**Design Specification**

**Chwimi : 취미분석 및 배송 시스템**

소프트웨어공학개론 이은석 교수님

2019-05-19

**01분반 Team #2**

|  |  |
| --- | --- |
| 학번 | 이름 |
| 2013311806 | 김탁영 |
| 2016312523 | 김연경 |
| 2013314834 | 장재혁 |
| 2015310776 | 조윤아 |

목차

[**1. Preface** 9](#_Toc9014897)

[**1.1 Objective** 9](#_Toc9014898)

[**1.2 Readership** 9](#_Toc9014899)

[**1.3 Document structure** 9](#_Toc9014900)

[A. Preface 9](#_Toc9014901)

[B. Introduction 9](#_Toc9014902)

[C. System Architecture 9](#_Toc9014903)

[D. User Management System 10](#_Toc9014904)

[E. About System 10](#_Toc9014905)

[**F. Notice System** 10](#_Toc9014906)

[G. Hobby Test System 10](#_Toc9014907)

[H. Subscribe Management System 10](#_Toc9014908)

[I. Review Management System 10](#_Toc9014909)

[J. Question Management System 11](#_Toc9014910)

[K. Protocol Design 11](#_Toc9014911)

[L. Database Design 11](#_Toc9014912)

[M. Testing Plan 11](#_Toc9014913)

[N. Development Environment 11](#_Toc9014914)

[O. Develop Plan 12](#_Toc9014915)

[P. Index 12](#_Toc9014916)

[Q. Reference 12](#_Toc9014917)

[**1.4 Version of the document** 12](#_Toc9014918)

[A. Version Format 12](#_Toc9014919)

[B. Version Management Policy 12](#_Toc9014920)

[C. Version Update History 13](#_Toc9014921)

[**2. Introduction** 14](#_Toc9014922)

[**2.1 Objective** 14](#_Toc9014923)

[**2.2 Applied Diagram** 14](#_Toc9014924)

[A. UML 14](#_Toc9014925)

[B. Package Diagram 15](#_Toc9014926)

[C. Class Diagram 16](#_Toc9014927)

[D. Deployment Diagram 17](#_Toc9014928)

[E. ER Diagram 18](#_Toc9014929)

[F. State Diagram 19](#_Toc9014930)

[G. Sequence Diagram 20](#_Toc9014931)

[**2.3 Applied Tools** 20](#_Toc9014932)

[A. Draw.io 20](#_Toc9014933)

[B. Django 21](#_Toc9014934)

[C. Recombee 21](#_Toc9014935)

[**2.4 Project Scope** 21](#_Toc9014936)

[A. User Management System 22](#_Toc9014937)

[B. About System 22](#_Toc9014938)

[C. Notice System 23](#_Toc9014939)

[D. Hobby Test System 23](#_Toc9014940)

[E. Subscribe Management System 24](#_Toc9014941)

[F. Review Management System 24](#_Toc9014942)

[G. Q&A Management System 25](#_Toc9014943)

[**3. System Architecture** 26](#_Toc9014944)

[**3.1 Objective** 26](#_Toc9014945)

[**3.2 System Organization** 26](#_Toc9014946)

[A. User Management System 26](#_Toc9014947)

[B. About System 27](#_Toc9014948)

[C. Notice System 27](#_Toc9014949)

[D. Hobby Test System 28](#_Toc9014950)

[E. Subscribe Management System 28](#_Toc9014951)

[F. Review Management System 29](#_Toc9014952)

[G. Question Management System 30](#_Toc9014953)

[**3.3 Package Diagram** 31](#_Toc9014954)

[**3.4 Deployment Diagram** 33](#_Toc9014955)

[**4. User Management System** 34](#_Toc9014957)

[**4.1 Objective** 34](#_Toc9014958)

[**4.2 Class Diagram** 34](#_Toc9014959)

[**4.3 Sequence Diagram** 37](#_Toc9014960)

[A. Login (not using kakao / having used kakao login before) 37](#_Toc9014961)

[B. login (using kakao) 38](#_Toc9014962)

[**C. Mypage** 39](#_Toc9014963)

[**4.4 State Diagram** 39](#_Toc9014964)

[A. Signup 39](#_Toc9014965)

[B. Login 40](#_Toc9014966)

[C. Mypage 40](#_Toc9014967)

[**5. About System** 41](#_Toc9014968)

[**5.1 Objective** 41](#_Toc9014969)

[**5.2 Class Diagram** 41](#_Toc9014970)

[**5.3 Sequence Diagram** 42](#_Toc9014971)

[**5.4 State Diagram** 42](#_Toc9014972)

[**6. Notice System** 43](#_Toc9014973)

[**6.1 Objective** 43](#_Toc9014974)

[**6.2 Class Diagram** 43](#_Toc9014975)

[A. Notice System (Admin use) 44](#_Toc9014976)

[B. Notice System (User use) Sequence Diagram 45](#_Toc9014977)

[A. Notice System (Admin use) 45](#_Toc9014978)

[B. Notice System (User use) 45](#_Toc9014979)

[**7. Hobby Test System** 46](#_Toc9014980)

[**7.1 Objective** 46](#_Toc9014981)

[**7.2 Class Diagram** 46](#_Toc9014982)

[**7.3 Sequence Diagram** 48](#_Toc9014983)

[**7.4 State Diagram** 49](#_Toc9014984)

[**8. Subscribe Management System** 50](#_Toc9014985)

[**8.1 Objective** 50](#_Toc9014986)

[**8.2 Class Diagram** 50](#_Toc9014987)

[A. Subscribe System 50](#_Toc9014988)

[B. Video System 51](#_Toc9014989)

[**8.3 Sequence Diagram** 52](#_Toc9014990)

[A. Subscribe System 52](#_Toc9014991)

[B. Video System 53](#_Toc9014992)

[**8.4 State Diagram** 54](#_Toc9014993)

[A. Subscribe System 54](#_Toc9014994)

[B. Video System 55](#_Toc9014995)

[**9. Review Management System** 56](#_Toc9014996)

[**9.1 Objective** 56](#_Toc9014997)

[**9.2 Class Diagram** 56](#_Toc9014998)

[**9.3 Sequence Diagram** 60](#_Toc9014999)

[A. View 60](#_Toc9015000)

[B. Write 61](#_Toc9015001)

[C. Update 62](#_Toc9015002)

[D. Comment 63](#_Toc9015003)

[**9.4 State Diagram** 64](#_Toc9015004)

[A. View 64](#_Toc9015005)

[B. Write 65](#_Toc9015006)

[C. Update 65](#_Toc9015007)

[D. Comment 66](#_Toc9015008)

[**10. Question Management System** 67](#_Toc9015009)

[**10.1 Objective** 67](#_Toc9015010)

[**10.2 Class Diagram** 67](#_Toc9015011)

[**10.3 Sequence Diagram** 70](#_Toc9015012)

[A. View 70](#_Toc9015013)

[B. Write 71](#_Toc9015014)

[C. Update 72](#_Toc9015015)

[D. Comment 73](#_Toc9015016)

[**10.4 State Diagram** 74](#_Toc9015017)

[A. View 74](#_Toc9015018)

[B. Write 75](#_Toc9015019)

[C. Update 76](#_Toc9015020)

[D. Comment 77](#_Toc9015021)

[**11. Protocol Design** 78](#_Toc9015022)

[**11.1 Objective** 78](#_Toc9015023)

[**11.2 JSON** 78](#_Toc9015024)

[**11.3 Protocol Design** 78](#_Toc9015025)

[A. Overview 78](#_Toc9015026)

[B. Sign up Protocol 79](#_Toc9015027)

[C. Log in Protocol 79](#_Toc9015028)

[D. My page Protocol 80](#_Toc9015029)

[E. About Protocol 80](#_Toc9015030)

[F. Notice Protocol 81](#_Toc9015031)

[G. Hobby Test Protocol 82](#_Toc9015032)

[H. Subscribe Protocol 82](#_Toc9015033)

[I. Video Protocol 83](#_Toc9015034)

[J. Review Protocol 84](#_Toc9015035)

[K. Question Protocol 86](#_Toc9015036)

[L. Comment Protocol 88](#_Toc9015037)

[**12. Database Design** 89](#_Toc9015038)

[**12.1 Objective** 89](#_Toc9015039)

[**12.2 ER Diagram** 89](#_Toc9015040)

[A. Entity 89](#_Toc9015041)

[B. Relationship 90](#_Toc9015042)

[**12.3 Relational Schema** 93](#_Toc9015043)

[**12.4 SQL DDL** 98](#_Toc9015044)

[**13. Testing Plan** 107](#_Toc9015045)

[**13.1 Objective** 107](#_Toc9015046)

[**13.2 Testing Policy** 107](#_Toc9015047)

[A. Component Testing 107](#_Toc9015048)

[B. System Testing 107](#_Toc9015049)

[C. Acceptance Testing 107](#_Toc9015050)

[**13.3 Test Case** 107](#_Toc9015051)

[**14. Development Environment** 111](#_Toc9015052)

[**14.1 Objective** 111](#_Toc9015053)

[**14.2 Programming Languages & IDE** 111](#_Toc9015054)

[**14.3 Coding Rule** 112](#_Toc9015055)

[**14.4 Version Management Tool** 112](#_Toc9015056)

[**15. Develop Plan** 113](#_Toc9015057)

[**15.1 Objective** 113](#_Toc9015058)

[**15.2 Gantt Chart** 113](#_Toc9015059)

[**16. Index** 114](#_Toc9015060)

[Figure 114](#_Toc9015061)

[Table 117](#_Toc9015062)

# **1. Preface**

## **1.1 Objective**

Preface에서는 본 문서의 예상되는 독자와 문서의 전반적인 구조, 그리고 각 부분의 역할에 대하여 제시한다. 그리고 버전 관리 정책, 버전 변경 기록, 그리고 문서의 변경사항들과 그에 대한 근거를 서술한다.

## **1.2 Readership**

본 문서의 독자는 시스템을 개발하는 소프트웨어 엔지니어, 시스템을 설계하는 Architecture, 개발에 참여하는 클라이언트 엔지니어, 고객 기술 지원을 위한 서비스 팀 등 본 문서에서 소개할 시스템과 관련된 모든 구성원으로 정의된다. 시스템을 개발 과정에서 외주 업체를 이용한다면, 해당 업체에서 개발에 관련된 모든 구성원 역시 독자에 포함된다.

## **1.3 Document structure**

### A. Preface

Preface에서는 본 문서의 예상되는 독자와 문서의 전반적인 구조, 그리고 각 부분의 역할에 대하여 제시한다. 그리고 버전 관리 정책, 버전 변경 기록, 그리고 문서의 변경사항들과 그에 대한 근거를 서술한다.

### B. Introduction

Introduction에서는 본 문서에서 시스템을 설계하고 설명하기 위해 사용된 모든 종류의 다이어그램, 도구(Tool)에 대해 서술한다.

### C. System Architecture

System Architecture에서는 팀에서 개발하고자 하는 시스템에 대하여 전반적으로 설명한다. Chwimi 시스템의 전체적인 구조를 설명하고, 시스템을 Block Diagram으로 나타내어 각각의 관계와 실제 사용 과정을 Package Diagram과 Deployment Diagram을 이용하여 설명한다. 각 시스템에 대한 자세한 설명 및 묘사는 4 ~ 14 장에 걸쳐서 기술된다.

### D. User Management System

사용자가 Chwimi 시스템을 이용하기 위해 Chwimi 시스템에서 제공하는 회원가입과 로그인 혹은 KAKAO 계정 연동을 통한 회원가입과 로그인을 하고 계정 정보, 취미 테스트 결과와 구독 내역의 사용자 정보를 제공하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 User Management의 구조를 표현하고 설명한다.

### E. About System

탄생 배경과 서비스 이용 가이드 그리고 어떤 종류의 취미 상품을 제공하는 지 등 Chwimi가 제공하는 서비스에 대해 소개하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 About System의 구조를 표현하고 설명한다.

### **F. Notice System**

관리자가 공지사항 제목 및 내용을 작성하여 사용자에게 제공하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 Notice System의 구조를 표현하고 설명한다.

### G. Hobby Test System

사용자 기본 정보(연령, 성별 등)와 테스트 과정에서 얻은 질문에 대한 사용자의 답변을 기반으로 취향을 분석하여 취미 상품을 추천하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 Hobby Test System의 구조를 표현하고 설명한다.

### H. Subscribe Management System

사용자가 취미 분석 결과를 바탕으로 취미 상품을 일정 기간 동안 배송 받을 것을 동의하여 결제하고 사용자의 이해를 돕기 위해 해당 취미 상품 이용 방법을 메인 페이지에 영상으로 제공하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 Subscribe System의 구조를 표현하고 설명한다.

### I. Review Management System

Chwimi 시스템에서 제공하는 취미 상품을 경험한 사용자가 해당 상품에 대한 후기를 작성하거나 다른 사용자가 작성한 후기를 읽고 검색할 수 있고, 관리자가 이에 댓글을 남기는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 Review System의 구조를 표현하고 설명한다.

### J. Question Management System

사용자가 궁금한 점을 질문하거나 다른 사용자가 작성한 질문과 그에 대한 답변을 읽고 검색할 수 있고, 관리자가 이에 댓글을 남기는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 Q&A System의 구조를 표현하고 설명한다.

### K. Protocol Design

Protocol Design에서는 Subsystem들이 상호작용하는 프로토콜에 대해 서술한다. 기본적으로 JSON이라는 Javascript Object Nation의 DATA 교환 형식을 이용하여 프로토콜에서 통신하는 메시지의 형식, 용도와 의미를 설명한다.

### L. Database Design

Database Design은 요구사항 명세서에서 기술한 데이터베이스 요구사항을 바탕으로 수정사항을 수정하여 다시 요구사항을 작성하였다. 요구사항을 바탕으로 ER Diagram을 작성하고, 이를 이용하여 Relational Schema를 작성하고, Normalization을 통해 Redundancy와 Anomaly를 제거한 후 마지막으로 SQL DDL을 작성한다.

### M. Testing Plan

Testing Plan은 시스템이 의도한 방향으로 실행되는지 확인하고 시스템 내부의 결함을 찾기 위해 testing을 설계하여 시행한다. 본 항목에서는 Test Policy 뿐만 아니라 여러 Test Case에 대하여 기술한다.

### N. Development Environment

Development Environment에서는 개발자의 환경에 대해 설명한다. 사용한 프로그래밍 언어와 IDE, 패키지에 대해 서술한다.

### O. Develop Plan

Develop Plan에서는 개발 계획에 대해 서술한다. 이때 Gantt chart를 이용하여 개발 계획과 실제 개발 흐름에 대해 명시한다.

### P. Index

Index에서는 본 문서의 도표, 다이어그램 및 사진의 인덱스를 표기한다. 이를 통해 원하는 정보가 문서의 몇 페이지에 있는지 알 수 있다.

### Q. Reference

Reference에서는 Design Specification을 작성하면서 참고한 자료 등을 명시한다.

## **1.4 Version of the document**

### A. Version Format

버전 번호는 Major number와 Minor number로 이루어져 있으며 (Major number).(Minor number)의 형식으로 표현한다. 문서의 버전은 1.0부터 시작한다.

### B. Version Management Policy

Design Specification을 수정할 때마다 버전을 업데이트한다. 다만 변경 간의 간격이 1시간 이내일 때에는 버전 번호를 업데이트 하지 않고 하나의 업데이트로 간주한다. 이미 완성된 파트를 변경할 때에는 Minor number를 변경하며, 새로운 부분을 추가하거나 문서의 구성이 예전에 비해 괄목할 변화가 있을 경우 Major number를 변경한다.

### C. Version Update History

|  |  |  |
| --- | --- | --- |
| Version | Modified Date | Explanation |
| 1.0 | 2019.05.09 | Preface, Introduction 초안 작성 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table . Version of the Document

# **2. Introduction**

## **2.1 Objective**

Introduction에서는 본 문서에서 시스템을 설계하고 설명하기 위해 사용된 모든 종류의 다이어그램, 도구(Tool)에 대해 서술한다.

## **2.2 Applied Diagram**

### A. UML

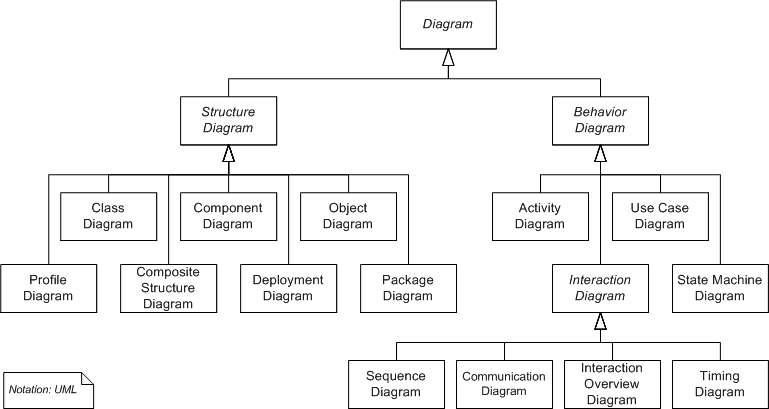


Figure . UML Diagram Hierarchy

UML(Unified Modeling Language)은 시스템 개발과 소프트웨어 설계에 대한 접근 방식을 표준화 하기 위해 개발된 통합 모델링 언어이다. 1997년에 OMG(Object Management Group)에 의해 표준으로 채택되어 관리되고 있으며, 2005년에는 ISO(International Organization for Standardization)에 의해 국제 표준으로 선정되었다.

UML은 객체지향 소프트웨어 집약 시스템을 개발할 때 시스템의 설명을 명세화, 시각화, 문서화하고자 할 때 사용된다. 이는 개발할 시스템의 유형, 개발 방법론, 개발에 사용되는 모든 프로그래밍 언어와 도구에 관계없이 적용될 수 있는 일반적인 표현 방법이기 때문에 본 문서에서도 UML을 사용하여 시스템을 설계한다.

UML에는 다양한 종류의 다이어그램이 존재하며 각 다이어그램은 완전히 별개의 것은 아니지만 기본적으로 목적이나 표현 방식의 차이가 존재한다.

### B. Package Diagram

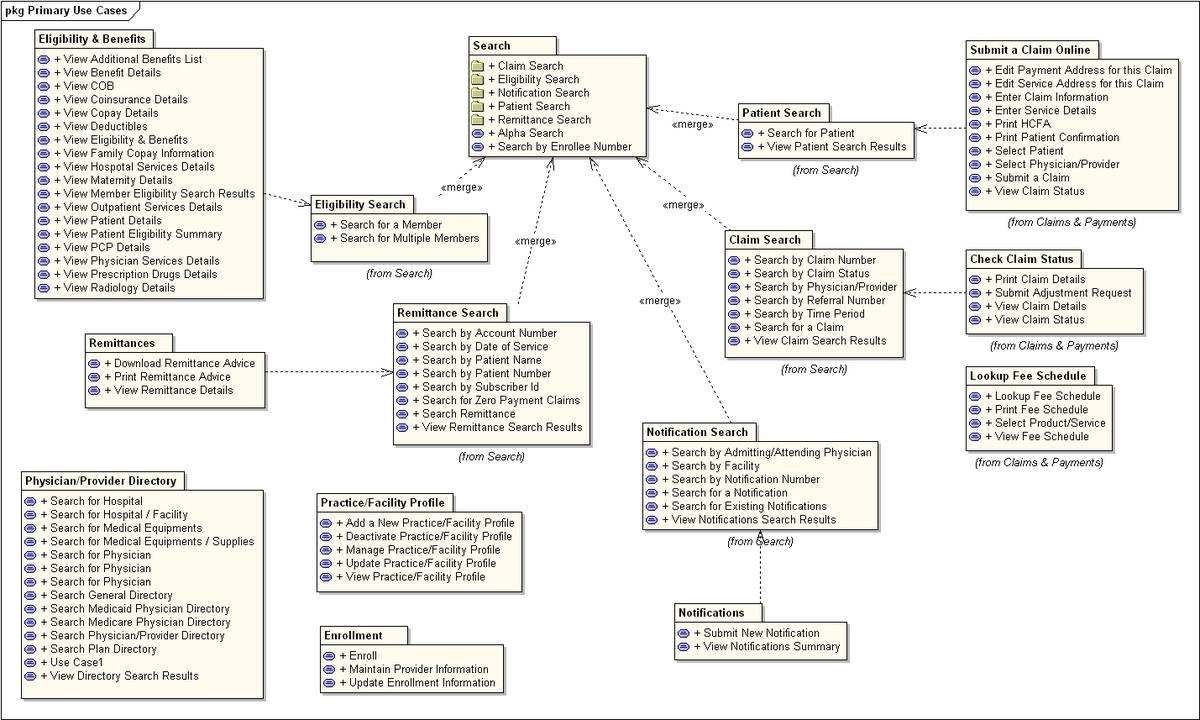


Figure . Example of Package Diagram

Package Diagram은 UML의 일종으로서 시스템 모델을 구성하는 패키지 간의 관계를 보여주는 다이어그램이다. 여기서 패키지는 추상적인 개념들을 관련 있는 요소로 그룹화 하는 것으로 클래스와 같은 여러 모델 요소들을 패키지로 그룹화 할 수 있으며, 또한 패키지는 다른 패키지의 요소로서 포함될 수 있다. 이때 패키지를 구성하고 구분하는 기준은 여러 사람이 동의할 수 있는 형태로 구성되어야 하고 패키지의 구성, 이름과 체계는 개발자들이 쉽게 이해하고 사용할 수 있도록 해야 한다.

