설계 명세서

Design Specification

2019학년도 1학기 소프트웨어공학개론

이은석 교수님

3조

박수현

김선민

이교영

장현수

한대룡

1. Preface

1.1. Objective

Preface에서는 본 문서의 독자를 정의하고, 구조를 소개한다. 구조를 소개할 때는 각 목차의 목적을 서술한다. 또, 문서의 버전에 대해 버전 포맷, 버전 관리 정책, 버전 업데이트 기록으로 나누어서 서술한다.

1.2. Readership

본 문서의 독자는 다음과 같다. 시스템을 직접 개발하는 소프트웨어 엔지니어, 시스템을 설계하는 아키텍처와 개발에 참여하는 모든 구성원을 독자로 정의한다. 만약, 시스템을 개발할 때 외주 업체를 이용한다면, 해당 업체에서 개발에 관련되는 모든 구성원 역시 독자에 포함한다. 즉, 본 문서의 독자는 본 문서에서 소개하는 시스템의 개발 및 유지 보수에 관련된 모든 구성원이다.

1.3. Document Structure

A. Preface

Preface에서는 본 문서의 독자를 정의하고, 구조를 소개한다. 구조를 소개할 때는 각 목차의 목적을 서술한다. 또, 문서의 버전에 대해 버전 포맷, 버전 관리 정책, 버전 업데이트 기록으로 나누어서 서술한다.

B. Introduction

Introduction에서는 본 문서에서 시스템의 설계할 때 사용하는 모든 종류의 다이어그램 및 툴에 관해 서술한다.

C. System Architecture

System Architecture에서는 우리 팀에서 개발하고자 하는 시스템에 대해 전반적으로 서술한다. 시스템의 전체적인 구조를 설명한다..

D. User Management System

실제 사용자가 사용하며 회원 가입과 로그인 기능을 통해 시스템을 이용하는 중 발생하는 데이터를 처리하고 도식화하는 사용자 관리 시스템의 설계를 설명한다.

또한, User가 상품을 Favorites의 담는 기능에 대한 설계를 설명한다. Class diagram, Sequence, diagram, State diagram을 통해 User Management System의 구조를 표현하고 설명한다.

E. Comparing System

사용자가 선택한 product의 각 국가별 가격을 원화로 환전하여 보여주는 Comparing System의 설계를 설명한다. Class diagram, Sequence diagram과 State diagram을 통해 Comparing System의 구조를 표현하고 설명한다.

F. Recommend System

사용자가 주로 검색한 옷과 유사한 옷을 추천해주는 시스템이다. Recommend DB에는 기존 Product DB의 옷 내에서 가장 유사한 옷들이 저장되어 있으며 이 DB를 이용하여 추천을 진행하게 된다. Recommend DB의 정보는 Product DB가 업데이트 될 때마다 함께 업데이트 된다. Recommend system 구현 과정은 다음과 같다. 상품에는 여러 카테고리(ready to wear, shoes, bags)가 존재하는데 이를 구별하지 않고 모든 상품의 이미지를 바탕으로 유사한 이미지를 제공한다. 따라서 해당 상품과 추천된 상품의 카테고리가 같은 경우 이는 유사한 이미지임을 나타내므로 이를 evaluation으로 설정하여 진행한다. Class diagram, Sequence diagram과 State diagram을 통해 Comparing System의 구조를 표현하고 설명한다.

G. Search System

구매자가 실제로 물건을 검색했을 때, 적합한 내용을 데이터베이스에서 불러오고, Search DB에 저장하는 시스템의 설계를 설명한다. Class Diagram, Sequence Diagram, State Diagram 을 통해 Seller Management System 의 구조를 표현하고 설명한다.

H. Protocol Design

Protocol Design에서는 각 각의 하위 시스템들이 상호작용 하는 프로토콜에 대하여 기술한다. Protocol의 기본적인 형식은 JSON으로 정하며, 통신과정의 메시지와 형식, 용도, 의미 등을 설명한다.

I. Database Design

요구사항 명세서에서 기술한 데이터베이스 요구사항을 바탕으로 수정사항을 반영하여 요구사항을 다시 작성하였다. 수정한 데이터베이스 요구사항을 바탕 으로 ER Diagram을 작성하고, 이를 이용하여 Relational Schema를 작성하고, Normalization을 통해 Redundancy와 Anomaly를 제거한 후 마지막으로 SQL DDL을 작성한다.

J. Testing Plan

Testing Plan에서는 테스트 정책과 테스트 케이스에 대해 설명한다. 본 장의 목적은 전체 시스템이 의도한대로 실행되는지를 검증하기 위한 과정을 사전에 계획하는 데에 있다. Testing Policy에서는 testing에 대한 단계적 접근을 설명 한다. Test Case에서는 각 Sub-System에서의 예시 케이스를 토대로 사용자를 판별하고 그에 따라 시스템에 기대하는 입, 출력 동작을 서술한다.

K. Development Environment

Development Environment에서는 개발자의 환경에 대해 설명한다. 사용한 프로그래밍 언어와 IDE에 대해 서술한다.

L. Development Plan

Development plan에서는 개발 계획에 대해 서술한다. 이때, Gantt Chart를 이용하여 개발 계획과 실제 개발 흐름에 대해 서술하고자 한다.

M. Index

Index에서는 본 문서의 표, 다이어그램 및 사진들의 인덱스를 표시한다. 이를 통해 원하는 정보가 문서의 몇 페이지에 있는 지 알 수 있다.

1.4. Version of the Document

A. Version Format

버전 번호는 Major number와 Minor number로 이루어져 있으며, (Major number).(Minor number)의 형태로 표현한다. 문서의 버전은 0.1부터 시작한다.

B. Version Management Policy

설계명세서를 수정할 때마다 버전을 업데이트한다. 다만 변경 간의 간격이 1시간 이내일 때에는 버전 번호를 업데이트하지 않고, 하나의 업데이트로 간주한다. 이미 완성된 파트 를 변경할 때에는 Minor number를 변경하며, 새로운 부분을 추가하거나 문서의 구성이 예전에 비해 괄목할 변화가 있을 경우 Major number를 변경한다.

C. Version Update History

Version	Modified Date	Explanation
1.0	2019.05.09	Sub-System 별 초안 완성
2.0	2019.05.12	Testing Plan 및 Database 확정
2.1	2019.05.14	Sub-System 최종 확정
3.0	2019.05.16	Preface, Introduction, Development Plan 완성
4.0	2019.05.19	전체 취합 및 완성
4.1	2019.06.09	Gantt Chart 추가

Table 1 Version Update History

2. Introduction

2.1. Objectives

Introduction에서는 해당 시스템 설계명세서 상에서 사용하는 모든 종류의 다이어그램 및 툴에 관해 설명한다.

2.2. Applied Diagram

A. UML

UML(Unified Modeling Language)은 객체 관련 표준화 기구 OMG(Object Modeling Technique)에서 발표한 모델링 언어로 시스템 개발 과정에서 요구사항을 분석하고 시스템의 설계 및 구현 하는 단계에서 개발자 간의 의사소통을 원활하게 하기 위해 만들어졌다. UML은 시스템 개발에서의 필요한 내용을 규칙에 따라 정해진 다이어그램을 활용하여 도형으로 나타나게 된다. 따라서, 프로그램에 전문적인 지식이 없더라도 시스템 개발 과정에서 전문가와 비전문가가 서로 대화 할 수 있는 도구로서 사용될 수 있다. UML은 프로그래밍 코드 언어, 시스템 규모와 관계없이 표현될 수 있으므로, 표현력이 높고 여러 부서간, 개발자간의 의사소통이 용이하다.

B. Package Diagram

Package란 해당 프로그램상에서 프로그램 컴포넌트로 나누었을 때, 개발자가 관리하기 쉬운 단위의 작은 집합이라고 볼 수 있다. 패키지는 기능적으로 관련된 여러 하위 class들로 이루어져있으며, 패키지 단위는 독립적인 형태로 다른 시스템에 응용하여 사용할 수 있다.
패키지는 UML 시스템 모델의 기본 구성 요소로, 전체 시스템이라 함은 다른 모든 패키지들, 다이어그램 및 요소를 포함 하는 개념으로 볼 수 있다. 하나의 패키지 또한 다른 패키지들로도 구성되는 상위의 패키지가 될 수도 있다.
본 문서 상의 패키지 다이어그램은 각 class별 속성(Attribute) 및 해당 클래스의 기능(Methods)을 보여주고 상속 등 클래스와 클래스 간의 관계를 나타낸다.

C. Deployment Diagram

Deployment diagram이란 해당 시스템을 작동하기 위해 사용되는 네트워크, 하드웨어 및 소프트웨어 등이 어떻게 배치되어 있고 이러한 것들 간의 관계가 어떻게 나타나는 지를 보여주는 diagram을 말한다. 이들의 배치 상태를 다이어그램으로 표현함으로써 시스템의 구성요소 및 리소스에 대한 내용을 보다 빠르게 이해할 수 있게 된다.

Deployment diagram 상에서 사용되는 도형은 직육면체로 시스템에서 업무를 처리하는 능력을 가진 장치의 단위인 노드를 의미한다. 해당 다이어그램은 노드의 단위가 정해지고, 컴포넌트가 정의된 뒤 하드웨어의 사양이 확정되는 시점에 작성되어야 하므로, 시스템이 전부 완성된 뒤에 작성된다.

D. Class Diagram

클래스 다이어그램이란, 객체지향 프로그램 상에서 가장 기본이 되는 단위인 class들이 주체가 되어 나타나는 다이어그램이다. 해당 클래스에 대한 상세한 정보를 담고 있다. 예컨대, class의 속성값(Attribute), 메서드(Method) 과 해당 클래스에서 상속받고 있는 클래스, 구현한 인터페이스 등의 클래스와 관련된 내용은 모두 보여지게 된다. 클래스 다이어그램을 바탕으로 실제 코드를 생성하게 되므로 구체적으로 나타나게 된다.

E. State Diagram

State Diagram은 event-oriented diagram으로 시스템 내에서 해당 이벤트가 어떻게 작동하는 지를 다이어그램으로 상세히 기술하게 된다. 주로 상태가 변화하여 어떤 결과값이 도출 된다거나 상태가 다른 상태로 변화하는 경우, 시스템은 어떻게 돌아가는 지를 상태 중심적으로 표현한다. 따라서 해당 다이어그램을 통해 시스템 내에 어떤 상태가 존재하는지, 상태의 변화가 어떤 형태로 일어나게 되는 지를 알 수 있으며, 상태에 대한 시스템의 반응도 알아 볼 수 있다.

F. Sequence Diagram

Sequence Diagram은 User의 행위를 시작점으로 시스템 내에 actor로서 각 컴포넌트들간의 어떤 데이터의 흐름이 생성되고 진행되는 지를 시간에 따라 순차적으로 표현한 다이어그램이다. 이 다이어그램에서 컴포넌트 별 데이터의 상호작용이 명확하게 나타나게 되므로 각 컴포넌트의 역할이 명확하게 나타나며 DB에 접근하는 경우 접근 순서도 알아볼 수 있으므로 유용하다. 주로 시스템의 동적인 측면을 모델링하기 때문에 컴포넌트 사이의 데이터 교류, 메시지, 회귀메시지 및 제어블록 등을 표현하게 된다.

G. ER Diagram

ER Diagram은 데이터베이스에서 사용되는 다이어그램으로, 각 개체들 간의 관계를 표현하고 있다. 앞선 다이어그램들과 달리 UML에서 지정된 다이어그램은 아니며, 이와는 별개로 시스템의 데이터베이스를 구성하는 과정에서 entity간의 관계를 도형을 통해 알아볼 수 있다.

ER은 entity-relationship으로, 하나의 개체(entity)는 분리된 물체 하나를 표현한다. 개체는 사각형으로 표현되며, 관계(relationship)는 다이아몬드로 표현된다. 개체나 관계는 특성을 가질 수 있으며, 이 특성들은 관계 집합에 실선으로 연결된 타원형으로 표현한다.

2.3. Applied Tool

A. Diagram Tool

본 문서에서 UML을 활용하여 작성하는 모든 다이어그램(Sequence diagram, Class Diagram 등)은 draw.io라는 웹페이지에서 제공하는 tool을 활용하여 작성한다. 구글 드라이브와 연동이 되어 있어 작성자의 파악이 쉬우며, 실시간으로 수정사항에 대한 반영 및 공유가 가능하기 때문에 유용하다.

3. System Architecture

3.1. Objective

System Architecture 에서는 해당 시스템의 Architecture에 대한 고수준에서의 개요를 보여준다. 또한 시스템 기능의 전체적 분포를 보여준다.

3.2. System Organization

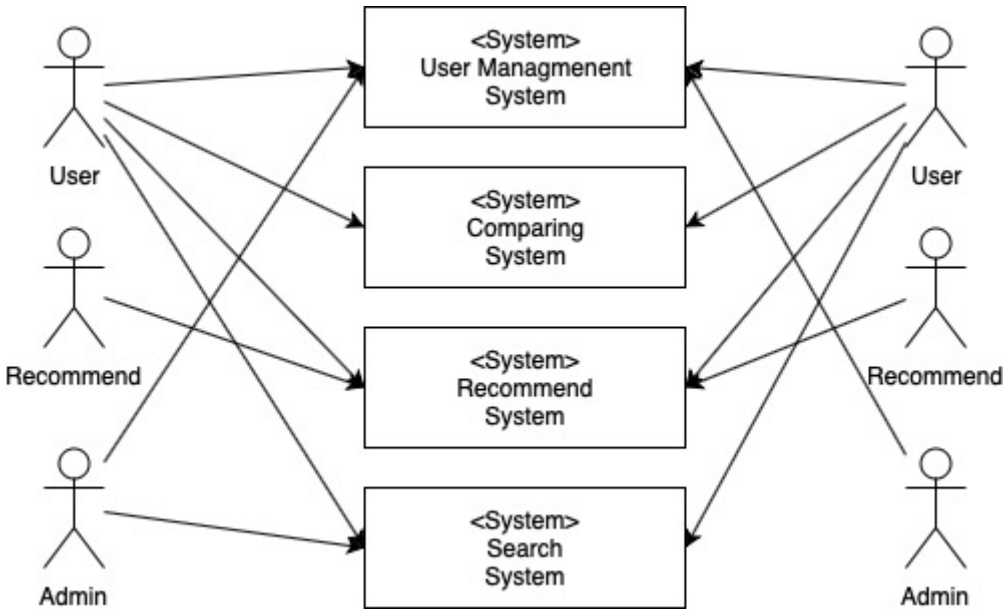

Figure 1 System Architecture

A. User Management System

User Management System은 사용자가 서비스를 이용하기 위한 회원가입, 로그인 등의 서비스와 Favorites에 담은 상품 정보를 열람할 수 있는 기능 및 상품 추가, 삭제에 대한 기능과 관련된 시스템이다.

