Dealistic

Design Specification



|  |  |
| --- | --- |
| Student Number | Name |
| 2014312794 | 김준현 (Junhyun Kim) |
|  |  |
|  |  |
|  |  |
|  |  |

Contents

[Preface 7](#_Toc9315820)

[Objectives 7](#_Toc9315821)

[Readership 7](#_Toc9315822)

[Document Structure 7](#_Toc9315823)

[Introduction 8](#_Toc9315824)

[Objectives 8](#_Toc9315825)

[Applied Diagram 8](#_Toc9315826)

[UML 8](#_Toc9315827)

[Package Diagram 8](#_Toc9315828)

[Deployment Diagram 9](#_Toc9315829)

[Class Diagram 9](#_Toc9315830)

[State Diagram 9](#_Toc9315831)

[Sequence Diagram 10](#_Toc9315832)

[ER Diagram 11](#_Toc9315833)

[Applied Tool 11](#_Toc9315834)

[Draw.io 11](#_Toc9315835)

[PowerPoint 12](#_Toc9315836)

[ERDPlus 12](#_Toc9315837)

[Project Scope 12](#_Toc9315838)

[System Architecture – Overall 13](#_Toc9315839)

[Objectives 13](#_Toc9315840)

[System Organization 13](#_Toc9315841)

[Package Diagram 13](#_Toc9315842)

[Deployment Diagram 13](#_Toc9315843)

[System Architecture – Frontend 14](#_Toc9315844)

[Overall Architecture 14](#_Toc9315845)

[Subcomponents 14](#_Toc9315846)

[Ranking 14](#_Toc9315847)

[Item Detail 16](#_Toc9315848)

[Recommendation 18](#_Toc9315849)

[Mypage 19](#_Toc9315850)

[Search 21](#_Toc9315851)

[System Architecture – Backend 23](#_Toc9315852)

[Objectives 23](#_Toc9315853)

[Overall Architecture 23](#_Toc9315854)

[Subcomponents 23](#_Toc9315855)

[Backend Application 23](#_Toc9315856)

[Review Analyzing System 23](#_Toc9315857)

[Review Crawling System (Review Collecting System) 24](#_Toc9315858)

[Item Ranking System 25](#_Toc9315859)

[Protocol Design 26](#_Toc9315860)

[Objectives 26](#_Toc9315861)

[REST Protocol 26](#_Toc9315862)

[JSON 26](#_Toc9315863)

[Details 26](#_Toc9315864)

[Database Design 28](#_Toc9315865)

[Objectives 28](#_Toc9315866)

[ER Diagram 28](#_Toc9315867)

[Entities 29](#_Toc9315868)

[User 29](#_Toc9315869)

[Bookmark 30](#_Toc9315870)

[Authority 30](#_Toc9315871)

[Item 31](#_Toc9315872)

[Recommend Category 31](#_Toc9315873)

[Keyword 32](#_Toc9315874)

[Review 32](#_Toc9315875)

[Review Reference 33](#_Toc9315876)

[Relations 34](#_Toc9315877)

[Relational Schema 34](#_Toc9315878)

[SQL DDL 34](#_Toc9315879)

[Testing Plan 35](#_Toc9315880)

[Objectives 35](#_Toc9315881)

[Testing Policy 35](#_Toc9315882)

[Development Testing 35](#_Toc9315883)

[Release Testing 37](#_Toc9315884)

[User Testing 38](#_Toc9315885)

[Testing Case 38](#_Toc9315886)

[Development Plan 39](#_Toc9315887)

[Objectives 39](#_Toc9315888)

[Frontend Environment 39](#_Toc9315889)

[Vue.js 39](#_Toc9315890)

[Ionic 40](#_Toc9315891)

[Node.js 40](#_Toc9315892)

[Backend Environment 41](#_Toc9315893)

[Java 41](#_Toc9315894)

[Apache Tomcat 42](#_Toc9315895)

[Spring Framework 42](#_Toc9315896)

[Schedule 44](#_Toc9315897)

[Index 45](#_Toc9315898)

[Tables 45](#_Toc9315899)

[Figures 45](#_Toc9315900)

[Diagrams 45](#_Toc9315901)

# Preface

## Objectives

## Readership

## Document Structure

# Introduction

## Objectives

이번 챕터에서는 본 시스템의 설계에 사용된 다양한 다이어그램과 도구를 소개하고, 본 시스템의 개발 범위를 기술한다.

## Applied Diagram

### UML

UML is a general purpose and developmental modelling language and technique that combines different aspects of a system to represent relations, processes or results of an overall model or system. It is essential to mention that we have used it thoroughly in this document to visualize the workflow of the system.

Since it provides different modelling techniques and a handful subset of diagrams. It can be efficiently used to provide means of communication between developers and users as it covers wide range of symbols and definitions and it consists of the following diagrams: Package Diagram, Deployment Diagram, Class Diagram, State Diagram, Sequence Diagram and ER Diagram.

### Package Diagram

Package diagrams are kind of structural diagrams which show the arrangement and organization of model elements. Package diagram can show both structure and dependencies between sub-systems or modules in a more abstract way than other types of UML diagrams. This abstraction leads to the use of package diagrams in simplifying complex class diagrams by grouping them in packages.

### Deployment Diagram

The deployment diagram describes the physical deployment of information generated by the software program (artifact)on hardware components.

Deployment diagrams are made up of several UML shapes. The three-dimensional boxes, known as nodes, represent the basic software or hardware elements, or nodes, in the system. Lines from node to node indicate relationships, and the smaller shapes contained within the boxes represent the software artifacts that are deployed.

### Class Diagram

It is a diagram that is used to showcase the object classes of a system and the relationship between classes. One of the most fundamental reasons we are using it is because it provides a clear distinction between each class and show the hierarchy and dependency between them.

As far it goes for the inner structure of Class diagram, it consists of some fields indicating some variables, class methods and links or associations between classes.

### State Diagram

State Diagram is a technique to represent different states of a system and all possible next states based on some particular stimuli which triggers the change of the state.

This kind of diagram is very important to analyze different scenarios of the system as the states are represented as nodes and events as arcs which helps in identifying the behavior of the object classes defined in class diagram.

### Sequence Diagram

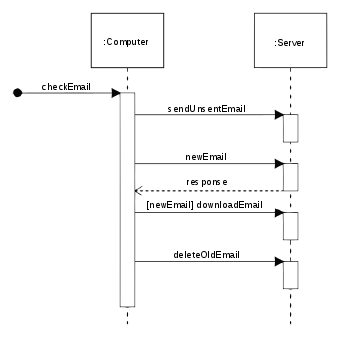


Figure 1: Example of Sequence Diagram

A sequence diagram to represent the interactions between the actors and objects of the system. To be more specific, the goal of this diagram is present the sequence of interaction and processes that take place in a specific use case instance so that a result could be generated. It is important to notice that the direction of the arrows here is essential to indicate the correct flow of actions.