패키지 내부의 모든 클래스들은 개념적, 기능적, 관리적 측면에서 유사한 성질을 가져야 하기 때문에 내부의 요소들은 밀접한 관련성(높은 응집도)을 가지고 다른 패키지의 요소들과는 약한 의존관계(강한 독립성)을 갖는다. 이때 패키지 간의 의존성은 labels/stereotypes로 묘사되어 계층 간의 통신 메커니즘을 나타낼 수 있다.

Package Diagram은 위에서 제시한 방법으로 시스템의 모델 요소들을 그룹화하고 시각적으로 제공함으로써 복잡한 시스템 구조 이해를 용이하게 한다.

### C. Class Diagram

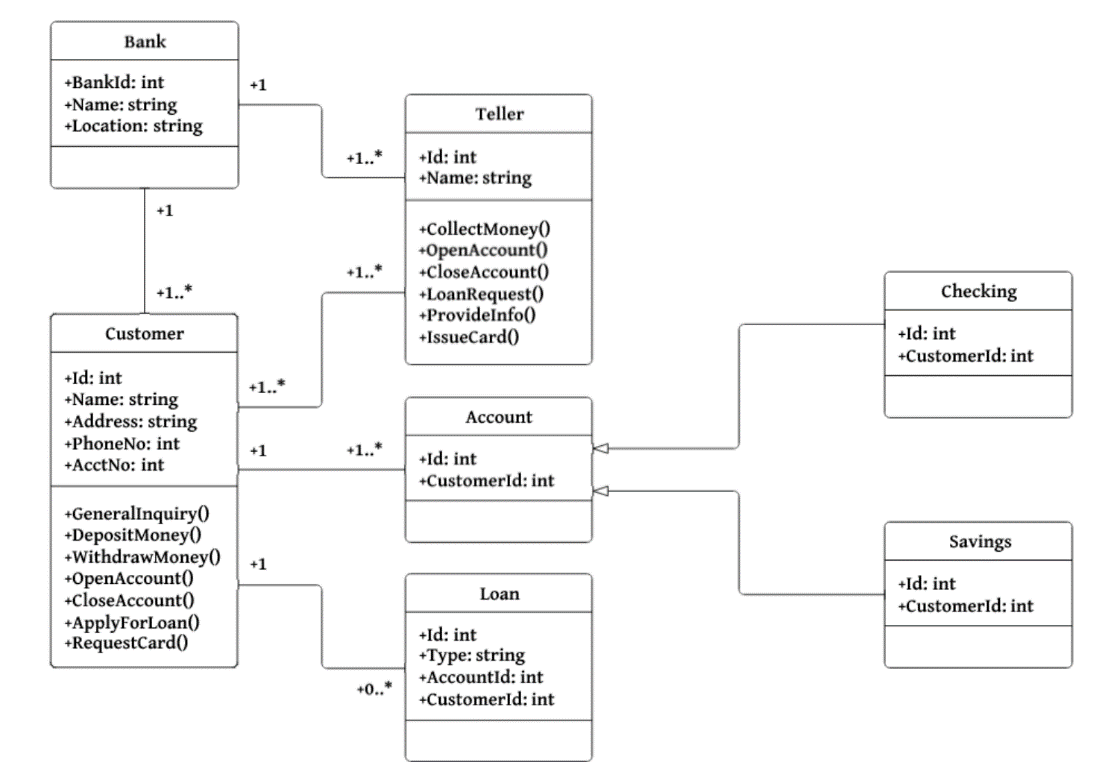


Figure . Example of Class Diagram

Class Diagram은 UML의 한 종류이자 객체지향 모델링의 기본 모델로, 시스템의 클래스 내부의 정적인 내용이나 클래스 간의 관계를 표기하는 다이어그램이다. 여기서 관계는 클래스 및 객체 다이어그램에서 발견되는 특정 유형의 논리적 연결을 의미한다. 클래스는 주요 요소들, 시스템 내의 상호작용, 시스템에서 구현해야 할 클래스들을 모두 나타내는데 각각의 클래스는 이름, 속성(Field)과 실행할 수 있는 작업(또는 메소드)의 세 개의 구획을 포함한다.

Class Diagram은 시스템 설계시에 여러 클래스들이 식별되고 그룹화 되기 때문에 이 사이의 정적 관계를 결정하거나 상속 관계 등을 명확히 표현할 수 있게 한다. 이를 통해 일부 또는 전체적인 시스템의 구조를 나타낼 수 있다.

### D. Deployment Diagram

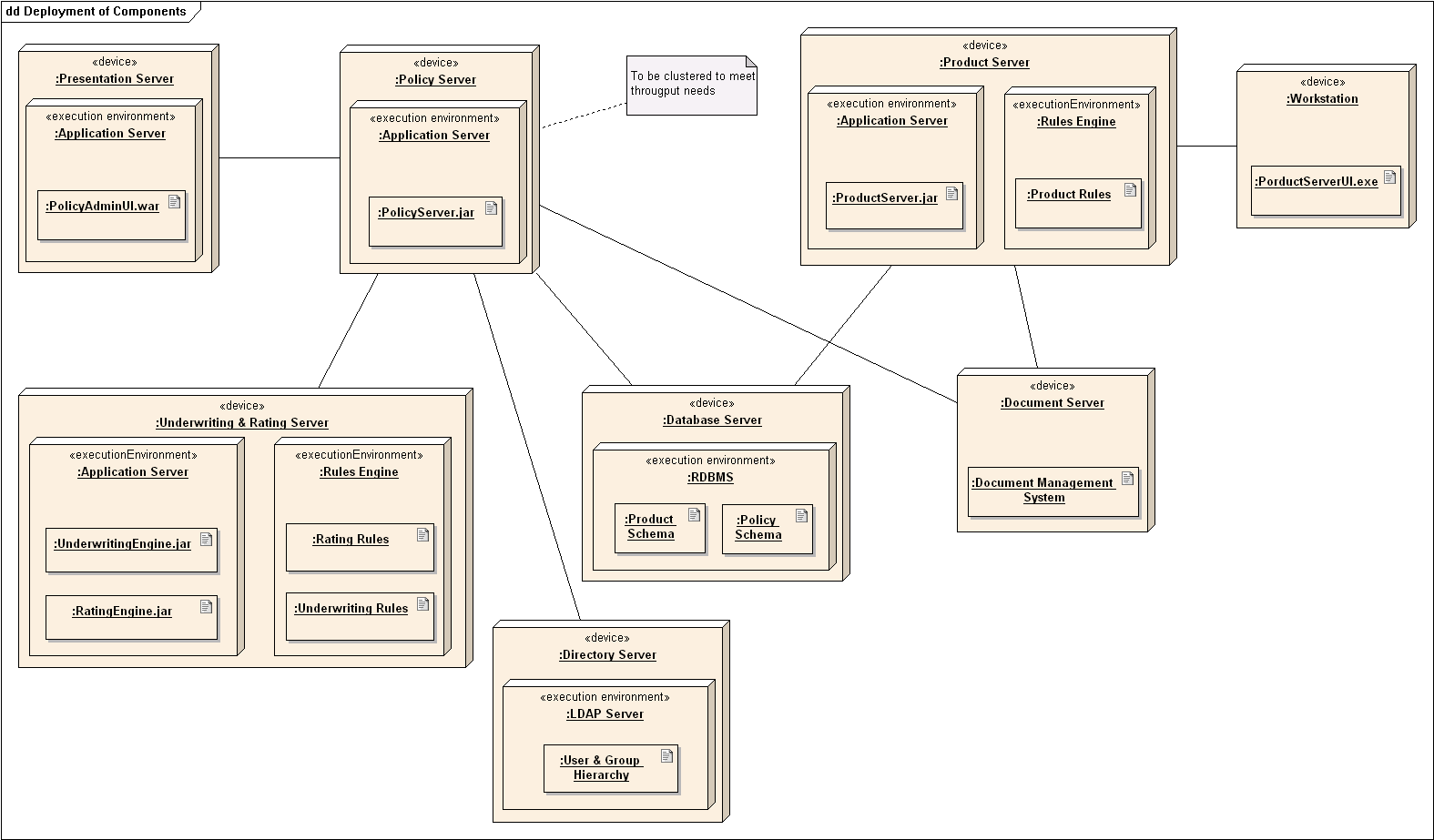


Figure . Example of Deployment Diagram

Deployment Diagram은 UML 중에서 시스템 자체의 측면을 설명하기 위해 물리적인 시스템 architecture를 모델링한 것으로 시스템의 소프트웨어와 하드웨어 컴포넌트 간의 관계 및 처리의 물리적 분배를 표현한다.

Deployment Diagram은 여러가지 UML 모양으로 구성된다. 노드라고 하는 3차원 상자는 시스템의 기본 소프트웨어 또는 하드웨어 요소 또는 다른 노드들의 집합을 나타낸다. 노드에서 노드로 가는 선은 관계를 나타내고, 상자 안에 포함된 더 작은 모양은 배포된 소프트웨어 architecture를 나타낸다.

Deployment Diagram은 소프트웨어 시스템이 배치, 실행될 하드웨어의 자원들을 정의하고 소프트웨어 구성요소가 어떤 하드웨어 자원에 탑재되어 실행될지를 정의한다.

### E. ER Diagram

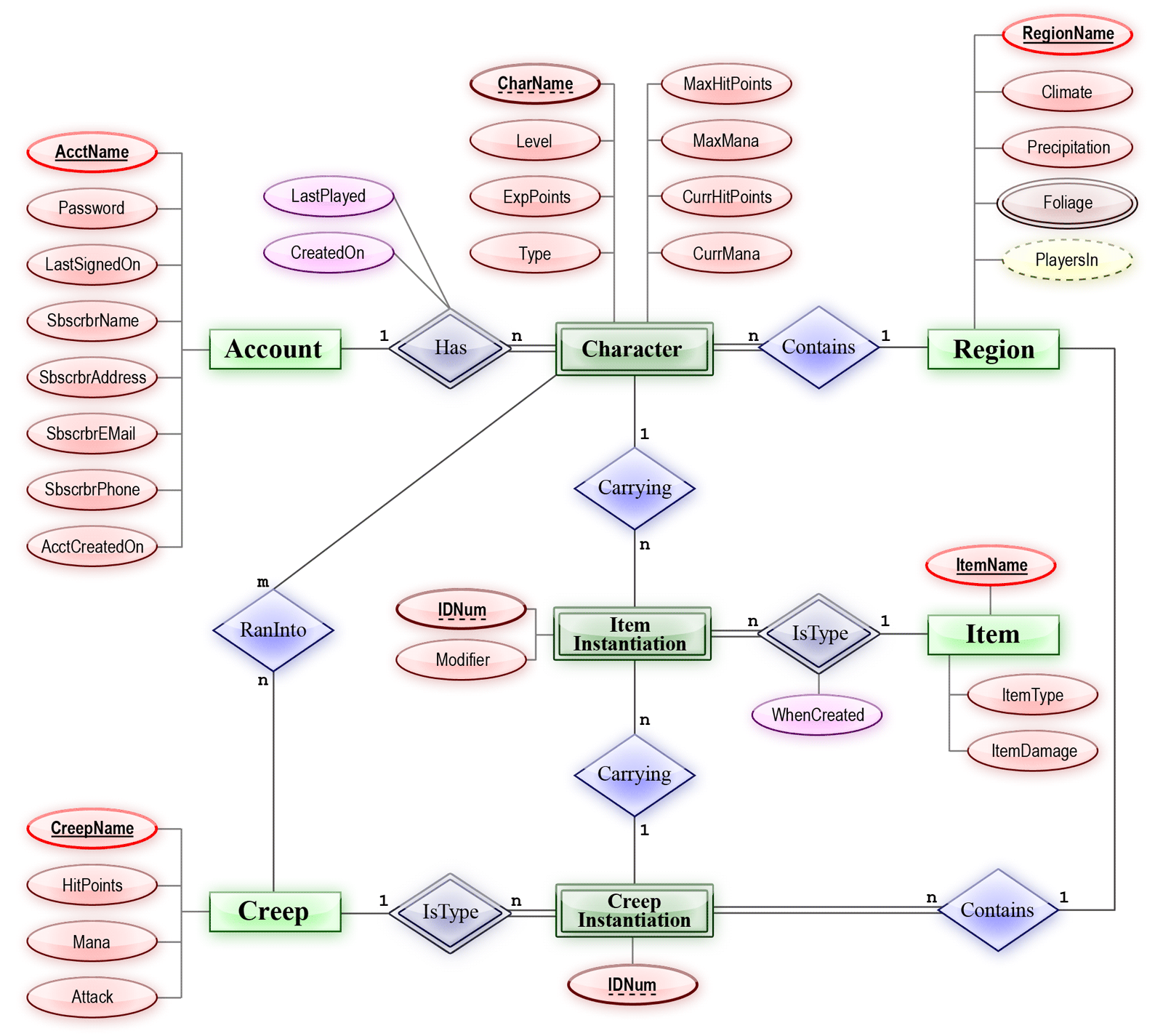


Figure . Example of ER Diagram

ER (Entity-Relationship) Diagram은 데이터베이스의 구조화된 데이터에 대한 일련의 표현으로 UML에 포함되는 다이어그램은 아니다. 데이터의 구조 및 그에 수반한 제약 조건들을 설계하는 방법 중 개체-관계 모델링이 존재하는데 이를 가시적으로 나타낸 것이다.

ER Diagram은 Entity(개체)와 Entity의 속성을 정의하고 Entity 간의 관계를 표시함으로써 데이터베이스의 논리적 구조를 보여주는 것이 가능하다. 여기서 Entity는 현실세계에서 객체로서 유무형의 정보대상으로 존재하면서 하나 이상의 속성으로 구성되어 구별될 수 있는 것을 의미하며, 같은 속성을 같는 개체들의 집합을 개체타입이라고 한다. 관계는 여러 개체 간에 존재하는 연관성으로서 하나 이상의 속성을 가질 수 있고 같은 관계들의 집합을 관계타입이라고 한다.

ER Diagram은 일대일 관계, 다중 관계, 제약조건 등의 다양한 표현이 가능하고 데이터에 대해 단순하면서도 정확한 이해를 용이하게 한다. 이에 데이터베이스 디자인을 스케치하는데 많이 사용되며 시스템에서 사용되는 정보 자원들의 저장 및 전달을 하는데 있어서 필수적이며 유용한 도구이다.

### F. State Diagram

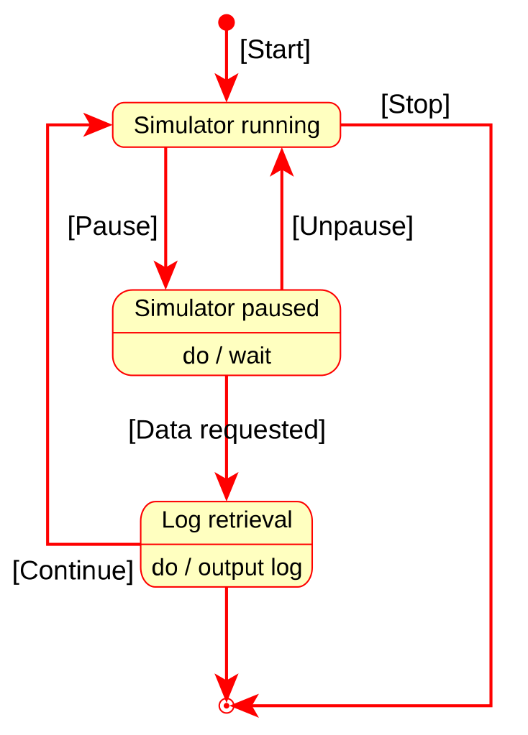


Figure . Example of State Diagram

State Diagram은 전체 시스템 혹은 한 객체에 대해서 이벤트가 발생할 때 객체의 가능한 모든 상태를 고려한 시스템의 동작을 설명하는데 사용되는 다이어그램이다. 이 동작은 하나 이상의 가능한 상태에서 발생하는 일련의 이벤트로 표시되고 분석된다. 각 다이어그램은 객체를 나타내며 시스템 전체에서 이러한 객체의 다양한 상태를 추적한다.

State Diagram은 상태의 변화를 일으키는 이벤트를 나타내거나 시스템의 동적 행동을 모델링하거나, 내부 또는 외부의 자극에 대한 클래스의 반응을 설명하는데 유용하다.

### G. Sequence Diagram

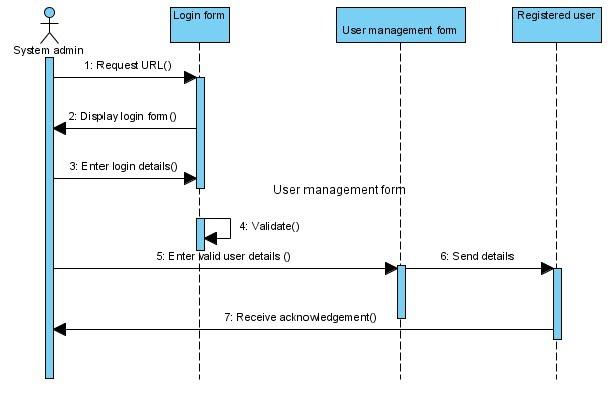


Figure . Example of Sequence Diagram

Sequence Diagram은 UML의 객체 그룹이 함께 작동하는 방법과 순서를 설명하는 interaction diagram의 일종이다. Sequence Diagram은 시간 순서대로 정렬된 객체 상호작용을 보여주는데 특히 시나리오에 포함된 객체 및 클래스와 시나리오 기능을 수행하는 데 필요한 객체 간에 교환되는 메시지 시퀀스를 묘사한다. 일반적으로 개발 중인 시스템의 논리적 뷰에서 Use Case 실현과 관련되어 많이 사용되고, Event Diagram이라고 표현되기도 하는 동적인 측면의 시스템 모델링 과정이다.

Sequence Diagram은 시스템의 활성 객체 간의 고급 상호 작용 모델링, Use case를 실현하는 협업 내의 객체 인스턴스 간의 상호작용 모델링, 작업을 실현하는 협업 내의 객체 간의 상호작용 모델링 등에 유용하다.

## **2.3 Applied Tools**

### A. Draw.io



Figure . Logo of Draw.io

Draw.io는 다이어그램을 작성 및 공유할 수 있는 개방형 플랫폼으로 웹 및 클라우드 기반 다이어그램 소프트웨어로, 이를 통해 흐름도, 프로세스 다이어그램, 조직도, UML 등을 보다 쉽게 작성할 수 있다. 구글 드라이브와 연동이 가능하기 때문에 팀 프로젝트에서 작업 상황 공유가 가능하다. 또한 작성된 다이어그램은 로컬파일로도 저장이 가능하다. 본 문서에 첨부되어 있는 시스템 설계도와 그 외 다이어그램들은 대부분 Draw.io를 이용하여 작성되었다.

### B. Django



Figure . Logo of Django

Django는 파이썬을 기반으로 하는 프레임워크를 제공하는 소프트웨어로이다. 웹 개발자들이 어려움을 겪는 부분을 고려하여 숙련된 개발자에 의해 만들어졌기 때문에 django를 통하여 쉽고 빠르게 좋은 디자인의 웹 페이지를 만들 수 있다. Chwimi는 django를 사용하여 안드로이드 웹앱을 개발할 것이다.

