책가방

전공서적 중심 도메인별 중고 거래 플랫폼

Design Specification

Team 4

Student Number	Student Name
2016310249	한 채 정
2017315482	문 원 준
2014314592	서 주 원
2015310280	주 원 영
2013310725	송 현 욱
2017315116	담딘바자르

Content

1. Preface

이 챕터에서는 이 문서를 읽을 예상 독자를 정의하고, 각 챕터에 대해 간단히 소개한다. 또한 각 목차의 목적 및 버전 관리 정책, 버전 변경 기록, 그리고 문서의 변경사항들과 변경에 대한 근거 등도 이 챕터에서 기술한다.

1.1 Readership

본 문서의 독자는 시스템을 개발하는 소프트웨어 엔지니어들과 시스템 설계 및 개발에 관여하는 모든 이해 관계자들이다. 만약 해당 시스템 개발을 위해 외부 업체가 참여한다면, 해당 업체의 개발 참여 인원 역시 독자에 포함한다.

1.2 Document Structure

A. Introduction

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

B. System Architecture

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

C. Protocol Design

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

D. 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.

E. Testing Plan

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

F. Development Plan

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

G. Index

본 문서에서 사용된 그림, 표, 다이어그램 등의 색인을 기술한다.

2. Introduction

2.1 Objectives

In this chapter, the tools and diagrams used in the System Design Specification are generally explained and introduced.

2.2 Applied Diagrams

A. Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a general-purpose, developmental artificial

language (modeling language) used to indicate system structure defined by a consistent set of rules to visualize the design of the system. It's integral part of the system design specification to visualize the behavior of the system using in both Static and Dynamic views: emphasizing the static structure of the system using objects, attributes, operations and relationships, and emphasizing the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects, respectively.

The following diagrams are used to ensure the factor of communication between developers and users using a broad scope of definitions and symbols:

Package Diagram, Deployment Diagram, Class Diagram, State Diagram, Sequence Diagram, and ER Diagram.

B. Package Diagram

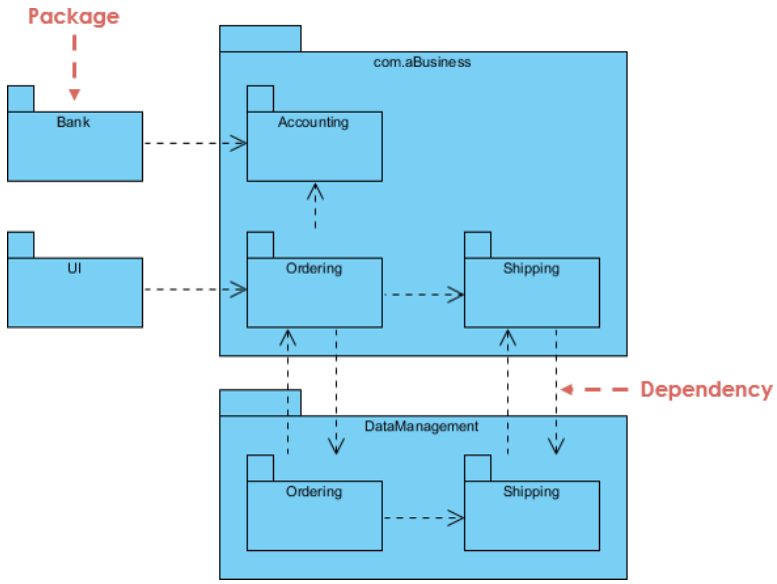

Figure 1 Package Diagram Example

Package diagram is a sort of structure diagram showing the arrangement and organization of model elements. Package diagram can show both structure and dependencies between sub-systems or modules showing different views of a system. Package diagram can be used to simplify complex class diagrams, it can group classes

into packages. A package is a collection of logically related UML elements.

C. Deployment Diagram

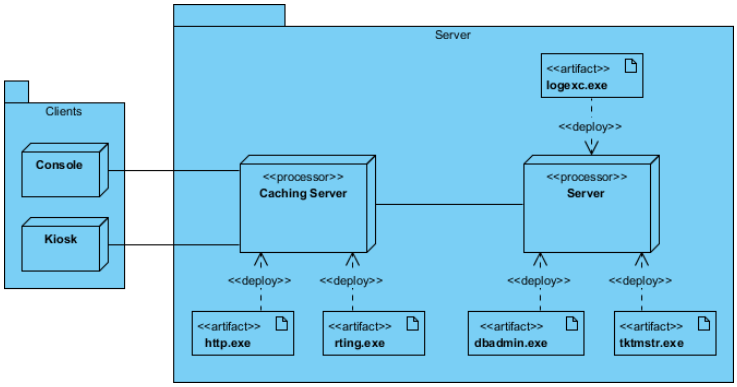

Figure 2 Deployment Diagram Example

Deployment Diagram is a sort of structure diagram used in modeling the physical aspects of an object-oriented system. Deployment diagram shows the configuration of tun time processing nodes and the components that inhibit on them. They are usually used to model the static deployment of a system. They are crucial for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering. The diagram is a collection of vertices and arcs.

D. Class Diagram

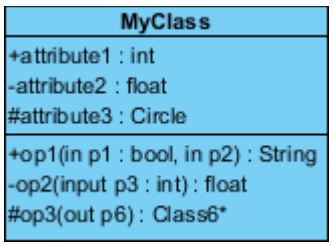

Figure 3 Class Diagram Example

Class diagram is a sort of static structure diagram describing the structure of a system by showing the system's classes, their attributes, operations (or methods), and the

relationships among objects display a clear abstract among every class and the hierarchy and dependency. The diagrams are interpreted as describing software abstractions or components with specifications and interfaces.

E. State Diagram

State Diagram is used to represent states of an object, which have behaviors and states, in a class and edges represent occurrences of events, and the states depend on their current condition. State Diagram shows the possible states of the system and the transitions causing change in the state.

F. Sequence Diagram

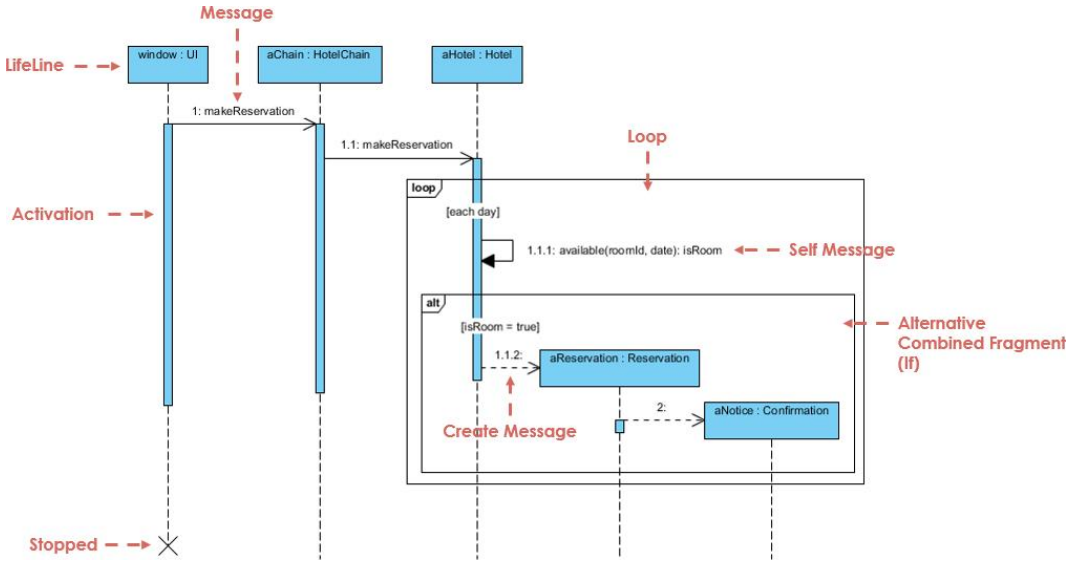

Figure 4 Sequence Diagram Example

Sequence Diagram displays the interaction that takes place in a collaboration realizing a use case or an operation. The diagram helps showing elements when they interact over time and they are organized according to object (horizontally) and time (vertically).

G. ER Diagram

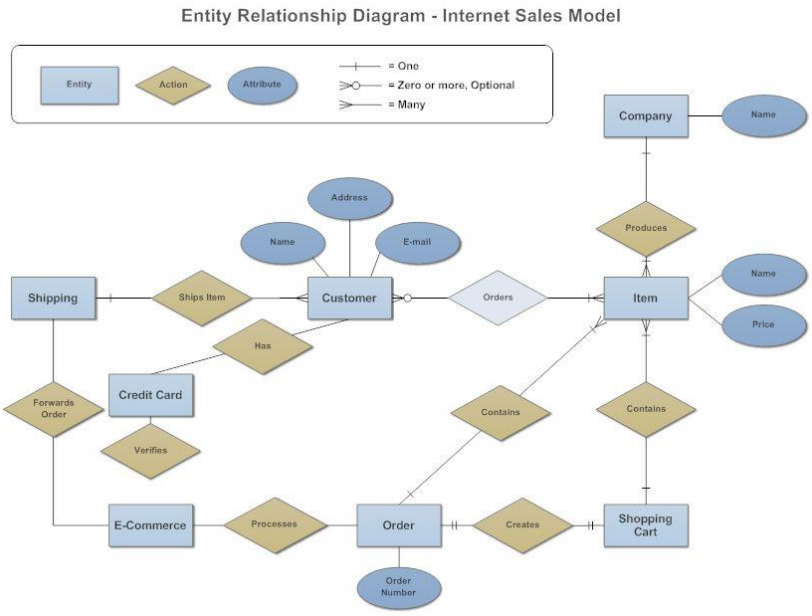

Figure 5 ER Diagram Example

ER Diagram describes the relationships of entity sets stores in a database. The diagram is used to sketch out the design of our database by defining the entities, their attributes, and showing the relationships between, generally, the logical structure of the database.

2.3 Applied Tool

A. Draw.io

Figure 6 Draw.io logo

Draw.io 는 Diagrams.net 에서 지원하는 온라인 모델링 툴이다. 기본적으로 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 있는 등 다이어그램 작성에 필요한 다양한 기능을 제공해주기 때문에 생산성이 매우

높다. 때문에 이 문서에서 사용된 다이어그램은 ER 다이어그램을 제외하면 모두 Draw.io 를 사용하여 작성되었다.

B. ERDPlus

Figure 7 ERDPlus logo

ERDPlus 는 ER Diagram 을 간단한 버튼 클릭으로 생성할 수 있게 해 주는 온라인 툴이다. ER Diagram 을 작성하는 데에서는 draw.io 에 비해 더 쉽고 빠르게 만들 수 있기 때문에 본 문서의 ER Diagram 들은 ERDPlus 를 사용하여 작성되었다.

2.4 Project Scope

본 시스템은 특정 학교 구성원들을 위해 특화된 중고 거래 플랫폼이며, 기존의 다른 중고 거래 플랫폼과는 달리 학과 및 사용자의 시간표에 기반한 추천 시스템을 도입하여 사용자에게 더 효율적으로 추천할 수 있는 것이 특징이다. 본 시스템의 핵심 기능은 이 추천 시스템에 기반한 거래 물품 추천 기능이며, 해당 추천 시스템과 프론트엔드 시스템, 그리고 백엔드 시스템이 서로 상호작용하도록 설계하였다.

