# Design Specification

TEAM 3

2015313527 이창원
2015313570 차미경
2014310631 신현호
2016311270 양희산
2016313354 위응주

# 1. Preface

## 1.1. Objective

Preface에서는 본 문서의 독자를 정의하고, 구조에 대해 소개한다. 구조를 설명할 때는 각 목차의 목적에 대해서 서술한다. 또한 이후 버전이 변경될 때 관리 정책, 버전 변경 기록, 그리고 문서의 변경사항들과 이들에 대한 근거들을 서술한다.

## 1.2. Readership

본 문서의 독자는 다음과 같이 정의된다. '감정분석 다이어리'를 개발하는 software engineer, 시스템 구조를 설계하는 architecture, 시스템을 유지 및 보수하는 software engineer, 그리고 개발에 참여하는 모든 stakeholder를 독자로 정의한다. 즉, 본 문서의 독자는 본 문서에서 소개하는 시스템의 개발 및 유지 보수에 관련된 모든 구성원이다.

## 1.3. Document Structure

A. Preface

Preface에서는 본 문서의 독자를 정의하고, 구조에 대해 소개한다. 구조를 설명할 때는 각 목차의 목적에 대해서 서술한다. 또한 이후 버전이 변경될 때 관리 정책, 버전 변경 기록, 그리고 문서의 변경사항들과 이들에 대한 근거들을 서술한다.

B. Introduction

Introduction에서는 본 문서에서 사용되는 모든 종류의 diagram과 tool에 대해 설명한다.

C.System Architecture

System Architecture에서는 본 프로젝트에서 개발하려는 시스템에 대해 전반적으로 서술한다. 해당 시스템을 Block Diagram을 이용하여 나타내어 전체적인 구조에 대해 설명하고 각 시스템들간의 관계에 대해 설명한다. 각 시스템에 대한 자세한 설명은 이후에 이뤄진다.

D.Diary System

하루의 일기를 기록하고 지난 일기를 볼 수 있는 diary system에 대해 설명한다. class diagram, sequence diagram 그리고 state diagram을 통해 시스템의 구조를 서술한다.

E.Emotion Extraction System

지난 일주일의 일기글을 분석해 사용자의 감정을 추출하고 관리해주는 emotion extraction system에 대해 설명한다. class diagram, sequence diagram 그리고 state diagram을 통해 시스템의 구조를 서술한다.

F.User Management System

회원가입, 로그인 기능을 통해 사용자가 본 시스템을 이용하는 도중에 발생한 데이터를 처리하는 user management system에 대해 설명한다. class diagram, sequence diagram 그리고 state diagram을 통해 시스템의 구조를 서술한다.

G.Protocol Design

Protocol Design을 통해 어떤 방식으로 sub-systems이 서로 메세지를 주고 받는지에 대해 서술한다. 기본적으로 JSON을 이용해, 통신하는 내용에 대해 설명한다.

H.Database Design

Database Design에서는 requirement specification에서 서술한 database requirement를 바탕으로 ER diagram을 작성하고 이를 통해 Relational model를 작성하였다. 이 Relation model을 통해 SQL DDL을 작성하였다.

I.Testing Plan

시스템이 요구사항을 만족하는지 확인하고 오류없이 잘 실행되는지 확인하기 위해 Testing Plan에서 test case에 대해 정의한다.

J.Development Environment

Development Environment에서는 개발하는 환경에 대해 서술한다. 사용하는 프로그래밍 언어와 IDE, package에 대해 서술한다.

K.Develop Plan

Develop Plan에서는 Gantt chart을 사용해 개발 계획에 대해 서술한다.

L.Index

Index에서는 문서에서 사용된 테이블, 다이어그램 그리고 그림의 index를 보여준다.

M.Reference

Reference는 본 문서를 작성하기 위해 참고되어진 웹사이트 등에 대한 정보를 보여준다.

### 1.4 Version of the Document

A.Version Format

version의 번호는 기본적으로 major.minor로 표현된다. 문서의 초기 버전은 1.0이다.

B.Version Management Policy

본 문서를 수정할 때마다 version을 업데이트해준다. 하지만, 변경시간이 1시간 이내일 경우는 버전 번호를 바꾸지 않고 하나의 업데이트로 가정한다. 기존에 작성된 파트의 내용을 변경시에는 minor number을 변경하며, 새로운 내용을 추가하거나 기존에 작성된 내용과 크게 달라진 부분이 있다면 major number을 변경한다.

C.Version Update History

| Version | Modified Date | Explanation |
|---|---|---|
| **1.0** | 2019.05.18 | 초기버전 |

Table 1. Version Update History Table

# 2. Introduction

## 2.1. Objectives

Introduction에서는 본 문서에서 사용되는 모든 종류의 diagram과 tool에 대해 설명한다.

## 2.2. Applied Diagram

A.UML


Figure 1. UML Logo

본 프로젝트에서 System modeling을 위해 사용할 언어는 UML(Unified Modeling Language)이다. UML은 소프트웨어 공학에서 이용되는 표준화된 범용 modeling language이며, 객체지향 소프트웨어 집약 시스템을 개발할 때 산출물을 명세화, 시각화 문서화 할 때 사용된다. 행위자, business process, 부품, 행위, 프로그래밍 언어 구문, database schema, 재사용할 수 있는 소프트웨어 구성요소를 포함하여 시스템을 시각화하는 표준안을 제공한다. 따라서 개발을 본격적으로 시작하기 전 시스템을 시각화하여 개발자뿐만 아니라 시스템과 관련된 stakeholder로 하여금 시스템에 대한 이해를 높여, stakeholder과 원만한 대화가 이뤄질 수 있다. UML은 비단 소프트웨어 개발 공정만이 아닌 다른 모든 공정에서 사용될 수 있다.

B.UML Diagrams

UML에는 총 14가지의 diagram 종류가 있다. 이 diagram을 통해 시스템을 여러가지 관점으로 표현할 수 있다. 본 프로젝트에서 이 diagram중 4개를 사용한다. 다음은 사용하는 diagram들에 대한 설명이다.

B.1. Class Diagram

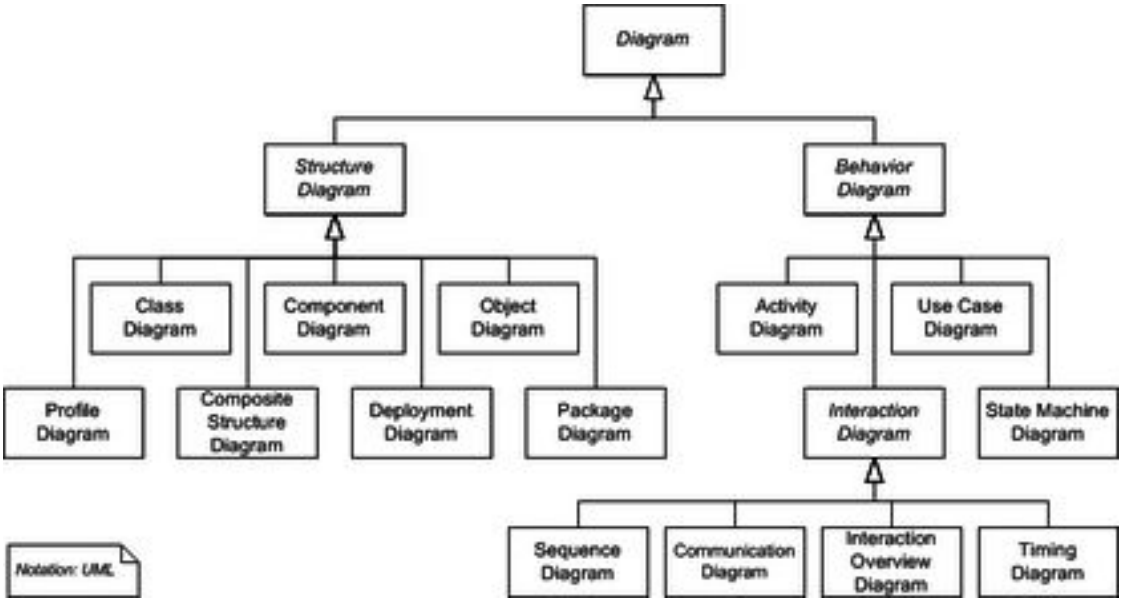


Figure 2. Example of Class Diagram

Class Diagram은 시스템의 classes, attributes, operation(또는 method) 및 객체간의 관계를 보여줌으로써 시스템의 구조를 보여주는 정적 구조 다이어그램이다. 하나의 class는 아래 그림과 같이 하나의 상자로 표현되며, 상자는 3부분으로 나눠진다.

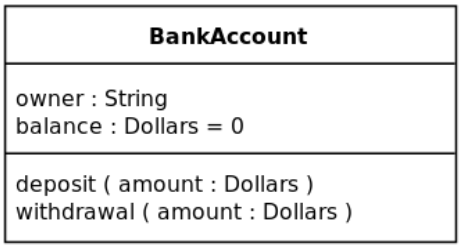


Figure3. A class represented with box containing three compartments

맨 위의 칸은 class의 이름이며, 두번째 칸은 class의 attributes이다. 마지막 칸은 class의 operation(또는 method)이다.

B.2. Sequence Diagram

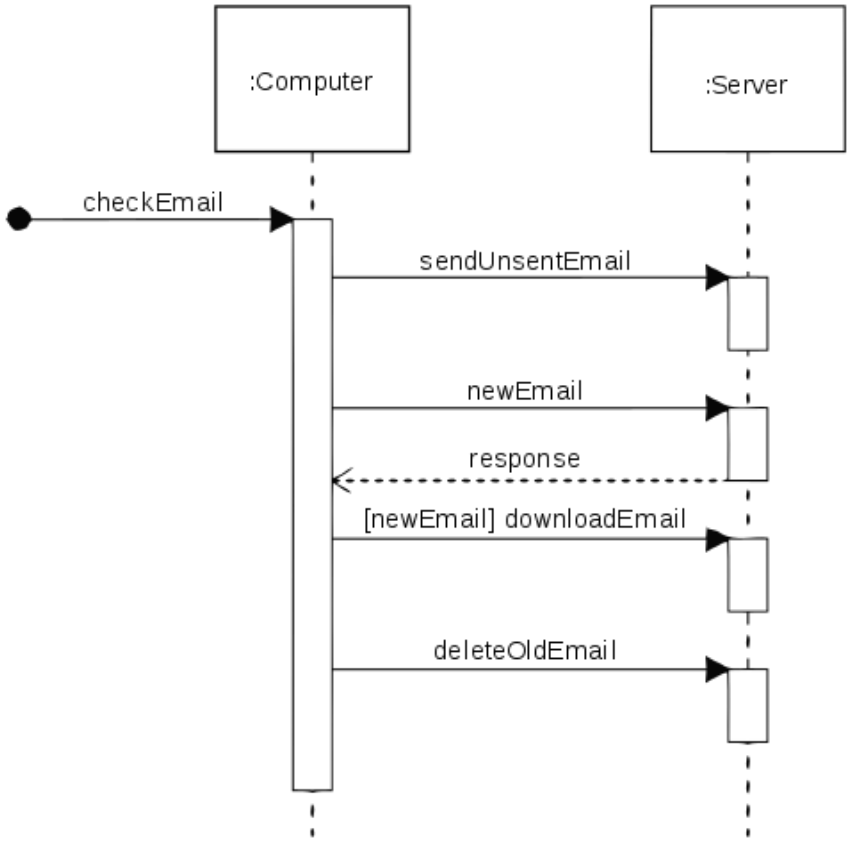


Figure 4.Example of Sequence Diagram

Sequence diagram은 시간의 순서대로 정렬된 객체들의 상호 작용을 보여준다. 시나리오에서 분석된 objects 및 classes와 시나리오에서 기술된 기능을 수행하는데에 이들이 어떻게 메세지를 교환받는지를 보여준다. sequence diagram은 event diagram 또는 event scenario라고도 불린다. 수직선은 서로다른 objects 또는 process이며 수평 화살표는 이들 사이 교환되는 메세지를 순서대로 보여준다.

