**DESIGN SPECIFICATION**



**TEAM #2**

2014310355 김현승

2013312665 나상희

2016314771 김지명

2013314364 강성필

**목차**

내용

[1. Preface 4](#_Toc8916674)

[1.1. Objective 4](#_Toc8916675)

[1.2. Readership 4](#_Toc8916676)

[1.3. Document Structure 4](#_Toc8916677)

[A. Preface 4](#_Toc8916678)

[B. Introduction 4](#_Toc8916679)

[C. System Architecture 4](#_Toc8916680)

[D. User Management System 5](#_Toc8916681)

[E. Posting System 5](#_Toc8916682)

[F. AR 3D Modeling System 5](#_Toc8916683)

[G. Search System 5](#_Toc8916684)

[H. Protocol Design 5](#_Toc8916685)

[I. Database Design 5](#_Toc8916686)

[J. Testing Plan 5](#_Toc8916687)

[K. Development Environment 5](#_Toc8916688)

[L. Index 6](#_Toc8916689)

[M. Reference 6](#_Toc8916690)

[1.4 Version of the document 6](#_Toc8916691)

[A. Version format 6](#_Toc8916692)

[B. Version management 6](#_Toc8916693)

[C. Version update history 6](#_Toc8916694)

[2. Introduction 6](#_Toc8916695)

[2.1. Objectives 6](#_Toc8916696)

[2.2. Applied Diagram 7](#_Toc8916697)

[A. UML 7](#_Toc8916698)

[B. Package Diagram 8](#_Toc8916699)

[C. Class Diagram 8](#_Toc8916700)

[D. State Diagram 9](#_Toc8916701)

[E. Sequence Diagram 10](#_Toc8916702)

[F. ER Diagram 11](#_Toc8916703)

[2.3. Applied Tool 12](#_Toc8916704)

[A. Google Vision 12](#_Toc8916705)

[B. ARcore 12](#_Toc8916706)

[C. Android Studio 13](#_Toc8916707)

[D. NodeJS 13](#_Toc8916708)

[E. MYSQL 14](#_Toc8916709)

[3. System Architecture 15](#_Toc8916710)

[3.1. Objectives 15](#_Toc8916711)

[3.2. System Organization 15](#_Toc8916712)

[A. User Management System 15](#_Toc8916713)

[B. Posting System 15](#_Toc8916714)

[C. AR 3D Modeling System 15](#_Toc8916715)

[D. Search System 15](#_Toc8916716)

[3.3. System Use Case Diagram 15](#_Toc8916717)

[3.4. Package Diagram 15](#_Toc8916718)

[4. User Management System 15](#_Toc8916719)

[4.1. Objectives 15](#_Toc8916720)

[4.2. Class Diagram 15](#_Toc8916721)

[A. User 15](#_Toc8916722)

[B. DB handler 15](#_Toc8916723)

[4.3. Sequence Diagram 15](#_Toc8916724)

[A. Select game within the list 15](#_Toc8916725)

[B. Search for games failed 15](#_Toc8916726)

[C. Search for games successful 15](#_Toc8916727)

[4.4. State Diagram 15](#_Toc8916728)

[5. Posting System 15](#_Toc8916729)

[5.1. Objectives 15](#_Toc8916730)

[5.2. Class Diagram 15](#_Toc8916731)

[A. Extractor 15](#_Toc8916732)

[B. Sender 15](#_Toc8916733)

[5.3. Sequence Diagram 15](#_Toc8916734)

[6. AR 3D Modeling System 15](#_Toc8916735)

[6.1. Objectives 16](#_Toc8916736)

[6.2. Class Diagram 16](#_Toc8916737)

[A. Sender 16](#_Toc8916738)

[B. PC Specification Compare System 16](#_Toc8916739)

[C. Database 16](#_Toc8916740)

[6.3. Sequence Diagram 16](#_Toc8916741)

[7. Search System 16](#_Toc8916742)

[7.1. Objectives 16](#_Toc8916743)

[7.2. Class Diagram 16](#_Toc8916744)

[A. DB Handler 16](#_Toc8916745)

[B. Search 16](#_Toc8916746)

[7.3. Sequence Diagram 16](#_Toc8916747)

[7.4. State Diagram 16](#_Toc8916748)

[7.5. Search Filter 16](#_Toc8916749)

[8. Protocol Design 16](#_Toc8916750)

[8.1. Objectives 16](#_Toc8916751)

[8.2. Protocol Description 16](#_Toc8916752)

[A. ?? protocol 16](#_Toc8916753)

[B. ?? Protocol 16](#_Toc8916754)

[C. ?? protocol 16](#_Toc8916755)

[9. Database Design 16](#_Toc8916756)

[9.1. Objective 16](#_Toc8916757)

[9.2. ER Diagram 16](#_Toc8916758)

[A. Entity 16](#_Toc8916759)

[B. Relationship 16](#_Toc8916760)

[9.3. Relational Schema 16](#_Toc8916761)

[A. table1 16](#_Toc8916762)

[B. 2 17](#_Toc8916763)

[C. 3 17](#_Toc8916764)

[D. 4 17](#_Toc8916765)

[10. Testing Plan 17](#_Toc8916766)

[10.1. Objectives 17](#_Toc8916767)

[10.2. Testing Policy 17](#_Toc8916768)

[10.3. Test Case 17](#_Toc8916769)

[A. User Management System 17](#_Toc8916770)

[B. Posting System 17](#_Toc8916771)

[C. AR 3D Modeling System 17](#_Toc8916772)

[D. Search System 17](#_Toc8916773)

[11. Development Environment 17](#_Toc8916774)

[11.1. Objectives 17](#_Toc8916775)

[11.2 Programming Language & IDE 17](#_Toc8916776)

[A. Programming Language 17](#_Toc8916777)

[12. Index 17](#_Toc8916778)

[12.1. Table Index 17](#_Toc8916779)

[12.2. Figure Index 17](#_Toc8916780)

[12.3. Diagram Index 17](#_Toc8916781)

[13. Reference 17](#_Toc8916782)

# Preface

## Objective

Preface 에서는 본 문서의 예상되는 독자들과 문서의 전반적인 구조, 각 부분의 역할에 대해 제시한다. 또한 버전 관리, 버전 변경 기록, 문서의 변경사항들과 근거에 대해 서술한다.

