<Comparewise >

-Design Specification-

Student Number	Name
2015313266	신한솔
2015312305	박원호
2016311821	한승하
2017314628	오승진
2017314738	나정우
2017312582	신정원

Contents

1. Preface

1.1 Objective

Preface에는 본 문서의 예상 독자층을 선정하고, 각 Chapter에 대해 간략한 설명을 제공한다.

1.2 Readership

본 문서는 Stakeholders를 비롯한 엔지니어, 등 다양한 독자 층을 선정하고 있다.

1.3 Document Structure

A. Introduction

Introduction chapter에서는 본 문서에서 사용된 다양한 다이어그램 및 다이어그램 생성을 위해 사용한 프로그램, 도구들을 소개하고 개발되어질 System이 지향하는 목표 및 전반적인 개발 범위를 기술한다.

B. System Architecture

System Architecture Chapter에서는 개발되어질 System의 Overall Design과 더불어 각 세부적인 Function, Subsystem의 전반적인 System Architecture을 다이어그램 및 도식과 함께 설명한다.

C. Protocol Design

Protocol Design Chapter 에서는 Frontend 와 Backend 간의 상호작용 및 통신을 규정하는 Protocol에 대해 기술한다. 또한 해당 상호작용에 사용된 인터페이스와 사용된 기술 및 서식 규정 등을 제시한다.

D. Database Design

Database Design Chapter에서는 Requirements Specification문서에 작성된 초기버전의 Database을 기반으로 System에 사용되어질 Database들을 ER Diagram으로 표현하고, Relational Schema및 No SQL기반의 Key-value Database Overall을 기술한다.

E. Testing Plan

Testing Plan Chapter에서는 본 개발 작업 이후 Testing을 진행하기 앞서 Testing에 대해 수립된 전반적인 방향성 및 계획과, Testing Policy를 제시한다.

F. Development Plan

Development Plan Chapter에서는 본 개발과정의 전체적인 계획과 앞으로의 개발 계획을 제시한다.

G. Index

Index는 본 문서에 사용된 도식 및 다이어그램과 그림들의 색인을 제공한다.

2. Introduction

2.1 Objective

본 Introduction chapter에서는 개발될 System의 Designing process에 사용된 다양한 다이어그램 및 도구 들을 소개하고, System의 전반적인 개발 범위를 기술한다.

2.2 Applied Diagram

A. UML Diagram

UML Diagram은 Unified Modeling Language로, 통합 모델링 언어를 의미한다. 이는 객체

지향 Software네 특화되어 있으며, System을 문서화, 및 Architecture를 구축하는데 사용되며, System의 전제적인 Workflow를 나타내는데 효율적이다.

UML Diagram은 User부터 Developer까지의 넓은 범위의 Communication을 위해 사용되며, 상황에 따라 다르게 사용되는 여러 종류의 UML Diagram이 존재한다. 이 문서에서 사용되는 Component Diagram, Sequence Diagram, ER Diagram을 사용 및 소개한다.

B. Package Diagram

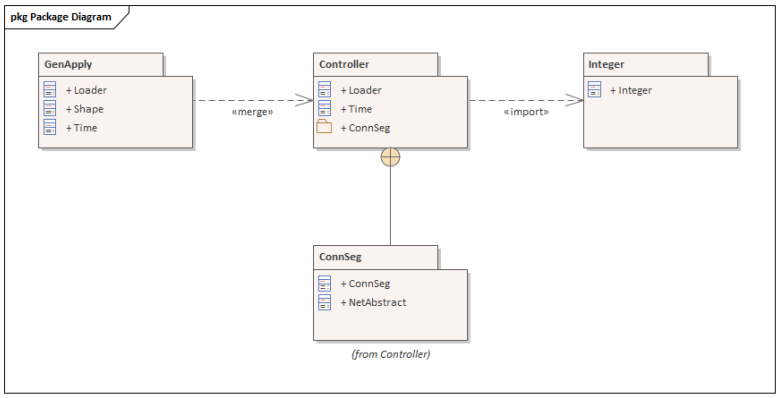

Figure 1. Package Diagram 예시

Package Diagram은 System의 각 요소들을 package단위로 묶어 표현하는 형식의 Diagram이다. 통합 모델링 언어 UML의 일종으로 System을 여러 개의 Subsystem 혹은 Package로 나누고, 각 Package 간의 상호관계를 표현하는데 유리하다.

C. Class Diagram

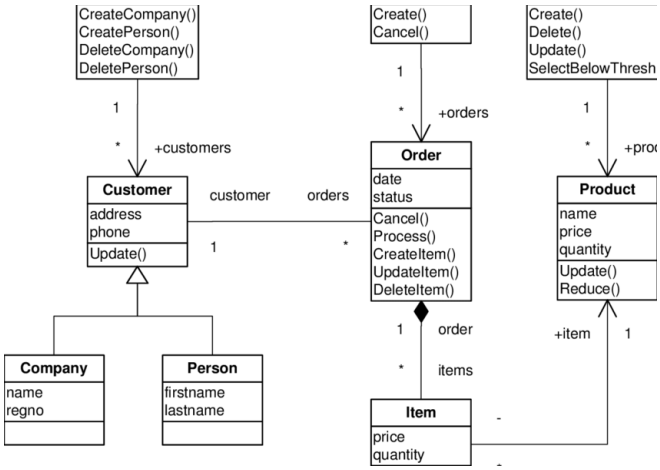

Figure 2. Class Diagram 예시

Class Diagram은 객체지향 System에서 사용되어진 각 객체 Class들을 나타내고, 각 Class 객체가 가지고 있는 Method및 Information들을 표시하며, 서로 다른 객체들 간의 상호관계 및 객체간 상호작용에 사용되는 Interface를 나타낸다.

D. Sequence Diagram

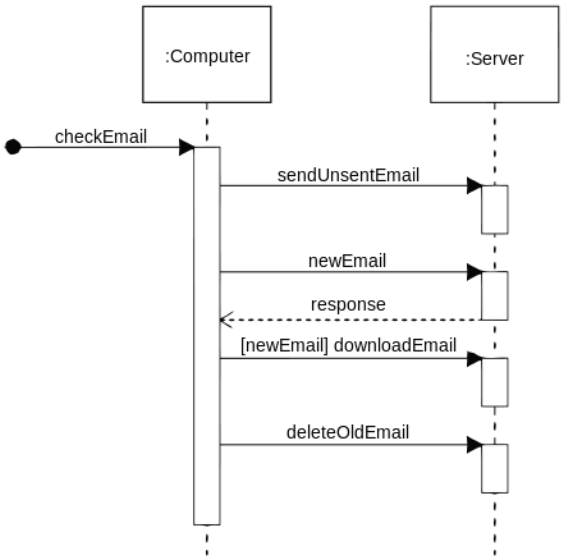

Figure 3. Sequence Diagram 예시

Sequence Diagram은 Event에 따라 변화되는 State들을 한눈에 이해할 수 있도록 그려진 다이어그램이다. 특정 Function혹은 특정 동작에 대한 전체 Sequence를 표시함으로써, 기능 혹은 동작이 이루어지는 과정을 표현할 수 있다.

E. ER Diagram

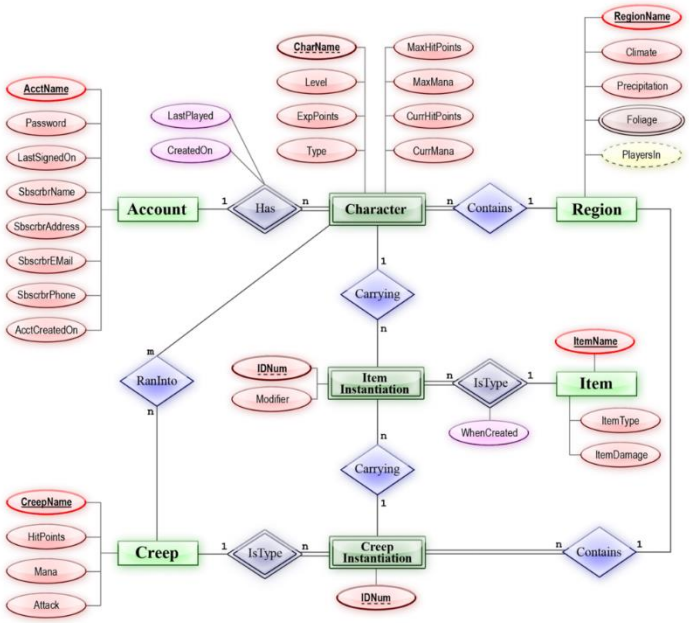

Figure 4. ER Diagram 예시

ER Diagram은 Entity-Relationship Model으로 Entity와 Entity의 속성의 관계를 나타낸 다이어그램이다. 일반적으로 Database를 설계, 기술할 때 사용되며 Relational Schema를 작성하는 베이스로 사용된다.

2.3 Applied Tool

A. Power point

Figure 5. Power Point Logo

Power Point는 MS사에서 제공하는 MS Office에 속해 있는 Tool으로, 주로 Ppt형식의 Presentation자료를 생성하는데 사용된다. Power Point는 매우 강력한 도형 생성 기능을 가지고 있어 Diagram을 그리는데 유용하게 사용될 수 있다.

B. Draw.io

Figure 6. Draw.io Logo

Draw.io는 다이어그램 생성에 특화되어 있는 Tool으로 여러가지 종류의 다이어그램을 생성, 편집하는데 굉장히 편리하다.

2.4 Project Scope

Comparewise는 기존의 커머스 어플리케이션들에 지나치게 많은 상품과 각 상품에 대한 지나치게 많은 데이터들이 등록되어 있어, 상품들을 하나하나 비교하기가 까다로운 문제를 해결하기 위해 소비자에게 강력하고 편리한 비교 UI를 제공하는 것이 궁극적인 목표이다. 개발 기간이 짧으며, 사이즈가 큰 System이 아니라는 점을 감안하여, 구글에서 제공하는 강력한 Backend Tool인 Firebase를 Backend로 사용하며 이에 잘 맞는 Google Cloud Platform 및 Node.js로 Backend를 구성한다, 웹 기반 Application임을 감안하여 PWA와 Vue.js 에 더하여 Facebook의 오픈소스 프로젝트인 React Native를 사용하여 Frontend를 구성한다.

Frontend System은 사용자에게 구매에 필요한 정보들을 제공 및 사용자와의 Communication을 위하여 동작한다. Backend system은 Frontend가 사용자에게 제공하기 위해 필요한 Data요청을 처리하며, 실시간으로 바뀌는 판매 사이트들을 Crawling 하여 제공하는 역할을 담당한다.

3. System Overall Architecture

3.1 Objective

System overall Architecture에서는 전반적인 System의 구조를 소개하며, System의 Sub System들과 각 Sub System간의 관계를 기술한다.