B.3. State Diagram

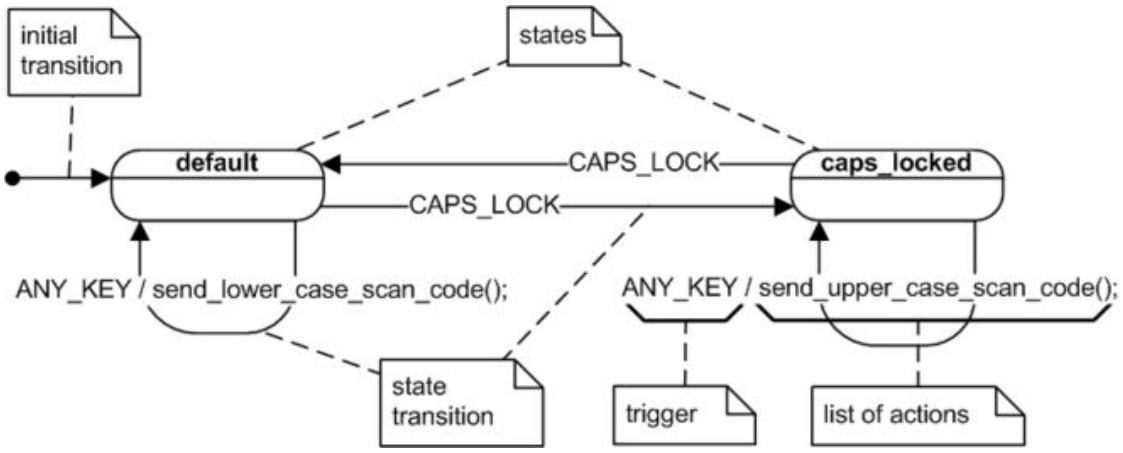


Figure 5. Example of State Diagram

State diagram은 한 객체의 상태의 변화를 표현한다. 실제 시스템을 사용시 발생하는 이벤트와 객체간 주고 받는 메세지는 무수히 많을 것이다. 따라서 모든 상태 변화에 따른 상황을 표현하기 보다는 핵심적인 객체와 상황에 주목한다.

B.4. Use Case Diagram

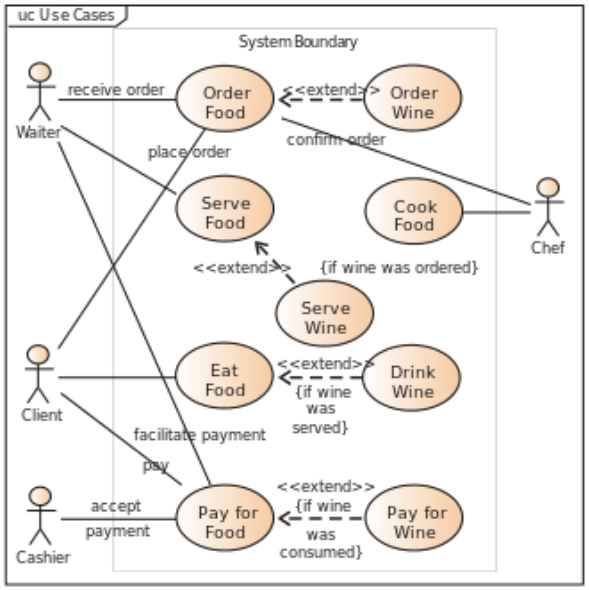


Figure 6. Example of Use Case Diagram

Use Case diagram은 사용자들간의 상호작용을 표현하고 사용자와 관련된 다양한 use case에 대해 보여준다. use case는 타원으로 나타낸다. Use case diagram은 high-level의 시스템 관점이다. 따라서 stakeholder들이 시스템을 이해하는데 유용한 수단이 될 수 있다.

### 2.3. Applied Tools

A.Draw.io


Figure 7. Draw.io Logo

Draw.io는 flowcharts, process diagrams, org charts, UML, ER 그리고 network diagrams를 만드는데 쓰이는 online diagram software이다. 비용을

지불하지 않고도 간편하게 UML, ER을 그릴 수 있고 저장도 가능하기 때문에 본 문서에 작성되는 Diagram은 대부분 draw.io를 사용하여 작성되었다.

B.Android Studio

Figure 8 . Android Studio Logo

Android Studio는 안드로이드 앱을 제작하기 위한 공식 IDE(Integrated Development Environment)이다. 다양한 운영체제에서 사용이 가능하다. 본 시스템은 안드로이드 앱이기 때문에 Android Studio를 개발 tool로 지정했다.

C.Firebase


Figure 9. Firebase Logo

Firebase는 2011년 Firebase, Inc가 개발하고 2014년 구글에 인수된 모바일 및 웹 애플리케이션 개발 플랫폼이다. 여러 오픈 소스 프로젝트가 있는데, 본 프로젝트에서는 GeoFire라는 Firebase 실시간 database를 이용하는 오픈 소스

라이브러리를 사용할 것이다. 이를 통해 key들의 집합을 저장하고 조회할 수 있다.

### 2.4. Project Scope

'나만의 감정 분석 다이어리'는 간편하게 일기를 기록할 수 있게 도와주며, 읽고 싶은 지난 일기를 쉽게 검색할 수 있도록 도와준다. 또한 일기의 기본적 기능 외에 본 시스템에는 감정 분석이라는 기능이 제공된다. 이는 일기를 통해 분석된 사용자의 감정을 피드백해주어 사용자의 정서적인 안정감을 찾도록 도와주는 역할을 해준다. '나만의 감정 분석 다이어리'는 크게 아래에 제시된 3가지의 system으로 구성된다.

첫번째, User management system은 사용자의 정보를 받고 관리하기 위한 system이다. User management system은 sign up subsystem, sign in subsystem 총 2가지의 subsystem으로 구성된다. sign up subsystem은 사용자가 본 시스템을 사용하기 위해 사용자 등록하는 부분을 담당하고, sign in subsystem은 등록된 id와 password를 통해 본 시스템에 로그인 하는 부분을 담당한다. 이 subsystem들은 아래 그림과 같은 구조를 통해 User management를 구성한다.

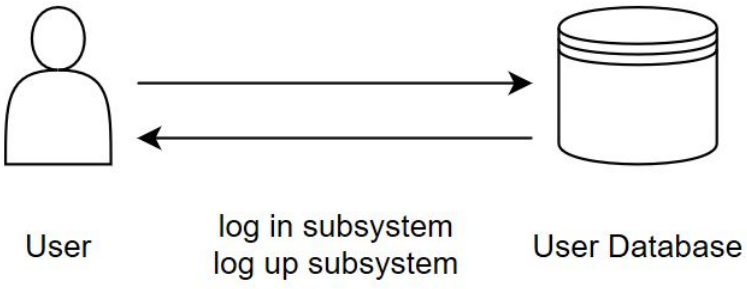


Figure 10. Architecture of User Management System

두번째, Diary system은 사용자가 일기를 기록하는 것과 관련된 system이다. Diary system은 daily record up subsystem, daily record down subsystem 총 2가지의 subsystem으로 구성된다. daily record up subsystem은 사용자가 일기를 작성하는 부분을 담당하고, daily record down subsystem은 사용자가

작성한 일기를 읽는 부분을 담당한다. 이 subsystem들은 아래 그림과 같은 구조를 통해 Diary system을 구성한다.

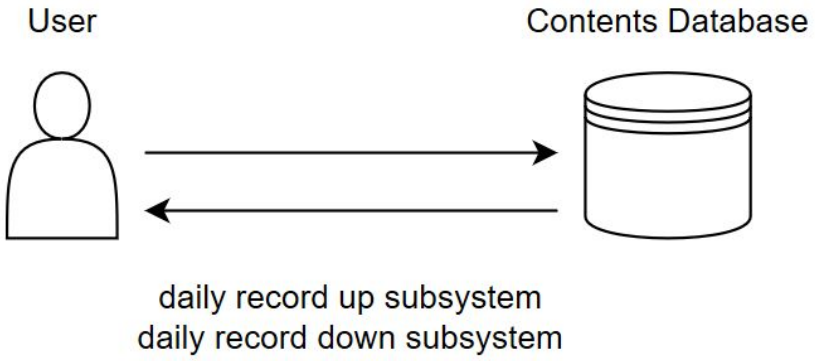


Figure 11. Architecture of Diary System

세번째, Emotion Extraction System은 사용자가 작성한 일기를 바탕으로 감정분석을 하는 것과 관련된 system이다. 아래 그림과 같은 구조를 통해 이뤄진다.

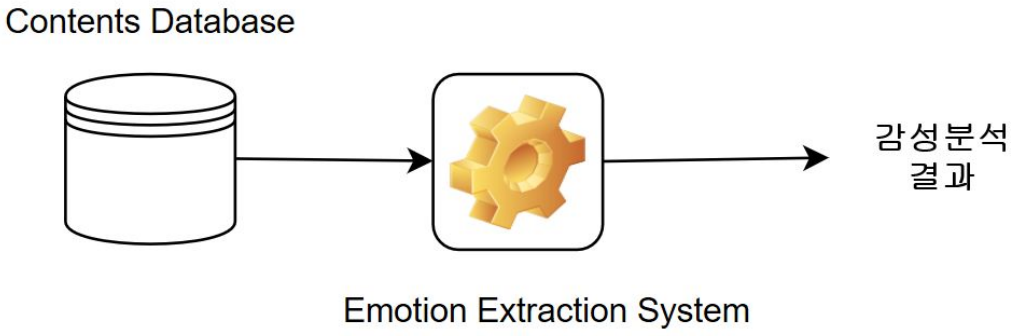


Figure 12. Architecture of Emotion Extraction System

# 3. System Architecture

## 3.1. Objectives

System Architecture에서는 현재 개발하는 시스템의 전체적인 구조에 대해 서술한다. Block diagram, Package diagram과 Deployment diagram을 이용해 시스템이 실제로 어떻게 사용되는지 설명한다.

## 3.2. System Organization

나만의 감정 다이어리는 Client-server Model을 사용하여 구현된다. 나만의 감정 다이어리 시스템은 서버를 이용하여 클라이언트로부터 오는 요청에 따라 크게 3개로 분류할 수 있다.