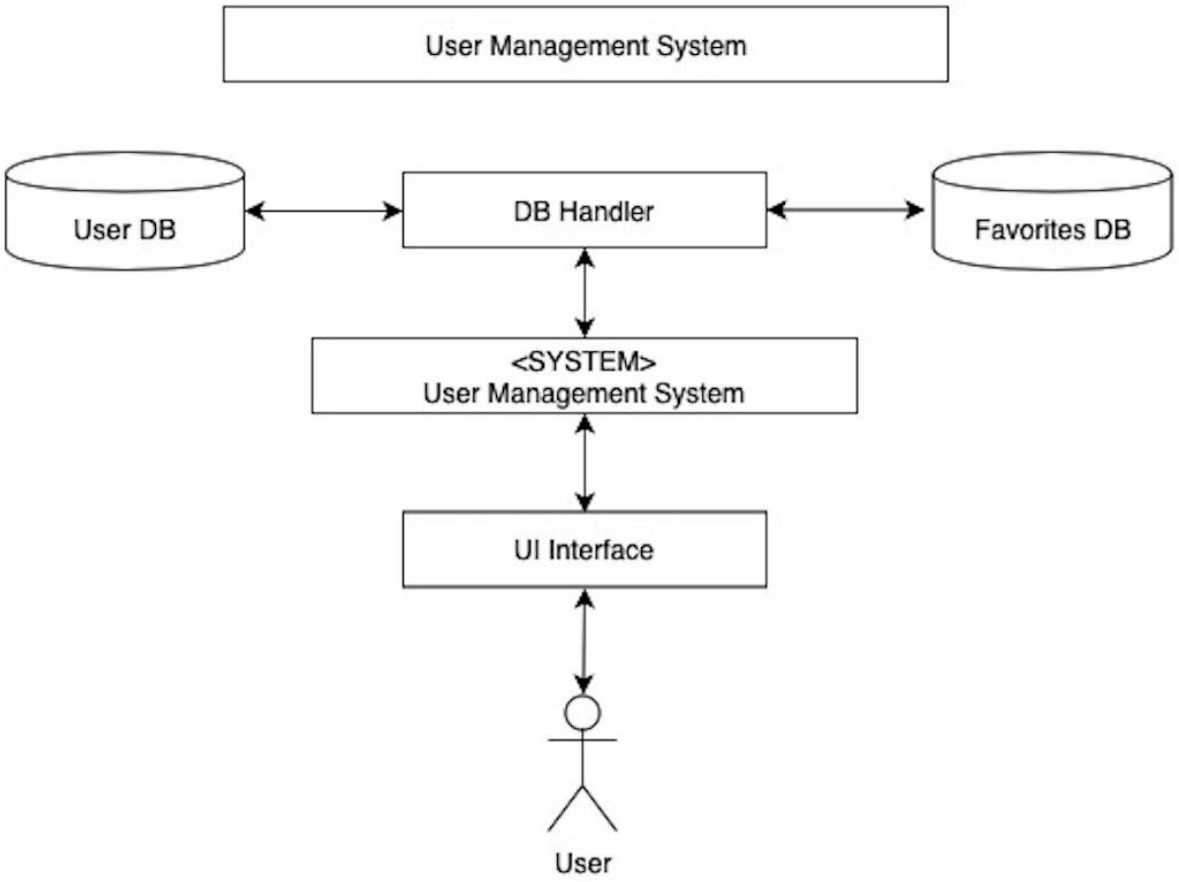

Figure 2 User Management System Architecture

B. Comparing System

comparing system은 사용자가 검색한 product의 각 국가별 가격을 원화로 환전하여 보여주는 시스템이다. Search System을 통해 product가 검색될 때 product DB에서 해당 상품의 각 국가별 가격정보를 가져온다. 이 정보를 실시간 환율API를 이용해 원화로 환전하여 사용자에게 제공한다.

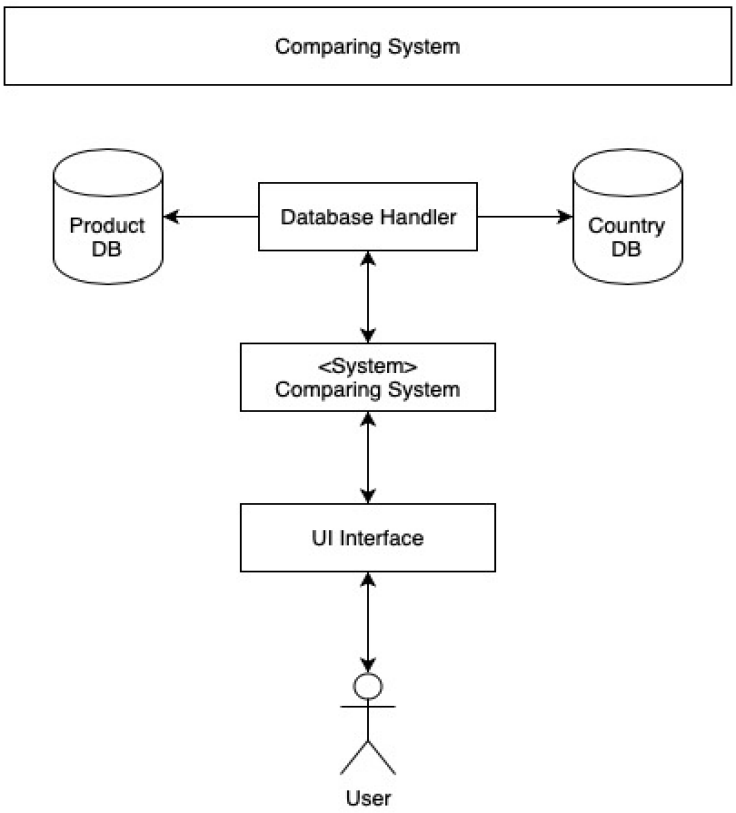

Figure 3 Comparing System Architecture

C. Recommend System

Recommend System은 사용자가 검색한 옷과 유사한 옷을 추천해주는 시스템이다. 현재 검색된 상품과 유사한 상품을 Recommend DB에서 검색해 가져온다. 상품의 상세 페이지의 하단에 유사 상품 N개를 보여준다. Recommend DB의 정보는 Product DB가 업데이트 될 때마다 함께 업데이트 된다.

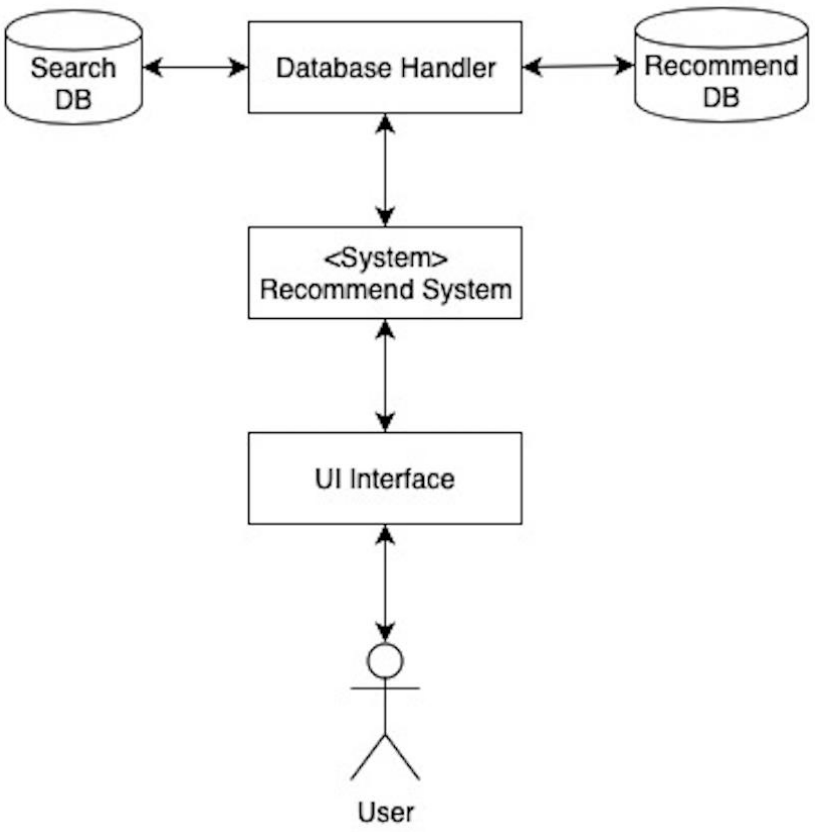

Figure 4 Recommend System Architecture

D. Search System

Search System은 사용자가 Product를 검색하고자 할 때 사용되는 시스템이다. 상품명이나 상품 code를 입력하면 Product DB에서 정보를 가져온다. 가져온 상품 정보의 목록은 result list로 사용자에게 보여지게 된다. 사용자가 상품을 클릭하게 되면 Search DB에 사용자의 Search log 가 저장되고, Product DB에서 해당 상품의 상세 정보를 가져와서 result 값으로 보여진다. 그리고 Product DB에서 조회수를 1 증가시킨다.