3.2 System Organization

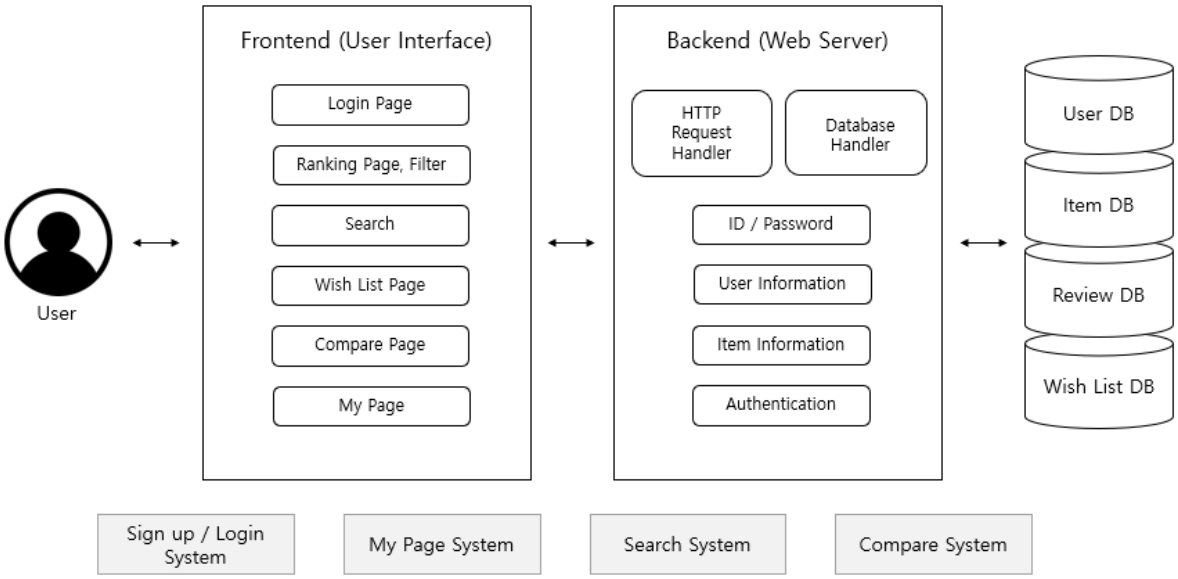

Diagram 1. Overall System Architecture

Comparewise의 Overall System구조는 위와 같다. Web Server기반의 Application이며, 요청과 요청에 따른 처리가 이어지는 System이기 때문에 Client-server Model을 적용하였다. Frontend와 Backend는 JSON을 사용하는 Http통신으로 Data를 교환하며, Backend는 Request Handler에 의해 요청에 따른 처리를 분배하며, Database Handler를 통해 필요한 정보를 Database에서 수집해 Frontend에게 응답한다.

3.3 Sign up / Login System

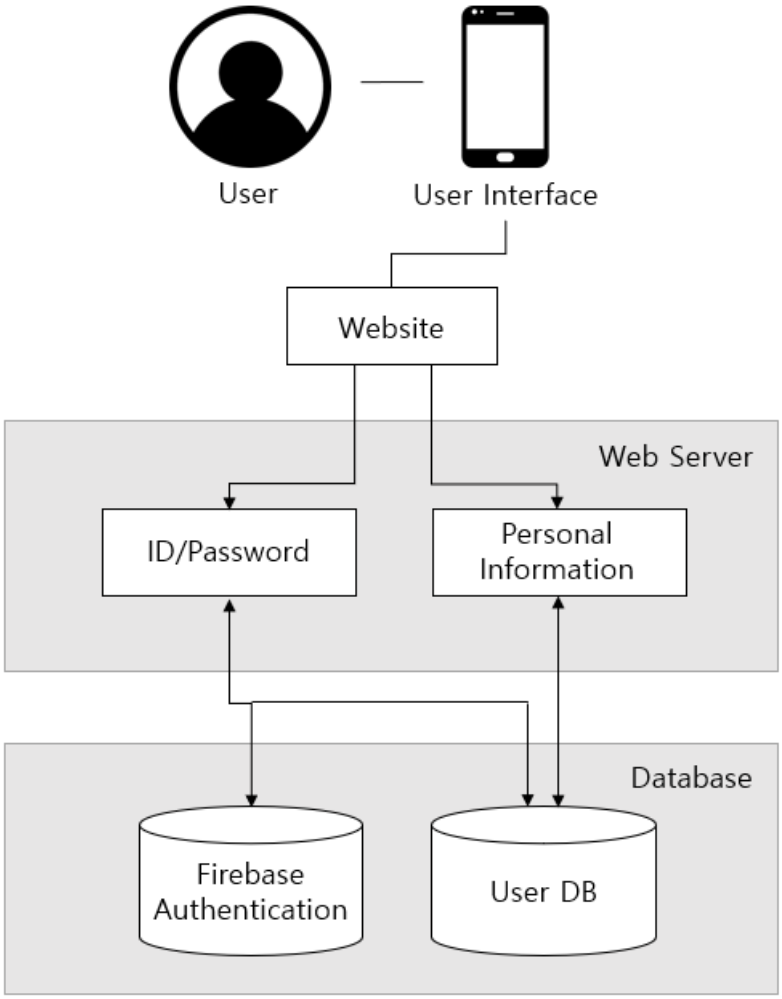

Diagram 2. Sign up / Login System Architecture

사용자의 회원가입부터, 로그인, 로그아웃을 지원하는 시스템이다. 회원가입은 사용자에게 정보를 입력 받아 DB에 가입 이력이 있으면 이를 알리고, 없으면 새로운 계정을 만들도록 지원하는 기능이다. 로그인은 사용자에게 받은 정보를 DB에 검색하여 ID와 Password가 모두 일치하는지 확인하는 기능이다. Firebase의 Authentication SDK를 통해 Google, Facebook, Twitter 등의 사이트와 연동하여 로그인이 가능하다.

3.4 Compare System

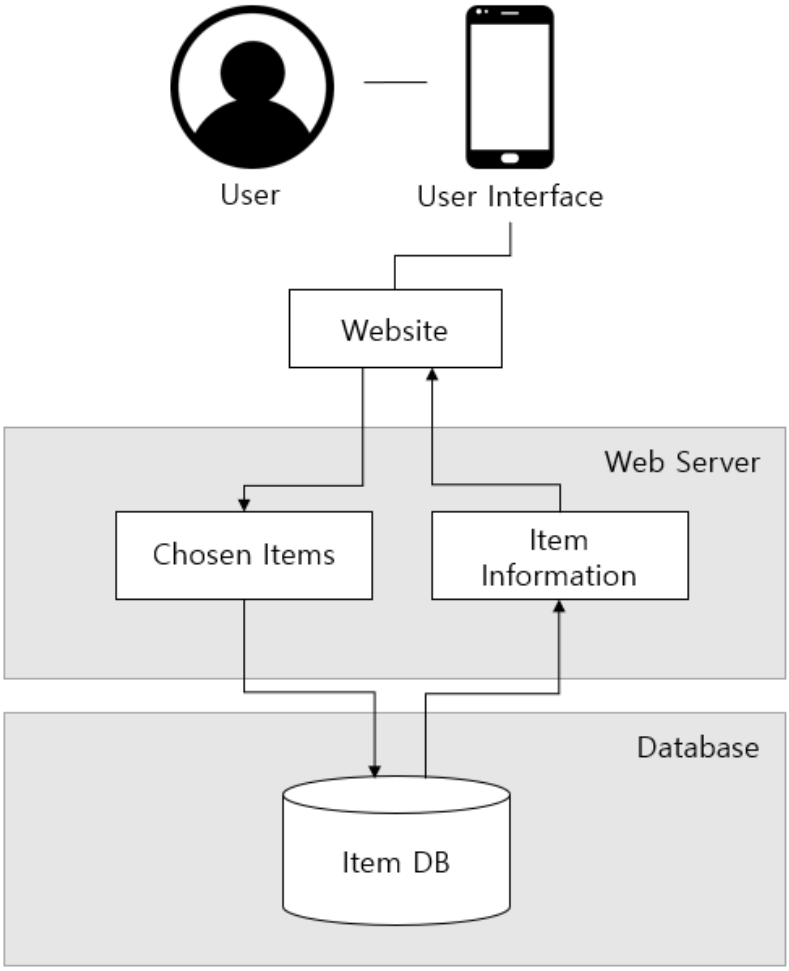

Diagram 3. Compare System Architecture

사용자가 선택한 두 개의 아이템의 정보를 한 눈에 비교할 수 있도록 제공하는 시스템이다. 사용자가 두 아이템을 클릭하여 선택하면, 카테고리 별로 정해진 지표들에 대해 데이터베이스에서 두 아이템의 정보를 찾아 제공한다.

3.5 My page System

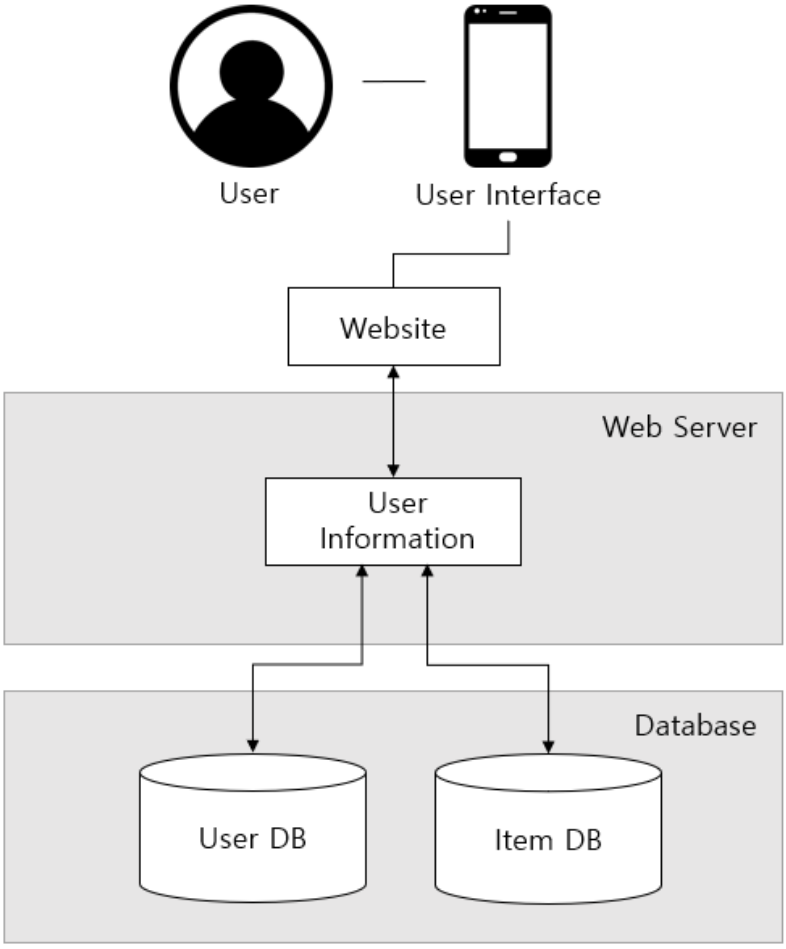

Diagram 4. My Page System Architecture

사용자의 개인 정보와 어플리케이션을 사용하며 유용한 기본적인 정보를 수정 및 확인할 수 있는 시스템이다. 사용자의 로그인 정보를 통해 데이터베이스에서 사용자의 개인 정보와 여러 아이템의 정보를 찾아 제공한다.

