되어야한다. 시스템의 보안을 위해 매 버전 소프트웨어의 개발이 완료된 후, 코드에 대한 리뷰를 진행해 보안상 문제가 없는지 확인할 것이며, Ostrolab, Appvigil, TestComplete 등의 어플리케이션 자동 분석 서비스를 이용해 개발자가 놓친 보안 이슈가 없는지 한번 더 꼼꼼히 확인할 것이다.

7.2.2. Release Test

소프트웨어 개발 프로젝트는 설계와 구현에서 그치는 것이 아닌, 소프트웨어의 배포까지 포괄한다. 즉, 최적화가 잘된 완성된 어플리케이션일지라도, 그 배포 방법이 잘못되었다면 어플리케이션의 구동 과정에서 문제가 발생할 수 있다. 따라서 개발자는 어플리케이션의 정식 배포 전에 Release Test를 통해 구현된 어플리케이션이 사용자에게 계획대로 사용될 수 있는지 테스트를 해야한다.

소프트웨어 배포 생명 주기(Software Release Life Cycle)에 따르면, 테스트는 소프트웨어의 '알파' 버전에서 시작하며, 이는 기본적인 구현만 완료된 버전이다. 13조는 알파버전부터 개발을 시작할 것이며, 다양한 테스트를 거쳐 베타 버전을 배포할 계획이다.



[Figure 35] Software Release Life Cycle

7.2.3. User Test

위의 테스트와 더불어, 실제 기능들이 계획한대로 사용자들이 이용하는지 확인하고, 여러 사용자가 이용하는 환경을 확인하는 User Test를 진행할 것이다. 이를 위해 약 30명의 테스트 유저들을 선정해 사용자들의 집에서 2주간 피부 진단 기능을 이용할 수 있도록 테스트 환경을 구성할 것이다. 2주간의

테스트가 지난 후, 각각의 사용자들로부터 의견을 전달받아 스마트 미러 시스템의 개선에 이용할 것이다.

7.2.4. Test Case

테스트 케이스들은 개발자들이 목표로 하는 기능, 성능, 그리고 예상치 못한 접근의 세가지 과제들을 확인할 수 있도록 작성되었다. 각각의 과제는 기획하지 않은 방향으로의 사용에 대한 테스트, 다중 사용자의 접근으로 AI 서버의 과부하를 유도하는 테스트, 네트워크 환경이 불안정한 환경에서의 테스트, 스마트 미러가 설치된 환경에 대한 테스트 등 다양한 테스트 케이스를 생성해 소프트웨어를 테스트하고, 이를 기반으로 시스템 평가서를 작성하여 소프트웨어 최적화에 활용할 것이다.

8. Development Plan

8.1. Objectives

이 장에서는 스마트 미러 시스템 개발에 사용된 여러 외부 프로그램들에 대해 기술한다. 프론트엔드, 백엔드, AI 파트로 나누어 기술하며, 추가적으로 제약조건과 기본 가정 및 의존성에 대해 기술한다.

8.2. Frontend Environment

8.2.1. Adobe Illustrator

Adobe Illustrator는 벡터 이미지를 제작하는 툴로, 스마트 미러 UI를 구성하는 백그라운드 이미지 및 아이콘을 제작하는데 사용한다.

8.2.2. Adobe Xd

Adobe Xd는 웹 디자인 및 모바일 어플리케이션의 UI 및 UX디자인을 위한 프로토타이핑 프로그램이다. 이 프로그램의 특징으로는 디자인한 어플리케이션을 바로 시각화할 수 있고, 조원들과 공유하며 작업하기에 용이하다.

8.2.3. Xcode

Objective-C 혹은 Swift 언어로 iOS 어플리케이션을 개발할 수 있는 통합 개발환경(IDE; Integrated Development Environment)이다. 스마트 미러 시스템을 보조하는 모바일 어플리케이션의 아이폰용 버전을 제작하기 위해 사용한다.

8.2.4. Android Studio

Java 혹은 Kotlin 언어로 안드로이드 어플리케이션을 개발할 수 있는 통합 개발환경이다. 기본적인 안드로이드 어플리케이션의 틀을 제공하며, 안드로이드 에뮬레이터를 활용해 기기 없이도 어플리케이션의 동작을 바로 확인할 수 있으며, 안드로이드 기기를 연결해 별도의 인증 없이 바로 설치 가능하다. 스마트 미러 시스템을 보조하는 모바일 어플리케이션의 안드로이드 버전을 제작하기 위해 사용한다.

8.3. Backend Environment

8.3.1. Node.js

Node.js는 JavaScript 런타임으로, 다양한 JavaScript 어플리케이션을 실행할 수 있도록 만든 프로그램이다. 따라서 Node.js를 이용해 JavaScript를 서버단에서도 사용 가능하도록 하기 위해 Node.js를 이용한다. JavaScript는 Non-blocking I/O와 단일 스레드 이벤트 루프를 통한 높은 처리 성능을 가지고 있는 것이 특징이다. 또한 내장 HTTP 서버 라이브러리를 포함하고 있어 아파치 등의 별도 소프트웨어 없이 동작하는 것이 가능하다.