## Readership

Design specification은 시스템을 개발하는 개발자 및 유지 보수를 담당하는 사람들을 재상으로 작성되었다. 시스템 개발을 담당하는 소프트웨어 엔지니어, 시스템을 설계하는 시스템 설계자 등 모두가 Design specification의 독자로 설정되어 있다.

## Document Structure

### Preface

Preface에서는 문서의 예상되는 독자들과 문서의 전반적인 구조, 각 부분의 역할을 제시한다. 또한 버전 관리, 버전 변경 기럭, 문서의 변경사항들과 근거에 대해 제시한다.

### Introduction

Introduction단계에서는 시스템을 설계하는데 사용되는 다양한 Diagram들과 UML 그리고 적용된 Tool들을 소개한다.

### System Architecture

System Architecture단계에서는 개발하려는 시스템의 구조를 설명하고 각 sub-system에 대해서 기술한다. 또한 사용자들의 Use case를 기술한다.

### User Management System

사용자들을 관리하는 시스템에 대한 설명을 기술한다. 사용자들의 회원가입, 로그인, 프로필 관리, 사용자의 게시물 관리 등을 포함한다.

### Posting System

사용자가 게시글을 Posting하는 시스템에 대한 설명을 기술한다. 사용자가 게시하는 글을 데이터베이스에 저장, 사용자가 입력한 사진을 바탕으로 Tag 추천 등의 과정을 포함한다.

### AR 3D Modeling System

사용자가 입력한 사이즈를 바탕으로 3D모델을 만들고 AR을 통해 실제 사이즈를 체감할 수 있도록 카메라를 통해 보여주는 과정 등을 포함한다.

### Search System

사용자의 프로필 정보, 또는 사전에 입력한 관심 Tag를 바탕으로 사용자의 화면에 관련 게시글을 올려주는 과정, 사용자가 입력한 검색어 또는 Tag를 바탕으로 관련 게시글을 올려주는 과정 등을 포함한다.

### Protocol Design

시스템안에서 Sub-System들이 어떠한 방식으로 정보를 주고받는지 설정하고 통신하는 세부 내용에 대해 기술한다.

### Database Design

Database Design에서는 Requirement Specification을 기반으로 한 데이터베이스를 설계하고 설계한 데이터베이스르 설명한다. 설계된 데이터베이스릐 ER Diagram을 제시하고 이에 대한 Relation Schema를 서술한다.

### Testing Plan

Testing Plan에서는 예상되는 시나리오를 바탕으로 한 전체 시스템의 Test case를 설명하고 관련 Testing을 제시한다.

### Development Environment

시스템 개발을 위해 사용된 개발 환경과 언어를 기술한다.

### Index

Diagram과 Table등의 인덱스를 정리한다.

### Reference

문서를 작성하는데 참조된 문서들을 기술한다.

## 1.4 Version of the document

### A. Version format

버전 번호는 0.1을 시작으로 major.minor[.maintenance]로 구성된다.

### B. Version management

Requirement specification을 수정할 때 마다 버전을 업데이트 한다. 새로운 부분을 추가하거나 구성 순서를 바꾸는 등의 큰 변화가 있을 경우 major number를 변경한다. 기존에 완성된 파트에 사양을 추가하거나 변경을 가하는 경우 minor number를 변경한다. 단순 오탈자를 수정하거나 문서 디자인을 보기 좋게 수정할 때엔 maintenance 번호를 추가하여 변경한다.

### C. Version update history

|  |  |  |
| --- | --- | --- |
| Version | Modified Date | Modified Contents |
| 0.1 | 2019-05-15 | 목차 생성, Preface 작성 |
| 1.0 | 2019-05-16 | Introduction 작성 |