프론트엔드 시스템은 시스템과 사용자와의 상호작용을 담당하며, 백엔드 시스템은 프론트엔드 시스템에서 오는 데이터 요청에 응답하고, 요청에 따라 DB 를 수정하거나 추천 시스템을 호출하는 등 요청의 전반적인 처리를 담당한다. 사용자가 물품 검색 화면에 진입하거나 시간표에 따른 추천 기능을 호출하면 추천 시스템은 해당하는 사용자에 대한 정보를 분석하여 사용자에게 필요한 아이템을 추천하여 효율적인 구매를 돕는다.

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

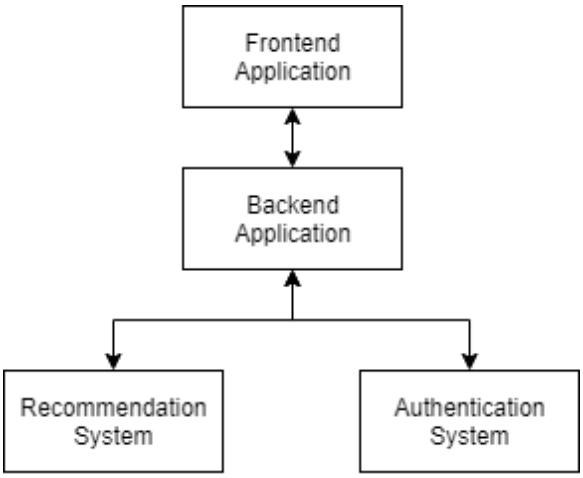

Diagram 1 Overall System Organization

본 시스템은 기본적으로 Model View Controller 모델을 적용해 설계했다. Frontend Application 은 사용자와의 상호작용 및 View 를 출력하여 사용자에게 보여준다. 또한 Frontend Application 은 Backend Application 과 REST 기반의 통신을 사용하여 데이터를 송수신한다. Backend Application 은 Frontend Application 으로부터 들어오는 요청들을 컨트롤러들에 분배하며, 필요한 정보가 있을 경우 데이터베이스에서 해당하는 정보를 가져와 JSON 으로 변환한 뒤 컨트롤러나 Frontend Application 에 전달한다.

Recommendation System 은 Backend Application 으로부터 요청이 있을 경우, 해당하는 유저 정보 및 스케쥴 정보를 분석하여 상품 목록과 유저 간의 유사도를 계산한다. Recommendation System 은 계산된 유사도를 바탕으로 유저에게 적합한 상품 목록을 돌려준다.

Authentication System 은 Backend Application 로부터의 요청을 검증해주며 로그인, 로그아웃, 그리고 회원가입을 담당한다.

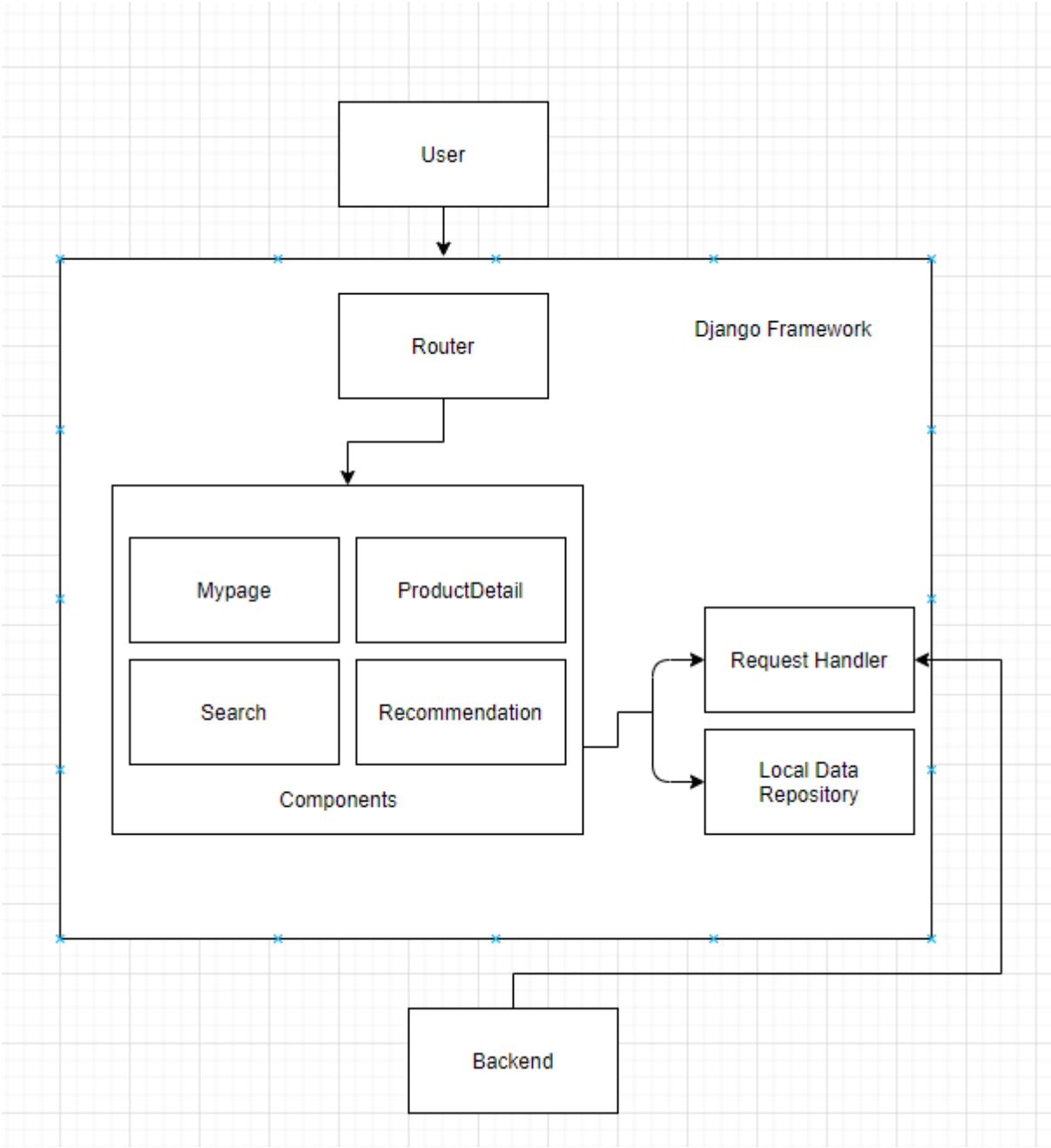

Diagram 2 System Architecture - Frontend

사용자와의 상호작용을 보여주는 시스템으로, Django 프레임 워크를 통해 각 컴포넌트를 관리한다. Mypage, ProductDetail, Search, Recommendation 컴포넌트가 존재하며, 컴포

넌트간의 공유자원의 경우 Local Data Repository 서브 시스템에서 관리한다. 각 컴포넌트가 백엔드와의 통신을 전담하는 Request Handler에게 필요한 데이터를 요청하면, Request Handler는 그 요청을 전환해 백엔드에 전달한다.

3.2 Overall Architecture: Backend

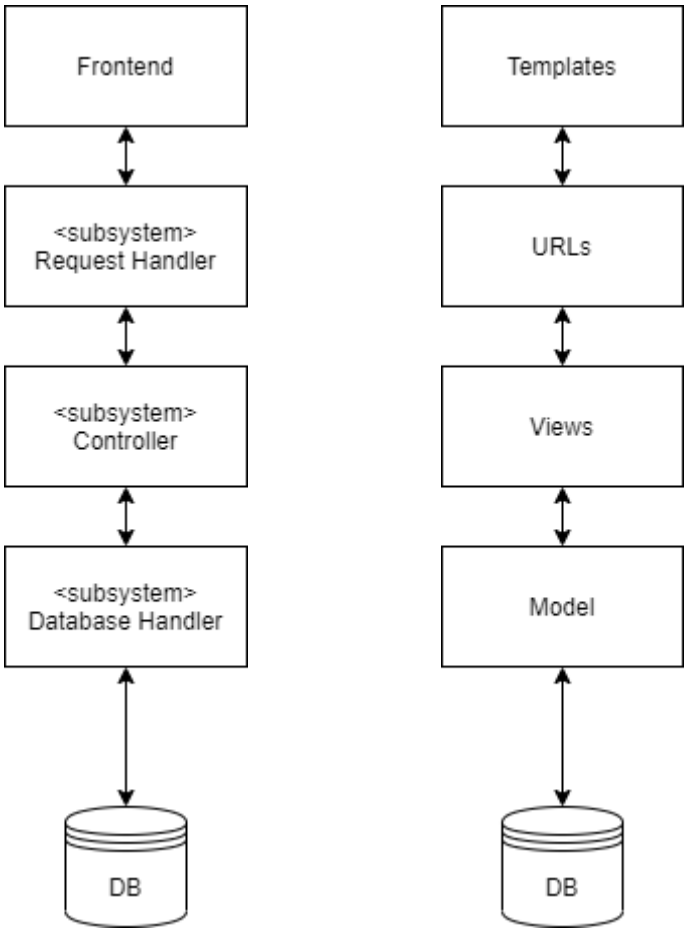

Diagram 3 System Architecture - Backend

Backend system 의 전체 구조는 위와 같으며, Django framework 를 사용한다. Frontend 와 RESTful API 로 통신하며, 하나의 application server 를 운영하여 내부에서 데이터베이스 접근을 통해 데이터 처리를 한다.

3.3 Overall Architecture: Recommendation

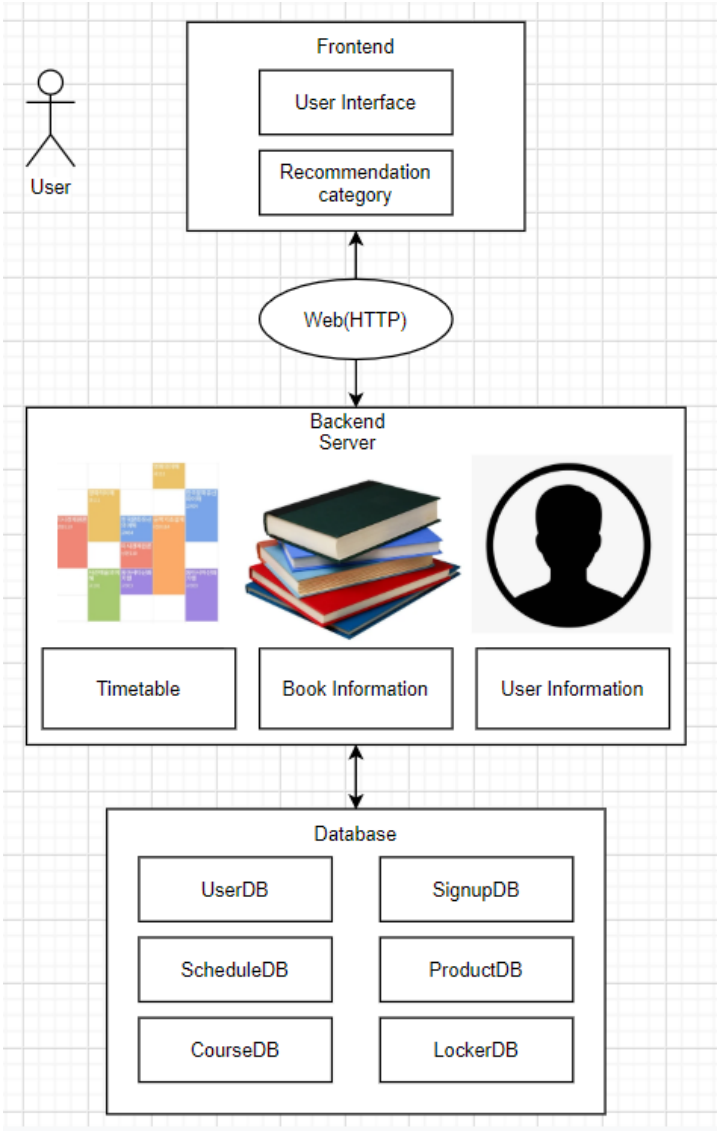

Diagram 4 Overall System Architecture – Recommendation

추천 시스템의 전체적인 구조는 위와 같다. 데이터베이스에서 필요한 정보를 호출해 머신러닝을 기반으로 추천하여 백엔드로 추천정보를 전달하는 구조를 가진다.

