DEALISTIC

Design Specification

Student Number	Name
2014312794	김준현 (Junhyun Kim)
2017314461	모하메드(Muhammad Shakeel)
2016315379	최지혜 (Jihye Choi)
2016315426	허준범 (Junbeom Heo)
2015312406	유도영 (Doyeong Yoo)

Contents

1.Preface

This chapter defines the expected readership of the document, and briefly introduces the content of each chapter. This chapter also describes version history including a rationale for the creation of a new version and a summary of the changes made in each version.

1.1. Readership

본 문서는 다양한 독자에게 읽힐 것을 상정하고 있다. 따라서 각 부분을 서술하는 데 있어 어떠한 독자층을 상정하고 있는지를 설명한다.**

1.2. Document Structure

1. Introduction

   본 문서를 서술하는 데 사용된 다양한 다이어그램과 표현 도구들에 대해 설명하고, 본 소프트웨어 프로젝트가 다루는 시스템의 범위에 대해 서술한다.

2. System Architecture

   시스템과 각 서브시스템의 구조를 개괄적으로 기술하고, 시스템의 전체 기능이 각 서브시스템과 하드웨어에 어떻게 할당되었는지 설명한다.

3. Protocol Design

   시스템의 각 컴포넌트, 특히 프론트엔드 시스템과 백엔드 시스템간의 상호작용을 규정하는 인터페이스와 프로토콜을 어떻게 구성하는지에 대해 기술하고, 해당 인터페이스가 어떤 기술에 기반해 있는지 설명한다.

4. Database Design

   Requirements 문서에서 규정된 데이터베이스 요구 사항을 기반으로, 각 데이터 엔티티의 속성과 관계를 ER diagram 을 통해 표현하고 최종적으로 Relational Schema, SQL DDL 를 작성한다.

5. Testing Plan

   미리 작성된 Test 를 이용해, verification 과 validation 을 시행한다. 이 Test 작성에 대한 계획을 설명한다.**

6. Development Plan

   시스템을 구현하는 데 필요한 개발 도구와 프로그래밍 언어, 라이브러리 등의 개발 환경에 대해 설명하고, 시스템 개발 일정을 기술한다.

7. Index

   본 문서에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

2. Introduction

2.1. Objectives

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

2.2. Applied Diagram

A.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.

B. 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.

C. 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.

D. 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.

E. 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.

F. Sequence Diagram

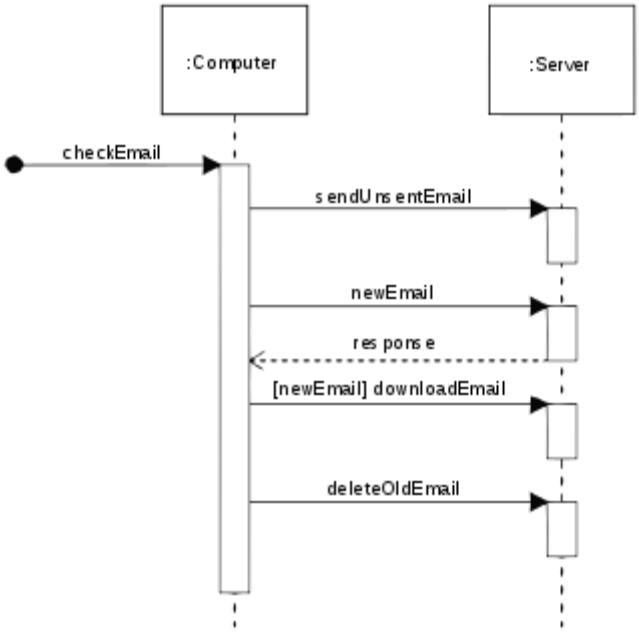


Figure 1: Example of Sequence Diagram

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.

G. ER Diagram

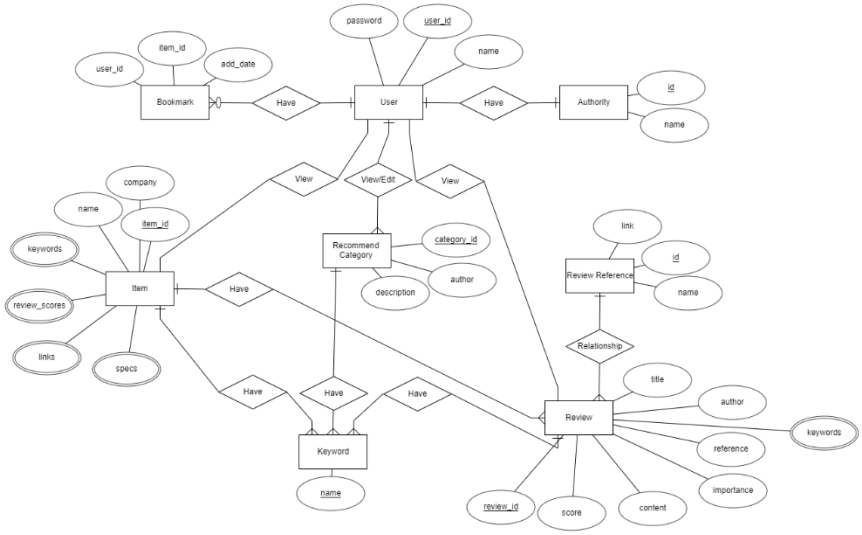


Figure 2: Example of ER Diagram

ER Diagram 은 Entity 가 가지고 있는 속성과 Entity 간의 관계를 나타낸 다이어그램이다. 이 다이어그램은 주로 데이터베이스를 설계하는 데 사용되며, 해당 다이어그램을 기반으로 Relational Schema, SQL DDL 을 작성하게 된다.

2.3. Applied Tool

A. Draw.io

Figure 3: Draw.io Logo

Draw.io 는 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.

B. PowerPoint

Figure 4: Powerpoint Logo

Powerpoint 는 그래픽 프레젠테이션 툴이다. 주로 발표용으로 사용되지만 내장된 도형 작성 기능이 매우 강력하기 때문에 draw.io 에서 만들기 힘든 복잡한 다이어그램을 작성하기 위해 사용하였다.

C. ERDPlus

Figure 5: ERDPlus Logo

ERDPlus 는 ER Diagram 을 간단한 버튼 클릭으로 생성할 수 있게 해 주는 온라인 툴이다. 적은 노력으로 ER DIagram 을 draw.io 에 비해 간단하게 만들 수 있기 때문에 ER Diagram 을 작성하는 데 사용하였다.

2.4. Project Scope

본 시스템은 상품 평점의 신뢰도가 낮은 상황에서 사용자들이 일일이 직접 리뷰를 읽고 신뢰성을 측정해야 했던 기존의 오픈마켓 서비스들의 단점을 극복하기 위해 리뷰에 자연어 처리 시스템을 도입하여 리뷰의 신뢰성을 판단할 수 있는 기반 지식 없이도 사용자들이 믿을 수 있는 리뷰에 기반한 상품 선택을 도와주는 시스템이다. 본 시스템의 핵심 기능은 리뷰의 분석 기능이며, 해당 기능을 중심으로 각 서브시스템들이 상호작용하도록 설계하였다.

먼저 Frontend System 은 시스템과 사용자와의 상호작용을 담당하며, Backend System Frontend System 에서 오는 데이터 요청에 응답하고 Review Collecting System, Review Analysis System 를 실행시키는 역할을 담당한다. 사용자가 리뷰를 작성하거나 Review Collecting System 에서 정기적으로 오픈마켓에서 리뷰의 목록을 가져오면, Review Analysis System 은 리뷰를 분석해 리뷰의 긍정 평가 정도와 중요도를 측정한다. Review Analysis System 이 분석한 리뷰를 데이터베이스에 전달하면, Item Ranking System 과 Recommendation System 은 추가된 정보를 바탕으로 각 상품의 평가 수치와 추천 카테고리를 수정한다.

3. System Architecture – Overall

3.1. Objectives

이번 챕터에서는 본 시스템의 전체적인 구조를 설명한다. 시스템 전체의 구조와 각 서브시스템의 개략적인 구조, 서브시스템 간의 관계를 서술한다.

3.2. System Organization

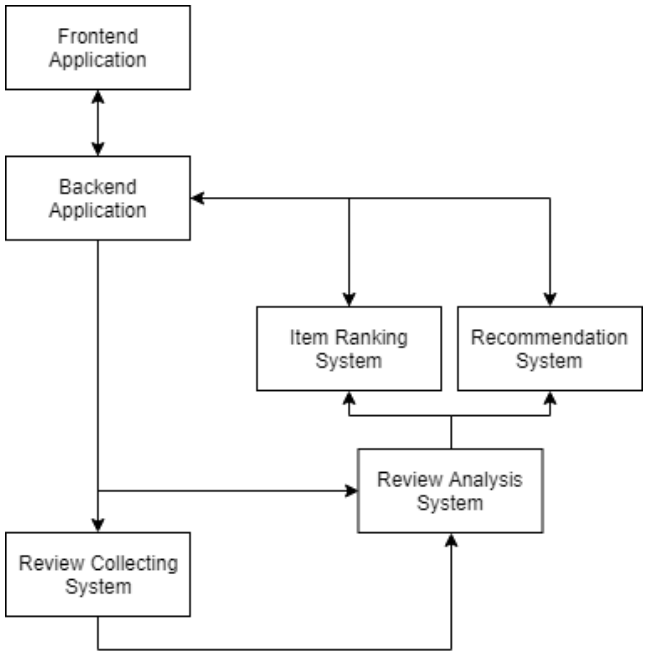


Diagram 1: Overall System Organization

본 서비스는 클라이언트-서버 모델을 적용해 설계했으며, Frontend Application 이 사용자와의 모든 상호작용을 맡고, 프론트엔드 애플리케이션과 백엔드 애플리케이션은 JSON 을 기반으로 한 HTTP 통신으로 데이터를 송수신한다. 백엔드 애플리케이션은 프론트엔드로부터 들어오는

각 요청을 컨트롤러로 분배하고, 필요한 객체 정보를 데이터베이스로부터 가져와 JSON 포맷으로 가공한 뒤 전달한다.