3.6 Search System

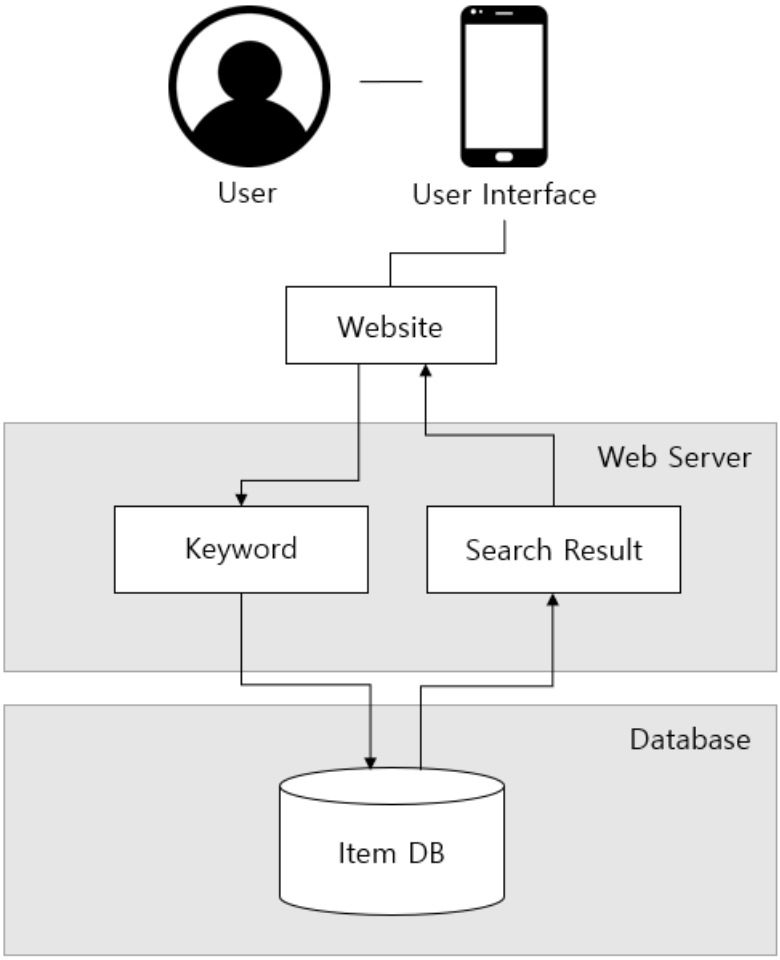

Diagram 5. Search System Architecture

사용자가 원하는 아이템을 검색할 수 있는 시스템이다. 검색하길 원하는 키워드를 입력하면 데이터베이스에서 그에 대한 정보를 찾아 검색 결과를 보여준다.

4. System Architecture – Frontend

4.1 Objective

System Architecture에서 사용자 인터페이스에 해당하는 Frontend 시스템을 이루는 Component들의 구성을 Class Diagram과 Sequence Diagram으로 도식화 및 설명한다.

4.2 Subcomponents

A. Ranking

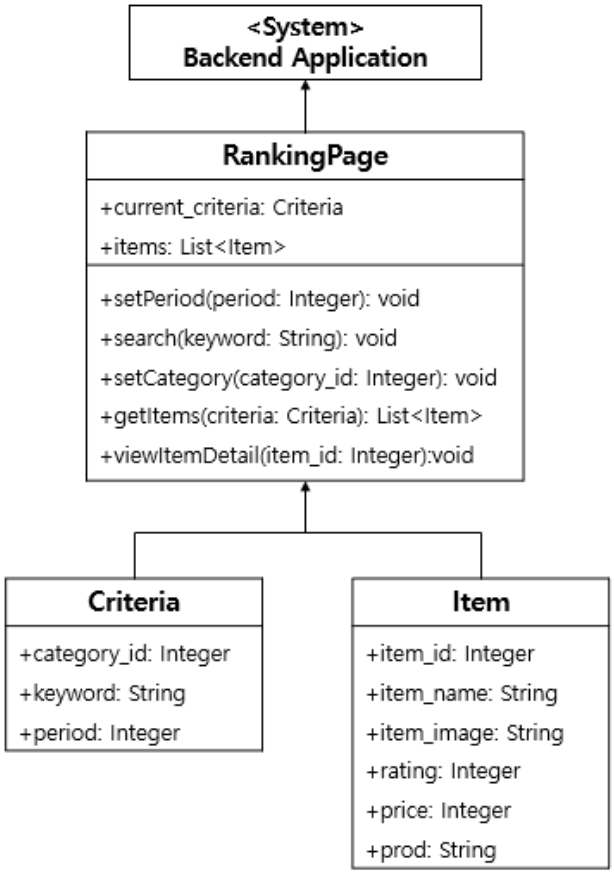

Diagram 6. Ranking Class Diagram

1. Class Diagram

1. RankingPage - 랭킹 페이지 객체

A. Attributes

+ current_criteria : 사용자가 랭킹페이지 상품의 List를 데이터베이스로부터 불러오는데

사용되는 검색 기준

+ items : 검색 조건에 해당하여 데이터베이스로부터 불러온 상품 목록 List

B. Method

+ setPeriod(period: Integer) : 랭킹 보기를 원하는 기간을 설정한다.

+ search(keyword: String) : 키워드로 검색 후 상품들의 랭킹을 보여준다

+ setCategory(category_id: Integer) : 해당 카테고리를 클릭해 설정한다.

+ getItems(criteria: Criteria) : 검색 조건에 해당하는 상품들을 데이터베이스로부터, List 형식으로 부른다.

+ viewItemDetail(item_id: Integer) : 사용자가 선택한 상품의 정보를 조회한다. 이 때, 프론트 엔드는 Item Detail Page 객체로 전환된다.

2. Criteria - 검색조건 객체, DTO(Data Transfer Object)

A. Attribute

+ category_id : 검색하고자 하는 카테고리의 ID

+ keyword : 검색 조건에 사용되는 키워드

+ period : 랭킹 기한을 정하는 Integer(0: 월별 1: 주별 2: 일별)

3. Item - 상품에 대한 정보를 저장하는 객체, DTO(Data Transfer Object)

A. Attribute

+ item_id : 해당 상품의 고유한 ID이다.

+ item_name : 해당 상품의 이름이다.

+ item_image : 해당 상품 이미지의 URL이다.

+ rating : 해당 상품의 리뷰 평점이다.

+ price : 해당 상품의 가격이다.

+ prod : 제조사

2. Sequence Diagram

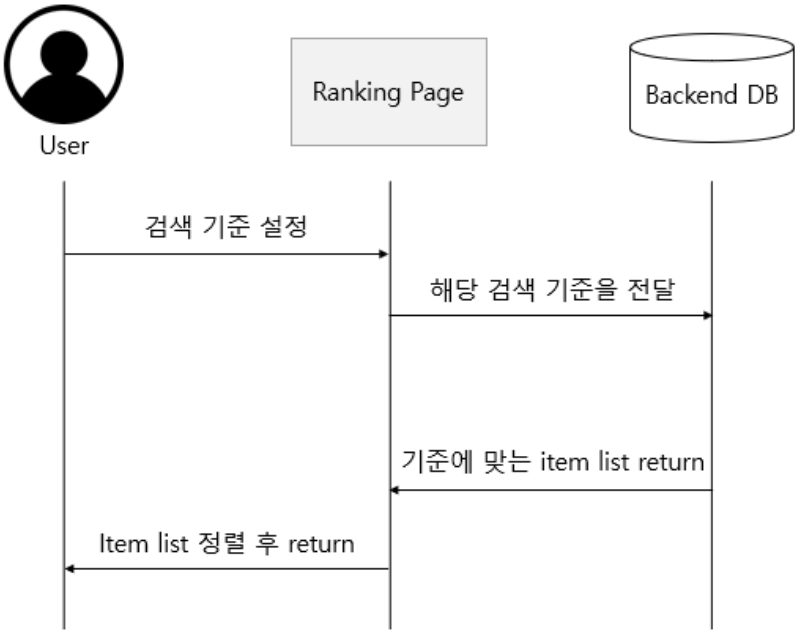

Diagram 7: Ranking, Sequence Diagram

B. ItemDetail

1. Class diagram

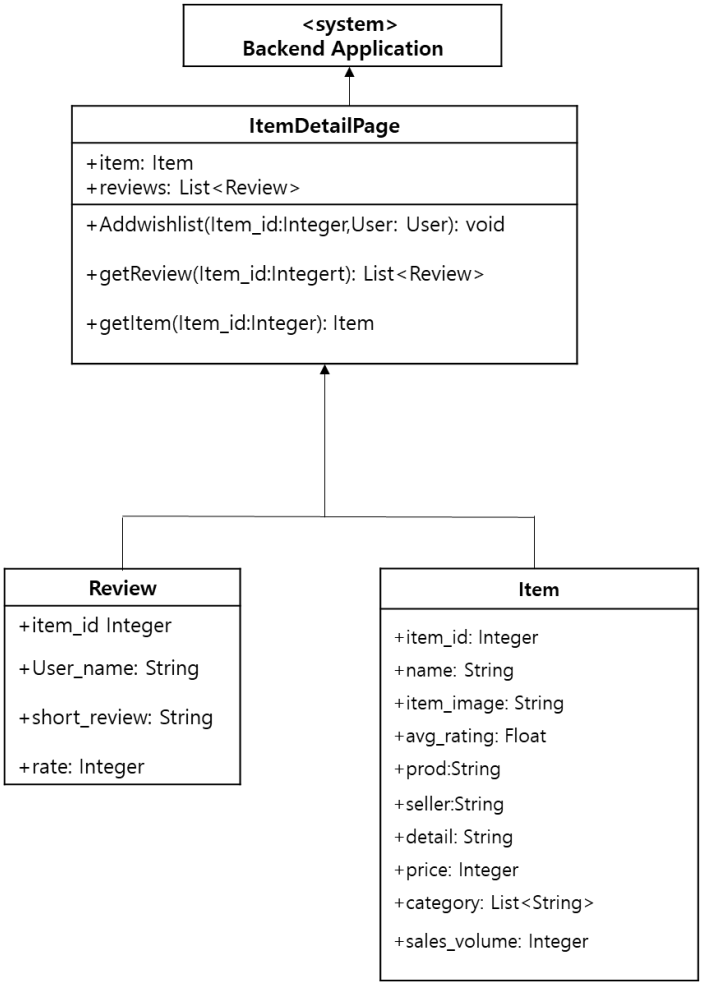

Diagram 8 ItemDetail Class diagram

1.ItemDetailPage-ItemDetailPage 객체

A. Attribute

+Item

+Review: Item에 대한 사용자들의 Review의 List

B. Method

+Addwishlist(Item_id:Integer,User: User):: 사용자의 wishlist에 item을 추가

+getReview(Item_id:Integert > : Item에 대한 Review List를 로드

+getItem(Item_id:Integer):: 해당 Item에 대한 정보를 로드

2. Review

A. Attribute

+item_id Integer :