4. System Architecture : Frontend

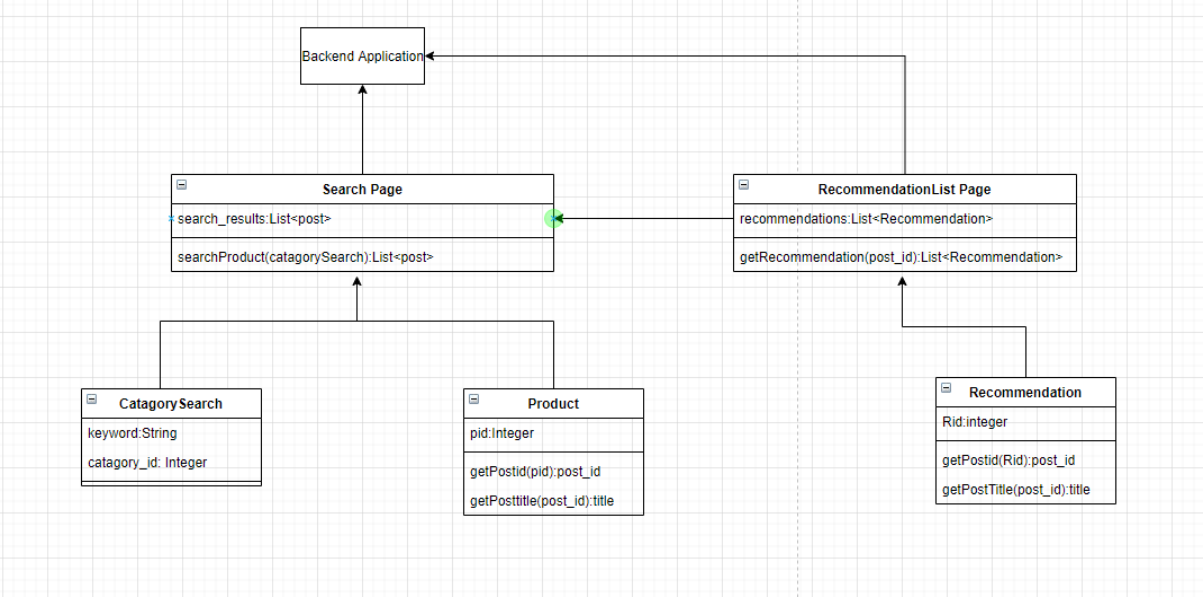

Diagram 5 System Architecture – Frontend - Search

1. Search Page

a. attribute

+search_results: 검색조건을 만족한 상품목록

b. method

+searchProduct: 카테고리조건을 만족하는 상품들을 검색한다.

2. CatagorySearch

a. attribute

+keyword: 검색 키워드

+category: 카테고리

3. Product

a. attribute

+pid: 상품

b. method

+getPostid(pid): 해당 상품의 게시글을 가져오는 메소드

+getPosttitle(post_id): 해당 상품의 제목을 가져오는 메소드

4. Recommendation Page

a. attribute

+recommendation: 추천 상품 목록

b. method

+getRecommendation: 추천 상품목록을 가져오는 메소드

5. Recommendation

a. attribute

+Rid: 추천상품

b. method

+getPost(Rid): 해당 추천 상품 게시글을 가져오는 메소드

+getPostTitle(post_id): 해당 추천 상품 게시글의 제목을 가져오는 메소드

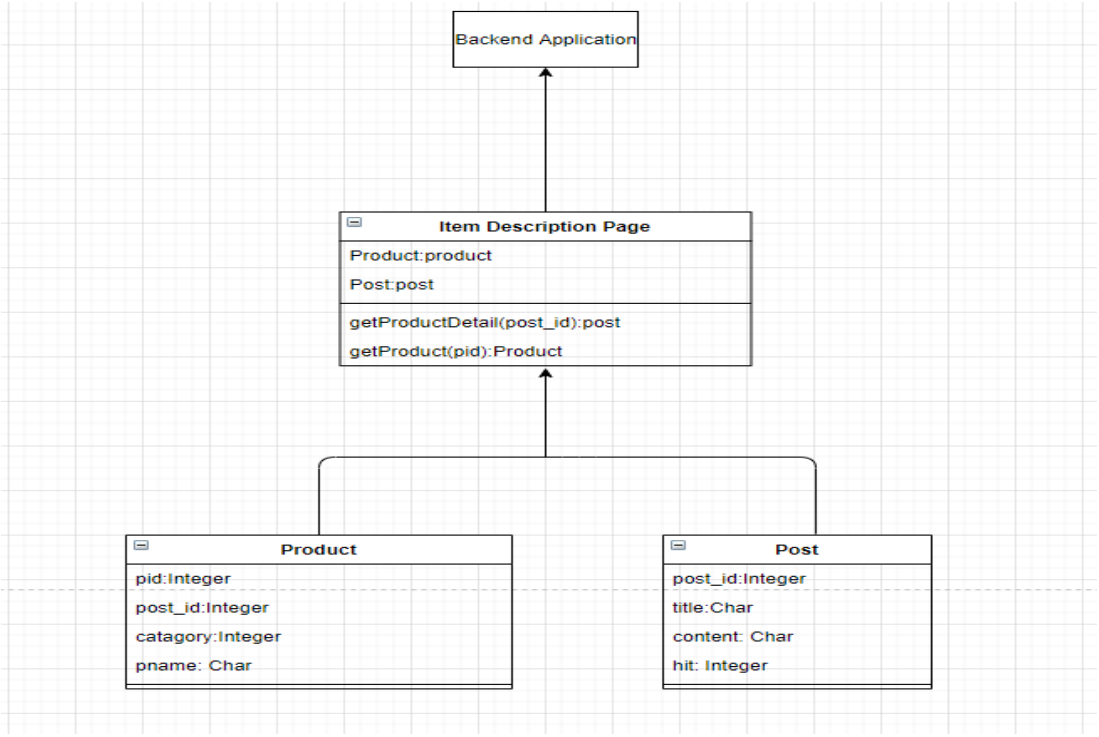

Diagram 6 System Architecture – Frontend - Item

6. Item Description page

a. attribute

+Product: 상품

+Post: 게시글

b. method

+getProductDetail(post_id): 해당 상품 게시글의 내용을 가져오는 메소드

+getProduct(pid): 해당 상품을 가져오는 메소드

7. Product

a. attribute

pid: 상품의 고유번호

post_id: 해당 상품 게시글의 고유번호

category: 해당 상품의 카테고리

pname: 해당 상품의 이름

8. Post

a. attribute

Post_id: 해당 게시글의 고유 번호

Title: 해당 게시글의 제목

Content: 해당 게시글의 내용

Hit: 해당 게시글의 조회수

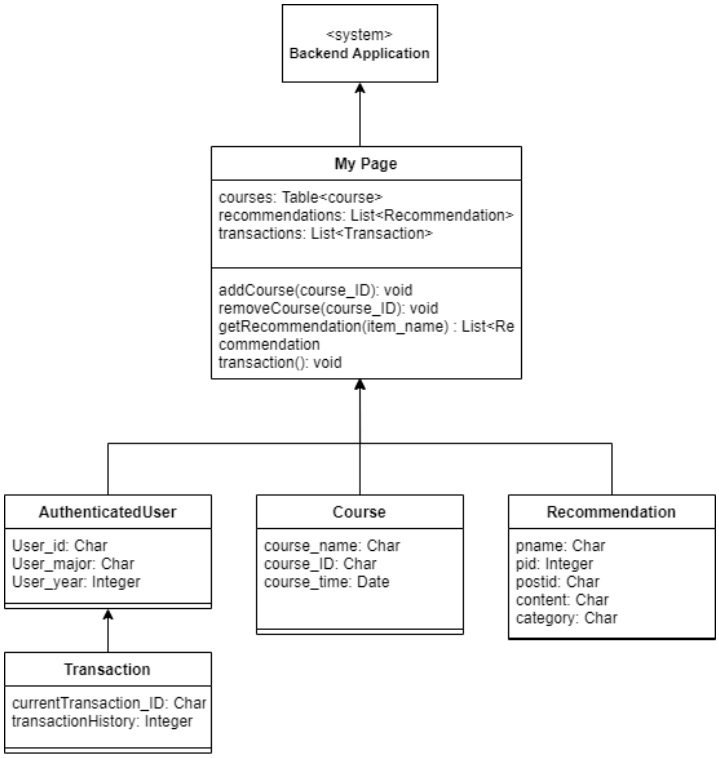

Diagram 7 System Architecture – Frontend - My Page

9. My Page

a. attribute

courses: courses to be displayed

recommendations: recommended products in a list

transaction: transaction history in a list

b. method

addCourse(course_ID): add courses into schedule
removeCourse(course_ID): remove courses from schedule
getRecommendation(pname): Recommended products
transaction(): to display transaction history

10. AuthenticatedUser

User_id: User ID
User_major: User Major
User_year: User's student year

11. Course

course_name: Course name
course_ID: Course ID number
course_time: Course Time

12. Recommendation

pname: product name
pid: product ID
postid: Post ID
content: Product description
category: Category

13. Transaction

currentTransaction_ID: On going transaction ID
transactionHistory: Transaction history in a list

5. System Architecture: Backend

5.1 Objectives

이 챕터에서는 frontend 와 통신할 backend 서버와 그 안의 sub-system 들에 대해 설명한다.