### ER Diagram

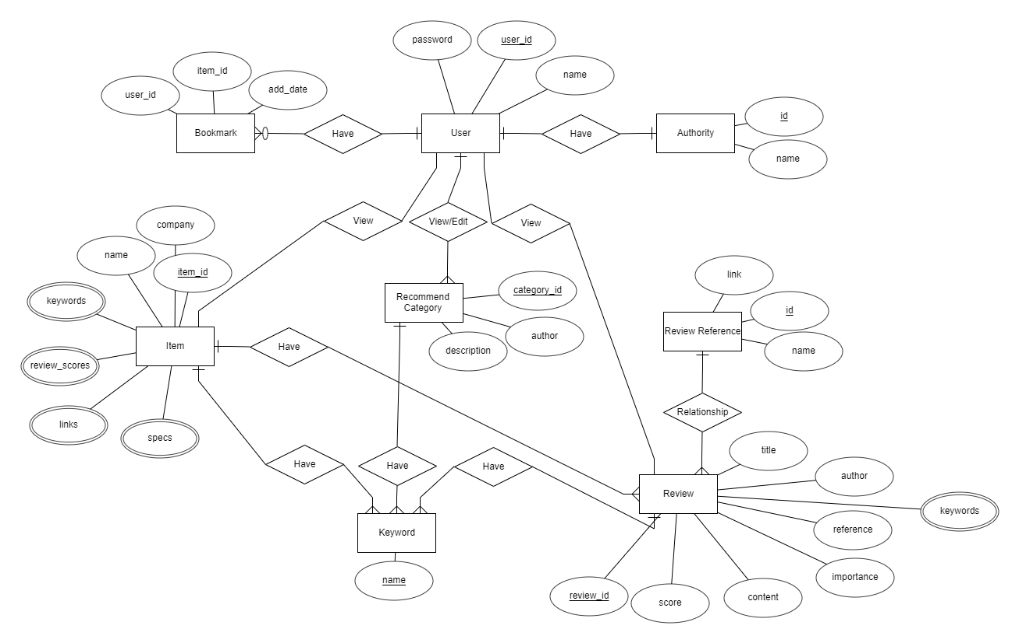


Figure 2: Example of ER Diagram

ER Diagram은 Entity가 가지고 있는 속성과 Entity간의 관계를 나타낸 다이어그램이다. 이 다이어그램은 주로 데이터베이스를 설계하는 데 사용되며, 해당 다이어그램을 기반으로 Relational Schema, SQL DDL을 작성하게 된다.

## Applied Tool

### Draw.io

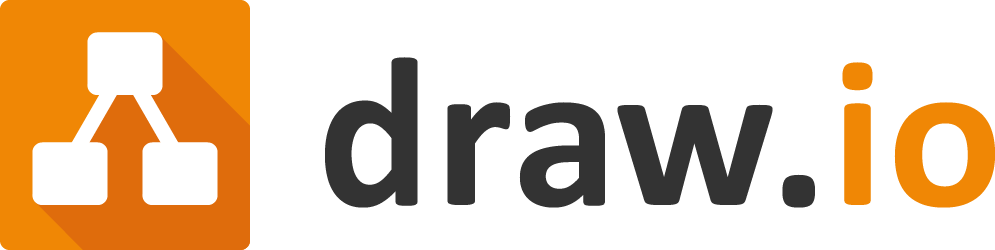


Figure 3: Draw.io Logo

Draw.io는 온라인 모델링 툴로서 많은 기본 템플릿과 도형을 제공하기 때문에 사용자가 직접 다이어그램에 사용하기 위해 도형을 만들 필요가 없다. 또한 도형 간 연결선을 간단하게 만들 수 있고, 격자에 위치를 맞출 수 있기 때문에 도형을 정렬하기 편리하다. 이 문서에서 사용된 대부분의 다이어그램은 본 도구로 작성되었다.

### PowerPoint



Figure 4: Powerpoint Logo

Powerpoint는 그래픽 프레젠테이션 툴이다. 주로 발표용으로 사용되지만 내장된 도형 작성 기능이 매우 강력하기 때문에 draw.io에서 만들기 힘든 복잡한 다이어그램을 작성하기 위해 사용하였다.

### ERDPlus



Figure 5: ERDPlus Logo

ERDPlus는 ER Diagram을 간단한 버튼 클릭으로 생성할 수 있게 해 주는 온라인 툴이다. 적은 노력으로 ER DIagram을 draw.io에 비해 간단하게 만들 수 있기 때문에 ER Diagram을 작성하는 데 사용하였다.

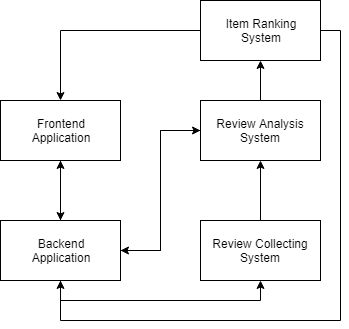
## Project Scope

# System Architecture – Overall

## Objectives

이번 챕터에서는 본 시스템의 전체적인 구조를 설명한다.

## System Organization



## Package Diagram

## Deployment Diagram

# System Architecture – Frontend

## Overall Architecture

## Subcomponents

### Ranking

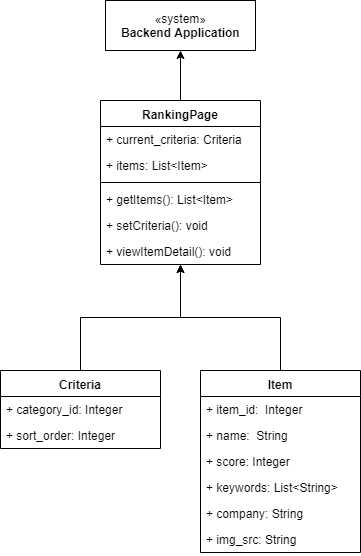


Diagram 1: System Architecture - Frontend - Ranking

1. RankingPage – 랭킹 페이지 객체
   1. attributes

+ current\_criteria: 랭킹 페이지의 상품 목록을 불러오는 데 사용하는 검색기준

+ items: 랭킹 페이지에서 로드되어 있는 상품 목록

* 1. methods

+ getItems(criteria: Criteria): 검색 조건에 해당하는 상품 목록을 백엔드 서버에서 가져온다.