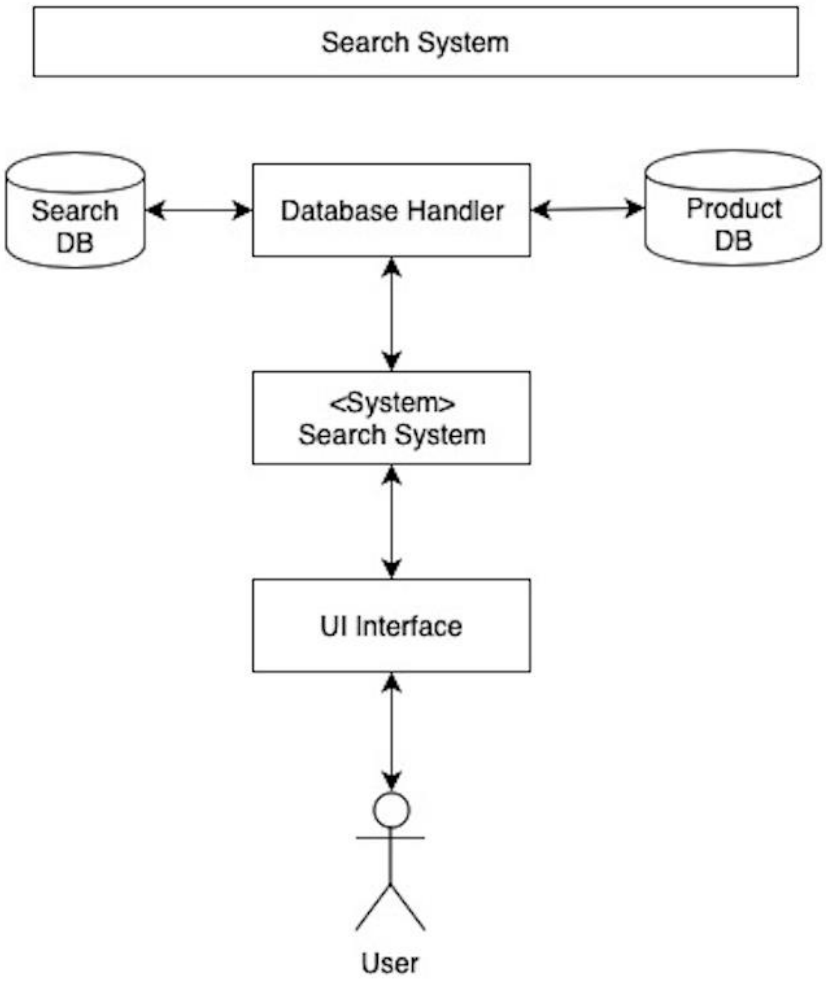

Figure 5 Search System Architecture

3.3. Package Diagram

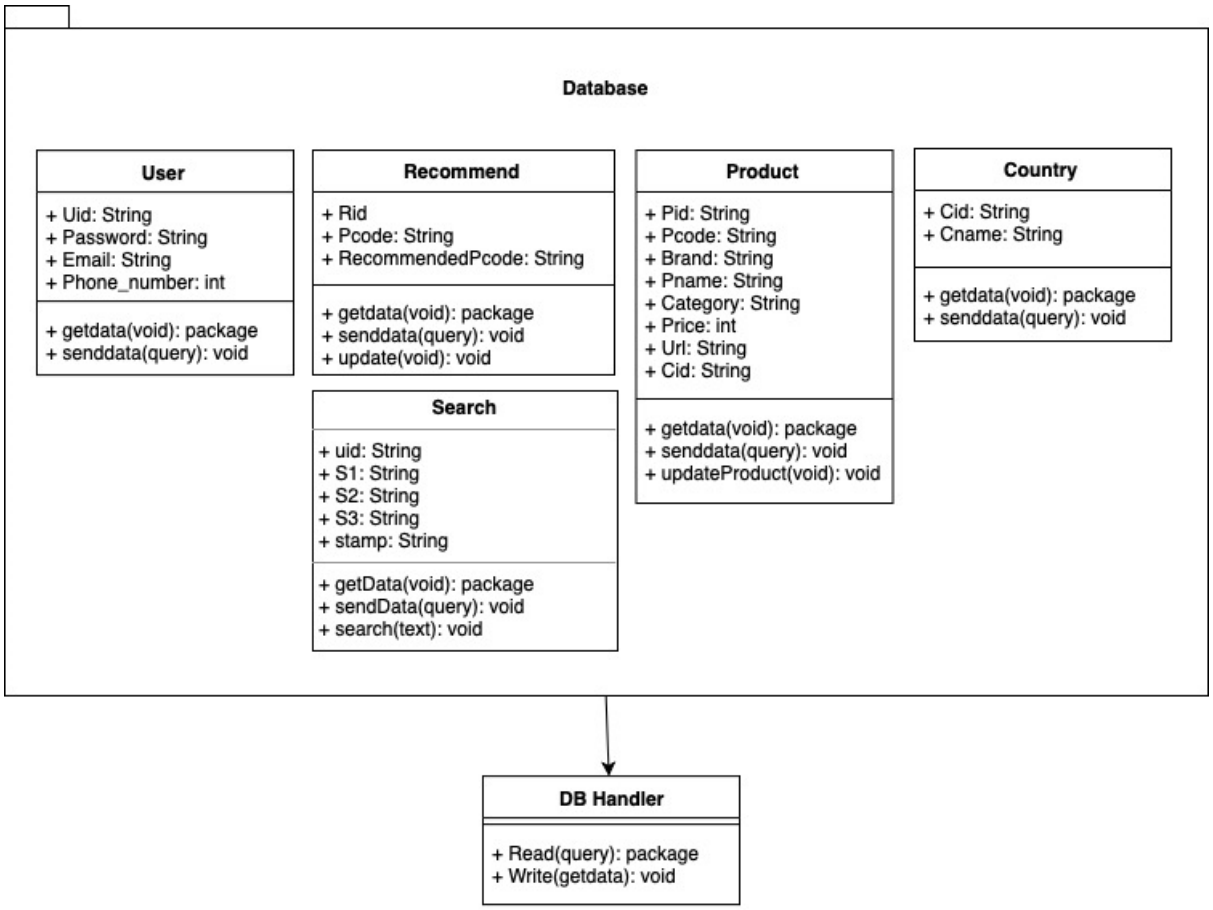

Diagram 1 Package Diagram

3.4. Deployment Diagram

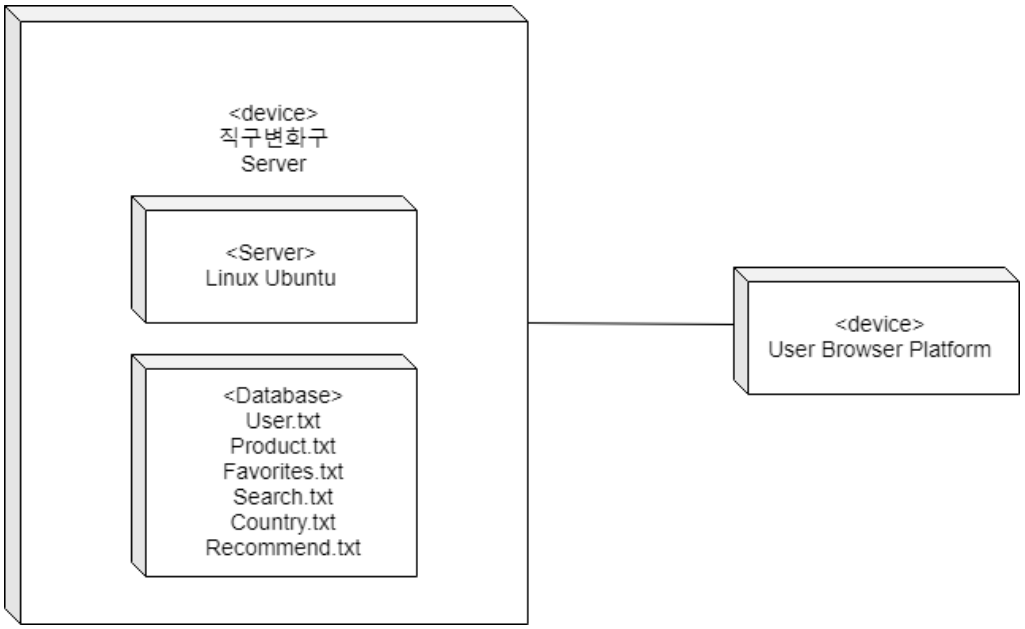

Diagram 2 Deployment Diagram

4. User Management System

4.1. Objectives

회원 가입과 로그인 과정에서 발생하는 데이터의 처리를 진행하는 사용자 관리 시스템의 설계를 설명한다. 또한 관심상품(Favorites)에 담는 시스템의 설계를 설명한다. Class diagram, Sequence diagram과 State diagram을 통해 User Management System의 구조를 표현하고 설명한다.

4.2. Class Diagram

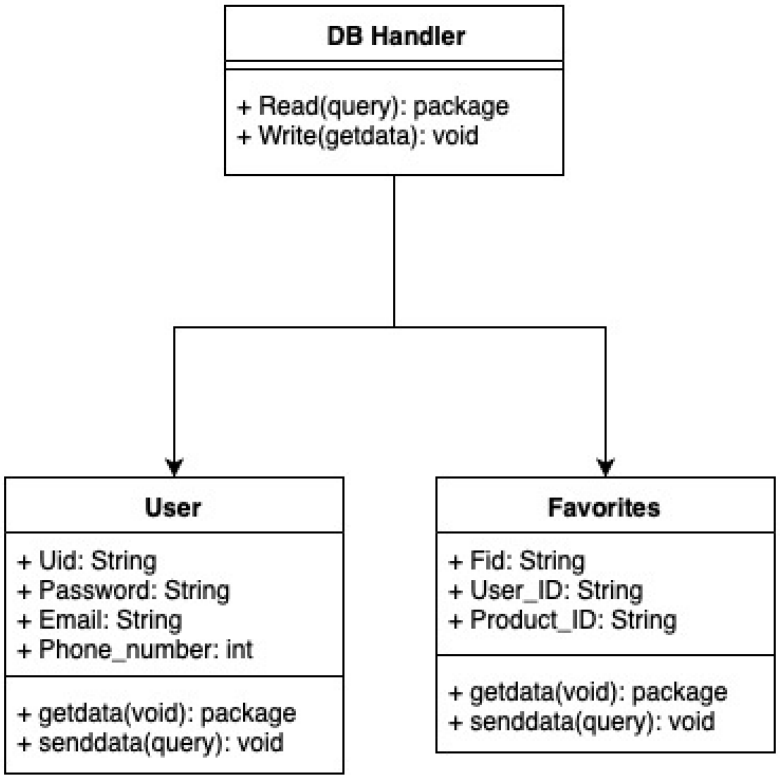

Diagram 3 User Management System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+package Read(query) : 해당되는 DB에서 원하는 데이터를 읽어온다.

+void Write(package) : 해당되는 DB에 데이터를 저장한다.

B. User

B.1. Attributes

+ Uid : 해당 User 계정의 ID

+ Password : 해당 User 계정의 Password

+ Email : 해당 User 계정 사용자의 연락가능한 Email

+ Phone_number : 해당 User 계정의 연락처

B.2. Methods

+ package getdata() : 해당되는 DB에서 원하는 데이터를 얻어온다.

+ void senddata(package) : 해당되는 DB에 데이터를 보낸다.

C. Favorites

C.1. Attributes

+ Fid : Favorites를 구분하는 ID

+ User_ID : 아래의 Product를 추가한 User 계정의 ID

+ Product_ID : User가 추가한 Product의 ID

C.2. Methods

+ package getdata() : 해당되는 DB에서 원하는 데이터를 얻어온다.

+ void senddata(package) : 해당되는 DB에 데이터를 보낸다.