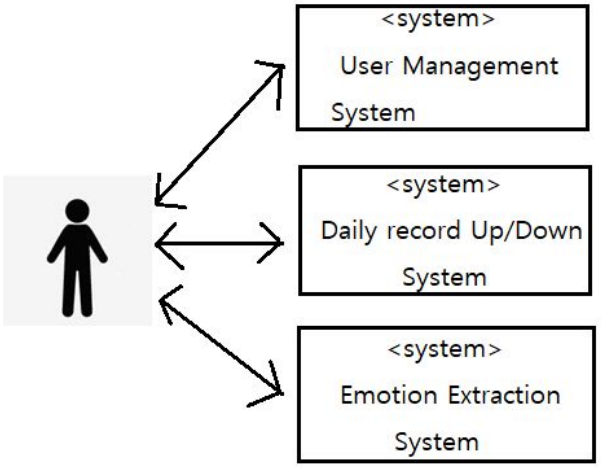


Figure 13. System Organization

서버 및 database, 감정 기능 분석 api는 구글에서 제공하는 firebase를 통하여 이루어지며, User management system, Daily record Up/Down system(이후 Dairy System이라 명칭함), Emotion Extraction system(이후 Emotion Analysis System이라 명칭함)은 서버 및 데이터베이스와 연결되어 이러한 기능들을 수행한다.

A. Dairy System

Dairy System은 사용자가 작성한 Dairy의 내용을 데이터베이스에 저장하고, 이전에 작성한 Dairy의 내용을 데이터베이스에서 가져오는 기능을 한다.

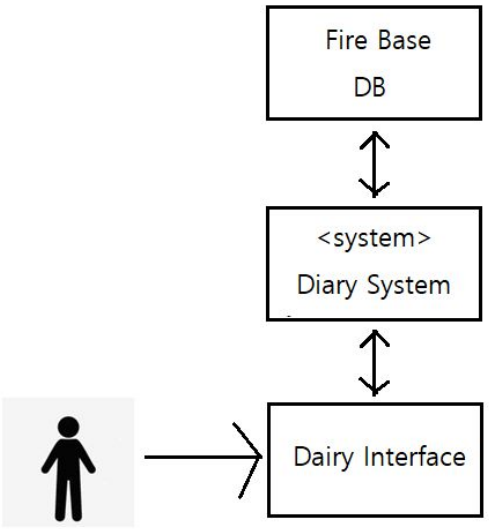


Figure 14. Diary System

B. User Management System

User Management System은 사용자의 이용과 관련된 시스템이다. 사용자의 회원가입과 로그인 기능을 제공하는 2 개의 하위 시스템으로 나뉜다.

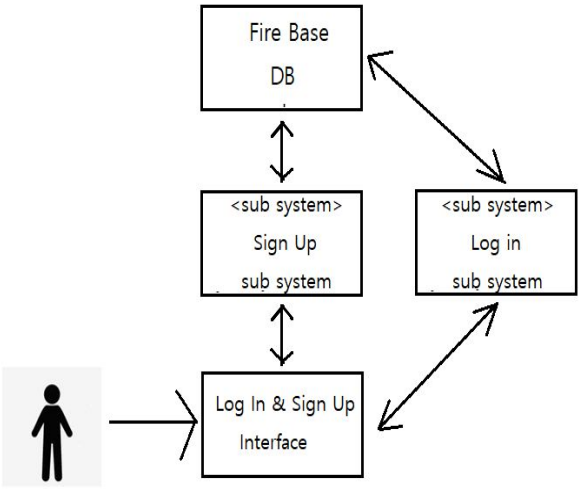


Figure 15. User Management System

C. Emotion Extraction System

Emotion Extraction System은 사용자가 입력한 dairy의 텍스트 부분을 분석하여 사용자의 감정 수준을 알 수 있게 해주는 기능이다.

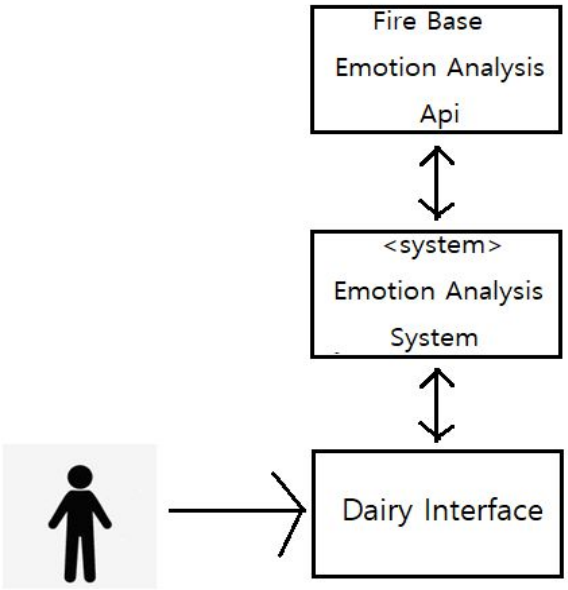


Figure 16. Emotion Extraction System

# 4. Diary System

## 4.1 Objectives

Diary System은 다이어리를 작성과 관련된 모든 기능을 포함한다. 다이어리를 작성할 때 채울 수 있는 내용에는 날짜, 날씨, 위치, 해시태그, 제목, 본문, 사진 및 동영상 등이 있다. User DB와 연동되어 다이어리 작성 기능을 제공한다. Class Diagram, Sequence Diagram, State Diagram을 통해 Diary System 구조와 사용자 및 DB와의 상호 작용을 설계한다.

## 4.2 Class Diagram

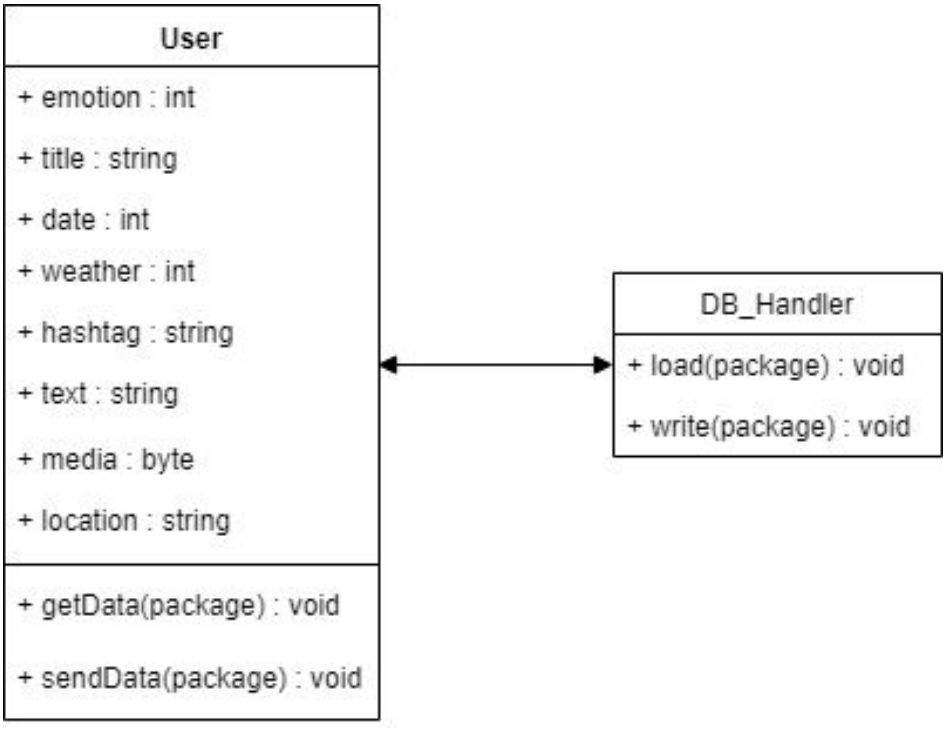


Diagram 1 Diary System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+ void load(package) : DB에서 데이터를 읽어온다.
+ void write(package) : DB에 데이터를 저장한다.

B. User

B.1. Attributes

+ emotion : 다이어리 텍스트의 감정추출 값
+ title : 다이어리의 제목
+ date : 다이어리의 날짜
+ weather : 다이어리의 날씨 정보
+ hashtag : 다이어리의 해시태그
+ text : 다이어리의 본문
+ media : 다이어리에 첨부된 사진 및 동영상
+ location : 다이어리의 위치 정보

B.2. Methods

+ void getData(package) : DB로부터 데이터를 받는다.
+ void sendData(package) : DB로 데이터를 보낸다.

### 4.3 Sequence Diagram

A. Diary load and write

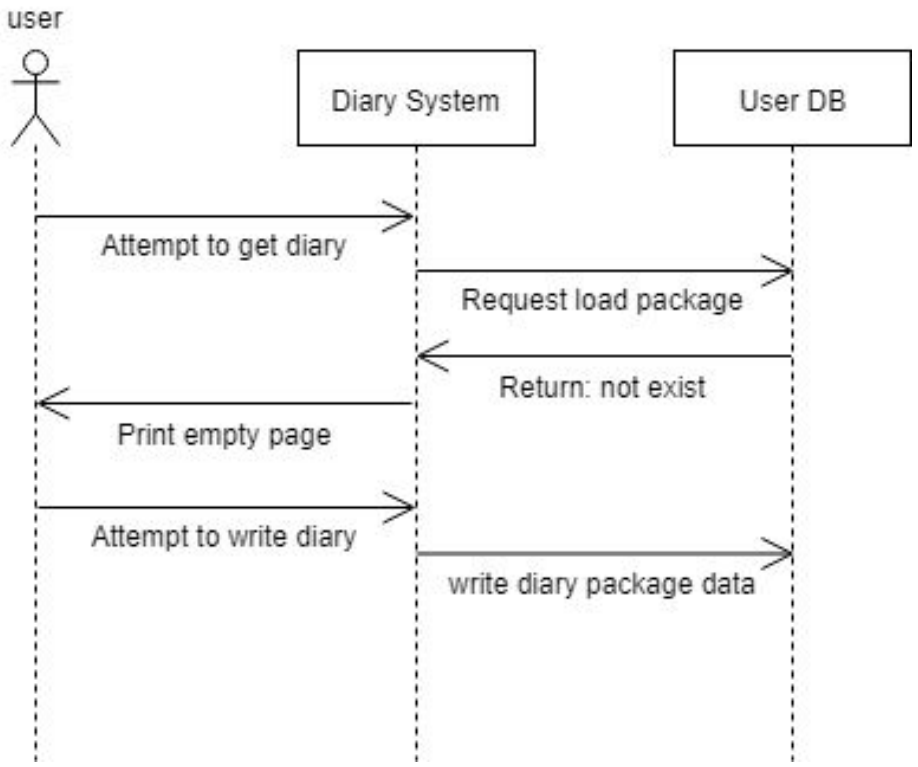


Diagram 2 Diary System Sequence Diagram

### 4.4 State Diagram

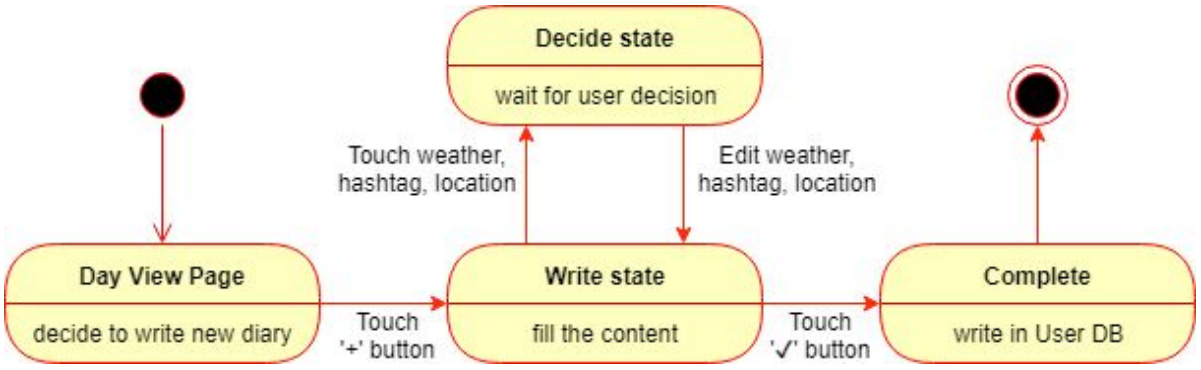


Diagram 3 Diary System State Diagram

# 5. Emotion Extraction System

## 5.1. Objectives

Emotion Extraction System은 사용자가 작성한 다이어리의 본문 내용을 분석하여 감정상태를 추출하고 기준치보다 우울도가 낮은 경우 힐링을 추천하는 기능이다. 추출한 감정상태 값을 이용하여 감정곡선 그래프를 그리고 사용자는 다이어리를 작성한 후 감정곡선을 볼 수 있다. Emotion Graph DB, User DB와 연동되어 감정추출 기능을 제공한다. Class Diagram, Sequence Diagram, State Diagram을 통해 Emotion Extraction System 구조와 사용자 및 DB와의 상호 작용을 설계한다.