+User_name: String : review를 등록한 사용자의 이름

+short_review: String : 간단한 Review의 내용

+rate: Integer : 상품에 대한 평점

3. Item-위와 동일

2. Sequence diagram

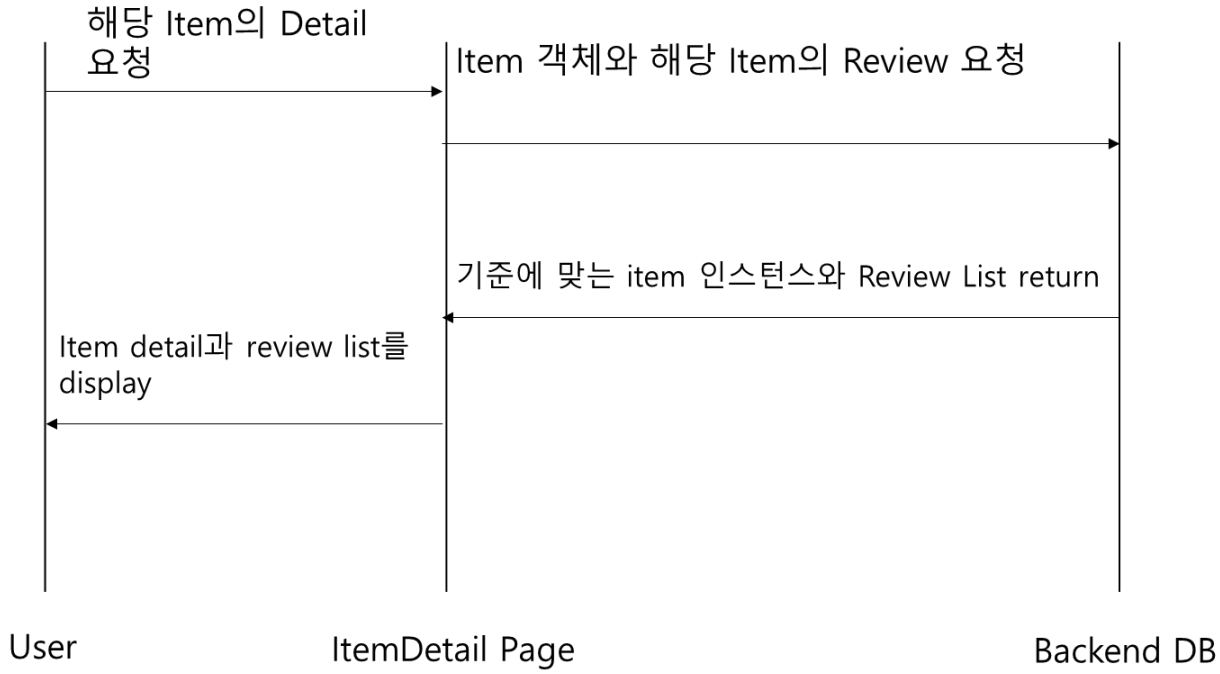

Diagram 9 ItemDetail Sequence diagram

C. MyPage

1. Class diagram

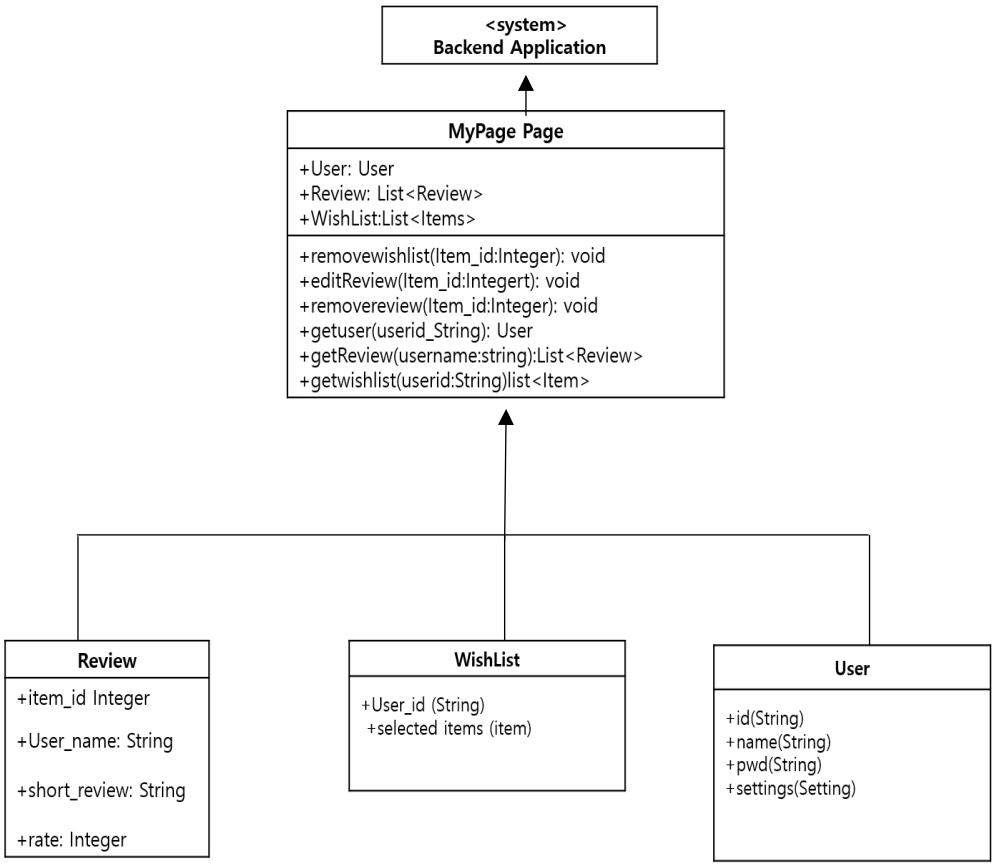

Diagram 10 MyPage Class diagram

1. MyPagePage-Mypage 객체

   A. Attribute:

      +User: 사용자의 정보를 담은 객체

      +Review :사용자가 작성한 Review List

      +WishList: 사용자의 WishList item list

   B. Method

      + removewishlist(Item_id:Integer): item_id에 해당하는 Item을 wishlist를 지운다.

      +editReview(Item_id:Integert): 해당 item_id에 해당하는 Item의 Review를 편집

하는 화면으로 전환

+removereview(Item_id:Integer): 해당 item_id에 해당하는 review를 지운다.

+getuser(user_id: String): 해당 user_id의 user객체를 데이터베이스로부터 load 한다.

+getReview(username:string):User가 작성한 Review객체를 List형태로 load한다.
+getWishList(userid:String):User가 WishList로 등록한 Item들을 List형태로 load 한다.

D. Compare

1. Class Diagram

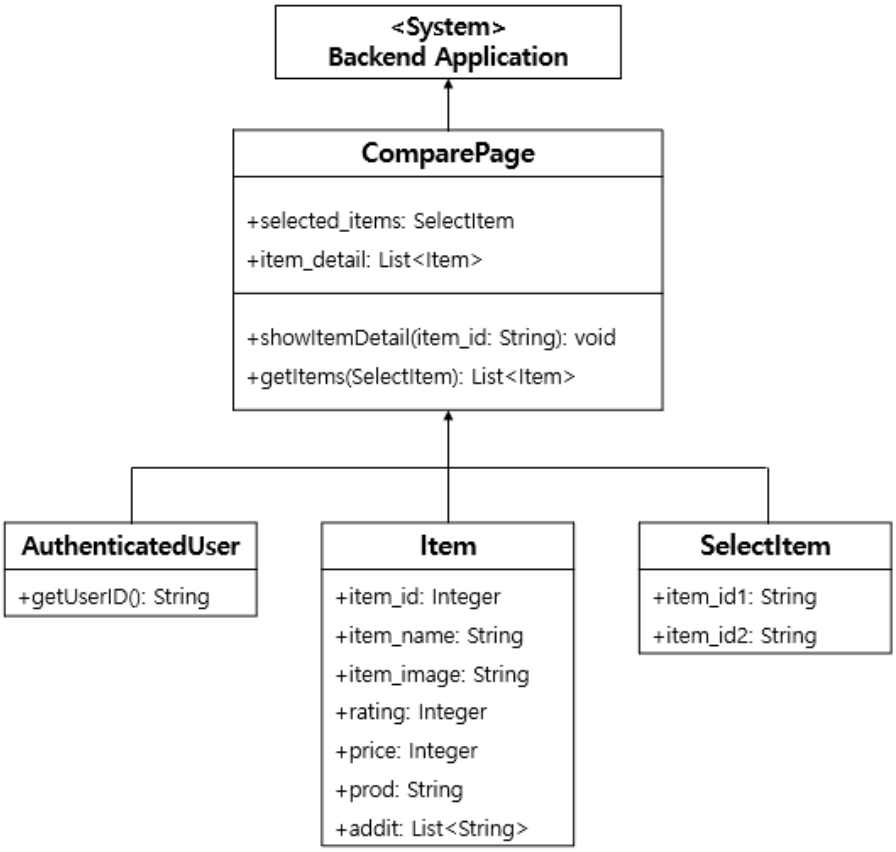

Diagram 11: Compare, Class Diagram

1. ComparePage - 비교 페이지 객체

A. Attributes

+ selected_items : 사용자가 선택한 두 개의 상품명

+ item_detail : 비교를 위한 상품의 세부 정보

B. Methods

+ showItemDetail(item_id: String) : 상품의 세부 정보를 보여주는 창을 연다. 이 때, item Detail Page로 전환된다.

+ getItems(SelectItem) : 비교를 위한 상품의 세부 정보를 얻는다.