5.2 Overall Architecture

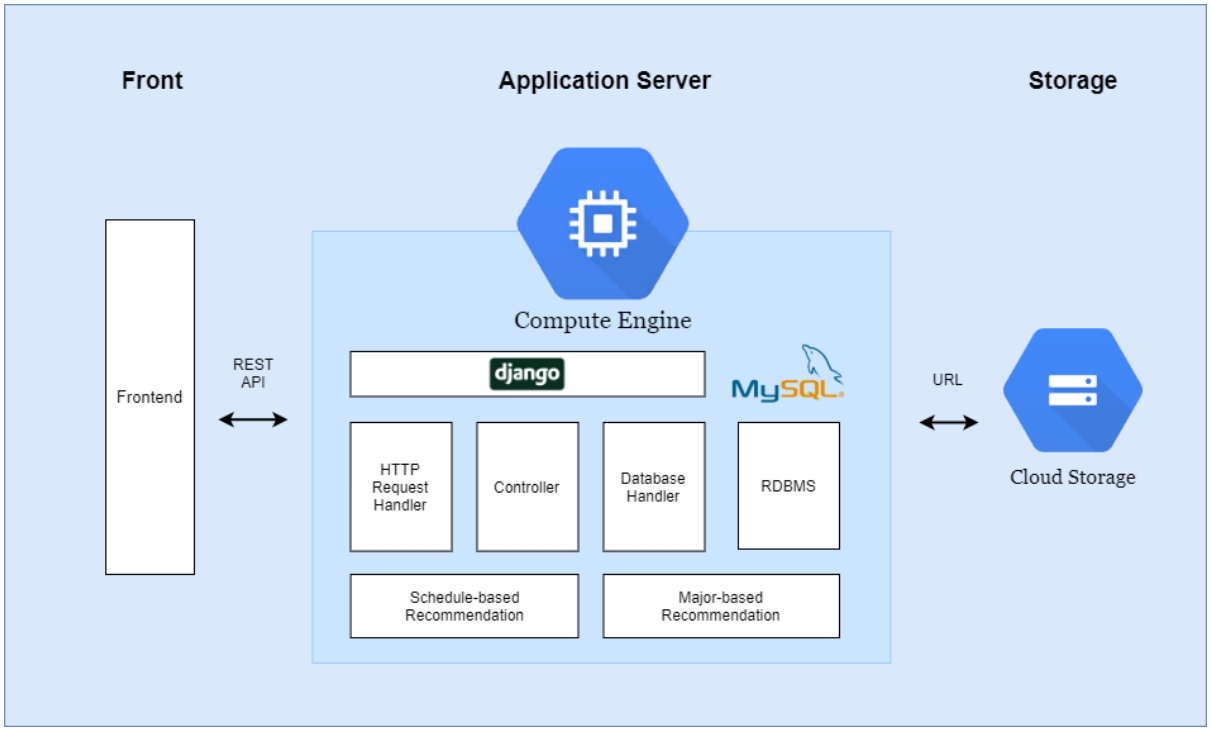

Diagram 8 System Architecture – Backend - Overall

5.3 Subcomponents

A. Application Server

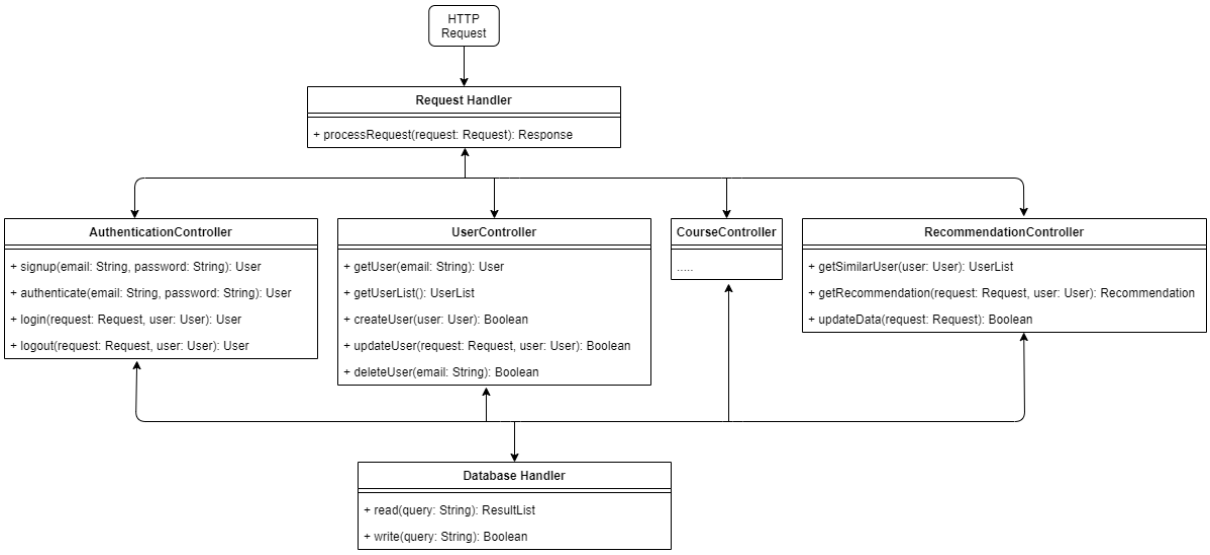

Diagram 9 System Architecture – Backend – Application Server

B. Class Description:

1. Request Handler: 프론트엔드 요청 처리 객체

   A. processRequest(request: Request): 서버로 들어온 요청을 각 컨트롤러로 분배해 처리하고 요청을 반환하는 메서드

2. Entity Controllers(공통)

   A. attributes: black box

   B. getEntity(entity_id: EntityId): 해당 엔티티의 정보를 가져오는 메서드

   C. getEntityList(): 엔티티의 전체 정보를 가져오는 메서드

D. createEntity(entity: Entity): 해당 엔티티 정보를 데이터베이스에 등록하는 메서드

E. updateEntity(entity: Entity): 해당 엔티티 정보를 수정하는 메서드

F. deleteEntity(entity_id: EntityId): 해당 엔티티 정보를 데이터베이스에서 삭제하는 메서드

3. Authentication Controller: 자격 증명 및 회원가입 요청 처리 객체

   A. signup(email: String, password: String): 새로운 유저에 대한 회원가입 프로세스를 진행하고 데이터베이스에 새로운 엔티티를 추가시키는 메서드

   B. authenticate(email: String, password: String): 입력된 유저의 정보를 데이터베이스와 비교하여 인증하는 메서드

   C. login(user: User): 입력된 요청에 따라 로그인 프로세스를 진행하고 유저 정보를 돌려주는 메서드

   D. logout(user: User): 입력된 요청에 따라 로그아웃 프로세스를 진행하는 메서드

4. Recommendation Controller: 추천 서비스 처리 객체

   A. getSimilarUser(user: User): 해당 유저와 비슷한 유저를 반환하는 메서드

   B. getRecommendation(user: User): 해당 유저에게 줄 추천 정보를 반환하는 메서드

   C. updateData(data: Data): 추천 시스템의 데이터를 업데이트 하는 메서드

6. System Architecture: Recommendation

6.1 Objectives

Recommendation System 은 물품을 추천하는 서비스로 사용자는 책가방의 메인 페이지에서 이 시스템을 경험한다. 이 챕터에서는 Recommendation System 의 기능을 클래스 다이어그램과 시퀀스 다이어그램을 이용해서 설명한다.

6.2 Detailed Architecture : First Recommendation

A. Class Diagram

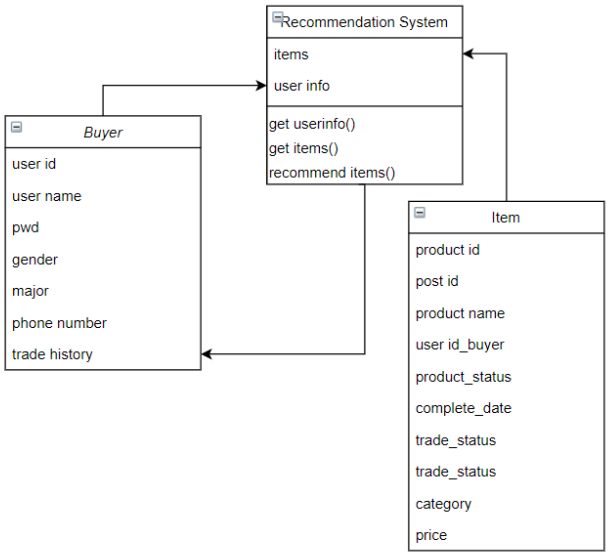

Diagram 10 System Architecture – Recommendation - Class Diagram 1

-Class Description.

[Recommendation System] 사용자의 전공, 성별, 학년, 거래 이력 등의 정보로 비슷한 다른 사용자의 관심 상품을 추천한다.

B. Sequence Diagram

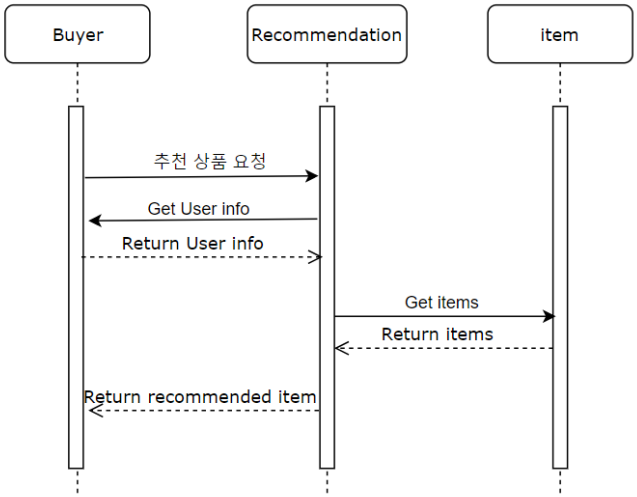

Diagram 11 System Architecture - Recommendation - Sequence Diagram 1

6.3 Detailed Architecture : Second Recommendation

A. Class Diagram

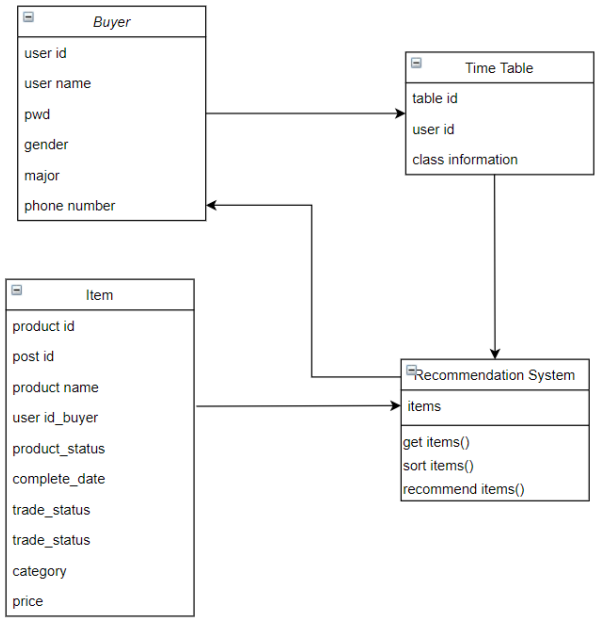

Diagram 12 System Architecture - Recommendation - Class Diagram 2

- Class Description.

[Recommendation System] 사용자의 시간표 정보로 비슷하거나 같은 수업을 듣는 다른 사용자의 관심 상품을 추천한다.