## 5.2. Class Diagram

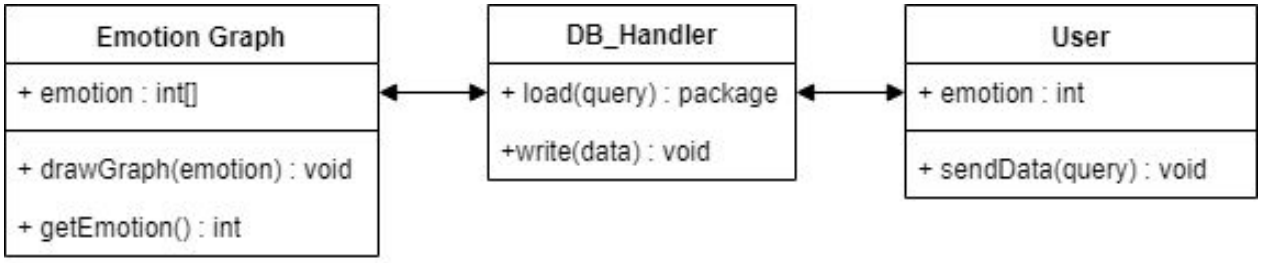


Diagram 4 Emotion Extraction System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+ package load(query) : 해당되는 DB에서 원하는 데이터를 읽어온다.

+ void write(data) : 해당되는 DB에 데이터를 저장한다.

B. Emotion Graph

B.1. Attributes

+ emotion : 그래프를 그릴 감정추출 값 저장 배열

B.2. Methods

+ void drawGraph(emotion) : 감정추출 값으로 감정곡선 그래프를 그린다.

+ int getEmotion() : DB로부터 감정추출 값을 받는다,

C. User

C.1. Attributes

+ emotion : 다이어리의 텍스트의 감정추출 값

C.2. Methods

+ void sendData(query) : DB로 데이터를 보낸다.

### 5.3. Sequence Diagram

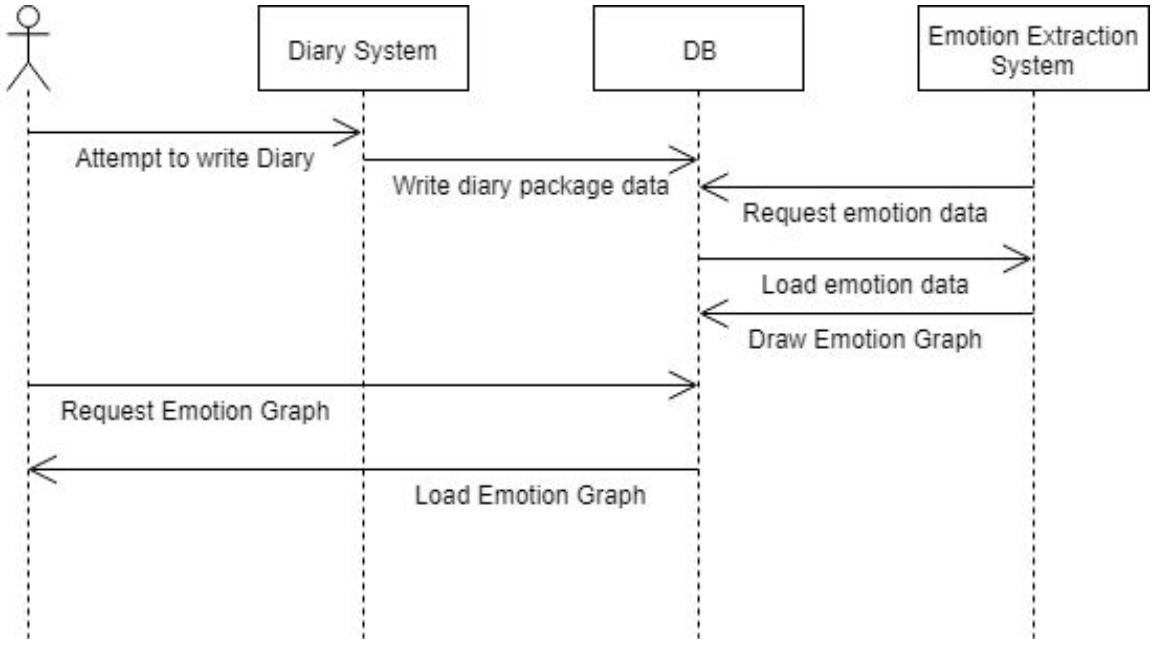


Diagram 5 Emotion Extraction System Sequence Diagram

### 5.4. State Diagram

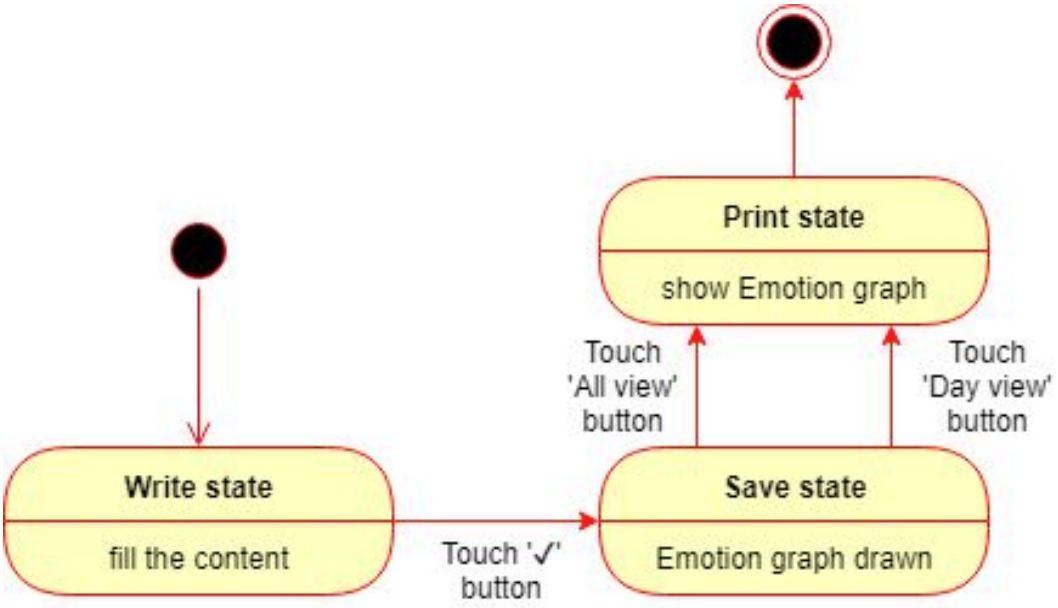


Diagram 6 Emotion Extraction System State Diagram

# 6. User Management System

## 6.1. Objectives

User Management System은 '나의 비밀 다이어리' 어플에 가입한 회원의 계정을 관리한다. Member DB와 연동되어 회원 가입과 로그인 기능을 제공한다. Class Diagram, Sequence Diagram, State Diagram을 통해 User Management System의 구조와 회원 및 DB와의 상호 작용을 설계한다.

## 6.2. Class Diagram

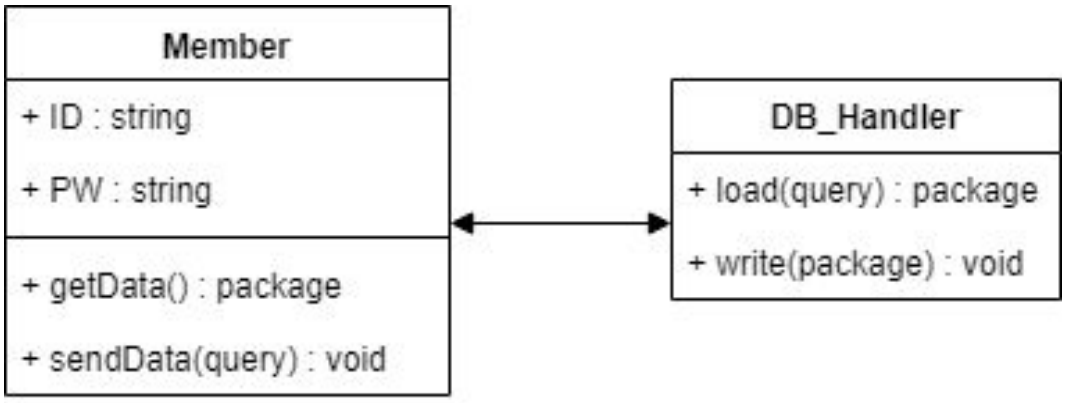


Diagram 7 User Management System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+ package load(query) : DB에서 원하는 데이터를 가져온다.

+ void write(package) : DB에 데이터를 저장한다.

B. Member

B.1. Attributes

+ ID : 사용자의 아이디 정보

+ PW : 사용자의 패스워드 정보

B.2. Methods

+ package getData() : DB로부터  데이터를 받는다.

+ void sendData(query) : DB로 데이터를 보낸다.