4.3. Sequence Diagram

A. Sign in

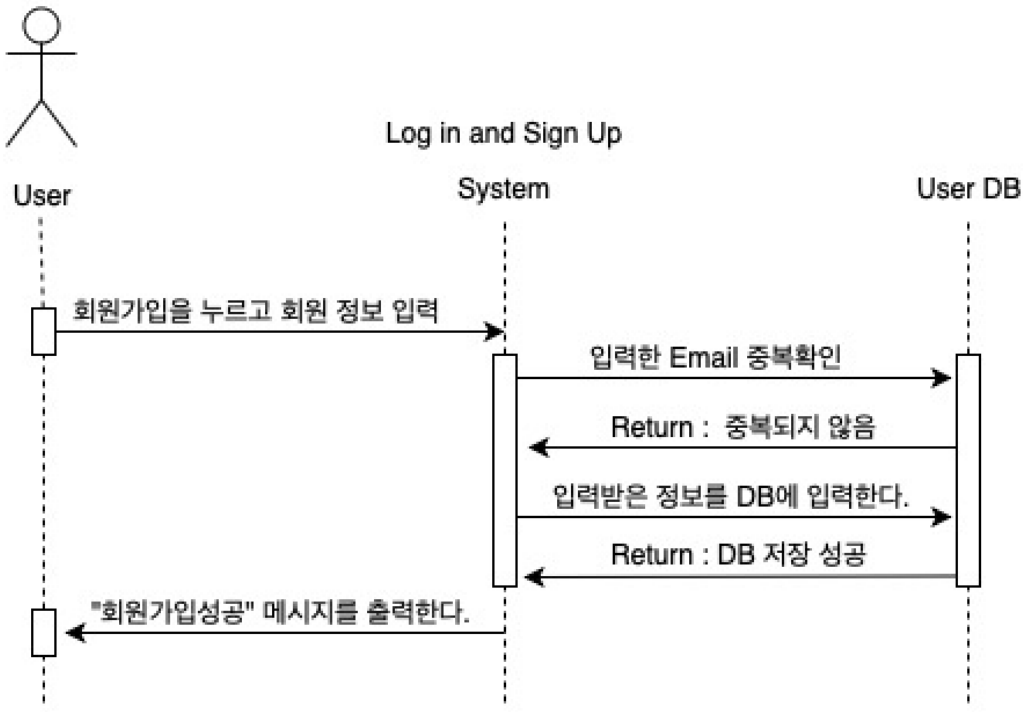

Diagram 4 User Management System Sign-in Sequence Diagram

B. Log in

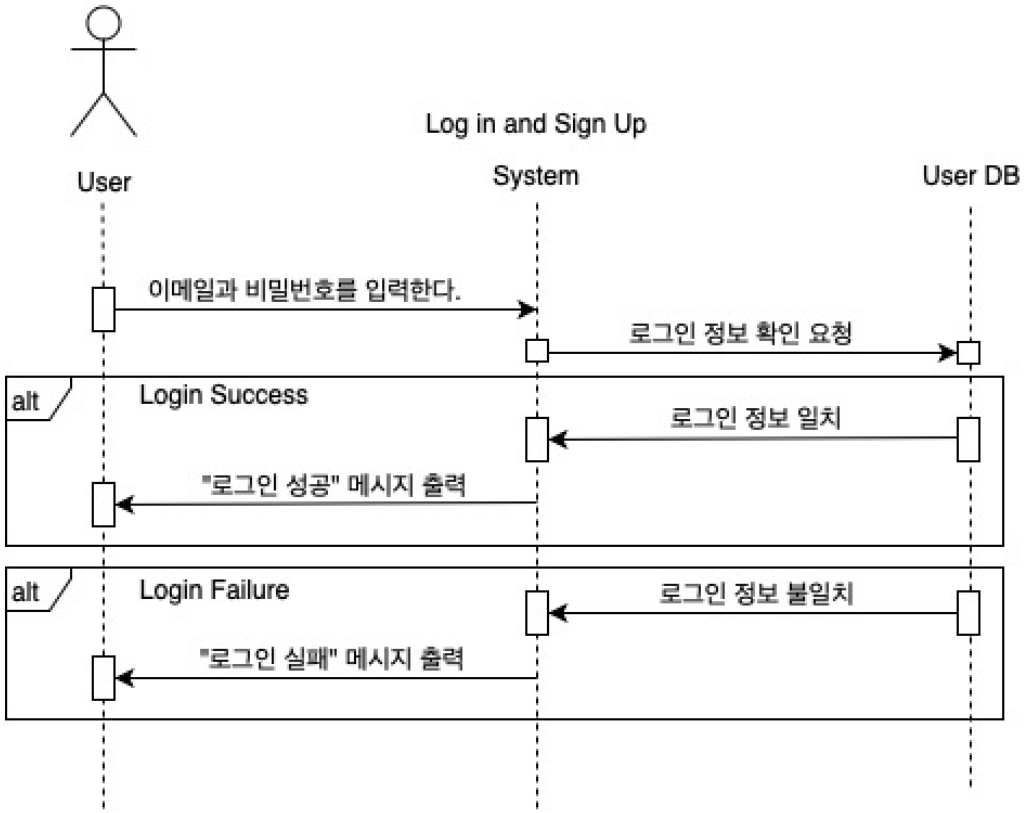

Diagram 5 User Management System Log-in Sequence Diagram

C. Managing Favorites

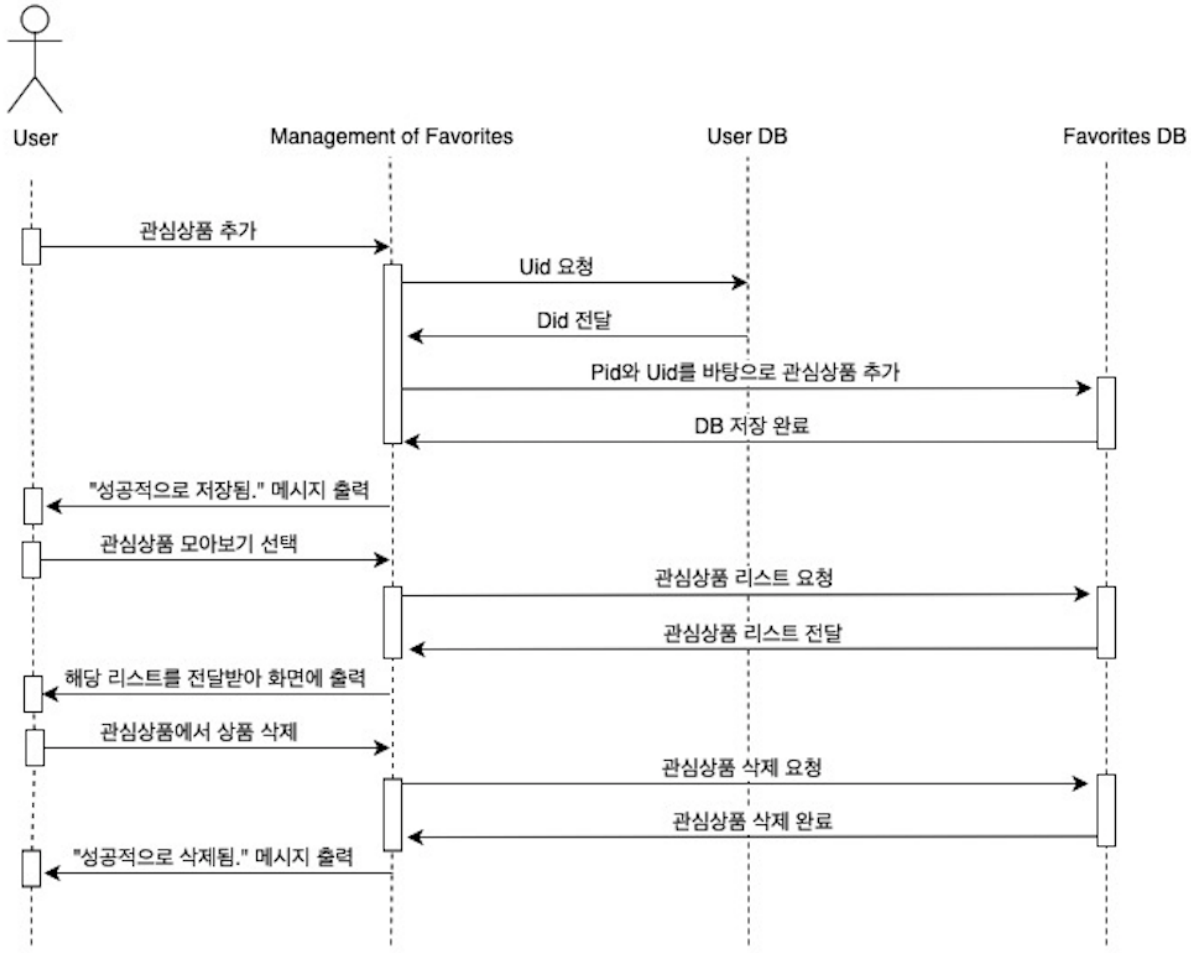

Diagram 6 User Management System Managing Favorites Sequence Diagram

4.4. State Diagram

A. Sign Up

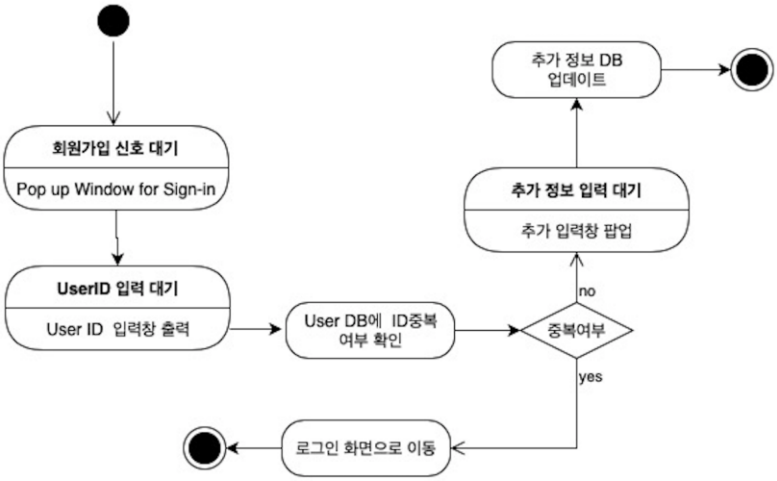

Diagram 7 User Management System Sign-Up State Diagram

B. Log in

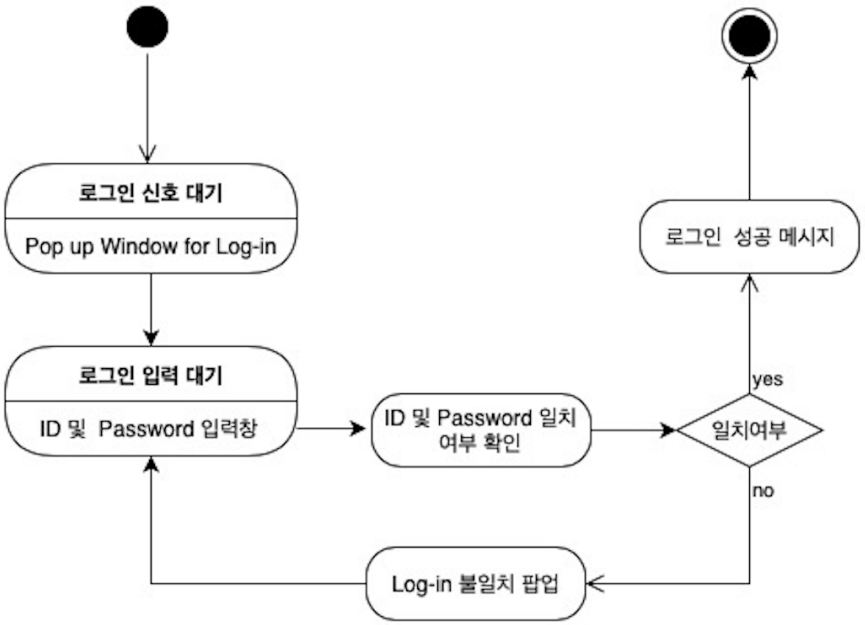

Diagram 8 User Management System Log-in State Diagram

C. Managing Favorites

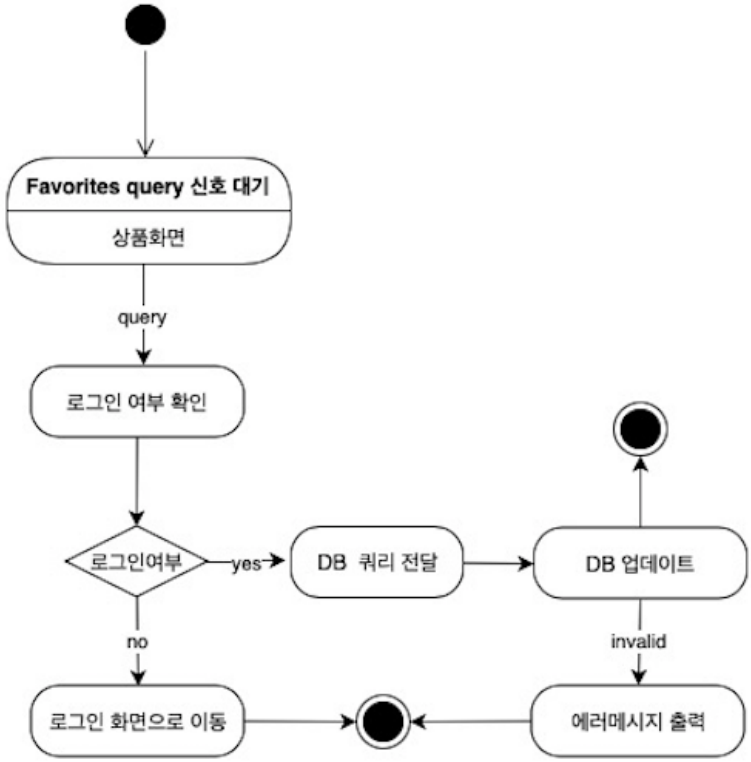

Diagram 9 User Management System Managing Favorites State Diagram

5. Comparing System

5.1. Objectives

사용자가 선택한 product의 각 국가별 가격을 원화로 환전하여 보여주는 Comparing System의 설계를 설명한다. Class diagram, Sequence diagram과 State diagram을 통해 Comparing System의 구조를 표현하고 설명한다.

5.2. Class Diagram

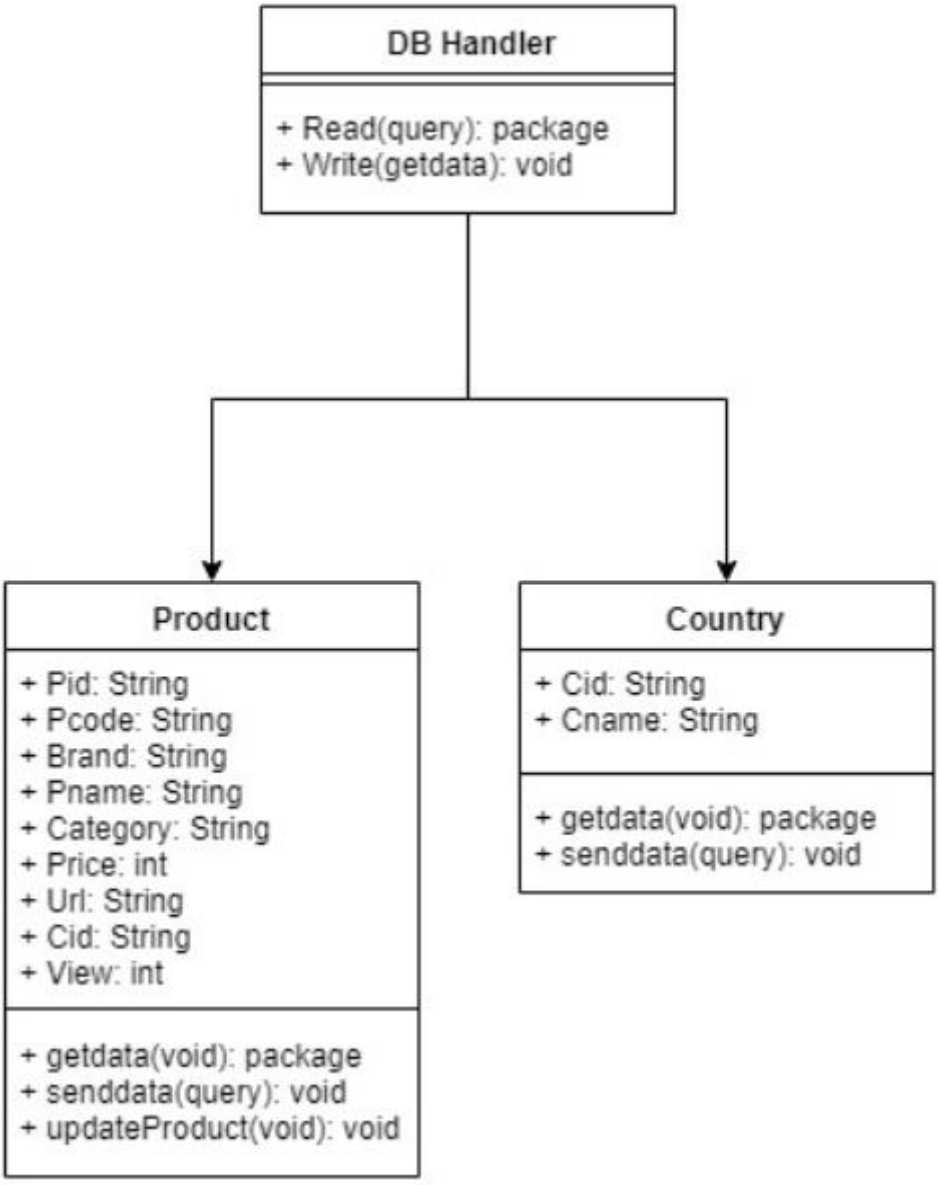

Diagram 10 Comparing System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+package Read(query) : 해당되는 DB에서 원하는 데이터를 읽어온다.

+void Write(package) : 해당되는 DB에 데이터를 저장한다.

B. Product

B.1. Attributes

+ Pid : 해당 Product의 id

+ Pcode: 해당 Product의 상품 품번

+ Brand: 해당 Product의 브랜드 이름

+ Pname: 해당 Product의 상품명

+ Category: 해당 Product의 상품 카테고리

+ Price: 해당 Product의 상품 가격

+ Url: 해당하는 국가의 상품 상세 페이지 url

+ Cid: 해당 Product의 국가 코드

+ View: 해당 Product의 조회수

B.2. Methods

+ package getdata() : 해당되는 DB에서 원하는 데이터를 얻어온다.

+ void senddata(package) : 해당되는 DB에 데이터를 보낸다.

+ void updateProduct(): 크롤링으로 상품 데이터를 업데이트한.

C. Country

C.1. Attributes

+ CID : 해당 country의 id

+ Cname: 해당 국가의 이름

C.2. Methods

+ package getdata() : 해당되는 DB에서 원하는 데이터를 얻어온다.

+ void senddata(package) : 해당되는 DB에 데이터를 보낸다.

5.3. Sequence Diagram

A. Comparing

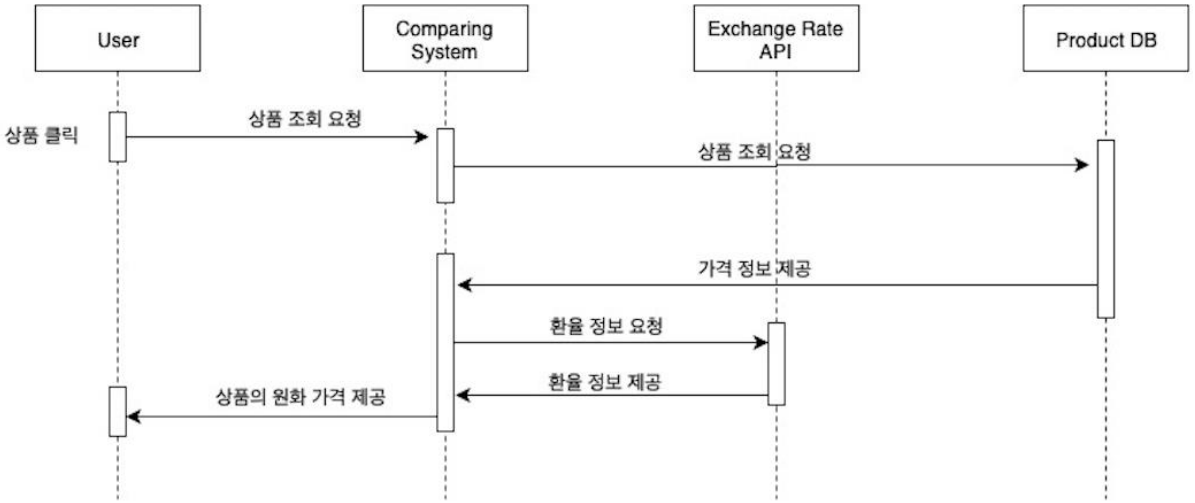

Diagram 11 Comparing System Sequence Diagram

5.4. State Diagram

A. Comparing

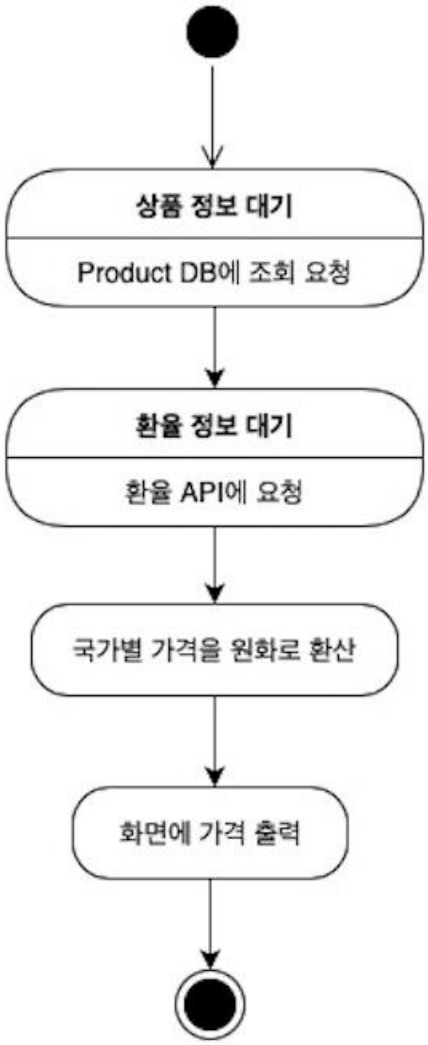

Diagram 12 Comparing System State Diagram

6. Recommend System

6.1. Objectives

사용자가 주로 검색한 옷과 유사한 옷을 추천해주는 시스템이다. Recommend DB에는 기존 Product DB의 옷 내에서 가장 유사한 옷들이 저장되어 있으며 이 DB를 이용하여 추천을 진행하게 된다. Recommend DB의 정보는 Product DB가 업데이트 될 때마다 함께 업데이트 된다. Class diagram, Sequence diagram과 State diagram을 통해 Comparing System의 구조를 표현하고 설명한다. 그리고 마지막으로 추천시스템에 이용한 모델과 추천된 상품의 평가 방법에 대해 소개한다.