2. AuthenticatedUser : 사용자 인증을 위한 객체

3. Item - 상품 객체, Ranking의 상품 정의와 동일, addit attribute만 추가

A. Attributes

+ addit: 카테고리 별로 정해진 Compare를 위한 지표들의 List

4. SelectItem - 상품 선택 객체

A. Attributes

+ item_id1 : 사용자가 선택한 첫번째 상품

+ item_id2 : 사용자가 선택한 두번째 상품

2. Sequence Diagram

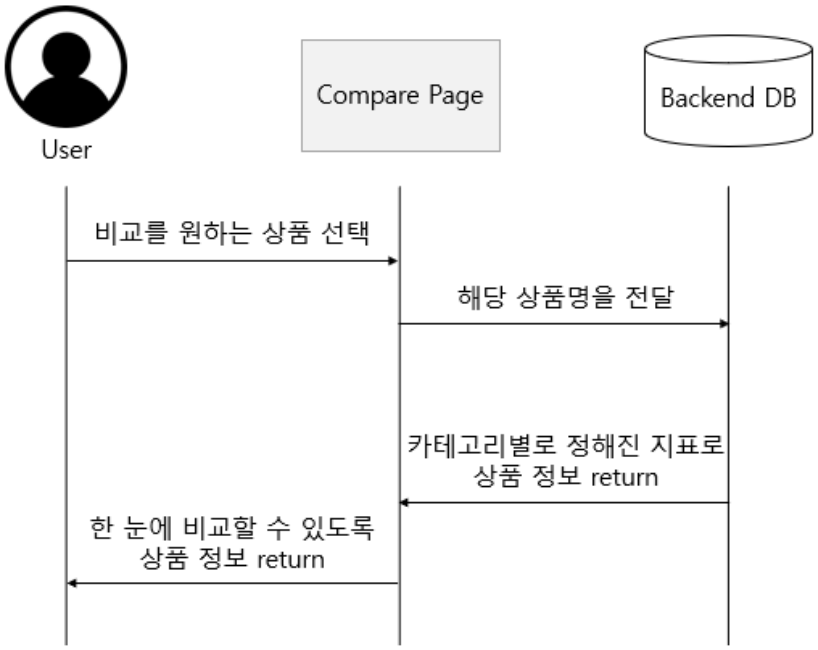

Diagram 12: Compare, Sequence Diagram

5. System Architecture – Backend

5.1. Objectives

System Architecture – Backend에서는 사용자가 직접 사용하는 부분인 frontend를 뺀 Backend의 웹서버에 대한 시스템 및 하위 시스템의 구조를 소개한다.

5.2. Overall Architecture

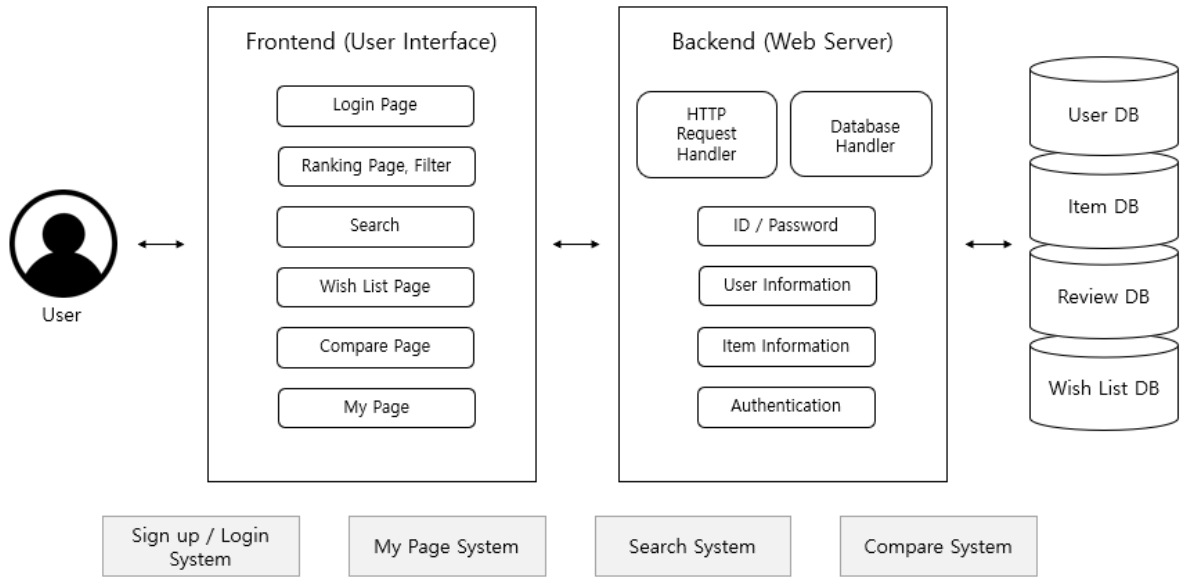

Diagram 13. System Overall Architecture

5.3. Subcomponents

A. Application Server

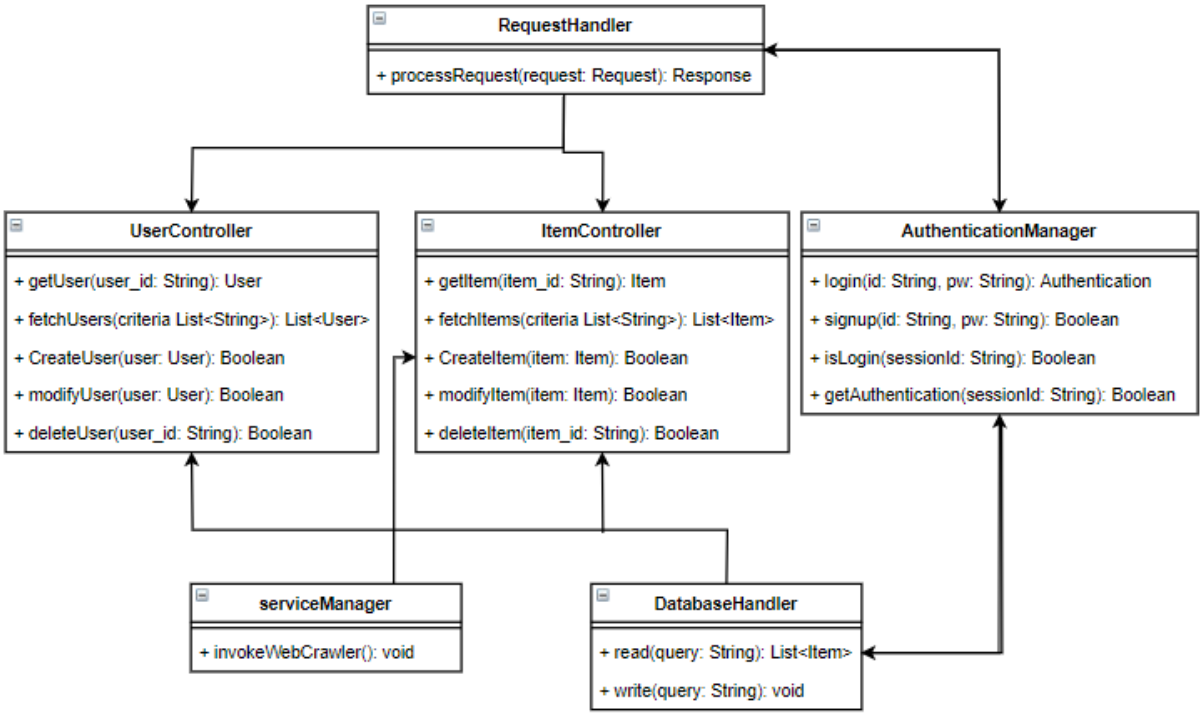

Diagram 14. Application Server Class Diagram

1) RequestHandler: 프론트 엔드로부터의 요청을 처리하는 객체

1-1) processRequest(request: Request): 요청을 각 컨트롤러와 매니저에 알맞게 분배하고 그에 대한 응답을 반환하는 메서드

2) Controllers

2-1) getEntity(entity_id): 엔티티를 가져오는 메서드
2-2) fetchEntities(criteria: List<String>) 검색 조건에 맞는 엔티티의 리스트를 가져오는 메서드
2-3) createEntity(entity: Entity): 엔티티 생성 메서드
2-4) modifyEntity(entity: Entity): 엔티티 수정 메서드
2-5) deleteEntity(entity_id): 엔티티 삭제 메서드

3) Service Manager

3-1) invokeWebCrawler(): 웹 크롤러를 실행하는 메서드

4) Authentication Manager

4-1) login(id: String, pw: String): ID/PW로 로그인하고 Authentication을 반환하는 메서드
4-2) signup(id: String, pw: String): 사용자가 사용할 계정의 ID/PW를 DB에 저장하는 메서드
4-3) isLogin(token: String): 사용자가 Authentication을 확보한 상태인지 확인하는 메서드
4-4) getAuthentication(sessionId: String): 로그인 토큰을 바탕으로 현재 접속한 사용자의 정보를 불러오는 메서드

5) Database Handler

5-1) read(query: String): 데이터베이스 조회 쿼리를 데이터베이스에 질의하는 메서드
5-2) write(query: String): 데이터베이스 입력 쿼리를 데이터베이스에 질의하는 메서드