+ setCriteria(criteria: Criteria): 검색 조건을 설정한다.

+ viewItemDetail(item\_id: Integer): 특정 상품의 자세한 정보를 조회한다. (Item Detail 컴포넌트로 이동)

1. Criteria – 검색 조건 객체(DTO)
   1. attributes

+ category\_id: 상품 분류 카테고리 ID

+ sort\_order: 정렬 순서(오름차순, 내림차순)

1. Item – 상품 객체(DTO)[[1]](#footnote-2)
   1. attributes

+ item\_id: 상품 ID

+ name: 상품 이름

+ score: 상품 평가 점수

+ keywords: 상품 연관 키워드 목록

+ company: 상품 제조사

+ img\_src: 상품 썸네일 사진 주소

### Item Detail

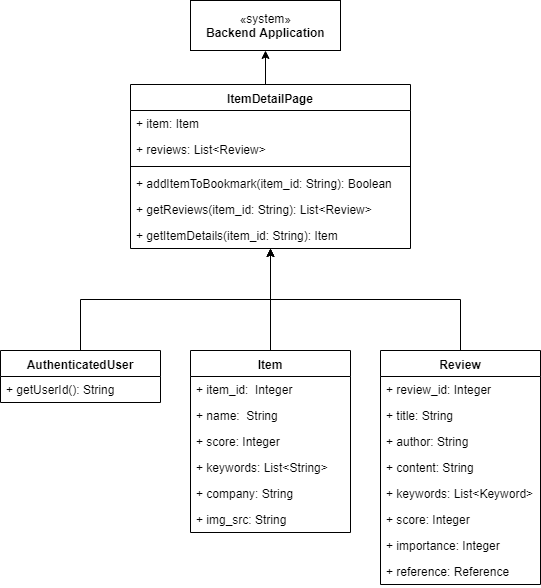


Diagram 2 - System Architecture - Frontend - Item Detail

1. ItemDetailPage – 상품 상세조회 페이지 객체
   1. attributes

+ item: 현재 조회 아이템 객체

+ reviews: 현재 조회 아이템 관련 리뷰 목록

* 1. methods

+ addItemToBookmark(item\_id: String): 상품을 사용자 북마크에 추가

+ getReviews(item\_id: String): 해당 상품의 리뷰 목록을 데이터베이스에서 조회

+ getItemDetails(item\_id: String): 상품의 상세 정보를 조회

1. AuthenticatedUser – 인증 유저 객체[[2]](#footnote-3)
2. Item – 상품 객체(DTO)[[3]](#footnote-4)
3. Review – 상품 리뷰 객체(DTO)
   1. attributes

+ review\_id: 리뷰 ID

+ title: 리뷰 제목

+ author: 리뷰 작성자

+ content: 리뷰 내용

+ keywords: 리뷰 관련 키워드

+ score: 리뷰 긍정 평가 정도 (1~100)

+ importance: 리뷰 중요도 (1~100)

+ reference: 리뷰 출처

### Recommendation

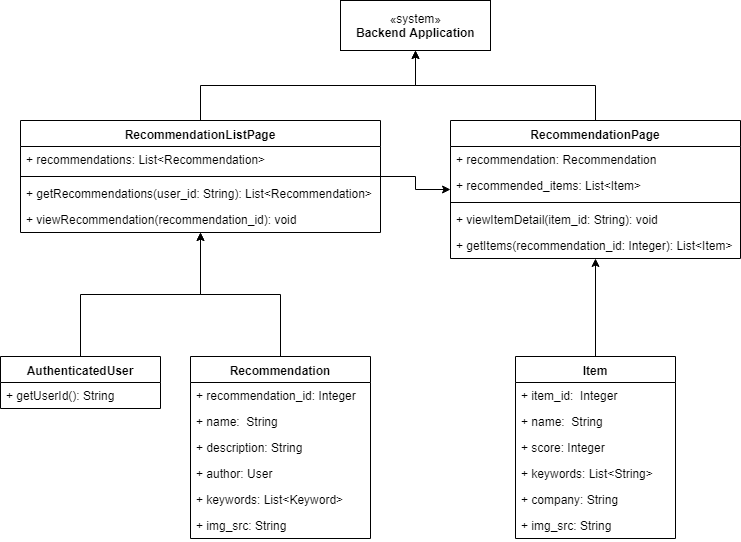


Diagram 3: System Architecture - Frontend – Recommendation

1. RecommendationListPage – 추천 카테고리 목록 페이지 객체
   1. attributes

+ recommendations: 추천 카테고리 목록

* 1. methods

+ getRecommendations(user\_id: String): 사용자 정보 기반으로 추천 카테고리 목록을 가져오는 메서드

+ viewRecommendation(recommendation\_id: Integer): 특정 추천 카테고리의 세부 정보를 확인하는 메서드 (Recommendation Page로 이동)

1. RecommendationPage – 추천 아이템 목록 페이지 객체
   1. attributes

+ recommendation: 현재 추천 카테고리 세부정보

+ recommended\_items:

1. AuthenticatedUser – 인증 유저 객체
2. Recommendation – 추천 카테고리 객체 (DTO)
3. Item – 상품 객체[[4]](#footnote-5) (DTO)

### Mypage

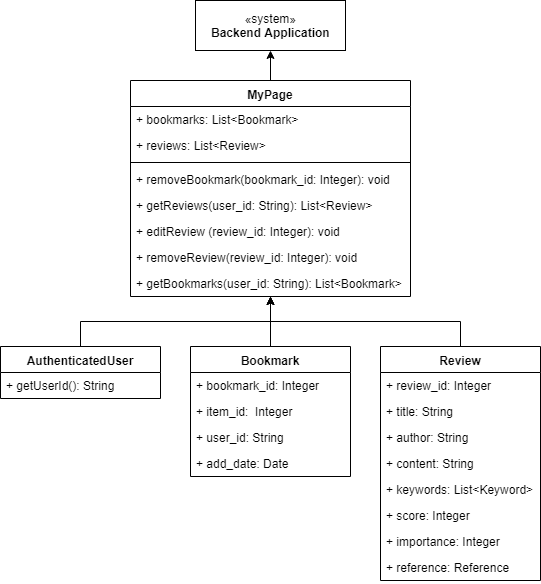


Diagram 4: System Architecture - Frontend – Mypage

1. Mypage – 마이페이지 객체
   1. attributes

+ bookmarks: 사용자가 추가한 북마크 목록

+ reviews: 사용자가 작성한 리뷰 목록

* 1. methods

+ removeBookmark(bookmark\_id: Integer): 추가한 북마크를 삭제하는 메소드

+ editReview(review\_id: Integer): 사용자가 작성한 리뷰를 수정하는 메소드

+ removeReview(review\_id: Integer): 사용자가 작성한 리뷰를 삭제하는 메소드

+ getReviews(user\_id: String) 사용자가 작성한 리뷰를 가져오는 메소드

+ getBookmarks(user\_id: String) 사용자가 추가한 모든 북마크 목록을 가져오는 메소드

1. AuthenticatedUser: 인증 유저 객체
2. Bookmark: 상품 북마크 객체 (DTO)
   1. attributes

+ bookmark\_id: 북마크 ID (PK)