### C. Recombee



Figure . Logo of Recombee

Recombee는 딥 러닝 모델(Deep Learning Model)을 기반으로 하는 추천 모델을 제공하는 소프트웨어이다. 추천 모델로는 사용자 개별 맞춤 추천을 위해 협업 필터링 모델(Collaborative Filtering Model)과 내용 기반 필터링 모델(Content-based Filtering Model)을 합친 앙상블 모델 (Ensembles Model)을 제공한다. Chwimi는 Recombee의 추천 모델을 기반으로 사용자가 업로드한 게시물을 추가적으로 분석하여 추천 모델의 정확도를 높일 것이다.

## **2.4 Project Scope**

Chwimi는 사용자의 성향을 바탕으로 맞춤 취미 상품을 추천해주고 랜덤하게 배송해준다. 취미 상품을 구독한 사용자에게는 이해를 돕기 위해 해당 취미 상품 이용 방법을 영상으로 제공한다. 또한 사용자가 Chwimi 서비스나 취미 상품에 대해 궁금한 점과 경험한 상품 후기를 작성하거나 다른 사용자가 작성한 내용을 볼 수 있으며, 관리자는 이에 대한 댓글 작성을 통해 사용자와 커뮤니케이션 한다. Chwimi는 이러한 기능들을 기반으로 취미를 찾는데 어려움을 느끼는 시니어가 늘어난 여가 시간을 활용할 수 있도록 돕는 시스템이다.

Chwimi 는 크게 7개의 서브 시스템으로 구성되는데 이는 다음과 같다.

### User Management System

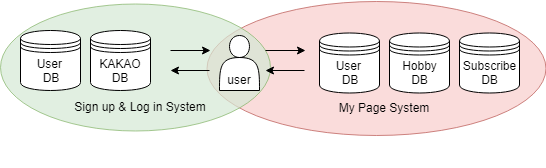


Figure . User Management System

User Management System은 사용자 정보를 받고 관리하기 위한 시스템이다. 이는 회원가입 기능을 제공하는 Sign up System, 로그인 기능을 제공하는 Log in System과 계정 정보, 취미 테스트 결과와 구독 내역의 사용자 정보를 제공하는 My page System의 3개의 하위 시스템으로 구성된다.

### About System

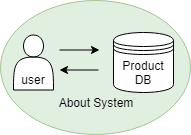


Figure . About System

About System은 Chwimi가 제공하는 서비스에 대해 소개하는 시스템이다. Chwimi의 탄생 배경과 서비스 이용 가이드 그리고 어떤 종류의 취미 상품을 제공하는지 간략하게 설명하며, 해당 정보는 상품 DB에 저장한다.

### Notice System

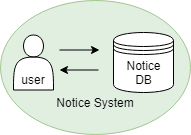


Figure . Notice System

Notice System은 관리자가 작성한 공지사항을 사용자가 볼 수 있도록 하는 시스템이다. 공지사항은 공지 DB에 저장해 두고 사용한다.

### D. Hobby Test System

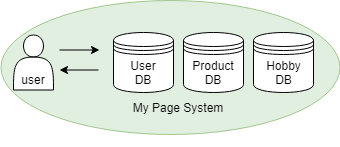


Figure . Hobby Test System

Hobby Test System은 사용자의 기본 정보와 테스트 과정에서 얻은 질문에 대한 사용자의 답변을 기반으로 취향을 분석하여 취미 상품을 추천하는 시스템이다. 취미 테스트 결과를 취미 DB에 저장해 두고 사용한다.

### E. Subscribe Management System

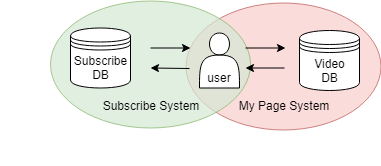


Figure . Subscribe Management System

Subscribe Management System은 구독 정보를 받고 관리하기 위한 시스템이다. 이는 취미 테스트를 마친 사용자가 취미 서비스를 구독하는 Subscribe System과 구독을 완료한 사용자에게 취미 상품 이용 방법을 영상으로 제공하는 Video System의 2개의 하위 시스템으로 구성된다.

### F. Review Management System

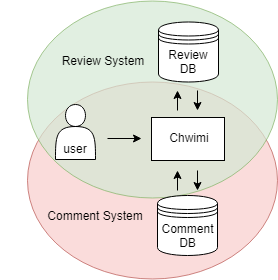


Figure . Review Management System

Review Management System은 사용자로부터 후기를 받고 관리하기 위한 시스템이다. 이는 취미 상품을 경험한 사용자가 해당 상품에 대한 후기를 작성하거나 다른 사용자가 작성한 후기를 읽고 검색할 수 있는 Review System과 관리자가 이에 댓글을 남기는 Comment System의 2개의 하위 시스템으로 구성된다.

### G. Q&A Management System

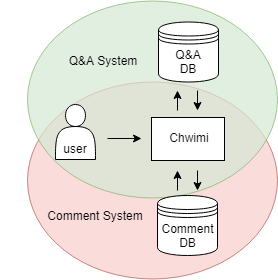


Figure . Q&A Management System

Q&A Management System은 사용자로부터 질문을 받고 관리하기 위한 시스템이다. 이는 사용자가 Chwimi 서비스를 사용함에 있어서 궁금한 점을 질문으로 작성하거나 다른 사용자가 작성한 질문 읽고 검색할 수 있는 Review System과 관리자가 이에 댓글을 남기는 Comment System의 2개의 하위 시스템으로 구성된다.

# **3. System Architecture**

## **3.1 Objective**

System Architecture에서는 팀에서 개발하고자 하는 시스템에 대하여 전반적으로 설명한다. Chwimi 시스템의 전체적인 구조를 설명하고, 시스템을 Block Diagram으로 나타내어 각각의 관계와 실제 사용 과정을 Package Diagram과 Deployment Diagram을 이용하여 설명한다.

## **3.2 System Organization**

### A. User Management System

사용자 관리 시스템은 사용자의 회원가입, 로그인, 마이페이지 등의 사용자의 시스템 사용을 관리하는 시스템이다.

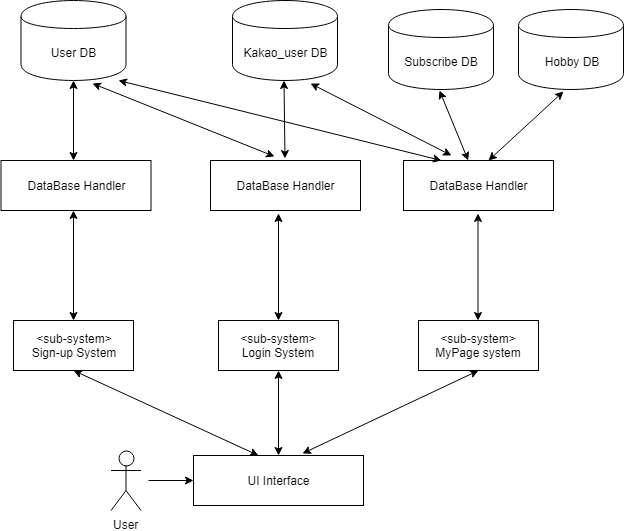


Figure . User Management System Architecture

### B. About System

About 시스템은 Chwimi 서비스의 정보, Chwimi 서비스를 이용하는 방법, Chwimi서비스가 제공하는 취미상품 등의 정보를 사용자에게 안내하는 시스템이다.

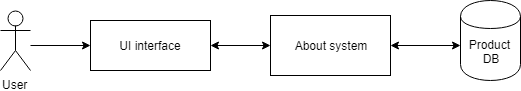


Figure . About System Architecture

### C. Notice System

공지사항 시스템은 관리자와 사용자 두 대상에 의해 사용되어진다. 관리자에게는 공지사항 페이지에 공지사항을 등록할 수 있도록 하며, 사용자에게는 등록된 공지사항을 읽을 수 있도록 해준다.

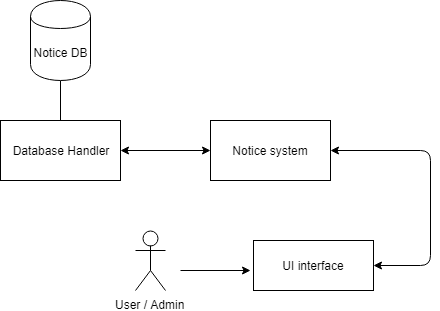


Figure . Notice System Architecture

### Hobby Test System

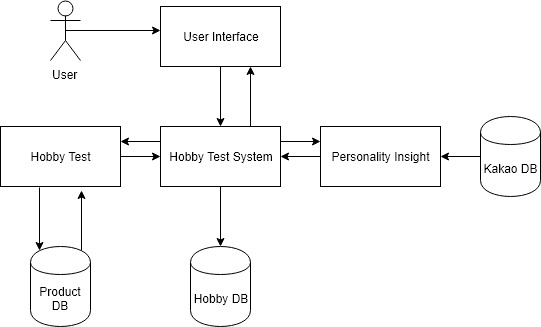


Figure . Hobby Test System Architecture

### Subscribe Management System

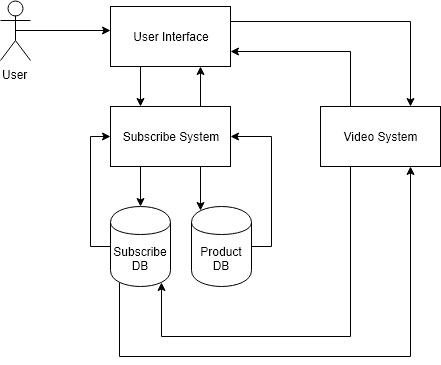


Figure . Subscribe Management System Architecture

### Review Management System

후기 관리 시스템은 사용자가 취미 상품을 배송받은 후 상품, 배송, 가격 면에서 평점과 후기를 남기는 시스템이다. 관리자는 댓글로 피드백을 남길 수 있다.

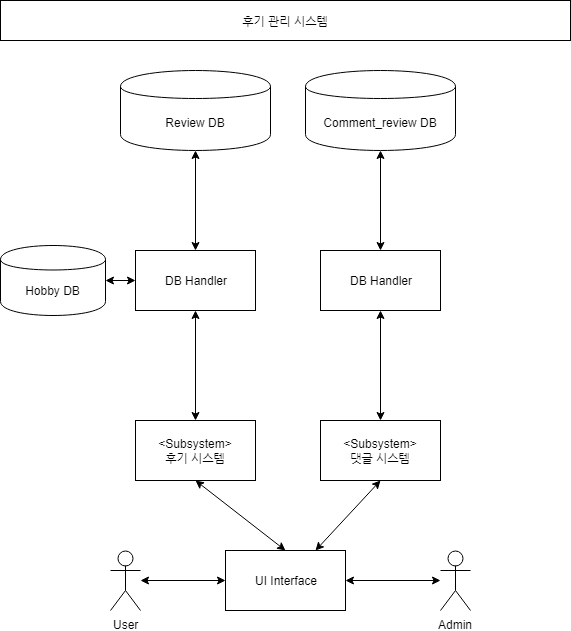


Figure . Review System Architecture

### Question Management System

질문 관리 시스템은 사용자가 게시판을 통해 서비스 이용과 관련된 문의사항을 업로드할 수 있고, 관리자가 이에 대한 답변을 남길 수 있는 시스템이다.

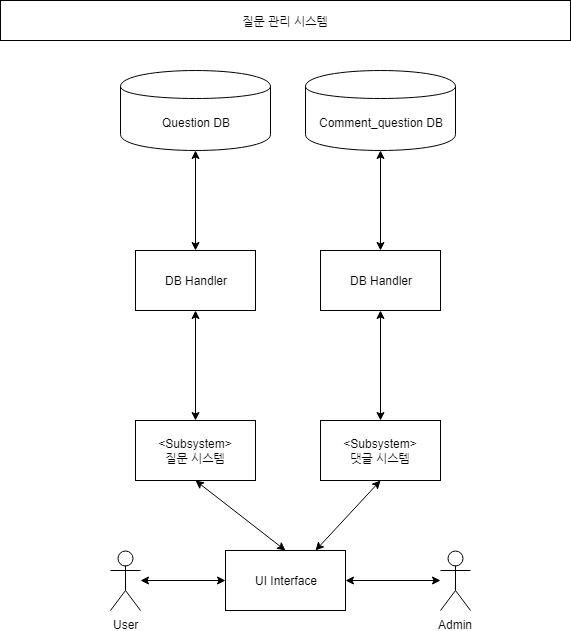


Figure . Question System Architecture

## **3.3 Package Diagram**

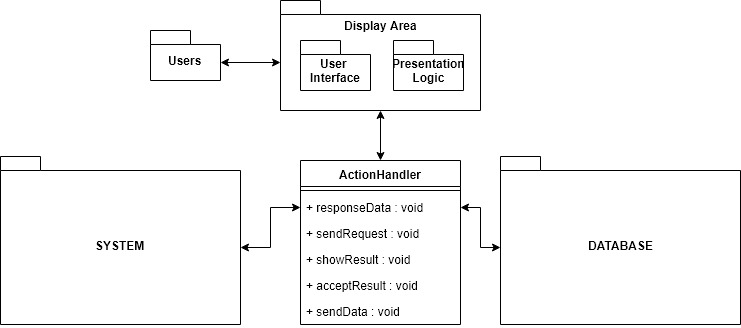


Figure . Overall Package Diagram

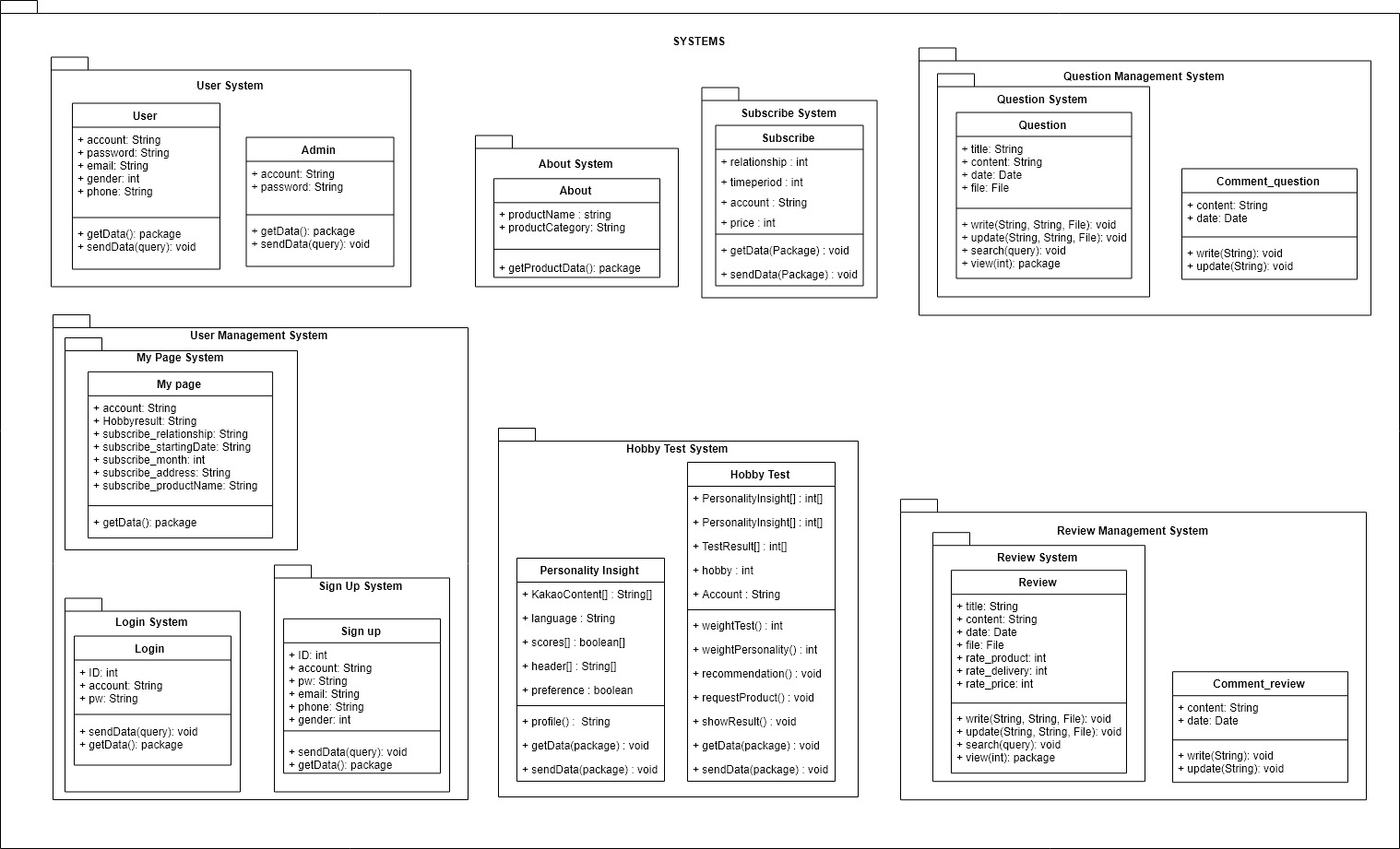


Figure . System Package Diagram

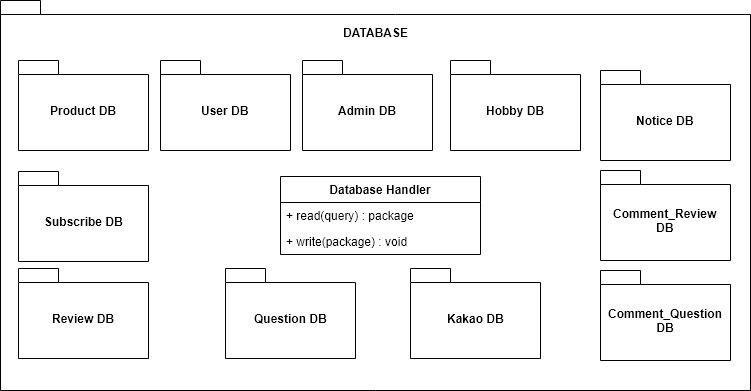


Figure . Database Package Diagram

## **3.4 Deployment Diagram**

## 

Figure . Deployment Diagram

# **4. User Management System**

## **4.1 Objective**

사용자가 Chwimi 시스템을 이용하기 위해 Chwimi 시스템에서 제공하는 회원가입과 로그인 혹은 KAKAO 계정 연동을 통한 회원가입과 로그인을 하고 계정 정보, 취미 테스트 결과와 구독 내역의 사용자 정보를 제공하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 User Management의 구조를 표현하고 설명한다.

## **4.2 Class Diagram**

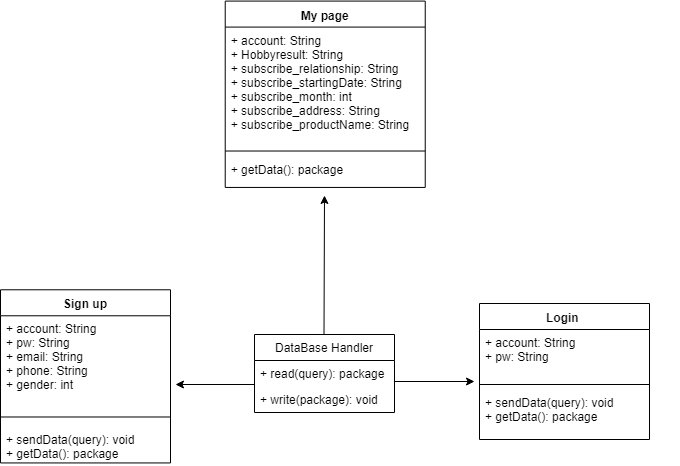


Figure . User Management System Class Diagram

A. DB Handler: 데이터베이스에 새로운 테이블을 만들거나 기존 테이블에 변경, 삭제 등의 수정을 가할 수 있도록 하는 클래스. 주로 원하는 데이터를 검색해 가져오거나 새로운 데이터를 테이블에 저장하는 것을 용이하게 만든다.

A.1 attributes

해당사항 없음

A.2 methods

+ package read(query): 해당 DB에서 원하는 데이터를 가져온다.

+ void write(package): 해당 DB에 새로운 데이터를 저장한다.

B. Sign up

B.1 attributes

+ account: 사용자가 설정한 계정

+ pw: 사용자가 설정한 비밀번호

+ email: 사용자의 이메일 주소

+ phone: 사용자의 핸드폰 번호

+ gender: 사용자의 성별

B.2 methods

+ void sendData(query) : user DB에 등록할 사용자의 데이터를 보낸다.

+ package getData(): 해당 사용자의 데이터를 user DB로부터 받아온다.

C. Login

C.1 attributes

+ account: 사용자가 설정한 계정

+ pw: 사용자가 설정한 비밀번호

C.2 methods

+ void sendData(query): userDB에 로그인할 사용자의 데이터를 보낸다.