백엔드 애플리케이션은 일정 시간마다 Review Collecting System 을 실행하는데, 요청을 받은 Review Collecting System 은 저장되어 있는 목표 크롤링 사이트의 목록과 목표 상품 목록을 이용해 해당 사이트에서 리뷰의 정보를 수집한다. 리뷰 정보 수집이 완료되면, 최소한의 가공을 거친 리뷰 목록을 Review Analysis System 에게 전달하고, Review Analysis System 은 전달받은 리뷰를 Google Natural Language API 를 이용해 분석 후 필요한 정보를 데이터베이스에 저장한다. 이후 Item Ranking System 과 Recommendation System 이 업데이트된 리뷰 데이터베이스를 이용해 각 상품의 점수 순위와 추천 카테고리를 업데이트한다. 이후 다시 사용자가 정보를 요청할 경우 업데이트된 정보를 전달한다.

A. Frontend Application

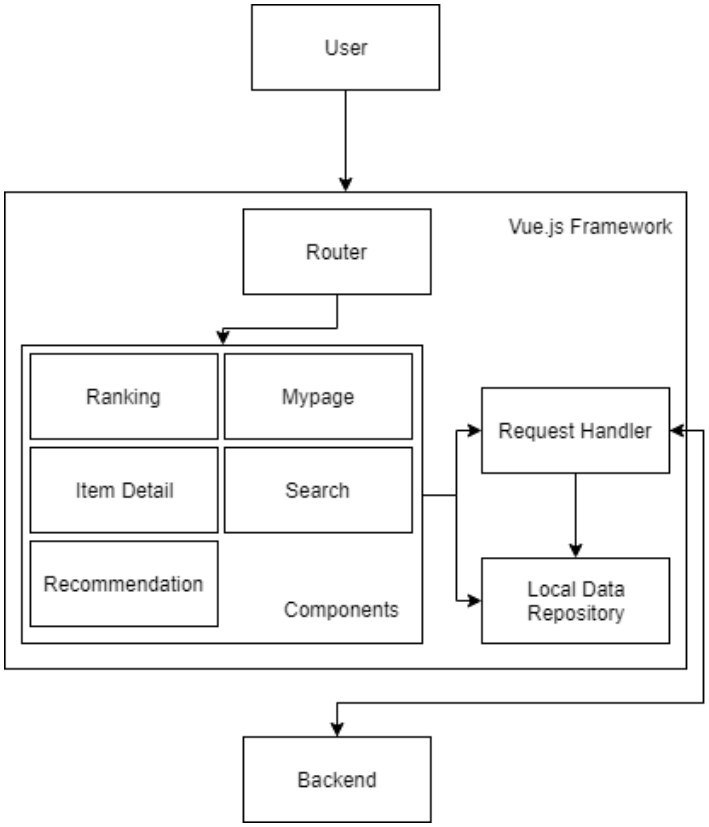


Diagram 2: System Architecture – Frontend

사용자와의 상호작용을 전담하는 시스템으로, Vue.js 프레임워크를 통해 각 컴포넌트를 관리한다. Ranking, Mypage, Item Detail, Search, Recommendation 컴포넌트가 존재하며, 컴포넌트간 공유하는 Shared Resource 의 경우 Local Data Repository 서브시스템에서 관리한다. 각 컴포넌트가 백엔드와의 통신을 전담하는 Request Handler 에게 필요한 데이터를 요청하면, Request Handler 는 미리 정해 둔 적절한 프로토콜로 요청을 변형해 백엔드에 전달하게 된다.

B. Backend Application

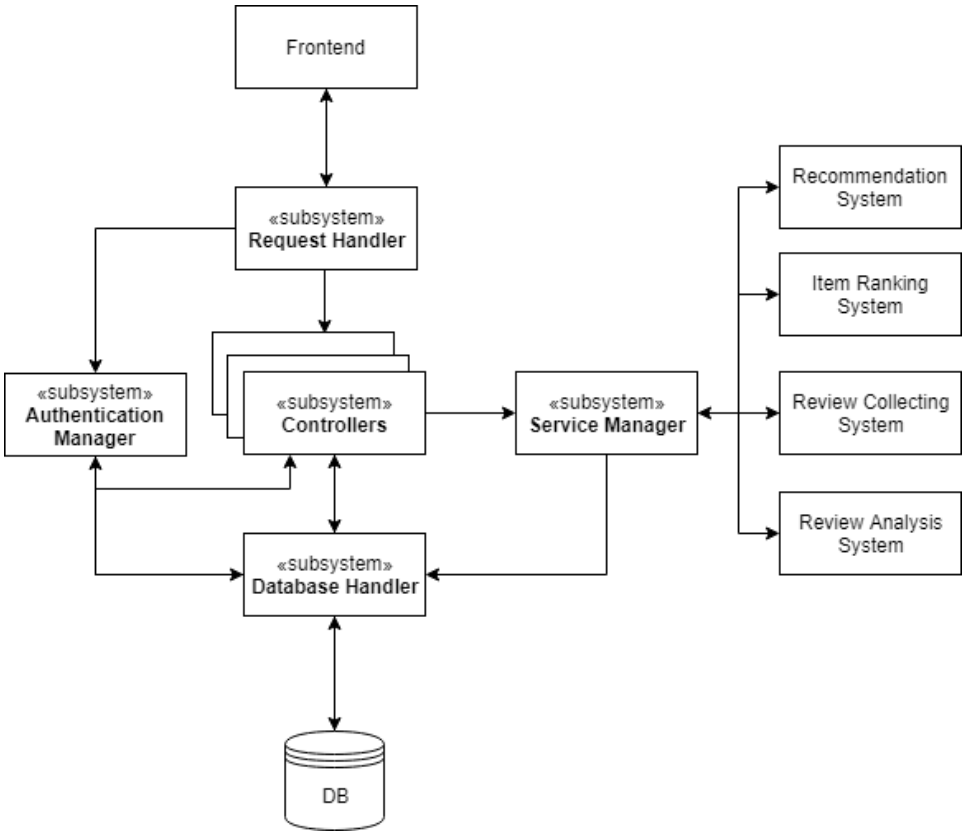


Diagram 3: System Architecture - Backend

4. System Architecture – Frontend

4.1. Objective

전체 시스템 아키텍처 중 사용자와의 상호작용을 담당하는 프론트엔드 시스템의 구조와 각 컴포넌트의 구성, 컴포넌트간의 관계를 서술한다.

4.2. Subcomponents

A. Ranking

1. Class Diagram

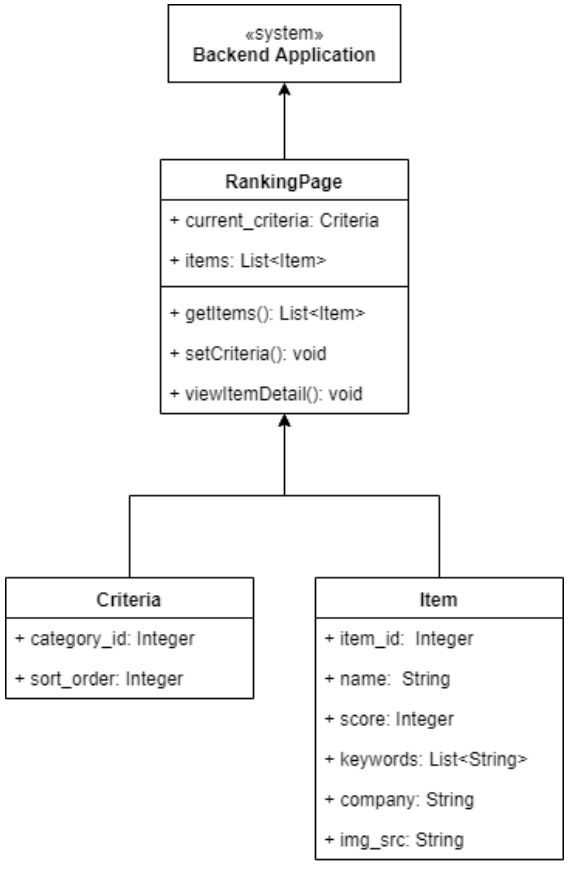


Diagram 4: System Architecture - Frontend - Ranking

1. RankingPage – 랭킹 페이지 객체

   A. attributes

   + current_criteria: 랭킹 페이지의 상품 목록을 불러오는 데 사용하는 검색기준

   + items: 랭킹 페이지에서 로드되어 있는 상품 목록

   B. 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)

2. Criteria – 검색 조건 객체(DTO)

   A. attributes

   + category_id: 상품 분류 카테고리 ID

   + sort_order: 정렬 순서(오름차순, 내림차순)

3. Item – 상품 객체(DTO)[1]

   A. attributes

   + item_id: 상품 ID

   + name: 상품 이름

   + score: 상품 평가 점수

[1] 백엔드에서 사용되는 Item 객체는 더 많은 attribute 가 존재하지만, API 를 통해 받아오는 JSON 객체에는 해당 기능에 필요한 attribute 만 담겨 있다.