6.2. Class Diagram

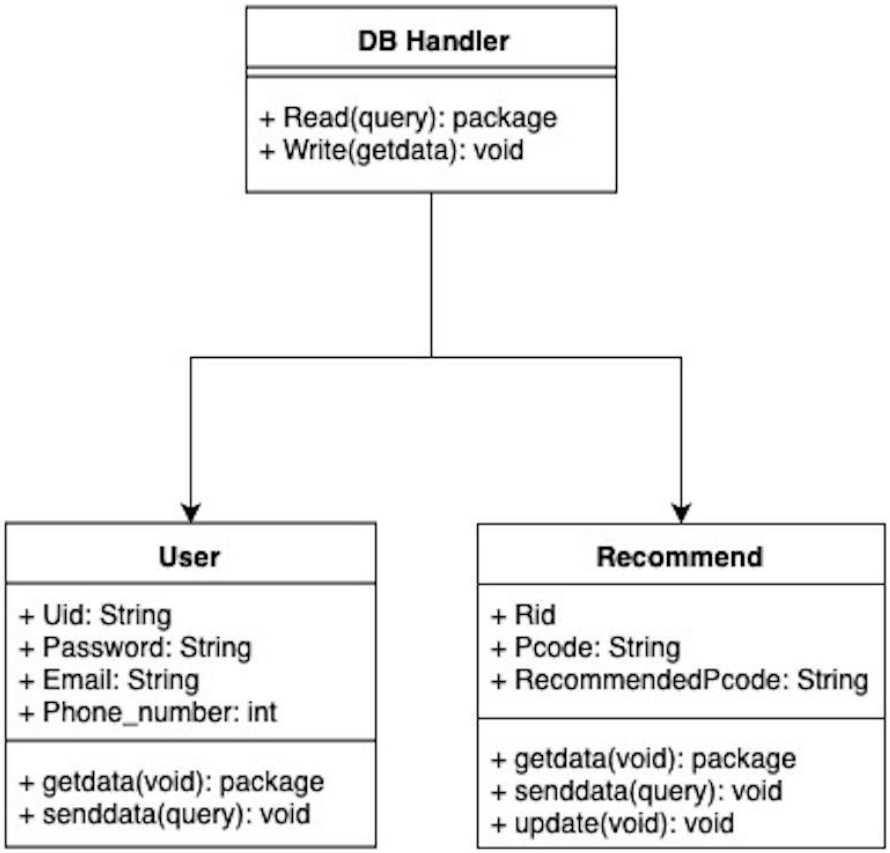

Diagram 13 Recommend System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+package Read(query) : 해당되는 DB에서 원하는 데이터를 읽어온다.

+void Write(package) : 해당되는 DB에 데이터를 저장한다

B. User

B.1. Attributes

+ ID : 해당 User의 id

+ Password: 해당 User의 비밀번호

+ Email: 해당 User의 Email

+ Phone_number: 해당 User의 핸드폰 번호

B.2. Methods

+ package getdata() : 해당되는 DB에서 원하는 데이터를 얻어온다.

+ void senddata(package) : 해당되는 DB에 데이터를 보낸다.

C. Recommend

C.1. Attributes

+ Pcode : product 고유번호 값

+ recommended pcode : 해당 코드 product와 유사한 product의 고유 번호 값

C.2. Methods

+package getData(): 해당되는 DB에서 원하는 data를 읽어온다.

+void sendPquery(Personalizing): DB Handler에 Personalized query를 보낸다.

+void update(): Product DB의 변경사항을 반영하여 Recommend DB를 갱신한다.

6.3. Sequence Diagram

A. Recommend

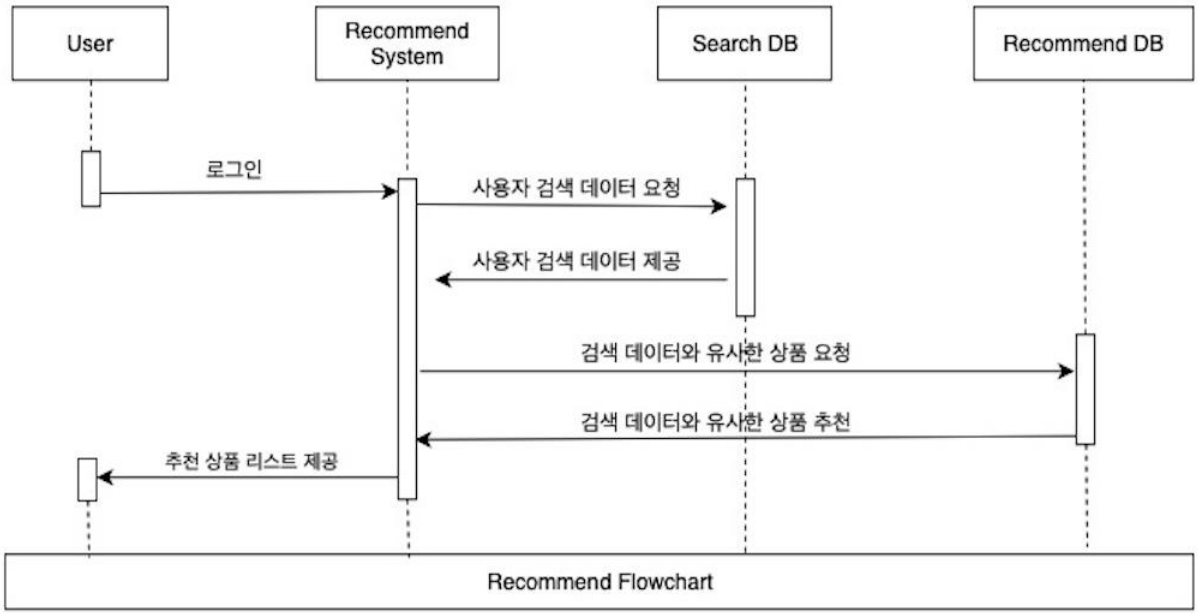

Diagram 14 Recommend System Sequence Diagram

6.4. State Diagram

A. Recommend

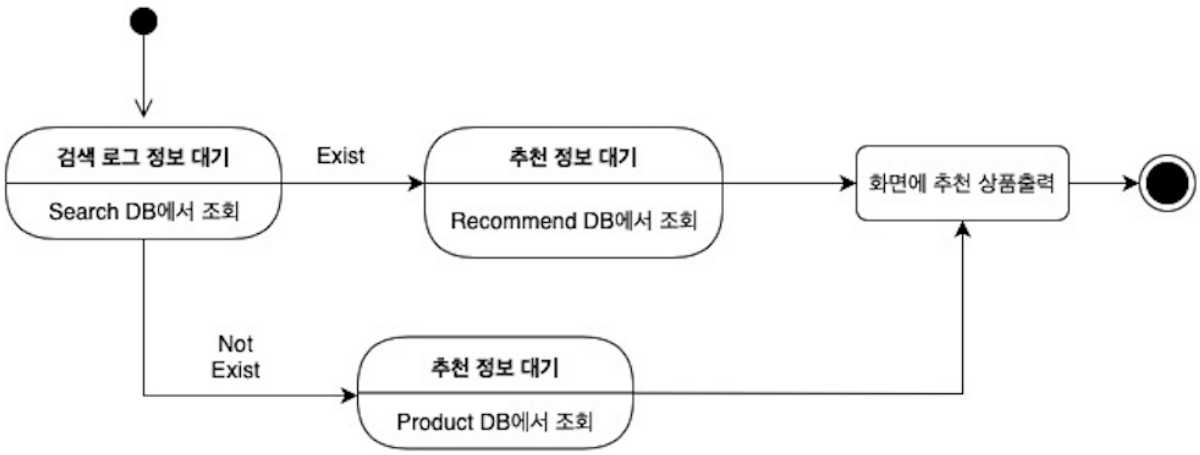

Diagram 15 Recommend System State Diagram

6.5. Model and Evaluation

A. Model

Transfer Learning의 대표적인 모델인 VGG16 모델을 이용하였다. 물론 우리가 갖고 있는 데이터 만으로 학습을 시켜 시스템을 구현할 수도 있었다. 하지만, 샘플의 수가 그리 많지 않고, 또한 이미지 데이터가 ready to wear, shoes, bags라는 카테고리의 이미지 밖에 존재하지 않았기에 이를 통한 feature extraction은 제한적일 것이라 판단하였다. 따라서, 이미 수많은 데이터를 바탕으로 학습이 된 모델을 이용하여 이미지의 특징을 추출하는 것이 효율적이고, 좋은 성능을 낼 것이라 판단하였다.

우리의 목적은 이미지 기반 추천이기에 target variable이 존재하지 않았으므로 일반적인 Transfer Learning처럼 VGG16 모델을 이용하여 feature을 추출하고 이를 다시 학습시키는 방법을 이용한 것이 아니라 추출한 feature를 바탕으로 cosine similarity를 구해서 가장 유사한 상품을 제공하는 방식으로 진행하였다.

B. Evaluation

앞서 말한 것처럼 target variable이 존재하지 않으므로 기존의 딥러닝 모델의 성능 측정 metric을 이용하지 않았으며 다음과 같은 방식으로 평가하였다. 유사한 이미지를 구할 때 상품 카테고리를 따로 구분하지 않고 모든 이미지 중에 거리가 가장 가까운 이미지를 제공하였다.

그런데 카테고리가 같은 경우 이미지 역시 대체로 같기 때문에 따로 카테고리를 구분하지 않아도 같은 카테고리의 상품이 제공된다면 유사한 이미지를 제공하는 가에 대한 평가로 사용할 수 있을 것이라 판단하였다. 따라서 전체 상품에 대하여 목표 상품과 유사한 이미지 상품의 카테고리를 비교하여 둘의 카테고리가 얼마나 일치하는지를 바탕으로 평가하고자 한다.

6.1. Objectives

사용자가 주로 검색한 옷과 유사한 옷을 추천해주는 시스템이다. Recommend DB에는 기존 Product DB의 옷 내에서 가장 유사한 옷들이 저장되어 있으며 이 DB를 이용하여 추천을 진행하게 된다. Recommend DB의 정보는 Product DB가 업데이트 될 때마다 함께 업데이트 된다. Class diagram, Sequence diagram과 State diagram을 통해 Comparing System의 구조를 표현하고 설명한다.

7. Search System

7.1. Objectives

구매자가 실제로 물건을 검색했을 때, 적합한 내용을 데이터베이스에서 불러오고, Search DB에 저장하는 시스템의 설계를 설명한다.

7.2 Class Diagram

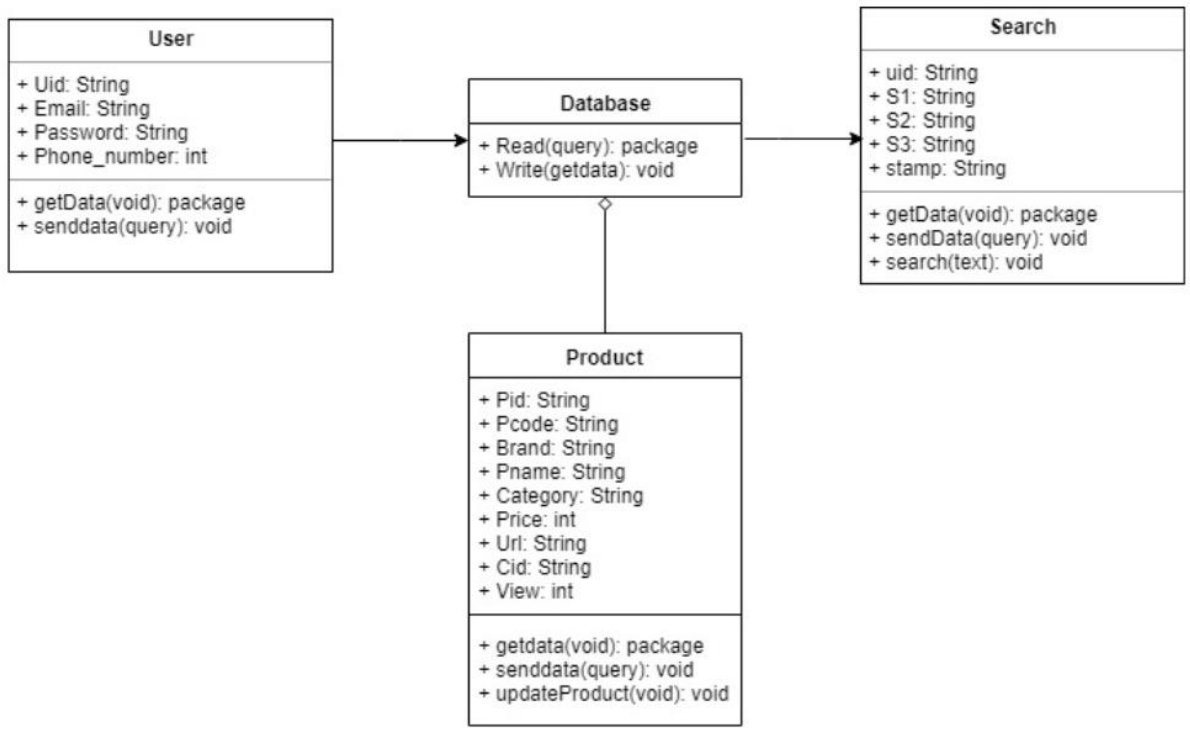

Diagram 16 Search System Class Diagram

A. DB Handler

A.1. Attribute

해당 사항 없음.

A.2. Methods

+package Read(query): 해당되는 DB에서 원하는 data를 읽어온다

+void Write(getdata): 해당되는 DB에 data를 저장한다.

B. Search

B.1. Attributes

+uid: user 고유 번호 값

+S1: 검색 정보 1

+S2: 검색 정보 2

+S3: 검색 정보 3

+stamp: 타임 스탬프 정보

B.2. Methods

+package getdata(): 해당되는 DB로부터 data를 가져온다.

+void sendData(query): 해당되는 DB에 data를 보낸다.

+void search(text): text를 입력 받는다.