B. Sequence Diagram

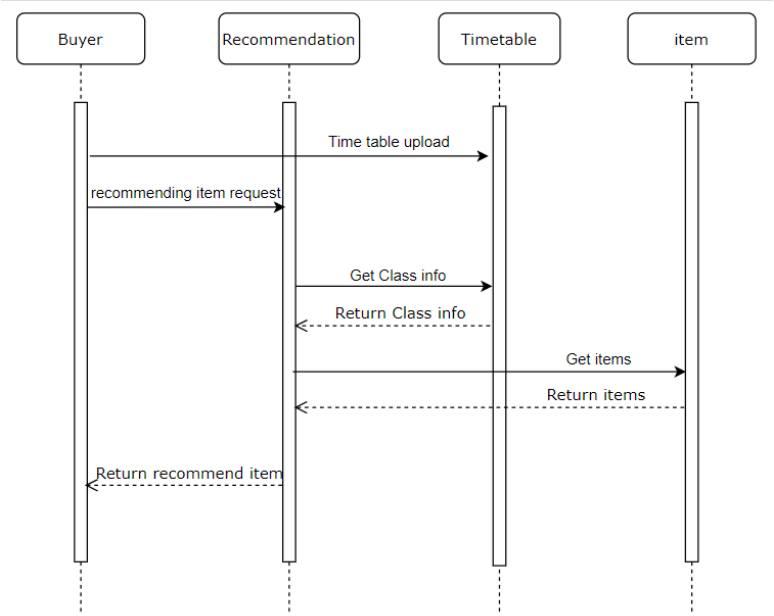

Diagram 13 System Architecture - Recommendation - Sequence Diagram 2

7. Protocol

7.1 Objectives

이 챕터에서는 각 서브시스템이 데이터 통신을 하기 위한 인터페이스를 정의한다. 이 때 인터페이스는 RESTful API 를 사용하며 각 API 의 모델과 약속에 대해 기술한다.

7.2 RESTful API

Figure 8 Django REST framework logo

본 시스템은 프론트엔드와 백엔드 사이의 통신에 웹 인터페이스(HTTP 프로토콜)를 사용하며, REST API 의 형태로 서비스를 제공한다. 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 인터페이스인 POST, GET, PUT, DELETE 4 개의 메서드가 사용된다.
3. 표현(Representation) - 행위에 대한 내용을 표현한다. 주로 JSON, XML 등이 사용되며, 본 시스템에서는 JSON 을 사용할 것이다.

REST API 를 사용하여 개발을 할 경우, 프론트엔드와 백엔드를 분리시켜 개발이 가능하여 개발 효율성이 높아지고, 또한 프로토콜이 이해하기 쉬워지는 장점이 있다. 이를 통해 본 시스템에서 개발하는 프론트엔드 클라이언트뿐 아니라 다른 시스템에서 서버의 자원을 쉽게 이용할 수 있기 때문에 확장성이 높다.

Django REST framework (DRF)는 이러한 REST API 를 쉽게 구축할 수 있도록 해주는 Django 의 framework 이다. 본 시스템에서는 DRF 를 사용하여 REST API 의 기능을 구현할 것이다.

7.3 JSON

Figure 9 JSON logo

JSON(JavaScript Object Notation)은 "속성-값 쌍" 또는 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위한 데이터 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. 또한 JSON 은 언어 독립형 데이터 포맷으로, 구문 분석 및 JSON 데이터 생성을 위한 코드는 C, C++, 파이썬 등 수많은 언어에서 쉽게 이용할 수 있다.

6.4 Details

A. Authentication

Sign up

Method	POST

URI	/authentication/signup/
Request Body	email: 사용자 E-mail pwd: 사용자 비밀번호 uname: 사용자 이름 gender: 0 or 1 (남or여) major: 전공 phone: 핸드폰번호
Response Body	result: (true/false) message: (success/실패 이유)

Login

Method	POST
URI	/authentication/login/
Request Body	email: 사용자 E-mail pwd: 사용자 비밀번호
Response Body	result: (true/false) message: (success/실패 이유)

Logout

Method	GET
URI	/authentication/logout/
Request Body	-
Response Body	result: (true/false) message: (success/실패 이유)

B. User

Read

Method	GET
URI	/api/users/{id}
Request Body	-
Response Body	User 정보

Update

Method	PUT
URI	/api/users/{id}
Request Body	(변경사항만) email: 사용자 E-mail uname: 사용자 이름 gender: 0 or 1 (남or여) major: 전공 phone: 핸드폰 번호
Response Body	User 정보

C. Product

Read(All)

Method	GET
URI	/api/products/

Request Body	-
Response Body	Product 리스트

Read(Detail)

Method	GET
URI	/api/products/{id}
Request Body	-
Response Body	Product 정보

Create

Method	POST
URI	/api/products/
Request Body	post_id: 게시글 id uid_buyer: 구매자 id(0 or else) complete_date: 구매 완료 시점(Null or Datetime) pname: 상품 이름 category: 상품 카테고리 price: 상품 가격 status: (0 or 1 or 2)
Response Body	Product 정보

Update

Method	PUT

URI	/api/products/{id}
Request Body	(바꿀정보) post_id: 게시글 id uid_buyer: 구매자 id(0 or else) complete_date: 구매 완료 시점(Null or Datetime) pname: 상품 이름 category: 상품 카테고리 price: 상품 가격 status: (0 or 1 or 2)
Response Body	Product 정보

Delete

Method	DELETE
URI	/api/products/{id}
Request Body	-
Response Body	Code 204

D. Post

Read(All)

Method	GET
URI	/api/posts/
Request Body	-
Response Body	posts 리스트

Read(Detail)

Method	GET
URI	/api/posts/{id}
Request Body	-
Response Body	Post 정보

Create

Method	POST
URI	/api/posts/
Request Body	uid: 판매자 id title: 게시글 제목 content: 게시글 내용
Response Body	code: (200 OK / 400 Bad Request) content: (post_id / 실패 이유 message)

Update

Method	PUT
URI	/api/posts/{id}
Request Body	(바꿀정보) title: 게시글 제목 content: 게시글 내용
Response Body	code: (200 OK / 400 Bad Request) content: (post_id / 실패 이유 message)

Delete

Method	DELETE
URI	/api/posts/{id}
Request Body	-
Response Body	code: (200 OK / 403 Forbidden)

E. Schedule

Read(All)

Method	GET
URI	/api/schedules/
Request Body	-
Response Body	schedule 리스트

Read(Detail)

Method	GET
URI	/api/schedules/{id}
Request Body	-
Response Body	schedule 정보

Create

Method	POST
URI	/api/schedules/
Request Body	cid: 강의 id
Response Body	code: (200 OK / 400 Bad Request) content: (schedule_id / 실패 이유 message)

Update

Method	PUT
URI	/api/schedules/{id}
Request Body	(바꿀 정보) cid: Course ID
Response Body	code: (200 OK / 403 Forbidden)

Delete

Method	DELETE
URI	/api/schedules/{id}
Request Body	-
Response Body	code: (200 OK / 403 Forbidden)

F. Course

Read(All)

Method	GET
URI	/api/courses/
Request Body	-
Response Body	course 리스트

Read(Detail)

Method	GET
URI	/api/courses/{id}
Request Body	-
Response Body	course 정보

Create

Method	POST
URI	/api/courses/
Request Body	c_number: 학수번호 cname: 수업 이름 professor: 교수 이름 time: 수업 시간
Response Body	course 정보

Update

Method	PUT
URI	/api/courses/{id}
Request Body	(바꿀 정보) c_number: 학수번호 cname: 수업 이름 professor: 교수 이름 time: 수업 시간
Response Body	course 정보

G. Locker

Read(All)

Method	GET
URI	/api/lockers/
Request Body	-
Response Body	locker 리스트

Read(Detail)

Method	GET
URI	/api/lockers/{id}
Request Body	-

Response Body	locker 정보

Update

Method	PUT
URI	/api/lockers/{id}
Request Body	(바꿀 정보) l_number: 사물함 번호 status: 현재 상태 location: 사물함 위치
Response Body	locker 정보

8. Database Design

8.1 Objectives

요구사항 명세서에서 작성한 데이터베이스 요구사항을 기반으로 하여 세부적인 데이터베이스 설계를 기술한다. ER Diagram을 통해 전체적인 Entity의 relationship에 관하여 표현하고 Relational Schema, SQL DDL 명세를 만든다.

8.2 ER Diagram

본 시스템에서는 user, locker, schedule, course, product, post, signup 총 7개의 DB entity를 이용한다. ER Diagram 은 각각의 entity와 그들의 관계를 이어주는 점선으로 나타내어지며, 특정 entity가 다른 entity 와 1 : M 관계를 가질 수 있을 때에는 해당 entity 쪽에는 삼지창 모양 선을 세 개 그어 나타낸다. 각각의 entity가 가지는 속성들은 타원으로 표현이 된다. 이 중, Key attribute 즉, 다른 객체들과 중복되지 않는 고유한 값을 가진

Attribute는 주로 객체식별에 사용되는데 이는 타원 안에 밑줄을 그어 표시한다. 또한, 복합키를 기본키로 사용하는 객체들은 기본키에 사용하는 모든 컬럼에 대하여 밑줄을 그어 나타낸다. 각 Entity간의 관계에 대하여는 Entity 사이의 마름모를 사용하여 relationship을 표기한다.