8.3.2. mongoDB

mongoDB는 비관계형 데이터베이스(NoSQL)의 한 종류로, C++로 작성된 오픈 소스 문서지향 데이터베이스이다. 문서(Document)란, Key-Value 쌍으로 이루어진 데이터 구조로, JSON 혹은 파이썬의 dictionary와 같은 구조이다. 따라서 관계형 데이터베이스와 다르게 다양한 형태의 데이터를 저장할 수 있고, 데이터 모델의 유연한 변화가 가능하다. 스마트 미러 시스템의 유저 정보와, 사진 데이터, 진단 결과 정보를 저장하기에 용이하다.

8.3.3. Jenkins

Jenkins는 소프트웨어 개발의 지속적인 통합과 지속적인 전달 환경을 구축하기 위한 툴이다. Jenkins를 이용하면 구현된 요소들에 대한 통합, 빌드, 테스트가 빠르고 편리하다.

8.4. AI Environment

8.4.1. PyTorch

PyTorch는 딥러닝 구현을 위한 파이썬 기반의 오픈 소스 머신러닝 라이브러리이다. 간결하고 빠른 구현이 특징이고, 파이썬의 라이브러리와 호환성이 높으며, 학습 및 추론 속도가 빠르고 다루기 쉽다는 장점이 있다. 스마트 미러 시스템의 피부 분석 학습을 위해 오픈 소스 데이터를 활용하고, PyTorch를 이용해 딥러닝 서버를 구현한다.

8.4. Constraints

피부 건강 관리를 위한 스마트 미러 시스템은 이 문서와 소프트웨어 요구사항 명세서의 내용에 따라 구현될 예정이다. 이 외의 세부적인 구현 방향성은 아래의 제약사항을 따르며, 이 외에 고객의 추가적인 요구사항에 따라 진행된다.

- 카메라, 내부 데이터 접근 등 스마트 미러 및 사용자 기기에 반드시 접근 권한을 요구하며, 권한의 변경은 기기 내에서 가능해야 한다.
- 데이터의 균일함을 위해 스마트 미러의 카메라 및 디스플레이 옵션을 조절할 수 있으며, 이에 대한 사용자의 동의를 구해야한다.
- 사용자의 데이터는 의료 데이터의 일종으로 구분되어 보안에 더욱 신경써야 한다.
- 피부 진단을 위한 사용자의 사진 촬영은 반드시 스마트 미러를 통해서만 가능하며, 스마트 미러의 설정을 유동적으로 변경해 데이터의 균일함을 보장해야한다.

- 스마트 미러의 피부 진단 결과는 단순 참고용일 뿐이며, 법적 효력을 가질 수 없다.
- 스마트 미러와 모바일 어플리케이션의 UI는 쉽게 이용할 수 있도록 직관적이어야하며, UX의 개선을 위해 물체의 배치, 제어 방법과 전체적인 과정도 고려해야한다.
- 오픈 소스 데이터와 소프트웨어를 최대한 활용한다.
- 시스템의 비용과 유지보수 비용을 고려한다.
- 주기적으로 소스코드를 최적화하여 자원의 낭비 및 성능 저하를 예방한다.
- 최소 Android 7.0 (API 24) 버전 조건에 맞추어 개발하며, Android 12 버전에서 테스트를 진행한다.
- 최소 iOS 14.0 버전 조건에 맞추어 개발하며, iOS 15.0 버전에서 테스트를 진행한다.

8.5. Assumptions and Dependencies

이 문서에서 구현할 어플리케이션은 최소 Android 7.0 (API 24)를 사용하는 스마트 미러 혹은 모바일 기기, 그리고 iOS 14.0 이상의 버전을 사용하는 모바일 기기에서 사용된다는 가정 하에 작성되었다. 따라서, 명시된 운영체제 이외의 다른 운영체제 에서는 기능이 원활하게 작동되지 않을 수 있다.

또한 피부 건강 관리를 위한 스마트 미러 시스템은 오픈 소스 데이터와 툴에 의존성을 가지며, 카메라, 푸시 알림, 내부 데이터 접근 등 사용자 기기에 대한 접근 권한들을 요구할 수 있으며, 허용되지 않은 권한에 대해서는 어플리케이션의 정상적인 사용이 제한될 수 있다.

9. Supporting Information

9.1. Software Design Specification

이 소프트웨어 디자인 명세서는 IEEE Recommendation (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016).의 서식을 따라 작성되었다.

9.2. Document History

[Table 71] Document History

Version	Version	Description	Writer
2022/05/05	0.1	Preface, Introduction	백송현
2022/05/06	1.0	Overall Architecture	백송현
2022/05/06	1.1	System Architecture – Frontend	박민서
2022/05/07	1.2	System Architecture - Backend	정민석
2022/05/07	1.3	System Architecture – AI	이재혁
2022/05/08	1.4	Testing Plan	백송현
2022/05/09	1.5	Development Plan	백송현
2022/05/10	1.6	System Architecture – Frontend, Backend	박민서, 정민석
2021/05/11	1.7	System Architecture – Backend, AI	정민석, 이재혁
2022/05/12	1.11	Overall Architecture	백송현
2022/05/13	1.12	Introduction	백송현
2022/05/15	1.13	Revision of style	설채은