+ keywords: 상품 연관 키워드 목록

+ company: 상품 제조사

+ img_src: 상품 썸네일 사진 주소

2. Sequence Diagram

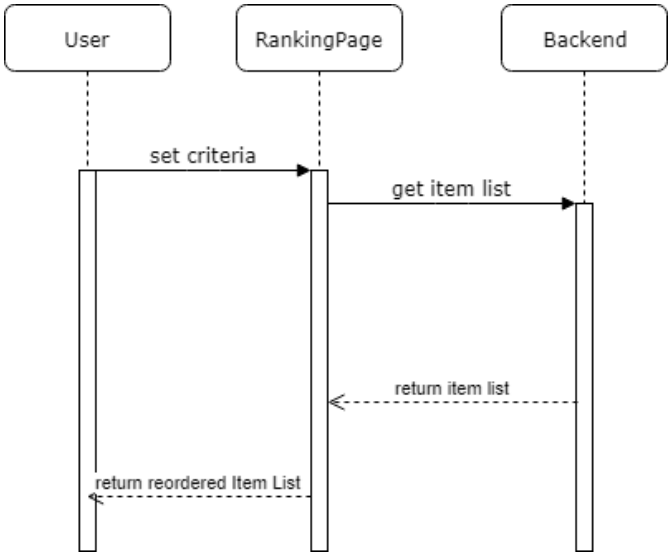


Diagram 5: System Architecture – Frontend - Ranking - Sequence Diagram

B. Item Detail

1. Class Diagram

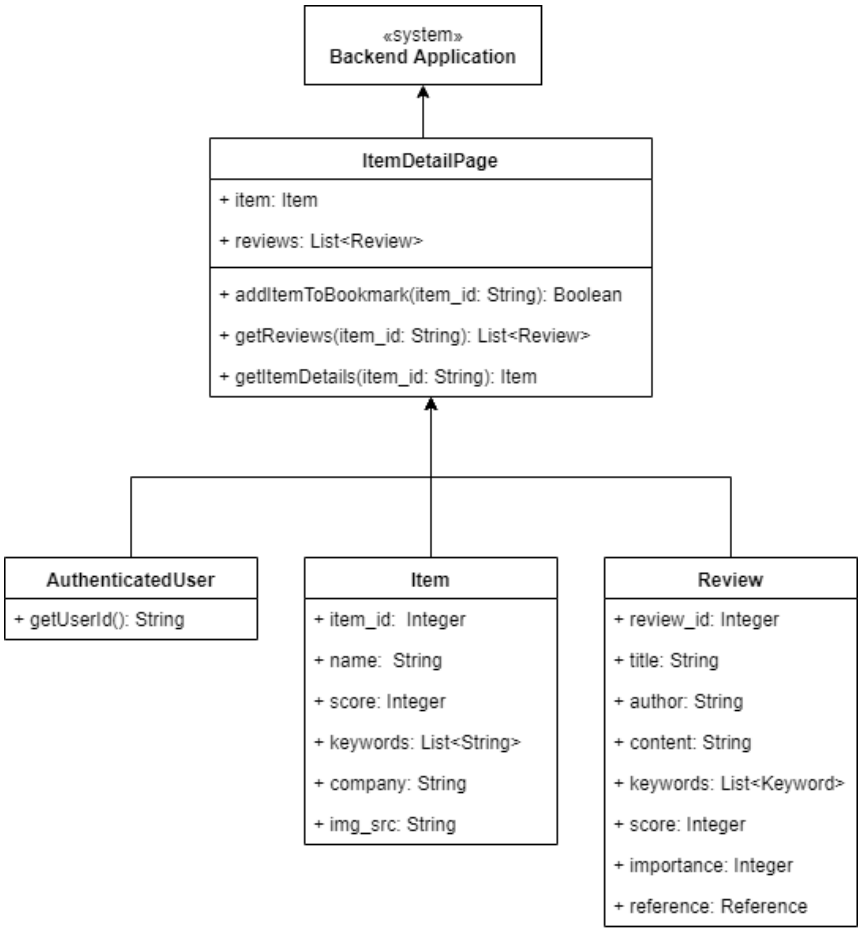


Diagram 6 - System Architecture - Frontend - Item Detail

1. ItemDetailPage – 상품 상세조회 페이지 객체

   A. attributes

   + item: 현재 조회 아이템 객체

   + reviews: 현재 조회 아이템 관련 리뷰 목록

   B. 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

2. AuthenticatedUser – 인증 유저 객체[2]
3. Item – 상품 객체(DTO)[3]
4. Review – 상품 리뷰 객체(DTO)

A. 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

[2] 해당 객체는 Vue 프레임워크 로컬 저장소에서 자체적으로 관리하는 사용자 인증 객체로써, 모든 프론트엔드 컴포넌트에서 접근할 수 있다.

[3] Ranking 의 상품 객체 정의와 동일함.

2. Sequence Diagram

C. Recommendation

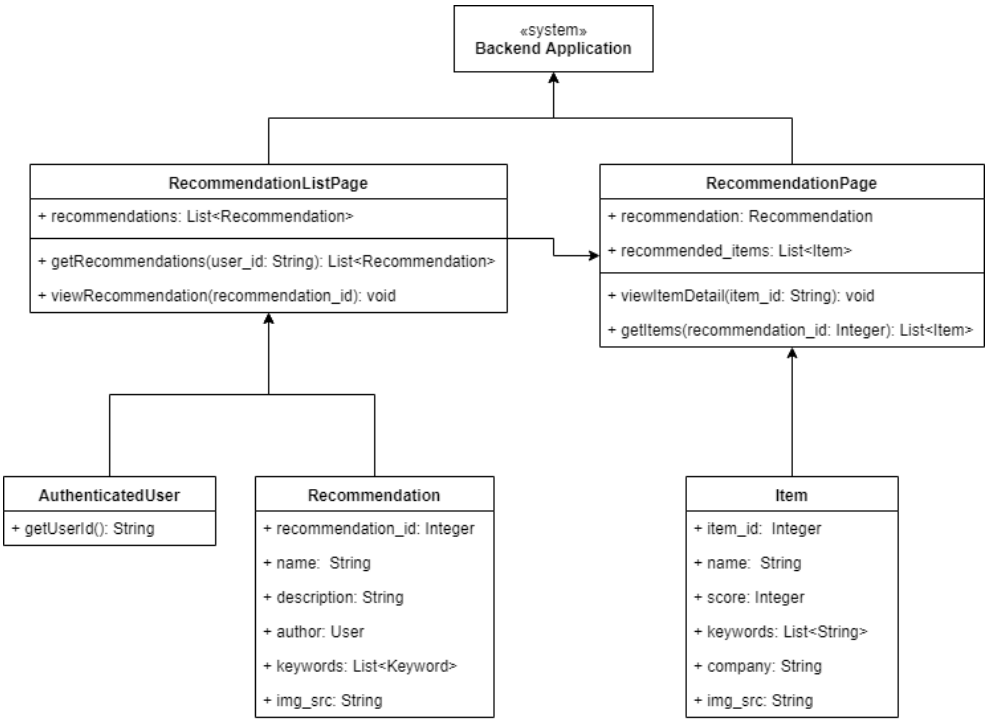


Diagram 7: System Architecture - Frontend – Recommendation

1. RecommendationListPage – 추천 카테고리 목록 페이지 객체

   A. attributes

   + recommendations: 추천 카테고리 목록

   B. 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 Page 로 이동)

2. RecommendationPage – 추천 아이템 목록 페이지 객체

   A. attributes

+ recommendation: 현재 추천 카테고리 세부정보

+ recommended_items: 현재 추천 카테고리를 만족하는 상품 목록

B. methods

+ viewItemDetail(item_id: Integer): 상품 세부 정보 조회(ItemDetail 페이지로 이동)

+ getItems(recommendation_id: Integer): 상품 조회

3. AuthenticatedUser – 인증 유저 객체

4. Recommendation – 추천 카테고리 객체 (DTO)

A. attributes

+ recommendation_id: 추천 카테고리 ID (PK)

+ name: 추천 카테고리 이름

+ description: 카테고리 설명

+ author: 작성자

+ keywords: 추천 카테고리에 포함되어 있는 키워드

+ img_src: 추천 카테고리 썸네일 이미지 주소

5. Item – 상품 객체[4] (DTO)

[4] Rank 의 상품 정의와 동일함

D. Mypage

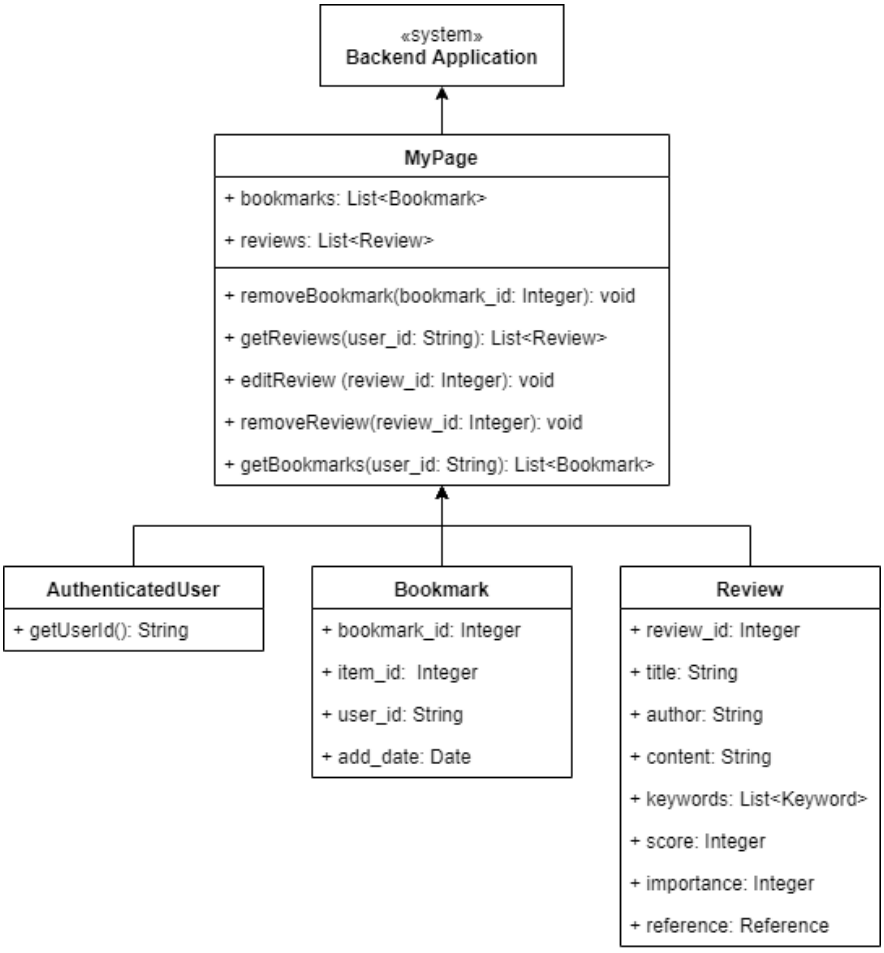


Diagram 8: System Architecture - Frontend – Mypage

1. Mypage – 마이페이지 객체

   A. attributes

   + bookmarks: 사용자가 추가한 북마크 목록

   + reviews: 사용자가 작성한 리뷰 목록

   B. 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

2. AuthenticatedUser: 인증 유저 객체

3. Bookmark: 상품 북마크 객체 (DTO)

   A. attributes

   + bookmark_id: 북마크 ID (PK)