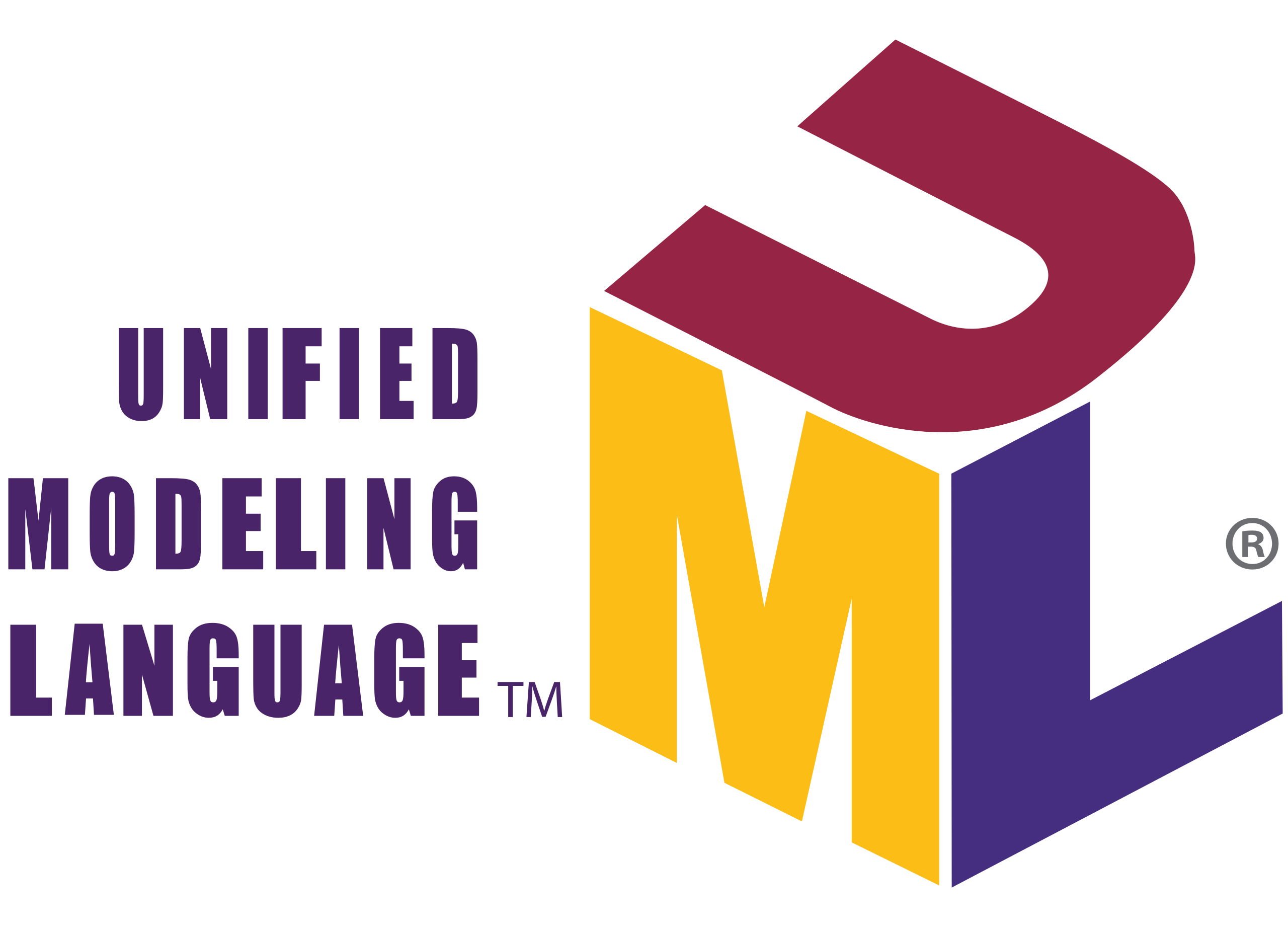
# Introduction

## Objectives

본 문서에서 사용되는 다이어그램이나 시스템을 설계하는데 사용되는 툴들에 대한 설명을 기술한다.

## Applied Diagram

### UML

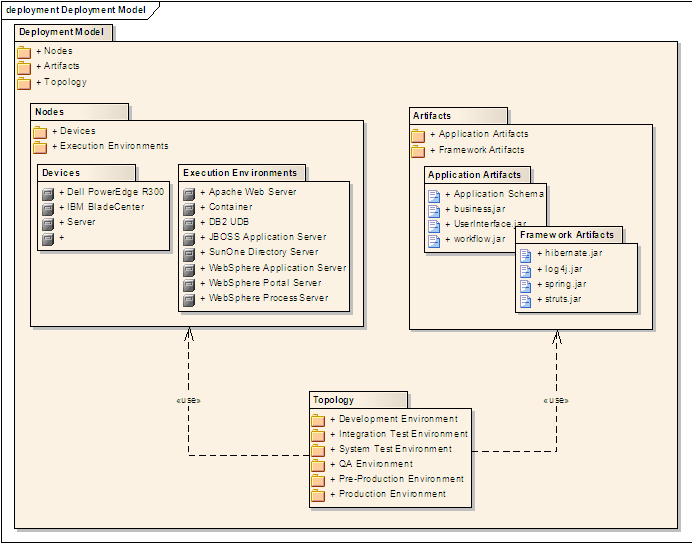


UML(Unified Modeling Language, 통합 모델링 언어)이란 OMG(객체 관련 표준화 기구)에서 객체 모델링 기술(OMT), OOSE방법론등을 연합하여 만든 통합 모델링 언어로 객체 지향적 분석 ·설계 방법론의 표준 지정을 목표로 하고 있다. UML은 총 14가지의 다이어그램을 포함하며 각각 7개씩 구조 다이어그램, 행위 다이어그램 2가지로 분류된다. 구조 다이어그램은 Class Diagram, Component Diagram, Composite Structure Diagram, Deployment Diagram, Object Diagram, Package Diagram, and Profile Diagram을 포함하며 시스템의 구조적인 것들을 다이어그램으로 표현하여 이해를 돕는다. 행위 다이어그램은 Activity Diagram, Communication Diagram, Interaction Overview Diagram, Sequence Diagram, State Diagram, Timing Diagram, and Use Case Diagram을 포함하며 구성요소들, 또는 요소들 간의 변화나 흐름, 주고받는 데이터 등의 동작을 표현한다.

UML을 이용하면 요구 분석, 시스템 설계, 구현 등의 과정에서 생길 수 있는 개발자 사이의 의사소통의 문제를 해소할 수 있다. 또한 모델링에 대한 표현력이 강하고 비교적 모순이 적은 논리적인 표기법을 가진 언어이기 때문에 개발자 간의 의사소통이 쉬워지며 생략되거나 불일치 되는 모델링 구조에 대한 지적도 용이하고, 개발하려는 시스템의 규모에 상관없이 모두 적용 가능하다.

본 문서에서 사용될 다이어그램으로는 Package Diagram, Class Diagram, State Diagram, State Diagram, Sequence Diagram 등이 있다.