7.3. Sequence Diagram

A. Search

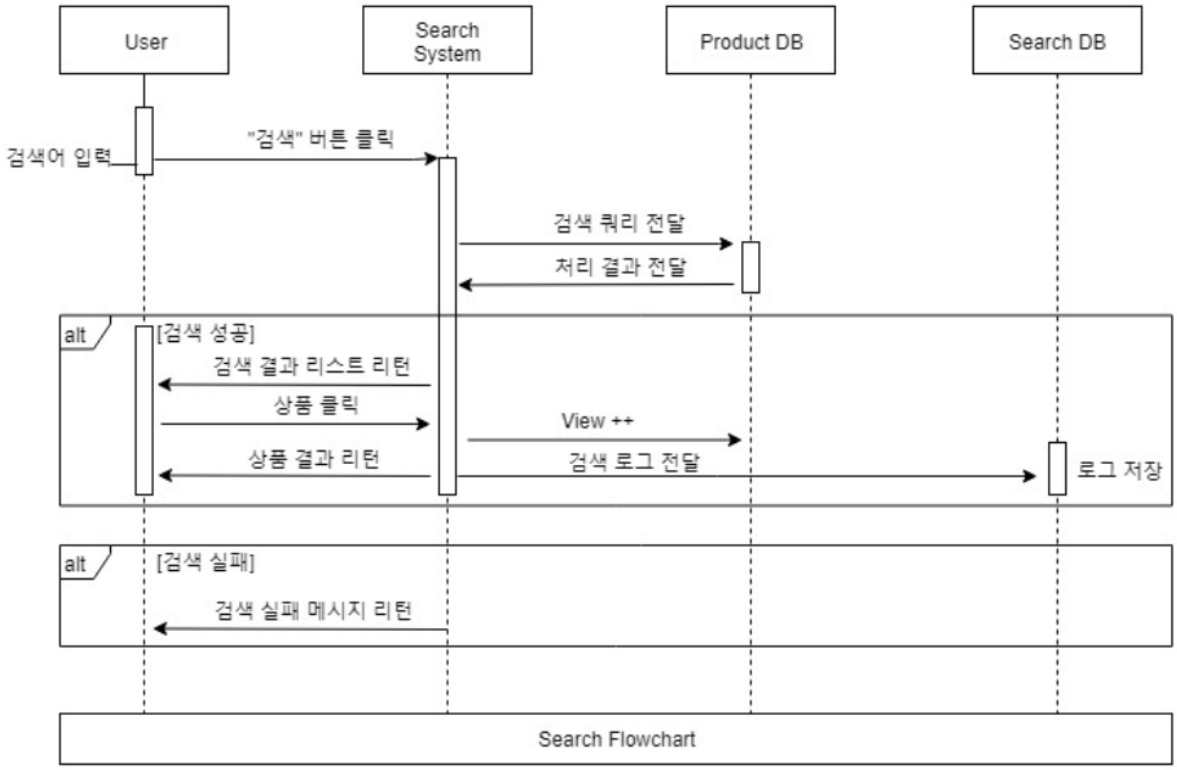

Diagram 17 Search System Sequence Diagram

7.4. State Diagram

A. Search

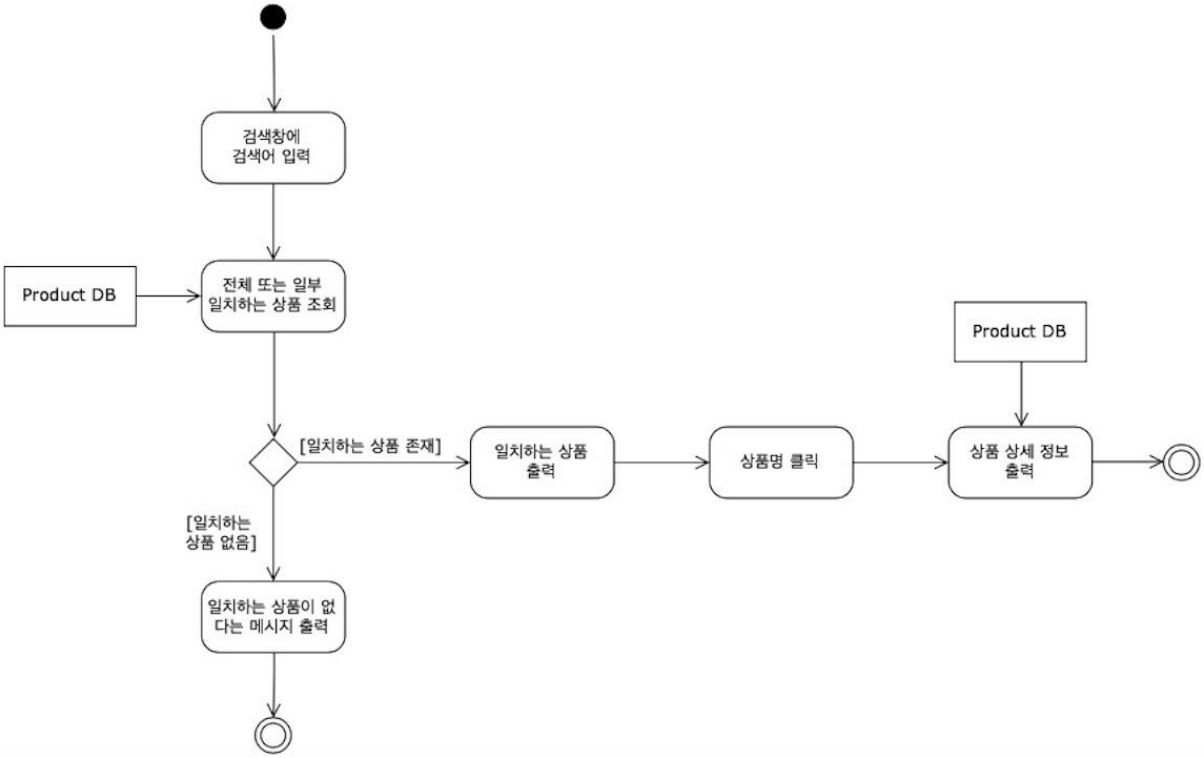

Diagram 18 Search System State Diagram

8. Protocol Design

8.1. Objectives

Protocol Design에서 각 하위 시스템들이 상호작용하는 프로토콜에 대하여 기술한다. 해당 프로토콜을 요청(request)하는 메시지, 응답(response)하는 메시지로 구분되어 각 메시지별 속성(Attribute)와 값(Value)를 표현한다.

8.2. Protocol Description

A. Overview

HTTP 통신에서 Client와 Server 사이에 전송되는 메시지의 형태를 용도별로 정의한다. Client의 요청(Request) 메시지와 응답(Response) 메시지로 구분한다. 해당절의 표는 캡션을 생략한다.

B. Login Protocol

a. Request

Attribute	Value
ID	User의 ID
Password	User의 password

b. Response

Attribute	Value
login_success	로그인 성공
login_fail	로그인 실패

C. User Registration Protocol

a. Request

Attribute	Value
ID	User의 ID
Password	User의 Password
Phone	User의 연락처
email	User의 email

b. Response

Attribute	Value
register_success	회원가입 성공
register_fail	회원가입 실패

D. Favorites Add/Delete Protocol

a. Request(Add)

Attribute	Value
User_ID	User의 ID
Product_ID	관심상품으로 등록하고자 하는 상품의 Pcode

b. Request(Delete)

Attribute	Value
User_ID	User의 ID
Product_ID	관심상품에서 삭제하고자 하는 상품의 Pcode
FID	관심상품에서 삭제하고자 하는 내역의 Favorites ID

c. Response

Attribute	Value
f_modify_sucess	수정사항 적용 성공
f_modify_fail	수정사항 적용 실패

E. Favorites View Protocol

a. Request

Attribute	Value
User_ID	Favorites를 조회하고자 하는 User의 ID

b. Response

Attribute	Value
f_view_success	Favorites 조회 성공
f_view_fail	Favorites 조회 실패

F. Product Comparing Protocol

a. Request

Attribute	Value
Pcode	국가별 가격을 비교하고자 하는 Product의 pcode

b. Response

Attribute	Value
compare_failure	가격 계산 및 출력 실패
price	가격 계산 및 출력 성공 시의 국가별 가격

G. Exchange Rate Protocol

a. Request

Attribute	Value
cid	환율을 조회하려는 Country의 cid

b. Response

Attribute	Value
exchange_rate_fail	환율 조회 실패
exchange_rate	환율 조회 성공 시 국가별 환율

H. Search Protocol

a. Request

Attribute	Value
Search_word	검색할 단어

b. Response

Attribute	Value
Search_fail	검색 실패
Product_name	Product의 이름
Product_price	Product의 가격
Product_image	Product의 이미지

I. Recommend Protocol

a. Request

Attribute	Value
Product_code	Product의 품번

b. Response

Attribute	Value
Product_code	Recommended Product의 품번 N개

9. Database Design

9.1. Objectives

Database Design은 요구사항 명세서에서 기술한 데이터베이스 요구사항을 바탕으로 작성하였다. 요구사항을 바탕으로 ER Diagram을 작성하고, 이를 이용하여 Relational Schema를 작성하고, 최종적으로 SQL DDL을 작성한다.

9.2. ER Diagram

개체는 분리된 물체 하나를 표현한다. 개체는 사각형으로 표현되며, 관계는 다이아몬드로 표현된다. 개체나 관계는 특성을 가질 수 있으며, 이 특성들은 관계 집합에 실선으로 연결된 타원형으로 표현한다.
관계는 두개 이상의 개체들의 연관 관계를 표현한다. 모든 개체는 고유하게 식별되는 특성 집합을 가지고 있어야 하며, 최소한의 고유 식별 특성 집합은 개체의 기본 키(Primary Key)라 불린다.

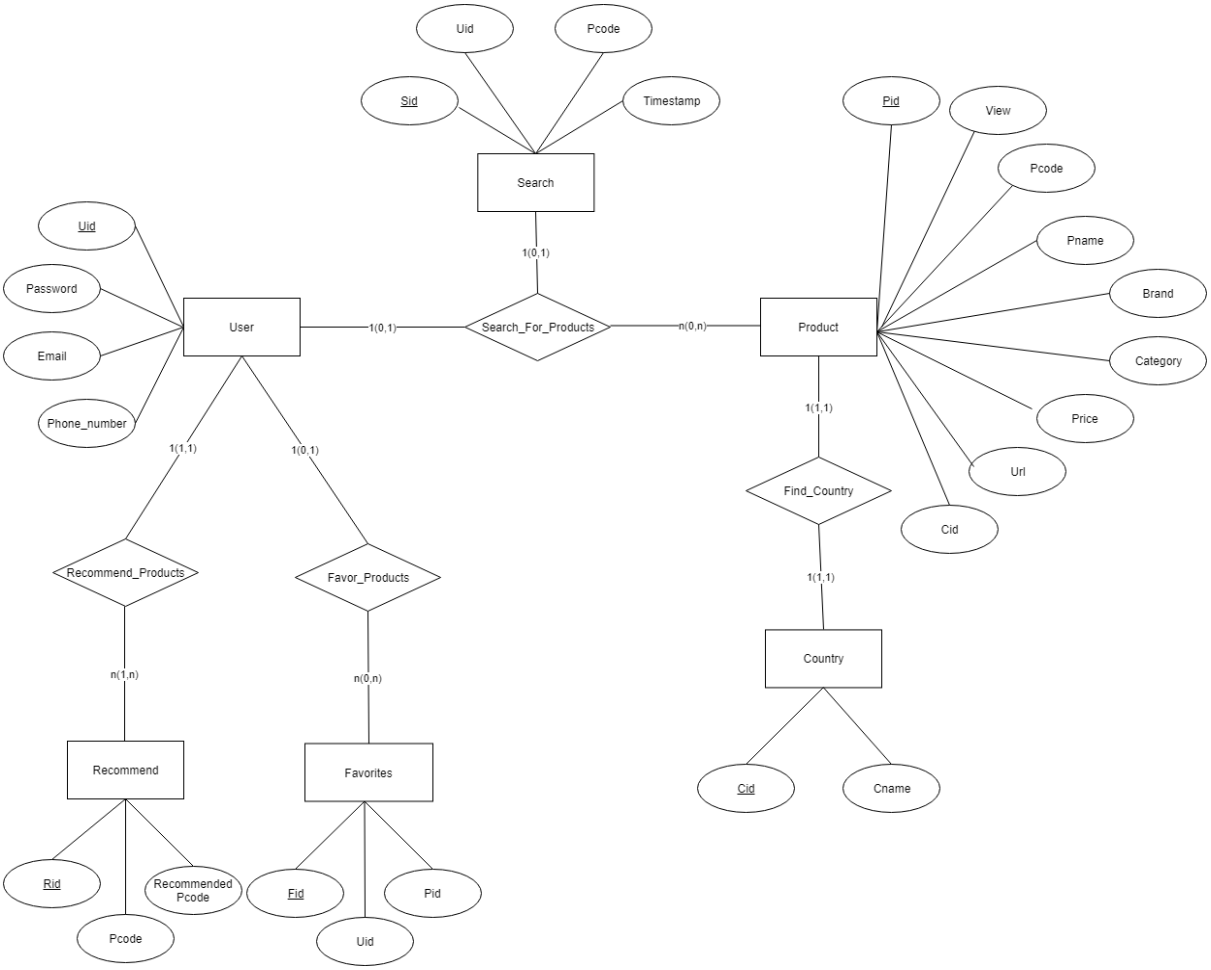

Diagram 19 ER Diagram

A. Entity

A.1. User

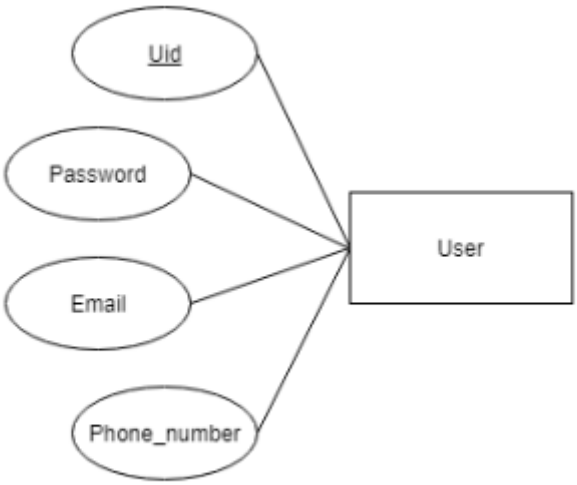

Diagram 20 ER Diagram User Entity

User는 사용자 정보를 나타낸다. Uid, Password, Email, Phone_number의 속성을 가지고 있으며, Primary Key는 Uid이다.