   + item_id: 상품 ID

   + user_id: 사용자 ID

   + add_date: 추가된 일자

4. Review: 리뷰 객체 (DTO)[5]

[5] Item Detail 의 리뷰 객체 정의와 동일함

E. Search

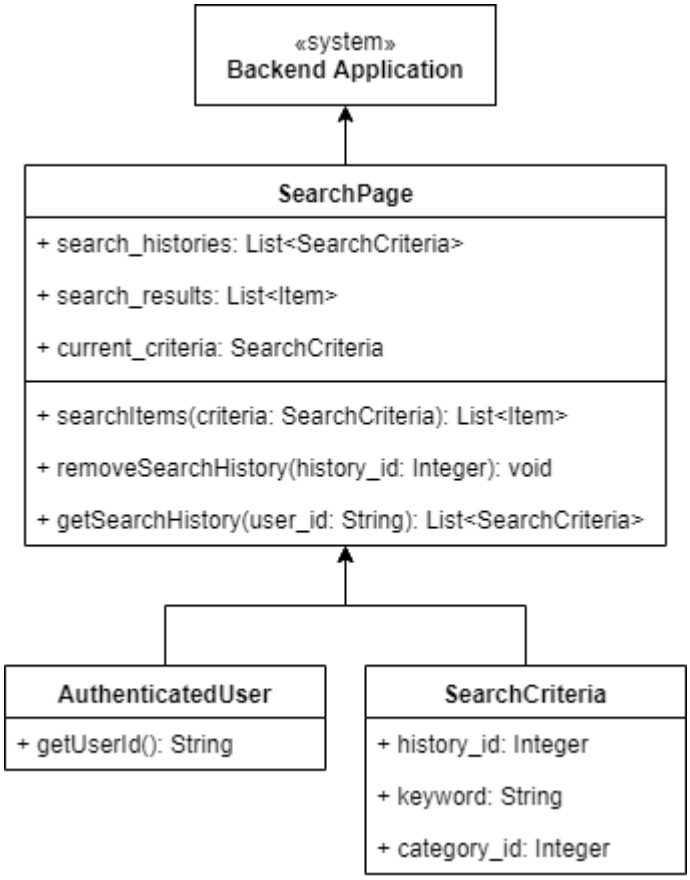


Diagram 9: System Architecture - Frontend- Search

1. SearchPage – 검색 페이지 객체

   A. attributes

   + search_histories: 사용자의 검색 기록

   + search_results: 검색 조건에 따른 검색 결과 상품 목록

   + current_critera: 현재 검색 조건

   B. 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.

2. AuthenticatedUser – 사용자 인증 객체
3. SearchCriteria – 검색 조건 객체 (DTO)
   A. history_id: 검색 조건 기록용 ID
   B. keyword: 검색 조건으로 쓰인 키워드
   C. category_id: 검색 조건 카테고리

5. System Architecture – Backend

5.1. Objectives

이번 챕터에서는 전체 시스템 중 사용자와의 상호작용을 담당하는 프론트엔드를 제외한 백엔드 시스템과 각 서브시스템의 구조에 대해 설명한다.

5.2. Overall Architecture

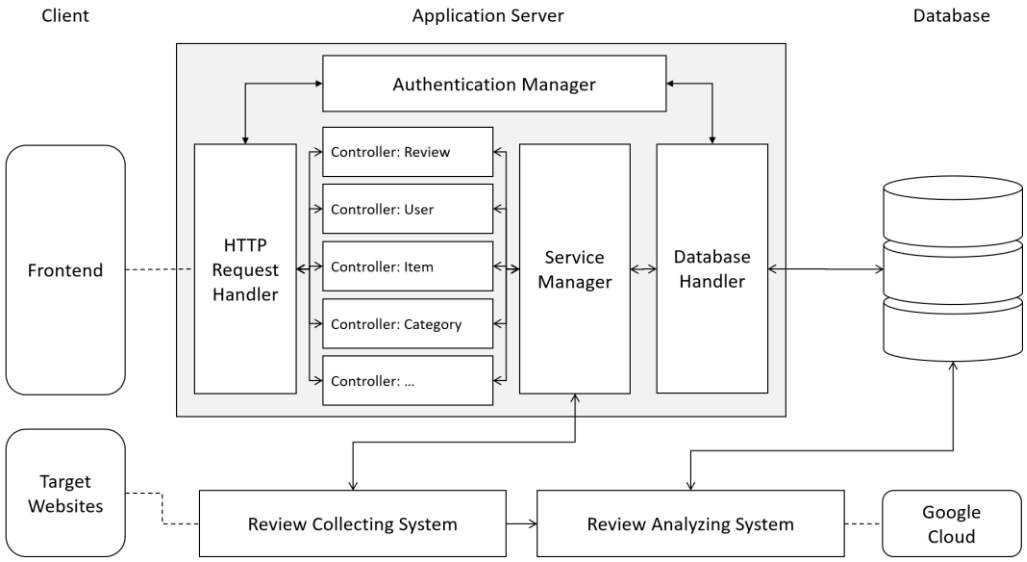


Diagram 10: System Architecture - Backend - Overall

백엔드 시스템의 전체적인 구조는 위와 같은데, Frontend 와의 직접적인 통신을 담당하는 Application Server(회색 직사각형)에서 데이터베이스, Review Collecting System, Review Analysis System 을 호출하는 구조로 되어 있다.

5.3. Subcomponents

A. Application Server

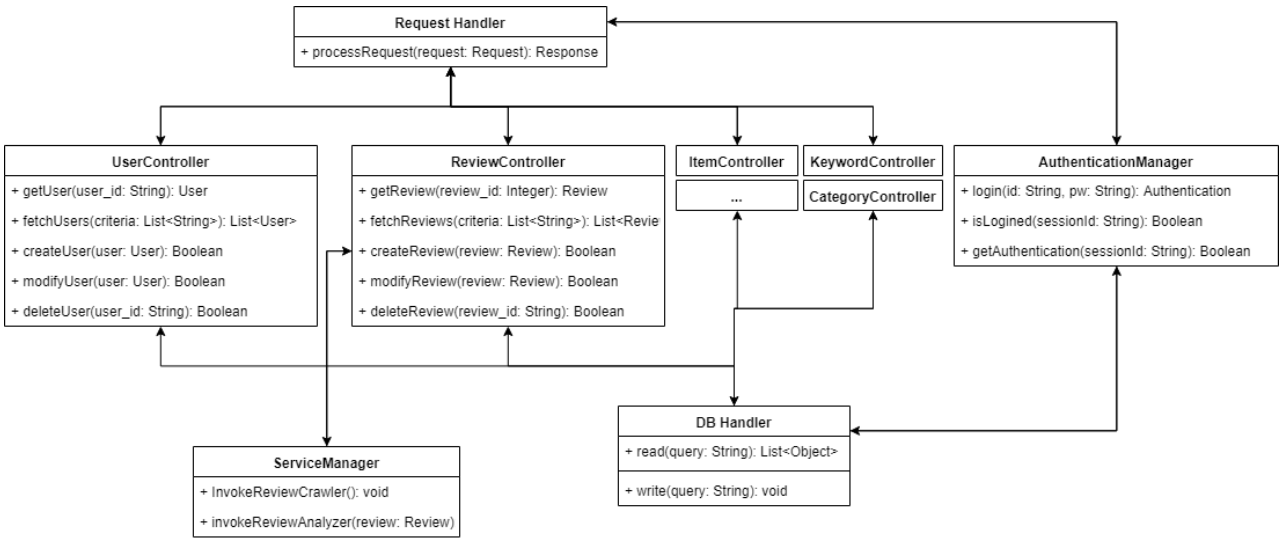


Diagram 11: System Architecture – Backend – Application Server

1. Request Handler: 프론트엔드 요청 처리 객체
   A. attributes: 없음
   B. processRequest(request: Request): 서버로 들어온 요청을 각 컨트롤러로 분배해 처리하고 요청을 반환하는 메서드
2. Controllers(공통)
   A. attributes: 없음
   B. getEntity(entity_id): 엔티티를 가져오는 메서드
   C. fetchEntities(criteria: List<String>) 검색 조건에 맞는 엔티티의 리스트를 가져오는 메서드
   D. createEntity(entity: Entity): 엔티티를 생성하는 메서드
   E. modifyEntity(entity: Entity): 엔티티를 수정하는 메서드