+ package getData(): 해당 사용자의 데이터를 user DB로부터 받아온다.

D. MyPage

D.1 attributes:

+ account: 사용자가 설정한 계정

+ Hobbyresult: 해당 사용자의 취미 분석 테스트 결과

+ subscribe\_relationship: 구독자와 사용자간의 관계

+ subscribe\_startingDate: 구독시작 시기

+ subscribe\_month: 구독 서비스를 받는 기간

+ subscribe\_address: 구독 서비스가 제공되는 주소

+ subscribe\_productName: 제공되는 구독 서비스의 이름

D.2 methods

+ package getData(): 해당 사용자의 데이터를 subscribe DB, hobby DB, user DB에서 받아온다.

## **4.3 Sequence Diagram**

### A. Login (not using kakao / having used kakao login before)

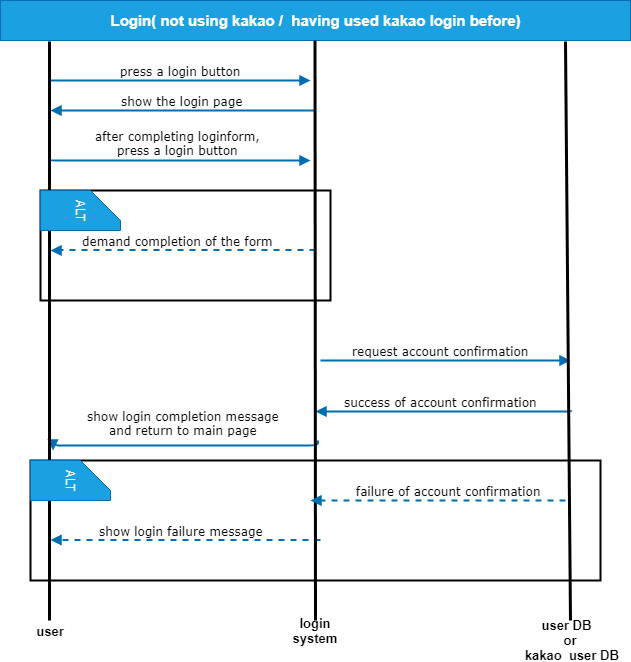


Figure . Login (not using Kakao) System Sequence Diagram

### B. login (using kakao)

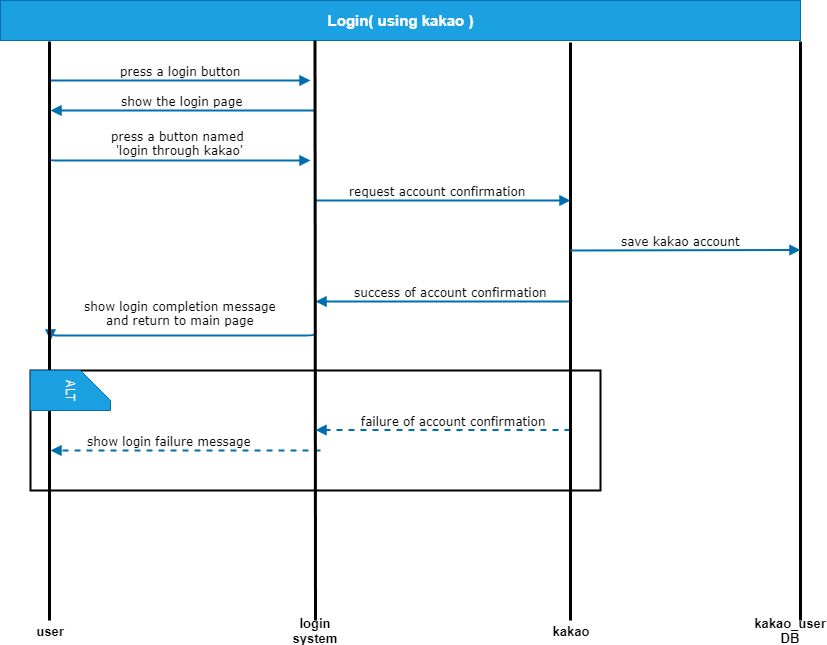


Figure . Login (Using Kakao) System Sequence Diagram

### **C. Mypage**

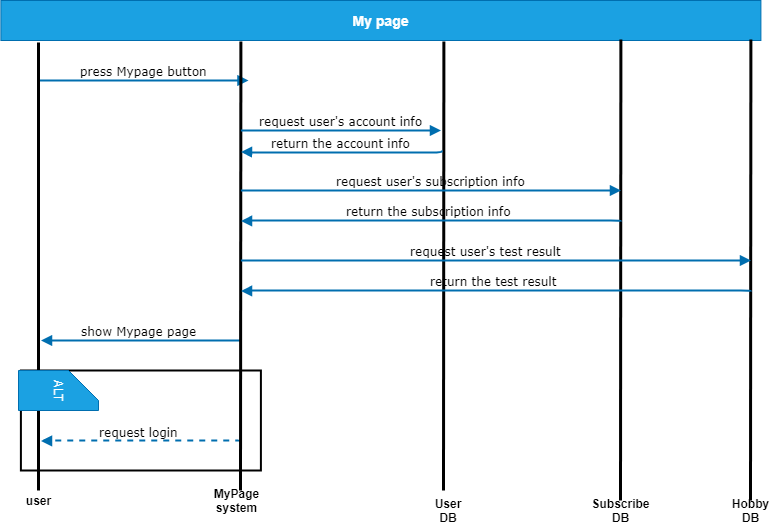
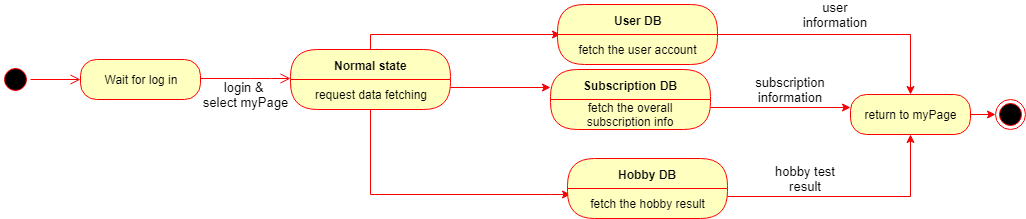


Figure . MyPage System Sequence System

## **4.4 State Diagram**

### A. Signup



*Figure 33. Signup State Diagram*

### B. Login

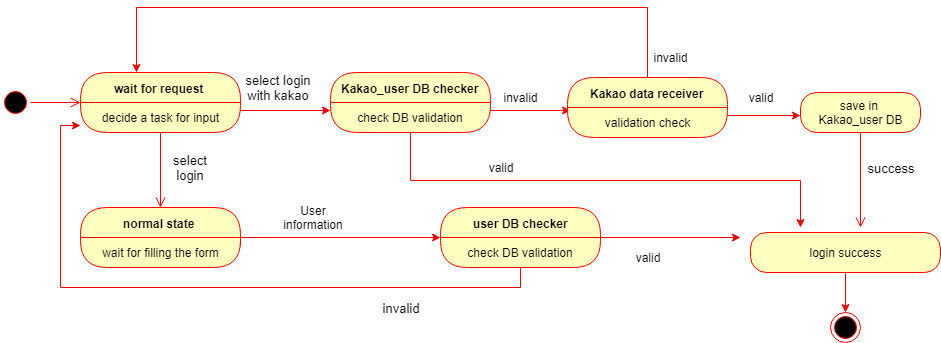


Figure . Login System Sequence Diagram

### C. Mypage

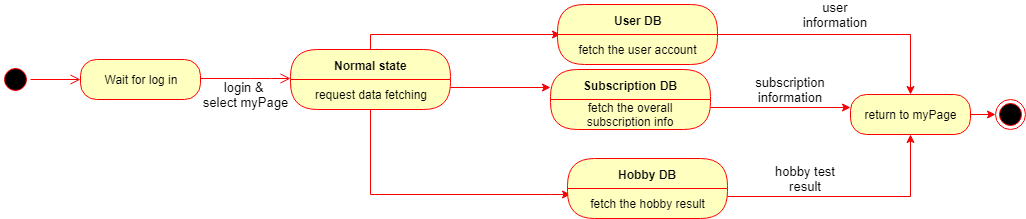


Figure . Mypage Sequence Diagram

# **5. About System**

## **5.1 Objective**

탄생 배경과 서비스 이용 가이드 그리고 어떤 종류의 취미 상품을 제공하는 지 등 Chwimi가 제공하는 서비스에 대해 소개하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 About System의 구조를 표현하고 설명한다.

## **5.2 Class Diagram**

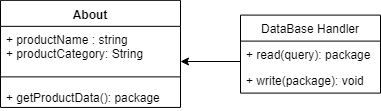


Figure . About System Class Diagram

A. DB handler

A.1 Attributes

해당사항 없음

A.2 Method

+ read(query): 해당 DB에서 원하는 데이터를 가져온다.

+ write(package): 해당 DB에 새로운 데이터를 저장한다.

B. About System

B.1 Attributes

+ productName : 서비스가 제공하는 취미상품명

+ productCategory: 서비스가 제공하는 취미상품이 속한 카테고리

B.2 method

+ getProductData(): Product DB에서 데이터를 받는다.

## **5.3 Sequence Diagram**

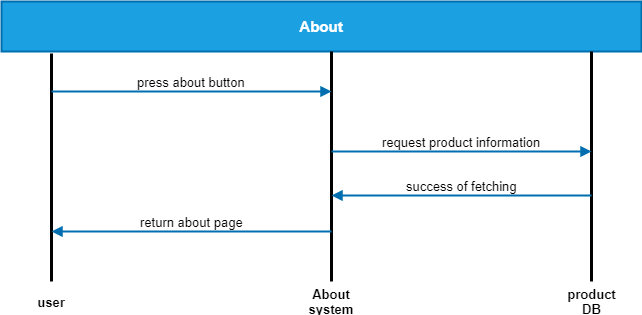


Figure . About System Sequence Diagram

## **5.4 State Diagram**

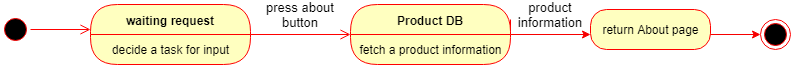


Figure . About System State Diagram

# **6. Notice System**

## **6.1 Objective**

관리자가 공지사항 제목 및 내용을 작성하여 사용자에게 제공하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 Notice System의 구조를 표현하고 설명한다.

## **6.2 Class Diagram**

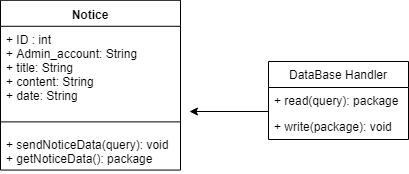


Figure . Notice System Class Diagram

A. DB handler

A.1 Attributes

해당사항 없음

A.2 Method

+ read(query): 해당 DB에서 원하는 데이터를 가져온다.

+ write(package): 해당 DB에 새로운 데이터를 저장한다.

B. Notice System

B.1 Attributes

+ ID: 시스템 내부에서 공지사항을 관리하기 위해 부여된 아이디

+ Admin\_account: 공지사항을 등록한 관리자의 계정

+ title: 공지사항의 제목

+ content: 공지사항의 내용

+ date: 공지사항이 등록된 날짜

B.2 methods

+ sendNoticeData(query): NoticeDB에 관리자에 의해 등록될 공지사항 데이터를 보낸다.

+ getNoticeData() : NoticeDB에서 사용자에 의해 요청된 데이터를 받는다.

**6.3 Sequence Diagram**

### A. Notice System (Admin use)

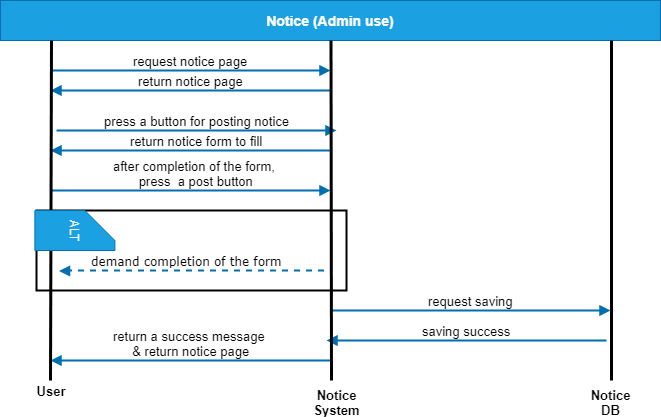
****

Figure . Notice System (Admin Use) Sequence Diagram

### B. Notice System (User use) Sequence Diagram

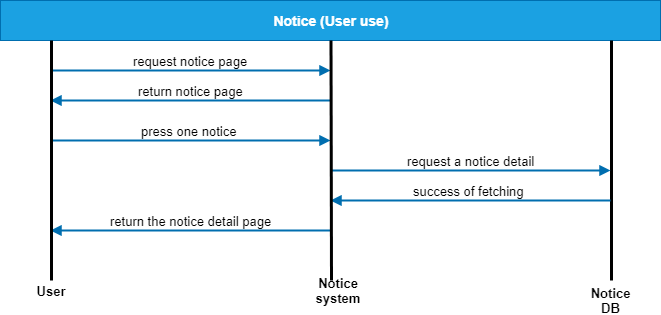
****

Figure . Notice System (User Use) Sequence Diagram

**6.4 State Diagram**

### A. Notice System (Admin use)

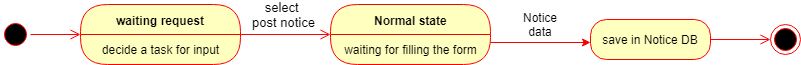
****

Figure . Notice System(Admin Use) State Diagram

### B. Notice System (User use)

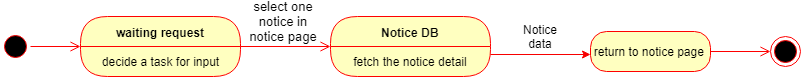
****

Figure . Notice System(User use) State Diagram

# **7. Hobby Test System**

## **7.1 Objective**

사용자 기본 정보(연령, 성별 등)와 테스트 과정에서 얻은 질문에 대한 사용자의 답변을 기반으로 취향을 분석하여 취미 상품을 추천하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 Hobby Test System의 구조를 표현하고 설명한다.

## **7.2 Class Diagram**

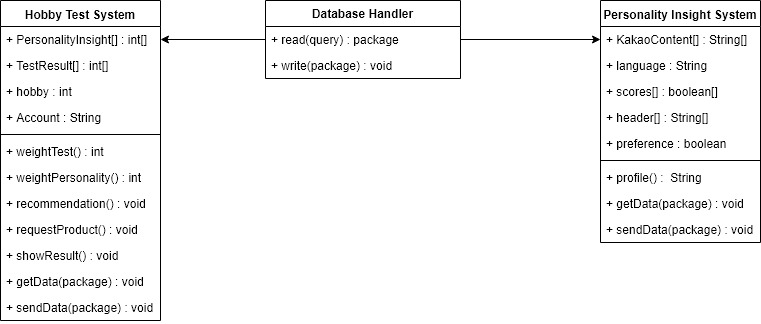


Figure . Hobby Test System Class Diagram

A. Database Handler : DB에 연결하여 테이블을 조합 또는 업데이트하며, 데이터 저장, 수정, 검색 및 삭제 등을 쉽게 할 수 있도록 도움을 주는 클래스

A.1 Attribute

없음

A.2 Methods

+ read() : 작업을 필요로하는 Data를 DB에서 읽어온다

+ write() : DB에 Data를 저장한다

B. Hobby Test System : 해당 유저의 취미 정보를 산출 또는 존재하는 정보를 송출하는 시스템

B1. Attribute

+ Account : 해당 유저의 계정 (식별)

+ PersonalityInsight [] : Hobby DB로부터 갖고온 Personality Insight System의 산출물

+ TestResult[] : Hobby Test System에서 recommendation() Method의 산출물

+ Hobby[] : TestResult[]로 부터 산출된 해당 유저의 선호 취미

B2. Methods

+ weightTest() : Chwimi시험에 대한 가중치 산출

+ weightPersonality() : Personality Insight 가중치 산출

+ recommendation() : 추천 시스템

+ requestProduct() : Product DB로부터 Hobby에 맞는 제품 데이터 요구

+ showResult() : 결과를 송출

+ getData() : DB로부터 Data를 읽어온다

+ sendData() : DB에 Data를 업데이트

C. Personality Insight System : 해당 유저의 Kakao Story의 Content Data를 받아와서 IBM Personality Insight API를 사용해서 해당 유저의 성격 및 선호도를 산출하는 시스템

C1. Attributes

+ KakaoContent[] : Kakao DB로부터 읽어온 Content Data

+ language : Content Data의 언어

+ scores[] : profile() Method의 산출물, 각 항목에 대한 숫자

+ header[] : profile() Method의 산출물, 각 항목의 제목

+ preference : profile() Method의 산출물, 해당 유저의 선호 사항

C2. Methods

+ profile() : IBM에서 제공하는 Personality Insight API의 Method

+ getData() : DB로부터 Data를 읽어온다

+ sendData() : DB에 Data를 업데이트

## **7.3 Sequence Diagram**

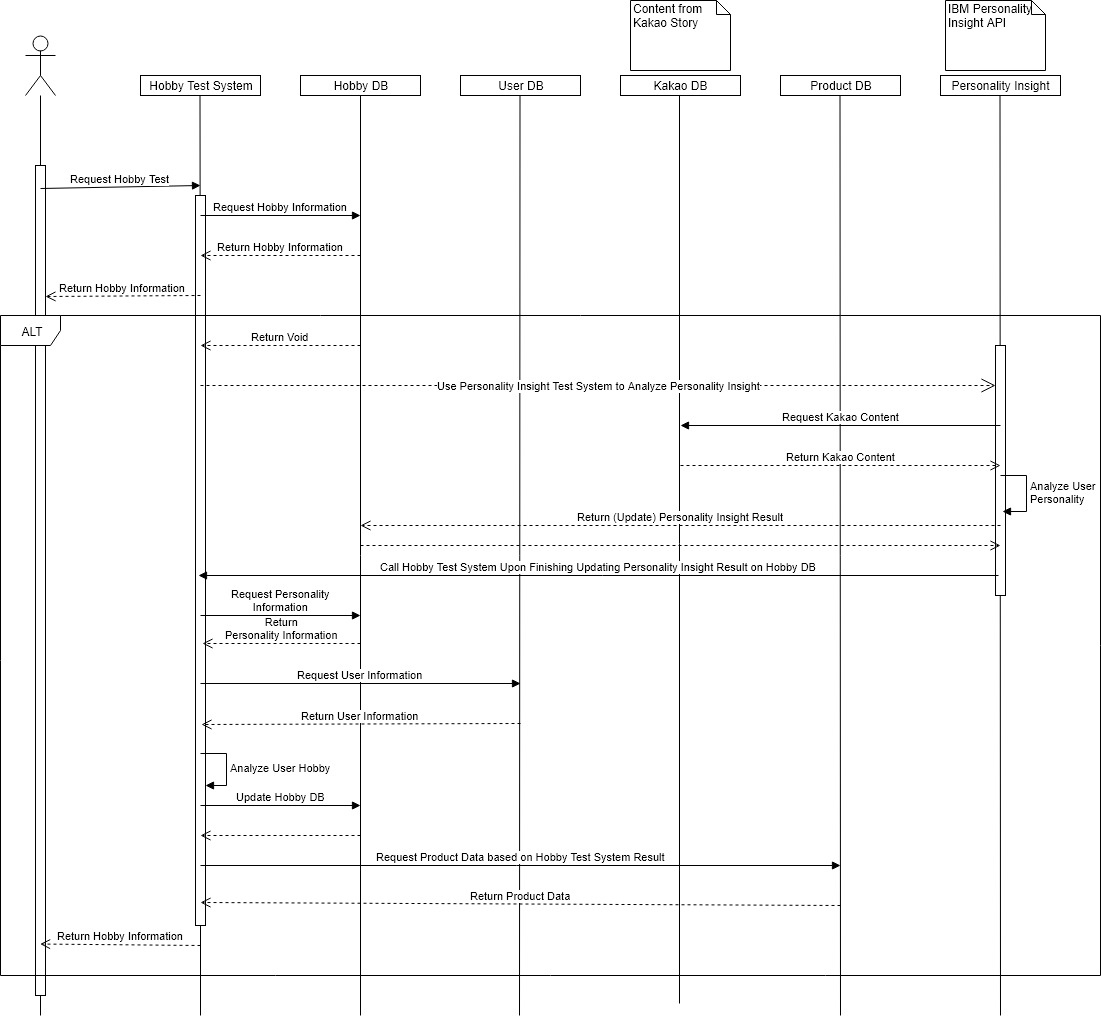


Figure . Hobby Test System Sequence Diagram

## **7.4 State Diagram**

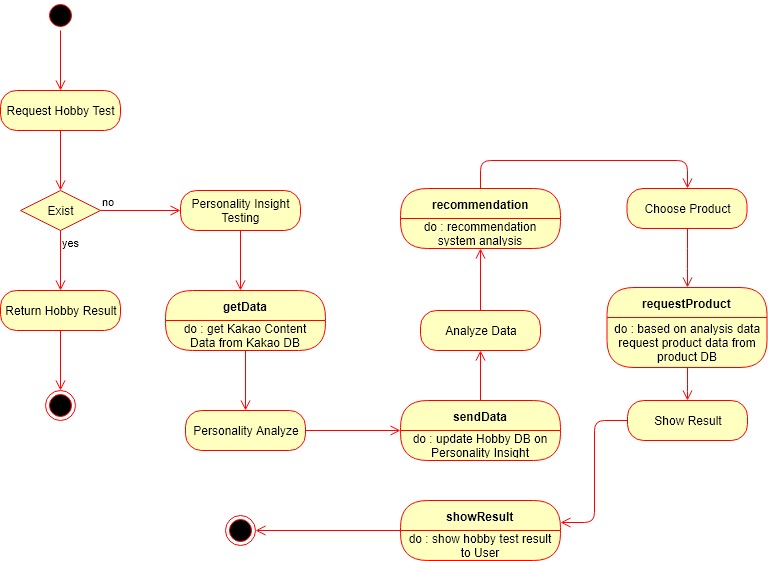
****

Figure . Hobby Test System State Diagram

# **8. Subscribe Management System**

## **8.1 Objective**

사용자가 취미 분석 결과를 바탕으로 취미 상품을 일정 기간 동안 배송 받을 것을 동의하여 결제하고 사용자의 이해를 돕기 위해 해당 취미 상품 이용 방법을 메인 페이지에 영상으로 제공하는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 Subscribe System의 구조를 표현하고 설명한다.