### 6.3. Sequence Diagram

A.  Sign Up

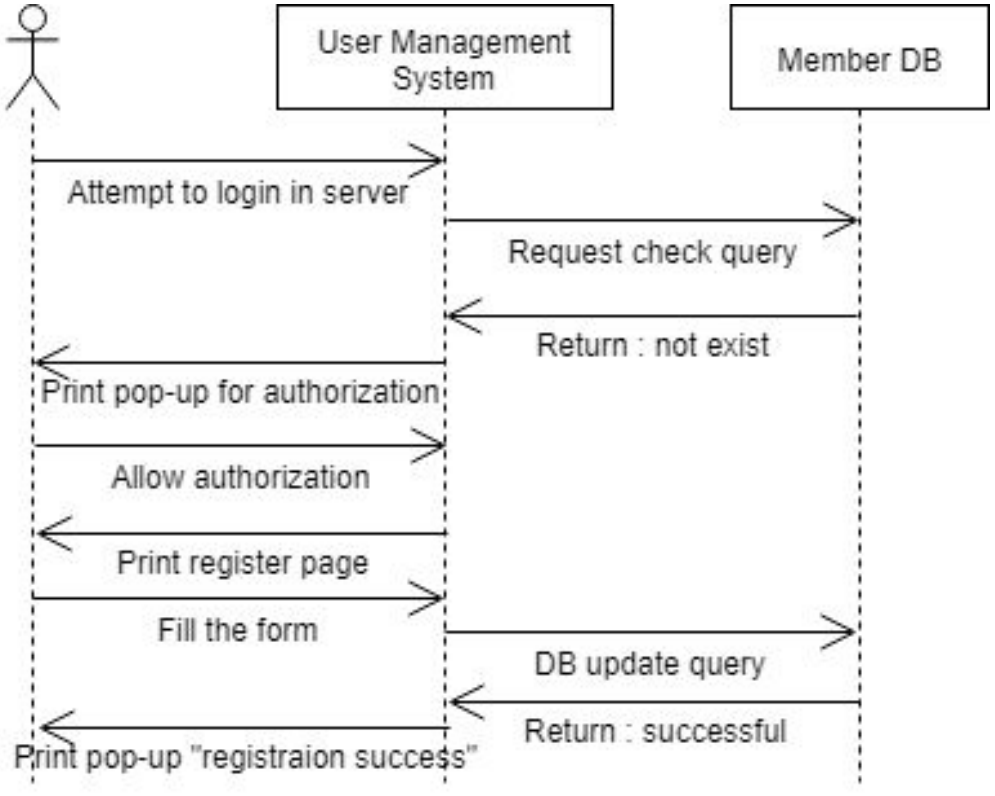


Diagram 8 User Management System(Sign up) Class Diagram

B. Login

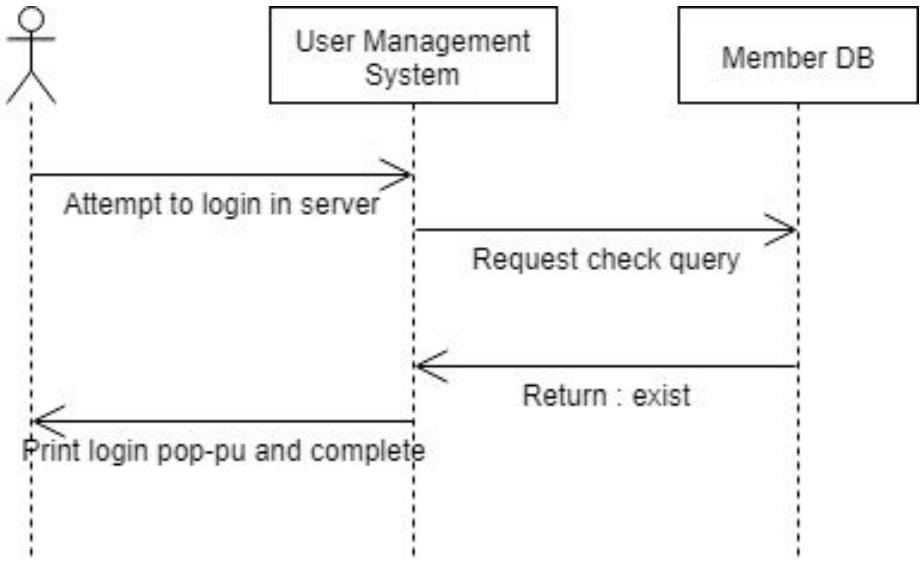


Diagram 9 User Management System(Login) Class Diagram

### 6.4. State Diagram

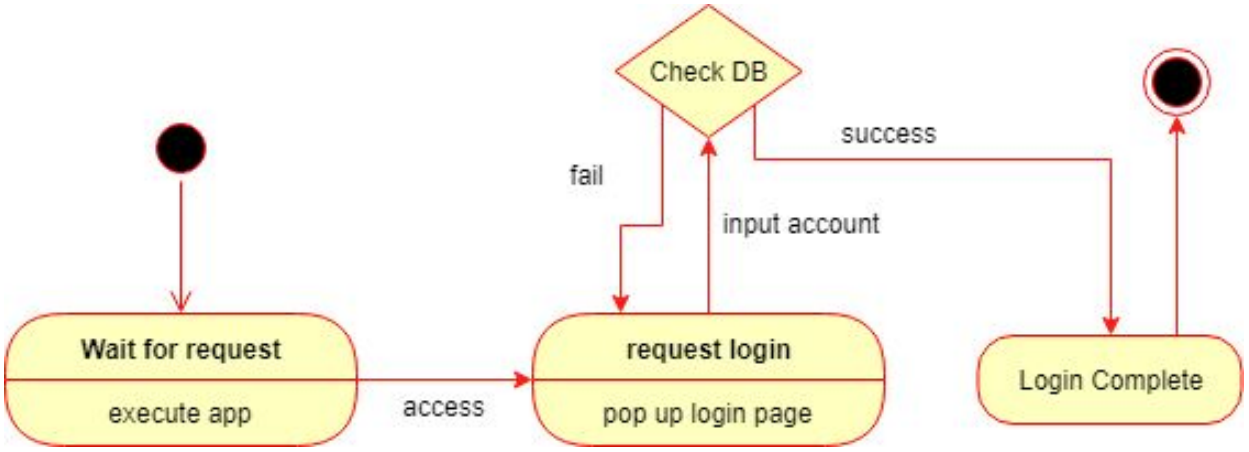


Diagram 10 User Management System State Diagram

# 7. Protocol Design

## 7.1. Objectives

Protocol Design에서는 Subsystem들이 상호작용하는 protocol에 대해 기술한다. protocol의 형식은 JSON(JavaScript Object Notation)을 기본으로 삼으며, protocol에서 통신하는 message의 형식, 용도 그리고 의미를 서술한다.

## 7.2. JSON

JSON은 사람이 읽고 쓰는데 사용하기 편한 data 교환 형식이다. 이것은 사람뿐만 아니라 기계가 parsing하고 generating하는데에도 쉽게 쓰일 수 있다. JSON은 텍스트 형식으로써 다른 언어와 독립된 완전한 하나의 언어이지만, 프로그래머에게 친숙한 C, C++, C#, Java, JavaScript, Perl, Python등과 같은 C-family 언어의 관습을 따르고 있다.

## 7.3. Protocol Design

A.Overview
HTTP통신에서 client와 server가 서로 전송하는 메세지의 형태를 용도별로 정의한다. Client에서의 요청(request) 메세지와 server에서의 응답(response)메세지로 구분한다.

B. Registration Protocol

B.1. Request

| Attribute | Value |
|---|---|
| **Uid** | 사용자의 ID |
| **Pw** | 사용자의 ID에 해당하는 password |

Table 2. Registration Protocol Request

B.2. Response

| Attribute | Value |
|---|---|
| **Reg_success** | 회원가입 성공 여부(True/False) |

Table 3. Registration Protocol Response

C. Login Protocol

C.1. Request

| Attribute | Value |
|---|---|
| **Id** | 사용자의 ID |
| **Pw** | 사용자의 ID에 해당하는 password |

Table 4. Login Protocol Request

C.2. Response

| Attribute | Value |
|---|---|
| **Login_success** | 로그인 성공 여부(True/False)<br>True일 경우 세션 생성 |

Table 5. Login Protocol Response

D. Uploading Picture Protocol

D.1. Request

| Attribute | Value |
|---|---|
| **UPUid** | 사진을 업로드하는 사용자의 ID |

| | |
|---|---|
| **Date** | 해당 다이어리의 날짜 |
| **Picture** | 업로드할 사진 |

Table 6. Uploading Picture Protocol Request

D.2. Response

| Attribute | Value |
|---|---|
| **UP_success** | 사진 업로드 성공 여부(True/False) |

Table 7. Uploading Picture Protocol Response

E. Uploading Text Protocol

E.1. Request

| Attribute | Value |
|---|---|
| **UTUid** | 글을 업로드하는 사용자의 ID |
| **Date** | 해당 다이어리의 날짜 |
| **Text** | 업로드할 글 |

Table 8. Uploading Text Protocol Request

E.2. Response

| Attribute | Value |
|---|---|
| **UT_success** | 글 업로드 성공 여부(True/False) |

Table 9. Uploading Text Protocol Response

F. Saving Place Protocol

### F.1. Request

| Attribute | Value |
|---|---|
| **SPUid** | 장소를 저장하는 사용자의 ID |
| **Date** | 해당 다이어리의 날짜 |
| **Place** | 저장할 장소 |

Table 10. Saving Place Protocol Request

### F.2. Response

| Attribute | Value |
|---|---|
| **SP_success** | 장소 저장 성공 여부(True/False) |

Table 11. Saving Place Protocol Response

## G. Saving Weather Protocol

### G.1. Request

| Attribute | Value |
|---|---|
| **SWUid** | 날씨를 저장하는 사용자의 ID |
| **Date** | 해당 다이어리의 날짜 |
| **Weather** | 저장할 날씨 |

Table 12. Saving Weather Protocol Request

### G.2. Response

| Attribute | Value |
|---|---|

| | |
|---|---|
| **SW_success** | 날씨 저장 성공 여부(True/False) |

Table 13. Saving Weather Protocol Response

## H. Uploading Tag Protocol

### H.1. Request

| Attribute | Value |
|---|---|
| **UTUid** | 태그를 업로드하는 사용자의 ID |
| **Date** | 해당 다이어리의 날짜 |
| **Tag** | 저장할 tag의 내용 |

Table 14. Uploading Tag Protocol Request

### H.2. Response

| Attribute | Value |
|---|---|
| **UT_success** | 태그 업로드 성공 여부(True/False) |

Table 15. Uploading Tag Protocol Response

## I. Searching Tag

### I.1. Request

| Attribute | Value |
|---|---|
| **STUid** | tag를 검색하는 사용자의 ID |
| **Tag** | 검색할 tag 내용 |

Table 16. Searching Tag Protocol Request

### I.2. Response

| Attribute | Value |
|---|---|
| **ST_success** | 태그 검색 성공 여부 |
| **Date** | 태그와 관련된 일기의 날짜 |

Table 17. Searching Tag Protocol Response

## J. Showing Selected Diary

### J.1. Request

| Attribute | Value |
|---|---|
| **SSUid** | 태그와 관련된 날짜중 특정 날짜를 선택하는 사용자의 ID |
| **Selected_date** | 태그와 관련된 날짜중 사용자가 선택한 특정 날짜 |

Table 18. Showing Selected Diary Protocol Response

J.2. Response

| Attribute | Value |
| --- | --- |
| **SS_success** | 태그와 관련된 특정 날짜 선택 성공 여부 |
| **Diary** | 선택된 날짜의 다이어리 내용 |

Table 19. Showing Selected Diary Protocol Response

K. Selecting Date Protocol

K.1. Request

| Attribute | Value |
| --- | --- |
| **SDUid** | 특정 날짜를 선택하는 사용자의 ID |
| **SDate** | 사용자가 선택한 날짜 |

Table 20. Selecting Date Protocol Request

J.2. Response

| Attribute | Value |
| --- | --- |
| **SD_success** | 날짜 선택 성공 여부 |
| **Diary** | 선택된 날짜의 다이어리 내용 |

Table 21. Selecting Data Protocol Response

L. Analyzing Feeling Protocol

L.1. Request

| Attribute | Value |
| --- | --- |

| | |
|---|---|
| **AFUid** | 감정분석이 되는 사용자의 ID |
| **TEXT** | 해당 사용자의 최근 일주일간의 글 |

Table 22. Analyzing Feeling Protocol Request

L.2. Response

| Attribute | Value |
|---|---|
| **AF_success** | 감정분석의 성공 여부 |
| **Feeling** | 분석된 감정 |

Table 23. Analyzing Feeling Protocol Protocol Response

# 8. Database Design

## 8.1 Objective

Database design에서는 requirement specification을 기반으로 만들 소프트웨어에 필요한 데이터베이스를 설계하고 설명한다. ER diagram을 통해 데이터베이스의 entity와 entity간의 relation을 제시하고, ER diagram을 기반으로 relational schema를 제시한다. Relational schema를 통해 table의 attribute간의 dependency를 확인하고 normalization과정을 통해 data redundancy를 줄일 수 있는지 확인하고, SQL DDL을 작성한다.