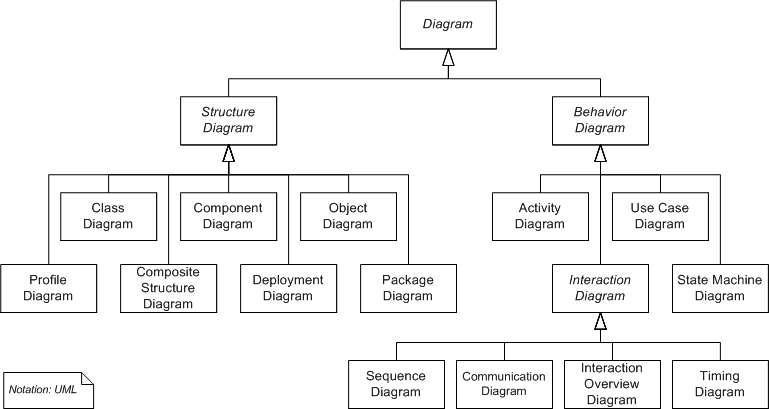
### Package Diagram



Package Diagram은 UML의 구조 다이어그램 중 하나로 Class와 같은 여러 모델 요소들을 그룹화하고, 패키지를 구성하여 패키지 사이들의 관계를 표현한 diagram이다. 패키지는 class의 집합으로 볼 수 있는데, use case, activity diagram 등을 담을 수도 있다. 또한 패키지가 다른 패키지를 담을 수 있다.

Package를 구성하고 구분하는 기분은 여러 사람이 동의할 수 있는 형태로 구성되어야 하고, Package의 구성과 이름, 체계는 개발자들이 쉽게 이해할 수 있게 해야 한다. 같은 package로 분류되는 class들은 개념적, 관리적, 기능적 측면에서 유사한 면을 가져야 한다.

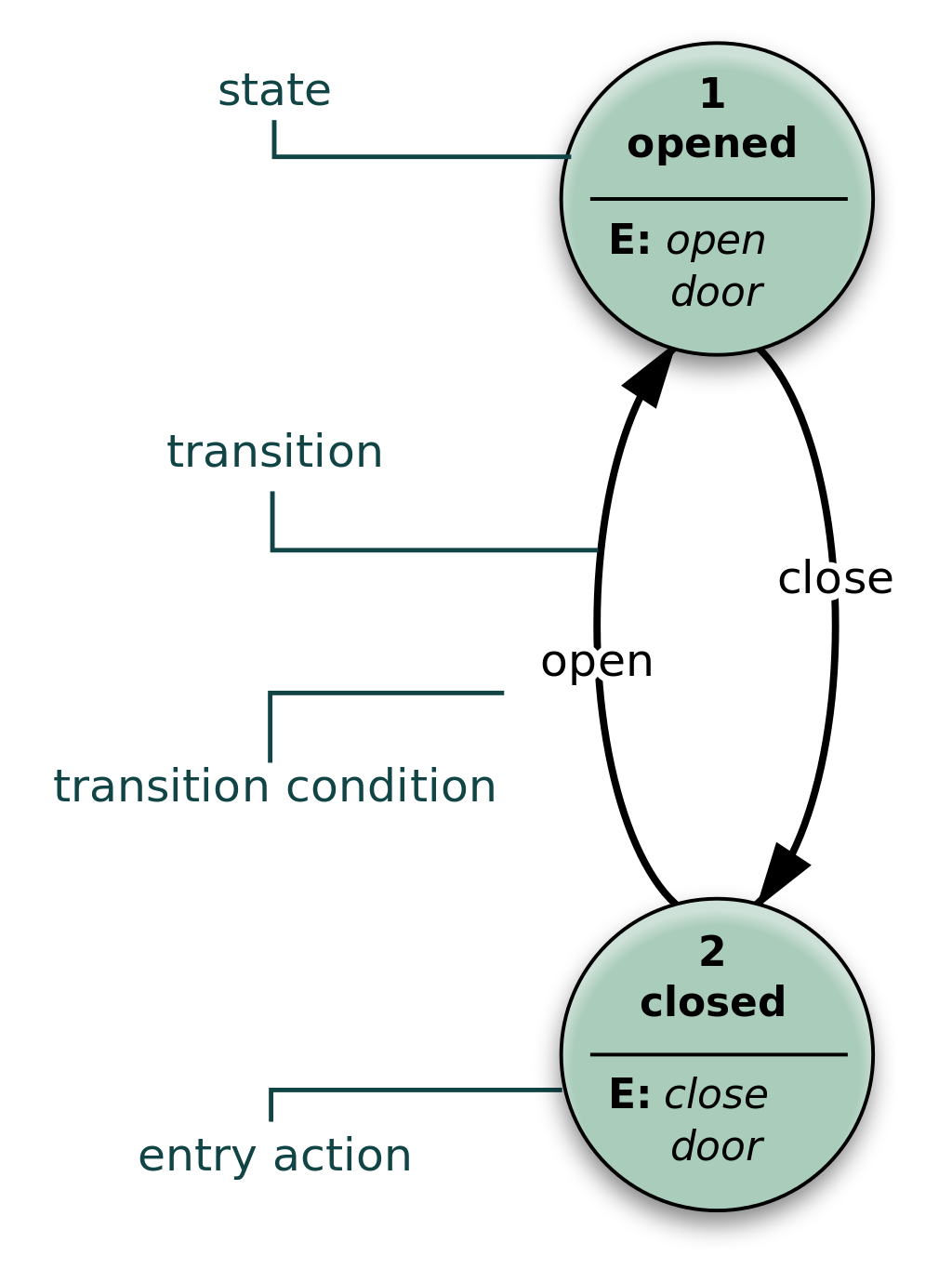
### Class Diagram



Class Diagram은 UML의 구조 다이어그램 중 하나로 시스템의 클래스, 속성, 작업 및 객체 간의 관계를 보여주어 시스템의 구조를 설명하는 다이어그램이다. 시스템의 일부 또는 전체의 구조를 나타낼 수 있고, 클래스들 간의 관계와 의존관계를 쉽고 명확하게 파악할 수 있게 한다.