## **8.2 Class Diagram**

### A. Subscribe System

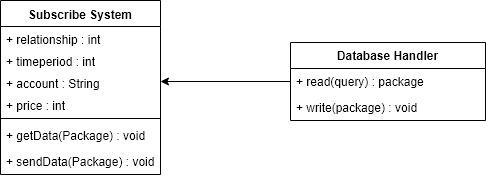


Figure . Subscribe System Class Diagram

A. Database Handler : DB에 연결하여 테이블을 조합 또는 업데이트하며, 데이터 저장, 수정, 검색 및 삭제 등을 쉽게 할 수 있도록 도움을 주는 클래스

A1. Attribute

없음

A2. Methods

+ read() : 작업을 필요로하는 Data를 DB에서 읽어온다

+ write() : DB에 Data를 저장한다

B. Subscribe System : 해당 유저가 구독을 신청하면 구독 여부를 업데이트 하는 시스템

B1. Attribute

+ relationship : 해당 유저와의 관계

+ timeperiod : 해당 유저가 구독하려는 개월 수

+ account : 해당 유저 계정

+ price : 구독 가격

B2. Methods

+ getData() : DB로부터 Data를 읽어온다

+ sendData() : DB에 Data를 업데이트

### B. Video System

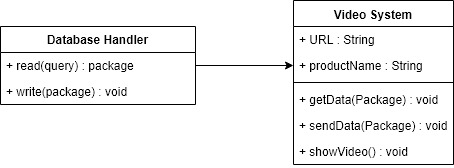


Figure . Video System Class Diagram

A. Database Handler : DB에 연결하여 테이블을 조합 또는 업데이트하며, 데이터 저장, 수정, 검색 및 삭제 등을 쉽게 할 수 있도록 도움을 주는 클래스

A1. Attribute

없음

A2. Methods

+ read() : 작업을 필요로하는 Data를 DB에서 읽어온다

+ write() : DB에 Data를 저장한다

B. Video System : 해당 유저가 Main Page 접속시 구독 정보를 토대로 영상을 Main Page에 송출하는 시스템

B1. Attribute

+ URL : Product DB로부터 갖고온 해당 제품에 대한 영상 URL

+ productName : Hobby DB로 부터 받은 제품 이름

B2. Methods

+ getData() : DB로부터 Data를 읽어온다

+ sendData() : DB에 Data를 업데이트

+ showVideo() : 영상을 송출

## **8.3 Sequence Diagram**

### A. Subscribe System

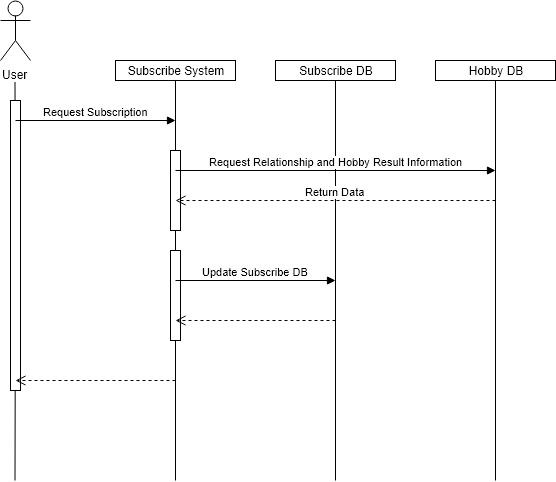


Figure . Subscribe System Sequence Diagram

### B. Video System

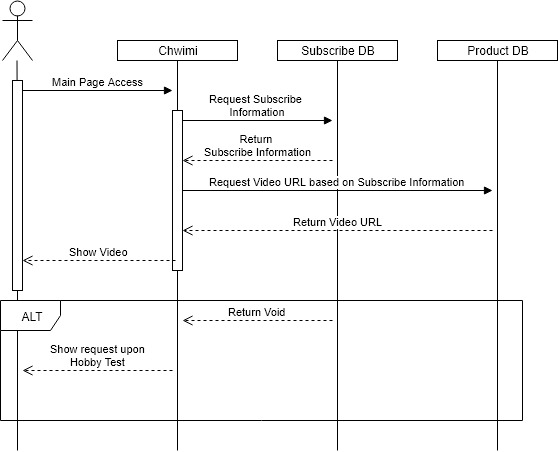


Figure . Video System Sequence Diagram

## **8.4 State Diagram**

### A. Subscribe System

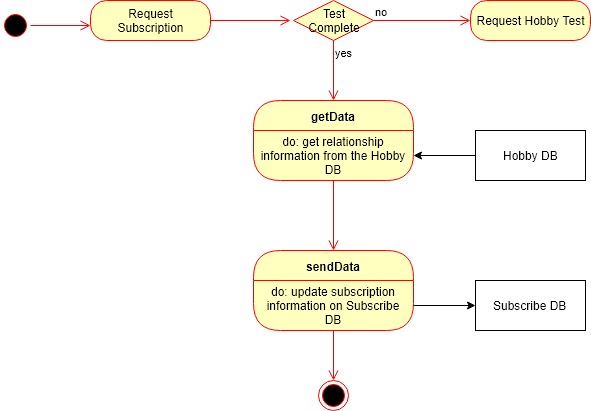


Figure . Subscribe System State Diagram

### B. Video System

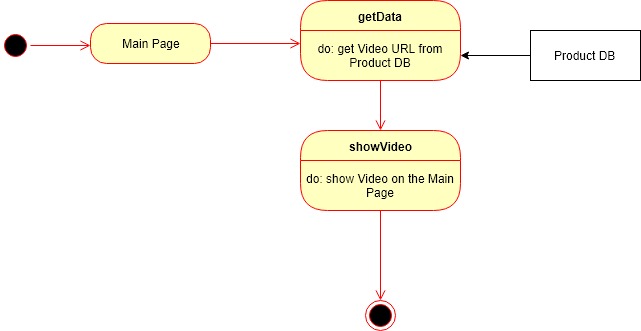


Figure . Video System State Diagram

# **9. Review Management System**

## **9.1 Objective**

Chwimi 시스템에서 제공하는 취미 상품을 경험한 사용자가 해당 상품에 대한 후기를 작성거나 다른 사용자가 작성한 후기를 읽고 검색할 수 있고, 관리자가 이에 댓글을 남기는 시스템을 설명한다. Class Diagram, Sequence Diagram과 State Diagram을 통해 Review System의 구조를 표현하고 설명한다.

## **9.2 Class Diagram**

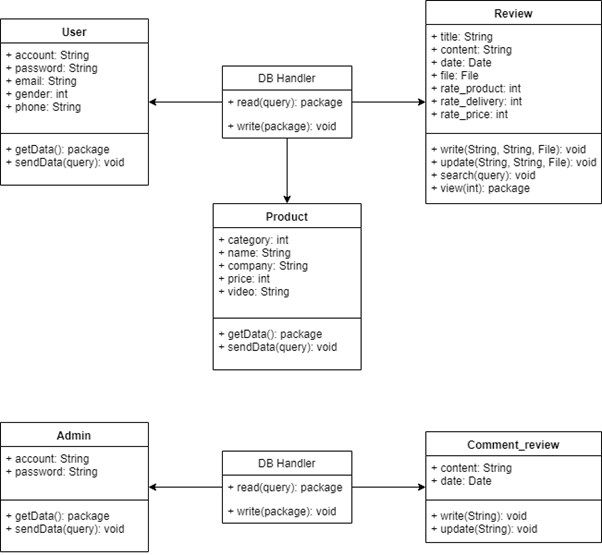


Figure . Review Management System Class Diagram

1. **DB Handler**

A.1. Attributes

해당 사항 없음.

A.2. Methods

+ package read(query): 해당 DB에서 원하는 데이터를 읽어온다.

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

1. **User**

B.1. Attributes

+ account: 사용자 아이디

+ password: 사용자 비밀번호

+ email: 사용자 이메일

+ gender: 사용자 성별

+ phone: 사용자 전화번호

B.2. Methods

+ package getData(): DB로부터 데이터를 받는다.

+ void sendData(query): DB로 데이터를 보낸다.

1. **Product**

C.1. Attributes

+ category: 상품 카테고리

+ name: 상품명

+ company: 상품 제조회사

+ price: 상품 가격

+ video: 상품 비디오 링크

C.2. Methods

+ package getData(): DB로부터 데이터를 받는다.

+ void sendData(query): DB로 데이터를 보낸다.

1. **Review**

D.1. Attributes

+ title: 후기 제목

+ content: 후기 내용

+ date: 작성 날짜

+ file: 첨부 파일

+ rate\_product: 상품 평점

+ rate\_delivery: 배송 평점

+ rate\_price: 가격 평점

D.2. Methods

+ void write(String, String, File): 글 작성

+ void update(String, String, File): 글 수정

+ void search(query): 글 검색

+ package view(int): 해당 글 보기

1. **Admin**

E.1. Attributes

+ account: 관리자 아이디

+ password: 관리자 비밀번호

E.2. Methods

+ package getData(): DB로부터 데이터를 받는다.

+ void sendData(query): DB로 데이터를 보낸다.

1. **Comment\_review**

F.1. Attributes

+ content: 후기 댓글 내용

+ date: 작성 날짜

F.2. Methods

+ void write(String): 댓글 작성

+ void update(String): 댓글 수정

## **9.3 Sequence Diagram**

### A. View

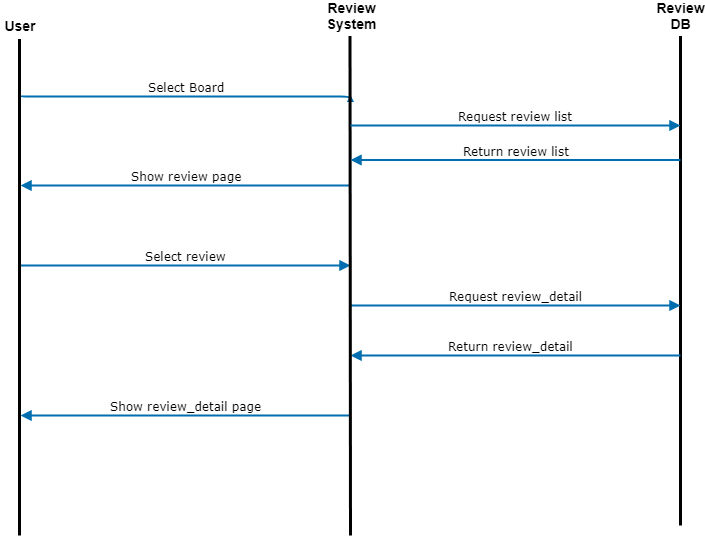


Figure . View Sequence Diagram

### B. Write

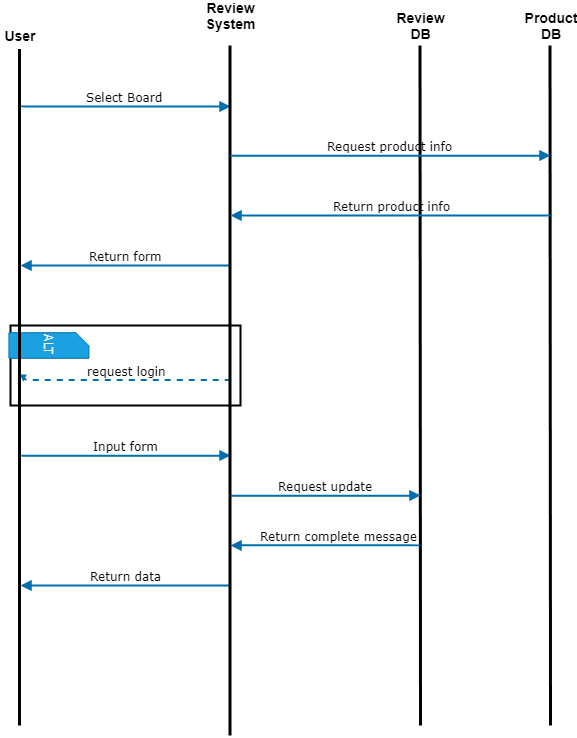


Figure . Write Sequence Diagram

### C. Update

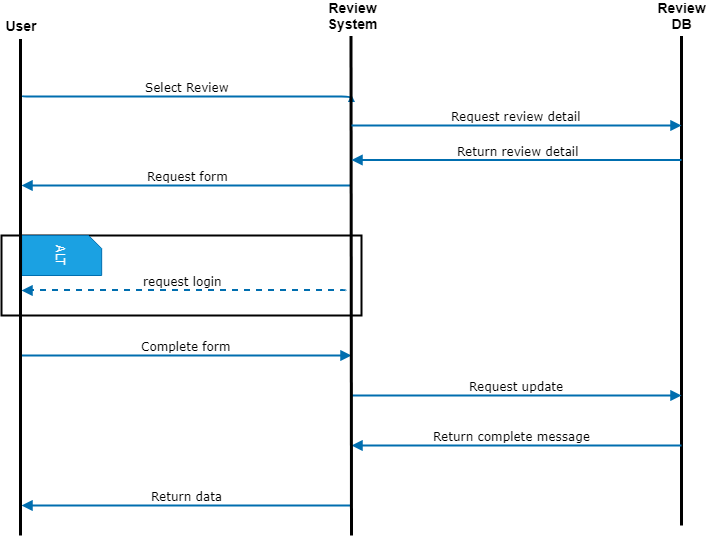


Figure . Update Sequence Diagram

### D. Comment

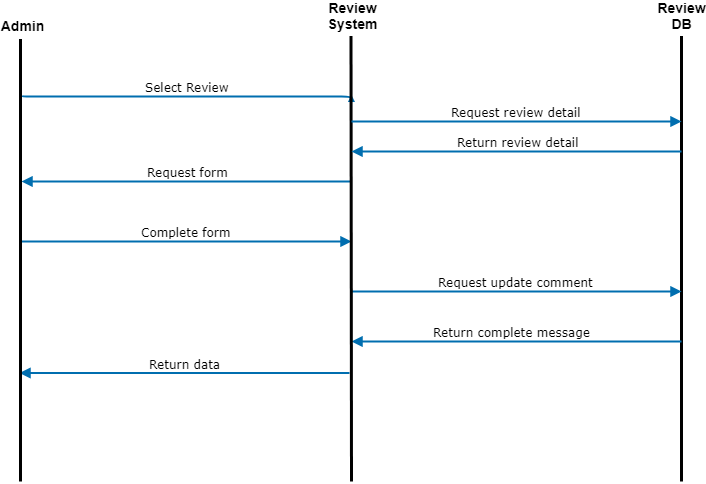


Figure . Comment Sequence Diagram

## **9.4 State Diagram**

## View

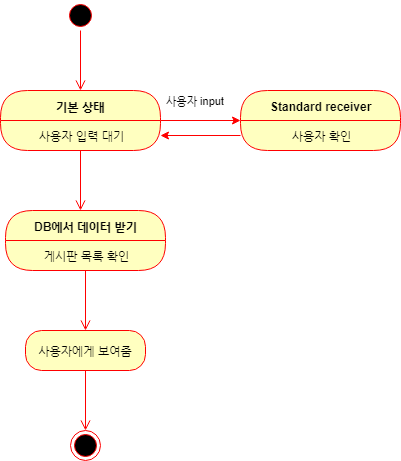


Figure . View State Diagram

### B. Write

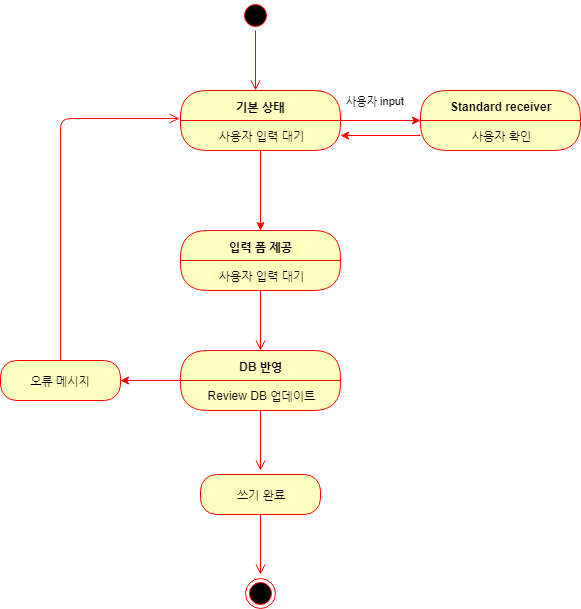


Figure . Write State Diagram

### C. Update

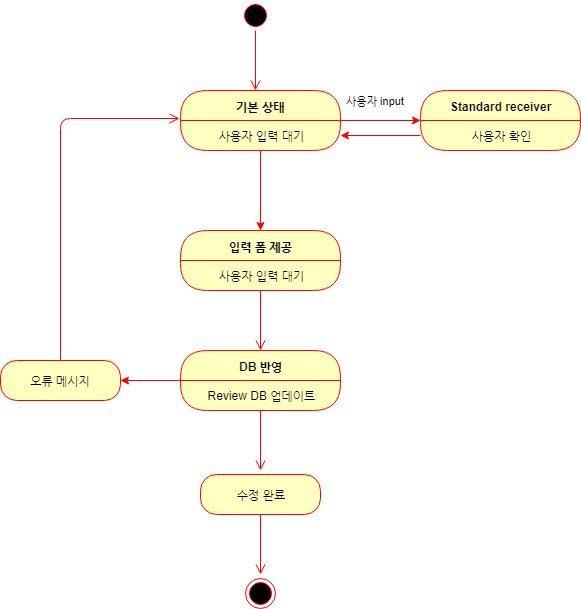


Figure . Update State Diagram

### Comment

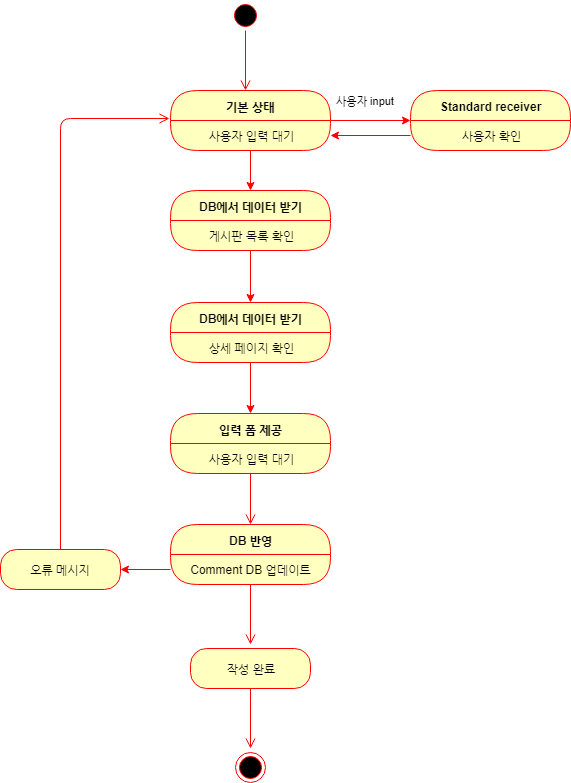


Figure . Comment State Diagram

# **10. Question Management System**

## **10.1 Objective**

사용자가 궁금한 점을 질문하거나 다른 사용자가 작성한 질문과 그에 대한 답변을 읽고 검색할 수 있고, 관리자가 이에 댓글을 남기는 시스템을 설명한다. 게시물과 댓글의 CRUD 과정에서 발생하는 데이터의 처리와 이를 출력하는 시스템의 설계를 Class Diagram, Sequence Diagram과 State Diagram을 통해 Q&A System의 구조를 표현하고 설명한다.