## 8.2 ER Diagram

A. Entity

A.1 User

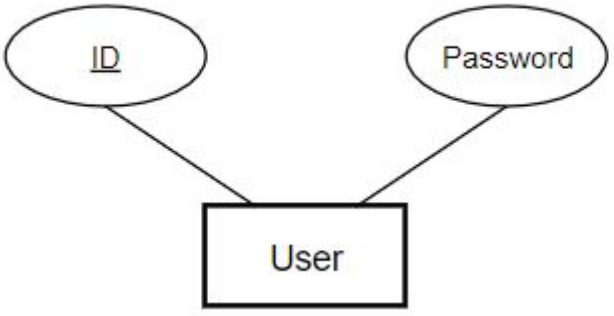


Diagram 11. User Entity Diagram

User entity는 사용자의 정보를 나타낸다. 각 사용자는 고유의 ID를 가지고 있으며, id에 대응되는 하나의 password를 가진다. 사용자를식별하는 primary key로는 ID를 사용한다.

A.2 Diary

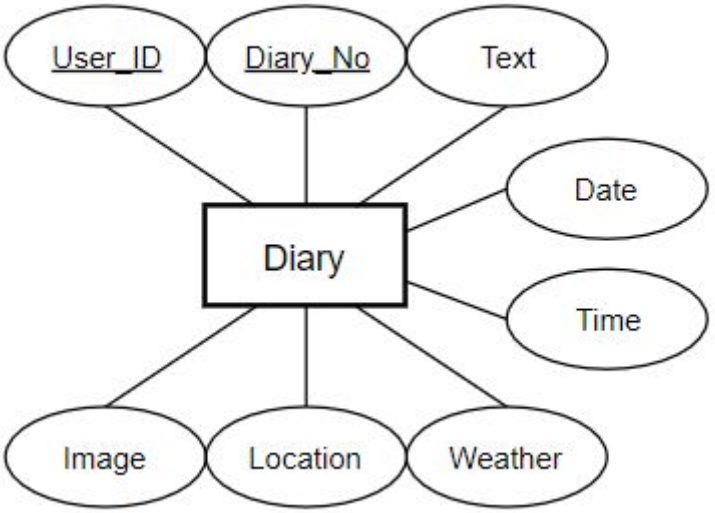


Diagram 12. Diary Entity Diagram

Diary Entity는 diary의 정보를 나타낸다. 먼저, diary를 식별할 정보로서diary를 작성한 사용자의 정보인 User_ID를 저장하고, 그 사용자의 diary를 구별할 Diary_No를 가지고 있다. 또한 Diary의 내용을 담은 정보는 Text, Image가 있고, diary의 작성 시간인 Date와 Time의 정보가 있고, Diary를 작성할 때의 사용자의 location과 weather정보를 저장한다. Diary entity의 primary key로는 User_ID, Diary_No 두개의 attribute를 사용한다.

A.3 Tags

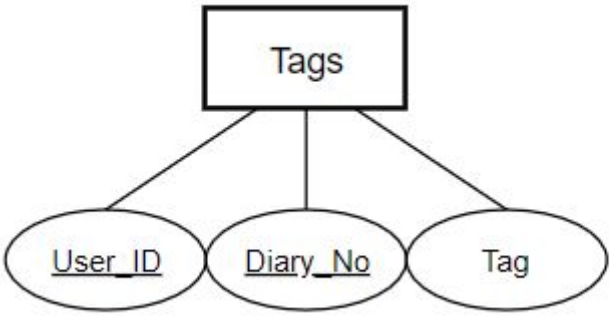


Diagram 13. Tags Entity Diagram

Tags Entity는 사용자의 Tag의 정보를 나타낸다. Tag를 diary의 attribute에 넣지 않고 분리한 이유는 tag를 검색할 때, 빠른 search를 위해서다. Tags의 attribute를 보면, 어떤 사용자가 사용했는지를 식별할 User_ID, 어떤 diary의 tag인지를 식별할지의 Diary_No, tag의 내용인 Tag가 있다. Primary Key로는 User_ID, Diary_No가 있다.

B. Relation Schema

B.1 User Has Diary

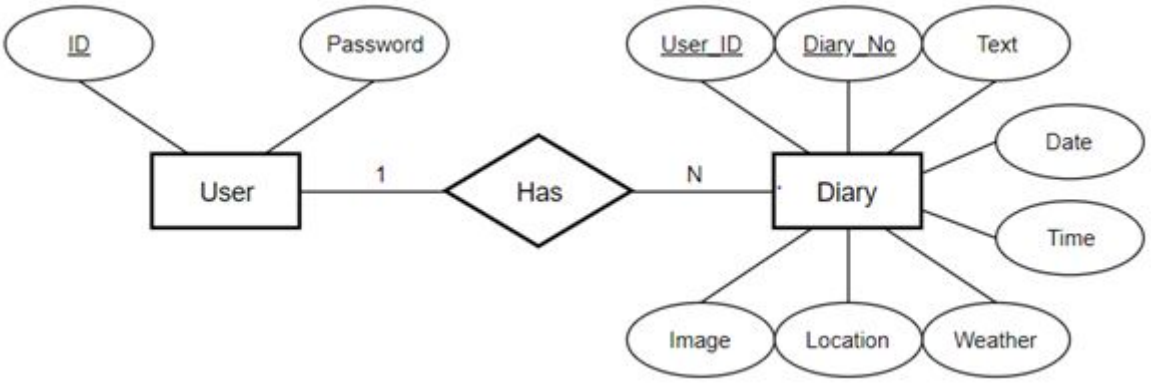


Diagram 14. User Has Diary Relation Diagram

User가 diary를 소유하고 있으므로 user과 diary 사이에는 'Has' Relationship이 있고, 한 명의 user가 여러 개의 diary를 가질 수 있으므로 1:N 관계를 가진다.

B.2 Diary Has Tags

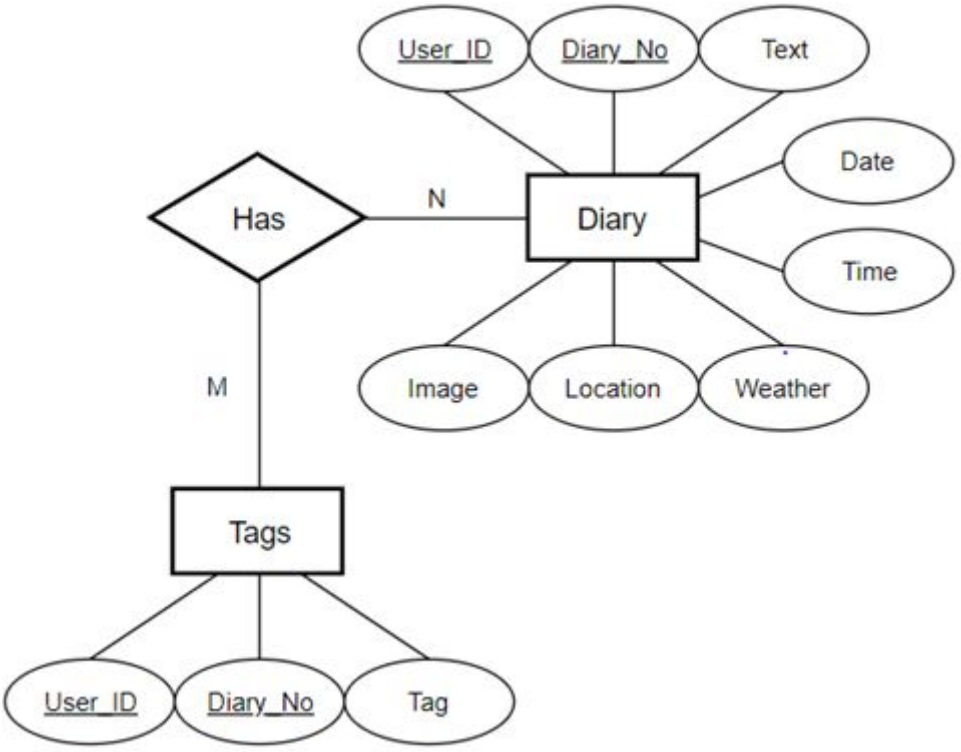


Diagram 15. Diary Has Tags Relation Diagram

Diary마다 유저가 설정한 Tag를 가지고 있으므로 Diary와 Tag 사이에는 ‘Has’ relationship이 있다. 또한, 하나의 다이어리는 여러 개의 tag를 가지고 있을 수 있으므로 M:N의 관계가 있다.

C. ER Diagram

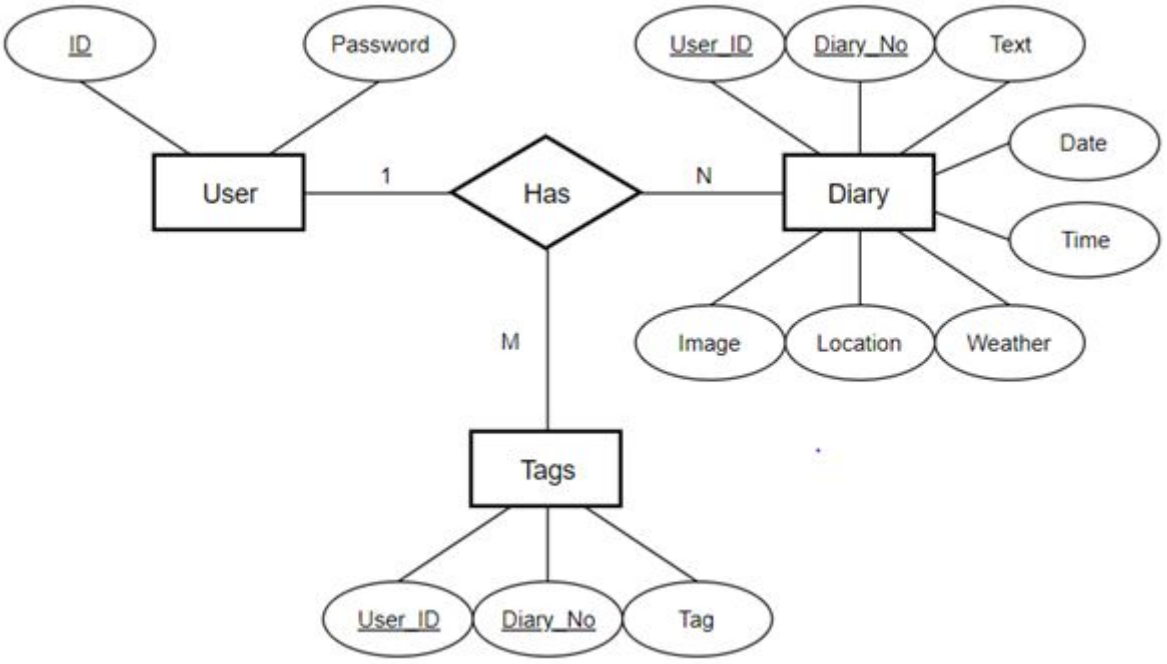


Diagram 16. ER Diagram

따라서 전체적인 ER Diagram은 위 그림처럼 나타낼 수 있다.