+ item\_id: 상품 ID

+ user\_id: 사용자 ID

+ add\_date: 추가된 일자

1. Review: 리뷰 객체 (DTO)[[5]](#footnote-6)

### Search

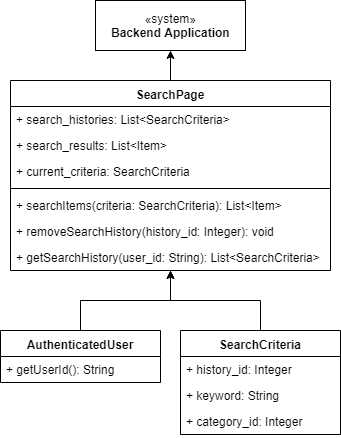


Diagram 5: System Architecture - Frontend- Search

1. SearchPage – 검색 페이지 객체
   1. attributes

+ search\_histories: 사용자의 검색 기록

+ search\_results: 검색 조건에 따른 검색 결과 상품 목록

+ current\_critera: 현재 검색 조건

* 1. methods

+ searchItems(criteria: SearchCriteria): 해당 검색 조건으로 상품을 검색한다.

+ removeSearchHistory(history\_id: Integer): 특정 검색 기록을 삭제한다.

+ getSearchHistory(user\_id: String): 사용자의 검색 기록을 조회한다.

1. AuthenticatedUser – 사용자 인증 객체
2. SearchCriteria – 검색 조건 객체 (DTO)
   1. history\_id: 검색 조건 기록용 ID
   2. keyword: 검색 조건으로 쓰인 키워드
   3. category\_id: 검색 조건 카테고리

# System Architecture – Backend

## Objectives

## Overall Architecture

스크린샷이(가) 표시된 사진

자동 생성된 설명

## Subcomponents

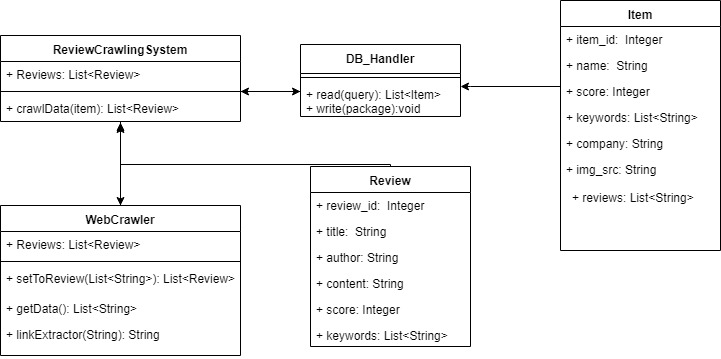
### Backend Application

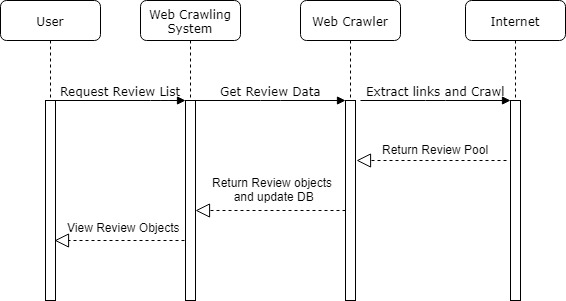
* Request Handler
* Controllers
* Service Manager
* Authentication Manager
* Database Handler

### Review Analyzing System

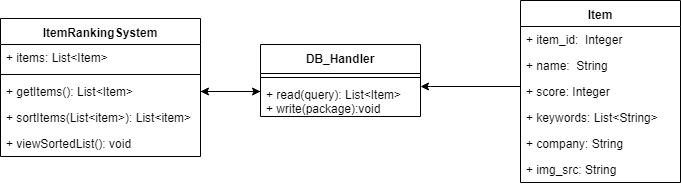
* Tokenizer

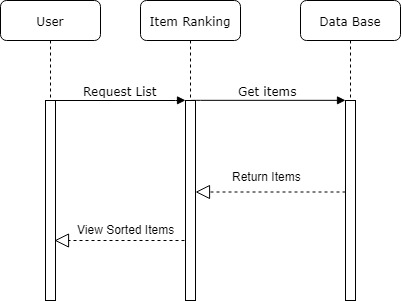
### Review Crawling System (Review Collecting System)





### Item Ranking System





# Protocol Design

## Objectives

이번 챕터에서는 각 서브시스템 간의 상호작용에 이용되는 프로토콜에 어떤 구조가 사용되는지 설명하고, 각 인터페이스가 어떻게 정의되어 있는지를 기술한다.

## REST Protocol

common protocol design for communicate between client-server

## JSON

**JSON**(**제이슨**[[1]](https://ko.wikipedia.org/wiki/JSON#cite_note-Pronunciation-1), JavaScript Object Notation)은 [속성-값 쌍](https://ko.wikipedia.org/w/index.php?title=%EC%86%8D%EC%84%B1-%EA%B0%92_%EC%8C%8D&action=edit&redlink=1)( attribute–value pairs and array data types (or any other serializable value)) 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 [개방형 표준](https://ko.wikipedia.org/wiki/%EA%B0%9C%EB%B0%A9%ED%98%95_%ED%91%9C%EC%A4%80) 포맷이다. 비동기 브라우저/서버 통신 ([AJAX](https://ko.wikipedia.org/wiki/Ajax))을 위해, 넓게는 [XML](https://ko.wikipedia.org/wiki/XML)([AJAX](https://ko.wikipedia.org/wiki/Ajax)가 사용)을 대체하는 주요 데이터 포맷이다. 특히, [인터넷](https://ko.wikipedia.org/wiki/%EC%9D%B8%ED%84%B0%EB%84%B7)에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 [컴퓨터 프로그램](https://ko.wikipedia.org/wiki/%EC%BB%B4%ED%93%A8%ED%84%B0_%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%A8)의 [변수](https://ko.wikipedia.org/wiki/%EB%B3%80%EC%88%98)값을 표현하는 데 적합하다.