Class Diagram의 Class는 주요 요소, 응용 프로그램의 상호 작용 및 프로그래밍 할 클래스를 모두 나타낸다. Class Diagram은 시스템 내부에 존재하는 클래스들을 선별하여 나타내고 각 클래스의 속성과 행위를 표현한다. Class들 사이에 여러가지 관계를 가질 수 있으며 상속관계, 의존관계가 나타날 수도 있다. Class Diagram을 그릴 때, 항상 추상화 단계를 고려하여 그려 클래스 간 인터페이스를 빠른 단계에서 명확하게 할 수 있다.

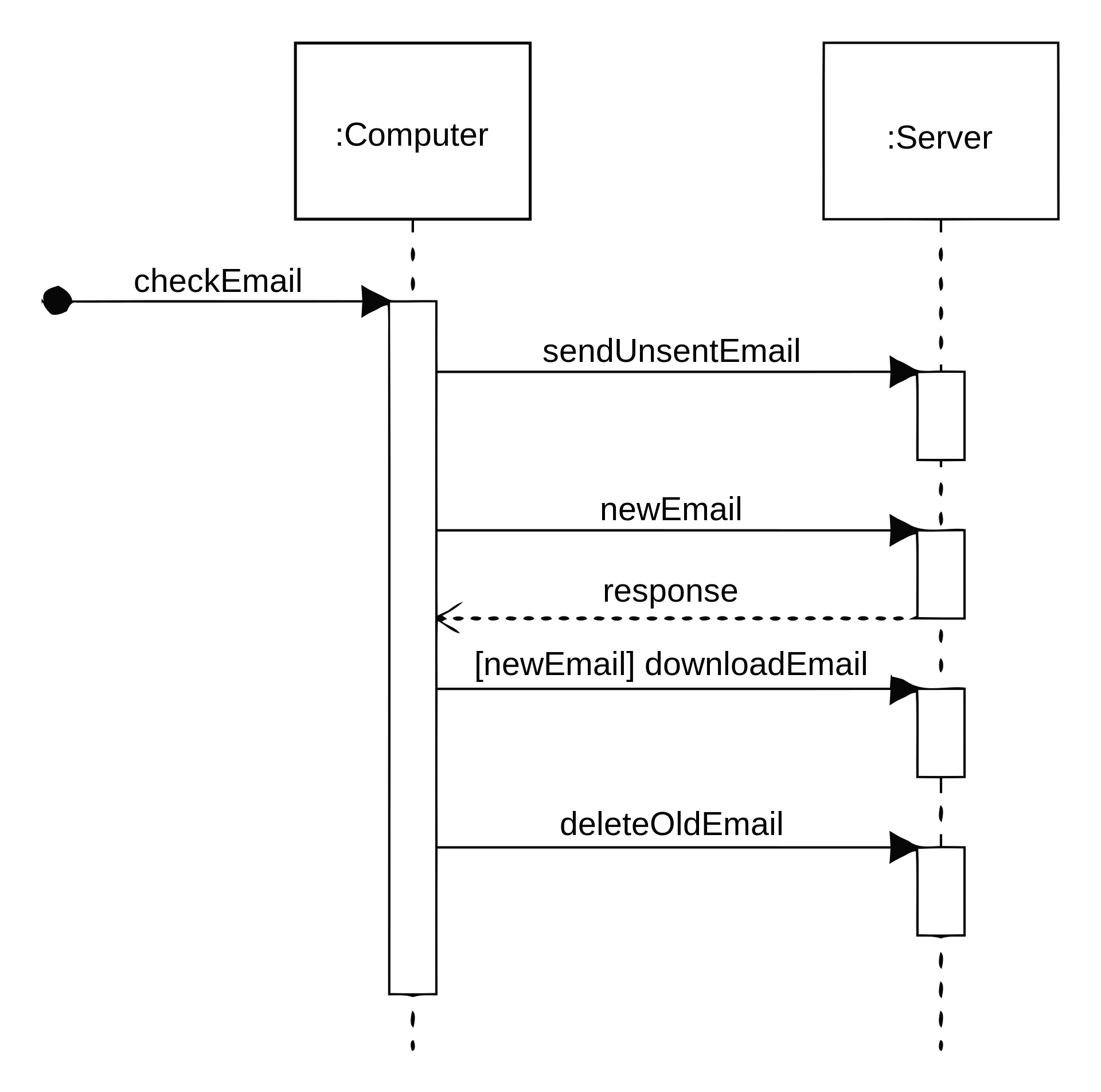
### State Diagram



State diagram은 UML의 행위 다이어그램 중 하나로 한 객체의 상태와 발생한 event로 인한 state 변화를 다이어그램으로 나타낸 것이다. 시스템, 혹은 객체에 대해서 event에 따른 반응과 생명주기 동안의 상태 변화를 모델링한 다이어그램으로 주로 하나의 Class의 내부의 구성요소나 객체가 외부 event에 반응하는 동적 상황을 기술하기 위한 것이기 때문에, 동적 모델로 분류된다.

State Diagram을 통해 event에 대한 객체의 상태변화를 분석할 수 있고, 객체가 가질 수 있는 상태의 종류나 수에 대한 정보를 얻을 수 있다. 또한 객체의 속성과 수행해야 하는 동작을 정의하고 검증할 수 있다.