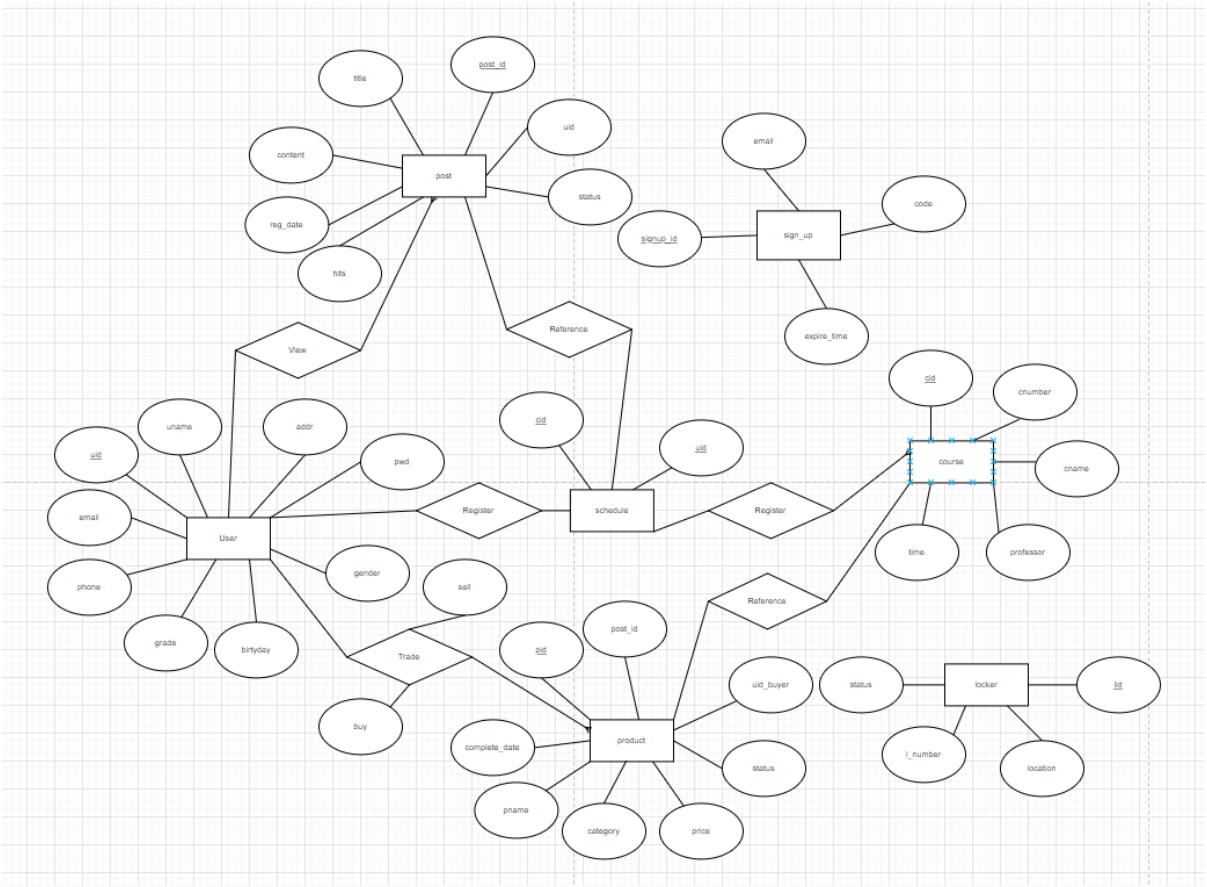

Diagram 14 Overall ER diagram

1. Entities

A. User

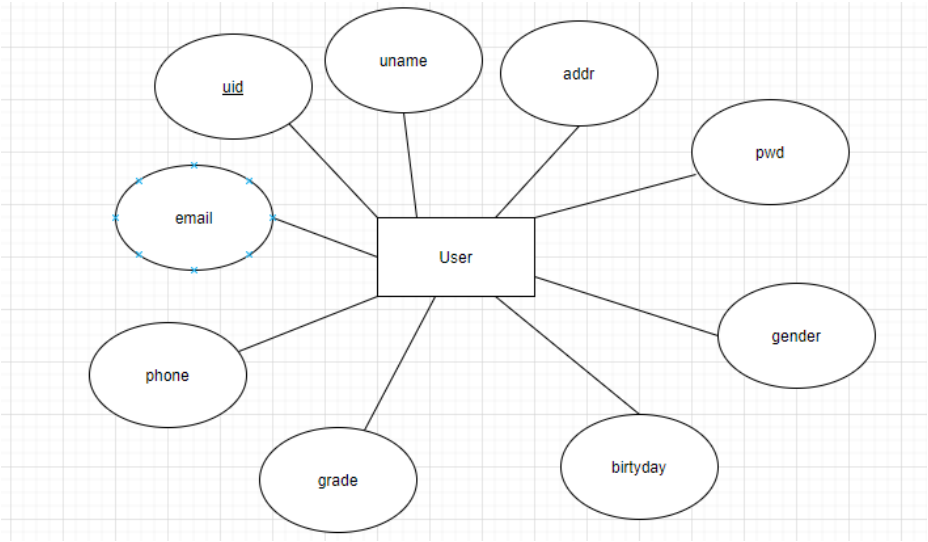

Diagram 15 Overall ER diagram - User Entity

User Entity는 사용자의 정보를 표현한다. uid 속성이 primary key이며, 이름, 주소, 패스워드, 성, 생년월일, 학년, 전화번호, 이메일의 정보를 가지고 있다.

B. Course

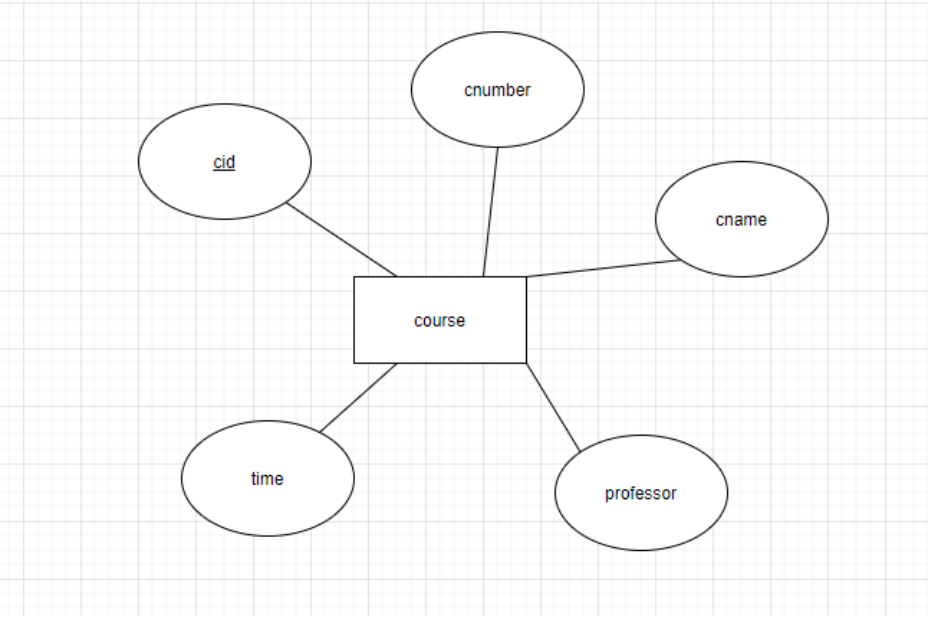

Diagram 16 Overall ER diagram - Course Entity

Course Entity는 schedule에서 저장할 강의들의 정보를 표현한다. cid가 primary key이며, 학수번호, 강의이름, 교수님, 시간대의 정보를 가지고 있다.

C. Locker

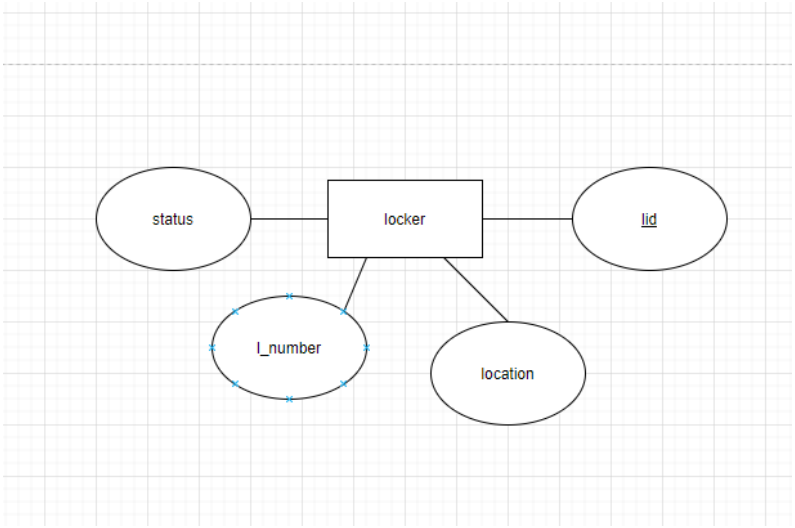

Diagram 17 Overall ER diagram - Locker Entity

Locker Entity는 사물함거래 시, 이용할 사물함에 대한 정보를 가지고 있다. lid 가 primary key이다. Status는 사물함이 이용 중인지를 상태를 표현한 속성이고 이외에도 사물함번호, 사물함 위치의 정보를 가지고 있다.

D. Post

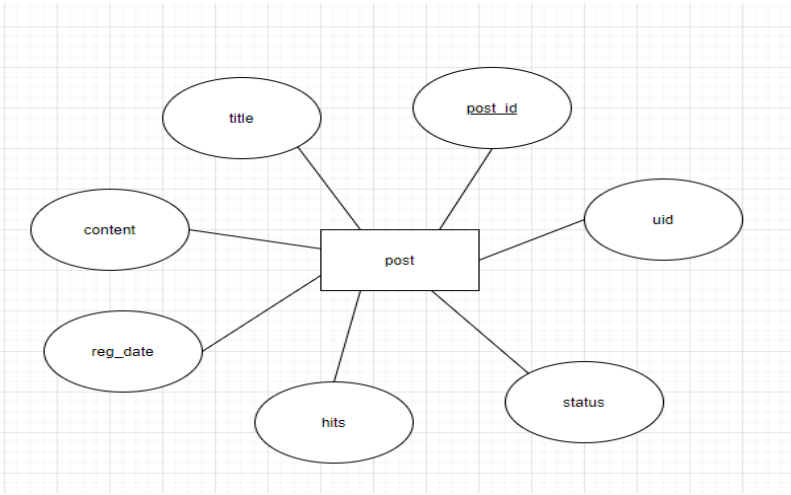

Diagram 18 Overall ER diagram - Post Entity

Post Entity는 판매 게시글들의 정보를 가지고 있다. Post_id가 primary key이다. uid 는 글쓴이가 누구인지를 저장하며 hit는 조회수를, status는 현재 게시글이 거래중인지 아닌지의 상태 정보를 저장한다. 이외에도 게시일, 내용, 제목의 정보를 가지고 있다.

E. Product

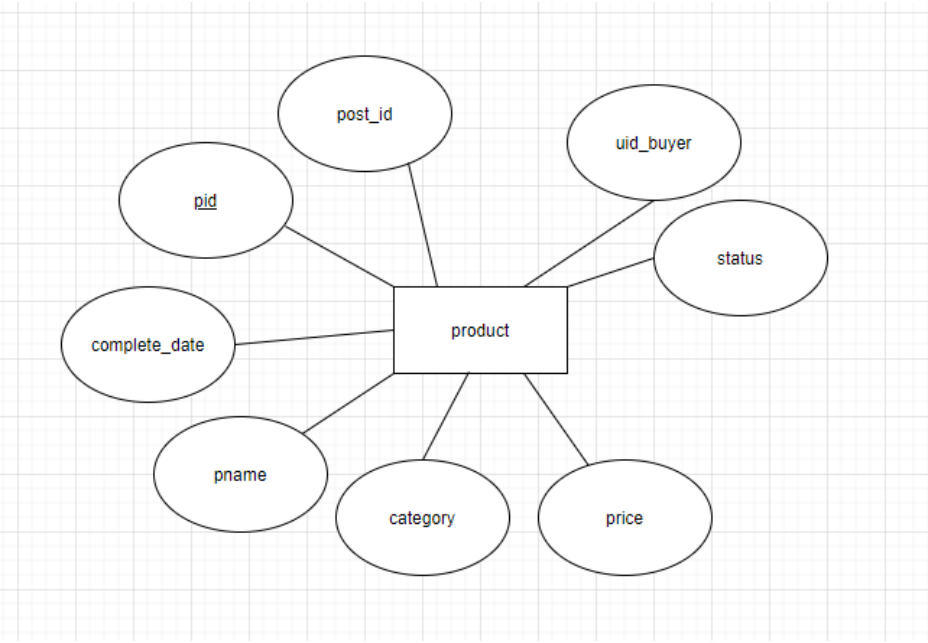

Diagram 19 Overall ER diagram - Product Entity

Product Entity는 상품에 대한 정보를 저장한다. pid 가 primary key이다. post_id는 해당 상품의 게시글의 primary key값이며 uid_buyer는 해당 상품을 사간사람의 정보를 저장하는데, 아직 팔리지 않았다면 0이 된다. 이외에도 거래완료시간, 상품 이름, 상품 카테고리, 가격, 상태의 정보를 저장한다.