F. deleteEntity(entity_id): 엔티티를 삭제하는 메서드

3. Service Manager

A. invokeReviewCrawler(): 리뷰 크롤러를 실행하는 메서드

B. invokeReviewAnalyzer(review: Reviwer): 해당 리뷰에 대한 분석을 실행하는 메서드

4. Authentication Manager

A. login(id: String, pw: String): ID/PW 로 로그인하고 인증 토큰을 발급하는 메서드

B. isLogin(token: String): 인증이 필요한 로직에서 사용자가 유효한 로그인 토큰을 가지고 있는지 확인하는 메서드

C. getAuthentication(sessionId: String): 로그인 토큰을 바탕으로 현재 접속한 사용자의 정보를 불러오는 메서드

5. Database Handler

A. read(query: String): 데이터베이스 조회 쿼리 스트링을 데이터베이스에 질의한다.

B. write(query: String): 데이터베이스 입력 쿼리 스트링을 데이터베이스에 질의한다.

B. Review Crawling System (Review Collecting System)

1. Class Diagram

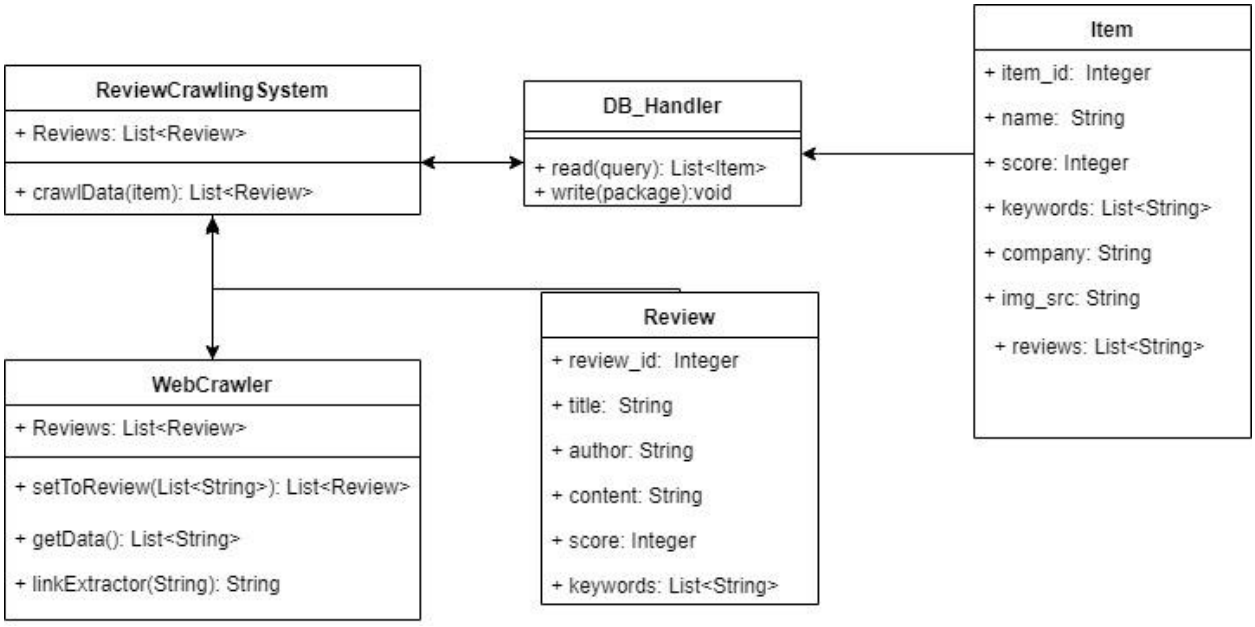


Diagram 12: System Architecture - Backend - Review Crawling System

- Classes Description:

1) ReviewCrawlingSystem Class: this class works as a driver class that calls some other functions from different classes. In addition, it contains the following fields and methods:

   a) Reviews field: List of all possible reviews to be loaded from the crawling system.

   b) crawlData method: this function will interact with the WebCrawler class methods which are the most essential focus here.

2) DB_Handler Class: this class interacts with the backend data base through two main methods:

   a) read method: a method for retrieving some data from the data base in case of existing or loaded reviews.

b) write method: a method for updating some data in the data base in case of the need for storing the loaded reviews from the crawling system.

3) WebCrawler Class: this class has the actual implementation for the web crawling mechanism as it consists of 3 main methods:

a) linkExtractor method: this function will extract and get some links from given set of pages and discover some reviews.

b) getData method: this function will get the return value from linkExtractor and store the data in the string list.

c) setToReview method: this function will set all the given attributes of the review object by splitting the string passed to it.

4) Item and Review Classes: these two classes are some basic classes which are used to store objects that can be used for further manipulation.

2. Sequence Diagram

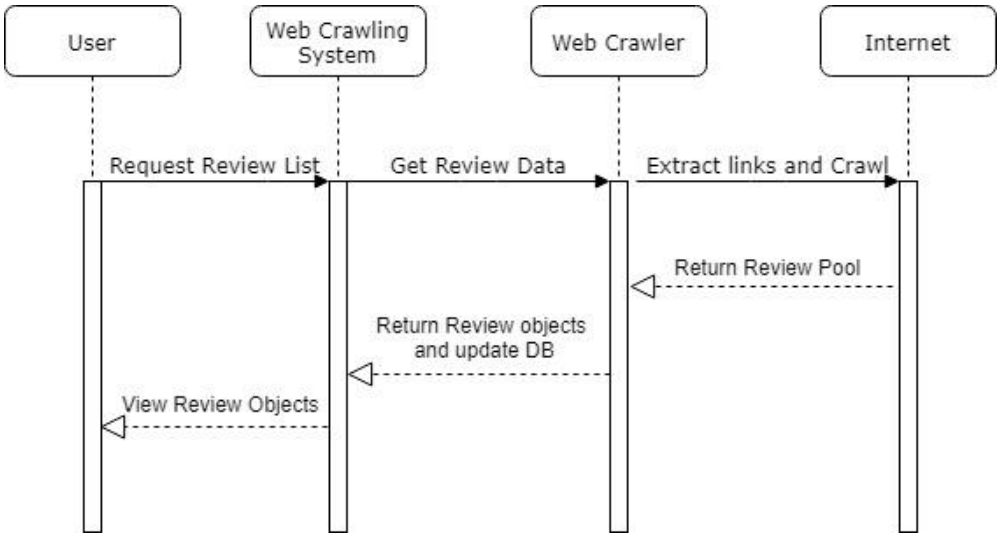


Diagram 13: System Architecture - Backend - Review Crawling System - Sequence Diagram

C. Item Ranking System

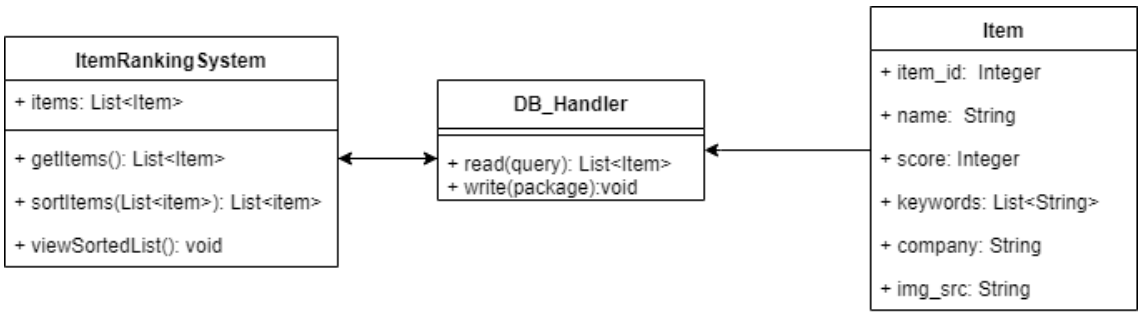


Diagram 14: System Architecture - Backend - Item Ranking System

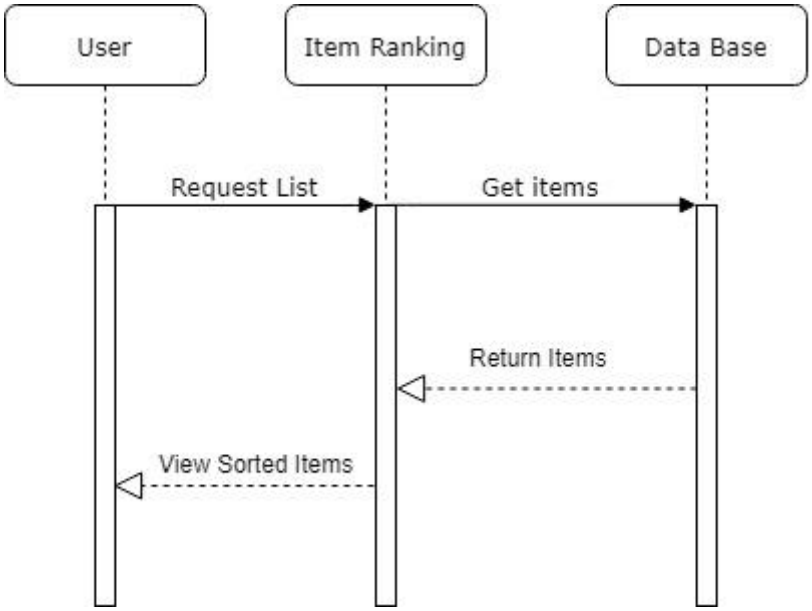


Diagram 15: System Architecture - Backend - Item Ranking System - Sequence Diagram

D. Review Analyzing System

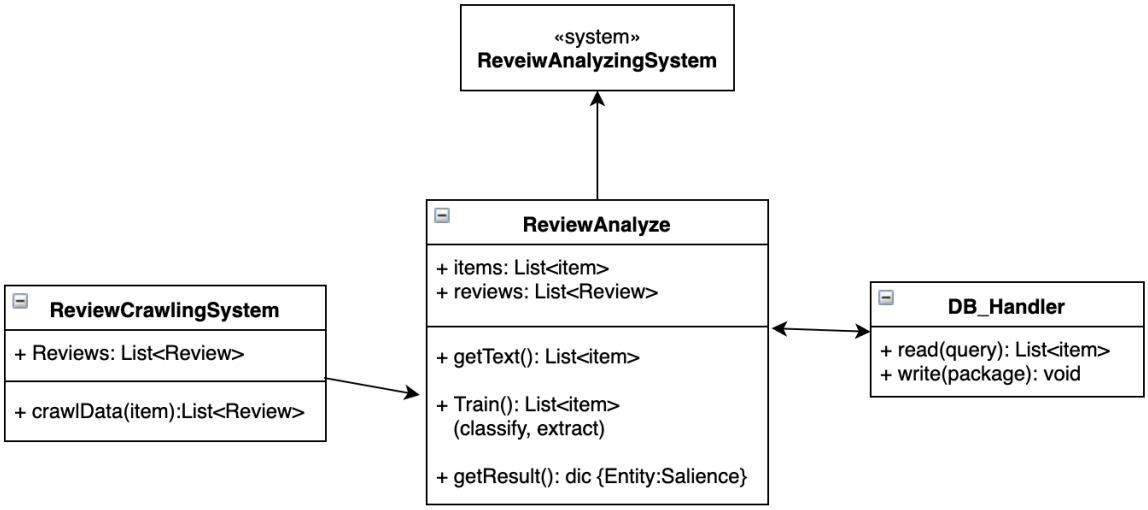


Diagram 13 : System Architecture – Backend – Review Analyzing System

6. Protocol Design

6.1. Objectives

이번 챕터에서는 각 서브시스템 간의 상호작용, 특히 프론트엔드 시스템과 백엔드 애플리케이션 서버 시스템간 상호작용에 이용되는 프로토콜에 어떤 구조가 사용되는지 설명하고, 각 인터페이스가 어떻게 정의되어 있는지를 기술한다.