A.2. Product

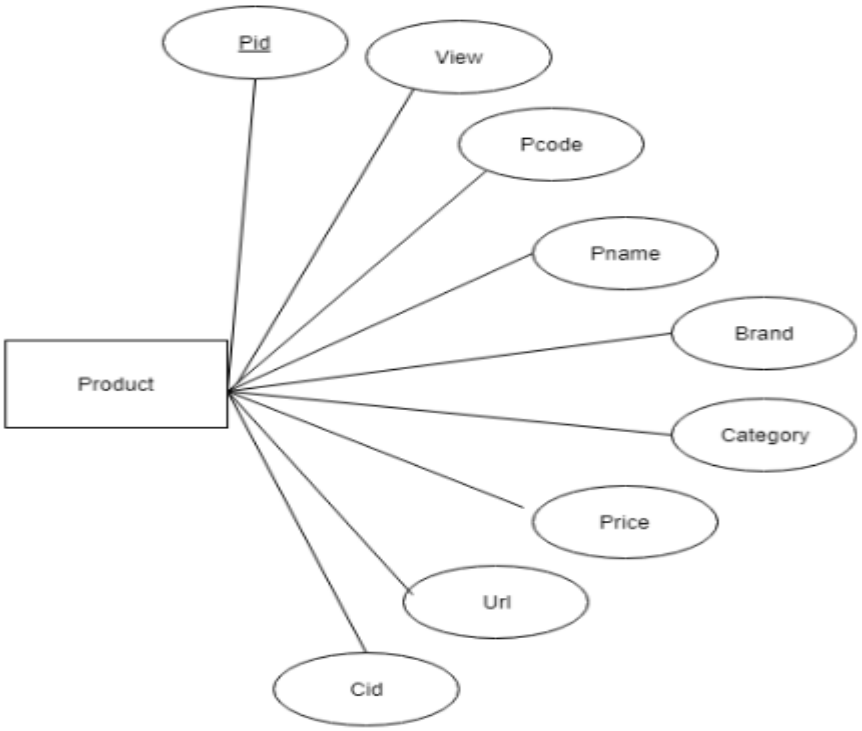

Diagram 21 ER Diagram Product Entity

Product는 상품 정보를 나타낸다. Pid, Pcode, Pname, Brand, Category, Price, Url, Cid의 속성을 가지고 있으며, Primary Key는 Pid이다.

A.3. Country

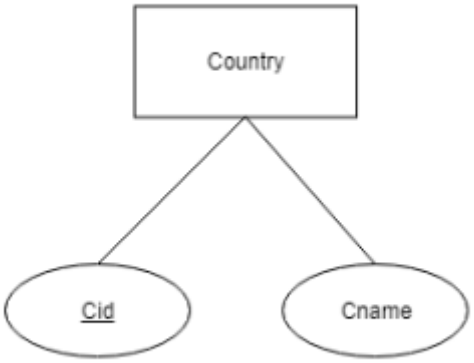

Diagram 22 ER Diagram Country Entity

Country는 나라에 대한 정보를 나타낸다. Cid, Cname의 속성을 가지고 있으며, Primary Key는 Cid이다.

A.4. Favorites

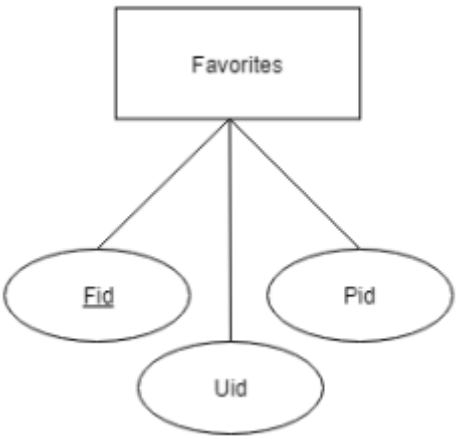

Diagram 23 ER Diagram Favorites Entity

Favorites는 user가 관심상품 추가를 한 상품에 대한 정보를 나타낸다. Fid, Uid, Pid의 속성을 가지고 있으며, Primary Key는 Fid이다.

A.5. Search

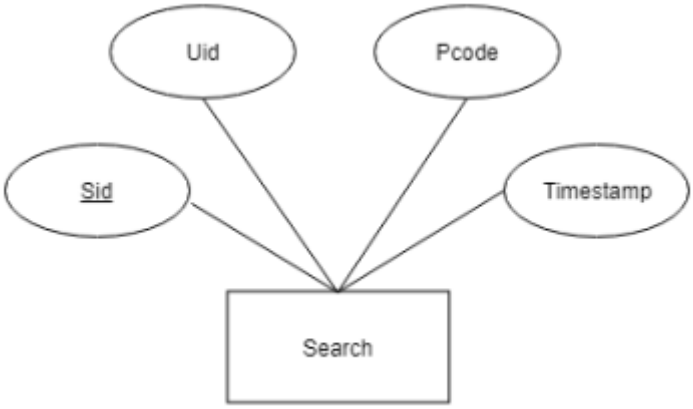

Diagram 24 ER Diagram Search Entity

Search는 user가 상품을 검색하는 기록에 대한 정보를 나타낸다. Sid, Uid, Pid, Timestamp의 속성을 가지고 있으며, Primary Key는 Sid이다.

A.6. Recommend

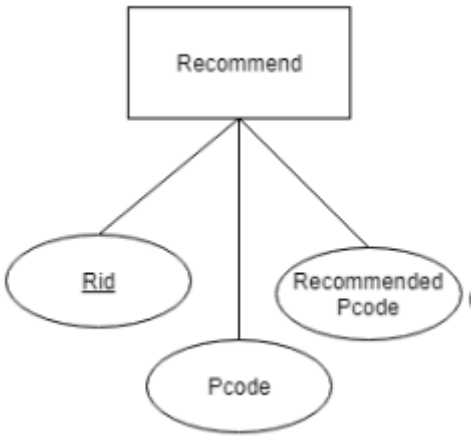

Diagram 25 ER Diagram Recommend Entity

Recommend는 user의 search log를 바탕으로 유사 상품을 추천에 대한 정보를 나타낸다. Rid, Pcode, Recommended Pcode의 속성을 가지고 있으며, Primary Key는 Rid이다.

B. Relationship

B.1. Search Product

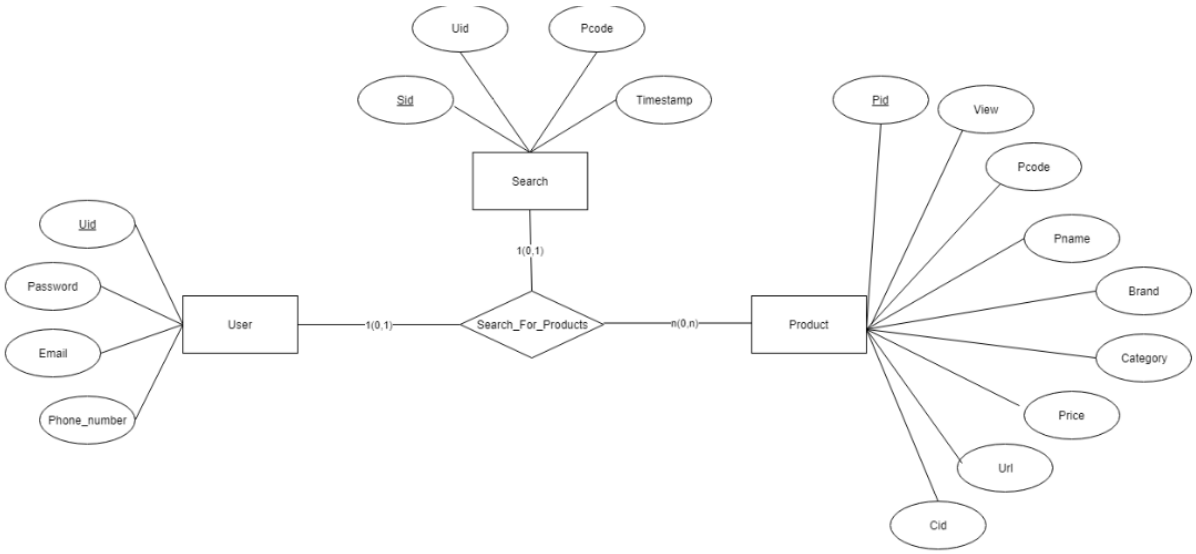

Diagram 26 ER Diagram Search Product Relationship

Search Product Relationship은 user가 상품을 검색하는 관계로, 1명의 사용자가 여러 상품을 검색할 수 있으며 모든 검색 기록은 1명의 user에 의해 이루어진다.

B.2. Recommend Product

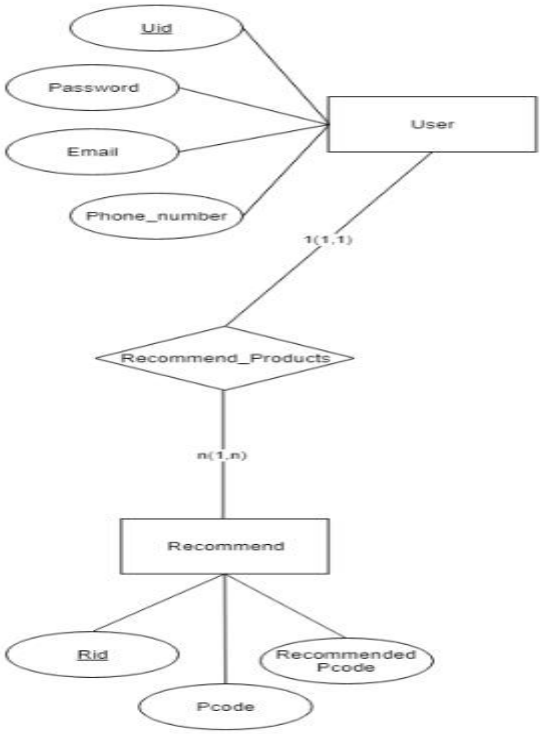

Diagram 27 ER Diagram Recommend Product Relationship

Recommend Product Relationship은 user의 search log를 이용해 그와 유사한 상품을 추천하는 관계로, 1명의 사용자가 여러 상품을 추천받을 수 있으며, 모든 추천 상품은 1명의 user에 의해 이루어진다.

B.3. Favor Product

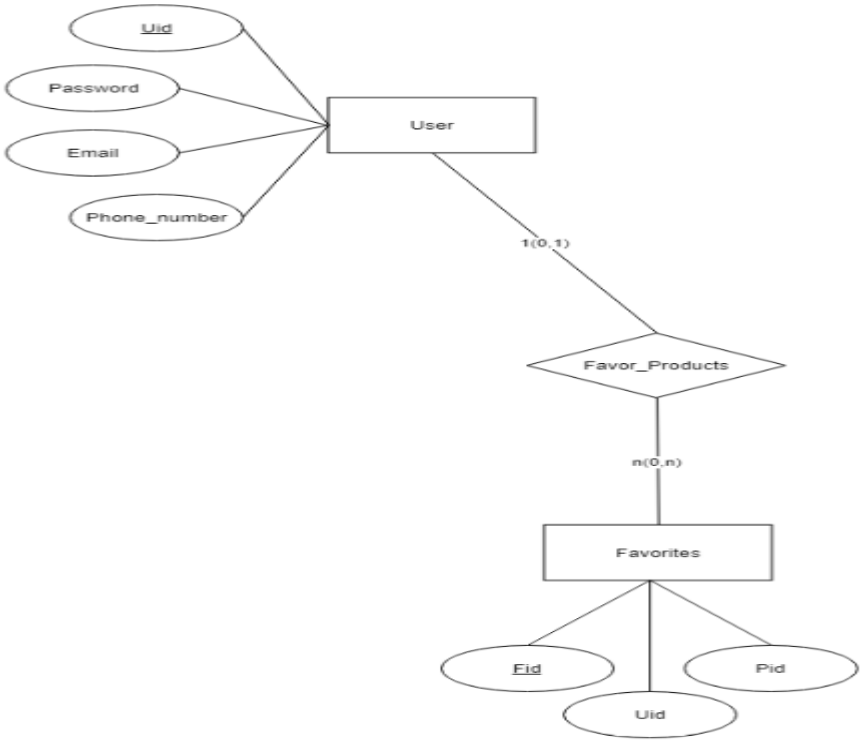

Diagram 28 ER Diagram Favor Product Relationship

Favor Product는 user가 관심 있는 상품을 등록한 관계로, 1명의 사용자가 여러 상품을 관심 상품 추가할 수 있으며, 모든 관심 상품은 1명의 user에 의해 이루어진다.

B.4. Find Country

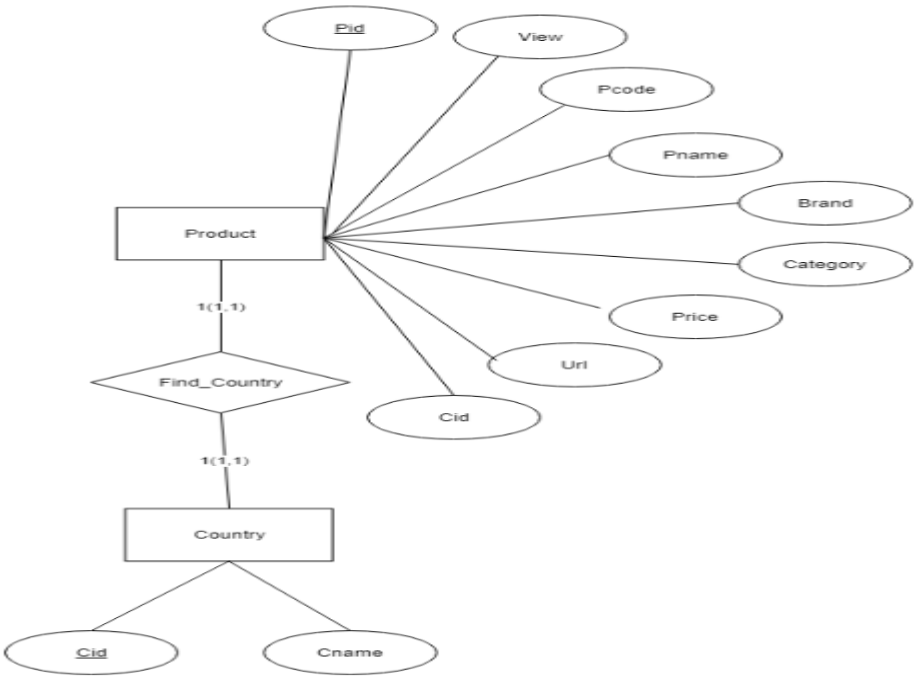

Diagram 29 ER Diagram Find Country Relationship

Find Country는 상품의 국가 이름을 찾는 관계로, 1개의 Country는 1개의 Country Name을 갖는다.

9.3. Relational Schema

A. User

Uid	Password	Email	Phone_Number

Primary Key (PK): Uid

Foreign Key (FK): 없음

FUNCT DEP (FD):

Uid → {Password, Email, Phone_Number}

Description: Entity User에 대한 테이블이다. 모든 속성에 Null값을 허용하지 않는다.

B. Product

Pid	Pcode	View	Pname	Brand	Category	Price	Url	Cid

Primary Key (PK): Pid

Foreign Key (FK): Cid

FUNCT DEP (FD):

Pid → {Pcode, Pname, Brand, Category, Price, Url, Cid, View}

Pcode → {Pid, Pname, Brand, Category, Price, Url, Cid, View}

Description: 상품에 대한 테이블이다. 모든 속성에 Null값을 허용하지 않는다.