F. Schedule

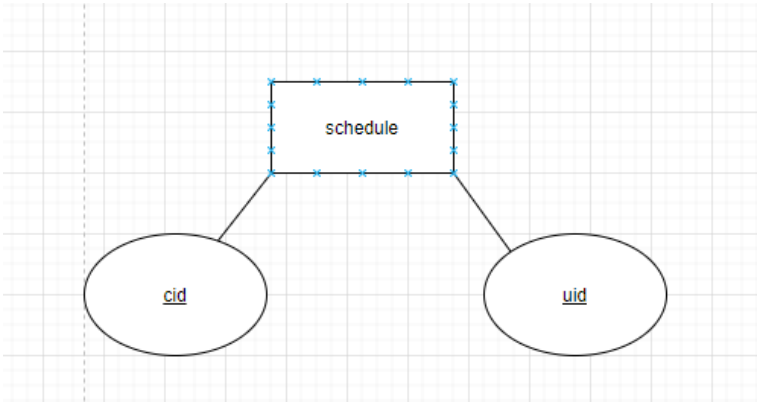

Diagram 20 Overall ER diagram - Schedule

Schedule Entity는 사용자의 시간표 정보를 저장한다. cid와 uid를 Foreign key로 받아 두개로 primary key를 구성한다.

G. Sign_up

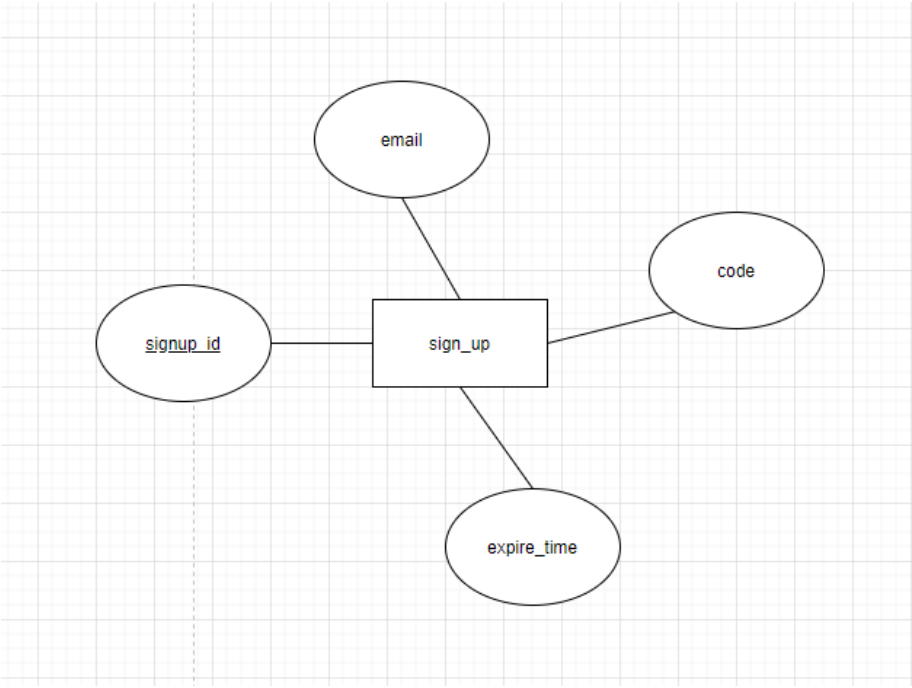

Diagram 21 Overall ER diagram - Sign Up Entity

Sign_up Entity는 사용자가 가입 시, 이메일 확인 정보를 저장한다. Sign_up_id가 primary key이며 email은 사용자가 코드를 받을 메일 주소를, expire_time은 제한 시간의 정보를 저장한다.

8.3 Relational Schema

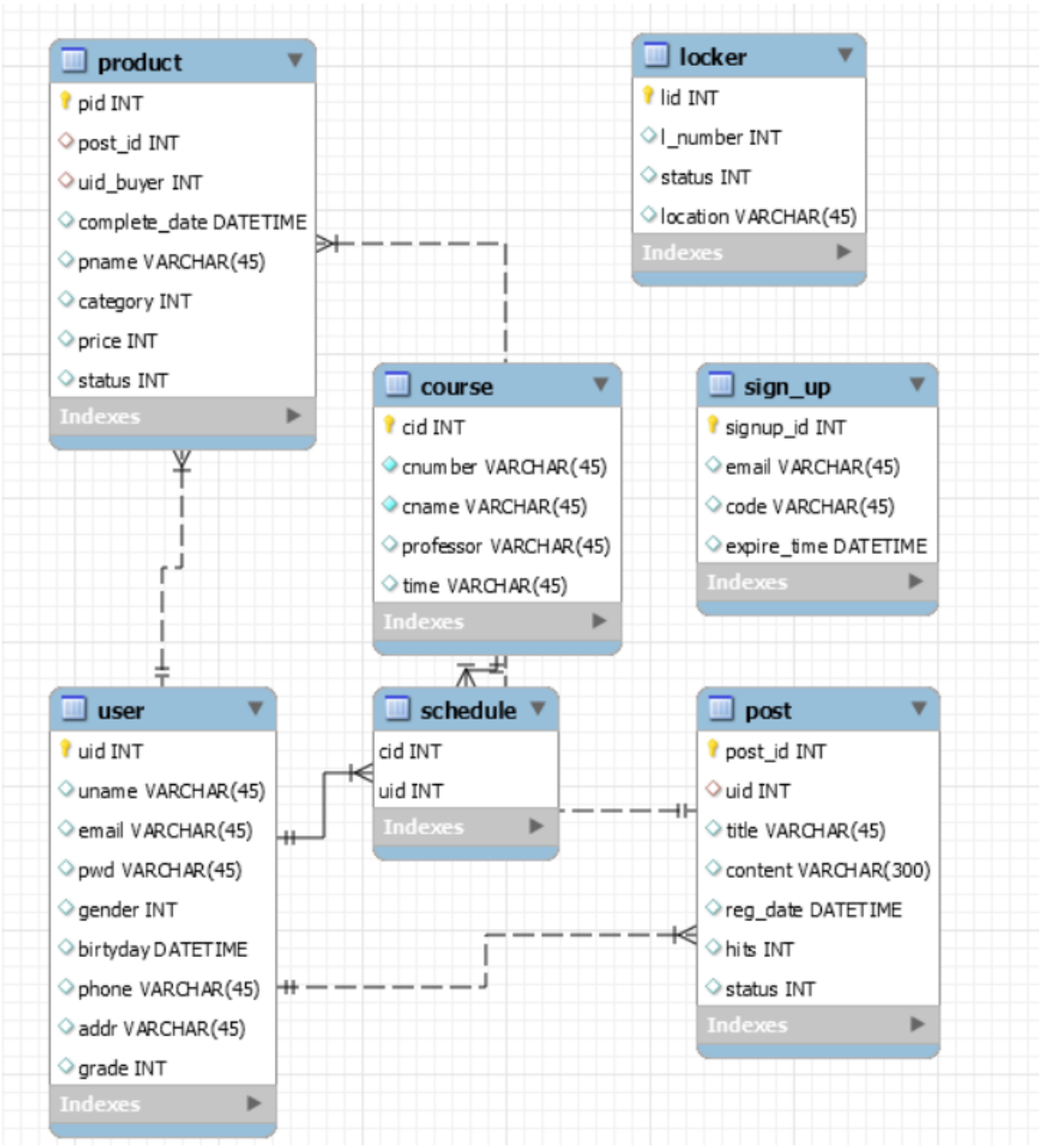

Diagram 22 Relational Schema

8.4 SQL DDL

user table

```
CREATE TABLE `backpack(swe)`.`user` (
  `uid` INT NOT NULL AUTO_INCREMENT,
  `uname` VARCHAR(45) NULL,
  `email` VARCHAR(45) NULL,
  `pwd` VARCHAR(45) NULL,
  `gender` INT NULL,
  `birtyday` DATETIME NULL,
  `phone` VARCHAR(45) NULL,
  `addr` VARCHAR(45) NULL,
  `grade` INT NULL,
  PRIMARY KEY (`uid`));
```

Table 1 User Table

post table

```
CREATE TABLE `backpack(swe)`.`post` (
  `post_id` INT NOT NULL AUTO_INCREMENT,
  `uid` INT NULL,
  `title` VARCHAR(45) NULL,
  `content` VARCHAR(300) NULL,
  `reg_date` DATETIME NULL,
  `hits` INT NULL,
  `status` INT NULL,
```

Table 2 Post Table

product table

```
CREATE TABLE `backpack(swe)`.`product` (
  `pid` INT NOT NULL AUTO_INCREMENT,
  `post_id` INT NULL,
  `uid_buyer` INT NULL,
  `complete_date` DATETIME NULL,
  `pname` VARCHAR(45) NULL,
  `category` INT NULL,
  `price` INT NULL,
  `status` INT NULL,
   PRIMARY KEY (`pid`),
   FOREIGN KEY(`uid_buyer`) REFERENCES user(`uid`),
```

Table 3 Product Table

course table

```
CREATE TABLE `backpack(swe)`.`course` (
  `cid` INT NOT NULL AUTO_INCREMENT,
  `cnumber` VARCHAR(45) NOT NULL,
  `cname` VARCHAR(45) NOT NULL,
  `professor` VARCHAR(45) NULL,
  `time` VARCHAR(45) NULL,
  PRIMARY KEY (`cid`));
```

Table 4 Course Table

schedule table

```
CREATE TABLE `backpack(swe)`.`schedule` (
  `cid` INT NOT NULL,
  `uid` INT NOT NULL,
  PRIMARY KEY (`cid`, `uid`),
  FOREIGN KEY(`cid`) REFERENCES `course`(`cid`),
  FOREIGN KEY(`uid`) REFERENCES `user`(`uid`));
  );
```

Table 5 Schedule Table

signup table

```
CREATE TABLE `backpack(swe)`.`sign_up` (
  `signup_id` INT NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(45) NULL,
  `code` VARCHAR(45) NULL,
  `expire_time` DATETIME NULL,
  PRIMARY KEY (`signup_id`));
```

Table 6 Sign Up Table

locker table

```
CREATE TABLE `backpack(swe)`.`locker` (
  `lid` INT NOT NULL AUTO_INCREMENT,
  `l_number` INT NULL,
  `status` INT NULL,
  `location` VARCHAR(45) NULL,
  PRIMARY KEY (`lid`));
```

Table 7 Locker Table

9. Testing Plan

8.1. Objectives

이 챕터에서는 본 시스템을 개발 완료하고 운영하기에 앞서 본 시스템이 기능적, 비기능적 요구사항들을 충족하였는지, 그리고 시스템을 운영하며 추후에 발생할 수 있는 문제점들을 규명하기 위한 테스트의 내용과 방식에 대해 기술한다.

8.2 Testing Policy

A. Development Testing

본 시스템을 개발하는 과정이나 실제 배포 시 발생할 다양한 문제들을 찾아내고 해결하기 위한 테스트이다. 이를 위한 테스트 방법으로 static code analyzing, peer code review, unit testing, 마지막으로 data flow analyzing 을 사용할

예정이며 이는 비 기능적 요구사항들을 중점적으로 충족하는지 테스트하도록 한다.

1) Usability

본 시스템의 이용층은 대학생과 관련 관계자들이며, 주 소비자가 될 대학생들의 수업 시간표를 인터페이스로 활용한다. 사용자의 기존 커머셜 서비스 이용 경험에 대한 익숙함을 유지시키면서 시간표를 활용한 대학생 특화 인터페이스를 제공하도록 한다. 시간표를 이용해 사용자에게 검색과 같은 절차 없이 시간표만으로 사용자의 수업에 요구되는 중고 서적 물품을 알 수 있도록 구매절차를 간소화한다. 또한 주 소비자가 되는 대학생은 인터넷 활용성이 높고 거래 목표 달성이 높다는 점을 고려하여 필요한 기능만 넣어 깔끔한 인터페이스를 제공하도록 한다.