6.2. REST API

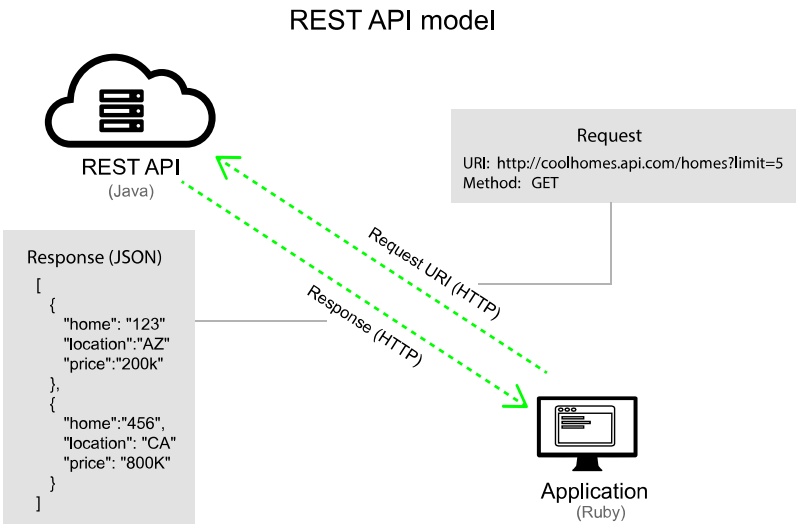


Figure 6: REST API model

본 시스템은 프론트엔드와 백엔드 사이의 통신에 웹 인터페이스, 즉 HTTP 를 이용하며, 요청과 응답 형식은 REST API 에 따른다. REST API 란 Representational State Transfer 의 약자로서, 서버에 저장되어 있는 각 자원을 이름으로 구분하여 해당 자원의 상태를 주고받는 API 의 설계 형식을 의미한다. REST API 는 크게 다음 세 부분으로 구성되어 있다.

1. 자원(Resource): URI
   - 서버가 보관하고 있는 데이터를 나타내며, 각 자원은 고유한 URI 를 가진다.

2. 행위(Verb): HTTP Method

   - 서버의 자원에 접근해 상태를 조작하기 위한 요청 행위로서, 각 조작 행위는 HTTP Method 를 통해 표현된다. (POST, GET, PUT, DELETE)

3. 표현(Representation): JSON

   - 클라이언트의 요청에 대한 서버의 응답 형식으로, 주로 JSON 이 사용된다.

클라이언트와 서버 간의 통신에 REST API 를 사용할 경우 서버와 클라이언트 간의 의존성이 줄어들고, 프로토콜이 이해하기 쉬워지는 장점이 있으며, 이를 통해 본 시스템에서 개발하는 프론트엔드 클라이언트뿐 아니라 다른 시스템에서 서버의 자원을 쉽게 이용할 수 있기 때문에 확장성이 높다.

6.3. JSON

Figure 7: JSON Logo

JSON(**제이슨**[1], JavaScript Object Notation)은 속성-값 쌍(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 포맷이다. 비동기 브라우저/서버 통신 (AJAX)을 위해, 넓게는 XML(AJAX 가 사용)을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합하다.

6.4. Details

A. Authentication

a. Login

- Request

Method	POST	
URI	/authentication/login (REST 예외)	
Request Body	id	사용자 ID
	password	사용자 패스워드

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (ID/PW 불일치인 경우)	
Success Response Body	success	true
Failure Response Body	success	false
	message	reason of failure

b. Signup

- Request

Method	POST	
URI	/authentication/signup(REST 예외)	
Request Body	id	사용자 ID
	name	사용자 이름
	password	사용자 패스워드

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (ID 중복인 경우)	
Success Response Body	-	-

Failure Response Body	message	reason of failure

A. User

a. Get

- Request

Method	GET	
URI	/users/:id	
Parameters	-	-
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	404 Not Found (id 에 해당하는 User 가 없음)	
Success Response Body	user	User Object
Failure Response Body	-	-

b. Search

- Request

Method	GET	
URI	/users	
Parameters	id	User ID
	name	User Name
	authority	User Authority
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	404 Not Found (검색 조건에 해당하는 User 가 없음)	
Success Response Body	users	User Object List

Failure Response Body	-	-

c. Add/Remove Bookmark

- Request

Method	POST/DELETE	
URI	/users/:id/bookmarks	
Parameters	item_id	상품 ID
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (이미 등록됨/없는 bookmark 임)	
Success Response Body	-	-
Failure Response Body	-	-

C. Item

a. Get

- Request

Method	GET	
URI	/items/:id	
Parameters	-	-
Header	-	-

- Response

Success Code	200 OK	
Failure Code	404 Not Found (id 에 해당하는 Item 이 없음)	
Success Response Body	item	Item Object
Failure Response Body	-	-

b. Search

- Request

Method	GET	
URI	/items	
Parameters	id	Item ID
	name	Item Name
	company	Item manufacture company
	keywords	Item Keywords
	score	Item Score
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	404 Not Found (검색 조건에 해당하는 Item 이 없음)	
Success Response Body	items	Item Object List

D. Review

a. Get

- Request

Method	GET	
URI	/reviews/:id	
Parameters	-	-
Header	-	-

- Response

Success Code	200 OK	
Failure Code	404 Not Found (id 에 해당하는 Review 가 없음)	
Success Response Body	review	review Object
Failure Response Body	-	-

b. Search

- Request

Method	GET	
URI	/reviews	
Parameters	id	Review ID
	reference	Review reference
	item	Review item
	keywords	Review mentioned keywords
	score	Review score
	importance	Review importance
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	404 Not Found (검색 조건에 해당하는 Item 이 없음)	
Success Response Body	reviews	Review Object List

c. Write

- Request

Method	POST	
URI	/reviews	
Parameters	title	Review title
	content	Review content
	item_id	Review item id
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	review_id	Created review id
Failure Response Body	message	fail reasons - 해당 item 에 대한 review 가 이미 있음

		- 본문에 비속어가 있음

d. Delete

- Request

Method	DELETE	
URI	/reviews/:id	
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK
Failure Code	403 Forbidden (삭제 권한 없음)

E. Recommendation

a. Get

- Request

Method	GET	
URI	/recommendations/:id	
Parameters	-	-
Header	-	-

- Response

Success Code	200 OK	
Failure Code	404 Not Found (id 에 해당하는 Recommendation 이 없음)	
Success Response Body	recommend	Recommendation category object
Failure Response Body	-	-

b. Search

- Request

Method	GET	
URI	/recommendations	
Parameters	id	Recommendation id
	category	Item category

Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	404 Not Found (검색 조건에 해당하는 Recommendation 이 없음)	
Success Response Body	recommend	Recommendation Object List

c. Write

- Request

Method	POST	
URI	/recommendations	
Parameters	name	Recommendation category name
	description	Recommendation description
	keywords	Recommendation keywords
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	recommendation_id	Created recommendation id
Failure Response Body	message	fail reasons - 본문에 비속어가 있음

d. Delete

- Request

Method	DELETE	
URI	/recommendation/:id	
Header	Authorization	사용자 인증 토큰

- Response

Success Code	200 OK
Failure Code	403 Forbidden (삭제 권한 없음)

7. Database Design

7.1. Objectives

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

7.2. ER Diagram

본 시스템에는 User, Item, Bookmark, Authority, Recommend Category, Keyword, Review Reference, Review 로 총 8 개의 Entity 가 존재한다. 각각의 Entity 는 네모 박스의 형태로 표현되고, Entity 간의 관계는 마름모꼴로 표현된다. 이때 특정 Entity 가 다른 Entity 와 복수의 관계를 가질 수 있을 때는 해당 엔티티쪽에 삼지창 모양 선을 세 개 그어 나타내며, 특정 Entity 가 없어도 되는 경우 Client Entity 쪽에 작은 동그라미를 표시한다. 각 Entity 가 가지는 Attribute 는 타원형으로 표현되는데, 각 Entity 를 식별하는 Unique key 는 라벨에 밑줄을 그어 표시하고, 여러 Attribute 를 허용하는 경우에는 테두리를 이중으로 긋는다.