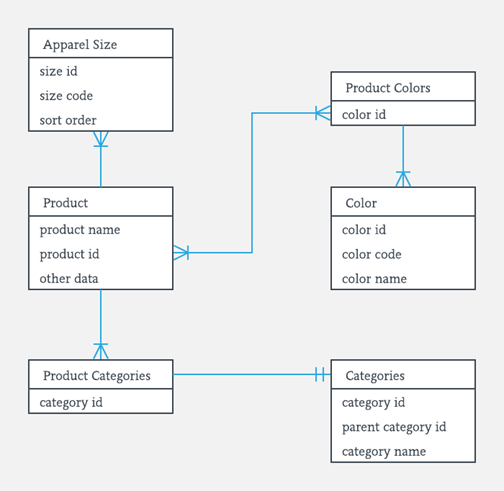
### Sequence Diagram



Sequence Diagram은 UML의 동적 diagram중 하나로 시스템 내의 객체 간의 상호작용을 시간 순으로 표현한 다이어그램이다. 시나리오로부터 object, class, sequence를 추출하여 객체 간 교환되는 메시지 시퀀스를 표현한다.

Sequence Diagram은 일반적으로 개발중인 시스템의 논리적 관점에서 Use case의 실현과 관련된다.

### ER Diagram



데이터베이스에서 각 객체들의 관계를 표현하기 위해 사용하는 다이어그램이다. 앞에 다이어그램들과는 다르게 UML에 속하지 않는다. 데이터 베이스에서 저장되는 데이터의 구조 및 관계를 표현해서 그릴 수 있는데 그러한 방법으로써, entity – relationship(ER)이다. ER에서 Entity는 현실 세계의 객체로써 유형 또는 무형 정보를 대상으로 한다. 따라서 entity이기 위해서는 attribute를 가져야한다. Relationship은 두개 이상의 entity 사이의 연관성을 기술한 것이다.

* 1. Applied Tool

### Google Vision



구글에서 제공하는 이미지 분석 API이다. 이미지를 넣으면 해당 이미지와 관련된 Class, Tag등을 추출해준다. 이 API를 이용하여 판매자가 판매하고자 하는 판매 상품의 사진을 보고 TAG를 추천해 주는 기능을 넣을 수 있다.

### ARcore



AR을 개발할 수 있도록 도와주는 AR 개발 툴이다. ARcore를 통해 개발한 시스템을 이용하여 판매자가 입력한 제품 사이즈를 바탕으로 서비스 사용자들이 실제 제품의 크기를 체감할 수 있도록 할 것이다

### Android Studio



안드로이드 어플을 만들 수 있는 개발 툴이다. 안드로이드 기반 어플을 만들 기 때문에 안드로이드 스튜디오를 통해 Front-End, user interface를 만들 것이다.

### NodeJS



서비스와 서버를 연결하는 작업을 도와주는 개발 툴이다. 지속적으로 데이터베이스와의 연동은 물론 구글비전 등 서버와 지속적인 연동이 필요한 서비스이기 때문에 NodeJS를 사용하여 Back- End를 구축할 것이다.

### MYSQL



데이터베이스 관리를 도와주는 개발 툴이다. 게시글, 사용자의 정보 등 여러가지의 데이터가 들어오기 때문에 데이터베이스의 확실한 관리가 필수 적이다. MYSQL를 사용하여 Back-End를 구축할 것이다.

# 3. System Architecture

## 3.1. Objectives

## 3.2. System Organization

### A. User Management System

### B. Posting System

### C. AR 3D Modeling System

### D. Search System

## 3.3. System Use Case Diagram

## 3.4. Package Diagram

# 4. User Management System

## 4.1. Objectives

## 4.2. Class Diagram

### A. User

### B. DB handler

## 4.3. Sequence Diagram

### A. Select game within the list

### B. Search for games failed

### C. Search for games successful

## 4.4. State Diagram

# 5. Posting System

## 5.1. Objectives

## 5.2. Class Diagram

### A. Extractor

### B. Sender

## 5.3. Sequence Diagram

# 6. AR 3D Modeling System

## 6.1. Objectives

## 6.2. Class Diagram

### A. Sender

### B. PC Specification Compare System

### C. Database

## 6.3. Sequence Diagram

# 7. Search System

## 7.1. Objectives

## 7.2. Class Diagram

### A. DB Handler

### B. Search

## 7.3. Sequence Diagram

## 7.4. State Diagram

## 7.5. Search Filter

# 8. Protocol Design

## 8.1. Objectives

본 항목에서는 AR기반 중고거래 플랫폼인 ‘Shark’에서, 전체 시스템을 이루는 Subsystem간의 데이터 교환과 같은 communication에 있어서 필요한 데이터 교환 형태, 즉 프로토콜(Protocol)에 대해서 기술하고자 한다.

이 프로그램의 두 subsystem 으로는 사용자가 직접 이용하는 안드로이드 어플리케이션(프론트엔드)와 데이터를 저장하고 불러오는 서버(백엔드)로 나눌 수 있는데, 웹 통신을 기반으로 작동하게 되므로 HTTP Request와 Response를 통해 작동하게 된다고 할 수 있다. 또한 이를 통해 데이터 형식의 경우 JSON을 사용하게 된다.

따라서 본 항목에서는 HTTP Request와, 이에 대한 응답의 형태인 JSON의 특징과 용도에 대해 간단히 설명한다. 또한 Request Type별로 존재하는 요청의 종류와 형식, 반환 서식까지 함께 설명한다.