### 8.3 Relational Schema

A. User

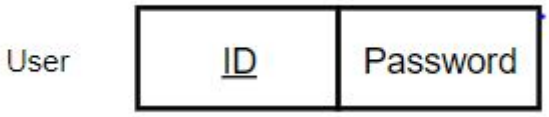


Diagram 17. User Schema

ID(Primary Key) : 사용자의 고유 id, 자료형은 Text를 사용한다.

Password : 사용자의 id에 대응하는 비밀번호. 자료형은 Text를 사용한다.

Functional Dependency : {ID} -> {Password}

Description : User table을 통해 사용자의 회원 정보를 저장한다. ID와 password로 구성되어 있으며, 두개의 value 모두 null값을 허용하지 않는다.

B. Diary


Diagram 18. Diary Schema

User_ID(Primary Key), (Foreign Key, reference User(ID)) : Diary의 작성자의 ID, 누가 diary를 작성했는지 확인하는 attribute이다. 자료형은 Text이다.

Diary_No(Primary Key) : 사용자에 따른 Diary의 번호이다. 자료형은 INT이다.

Text : 사용자가 작성한 Diary의 Text 내용이다. 자료형은 Text이다.

Image : 사용자가 입력한 Diary의 picture이다. 자료형은 BLOB이다.

Date : 사용자가 Diary를 작성한 날짜이다. 자료형은 Date이다.

Time : 사용자가 Diary를 작성한 시간이다. 자료형은 Time이다.

Location : 사용자가 Diary를 작성한 장소이다. 자료형은 Text이다.

Weather : 사용자가 Diary를 작성한 장소의 날씨정보이다. 자료형은 Text이다.

Functional Dependency : {User_ID, Diary_No} -> All
{Date, Time, Location} -> {Date, Time, Location, Weather}

Description : Diary Table에는 User_ID와 Diary_No에 맞는 Diary의 data를 저장한다. Image는 null값을 가질 수 있지만, 나머지는 모두 null값을 허용하지 않는다.

C. Tags

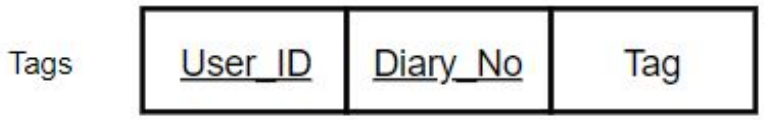


Diagram 19. Tags Schema

User_ID(Primary Key), (Foreigner Key reference User(ID)) : Tag에 해당하는 diary를 작성한 user의 id. 자료형은 Text이다.

Diary_No(Primary Key), (Foreigner Key reference Diary(Diary_No)) : Tag에 해당하는 Diary의 number. 자료형은 INT이다.

Tag : 태그 정보. 자료형은 Text이다.

Functional Dependency : {User_ID, Diary_No} -> {User_ID, Diary_No, Tag}

Description : Diary에 따른 tag가 저장되어 있는 table이다. 모든 data에 null값을 허용하지 않는다.

D. Relational Schema Diagram

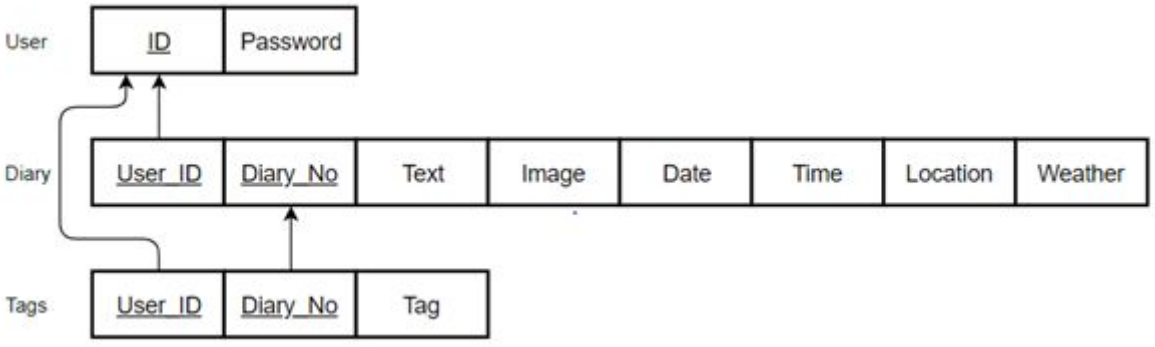


Diagram 20. Relational Schema Diagram

### 8.4 Normalization

Normalization은 데이터베이스의 설계를 재구성하는 방법으로, Table의 redundancy를 제거하고 table을 효율적으로 관리하기 위해 사용한다.
일반적으로 정규화는 크고 제대로 조직되지 않은 테이블과 관계들을 작고 잘 조직된 테이블과 관계들로 나누는 과정을 포함한다.

사실 위의 Relational schema는 normalization을 적용할 부분이 두 가지 밖에 없다. 먼저, Diary Table에서 {Date, Time, Location} -> {Weather}부분을 새로운 table로 분리할 수 있으나, 사용자의 시간, 장소가 모두 일치하는 경우가 많지 않다고 판단하여 normalization을 하지 않는 편이 더 낫다고 판단했다.

또한, {User_ID, Diary_No} -> {Tag} 의 functional dependency를 이용하여 Tags table과 Diary table을 분리했다. 이는 하나의 Diary에 여러 개의 tag가 저장될 수 있기 때문이었다.

### 8.5 SQL DDL

A. User

```
Create Table User {
        ID VARCHAR(15) NOT NULL,
        Password VARCHAR(20) NOT NULL,
        PRIMARY KEY (ID)
};
```

B. Diary

```
Create Table Diary {
        User_ID VARCHAR(15) NOT NULL,
        Diary_No INT NOT NULL,
        Text VARCHAR(1000) NOT NULL,
        Image BLOB,
        Date DATE NOT NULL,
        Time TIME NOT NULL,
        Location VARCHAR(100) NOT NULL,
        Weather VARCHAR(100) NOT NULL,
        PRIMARY KEY (User_ID, Diary_no)
        FOREIGN KEY (User_ID) REFERENCES User(ID)
};
```

C. Tags

```
Create Table User {
        User_ID VARCHAR(15) NOT NULL,
        Diary_No INT NOT NULL,
        Tag VARCHAR(20) NOT NULL,
        PRIMARY KEY (ID, Diary_No),
        FOREIGN KEY (User_Id) REFERENCES User(ID),
        FOREIGN KEY (Diary_No) REFERENCES Diary(Diary_No)
};
```

# 9. Testing Plan

## 9.1 Objection

시스템이 의도한 방향대로 실행되고 시스템 내부의 결함을 찾기 위해 testing을 한다. 이를 위해 설계단계에 미리 계획한다. Testing Plan에서는 Testing Policy와 여러 Test Case에 대해 기술한다.

## 9.2 Testing Policy

나만의 감정 다이어리 시스템의 개발에는 크게 3 단계로 나누어 testing을 진행한다. Developing Testing, Release Testing, User Testing으로 나눠지며, Developing Testing은 다시 Component Testing, Integrating Testing, System Testing, Acceptance Testing의 4 단계로 나눠진다.

A. Development Testing

개발 과정에서 수행되는 testing이다.

Component Testing은 각 component단위로 개발이 진행되면 각 component가 정상적으로 작동하는지 확인하는testing이다.

Integrating Testing은 component를 통합하는 과정에서 수행하는 testing이다.

System Testing은 모든 시스템을 합친 후 정상적으로 동작하는지 확인하는 testing이다.

Acceptance Testing은 사용자의 정보를 이용하여 시스템에 대한 사용자의 요구사항을 testing 하는 것이다.

B. RELEASE TESTING

출시 전 최종적으로 하는 testing이다. 모든 요구사항이 반영되었는지를 확인한다.

C. USER TESTING

사용자의 환경에서 진행하는 testing이다.

### 9.3 Test case

A. Dairy System

A.1 Daily record Up system

1) User:  Dairy의 내용을 입력한 후 'Save' button을 누른다.

2) 시스템 동작: 양식에 맞게 data가 입력되었는지 확인하고, 이를 데이터베이스에 저장한다.

2-1) (Daily record Up 성공) 시스템 동작: 토스트 형태로 데이터베이스 저장 성공 여부를 사용자에게 알려주고,  데이 뷰 화면으로 이동한다.

2-2) (Daily record Up  실패) 시스템 동작: 데이터 베이스에 중복된 데이터가 입력되거나, 양식에 맞는 데이터가 입력되지 않거나, 네트워크가 끊길 경우 토스트 형태로 데이터베이스 저장  실패 여부를 사용자에게 알려준다.

A.2 Daily record Down system

1) User:  보고자 하는 날짜를 선택한다.

2) 시스템 동작: 데이터베이스에 저장되어 있는 데이터를 가져온다.

2-1) (Daily record Down 성공) 시스템 동작: 데이 뷰 화면으로 이동한다.

2-2) (Daily record Down 실패) 시스템 동작: 네트워크가 끊겨 정상적으로 데이터를 받아오지 못한 경우 토스트 형태로 Daily record Down 실패 여부를 사용자에게 알려준다.

B. User Management System

B.1 Sign up sub system

1) User:  ID,PW 등 개인 정보를 입력한 후 ‘Sign Up’ button을 누른다.

2) 시스템 동작: 회원가입 양식에 맞게 data가 입력되었는지 확인하고, 이를 데이터베이스에 저장한다.

2-1) (회원가입 성공) 시스템 동작: 토스트 형태로 회원가입 성공 여부를 사용자에게 알려주고,  로그인 화면으로 이동한다.

2-2) (회원가입 실패) 시스템 동작: 데이터 베이스에 중복된 데이터가 입력되거나, 양식에 맞는 데이터가 입력되지 않거나, 네트워크가 끊길 경우 토스트 형태로 회원가입 실패 여부를 사용자에게 알려준다.

B.2 Log in sub system

1) User:  ID,PW 등 개인 정보를 입력한 후 ‘Log in’ button을 누른다.

2) 시스템 동작: 양식에 맞게 data가 입력되었는지 확인하고, 이를 데이터베이스에 저장되어 있는 데이터와 비교한다.

2-1) (로그인 성공) 시스템 동작: 토스트 형태로 로그인 성공 여부를 사용자에게 알려주고, Home 화면으로 이동한다.

2-2) (로그인 실패) 시스템 동작: 데이터 베이스와 다른 데이터가 입력되거나, 네트워크가 끊길 경우 토스트 형태로 로그인 실패 여부를 사용자에게 알려준다.

C. Emotion Extraction System

1) User: Dairy에 text data를 입력한 후 'Analysis' button을 누른다.

2) 시스템 동작: 양식에 맞게 data가 입력되었는지 확인하고, 이를 구글 자연어 처리 API를 통해 분석한다.

2-1) (분석 성공) 시스템 동작: 토스트 형태로 분석 성공 여부를 사용자에게 알려주고, 그 수치를 작성하고 있던 페이지에 나타낸다.

2-2) (분석 실패) 시스템 동작: 분석이 실패하거나, 네트워크가 끊길 경우 토스트 형태로 분석 실패 여부를 사용자에게 알려준다.