## **10.2 Class Diagram**

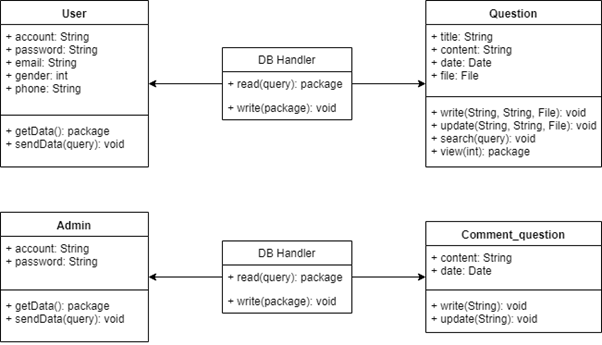


Figure . Question Management System Class Diagram

1. **DB Handler**

A.1. Attributes

해당 사항 없음.

A.2. Methods

+ package read(query): 해당 DB에서 원하는 데이터를 읽어온다.

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

1. **User**

B.1. Attributes

+ account: 사용자 아이디

+ password: 사용자 비밀번호

+ email: 사용자 이메일

+ gender: 사용자 성별

+ phone: 사용자 전화번호

B.2. Methods

+ package getData(): DB로부터 데이터를 받는다.

+ void sendData(query): DB로 데이터를 보낸다.

1. **Question**

C.1. Attributes

+ title: 질문 제목

+ content: 질문 내용

+ date: 작성 날짜

+ file: 첨부 파일

C.2. Methods

+ void write(String, String, File): 글 작성

+ void update(String, String, File): 글 수정

+ void search(query): 글 검색

+ package view(int): 해당 글 보기

1. **Admin**

E.1. Attributes

+ account: 관리자 아이디

+ password: 관리자 비밀번호

E.2. Methods

+ package getData(): DB로부터 데이터를 받는다.

+ void sendData(query): DB로 데이터를 보낸다.

1. **Comment\_question**

E.1. Attributes

+ content: 질문 댓글 내용

+ date: 작성 날짜

E.2. Methods

+ void write(String): 댓글 작성

+ void update(String): 댓글 수정

## **10.3 Sequence Diagram**

### A. View

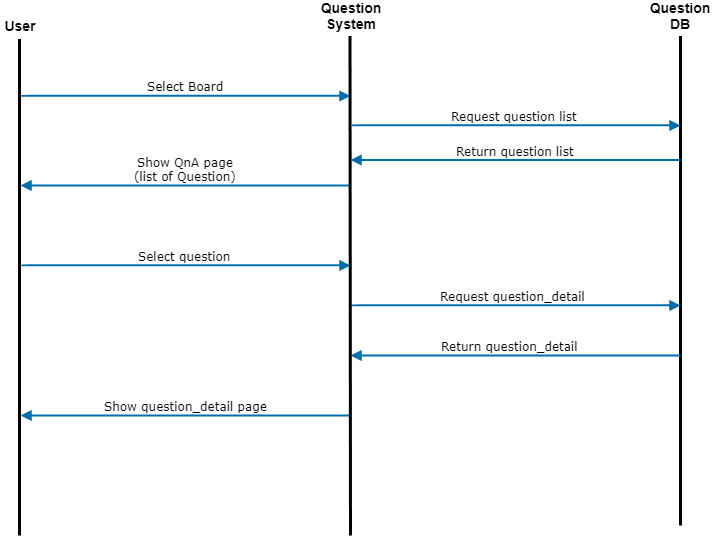


Figure . View Sequence Diagram

### B. Write

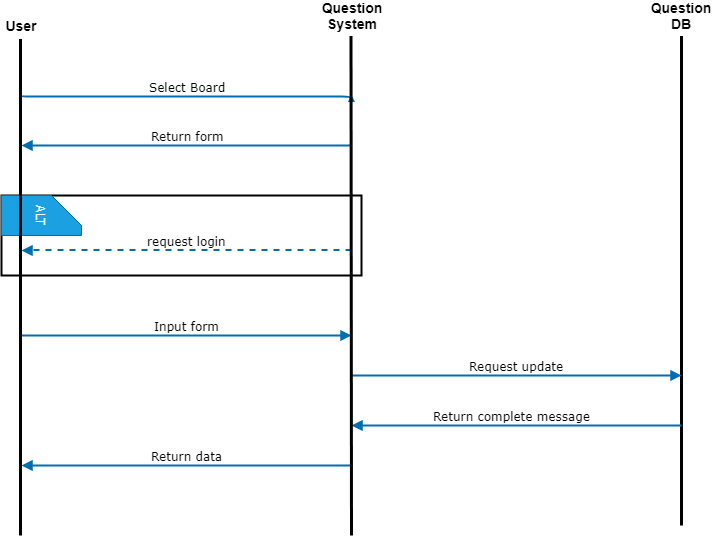


Figure . Write Sequence Diagram

### C. Update

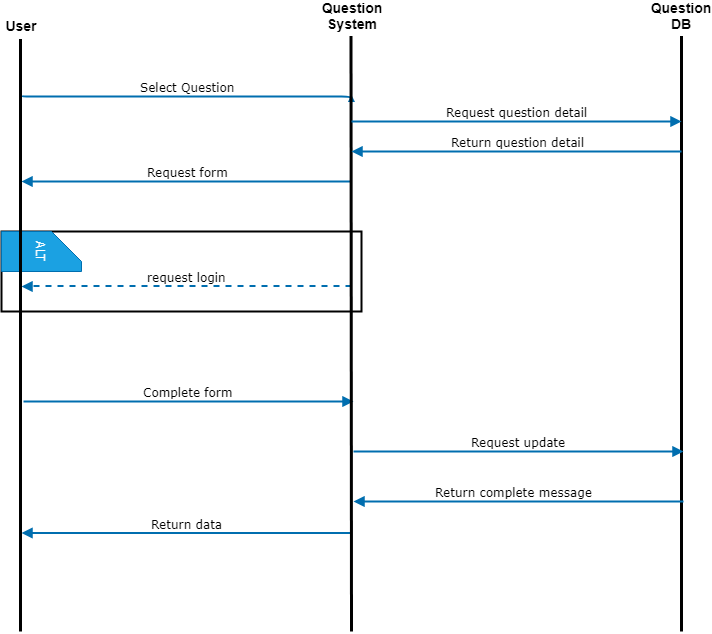


Figure . Update Sequence Diagram

### D. Comment

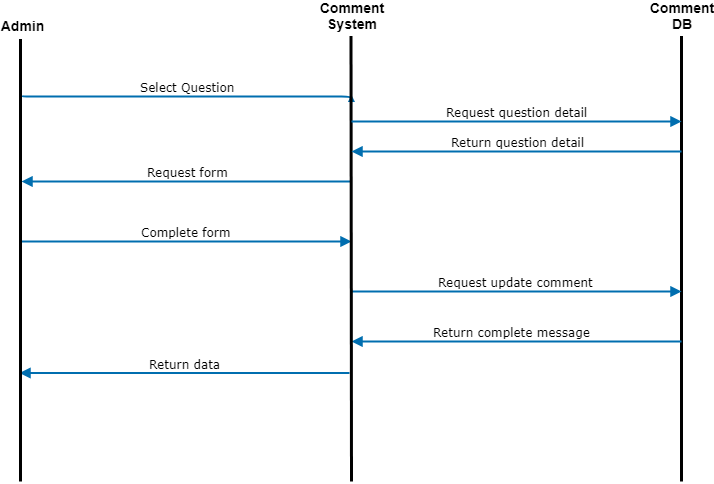


Figure . Comment Sequence Diagram

## **10.4 State Diagram**

### A. View

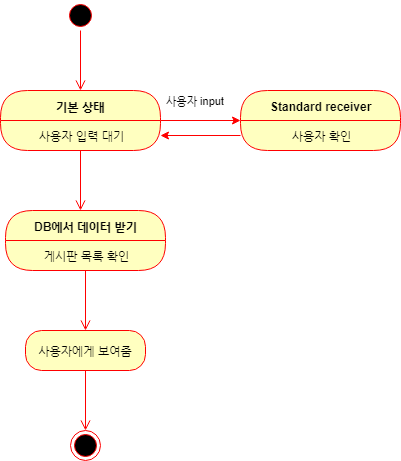


Figure . View State Diagram

### B. Write

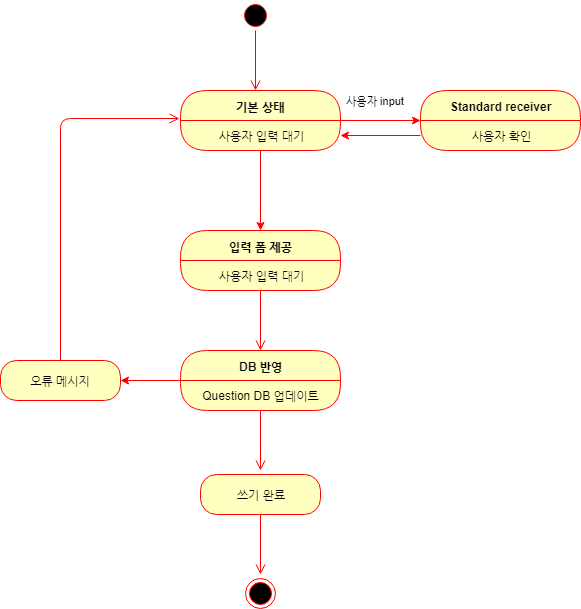


Figure . Write State Diagram

### C. Update

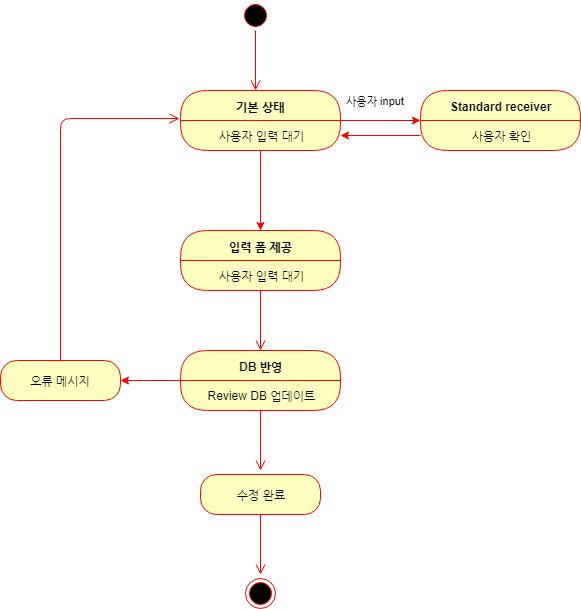


Figure . Update State Diagram

### D. Comment

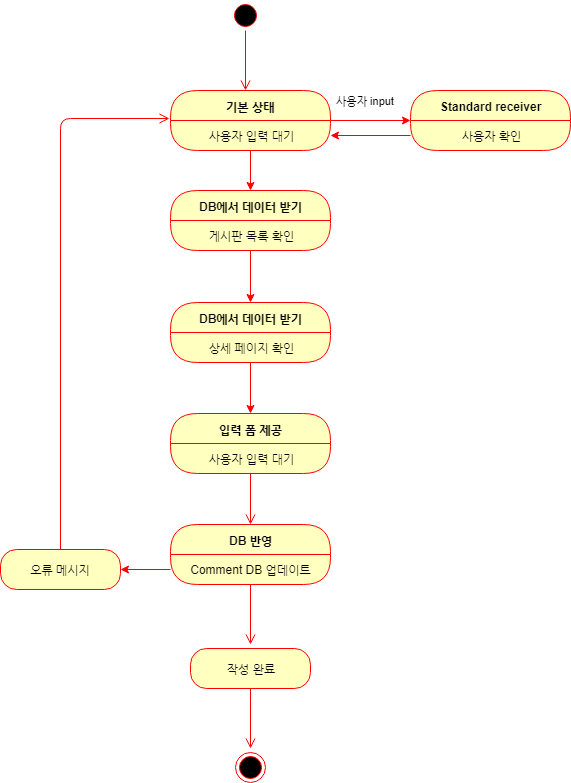


Figure . Comment State Diagram

# **11. Protocol Design**

## **11.1 Objective**

Protocol Design에서는 Subsystem들이 상호작용하는 프로토콜에 대해 서술한다. 프로토콜의 기본 형식은 JSON(Java Script Object Notation)을 기본으로 하며, 통신하는 메시지의 형식, 용도와 의미를 설명한다.

## **11.2 JSON**

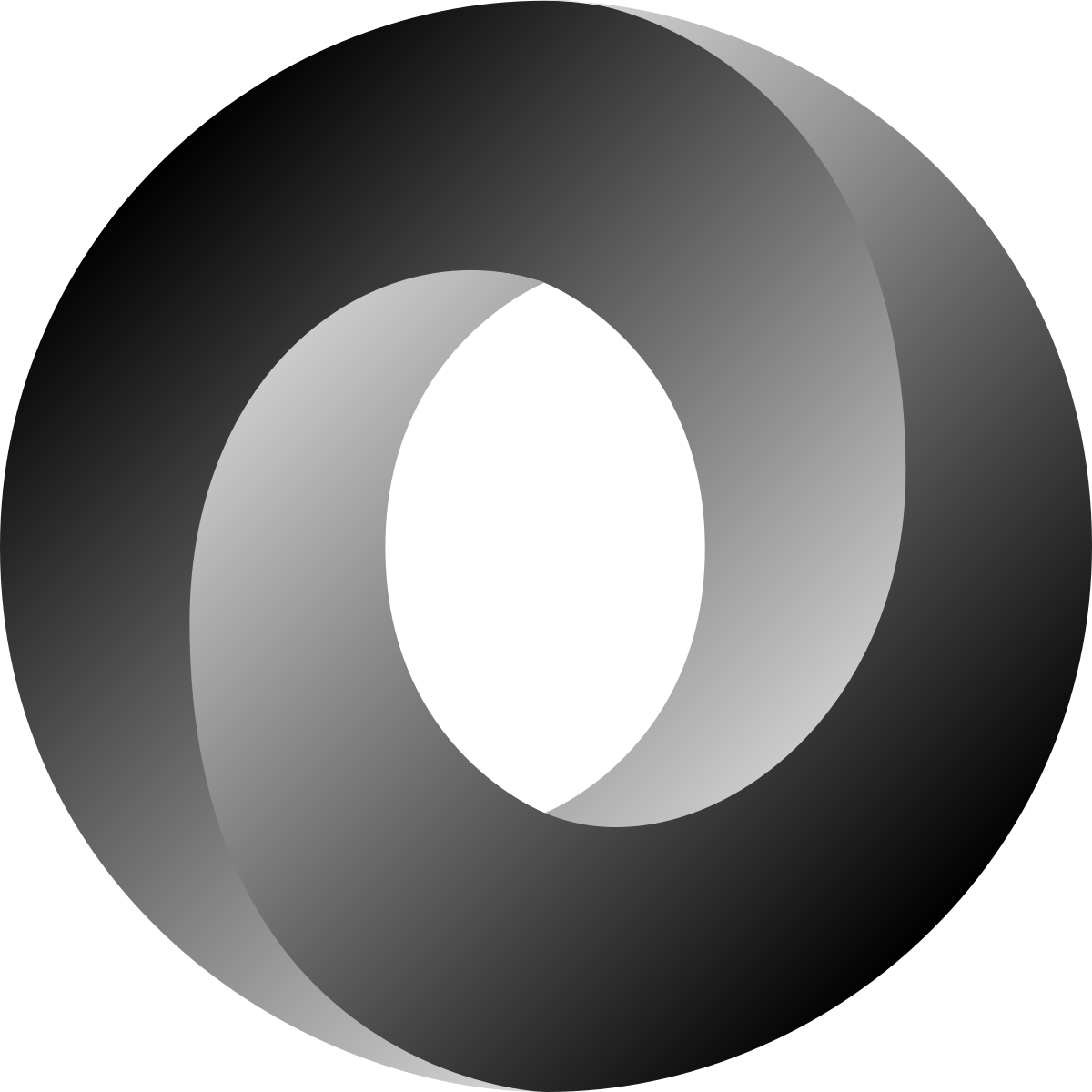


Figure . Logo of JSON

JSON은 속성-값 쌍 (Attribute-Value)으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 대표적인 데이터 교환 방식이다. 자바스크립트에 기반하여 만들어진 데이터 표현형식이지만 프로그래밍 언어나 플랫폼에 독립적인 특성을 갖고 있어 다양한 언어에서 JSON을 활용할 수 있다. 자료의 종류에 큰 제한이 없으며 컴퓨터 프로그램 변수 값을 표현하는데 적합하다. 또한 기능이 적어 파싱(Parsing)이 빠르다는 장점을 가지고 있다.

## **11.3 Protocol Design**

### A. Overview

HTTP 통신에서 client와 server 사이에서 전송되는 메시지의 형태를 용도 별로 정의한다. Client에서의 요청(Request) 메시지와 server에서의 응답(Response) 메시지로 구분한다.