## Details

* User
* Item
* Keyword
* RecommendCategory
* Bookmark
* Review

# Database Design

## Objectives

요구사항 명세서에서 작성한 데이터베이스 요구사항을 기반으로 하여 세부적인 데이터베이스 설계를 기술한다. ER Diagram을 통해 개괄적인 Entity 간의 관계를 기술하고, Relational Schema, SQL DDL 명세를 만든다.

## ER Diagram

본 시스템에는 User, Item, Bookmark, Authority, Recommend Category, Keyword, Review Reference, Review로 총 8개의 Entity가 존재한다. 각각의 Entity는 네모 박스의 형태로 표현되고, Entity 간의 관계는 마름모꼴로 표현된다. 이때 특정 Entity가 다른 Entity와 복수의 관계를 가질 수 있을 때는 해당 엔티티쪽에 삼지창 모양 선을 세 개 그어 나타내며, 특정 Entity가 없어도 되는 경우 Client Entity 쪽에 작은 동그라미를 표시한다. 각 Entity가 가지는 Attribute는 타원형으로 표현되는데, 각 Entity를 식별하는Unique key는 라벨에 밑줄을 그어 표시하고, 여러 Attribute를 허용하는 경우에는 테두리를 이중으로 긋는다.

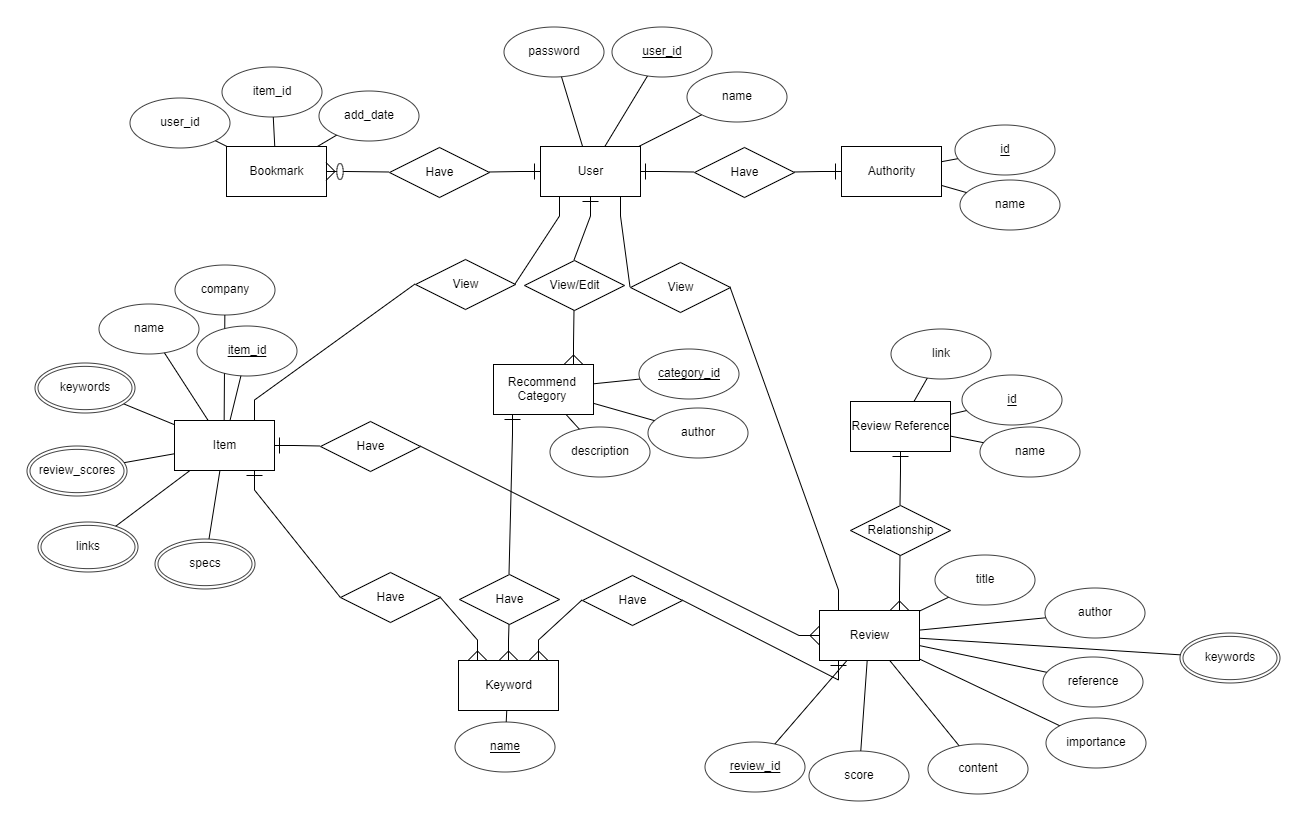


Diagram 6: Overall ER diagram

### Entities

#### User

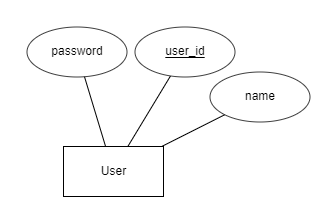


Diagram 7: ER diagram, Entity, User

User Entity는 사용자의 정보를 표현한다. user\_id 속성이 primary key이며, 이름, 패스워드 정보를 가지고 있다.

#### Bookmark

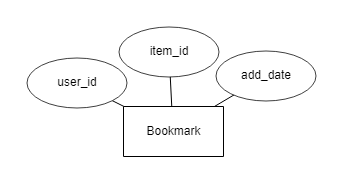


Diagram 8: ER diagram, Entity, Bookmark

Bookmark Entity는 각 사용자가 Item에 대해 설정한 즐겨찾기의 정보를 표현한다. Primary key는 존재하지 않으며, Item entity, User entity의 primary key를 조합한 composite key로 bookmark를 찾는다. 기타 속성으로는 bookmark가 추가된 날짜가 있다.

#### Authority

풀볼이(가) 표시된 사진

자동 생성된 설명

Diagram 9: ER diagram, Entity, Authority

Authority Entity는 각 사용자가 가질 수 있는 권한 정보를 나타낸다. Primary key는 id이며, 해당 권한의 이름인 name 속성이 있다.

#### Item

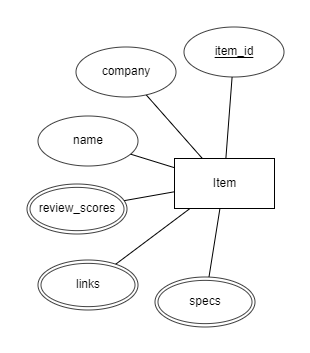


Diagram 10: ER diagram, Entity, Item

Item Entity는 각각의 상품에 대한 정보를 가지고 있으며, primary key는 item id이다. 이외 속성으로는name, company, specs, links, review\_scores가 있다. specs, links, review\_scores는 여러 개의 값을 가질 수 있다.