2) Security

본 시스템은 같은 대학교 학생들로 이루어진 커뮤니티이다. 따라서 이와 관계없는 사용자들이 해당 시스템에 가입하지 못 하도록 학교 인증을 확실히 해야 한다. 이는 학교 이메일 인증을 통해 해결될 수 있으며, unit testing 을 통해 검증한다.

또한 사용자의 개인정보는 데이터베이스에 저장되며 이 때 비밀번호와 같은 민감한 정보는 암호화되어 저장된다. 본 시스템은 sha256 해싱 암호화를 사용하며 이는 data flow analyzing 을 통해 제 3 자가 해당 정보를 알 수 있는지 테스트한다.

마지막으로 본인이 아닌 게시물에 대한 접근 권한은 통제되어야 하므로 인가되지 않은 사용자에 대한 접근 방지를 테스트한다.

3) Reliability

구매자가 잘못된 상품 현황으로 책가방 서비스에 대한 구매 실패 경험을 하지 않도록 이미 거래가 이루어지거나 거래중인 상품에 대해 빠르고 정확하게 정보를 제공하도록 한다. 또한 책가방은 사물함을 이용하여 직거래 서비스를 제공하고자 한다.사용자의 사물함 직거래 니즈가 발생했을 때, 그에 따른 정보를 정확하게

전달하여야 한다. 전달되어야 하는 정보로는 이용 가능한 사물함의 유무와 해당 위치 등이 있다.

B. Release Testing

본 시스템을 운영하던 중에 시스템 또는 서브시스템의 새로운 버전이 출시되어 적용시켜야 할 때 거쳐야 할 테스트이다. 이는 새로운 버전의 시스템이 기존의 시스템과 같이 결함이 없으며 각 요구사항들을 잘 충족하는지 테스트한다.

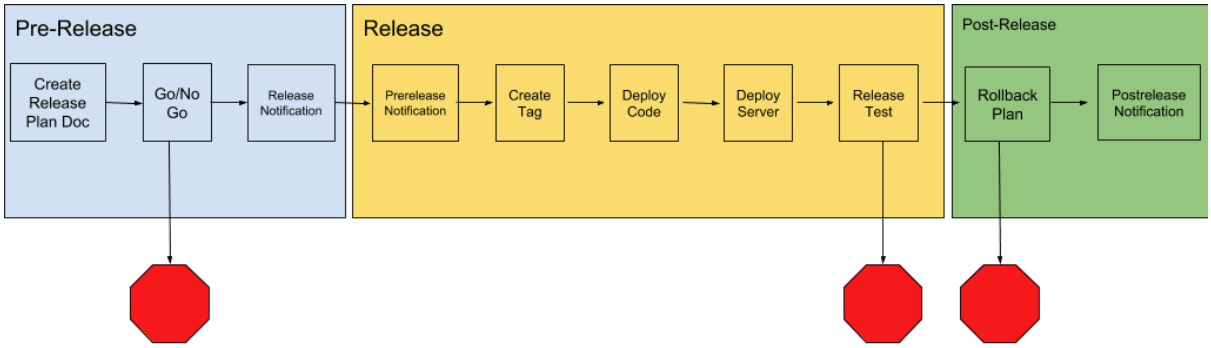

Figure 10 Release test

위 사진처럼 release test 는 3 단계에 걸쳐 이루어져야 한다. 이 때 주의할 점은 사용자들이 본 시스템을 업데이트 하지 않고 사용할 경우가 발생할 수 있는데, 강제로 업데이트 시킬지 아니면 버전별로 다르게 관리할 것인지에 대해 결정해야 한다.

C. User Testing

실제 유저 테스트를 통하여 앞선 비 기능적 요구사항인 usability 와 사용자의 기능적 요구사항을 충족하였는지 확인한다. 단, 본 시스템의 개발에 있어서 실제 유저의 테스트를 적용시키기 어려우므로 실제 유저를 대상으로 시스템을 이용하는 상황을 시나리오로 작성하고, 본 시스템 개발자들을 사용자로 가정하여 테스트를 진행하도록 한다.

D. Testing Case

앞서 언급하였던 것처럼 실제 유저를 대상으로 시스템을 이용하는 상황을 시나리오 작성하고, 그에 따른 flow 를 통해 test code 를 작성하고 테스트한다. 이에 대해서는 다음 'Code and Test Result'에서 자세히 다루도록 한다.

10. Development Plan

10.1 Front-End

Figure 11 Vue.js Logo

Vue.js is a front-end framework that enable single-file components helps improving code reuse and increasing productivity and it gives lots of flexibility and adaptability. Vue is useful for keeping front-end code clean. It's clear dependency-tracking observation system with asynchronized queueing ensures changes trigger individualistically during render.

Figure 12 JavaScript Logo

자바스크립트는 객체 기반의 스크립트 프로그래밍 언어이다. 이 언어는 웹 브라우저 내에서 주로 사용하며, 다른 응용 프로그램의 내장 객체에도 접근할 수 있는 기능을 가지고 있다. 또한 Node.js와 같은 런타임 환경과 같이 서버 사이드 네트워크 프로그래밍에도 사용되고 있다.

Figure 13 SS Logo

종속형 시트 또는 캐스케이딩 스타일 시트(Cascading Style Sheets, CSS)는 마크업 언어가 실제 표시되는 방법을 기술하는 언어로, HTML 과 XHTM 에 주로 쓰이며, XML 에서도 사용할 수 있다. W3C 의 표준이며, 레이아웃과 스타일을 정의할 때의 자유도가 높다.

마크업 언어가 웹사이트의 몸체를 담당한다면 CSS 는 옷과 액세서리 같은 꾸미는 역할을 담당한다고 할 수 있다. 즉, HTML 구조는 그대로 두고 CSS 파일만 변경해도 전혀 다른 웹사이트처럼 꾸밀 수 있다.

Figure 14 HTML Logo

HTML또는 하이퍼텍스트 마크업 언어(HyperText Markup Language, 문화어: 초본문표식달기언어, 하이퍼본문표식달기언어)는 웹 페이지를 위한 지배적인 마크업 언어다. HTML은 제목, 단락, 목록 등과 같은 본문을 위한 구조적 의미를 나타내는 것뿐만 아니라 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공한다. 그리고 이미지와 객체를 내장하고 대화형 양식을 생성하는 데 사용될 수 있다. HTML은 웹 페이지 콘텐츠 안의 꺾쇠 괄호에 둘러싸인 "태그"로 되어있는 HTML 요소 형태로 작성한다. HTML은 웹 브라우저와 같은 HTML 처리 장치의 행동에 영향을 주는 자바스크립트와 본문과 그 밖의 항목의 외관과 배치를 정의하는 CSS 같은 스크립트를 포함하거나 불러올 수 있다.

10.2 Back-End

Python

Figure 15 Python Logo

Python 은 동적 타이핑(dynamic typing) 범용 프로그래밍 언어로, 웹 어플리케이션뿐만 아니라 기계학습, 통계, 그리고 비쥬얼 컴퓨팅 등 수많은 분야에서 쓰이고 있다. 본 시스템에서는 Backend Application 을 작성하는데 Python 을 사용했는데, 이는 Python 이 생산성이 굉장히 높고 개발속도가 빠른 언어라 단기 프로젝트에 적합하며, Python 으로 작성된 다양한 오픈소스 웹 프레임워크가 존재하기 때문이다.

Django

Figure 16 Django Logo

Django 는 Python 기반 오픈소스 웹 프레임워크로 모델-뷰-컨트롤러 (MVC) 패턴을 따르고 있다. Djagno 의 장점은 컴포넌트들의 재사용성이 높고 빠른 개발이 가능하다는 것인데, 이는 데이터베이스 관리, 보안 문제 등의 다양한 문제들을 쉽게 해결할 수 있도록 지원해주는 다양한 기능들이 존재하기 때문이다. Django 는 현재 인스타그램, Disqus, 모질라 등 다양한 곳에서 웹 프레임워크로 사용되고 있으며, 본 시스템에서도 Backend Application 를 작성할 때 Django 를 사용하였다.

MySQL

Figure 17 MySQL Logo

MySQL 은 세계에서 가장 많이 쓰이고 있는 오픈 소스 관계형 데이터베이스 관리 시스템(RDBMS)이다. 다중 스레드, 다중 사용자 형식의 구조질의어 형식 데이터베이스 관리 시스템이며, 오라클이 관리 및 지원을 해주고 있다. 본 시스템에서는 Backend Application 의 데이터베이스를 관리하는 RDBMS 로써 MySQL 을 사용할 것이다. 이는 MySQL 이 오랜 기간 많은 이들에게 사용되어 온 만큼 다양한 소스가 존재하며, 팀원들이 MySQL 에 더 익숙하기 때문이기도 하다.

10.3 Recommendation System

Figure 18 Pandas Logo

판다스는 Python에서 사용하는 데이터 분석(Data Analysis) 라이브러리로, 행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있게 되며 보다 안정적으로 대용량의 데이터들을 처리하는데 매우 편리한 도구이다.

Figure 19 Selenium Logo

Selenium은 주로 웹앱을 테스트하는 웹 프레임워크이다. 또한 webdriver의 API를 통해 브라우저를 제어하기 때문에 자바스크립트에 의해 동적으로 생성되는 사이트의 데이터를 크롤링할 때 매우 유용하게 사용되는 스크래핑 도구이다. 이와 같이 사용하는 BeautifulSoup 모듈은 HTML과 XML을 파싱하는 데에 사용되는 파이썬 라이브러리이다.

** 스크랩핑(scraping) : 데이터를 수집하는 행위

** 크롤링(Crawling) : 조직적 자동화된 방법으로 월드와이드웹을 탐색 하는 것

Figure 20 GENSIM Logo

Gensim은 문서 사이의 유사도를 계산과 텍스트 분석을 돕는 자연어 처리 라이브러리로써

Word2Vec 알고리즘을 제공한다. 이는 원-핫 인코딩으로 단어를 표현하면 벡터에 단어의 의미가 들어가 있지 않기 때문에 의미가 담겨있지 않은 벡터들을 비교하려고 해도 비교할 수 없다는 단점을 극복하게 해준다. Word2Vec은 단어 간 유사성을 고려하기 위해 단어의 의미를 벡터해주는 라이브러리이다.

10.4 Schedule

PHASE	EXPECTED	ACTUAL
PROPOSAL	4/26	5/3(SUN)
REQUIREMENT SPECIFICATION	5/6	5/13(WED)
DESIGN SPECIFICATION	5/17	5/24(SUN)
FRONT END DEVELOPMENT	5/25	5/31(SUN)
BACK END – APPLICATION SERVER	5/25	5/31(SUN)
FRONT END, SERVER INTEGRATION AND TESTING	5/27	6/2(TUE)
RECOMMENDATION SYSTEM DEVELOPMENT	5/27	6/2(TUE)
SYSTEM INTEGRATION AND TESTING	5/29	6/5(FRI)

11. Index

11.1 Tables

11.2 Figures

11.3 Diagrams