## 8.2 HTTP Request



웹 기반 프로그램에서, 서버와 클라이언트의 통신은 HTTP 요청을 통해 이루어진다. 본 시스템은 AR 기능의 탑재를 위해 안드로이드 기반의 프론트엔드를 개발하게 되지만, 시스템 자체는 웹 페이지와 똑같이 동작하게 되므로 데이터베이스가 탑재된 웹 서버를 개발하게 되고, 따라서 웹 서버와의 통신 또한 HTTP Request를 따르게 된다.

HTTP Request는 주어진 리소스에 적절한 요청이 수행되게 하는 요청 메소드의 집합을 정의한다. 이러한 Request는 대표적으로 다음과 같은 세부 항목을 포함하게 된다.

* GET: 서버로부터 데이터 수신
* POST: 서버에 특정 데이터 entry 추가
* PUT: 서버의 특정 데이터 entry 변경
* DELETE: 서버의 특정 데이터 entry 삭제

## 8.3. Protocol Description

### A. Authentication protocol

A.1. Login

POST: /login

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| username | string | 로그인에 사용할 유저의 id |
| password | string | 로그인에 사용할 비밀번호 |

A.2. Logout

POST: /logout

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| 변수 없음 | - | 현재 세션의 ID를 로그아웃 처리 |

### B. User protocol

B.1. Get User Info

GET: /user/:user\_id

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| user\_id | int | 가져올 유저의 key값을 전달 |

출력 결과

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| username | string | 검색한 유저 id |
| password | string | 해당 유저의 비밀번호 |
| nickname | string | 해당 유저의 닉네임 |
| address | string | 해당 유저의 거주지 |
| score | int | 해당 유저의 신뢰도 점수 |
| interest | string | 해당 유저의 주요 관심 품목 |

B.2. Add User Info

POST: /user

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| username | string | 생성할 유저의 id |
| password | string | 생성할 유저의 password |

B.3. Update User Info

PUT: /user/:user\_id

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| password | string | 생성할 유저의 password |
| nickname | string | 생성할 유저의 닉네임 |
| address | string | 생성할 유저의 거주지 |
| interest | string | 생성할 유저의 주요 관심 품목 |

B.4. Delete User Info

DELETE: /user/:user\_id

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| 변수 없음 | - | 삭제할 유저의 key |

C. Post Protocol

C.1. Get Post Info

GET: /post/:post\_id

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| post\_id | int | 정보를 가져올 포스트의 key  미입력시 아래의 parameter를 이용해  특정 range의 post를 검색 |
| limit | int | 검색할 post의 최대 개수 |
| keyword | string | 검색에 사용할 keyword |

요청 결과(List)

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| user\_id | int | 포스트를 작성한 작성자의 key |
| title | string | 포스트의 제목 |
| keyword | string | 포스트의 연관 keyword(해시태그) |
| contents | string | 포스트의 내용 |
| image | string | 포스트에 업로드된 이미지의 파일명 |
| ar\_category | string | 포스트를 보여줄 때 사용할 ar 카테고리 |
| ar\_width | double | ar 모델의 너비 |
| ar\_height | double | ar 모델의 높이 |
| ar\_depth | double | ar 모델의 깊이 |

C.2. Add Post Info

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| user\_id | int | 포스트를 작성한 작성자의 key |
| title | string | 포스트의 제목 |
| keyword | string | 포스트의 연관 keyword(해시태그) |
| contents | string | 포스트의 내용 |
| image\_file | File | 포스트에 업로드할 이미지 파일 |

C.3. Modify Post Info

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| user\_id | int | 포스트를 작성한 작성자의 key |
| title | string | 포스트의 제목 |
| keyword | string | 포스트의 연관 keyword(해시태그) |
| contents | string | 포스트의 내용 |
| image\_file | File | 포스트에 업로드할 이미지 파일 |

C.4. Delete Post Info

DELETE: /post/:post\_id

|  |  |  |
| --- | --- | --- |
| 요청 변수명 | 타입 | 설명 |
| post\_id | int | 삭제할 포스트의 key |

# 9. Database Design

## 9.1. Objective

본 항목에서는 요구사항 명세서에서 기술한 데이터베이스 요구사항을 기반으로 하여, 앞으로 구축하게 될 데이터베이스 다이어그램을 작성한다. 디자인한 데이터베이스는 ER 다이어그램으로 표현되며, 작성한 ER 다이어그램을 통해 프로그램의 데이터가 구조화되어 서버에 저장된다.

## 9.2. Table

본 항목에서는 데이터베이스를 이루는 테이블 별 Field에 대해 기술한다.

### A. User

|  |  |  |
| --- | --- | --- |
| 필드명 | 타입 | 설명 |
| user\_id | int | 유저의 key |
| username | string | 아이디 |
| user\_id | int | 포스트를 작성한 작성자의 key |
| title | string | 포스트의 제목 |
| keyword | string | 포스트의 연관 keyword(해시태그) |
| contents | string | 포스트의 내용 |
| image | string | 포스트에 업로드된 이미지의 파일명 |
| ar\_category | string | 포스트를 보여줄 때 사용할 ar 카테고리 |

### B. Post

|  |  |  |
| --- | --- | --- |
| 필드명 | 타입 | 설명 |
| post\_id | int | 포스트의 key |
| title | string | 제목 |
| contents | string | 내용 |
| keyword | string | 키워드 |
| image\_name | string | 사진 파일 이름 |
|  |  |  |

C. Model

|  |  |  |
| --- | --- | --- |
| 필드명 | 타입 | 설명 |
| ar\_id | int | AR 모델의 key |
| model\_name | string | 3D 모델 파일 이름 |
| category | string | 해당 모델이 속하는 카테고리 이름 |

## 9.3. Relational Schema

본 항목에서는, 9.2에서 기술한 Table 간의 상호 관계에 대해 기술하고, 기술한 테이블 간 관계를 SQL로 옮기는 작업을 수행한다.