B. Web Crawling system

1) Class diagram

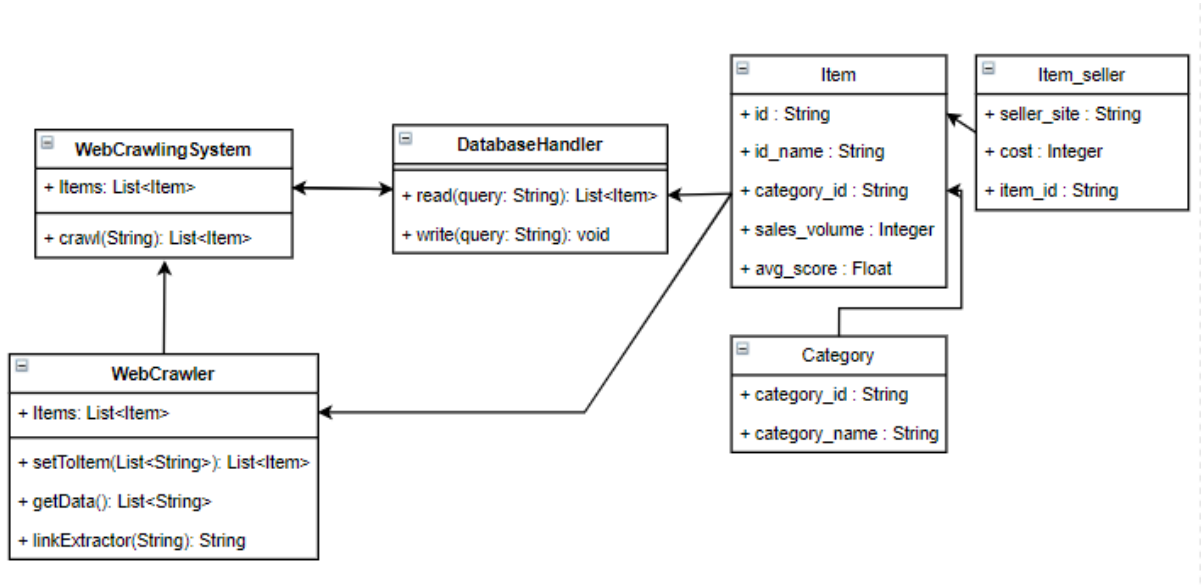

Diagram 15. Web Crawling System Class Diagram

2) Sequence diagram

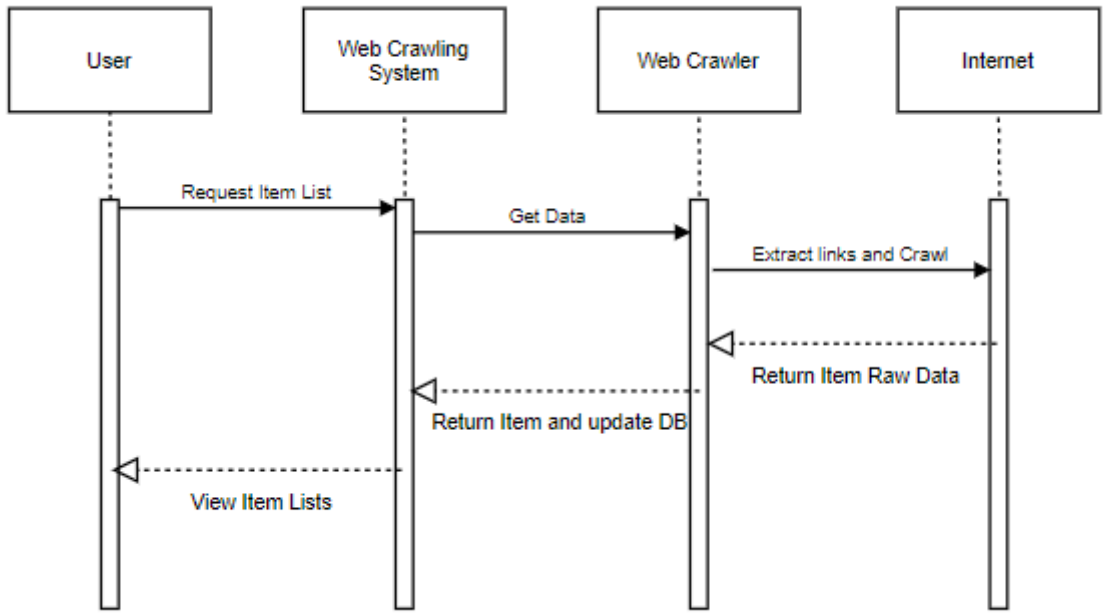

Diagram 16. Web Crawling System Sequence Diagram

C. Item Ranking System

1) Class diagram

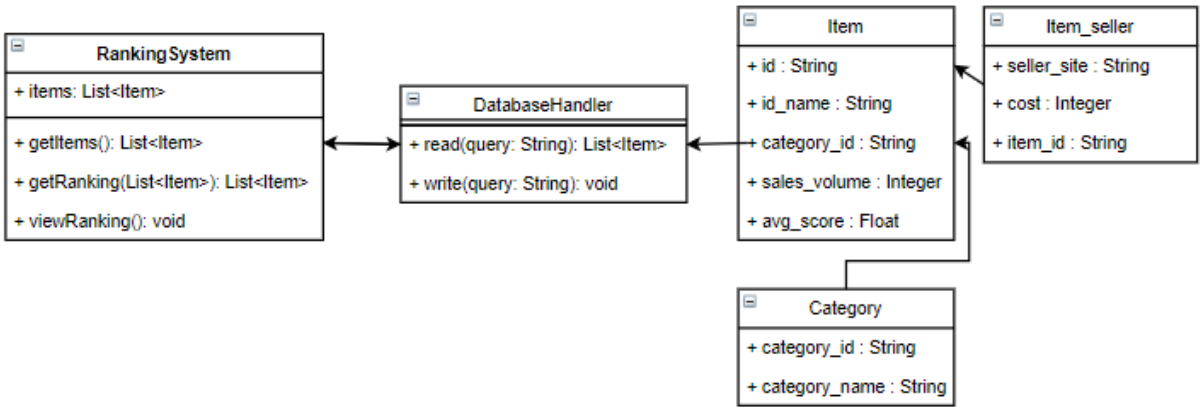

Diagram 17. Item Ranking System Class Diagram

2) Sequence diagram

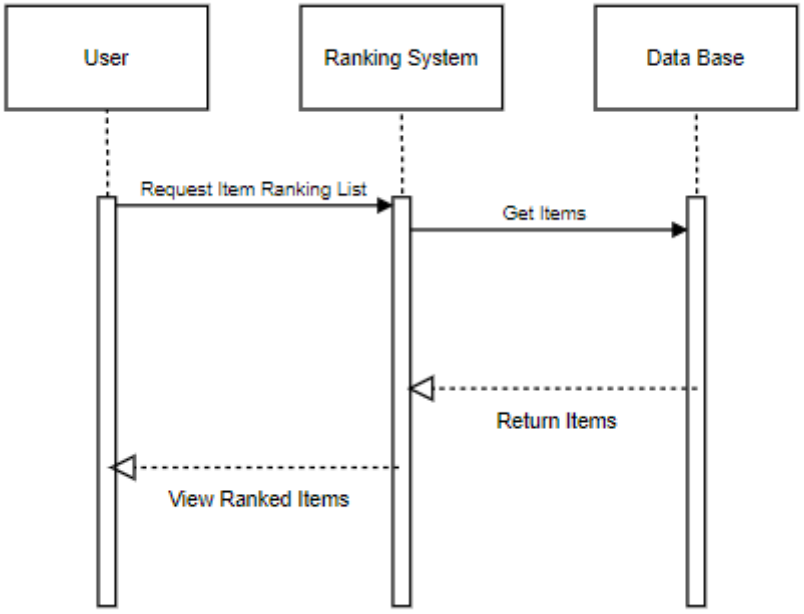

Diagram 18. Item Ranking System Sequence Diagram

D. Compare System

1) Class diagram

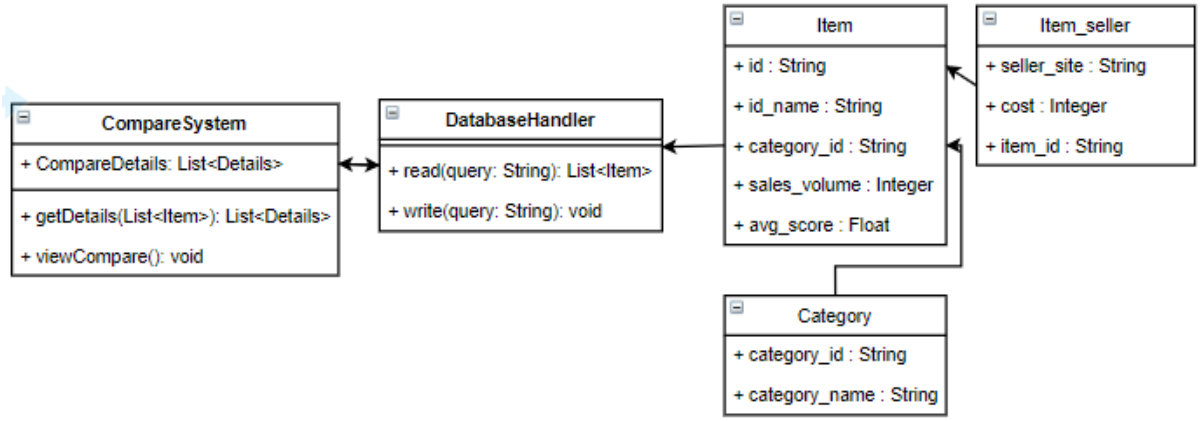

Diagram 19. Compare System Class Diagram

2) Sequence diagram

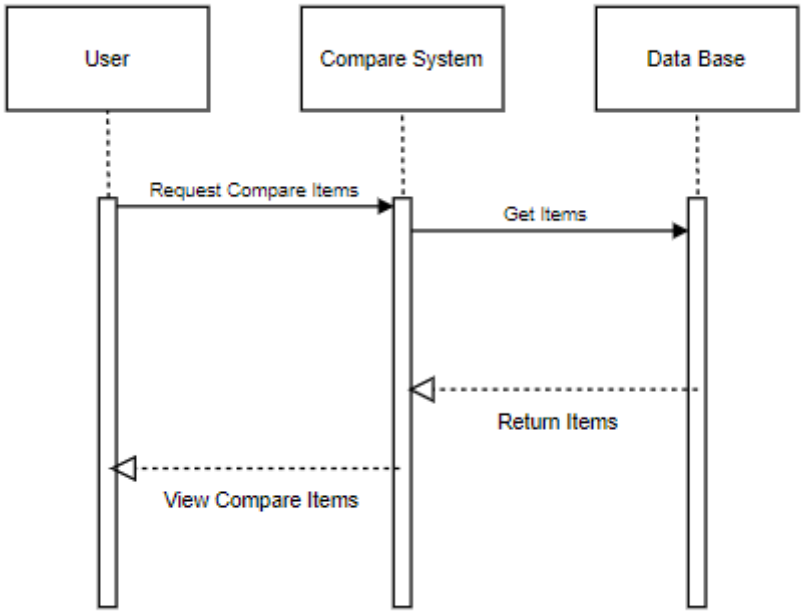

Diagram 20. Compare System Sequence Diagram

6. Protocol Design

6.1. Objectives

Protocol Design 에서는 프론트엔드와 백엔드에 해당하는 서브시스템들의 통신에 이용되는 프로토콜의 구조에 대한 내용을 설명하고, 각각의 인터페이스를 기술한다.

6.2. REST API

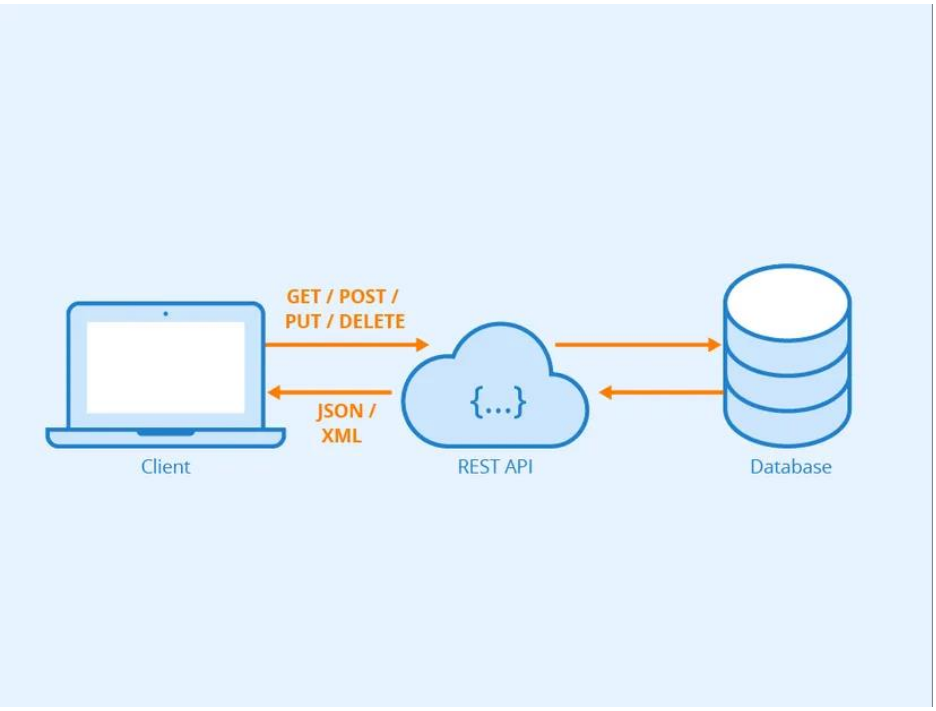

Figure 7. Rest API

본 시스템에서의 프론트 엔드와 벡 엔드의 통신으로 HTTP를 이용하며, 형식은 REST API를 따른다. REST API는 Representational State Transfer"의 약자로, 자원을 이름(자원의 표현)으로 구분하여 해당 자원의 상태(정보)를 주고 받는 모든 것을 의미한다. REST API는 HTTP URI를 통해 자원을 명시하고, POST, GET, PUT, DELETE와 같은 HTTP 메서드를 통해 해당 자원에 대한 CRUD Operation을 적용한다. REST API를 적용하면, HTTP 프로토콜의 인프라를 그대로 사용하므로 REST API 사용을 위한 별도의 인프라를 구출할 필요가 없으며, 서버와 클라이언트의 역할을 명확하게 분리한다.