#### Recommend Category

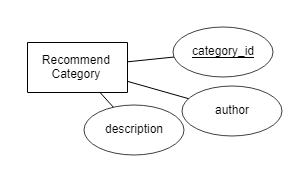


Diagram 11: ER diagram, Entity, Recommend Category

Recommend Category는 추천 카테고리에 대한 정보를 가지고 있다. primary key는 category\_id이며, 기타 속성으로는 author, description이 있다.

#### Keyword

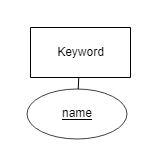


Diagram 12: ER diagram, Entity, Keyword

Keyword Entity는 각 상품을 묘사하는 키워드에 대한 정보를 가지고 있다. Keyword는 중복되지 않는 name 속성만 가지고 있다.

#### Review

전자기기이(가) 표시된 사진

자동 생성된 설명

Diagram 13: ER diagram, Entity, Review

Review Entity는 상품 리뷰에 대한 정보를 표현한다. primary key는 review\_id이며, 기타 속성으로는 title, author, keywords, reference, importance, content, score가 있다.

#### Review Reference

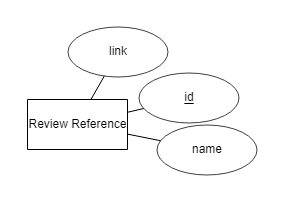


Diagram 14: ER diagram, Entity, Review Reference

Review Reference Entity는 상품의 리뷰가 작성된 출처에 대한 정보를 나타낸다. primary key는 id이며, 기타 속성으로 name, link가 있다.

### Relations

## Relational Schema

## SQL DDL

# Testing Plan

## Objectives

## Testing Policy

### Development Testing

-Development testing is aimed for synchronizing application of broad spectrum of defect prevention and detection strategies in order to reduce software development risks.

In this phase, we should proceed static code analyzing, data flow analyzing, metrics analyzing, peer code review, unit testing. We should focus on testing reliability, security, performance and regulatory compliance.

1. reliability:

There can be positive correlation of rate of re-purchasing and good grading, so we can use this barometer to evaluate the reliability for grading algorithm. For brief explanation, the way of measuring the accuracy of grading system is mapping similarity of re-purchasing.

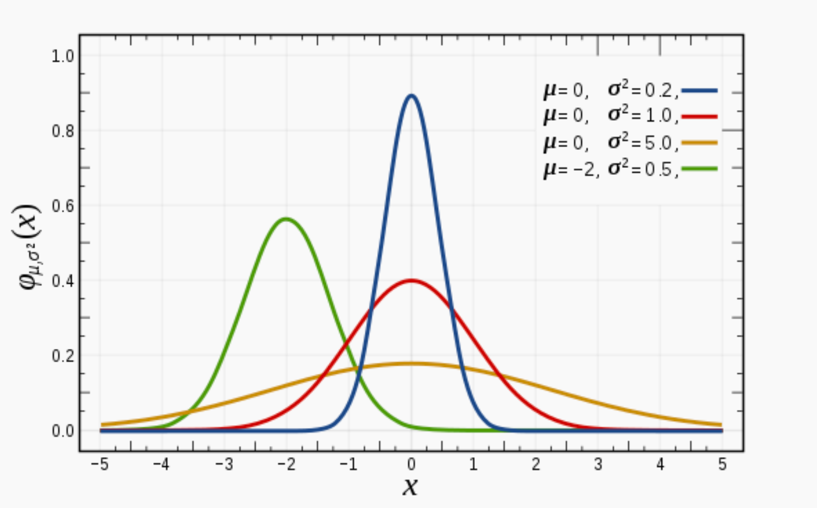
And we can observe tendency of grading for the same product as well, result of grade can be evaluated with average of grade by numerous customers. 

그림 1follow the tendency and check the reliability

1. security

When customers save their personal information by signing up, DB should protect private information by secure system. It can be achieved by hashing algorithm. Hashing function can convert information to meaningless number, and this process can be achieved by one way. So other users can`t catch the meaning of hashed value. –which is also called ‘encryption’.

And also DB structure should block the access of outer invading trials and should not lose important information. To test these features of development, we should review the codes that can be represented in page source itself. Because there are many cases that exposed private information in page source-which can be seen by any users. In other words, developers should review the structure of code carefully again, and should test its completeness by trial of hacking tests.

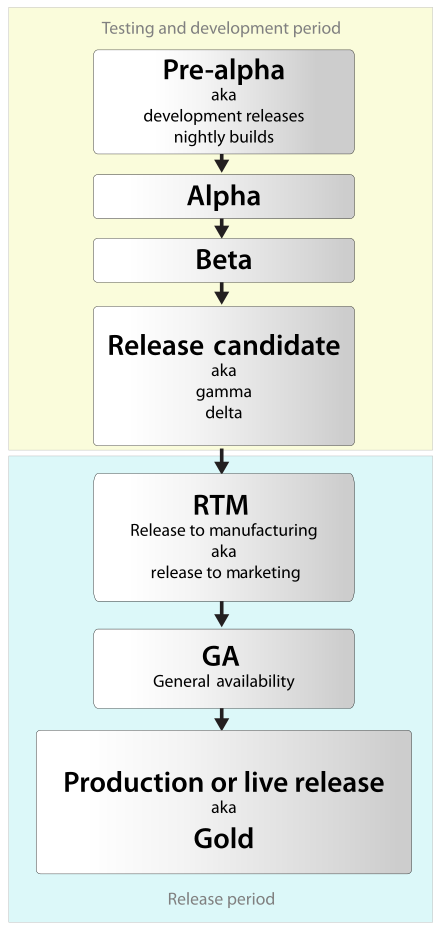
1. Performance

Performance can be evaluated by 1.speed 2. Accuracy 3. Robustness. When we decide Word processing algorithm, we can`t check all information but rather important features. This can be also explained by R-tree concept, and word processing algorithm itself should process information selectively to achieve high speed. This can be simply tested by measuring each processing time.

And accuracy can be checked by method of reliability section, and Robustness is the concept of measuring processing time when the size of loaded data is huge enough. To test this case, first load huge information to DB and test the processing time. By these methods, we can measure the performance.

### Release Testing

Release testing is a process to test the newer version /build of a software/ application to make sure that is flawless and doesn`t have any issues and works as intended. It has to be done prior to release.