# 10. Development Environment

## 10.1 Objective

Development Environment에서는 소프트웨어 개발 환경에 대해서 서술한다. 소프트웨어를 개발하는 데 사용할 통합 개발 환경(IDE), 프로그래밍 언어, 그리고 사용할 여러 system들에 대해 설명한다.

## 10.2 IDE and Programming Language

A. IDE & Database

A.1 Android Studio, Android


Figure 17. Android Studio

Android Studio는 구글이 JetBrains 사의 IntelliJ IDEA를 기반으로 만든 통합 개발 환경으로, Android 앱 개발을 위해 만들어졌다. Windows, Mac OS, Linux 기반 운영체제에서 모두 사용할 수 있으며, 기존에 Android 앱 개발로 사용하던 Eclipse ADT 개발을 중단하고 지원을 시작하였다. 우리의 앱 개발은 2019년 5월 18일 기준 최신버전인 Android Studio 3.4를 이용하여 개발에 사용할 예정이다.

Android Studio IDE를 선택한 이유는, 먼저 무료로 이용할 수 있고, 조원 중 한명이 사용해 본 경험이 있으며, android studio의 project를 조원의 대부분이 친숙한 JAVA언어로 프로그래밍 할 수 있다는 점에서 만장일치로 선택하게 되었다.

## A.2 Database


Figure 18. Firebase

Firebase는 비교적 최근에 만들어진 모바일 데이터베이스 서비스로, 무료 open source로 제공되는 Database이다. Firebase 모바일 데이터베이스는 대표적인 DBMS인 SQLite를 대체할 수 있는 real-time 모바일 데이터베이스로 성능이 좋고 빠르며, iOS 뿐만 아니라 안드로이드, React Native, Xamarin(자마린) 등 여러 플랫폼에서 사용할 수 있다. 데이터 모델을 따로 만들지 않고도 레이어를 쉽게 구현할 수 있으며 최적화된 앱 개발에 도움이 된다.

여러 모바일 데이터베이스 중 firebase을 선택한 이유는, 먼저 open source이므로 무료로 사용할 수 있고, AWS나 Naver cloud와 다르게 결제가 필요하지 않고, Android platform을 지원하며, 최근에 개발되었기 때문에 성능도 우수하기 때문에 선택했다. 비록 SQLite가 잘 알려져 있지만, SQLite를 사용해 본

조원이 없고, 처음 시작해야 하기 때문에 성능이 좋은 firebase 데이터베이스를 사용하기로 하였다.

### A.3 Operating System & Programming Language


Figure 19. Android

Android는 Linux 커널 및 기타 open source 소프트웨어의 수정된 버전을 기반으로 Google 이 개발한 모바일 운영체제이며 주로 스마트폰이나 태블릿과 같은 터치 스크린 모바일 장치를 위해 설계되었다. 안드로이드는 개발자들이 Java와 Kotlin언어로 응용 프로그램을 작성할 수 있게 하였으며, 컴파일된 바이트코드를 구동할 수 있는 런타임 라이브러리를 제공한다. 또한 안드로이드 소프트웨어 개발 키트(SDK)를 통해 응용 프로그램을 개발하는 데 필요한 각종 도구와 응용 프로그램 인터페이스(API)를 제공한다. 대다수의 사용자가 안드로이드 운영체제를 사용 중이며, 안드로이드 앱 개발 경험이 있는 팀원이 있어서 Diary 앱을 개발하는 데 선택하게 되었다.


Figure 20. Java

Java 프로그래밍 언어는 Sun Microsystems에서 1995년에 개발한 객체 지향 프로그래밍 언어이다. Android Studio의 개발 언어 중 하나이고, 세계적으로 굉장히 많이 사용되는 대중적인 프로그래밍 언어이다. 그러므로 수많은 open source가 존재하고, API들도 많이 존재한다.

Java 프로그래밍 언어를 선택한 이유는, Android studio에서 사용된다는 점이 가장 큰 이유이고, java 프로그래밍 언어는 대부분의 조원들이 사용할 수 있으며, 높은 안전성을 가지고 있다고 생각하여 선택하게 되었다.

### 10.3 Code Management System


Figure 21. Github

GitHub은 Git를 사용한 버전 제어를 위한 웹 기반 호스팅 서비스이다. Branch 기능을 이용하여 각자 개발한 기능들을 업로드할 수 있으며, merge 기능을 이용하여 integration 작업을 보다 쉽고 빠르게 할 수 있다. 또한 issue 탭을 이용하여 현재 프로그램의 문제점, bug등을 공유하고 토론할 수 있다.

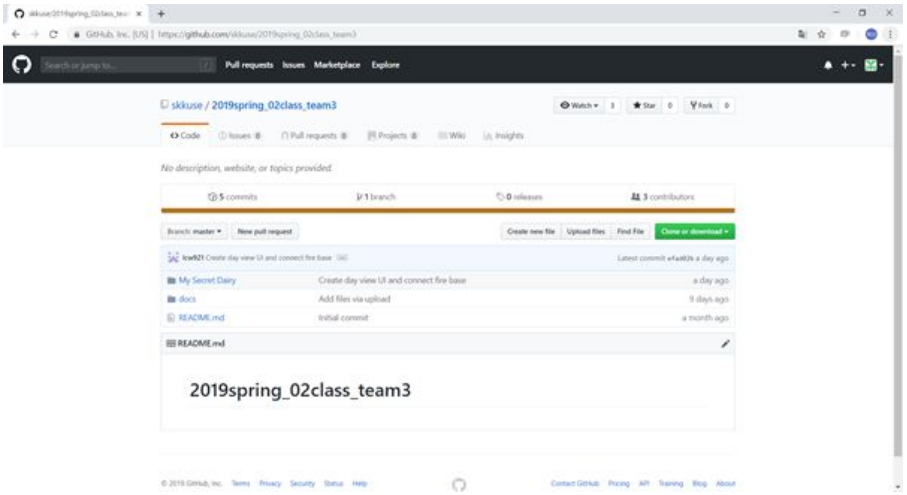


Figure 22. Team project Using Github

또한, 작업한 코드들과 문서들을 모두 github에 업로드하고, 공개할 것이다. 우리 팀의 github 주소는 https://github.com/skkuse/2019spring_02class_team3이다.

# 11. Development Plan

## 11.1 Objectives

Development plan에서는 개발 계획에 대해 서술한다. Gantt chart를 이용해서 전체적인 개발 수행 계획과 그 일정에 대하여 서술하겠다. 또한, 개발 수행 내역을 Gantt chart에 추가한다.

## 11.2 Gantt Chart

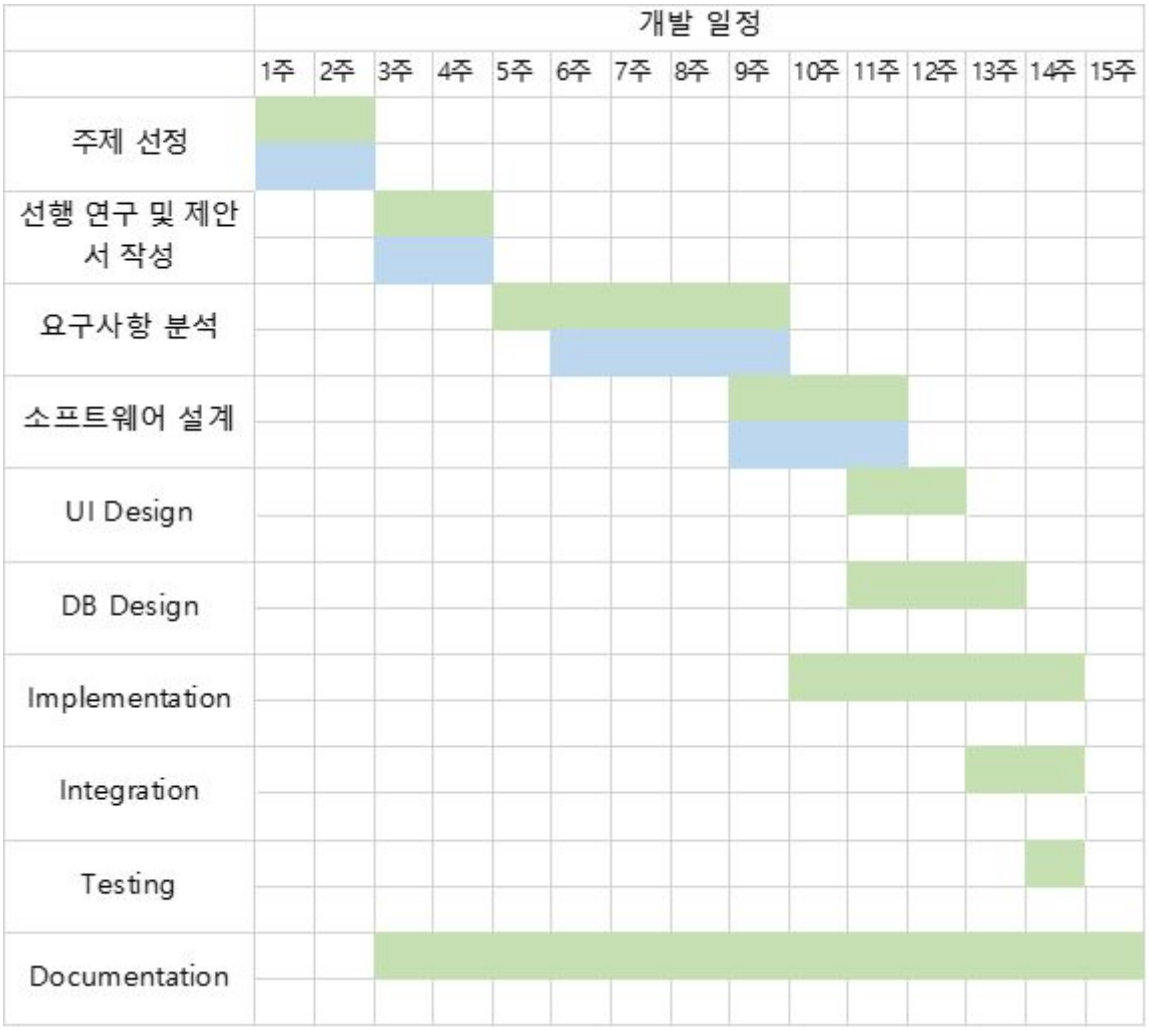


Table 24. Gantt Chart

# 12. Index

## 12.1 Table Index

## 12.2 Figure Index

### 12.3 Diagram Index

## 13. References


UML

FOLDOC (2001). Unified Modeling Language last updated 2002-01-03. Accessed 6 feb 2009.

Grady Booch, Ivar Jacobson & Jim Rumbaugh (2000) OMG Unified Modeling Language Specification Archived 2004년 10월 18일 - 웨이백 머신, Version 1.3 First Edition: March 2000. Retrieved 12 August 2008

JSON

https://www.json.org/

Class Diagram

Scott W. Ambler (2009) UML 2 Class Diagrams . Webdoc 2003-2009. Accessed Dec 2, 2009

Sequence Diagram

UML Distilled by Martin Fowler

Use Case Diagram

Siau, K., Lee, L. (2004). "Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in UML", *Requirements Engineering*, 9(4), 229-237.

Draw.io

https://www.draw.io/

Android Studio

https://developer.android.com/studio/index.html

Firebase
https://firebase.google.com/?hl=ko