### Join Type

A.1 Write Post

중고 상품의 판매 등록을 위해, 관련된 정보를 게시하는 란이 필요하며, 이렇게 올라가는 글의 단위를 포스트라고 한다. 포스트가 작성되기 위해서는 포스트를 작성하는 주체가 명시되어야 하며, 이를 일반적으로 작성자 또는 글쓴이라고 칭한다.

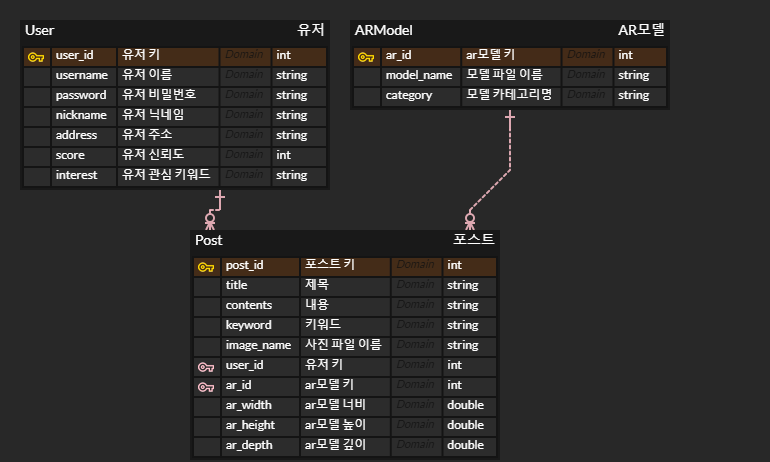
작성자와 포스트는 서로 다른 테이블에 위치해 있으므로, 이를 Table Join을 이용해 한 번에 호출할 수 있어야 한다. 모든 포스트는 누가 작성했는지에 대한 정보를 가지고 있어야 하므로 user에 대한 외부 참조 키를 Post 테이블에 추가하여, 모든 포스트에 대하여 작성한 유저의 정보를 Join하여 함께 확인할 수 있도록 한다.

A.2. AR Modeling

작성한 포스트의 정보를 이용해 모델을 AR을 통해 보여주기 위해서는, 저장된 모델의 정보를 포스트와 함께 보여줄 수 있어야 한다. 상품의 직접적인 정보이므로 포스트에 함께 등록되는 상품의 사이즈(높이, 너비 등)를, 어떠한 상품인지에 관한 모델 정보와 함께 보여줄 수 있어야 하므로 이 부분 역시 Table Join을 통해 한 번에 조회할 수 있어야 한다.

따라서 Model의 정보를 사전에 저장해 놓는 테이블을 하나 두고, 이를 Post 테이블에서 참조하는 방식을 통해 해당하는 모델의 파일 정보를 쉽게 검색할 수 있도록 한다.

1. ER Diagram



9.2의 Table과, 9.3의 Modeling을 통해 실제 Table을 표현하고, 각 Table의 관계를 나타낸 그림은 위와 같다. 확인할 수 있는 것처럼, User 테이블과 Post 테이블의 관계가 비식별 1:N 관계로, Post와 ARModel 테이블의 관계가 비식별 N:1 관계로 나타나고 있다.

### B. SQL Query

위의 ER Diagram을 참조해, SQL을 작성한 결과는 다음과 같다.

B.1. User Table

CREATE TABLE `유저` (

`유저 키` int NOT NULL,

`유저 이름` string NOT NULL,

`유저 비밀번호` string NOT NULL,

`유저 닉네임` string NULL,

`유저 주소` string NULL,

`유저 신뢰도` int NULL,

`유저 관심 키워드` string NULL

);

B.2. Post Table

CREATE TABLE `포스트` (

`포스트 키` int NOT NULL,

`제목` string NOT NULL,

`내용` string NOT NULL,

`키워드` string NULL,

`사진 파일 이름` string NULL,

`유저 키` int NOT NULL,

`ar모델 키` int NULL,

`ar모델 너비` double NULL,

`ar모델 높이` double NULL,

`ar모델 깊이` double NULL

);

B.3. ARModel Table

CREATE TABLE `AR모델` (

`ar모델 키` int NOT NULL,

`모델 파일 이름` string NULL,

`모델 카테고리명` string NULL

);

B.4. Relations

ALTER TABLE `유저` ADD CONSTRAINT `PK\_유저` PRIMARY KEY (

`유저 키`

);

ALTER TABLE `포스트` ADD CONSTRAINT `PK\_포스트` PRIMARY KEY (

`포스트 키`

);

ALTER TABLE `AR모델` ADD CONSTRAINT `PK\_AR모델` PRIMARY KEY (

`ar모델 키`

);

# 10. Testing Plan

## 10.1. Objectives

본 항목에서는 시스템 완성 후 시스템의 결함 여부를 파악하고 분석하기 위해, 시스템이 의도된 대로 작동하는지를 파악하는 시스템 테스팅에 대한 사전 계획을 구성한다. 이러한 Testing Plan 작업을, Testing에 대한 기준을 마련하는 Testing Policy와, 보다 상세한 input/output의 셋을 구성하는 Test Case로 세분화하여 설명한다.

## 10.2. Testing Policy

## 10.3. Test Case

### A. User Management System

### B. Posting System

### C. AR 3D Modeling System

### D. Search System

# 11. Development Environment

## 11.1. Objectives

## 11.2 Programming Language & IDE

### A. Programming Language

# 12. Index

## 12.1. Table Index

## 12.2. Figure Index

## 12.3. Diagram Index

# 13. Reference