Like above testing and development period, we should test it by alpha version first by developer and then release beta version to proceed final checking. By beta version testing, we can get feedback from real users and prolong the life cycle of software by this process.

### User Testing

User testing can be explained by the concept of usability testing. We should set up scenario and realistic situation that can proceed necessary user tests. We assume that there are 50 users for using Dealistic application and 10 products on the list. After setting this situation, we should check each concepts that are explained above, and compare its value for each iteration in tests.

### Testing Case

Testing case can be concretely described by setting exact number- 50 users and 10 products as written above. We can set review category 3~4 parts and then we can test our whole project application.

# Development Plan

## Objectives

이번 챕터에서는 실제 개발 단계에서 어떠한 기술과 개발 환경을 사용할 것인지를 기술하고, 개발 일정과 진행 상황을 설명한다.

## Frontend Environment

### Vue.js



Figure 6: Vue.js Logo

자바스크립트로 작성된 프론트엔드 프레임워크로서, 싱글 파일 컴포넌트 기법으로 하나의 컴포넌트에 필요한 HTML 템플릿, 자바스크립트, CSS 스타일시트 등을 하나의 파일에 모두 작성할 수 있어서 코드 집적도와 유지보수성을 높일 수 있으며, 프론트엔드에도 MVC 패턴을 적용하여 난잡해지기 쉬운 프론트엔드 코드를 정리할 수 있게 도와준다. 또한 프로그레시브 웹 앱(PWA)을 적용할 수 있기 때문에 웹 앱으로 배포하여 다양한 접속 환경에서도 네이티브 애플리케이션과 동일한 사용자 경험을 제공할 수 있다.

### Ionic

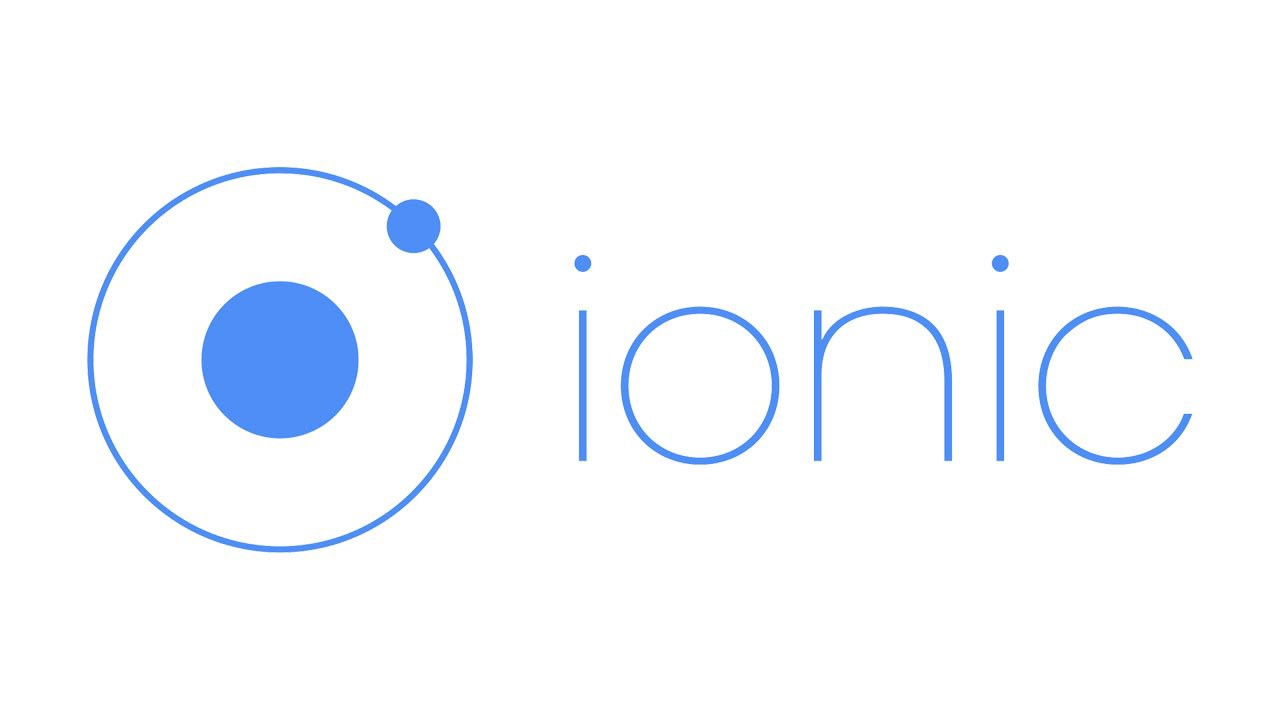


Figure 7: Ionic Logo

다양한 접속 디바이스를 타겟으로 컴파일해 해당 디바이스의 환경에 맞춘 UI 스타일을 사용할 수 있게 해 주는 프론트엔드 프레임워크로서, 개발자가 일일이 개별 디바이스 환경에 맞춘 UI 컴포넌트를 개발하지 않고 한 번의 코딩만으로 다양한 환경에 맞게 컴파일할 수 있기 때문에 생산성이 올라간다. 또한 Vue.js 와 결합해 작동할 수 있다.

### Node.js

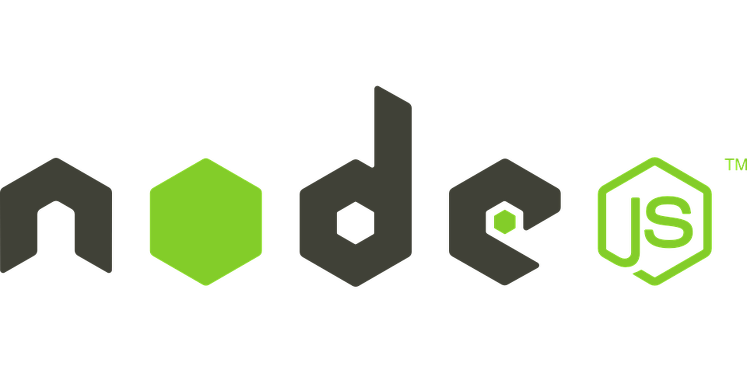


Figure 8: Node.js Logo

Node.js는 경량 서버 프레임워크로서 다양한 모듈을 활용할 수 있고

## Backend Environment

### Java



Figure 9: Java Logo



Java는 객체 지향 프로그래밍 언어로서, JSP(Java Server Page)와 Servlet 등의 기술을 이용해 다양한 웹 애플리케이션 서버를 구현하는 데 사용되고 있다. 본 시스템에서는 backend애플리케이션 서버를 만드는 데 사용했는데, 팀 멤버의 대부분이 자바 프로그래밍에 익숙하기 때문이다.