### B. Sign up Protocol

B.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Account | 사용자의 계정 |
| Pw | 사용자의 비밀번호 |
| Email | 사용자의 이메일 주소 |
| Phone | 사용자의 핸드폰 번호 |
| Gender | 사용자의 성별 |

Table . Sign Up Protocol Request

B.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Signup\_success | 회원가입 성공 여부 |

Table . Sign Up Protocol Response

### C. Log in Protocol

C.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Account | 사용자의 계정 |
| Pw | 사용자의 비밀번호 |

Table . Log In Protocol Request

C.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Login\_success | 로그인 성공 여부 |

Table . Log In Protocol Response

### D. My page Protocol

D.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Account | 사용자의 계정 |

Table . My Page Protocol Request

D.2 Response

|  |  |  |
| --- | --- | --- |
| Attribute | Value | |
| User\_info | 사용자 정보 | |
| Email | 사용자의 이메일 주소 |
| Phone | 사용자의 핸드폰 번호 |
| Gender | 사용자의 성별 |
| Hobby\_result | 사용자의 취미 분석 테스트 결과 | |
| Subscribe\_info | 구독 정보 | |
| Subscribe\_relationship | 구독자와 사용자의 관계 |
| Subscribe\_startingDate | 구독시작 시기 |
| Subscribe\_month | 구독 서비스를 받는 기간 |
| Subscribe\_address | 구독 서비스가 제공되는 주소 |
| Subscribe\_productName | 제공되는 구독 서비스의 이름 |

Table . My Page Protocol Response

### E. About Protocol

E.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Chwimi\_info | Chwimi서비스 정보 |

Table . About Protocol Request

E.2 Response

|  |  |  |
| --- | --- | --- |
| Attribute | Value | |
| Product\_id | 서비스가 제공하는 상품 번호 | |
| Product\_name | 취미상품명 |
| Product\_category | 취미상품이 속한 카테고리 |

Table . About Protocol Response

### F. Notice Protocol

F.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Notice\_id | 공지사항 번호 |

Table . Notice Protocol Request

F.2 Response

|  |  |  |
| --- | --- | --- |
| Attribute | Value | |
| Notice\_info | 서비스가 제공하는 상품 정보 | |
| Title | 공지사항 제목 |
| content | 공지사항 내용 |
| Date | 공지사항 등록 날짜 |

Table . Notice Protocol Response

### G. Hobby Test Protocol

G.1 Request

|  |  |  |
| --- | --- | --- |
| Attribute | Value | |
| Account | 사용자의 계정 | |
| Subject\_info | 취미테스트 대상자 정보 | |
| Subject\_relationship | 사용자와 취미테스트 대상 간의 관계 |
| Birth\_date | 취미테스트 대상자의 생일 |
| Gender | 취미테스트 대상자의 성별 |
| Kakao\_content | 카카오 스토리 계정 정보 | |
| Language | 컨텐츠 언어 |
| Post\_id | 포스트 번호 |
| content | 포스트 내용 |
|  | media | 포스트 내의 사진 |
| Test\_response | 취미 테스트 질문에 대한 답변 | |

Table . Hobby Test Protocol Request

G.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Test\_result | 취미 테스트 결과 |

Table . Hobby Test Protocol Response

### H. Subscribe Protocol

H.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Account | 사용자의 계정 |
| Time\_period | 사용자가 구독하려는 개월 수 |

Table . Subscribe Protocol Request

H.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Subscribe\_success | 구독 성공 여부 |

Table . Subscribe Protocol Response

### I. Video Protocol

I.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Account | 사용자의 계정 |
| Subscribe\_info | 구독 정보 |

Table . Video Protocol Request

I.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Vidoe\_url | 구독한 취미 상품에 대한 영상 URL |

Table . Video Protocol Response

### J. Review Protocol

J.1 Write / Update Review

J.1.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Account | 사용자의 계정 |
| Title | 후기 제목 |
| Content | 후기 내용 |
| Date | 작성 날짜 |
| File | 첨부파일 |
| Rate\_product | 상품 평점 |
| Rate\_delivery | 배송 평점 |
| Rate\_price | 가격 평점 |

Table . Write/Update Review Protocol Request

J.1.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Write\_success | 후기 작성/수정 성공 여부 |

Table . Write/Update Review Protocol Response

J.2 Write Review

J.2.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Review\_id | 후기 번호 |

Table . View Review Protocol Request

J.2.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Account | 사용자의 계정 |
| Title | 후기 제목 |
| Content | 후기 내용 |
| Date | 작성 날짜 |
| File | 첨부파일 |
| Rate\_product | 상품 평점 |
| Rate\_delivery | 배송 평점 |
| Rate\_price | 가격 평점 |
| Comment | 관리자가 작성한 댓글 |

Table . View Review Protocol Response

J.3 Search Review

J.3.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Product\_id | 취미 상품 번호 |

Table . Search Review Protocol Request

J.3.2 Response

|  |  |  |
| --- | --- | --- |
| Attribute | Value | |
| Search\_result | 검색 결과 | |
| Account | 사용자 계정 |
| Title | 후기 제목 |
| Content | 후기 내용 |
| Date | 작성 날짜 |
| media | 포스트 내의 사진 |

Table . Search Review Protocol Response

### K. Question Protocol

K.1 Write/Update Q&A

K.1.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Account | 사용자의 계정 |
| Title | 후기 제목 |
| Content | 후기 내용 |
| Date | 작성 날짜 |
| File | 첨부파일 |

Table . Write/Update Question Protocol Request

K.1.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Write\_success | 질문 작성/수정 성공 여부 |

Table . Write/Update Question Protocol Response

K.2 View Q&A

K.2.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Question\_id | 질문 번호 |

Table . View Question Protocl Request

K.2.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Account | 사용자의 계정 |
| Title | 후기 제목 |
| Content | 후기 내용 |
| Date | 작성 날짜 |
| File | 첨부파일 |
| Comment | 관리자가 작성한 댓글 |

Table . View Question Protocol Response

K.3 Search Q&A

K.3.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Keyword | 검색 키워드 |

Table . Search Question Protocol Request

K.3.2 Response

|  |  |  |
| --- | --- | --- |
| Attribute | Value | |
| Search\_result | 검색 결과 | |
| Account | 사용자 계정 |
| Title | 후기 제목 |
| Content | 후기 내용 |
| Date | 작성 날짜 |
| File | 첨부파일 |

Table . Search Question Protocol Response

### L. Comment Protocol

L.1 Request

|  |  |
| --- | --- |
| Attribute | Value |
| Content | 후기 내용 |
| Date | 작성 날짜 |

Table . Write/Update Comment Protocol Request

L.2 Response

|  |  |
| --- | --- |
| Attribute | Value |
| Write\_success | 댓글 작성/수정 성공 여부 |

Table . Write/Update Comment Protocol Response

# **12. Database Design**

## **12.1 Objective**

Database Design은 요구사항 명세서에서 기술한 데이터베이스 요구사항을 바탕으로 수정사항을 수정하여 다시 요구사항을 작성하였다. 요구사항을 바탕으로 ER Diagram을 작성하고, 이를 이용하여 Relational Schema를 작성하고, Normalization을 통해 Redundancy와 Anomaly를 제거한 후 마지막으로 SQL DDL을 작성한다.

## **12.2 ER Diagram**

### A. Entity

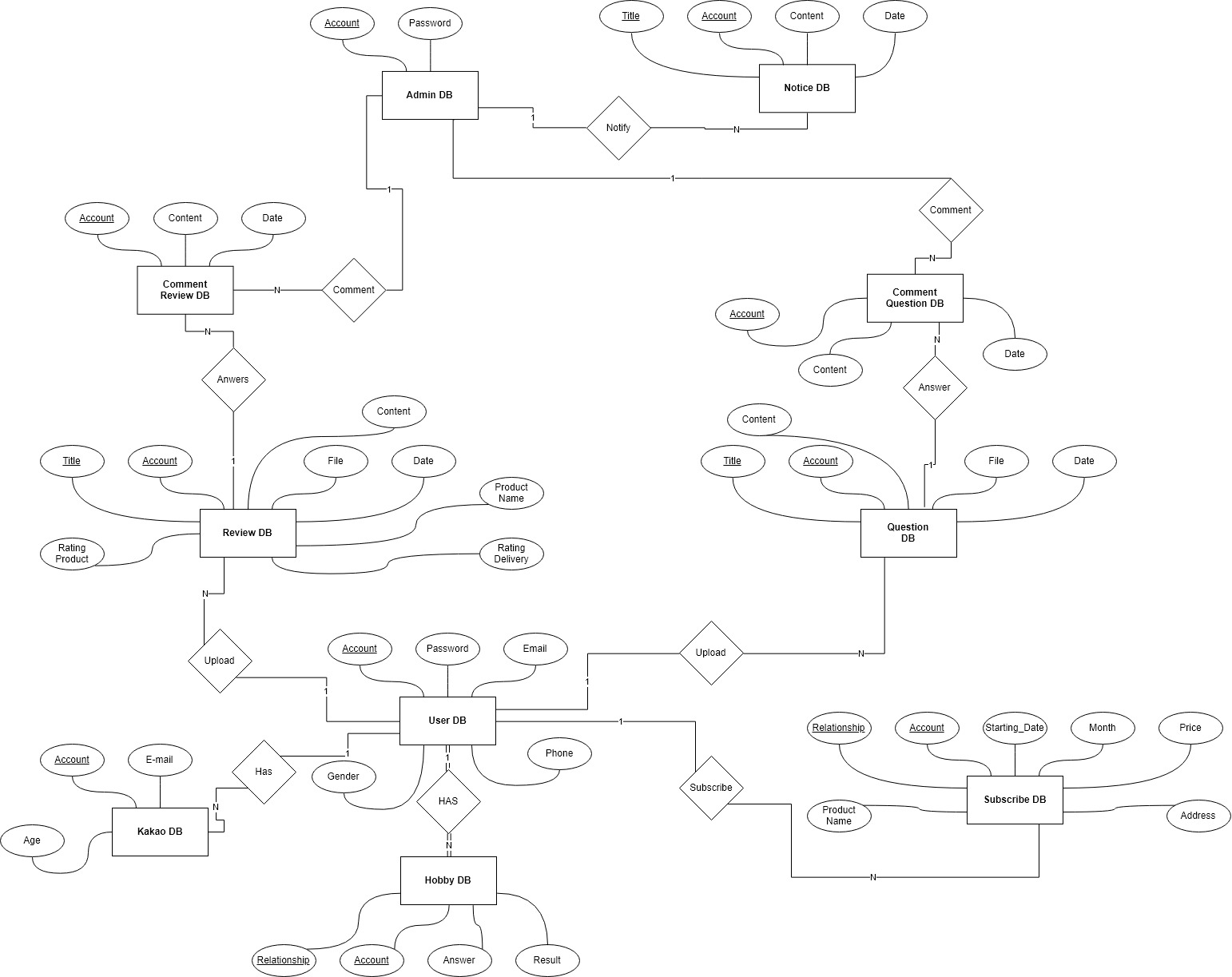


Figure . Overall ER Diagram

### Relationship

C1. User DB

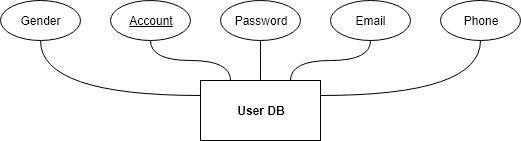


Figure . User DB ER Diagram

C2. Admin DB

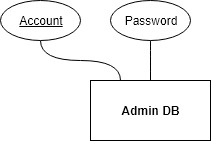


Figure . Admin DB ER Diagram

C3. Hobby DB

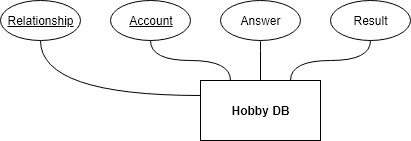


Figure . Hobby DB ER Diagram

C4. Subscribe DB

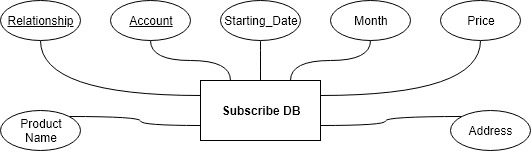


Figure . Subscribe DB ER Diagram

C5. Product DB

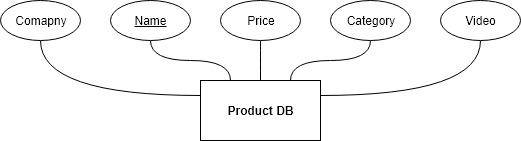


Figure . Product DB ER Diagram

C6. Question DB

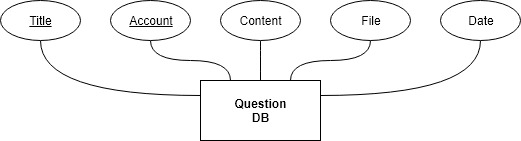


Figure . Question DB ER Diagram

C7. Review DB

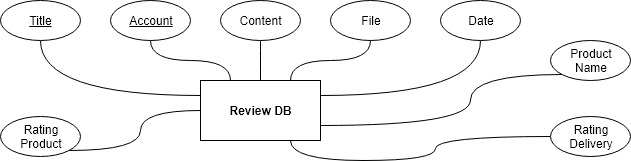


Figure . Review DB ER Diagram

C8. Comment Question DB

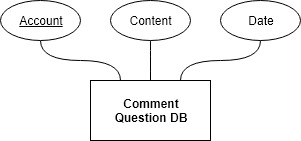


Figure . Comment Question DB ER Diagram

C9. Comment Review DB

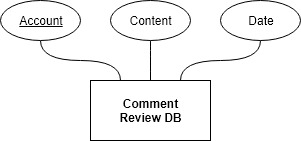


Figure . Comment Review DB ER Diagram

C10. Kakao DB

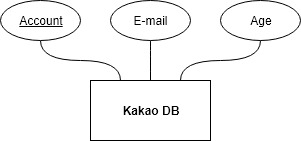


Figure . Kakao DB ER Diagram

C11. Notice DB

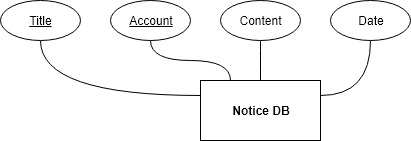


Figure . Notice DB ER Diagram

## **12.3 Relational Schema**

1. 사용자 DB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User\_account | Password | Email | Phone | Gender |

Primary key(PK) User\_account

Foreign Key(FK): 없음

Func Dep(FD): User\_account → {Password, Email, Phone, Gender}

Description: 회원가입 절차를 거친 고객의 정보를 보관하는 테이블이다. Primary Key로 User\_account를 가진다. Null값이 허용되는 변수는 없다.

2. 카카오 사용자 DB

|  |  |  |
| --- | --- | --- |
| User\_account | Content | password |

Primary key(PK): User\_account

Foreign Key(FK): 없음

Func Dep(FD): User\_account → {Content, password}

Description: 카카오를 이용하여 로그인한 고객의 정보를 보관하는 테이블이다. Primary Key로 User\_account를 가지며, Content만 Null값이 허용된다.

3. 관리자 DB

|  |  |
| --- | --- |
| Admin\_account | Password |

Primary key(PK): Admin\_account

Foreign Key(FK): 없음

Func Dep(FD): Admin\_account → Password

Description: 해당 시스템을 관리하는 관리자의 정보를 보관하는 테이블이다. Primary Key로 Admin\_account를 가지며, Null값이 허용되는 변수는 없다.

4. 취미 DB

|  |  |  |  |
| --- | --- | --- | --- |
| Relationship | User\_account | Answer | Result |

Primary key(PK): {Relationship, User\_account}

Foreign Key(FK): User\_account

Func Dep(FD): {Relationship, User\_account} → {Answer, Result}

Description: 사용자가 취미 분석 테스트 진행시 입력한 값과 테스트 분석 결과를 보관하는 테이블이다. Primary Key로 {User\_account, Relationship}을 가지며, Foreign Key로 User\_account를 가진다. Null값이 허용되는 변수는 없다.

5. 구독 DB

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User\_account | Relationship | Starting\_date | Month | Price | Address | Product\_name |

Primary key(PK): {User\_account, Relationship}

Foreign Key(FK): User\_account, Relationship

Func Dep(FD): {User\_account, Relationship} → {Starting\_date, Month, Price, Address, Product\_name}

Description: 사용자의 서비스 구독 정보를 보관하는 테이블이다. Primary Key로 {User\_account, Relationship}을 가지며, Foreign Key로 User\_account, Relationship을 가진다. Null값이 허용되는 변수는 없다.

6. 공지사항 DB

|  |  |  |  |
| --- | --- | --- | --- |
| Admin\_account | Title | Content | Date |

Primary key(PK): {Admin\_account, Title}

Foreign Key(FK): Admin\_account

Func Dep(FD):{Admin\_account, Title} → {Content, Date}

Description: 관리자가 등록한 공지사항 정보를 보관하는 테이블이다. Primary Key로 {Admin\_account, Title}을 가지며, Foreign Key로 Admin\_account를 가진다. Null값이 허용되는 변수는 없다.

7. 상품 DB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Category | Company | Price | Video |

Primary key(PK): Name

Foreign Key(FK): 없음

Func Dep(FD): Name → {Category, Company, Price, Video}

Description: 서비스로 제공될 취미 상품 정보를 보관하는 테이블이다. Primary Key로 Name을 가지며, Null값이 허용되는 변수는 없다.

8. 후기 DB

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User\_account | Date | Title | Content | File |
| Product | Rating\_Product | Rating\_Delivery | Rating\_Price |  |

Primary key(PK): {User\_account, Date, Title}

Foreign Key(FK): User\_account, Product

Func Dep(FD): {User\_account, Date, Title} → {Content, File, Product, Rating\_Product, Rating\_Delivery, Rating\_Price}

Description: 사용자가 등록한 후기를 보관하는 테이블이다. Primary Key로 {User\_account, Date, Title}을 가지며, Foreign key로 User\_account, Product를 가진다. File을 제외하고 Null값이 허용되는 변수는 없다.

9. 후기 댓글 DB

|  |  |  |  |
| --- | --- | --- | --- |
| Admin\_account | Review\_id | Content | Date |

Primary key(PK): {Admin\_account, review\_id,content}

Foreign Key(FK): Admin\_account, Review\_id

Func Dep(FD): {Admin\_account, Review\_id,Content} → {Date}

Description: 사용자가 등록한 후기에 대한 관리자의 댓글을 보관하는 테이블이다. Primary key로 {Admin\_account, Review\_id, content}를 가지며, Foreign key로 Admin\_account, Review\_id를 가진다. Null값이 허용되는 변수는 없다.

10. 질문 DB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User\_account | Date | Title | Content | File | Date |

Primary key(PK): {User\_account, Date, Title}

Foreign Key(FK): User\_account

Func Dep(FD): {User\_account, Date, Title} → {Content, File, Date}

Description: 사용자가 등록한 질문을 보관하는 테이블이다. Primary Key로 {User\_account, Date, Title}을 가지며, Foreign Key로 User\_account를 가진다. File을 제외하고 Null값이 허용되는 변수는 없다.

11. 질문 댓글 DB

|  |  |  |  |
| --- | --- | --- | --- |
| Admin\_account | Content | Question\_id | Date |

Primary key(PK): {Admin\_account, Content, Question\_id}

Foreign Key(FK): Admin\_account, Question\_id

Func Dep(FD): {Admin\_account, Content, Question\_id} → Date

Description: 사용자가 등록한 질문에 대한 댓글을 보관하는 테이블이다. Primary key로 {Admin\_account, Content, Question\_id}를 가지며, Foreign Key로 Admin\_account, Question\_id를 가진다. Null값이 허용되는 변수는 없다.

## **12.4 SQL DDL**

1. **User**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  | | --- | | **CREATE TABLE ‘User’(**  **‘User\_account’ VARCHAR(20) NOT\_NULL**  **‘Password’ VARCHAR(20) NOT\_NULL**  **‘Email’ VARCHAR(20) NOT\_NULL**  **‘Phone’ VARCHAR(20) NOT\_NULL**  **‘Gender’ INT(1) NOT\_NULL**  **PRIMARY KEY(‘User\_account’)**  **);** | |

**B. Kakao\_user**

|  |  |
| --- | --- |
|  |  |
|  | |  | | --- | | **CREATE TABLE ‘Kakao\_user’(**  **‘User\_account’ VARCHAR(20) NOT\_NULL,**  **‘Content’ VARCHAR(3000),**  **‘Password’ VARCHAR(20) NOT\_NULL,**  **PRIMARY KEY(‘User\_account’)**  **);** | |

**C. Hobby**

|  |  |
| --- | --- |
|  |  |
|  | |  | | --- | | **CREATE TABLE ‘Hobby’(**  **‘Relationship’ VARCHAR(10) NOT\_NULL,**  **‘User\_account’ VARCHAR(20) NOT\_NULL,**  **‘Answer’ VARCHAR(20) NOT\_NULL,**  **‘Result’ INT(1) NOT\_NULL,**  **PRIMARY KEY( ‘User\_account’, ’Relationship’),**  **FOREIGN KEY( ‘User\_account’) REFERENCES User(‘User\_account’) ON DELETE CASCADE**  **);** | |

**D. Subscribe**

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **CREATE TABLE ‘Subscribe’(**  **‘User\_account’ VARCHAR(20) NOT\_NULL,**  **‘Relationship’ VARCHAR(10) NOT\_NULL,**  **‘Starting\_Date’ DATETIME NOT\_NULL,**  **‘Month’ INT(1) NOT\_NULL,**  **‘Price’ INT(10) NOT\_NULL,**  **‘Address’ VARCHAR(20) NOT\_NULL,**  **‘Product\_Name’ VARCHAR(20) NOT\_NULL,**  **PRIMARY KEY(‘User\_account’, ’Relationship’),**  **FOREIGN KEY(‘User\_account’) REFERENCES Hobby(‘User\_account’) ON DELETE CASCADE,**  **FOREIGN KEY( ‘Relationship’) REFERENCES Hobby(‘Relationship’) ON DELETE CASCADE**  **);** | |

**E. Notice**

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **CREATE TABLE ‘Notice’(**  **‘Admin\_account’ VARCHAR(20) NOT\_NULL,**  **‘Title’ VARCHAR(20) NOT\_NULL,**  **‘Content’ VARCHAR(3000) NOT\_NULL,**  **‘Date’ DATETIME NOT\_NULL,**  **PRIMARY KEY(‘Admin\_account’,’Title’),**  **FOREIGN KEY(‘Admin\_account’) REFERENCES Admin(‘Admin\_account’) ON DELETE NO ACTION**  **);** | |

**F. Admin**

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **CREATE TABLE ‘Admin’(**  **‘User\_account’ VARCHAR(20) NOT\_NULL,**  **‘Password’ VARCHAR(20) NOT\_NULL,**  **PRIMARY KEY(‘Admin\_account’)**  **);** | |