6.3. JSON

Figure 8. JSON

JSON(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(AJAX가 사용)을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 자료를 주고받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수 값을 표현하는 데 적합하다.

6.4 Details

A. Authentication

1. Login

- Request

Method	Post	
URI	/authentication/login	
Parameter	id	User ID
	password	User Password

Table 1. Authentication Login Request

- Response

Success Code	200 OK
Failure Code	400 Bad Request (ID와 Password 불일치)

Success Response	success	true
Failure Response	success	false
	message	fail reason - ID나 Password가 일치하지 않습니다.

Table 2. Authentication Login Response

2. Signup

- Request

Method	POST	
URI	/authentication/signup	
Parameter	id	User ID
	password	User Password
	name	User name

Table 3. Authentication Signup Request

- Response

Success Code	200 OK	
Failure Code	400 Bad Request (ID 기존에 생성)	
Success Response	message	성공적으로 계정이 생성되었습니다.
Failure Response	message	fail reason - 이미 사용중인 ID입니다.

Table 4. Authentication Signup Response

B. User

1. Mypage

- Request

Method	GET	
URI	/User/:id	
Parameter	user_id	사용자의 id

Table 5. User Mypage Request

- Response

Success Code	200 OK	the data associated with the path in the GET request.
Failure Code	400 Bad Request	

Table 6. User Mypage Response

2. Review - write

-Request

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

Table 7. User Review(write) Request

-Response

Success Code	200 OK	Review Id
Failure Code	400 Bad Request	Fail message

Table 8. User Review(write) Response

3. Review - delete

- Request

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

Table 9. User Review(delete) Request

- Response

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

Table 10. User Review(delete) Response

C. Item

3. Ranking – search

- Request

Method	GET	
URI	/Ranking	
Parameters	type	카테고리인지, 기간별인지
	category	해당 랭킹 대상 카테고리 이름
	period	해당 랭킹 대상 날짜 범위

Table 11. Item Ranking(search) Request

- Response

Success Code	200 OK	List<Item>: 조건에 만족하는 랭킹에 속하는 Item List
Failure Code	404 Not Found(검색 조건에 해당 랭킹이 존재하지 않음)	

Table 12. Item Ranking(search) Response

4. Detail

- Request

Method	GET
URI	/Item/:item_id

Table 13. Item Detail Request

- Response

Success Code	200 OK	Item object: associated with the path in the GET request.
Failure Code	400 Bad Request	

Table 14. Item Detail Response

D. Compare

1. Select

- Request

Method	POST	
URI	/compare	
Parameter	item_id1	First selected item id

	item_id2	Second selected item id

Table 15. Compare Select Request

- Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response	Detail	Call Compare Detail page
Failure Response	message	fail reason - 같은 카테고리의 상품이 아닙니다.

Table 16. Compare Select Response

2. Detail

- Request

Method	GET	
URI	/compare/detail	
Parameter	item_id1	First selected item id
	item_id2	Second selected item id

Table 17. Compare Detail Request

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response	information	Selected item list detail
Failure Response	message	fail reason - 해당하는 상품을 찾을 수 없습니다.

Table 18. Compare Detail Response

7.Database Design

7.1. Objectives

이번 장에서는 세부적인 데이터베이스 설계에 대해 기술한다. NoSQL 기반의 데이터 모델링을 위해 필요한 데이터 객체 및 객체들 간의 관계를 파악하기 위해 ER Diagram과 Relational Schema를 그려 시스템 상의 객체들을 도식화하는 작업을 한다. 또한 Document Key/Value Store 형식의 NoSQL 데이터 구조 활용을 위해 저장되는 Value의 데이터 타입으로 JSON file을 만든다.

7.2. NoSQL database

본 시스템이 NoSQL Document Key/Value Store 구조를 활용한다. 우선 RDBMS가 아닌 NoSQL 형식을 취한 이유는 데이터 모델 자체를 독립적으로 설계하여 데이터를 여러 서버에 분산시키는 것을 용이하게 하고 유연한 스키마(Schema-less) 구조를 취함으로써 다양한 형태의 데이터를 저장하기 위함이다. 또한 NoSQL 형식의 핵심으로 데이터를 중복적으로 저장함으로써 한번에 데이터를 읽는 횟수를 줄이는 것을 목적으로 한다. NoSQL Document Key/Value Store 구조는 기본적으로 다른 NoSQL database 구조와 동일하게 key에 해당하는 value 필드에 데이터를 저장하는 것은 동일하지만 JSON file 형식의 Document에 데이터를 저장함으로써 데이터의 계층적인 구조를 표현하는 것이 가능해진다.

7.3. ER Diagram

본 시스템에는 User, Wish List, Review, Item, Item Seller, Category 총 6개의 Entity가 존재한다. 각각의 Entity는 네모 박스의 형태로 표현되고 해당 Entity의 Key에 해당하는 Attribute들은 네모 박스 안에 '+' 옆에 나타난다. Entity 간의 관계를 표현할 때에는 줄을 그어 표현을 하는데 특정 Entity가 다른 Entity와 복수의 관계를 가질 수 있을 때는 해당 Entity 쪽으로 삼지창 모양의 선을 세 개 그어 표현한다. 무조건적으로 Entity간의 관계가 성립되어야 하는 경우에는 상황에 따라 두 개 혹은 한개의 선을 그어 표시해준다.

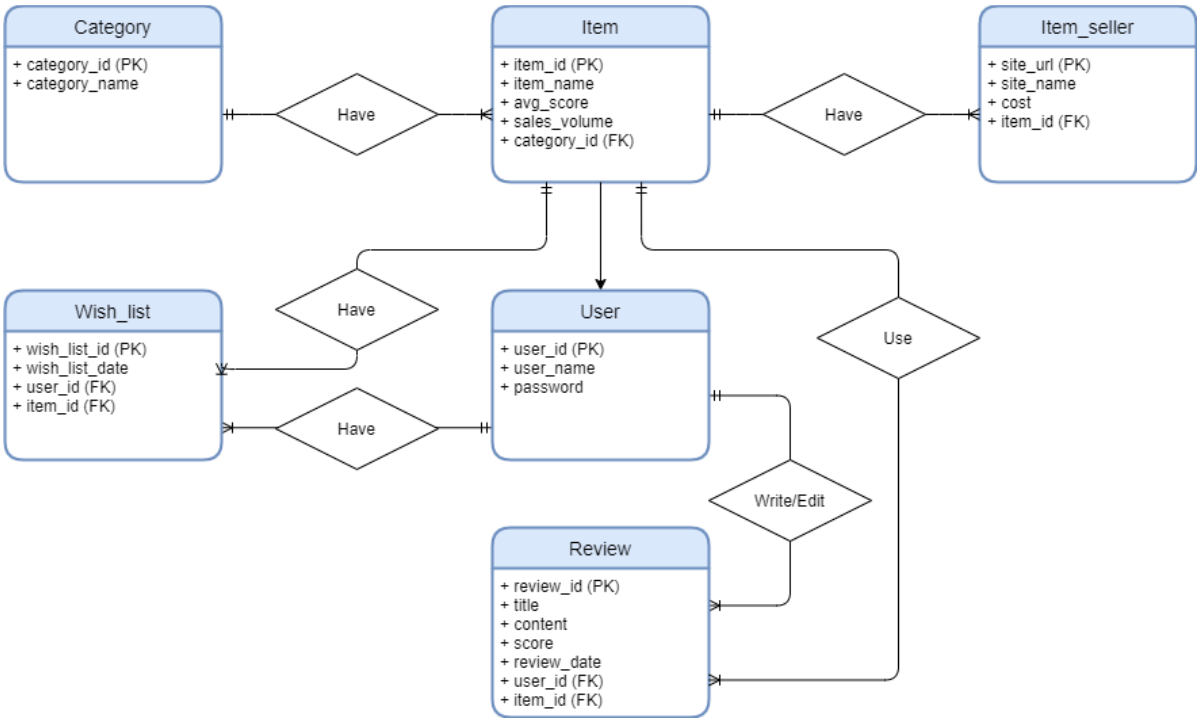

Diagram 21. Overall ER Diagram

<Entities>

A. Item

Diagram 22. Item Entity

Item Entity는 상품의 표준 정보를 표현한다. item_id 속성이 primary key이며 이름, 카테고리의 아이디, 여러 사이트에서 판매되는 해당 상품의 별점 평균, 전체 판매량에 대한 정보를 가지고 있다.

B. Item Seller

Diagram 23. Item Seller Entity

Item Seller Entity는 상품이 판매되는 사이트에서의 상품에 대한 정보를 표현한다. 상품이 판매되는 사이트의 URL 정보가 primary key에 해당한다. 사이트 이름, 상품 가격, 판매되는 상품의 아이디에 대한 정보를 가지고 있다.

C. Category

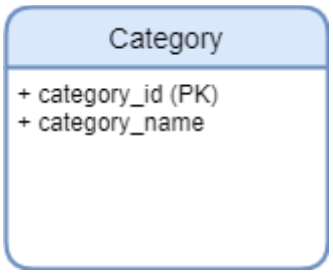

Diagram 24. Category Entity

Category Entity는 시스템 내에서 상품 분류에 쓰일 카테고리 정보를 표현한다. category_id 속성이 primary key에 해당한다. 상품이 분류되는 카테고리의 종류는 상이하기 때문에 상품마다 category_id를 key 값으로 지정해주고 category 종류들을 category_name에 해당하는 value 필드에 데이터로 넣어준다.

D. User

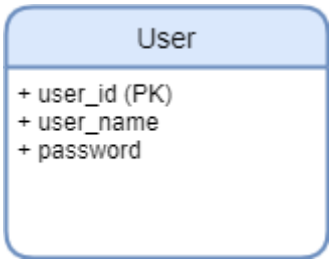

Diagram 25. User Entity

User Entity는 사용자의 정보를 표현한다. user_id 속성이 primary key이며 사용자의 이름, 패스워드 정보를 가지고 있다.

E. Wish List

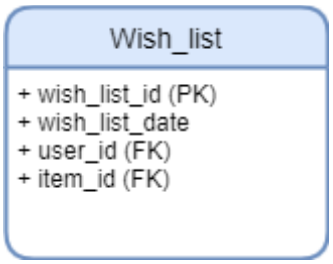

Diagram 26. Wish List Entity

Wish List Entity는 각 사용자가 Item에 대해 설정한 관심 상품의 정보를 표현한다. wish_list 속성이 primary key이며 user_id와 item_id 속성을 foreign key로 하여 각 사용자마다 관심 상품을 목록화 할 수 있게 한다. 관심 상품 등록 날짜에 대한 정보를 가지고 있어 wish_list에 담긴 상품들이 최근에 담긴 순서대로 정렬될 수 있게 할 수 있다. .