### Apache Tomcat



Figure 10: Apache Tomcat Logo



아파치 톰캣은 아파치 소프트웨어 재단의 어플리케이션 서버로 Java Servlet을 실행하고 JSP 코드가 포함되어 있는 웹페이지를 만들어 주는 WAS(Web Application Server)의 종류 중 하나이다. WAS는 웹 서버와 웹 컨테이너의 결합으로 다양한 기능을 컨테이너에 구현하여 다양한 역할을 수행할 수 있는 서버로 웹서버의 backend부분에서 J2EE나 J2SE 같은 JSP/Servlet을 처리해주는 역할을 한다.

### Spring Framework

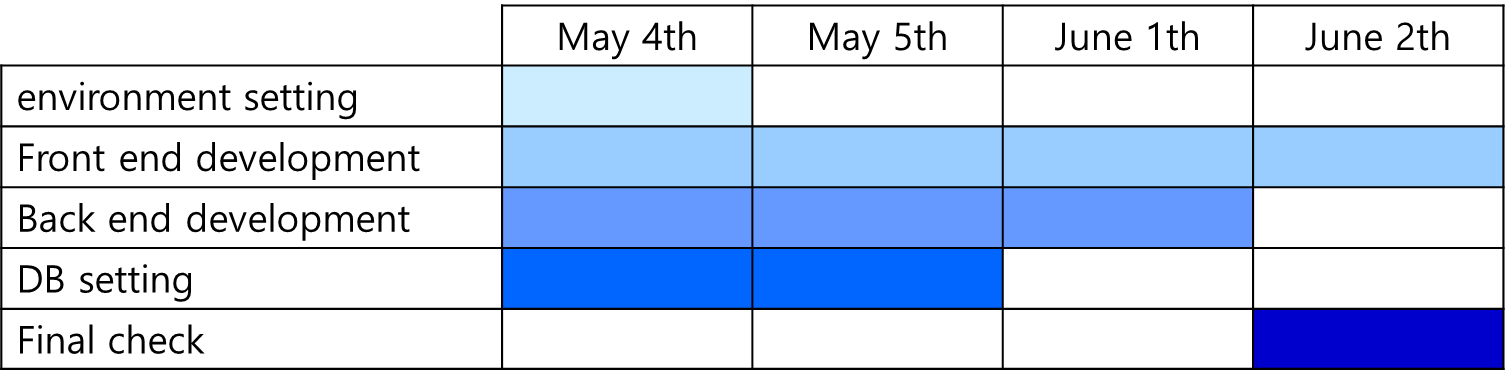


Figure 11: Spring Framwork Logo



Spring Framework는 자바(JAVA) 플랫폼을 위한 오픈소스(Open Source) 어플리케이션 프레임워크이다. Spring Framework는 자바 엔터프라이즈 개발을 편하게 해주는 오픈 소스 경량급 애플리케이션 프레임워크이다. 자바 개발을 위한 프레임워크로 종속 객체를 생성해주고, 조립해주는 도구로 스프링 프레임워크는 어떤 플랫폼에서도 종합적인 프로그래밍과 자바 기반의 현대 엔터프라이즈 어플리케이션의 Configuration Model을 제공한다. 스프링의 핵심요소는 어플리케이션 단위의 인프라를 제공한다는 것이다. 스프링은 기업용 어플리케이션의 plumbing에 초점을 맞추고 있다. 그래서 개발팀은 특정 배포 환경에서 불필요한 시도없이 어플리케이션의 비지니스 로직에 초점을 맞출 수 있다는 장점이 있다.

## Schedule



문서화 작업이 늦어져 예상 계획보다 지연되고 있다. 구현 단계에서 Parallel 개발을 통해 일정 단축이 필요하다.

# Index

## Tables

## Figures

[Figure 1: Example of Sequence Diagram 10](#_Toc9314949)

[Figure 2: Example of ER Diagram 11](#_Toc9314950)

[Figure 3: Draw.io Logo 11](#_Toc9314951)

[Figure 4: Powerpoint Logo 12](#_Toc9314952)

[Figure 5: ERDPlus Logo 12](#_Toc9314953)

[Figure 6: Vue.js Logo 39](#_Toc9314954)

[Figure 7: Ionic Logo 40](#_Toc9314955)

[Figure 8: Node.js Logo 40](#_Toc9314956)

[Figure 9: Java Logo 41](#_Toc9314957)

[Figure 10: Apache Tomcat Logo 42](#_Toc9314958)

[Figure 11: Spring Framwork Logo 42](#_Toc9314959)

## Diagrams

[Diagram 1: System Architecture - Frontend - Ranking 14](#_Toc9314935)

[Diagram 2 - System Architecture - Frontend - Item Detail 16](#_Toc9314936)

[Diagram 3: System Architecture - Frontend – Recommendation 18](#_Toc9314937)

[Diagram 4: System Architecture - Frontend – Mypage 19](#_Toc9314938)

[Diagram 5: System Architecture - Frontend- Search 21](#_Toc9314939)

[Diagram 6: Overall ER diagram 29](#_Toc9314940)

[Diagram 7: ER diagram, Entity, User 29](#_Toc9314941)

[Diagram 8: ER diagram, Entity, Bookmark 30](#_Toc9314942)

[Diagram 9: ER diagram, Entity, Authority 30](#_Toc9314943)

[Diagram 10: ER diagram, Entity, Item 31](#_Toc9314944)

[Diagram 11: ER diagram, Entity, Recommend Category 31](#_Toc9314945)

[Diagram 12: ER diagram, Entity, Keyword 32](#_Toc9314946)

[Diagram 13: ER diagram, Entity, Review 32](#_Toc9314947)

[Diagram 14: ER diagram, Entity, Review Reference 33](#_Toc9314948)

1. 백엔드에서 사용되는 Item 객체는 더 많은 attribute가 존재하지만, API를 통해 받아오는 JSON 객체에는 해당 기능에 필요한 attribute만 담겨 있다. [↑](#footnote-ref-2)
2. 해당 객체는 Vue 프레임워크 로컬 저장소에서 자체적으로 관리하는 사용자 인증 객체로써, 모든 프론트엔드 컴포넌트에서 접근할 수 있다. [↑](#footnote-ref-3)
3. Ranking의 상품 객체 정의와 동일함. [↑](#footnote-ref-4)
4. Rank의 상품 정의와 동일함 [↑](#footnote-ref-5)
5. Item Detail의 리뷰 객체 정의와 동일함 [↑](#footnote-ref-6)