**G. Product**

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **CREATE TABLE ‘Product’(**  **‘Name’ VARCHAR(20) NOT\_NULL,**  **‘Category’ INT(1) NOT\_NULL,**  **‘Company’ VARCHAR(20) NOT\_NULL,**  **‘Price’ INT(10) NOT\_NULL,**  **‘Video’ VARCHAR(80) NOT\_NULL,**  **PRIMARY KEY(‘Name’)**  **);** | |

**H. Review**

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **CREATE TABLE ‘Review’(**  **‘User\_account’ VARCHAR(20) NOT\_NULL,**  **‘Date’ DATETIME NOT\_NULL,**  **‘Title’ VARCHAR(20) NOT\_NULL,**  **‘Content’ VARCHAR(3000) NOT\_NULL,**  **‘File’ VARCHAR(300) NOT\_NULL,**  **‘Product’ VARCHAR(20) NOT\_NULL,**  **‘Rating\_Product’ INT(1) NOT\_NULL,**  **‘Rating\_Delivery’ INT(1) NOT\_NULL,**  **‘Rating\_Price’ INT(1) NOT\_NULL,**  **PRIMARY KEY(‘User\_account’, ‘Date’ ,’Title’),**  **FOREIGN KEY(‘User\_account’) REFERENCES ON User(‘User\_account’) ON DELETE NO ACTION,**  **FOREIGN KEY(‘Product’) REFERENCES ON Product(‘Name’) ON DELETE NO ACTION**  **);** | |

**I.Review\_comment**

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **CREATE TABLE ‘Review\_comment’(**  **‘Admin\_account’ VARCHAR(20) NOT\_NULL,**  **‘Review\_id’ INT(10) NOT\_NULL,**  **‘Content’ VARCHAR(1000) NOT\_NULL,**  **PRIMARY KEY( ‘Admin\_account’, ’Review\_id’, ’Content’),**  **FOREIGN KEY(‘Admin\_account’) REFERENCES ON Admin(‘Admin\_account’) ON DELETE NO ACTION,**  **FOREIGN KEY(‘Review\_id’) REFERENCES ON Review(‘Review\_id’) ON DELETE NO ACTION**  **);** | |

**J. Question**

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **CREATE TABLE ‘Question’(**  **‘User\_account’ VARCHAR(20) NOT\_NULL,**  **‘Date’ DATETIME NOT\_NULL,**  **‘Title’ VARCHAR(20) NOT\_NULL,**  **‘Content’ VARCHAR(3000) NOT\_NULL,**  **‘File’ VARCHAR(300) NOT\_NULL,**  **PRIMARY KEY(‘User\_account’, ‘Date’ ,’Title’),**  **FOREIGN KEY(‘User\_account’) REFERENCES ON User(‘User\_account’) ON DELETE NO ACTION**  **);** | |

**K. Question\_comment**

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **CREATE TABLE ‘Question\_comment’(**  **‘Admin\_account’ VARCHAR(20) NOT\_NULL,**  **‘Question\_id’ INT(10) NOT\_NULL,**  **‘Content’ VARCHAR(1000) NOT\_NULL,**  **‘Date’ DATETIME NOT\_NULL,**  **PRIMARY KEY( ‘Admin\_account’, ’Question\_id’, ’Content’),**  **FOREIGN KEY(‘Admin\_account’) REFERENCES ON Admin(‘Admin\_account’) ON DELETE NO ACTION,**  **FOREIGN KEY(‘Question\_id’) REFERENCES ON Question(‘Question\_id’) ON DELETE NO ACTION**  **);** | |

# **13. Testing Plan**

## **13.1 Objective**

Testing Plan은 시스템이 의도한 방향으로 실행되는지 확인하고 시스템 내부의 결함을 찾기 위해 testing을 설계하여 시행한다. 본 항목에서는 Test Policy 뿐만 아니라 여러 Test Case에 대하여 기술한다.

## **13.2 Testing Policy**

### A. Component Testing

Component test는 component 단위로 개발을 진행한 후, 각 component가 설계한 기능대로 작동하는지 확인하는 테스트 방법이다.

### B. System Testing

System test는 각 component를 점진적으로 통합하면서 진행하는 테스트 방법이다. System test에서는 통합된 시스템이 요구된 기능을 정상적으로 수행하는지 확인한다. Test case를 통하여 이를 검증한다. 이 과정에서 Emergent properties 역시 확인할 수 있다.

### C. Acceptance Testing

Acceptance test에서는 사용자의 실제 정보를 이용하여 시스템 요구사항이 지켜지는지 확인하는 방법이다.

## **13.3 Test Case**

1) 사용자는 웹 페이지에 접속한다.

2) 웹 페이지의 제목이 Chwimi를 가리킨다.

A. User Management System

A.1. Sign up

1) 사용자는 회원가입 페이지에 접속한다.

2) 사용자는 아이디 란에 아이디를 입력한다.

3) 사용자는 비밀번호 란에 비밀번호를 입력한다.

4) 사용자는 비밀번호확인 란에 비밀번호를 입력한다.

5) 사용자는 이메일 란에 이메일을 입력한다.

6) 사용자는 회원가입 버튼을 클릭한다.

6-1) 비밀번호와 비밀번호확인 입력 내용이 같지 않은 경우 에러 메시지를 출력한다.

7) 홈 페이지로 돌아간다.

A.2. Log in

1) 사용자는 로그인 페이지에 접속한다.

2) 사용자는 아이디 란에 아이디를 입력한다.

3) 사용자는 비밀번호 란에 비밀번호를 입력한다.

4) 사용자는 로그인 버튼을 클릭한다.

4-1) 아이디 또는 비밀번호가 데이터베이스와 일치하지 않으면 에러 메시지를 출력한다.

5) 홈 페이지로 돌아간다.

B. Notice System

1) 관리자는 로그인 페이지에 접속하여 로그인을 한다.

2) 관리자는 공지사항 페이지에 접속한다.

3) 관리자는 글쓰기 버튼을 클릭하고 제목과 내용을 입력한다.

4) 관리자는 작성 버튼을 클릭한다.

5) 공지사항 페이지로 돌아간다.

C. Hobby Test System

1) 사용자는 취미 테스트 페이지에 접속한다.

2) 사용자는 취미 테스트 시작 버튼을 클릭한다.

3) 사용자는 즐거웠던 취미 항목에 값을 입력한다.

4) 사용자는 성격 진단에 관련된 문항에 점수를 입력한다.

5) 사용자는 제출 버튼을 클릭한다.

6) 결과 페이지가 출력된다.

D. Subscribe Management System

1) 사용자는 구독 페이지에 접속한다.

2) 사용자는 취미 테스트 결과를 선택한다.

3) 사용자는 구독 개월 수를 선택한다.

4) 구독 버튼을 누르면 결제 모듈이 출력된다.

E. Review Management System

1) 사용자는 후기 페이지에 접속한다.

2) 사용자는 글쓰기 버튼을 클릭하고 제목과 내용을 입력한다.

3) 사용자는 이미지 파일을 첨부한다.

4) 사용자는 작성 버튼을 클릭한다.

5) 후기 페이지로 돌아간다.

6) 브라우저를 종료한다.

7) 관리자가 후기 페이지에 접속한다.

8) 후기 게시판 목록의 후기를 클릭한다.

9) 후기 상세 페이지 하단의 댓글 란에 내용을 입력한다.

10) 댓글 작성 버튼을 입력한다.

11) 후기 상세 페이지로 이동한다.

F. Question Management System

1) 사용자는 질문 게시판에 접속한다.

2) 사용자는 질문 버튼을 클릭한다.

3) 사용자는 제목과 내용을 입력한다.

4) 사용자는 작성 버튼을 클릭한다.

5) 질문 페이지로 돌아간다.

6) 브라우저를 종료한다.

7) 관리자가 질문 페이지에 접속한다.

8) 질문 게시판 목록의 질문을 클릭한다.

9) 상세 페이지 하단의 댓글 란에 내용을 입력한다.

10) 댓글 작성 버튼을 입력한다.

11) 질문 상세 페이지로 이동한다.

# **14. Development Environment**

## **14.1 Objective**

Development Environment에서는 개발자의 환경에 대해 설명한다. 사용한 프로그래밍 언어와 IDE, 패키지에 대해 서술한다.

## **14.2 Programming Languages & IDE**



Figure . Django with python

웹 서버 개발 언어로는 python 기반의 Django 웹 프레임워크를 사용한다. Django는 확장성이 좋아 다양한 패키지를 사용할 수 있으며 프로그래밍 언어 역시 직관적인 형태를 지니고 있기 때문에 효율적인 작업을 진행할 수 있다. 추천 시스템 서버에도 역시 python을 이용한다. 데이터베이스는 Django 내에서 기본적으로 제공하는 sqlite3를 사용한다.

웹 앱을 위한 안드로이드 언어는 java를 이용한다. 안드로이드 스튜디오는 안드로이드 개발을 위한 통합 환경을 제공하므로 이를 사용한다.



Figure . Android Studio

## **14.3 Coding Rule**

Chwimi 시스템에서 사용한 Coding Rule은 다음과 같다.

1. 개발 진행 상황을 파악하고 버전을 관리하기 위해 github을 통해 코드를 관리한다.
2. 기능 테스트와 단위 테스트를 활용하여 불필요한 코드를 최소화하고 코드의 안정성을 높인다.
3. Github을 이용하는 경우, 두 명 이상이 동시에 같은 파일을 작업하는 경우를 최소화하여 충돌이 나지 않도록 한다.

## **14.4 Version Management Tool**



Figure . Github

코드 관리와 공유의 편의를 위해 github을 사용하여 버전 관리를 진행한다. Github은 세계적으로 많이 이용되는 버전 관리 서비스이다. 많은 오픈 소스가 공유되어 있어 이를 활용하여 시스템에 적용할 수도 있다.

# **15. Develop Plan**

## **15.1 Objective**

Develop Plan에서는 개발 계획에 대해 서술한다. 이때 Gantt chart를 이용하여 개발 계획과 실제 개발 흐름에 대해 명시한다.

## **15.2 Gantt Chart**

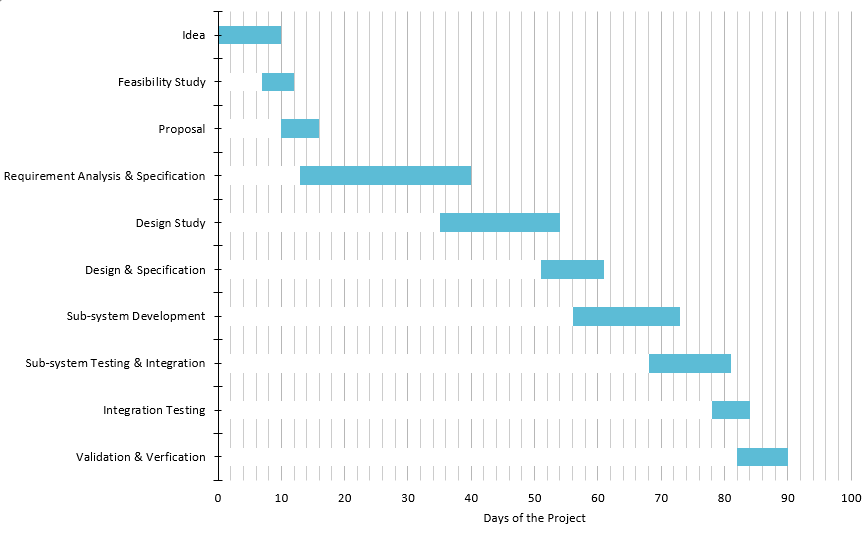


Figure . Gantt Chart

개발 계획과 실제 상황은 위의 Gantt chart와 같다. 기존에 계획했던 것보다 요구사항 명세서 작성에 많은 시간이 소요되었지만, 전반적인 진행 사항은 계획에서 크게 벗어나지 않았다.

시스템 전체 과정에서는 각각의 단계에 대한 문서화를 통해 요구사항과 설계 명세를 명확히 하려 노력하였다. 위의 정해진 일정 외에도 지속적으로 변경사항을 반영하도록 하고 이후 개발 일정 역시 계획에 맞게 최대한 진행할 수 있도록 노력할 예정이다.

# **16. Index**

## Figure

[Figure 1. UML Diagram Hierarchy 14](#_Toc9111757)

[Figure 2. Example of Package Diagram 15](#_Toc9111758)

[Figure 3. Example of Class Diagram 16](#_Toc9111759)

[Figure 4. Example of Deployment Diagram 17](#_Toc9111760)

[Figure 5. Example of ER Diagram 18](#_Toc9111761)

[Figure 6. Example of State Diagram 19](#_Toc9111762)

[Figure 7. Example of Sequence Diagram 20](#_Toc9111763)

[Figure 8. Logo of Draw.io 20](#_Toc9111764)

[Figure 9. Logo of Django 21](#_Toc9111765)

[Figure 10. Logo of Recombee 21](#_Toc9111766)

[Figure 11. User Management System 22](#_Toc9111767)

[Figure 12. About System 22](#_Toc9111768)

[Figure 13. Notice System 23](#_Toc9111769)

[Figure 14. Hobby Test System 23](#_Toc9111770)

[Figure 15. Subscribe Management System 24](#_Toc9111771)

[Figure 16. Review Management System 24](#_Toc9111772)

[Figure 17. Q&A Management System 25](#_Toc9111773)

[Figure 18. User Management System Architecture 26](#_Toc9111774)

[Figure 19. About System Architecture 27](#_Toc9111775)

[Figure 20. Notice System Architecture 27](#_Toc9111776)

[Figure 21. Hobby Test System Architecture 28](#_Toc9111777)

[Figure 22. Subscribe Management System Architecture 28](#_Toc9111778)

[Figure 23. Review System Architecture 29](#_Toc9111779)

[Figure 24. Question System Architecture 30](#_Toc9111780)

[Figure 25. Overall Package Diagram 31](#_Toc9111781)

[Figure 26. System Package Diagram 31](#_Toc9111782)

[Figure 27. Database Package Diagram 32](#_Toc9111783)

[Figure 28. Deployment Diagram 33](#_Toc9111784)

[Figure 29. User Management System Class Diagram 34](#_Toc9111785)

[Figure 30. Login (not using Kakao) System Sequence Diagram 37](#_Toc9111786)

[Figure 31. Login (Using Kakao) System Sequence Diagram 38](#_Toc9111787)

[Figure 32. MyPage System Sequence System 39](#_Toc9111788)

[*Figure 33. Signup State Diagram* 39](#_Toc9111789)

[Figure 34. Login System Sequence Diagram 40](#_Toc9111790)

[Figure 35. Mypage Sequence Diagram 40](#_Toc9111791)

[Figure 36. About System Class Diagram 41](#_Toc9111792)

[Figure 37. About System Sequence Diagram 42](#_Toc9111793)

[Figure 38. About System State Diagram 42](#_Toc9111794)

[Figure 39. Notice System Class Diagram 43](#_Toc9111795)

[Figure 40. Notice System (Admin Use) Sequence Diagram 44](#_Toc9111796)

[Figure 41. Notice System (User Use) Sequence Diagram 45](#_Toc9111797)

[Figure 42. Notice System(Admin Use) State Diagram 45](#_Toc9111798)

[Figure 43. Notice System(User use) State Diagram 45](#_Toc9111799)

[Figure 44. Hobby Test System Class Diagram 46](#_Toc9111800)

[Figure 45. Hobby Test System Sequence Diagram 48](#_Toc9111801)

[Figure 46. Hobby Test System State Diagram 49](#_Toc9111802)

[Figure 47. Subscribe System Class Diagram 50](#_Toc9111803)

[Figure 48. Video System Class Diagram 51](#_Toc9111804)

[Figure 49. Subscribe System Sequence Diagram 52](#_Toc9111805)

[Figure 50. Video System Sequence Diagram 53](#_Toc9111806)

[Figure 51. Subscribe System State Diagram 54](#_Toc9111807)

[Figure 52. Video System State Diagram 55](#_Toc9111808)

[Figure 53. Review Management System Class Diagram 56](#_Toc9111809)

[Figure 54. View Sequence Diagram 60](#_Toc9111810)

[Figure 55. Write Sequence Diagram 61](#_Toc9111811)

[Figure 56. Update Sequence Diagram 62](#_Toc9111812)

[Figure 57. Comment Sequence Diagram 63](#_Toc9111813)

[Figure 58. View State Diagram 64](#_Toc9111814)

[Figure 59. Write State Diagram 65](#_Toc9111815)

[Figure 60. Update State Diagram 65](#_Toc9111816)

[Figure 61. Comment State Diagram 66](#_Toc9111817)

[Figure 62. Question Management System Class Diagram 67](#_Toc9111818)

[Figure 63. View Sequence Diagram 70](#_Toc9111819)

[Figure 64. Write Sequence Diagram 71](#_Toc9111820)

[Figure 65. Update Sequence Diagram 72](#_Toc9111821)

[Figure 66. Comment Sequence Diagram 73](#_Toc9111822)

[Figure 67. View State Diagram 74](#_Toc9111823)

[Figure 68. Write State Diagram 75](#_Toc9111824)

[Figure 69. Update State Diagram 76](#_Toc9111825)

[Figure 70. Comment State Diagram 77](#_Toc9111826)

[Figure 71. Logo of JSON 78](#_Toc9111827)

[Figure 72. Overall ER Diagram 89](#_Toc9111828)

[Figure 73. User DB ER Diagram 90](#_Toc9111829)

[Figure 74. Admin DB ER Diagram 90](#_Toc9111830)

[Figure 75. Hobby DB ER Diagram 90](#_Toc9111831)

[Figure 76. Subscribe DB ER Diagram 91](#_Toc9111832)

[Figure 77. Product DB ER Diagram 91](#_Toc9111833)

[Figure 78. Question DB ER Diagram 91](#_Toc9111834)

[Figure 79. Review DB ER Diagram 92](#_Toc9111835)

[Figure 80. Comment Question DB ER Diagram 92](#_Toc9111836)

[Figure 81. Comment Review DB ER Diagram 92](#_Toc9111837)

[Figure 82. Kakao DB ER Diagram 93](#_Toc9111838)

[Figure 83. Notice DB ER Diagram 93](#_Toc9111839)

[Figure 84. Django with python 111](#_Toc9111840)

[Figure 85. Android Studio 112](#_Toc9111841)

[Figure 86. Github 112](#_Toc9111842)

[Figure 87. Gantt Chart 113](#_Toc9111843)

## Table

[Table 1. Version of the Document 13](#_Toc9013638)

[Table 2. Sign Up Protocol Request 79](#_Toc9013639)

[Table 3. Sign Up Protocol Response 79](#_Toc9013640)

[Table 4. Log In Protocol Request 79](#_Toc9013641)

[Table 5. Log In Protocol Response 79](#_Toc9013642)

[Table 6. My Page Protocol Request 80](#_Toc9013643)

[Table 7. My Page Protocol Response 80](#_Toc9013644)

[Table 8. About Protocol Request 80](#_Toc9013645)

[Table 9. About Protocol Response 81](#_Toc9013646)

[Table 10. Notice Protocol Request 81](#_Toc9013647)

[Table 11. Notice Protocol Response 81](#_Toc9013648)

[Table 12. Hobby Test Protocol Request 82](#_Toc9013649)

[Table 13. Hobby Test Protocol Response 82](#_Toc9013650)

[Table 14. Subscribe Protocol Request 82](#_Toc9013651)

[Table 15. Subscribe Protocol Response 83](#_Toc9013652)

[Table 16. Video Protocol Request 83](#_Toc9013653)

[Table 17. Video Protocol Response 83](#_Toc9013654)

[Table 18. Write/Update Review Protocol Request 84](#_Toc9013655)

[Table 19. Write/Update Review Protocol Response 84](#_Toc9013656)

[Table 20. View Review Protocol Request 84](#_Toc9013657)

[Table 21. View Review Protocol Response 85](#_Toc9013658)

[Table 22. Search Review Protocol Request 85](#_Toc9013659)

[Table 23. Search Review Protocol Response 85](#_Toc9013660)

[Table 24. Write/Update Question Protocol Request 86](#_Toc9013661)

[Table 25. Write/Update Question Protocol Response 86](#_Toc9013662)

[Table 26. View Question Protocl Request 86](#_Toc9013663)

[Table 27. View Question Protocol Response 87](#_Toc9013664)

[Table 28. Search Question Protocol Request 87](#_Toc9013665)

[Table 29. Search Question Protocol Response 87](#_Toc9013666)

[Table 30. Write/Update Comment Protocol Request 88](#_Toc9013667)

[Table 31. Write/Update Comment Protocol Response 88](#_Toc9013668)