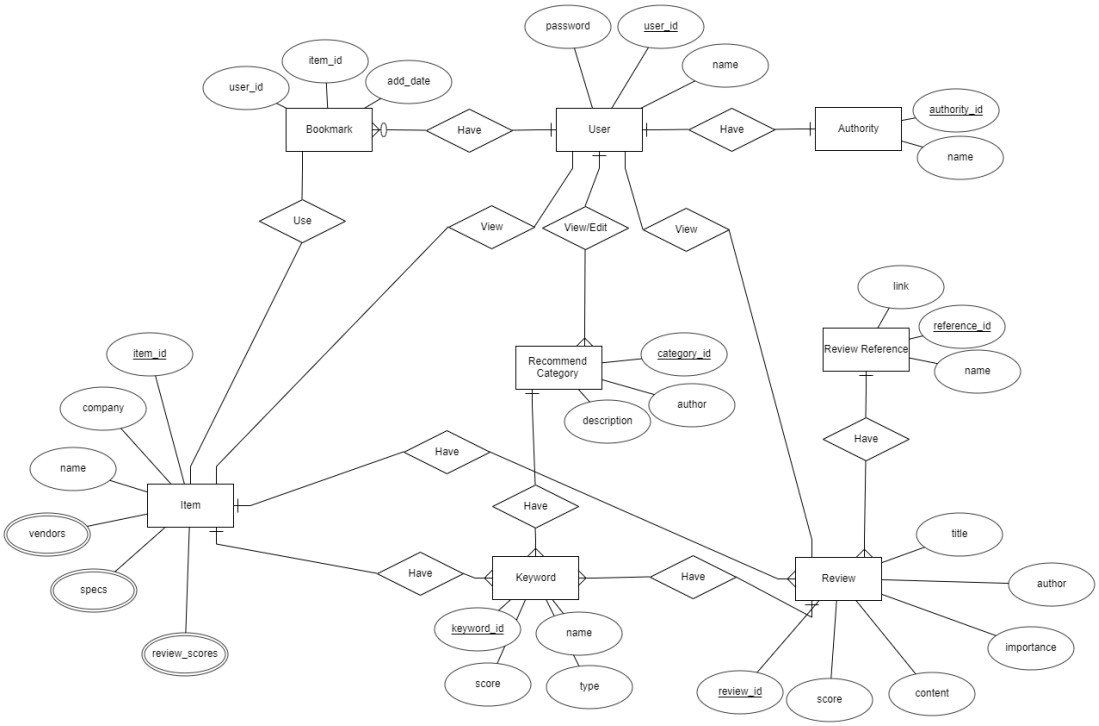


Diagram 16: Overall ER diagram

A. Entities

a. User

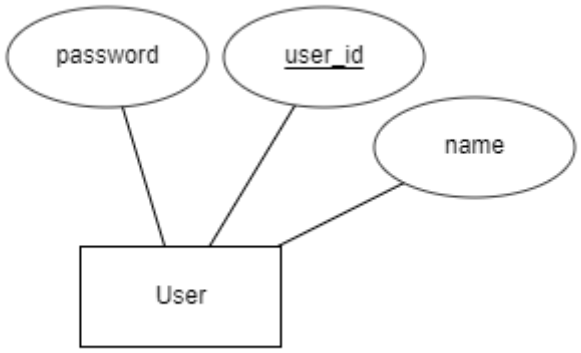


Diagram 17: ER diagram, Entity, User

User Entity 는 사용자의 정보를 표현한다. user_id 속성이 primary key 이며, 이름, 패스워드 정보를 가지고 있다.

b. Bookmark

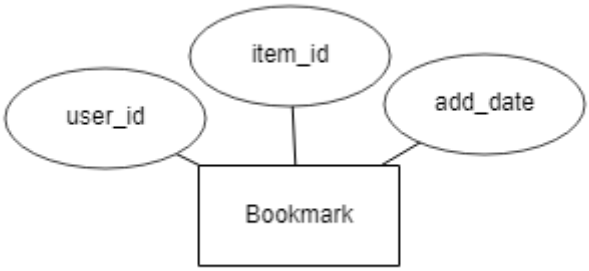


Diagram 18: ER diagram, Entity, Bookmark

Bookmark Entity 는 각 사용자가 Item 에 대해 설정한 즐겨찾기의 정보를 표현한다. Primary key 는 존재하지 않으며, Item entity, User entity 의 primary key 를 조합한 composite key 로 bookmark 를 찾는다. 기타 속성으로는 bookmark 가 추가된 날짜가 있다.

c. Authority

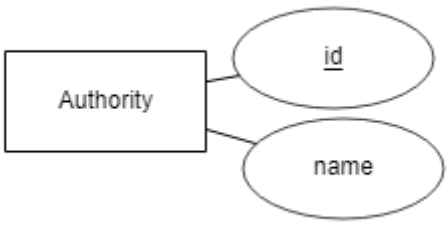


Diagram 19: ER diagram, Entity, Authority

Authority Entity 는 각 사용자가 가질 수 있는 권한 정보를 나타낸다. Primary key 는 id 이며, 해당 권한의 이름인 name 속성이 있다.

d. Item

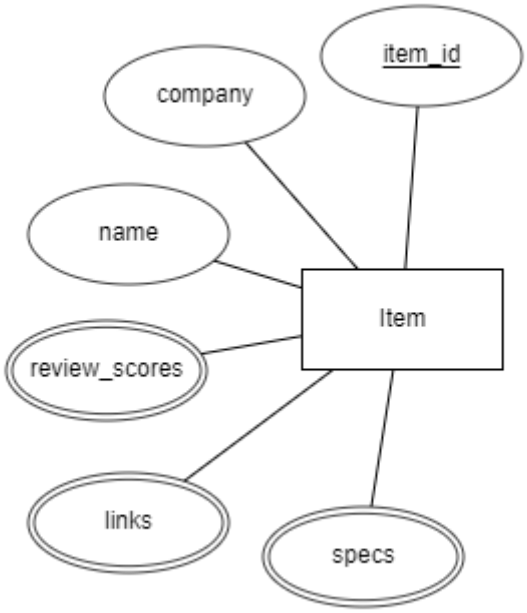


Diagram 20: ER diagram, Entity, Item

Item Entity 는 각각의 상품에 대한 정보를 가지고 있으며, primary key 는 item id 이다. 이외 속성으로는 name, company, specs, links, review_scores 가 있다. specs, links, review_scores 는 여러 개의 값을 가질 수 있다.

e. Recommend Category

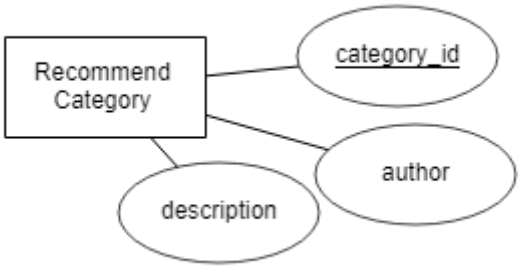


Diagram 21: ER diagram, Entity, Recommend Category

Recommend Category 는 추천 카테고리에 대한 정보를 가지고 있다. primary key 는 category_id 이며, 기타 속성으로는 author, description 이 있다.

f. Keyword

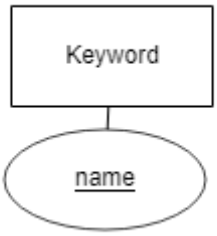


Diagram 22: ER diagram, Entity, Keyword

Keyword Entity 는 각 상품을 묘사하는 키워드에 대한 정보를 가지고 있다. Keyword 는 중복되지 않는 name 속성만 가지고 있다.

g. Review

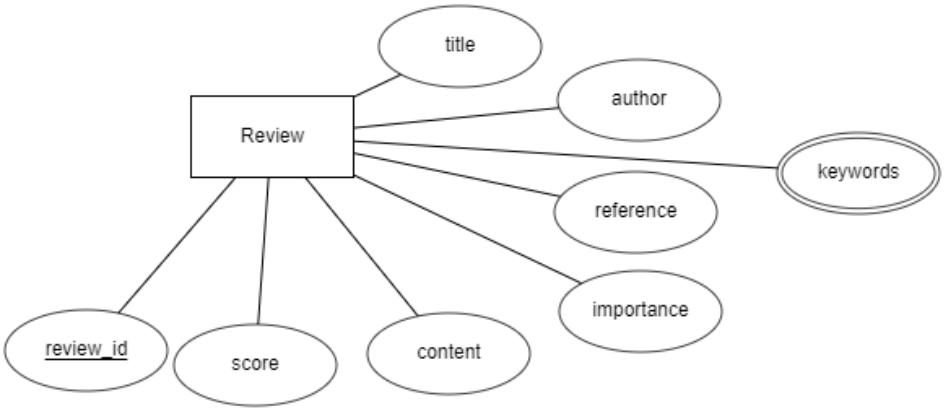


Diagram 23: ER diagram, Entity, Review

Review Entity 는 상품 리뷰에 대한 정보를 표현한다. primary key 는 review_id 이며, 기타 속성으로는 title, author, keywords, reference, importance, content, score 가 있다.

h. Review Reference

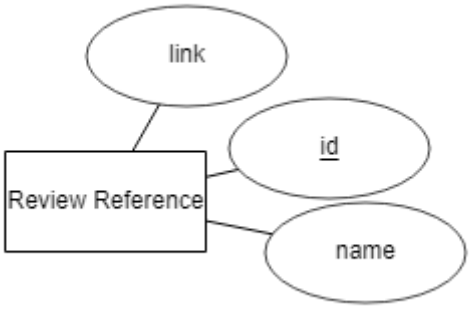
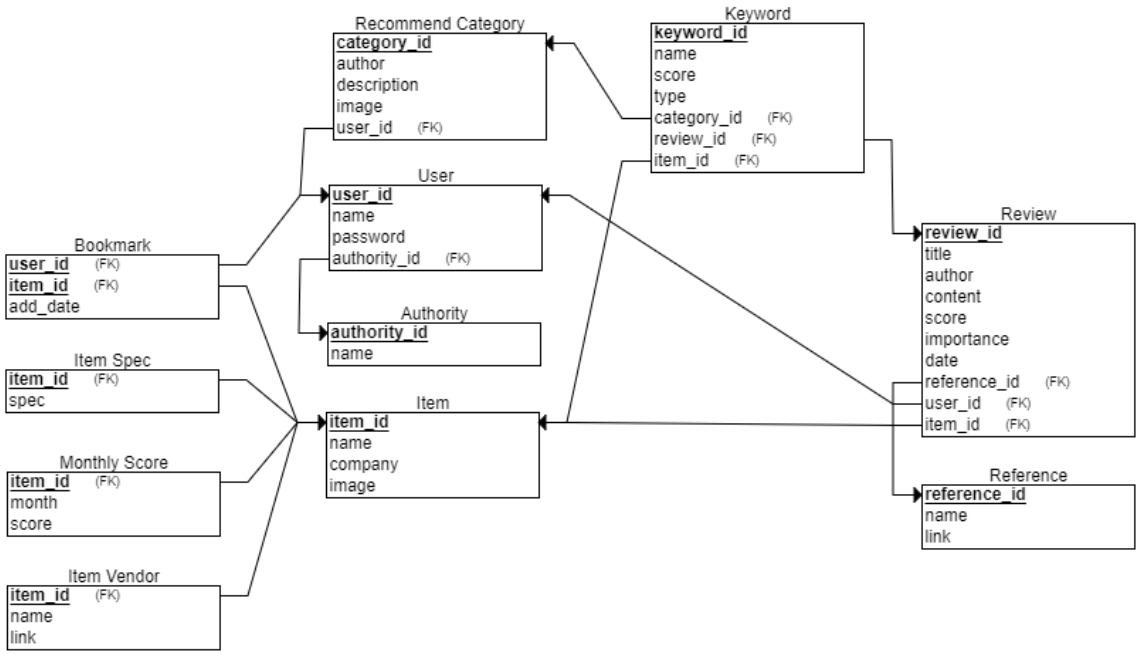


Diagram 24: ER diagram, Entity, Review Reference

Review Reference Entity 는 상품의 리뷰가 작성된 출처에 대한 정보를 나타낸다. primary key 는 id 이며, 기타 속성으로 name, link 가 있다.