F. Review

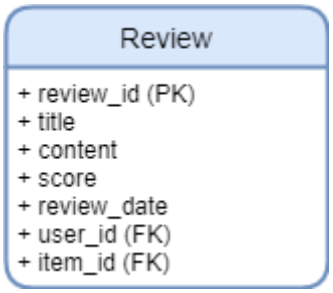

Diagram 27. Review Entity

Review Entity는 상품 리뷰에 대한 정보를 표현한다. review_id 속성이 primary key이며 리뷰 제목, 리뷰 내용, 별점, 리뷰 등록 날짜에 대한 정보를 가지고 있다. Review Entity에 저장된 특정 상품에 대한 별점은 Item Entity에서 해당 상품의 전체적인 평균 평점을 매기는데 활용된다.

7.4. Relational Schema

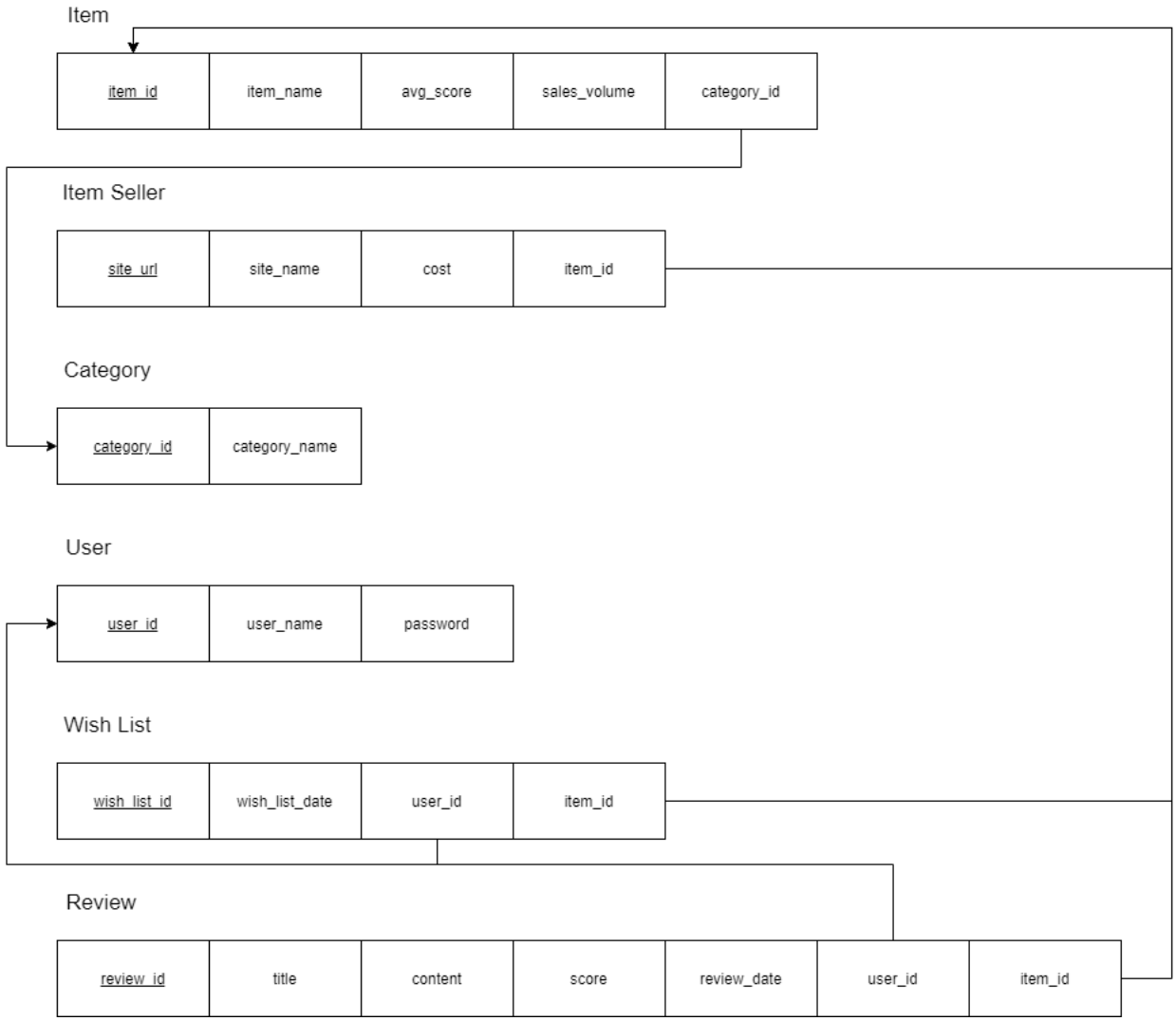

Diagram 28. Relational Schema

7.5. JSON Document

A. Item

```
{
    "item_id" : " ",
    "item_name" : " ",
    "avg_score" : " ",
    "sales_volume" : " ",
    "category_id" : " "
}
```

Diagram 29. Item JSON Document

B. Item Seller

```
{
    "site_url" : " ",
    "site_name" : " ",
    "cost" : " ",
    "item_id" : " ",
}
```

Diagram 30. Item Seller JSON Document

C. Category

```
{
    "category_id" : " ",
    "category_name" : " ",
}
```

Diagram 31. Category JSON Document

D. User

```
{
    "user_id" : " ",
    "user_name" : " ",
    "password" : " "
}
```

Diagram 32. User JSON Document

E. Wish List

```
{
    "wish_list_id" : " ",
    "wish_list_date" : " "
    "user_id" : " ",
    "item_id" : " "
}
```

Diagram 33. WIsh List JSON Document

F. Review

```
{
    "review_id" : " ",
    "title" : " ",
    "content" : " ",
    "score" : " ",
    "review_date" : " ",
    "user_id" : " ",
    "item_id" : " "
}
```

Diagram 34. Review JSON Document

8. Testing Plan

8.1 Objective

본 장에서는 개발되어질 System의 Testing 계획을 서술하고, 각 항목에 대한 Testing의 방향성을 제시한다.

8.2 Testing Policy

A. Development Testing

Development Testing은 개발되어질 system의 전체적인 Synchronization 및 오류 발견과 예방에 초점을 맞추어 개발 과정에서 일어날 수 있는 오류들을 방지하는 것을 목표로 한다. 이를 위해 코드 검토, Data flow 및 사용되어지는 Metrics에 대한 검토와, Peer Code Review등을 계획한다. 또한 Reliability, Security, Performance과 같은 Non functional requirements에 초점을 맞추어 Testing을 진행한다.

1) Reliability

Reliability는 전체적인 System의 신뢰성으로 Comparewise의 경우 판매처 및 어플리케이션에서 제공되어지는 Product에 대한 정보가 늘 최신 정보로 Up to date 되어 있어야 한다. 어플리케이션에 적용되는 정보들이 신뢰 가능한 사이트에서 기인되었는지에 대한 검토와, 실시간으로 올바르게 Crawling 되어지고 있는지에 대한 검토가 요구된다.

2) Security

Comparewise는 실질적인 구매가 진행되는 어플리케이션이 아니기에 실질적인 구매를 지원하는 어플리케이션만큼의 높은 보안수준은 요구되지 않으나, 고객들이 가입시 작성한 개인정보의 보안에 신경을 써야 한다. DB에 대한 접근 권한과, 고객의 ID/PW에 대한 암호화에 대한 검토가 요구된다.

B. Release Testing

System의 배포는 점진적인 방향으로 진행되어야 한다. 하지만 개발 및 Testing 기간이 짧은 것을 감안하여 초기 배포 버전에 대한 Testing만을 진행하는 것을 목표로 한다.

C. User Testing

실질적인 System사용에 있어 사용자의 입장에서 일어날 수 있는 오류들을 검토하여야 한다. 개발 이후 사용자의 입장에서 어플리케이션 사용시 일어날 수 있는 시나리오들을 가정하고, 각 시나리오들의 진행과정에서 오류가 없는지 검토하는 방식으로 이루어진다.

D. Testing Case

개발 기간이 촉박한 관계로 많은 Testing Case를 적용하는 것엔 무리가 있다. 본 Testing Plan은 실제보다 많이 축소된 크기인 50명의 유저와 카테고리별 3개의 제품을 가정하여 Testing을 진행한다.

9. Development plan

9.1 Objective

본 장에서는 실 계발 단계에서 사용되어질 개발 환경 및 Tool들을 소개하고, 개발 일정과 진행상황을 기록한다.

9.2 Frontend Environment

A. React Native

Figure 9. React Native

React는 굉장히 광범위한 웹 및 웹 기반 어플리케이션에서 사용되는 강력한 Frontend 라이브러리이다. React Native는 Facebook에서 제공하는 Opensource Mobile Application Framework로서, 기존 Platform기능에 더하여 React를 사용할 수 있도록 만들어진 Framework이다. Android및 IOS 어플리케이션 개발에 사용된다.

B. PWA

Figure 10. PWA

PWA (Progressive Web Applications) 는 Web을 기반으로 하는 어플리케이션들에 사용되는 어플리케이션들 이다. Native Application들과 다르게 URL을 통한 쉬운 접근이 가능하므로 사용자가 사용하기에 훨씬 편리하다는 장점이 있다.

C. Vue.js

Figure 11. Veu.js

Vue.js는 Web기반 어플리케이션들의 UI를 만들기 위해 사용되는 Java Script기반의 오픈소스 Framework이다. 개발자들이 익히기가 쉽다는 장점을 가지고 있으며 PWA및 Node.js와의 연동성이 좋다.

9.3 Backend Environment

A. Firebase

Figure 12. Firebase

Firebase는 구글에서 제공하는 Native및 PWA 개발 플랫폼이다. Firebase는 Application의 Backend를 구성하는 Authentication및 NoSQL기반 Key-value Database등을 제공하고 있어, JSON 형식의 Data교환이 편리하다. Server측 Infrastructure를 제공하여, 소규모 개발작업에서 Server구성에 드는 부하를 줄여주며, Realtime Database를 제공한다.

B. Google Cloud Platform

Figure 13. Google Cloud Platform

Google에서 제공하는 Cloud Computing Platform이다. Data상호작용 및 Categorization 상세화 및 Entity Analyze에 사용될 계획이다.

C. Node.js

Figure 14. Node.js

Java Script 기반의 Server Framework이다. 네트워크 기반 어플리케이션 개발에 특화되어 있는 Software Framework으로 Event-base Non-blocking I/O 와 Single thread Event loop을 통한 높은 처리성능을 가지고 있다.

9.4 Schedule

Proposal	5/3
Requirement Specification	5/13
Design Specification	5/24
Frontend Development	5/28
Backend Development	6/3
Testing and Review	6/7

Table 19. Testing Schedule

10. Index

10.1 Table

10.1 Figure

10.1 Diagrams