B. Relations

7.3. Relational Schema



7.4. SQL DDL

A. Authority

```
CREATE TABLE Authority
(
  authority_id INT NOT NULL,
  name VARCHAR(10) NOT NULL,
  PRIMARY KEY (authority_id)
);
```

B. Item

```
CREATE TABLE Item
(
  item_id INT NOT NULL,
  name VARCHAR(50) NOT NULL,
  company VARCHAR(20) NOT NULL,
  image_src VARCHAR(100) NOT NULL,
  PRIMARY KEY (item_id)
);
```

C. Reference

```
CREATE TABLE Reference
(
  reference_id INT NOT NULL,
  name VARCHAR(20) NOT NULL,
  link VARCHAR(50) NOT NULL,
  PRIMARY KEY (reference_id)
);
```

D. Monthly Score

```
CREATE TABLE Monthly_Score
(
  month DATE NOT NULL,
  score INT NOT NULL,
  item_id INT NOT NULL,
  PRIMARY KEY (item_id),
  FOREIGN KEY (item_id) REFERENCES Item(item_id)
);
```

E. Item Spec

```
CREATE TABLE Item_Spec
(
  spec VARCHAR(100) NOT NULL,
  item_id INT NOT NULL,
  PRIMARY KEY (item_id),
  FOREIGN KEY (item_id) REFERENCES Item(item_id)
);
```

F. Item Vendor

```
CREATE TABLE Item_Vendor
(
  name VARCHAR(20) NOT NULL,
  link VARCHAR(50) NOT NULL,
  item_id INT NOT NULL,
  PRIMARY KEY (item_id),
  FOREIGN KEY (item_id) REFERENCES Item(item_id)
);
```

G. User

```
CREATE TABLE User
(
  user_id INT NOT NULL,
  name VARCHAR(10) NOT NULL,
  password PASSWORD NOT NULL,
  authority_id INT NOT NULL,
  PRIMARY KEY (user_id),
  FOREIGN KEY (authority_id) REFERENCES Authority(authority_id)
);
```

H. Bookmark

```
CREATE TABLE Bookmark
(
  add_date DATE NOT NULL,
  user_id INT NOT NULL,
  item_id INT NOT NULL,
  PRIMARY KEY (user_id, item_id),
  FOREIGN KEY (user_id) REFERENCES User(user_id),
  FOREIGN KEY (item_id) REFERENCES Item(item_id)
);
```

I. Review

```
CREATE TABLE Review
(
  review_id INT NOT NULL,
  title VARCHAR(50) NOT NULL,
  author VARCHAR(10) NOT NULL,
  content VARCHAR(1000) NOT NULL,
  score INT NOT NULL,
  importance INT NOT NULL,
  date DATE NOT NULL,
  item_id INT NOT NULL,
  reference_id INT NOT NULL,
  user_id INT NOT NULL,
  PRIMARY KEY (review_id),
  FOREIGN KEY (item_id) REFERENCES Item(item_id),
  FOREIGN KEY (reference_id) REFERENCES Reference(reference_id),
  FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

J. Keyword

```
CREATE TABLE Keyword
(
  name VARCHAR(50) NOT NULL,
  keyword_id INT NOT NULL,
  score INT NOT NULL,
  type INT NOT NULL,
  category_id INT NOT NULL,
  review_id INT NOT NULL,
  item_id INT NOT NULL,
  PRIMARY KEY (keyword_id),
  FOREIGN KEY (category_id) REFERENCES Recommend_Category(category_id),
  FOREIGN KEY (review_id) REFERENCES Review(review_id),
  FOREIGN KEY (item_id) REFERENCES Item(item_id)
);
```

K. Recommend Category

```
CREATE TABLE Recommend_Category
(
  category_id INT NOT NULL,
  description VARCHAR(200) NOT NULL,
  image_src VARCHAR(100) NOT NULL,
  author_id INT NOT NULL,
  PRIMARY KEY (category_id),
  FOREIGN KEY (author_id) REFERENCES User(user_id)
);
```

8. Testing Plan

8.1. Objectives

8.2. Testing Policy

A. 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.

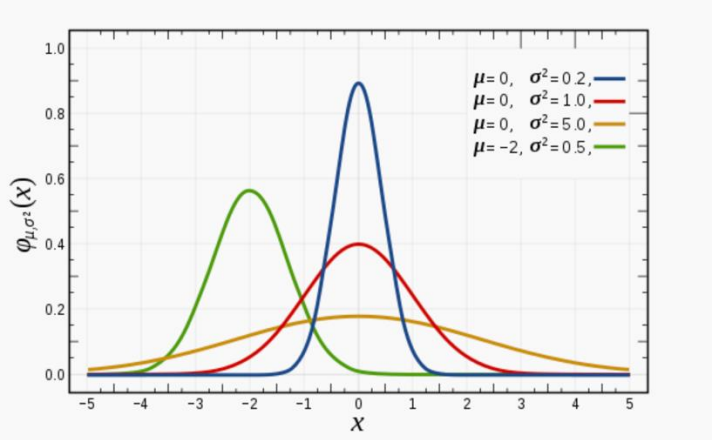


그림 2follow the tendency and check the reliability

2) 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.

3) Performance

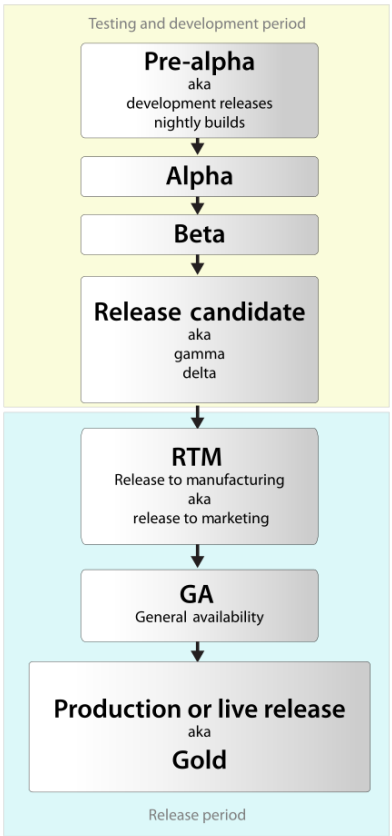
Performance can be evaluated by 1.speed 2. Accuracy 3. Robustness. When we decide Word processing algorithm, we can`t check all information but rather important features. This can be also explained by R-tree concept, and word processing algorithm itself

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.

B. 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.

C. 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.

D. 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.

9. Development Plan

9.1. Objectives

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

9.2. Frontend Environment

A. Vue.js

Figure 8: 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.

B. Ionic

Figure 9: 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.

C. Node.js

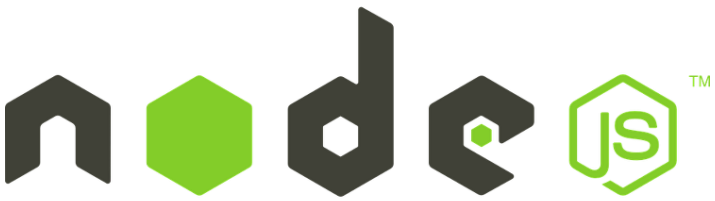


Figure 10: Node.js Logo

Node.js 는 경량 서버 프레임워크로서 다양한 모듈을 활용할 수 있고

9.3. Backend Environment

A. Java

Figure 11: Java Logo

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

B. Apache Tomcat

Figure 12: Apache Tomcat Logo

아파치 톰캣은 아파치 소프트웨어 재단의 어플리케이션 서버로 Java Servlet 을 실행하고 JSP 코드가 포함되어 있는 웹페이지를 만들어 주는 WAS(Web Application Server)의 종류 중

하나이다. WAS 는 웹 서버와 웹 컨테이너의 결합으로 다양한 기능을 컨테이너에 구현하여 다양한 역할을 수행할 수 있는 서버로 웹서버의 backend 부분에서 J2EE 나 J2SE 같은 JSP/Servlet 을 처리해주는 역할을 한다.

C. Spring Framework

Figure 13: Spring Framwork Logo

Spring Framework 는 자바(JAVA) 플랫폼을 위한 오픈소스(Open Source) 어플리케이션 프레임워크이다. Spring Framework 는 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 Model 을 제공한다. 스프링의 핵심요소는 어플리케이션 단위의 인프라를 제공한다는 것이다. 스프링은 기업용 어플리케이션의 plumbing 에 초점을 맞추고 있다. 그래서 개발팀은 특정 배포 환경에서 불필요한 시도없이 어플리케이션의 비지니스 로직에 초점을 맞출 수 있다는 장점이 있다.

9.4. Schedule

Phase	Expected	Actual
Concept & Proposal	**4/11**	**4/11**
Requirements & Design Documentation	**5/5**	**5/12**
Environmental Setting	**5/12**	**5/20**
Front End Development Ranking, Search, Item Detail, Mypage, Recommendation	**5/31**	**5/22**
Backend – Application Server	**5/31**	**6/8**
Backend – Review Analysis	**5/31**	**6/8**
Backend – Review Crawler and Database	**5/31**	**6/5**
System Integration and Testing	**6/9**	**6/10**

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

10. Index

10.1. Tables

10.2. Figures

10.3. Diagrams