C. Country

Cid	Cname

Primary Key (PK): Cid

Foreign Key (FK): 없음

FUNCT DEP (FD):

Cid → {Cname}

Description: 국가에 대한 테이블이다. 모든 속성에 Null값을 허용하지 않는다.

D. Favorites

Fid	Uid	Pid

Primary Key (PK): Fid

Foreign Key (FK): Uid, Pid

FUNCT DEP (FD):

Fid → {Uid, Pid}

Description: user가 관심상품 등록한 상품에 대한 테이블이다. 모든 속성에 Null값을 허용하지 않는다.

E. Search

Sid	Uid	Pid	Timestamp

Primary Key (PK): Sid

Foreign Key (FK): Uid, Pid

FUNCT DEP (FD):

Sid → {Uid, Pid, Timestamp}

Description: user가 검색한 상품에 대한 테이블이다. 모든 속성에 Null값을 허용하지 않는다.

F. Recommend

Rid	Pcode	Recommended Pcode

Primary Key (PK): Rid

Foreign Key (FK): Pcode, Recommended Pcode

FUNCT DEP (FD):

Rid → {Pcode, Recommended Pcode}

Description: 상품에 대하여 유사한 상품에 대한 테이블이다. 모든 속성에 Null값을 허용하지 않는다.

9.4. Normalization

정상화는 각 Relation을 Normal Form(1NF~4NF, BCNF)에 위배되는지 testing 하고 위반하는 경우 Relation들을 분해하는 작업이다. Relation 이 Normal Form에 위배되면 Redundancy, Anomaly 현상이 일어난다.

✓ Redundancy는 같은 정보가 table내에서 불필요하게 중복되는 현상이다.

✓Anomaly는 비정상의 문제로 Insert/Delete/Update anomaly가 있다.

-Insert anomaly : PK가 아닌 attribute를 새로 만들어 추가할 때, PK가 attribute에 아직 값이 없어 NULL로 생성되면 Entity Integrity를 위배하는 문제가 발생한다.

-Delete anomaly : 어떤 attribute의 value를 삭제할 시 해당 값을 가지고 있는 tuple들이 모두 삭제된다. 이는 원하지 않은 정보들까지 삭제될 수 있다는 점에서 문제가 발생한다.

-Update anomaly : 어떤 attribute의 value를 변경할 때 해당 값을 가지고 있는 모든 tuple들의 값을 변경해야 한다. 이는 일일이 해당 값을 다 바꿔주어야 한다는 점에서 비용이 많이 든다는 문제가 있다.

앞서 multi-valued attribute들에 대해 table을 따로 생성하여 중복의 문제를 해결해 주었으므로, X의 값에 대해 Y의 값이 여러 개 있는 MVD는 존재하지 않는다. (MVD : X→ Y) 따라서 이를 정규화하는 Normal Form인 4NF는 고려하지 않는다.
BCNF를 만족하는 모든 중복이 소거되므로 본 장에서는 BCNF를 중심으로 정규화한다.

9.5. SQL DDL

A. User

```
CREATE TABLE 'User'(
   'Uid' VARCHAR(45) NOT NULL,
   'Email' VARCHAR(45) NOT NULL,
   'Password' VARCHAR(45) NOT NULL,
   'Phone_num' VARCHAR(45) NOT NULL,
   PRIMARY KEY ('Uid'),
)ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Uid는 PRIMARY KEY 이므로 NOT NULL이다.

B. Favorites

```
CREATE TABLE 'Favorites' (
        'Fid' VARCHAR(45) NOT NULL,
        'Pid' VARCHAR(45) NOT NULL,
        'Uid' VARCHAR(45) NOT NULL,
        PRIMARY KEY ('Fid'),
        FOREIGN KEY ('Uid') REFERENCES 'User' ('Uid') ON UPDATE CASCADE ON
DELETE CASCADE,
        FOREIGN KEY ('Pid') REFERENCES 'Product'('Pid')  ON UPDATE CASCADE ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Fid는 PRIMARY KEY 이므로 NOT NULL이다.

C. Product

```
CREATE TABLE 'Product'(
  'Pid' VARCHAR(45) NOT NULL,
  'Pcode' VARCHAR(45) NOT NULL,
  'Brand' VARCHAR(45) NOT NULL,
  'Pname' VARCHAR(45) NOT NULL,
  'Category' VARCHAR(45) NOT NULL,
  'Price' INT(11) DEFAULT NULL,
  'URL' VARCHAR(45) NOT NULL,
  'Cid' VARCHAR(45) NOT NULL,
  'View' INT(11) DEFAULT NULL,
  PRIMARY KEY ('Pid'),
  FOREIGNE KEY ('Cid') REFERENCES 'Country'('Cid') ON UPDATE CASCADE ON DELETE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Pid는 PRIMARY KEY 이므로 NOT NULL이다.

D. Country

```
CREATE TABLE 'Country'(
  'Cid' VARCHAR(45) NOT NULL,
  'Cname' VARCHAR(45) NOT NULL,
  PRIMARY KEY ('Cid')
)ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Cid는 PRIMARY KEY 이므로 NOT NULL이다.

E. Search

```
CREATE TABLE 'Search''(
  'Sid' VARCHAR(45) NOT NULL AUTO_INCREMENT,
  'Uid' VARCHAR(45) NOT NULL,
  'Pid' VARCHAR(45) NOT NULL,
  'S1' VARCHAR(45) DEFAULT NULL,
  'S2' VARCHAR(45) DEFAULT NULL,
  'S3' VARCHAR(45) DEFAULT NULL,
  'S4' VARCHAR(45) DEFAULT NULL,
  'S5' VARCHAR(45) DEFAULT NULL,
  'timestamp' INT(2) DEFAULT NULL,
  PRIMARY KEY ('Sid')
  FOREIGN KEY ('Uid') REFERENCES 'User' ('Uid') ON UPDATE CASCADE ON DELETE
CASCADE
  FOREIGN KEY ('Pid') REFERENCES 'Product' ('Pid) ON UPDATE CASCADE ON DELETE
CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Sid는 PRIMARY KEY 이므로 NOT NULL이다.

F. Recommend

```
CREATE TABLE ‘Recommend’ (
    ‘RID’ VARCHAR(45) NOT NULL
    ‘Pcode’ VARCHAR(45) NOT NULL
    ‘RecommendedPcode’ VARCHAR(45)
    PRIMARY KEY(‘RID’)
    FOREIGN KEY(‘Pcode’) REFERENCES ‘PRODUCT(‘Pid’) ON UPDATE, CASCADE ON DELETE CASCADE
    FOREIGN KEY(‘RecommendedPcode’) REFERENCES ‘PRODUCT(‘Pid’) ON UPDATE, CASCADE ON DELETE CASCADE
)ENGINE=InnoDB DEFAULT CHARSET=utf8
```

RID는 PRIMARY KEY이므로 NOT NULL이다.

10. Testing Plan

10.1. Objectives

Testing Plan에서는 테스트 정책과 테스트 케이스에 대해 설명한다. 본 장의 목적은 전체 시스템이 의도한대로 실행되는지를 검증하기 위한 과정을 계획하는 데에 있다.

Testing Policy에서는 testing에 대한 단계적 접근을 설명한다. Test Case에서는 각 Sub-System에서의 예시 케이스를 토대로 사용자를 판별하고 그에 따라 시스템에 기대하는 입,출력 동작을 서술한다.

10.2. Testing Policy

직구변화구의 시스템 개발에서는 크게 세 단계로 나누어 Testing을 진행한다. 이는 Development Testing, Release Testing, User Testing으로 나뉜다. Development Testing은 다시 Unit Testing, Component Testing, System Testing으로 나뉜다.

A. Development Testing

Unit Testing은 각 프로그램 단위나 모듈, 클래스들을 Testing하는 것이다. 이는 각 Unit을 개발한 개발자들이 책임지고 Testing한다. 원래의 기능에 적합하게 동작하는지를 확인하여, 검증된 것만 Commit 하도록 한다.

Component Testing 또한 각 Component을 개발한 개발자들이 책임지고 Testing한다. 각각의 Unit을 모아 복합 Component로 만들며, 이 때 주어진 protocol 혹은 Interface 등에 어긋나지 않는지를 확인한다.

System Testing은 각 Sub-system 단위로 진행한다. 각 Sub-system 요소들을 Incrementally Integrating 하여 Testing을 진행한다. 이러한 각 요소들이 합쳐진 이후에 전체 시스템에 대하여 최종 Testing을 진행한다.

B. Release Testing

Release Testing은 해당 시스템이 고객의 요구에 맞게 개발되었는지를 확인하는 Testing이다. 이를 위해서 요구사항 명세서에 서술된 각각의 요구사항과 비교하여

Testing을 진행하도록 한다. 인원이 부족한 관계로 별도의 QA(Quality Assurance) 팀을 생성하여 진행하지는 않으며, Development Testing이 종료된 이후에 개발 팀에서 진행한다.

C. User Testing

Release Testing이 완료된 후, User Testing을 진행한다. 사용자들이 직접 참여하여 사용자 환경에서 시스템을 Testing하는 것이다. User Testing에는 Alpha Testing, Beta Testing, Acceptance Testing이 있다. 전체 과정에서의 효과적인 Testing을 위해서는 실제 직구 구매자를 대상으로 하는 것이 실제 Domain User들의 피드백을 획득할 수 있을 것이다. 이를 위해 대학생 직구 구매자들을 대상으로 Testing에 대한 홍보를 진행하여 user를 모아 Testing을 실시한다.

10.3. Test Case

A. User Management System

A.1. When a User tries to Sign up

User: '회원가입' 버튼을 누른다.
System: User ID를 입력하는 창을 띄운다
User: User ID를 입력한다.
System: 중복검사를 진행한다.

(1) 중복되는 경우

System: "해당 User ID가 이미 존재합니다. 다른 ID를 입력해주세요" 출력

System: 회원가입 정보를 입력하는 창을 띄운다.
User: Password, Password재입력, Email, Phone을 입력한 뒤, 가입 버튼을 누른다.
System: Password가 동일하게 입력되었는지 확인

(2) 비밀번호 불일치 경우

System: "비밀번호를 동일하게 입력해주세요" 출력

System: User의 입력 내용을 User DB에 저장한다.

A.2. When a User tries to Log in

User: ID와 password를 입력한다.
. User: '로그인'버튼을 누른다.
System: User DB상의 ID와 password가 입력한 내용과 일치하는 지 확인한다.

(1) 불일치 하는 경우

System: "입력한 ID와 password가 일치하지 않습니다." 출력

System: User 로그인 상태로 메인화면으로 이동한다.

A.3. When a User tries to add product in Favorites

User: 상품페이지에서 add Favorites버튼을 누른다.
System: 해당 상품을 Favorites DB에 저장한다.

(1) 해당 상품이 이미 Favorites에 있는 경우
System: add Favorites버튼이 del Favorites의 기능을 한다.

System: "성공적으로 추가되었습니다." 출력한다.

A.4. When a User tries to view product in Favorites

User: Favorites 보기 버튼을 누른다.
System: Favorites DB에서 User_ID에 해당하는 모든 상품코드을 가져온다.
System: Favorites 페이지로 이동한다.
System: Product DB에서 상품코드와 일치하는 상품을 검색한다.
System: DB에서 가져온 Product를 출력한다.

A.5. When a User tries to delete product in Favorites

User: Favorites 페이지 상에서 제거하고자 하는 상품의 삭제 버튼을 누른다.
System: Favorites 항목의 FID를 전달받아 Favorites DB에서 행을 삭제한다.
System: Favorites 페이지를 새로 불러온다.

B. Comparing System

B.1. When a User Clicks a Product

User: 상품을 클릭한다

System: Product DB에서 해당 상품의 데이터를 가져온다.

System: 해당 국가의 환율을 조회하여 원화 가격으로 계산한다.

(1) Exchange Rate API가 작동하지 않을 시

System: 가장 최근에 읽어온 환율 정보를 사용한다.

System: 해당 상품 데이터와 변환된 가격을 화면에 노출한다.

C. Recommend System

C.1. When a User Clicks a Product

User: Product을 클릭한다.

System: 해당 Product의 상품 상세 페이지로 이동한다.

System: Recommend DB로부터 해당 Product와 유사한 Product 4개를 읽어온다.

System: 상품 상세 페이지의 하단에 노출한다.

D. Search System

D.1. When a User tries to Search a product

User: 상품에 대한 검색어를 입력한다.

System: 입력 받은 검색어를 Product DB가 가진 attribute 중 name과 비교하여 일치하는 품목을 가져온다.

System: 해당 품목을 노출한다.

D.2. Collecting Search Log

User: 상품에 대한 검색어를 입력한다.

System: 해당 User가 검색한 내역을 Search DB에 저장한다.

11. Development Environment

11.1. Objectives

Development Environment에서는 개발자의 환경에 대해 설명한다. 사용한 프로그래밍 언어와 통합개발환경(IDE)에 대해 서술한다.

11.2. Programming Language & IDE

A. Programming Language

직변은 상품에 대한 여러 국가의 정보를 한번에 모아 보여주고 검색 사항을 분석해 추천 상품을 보여주는 딥러닝 기반 웹 서비스이다. 따라서 동적인 웹 사이트 개발을 위한 프론트엔드 프레임워크로 장고와 부트스트랩을, 데이터베이스 구축을 위해 SQLite가 활용된다.

또한 딥러닝 기반 추천서비스를 위해 구글의 딥러닝, 인공지능 프레임워크 keras와 vgg16 model을 사용한다.

Figure 6 Development Tools

B. IDE

GitHub의 Atom을 통합개발환경(IDE)로 사용한다. Atom은 C, C++, JAVA Script, Python, HTML, 등 다양한 언어의 개발 환경을 제공한다.

Figure 7 Atom

11.3. Coding Rule

직변 시스템에서 사용한 Coding Rule은 다음과 같다

A. 서버에 바로 작업하는 경우 버전 관리가 어렵기 때문에 Github를 이용하여 코드를 관리한다.
B. Github를 이용하는 경우, 여러 개발자가 동시에 같은 파일을 작업하는 경우가 없도록 한다.

11.4. Version Management Tool

효율적 코드 관리 및 공유를 위해 Github를 사용한다. GitHub는 분산 버전 관리 툴인 Git을 사용하는 프로젝트를 지원하는 웹호스팅 서비스로, 세계적으로 사용되는 Git 호스팅 사이트이다. GitHub는 영리 서비스와 동시에 오픈소스를 위한 무상 서비스를 모두 제공한다.

Figure 8 Github Octocat

12. Development Plan

12.1. Objectives

Development plan에서는 개발 계획에 대해 서술한다. 이때, Gantt Chart를 이용하여 개발 계획과 실제 개발 흐름에 대해 서술하고자 한다.

12.2. Gantt Chart

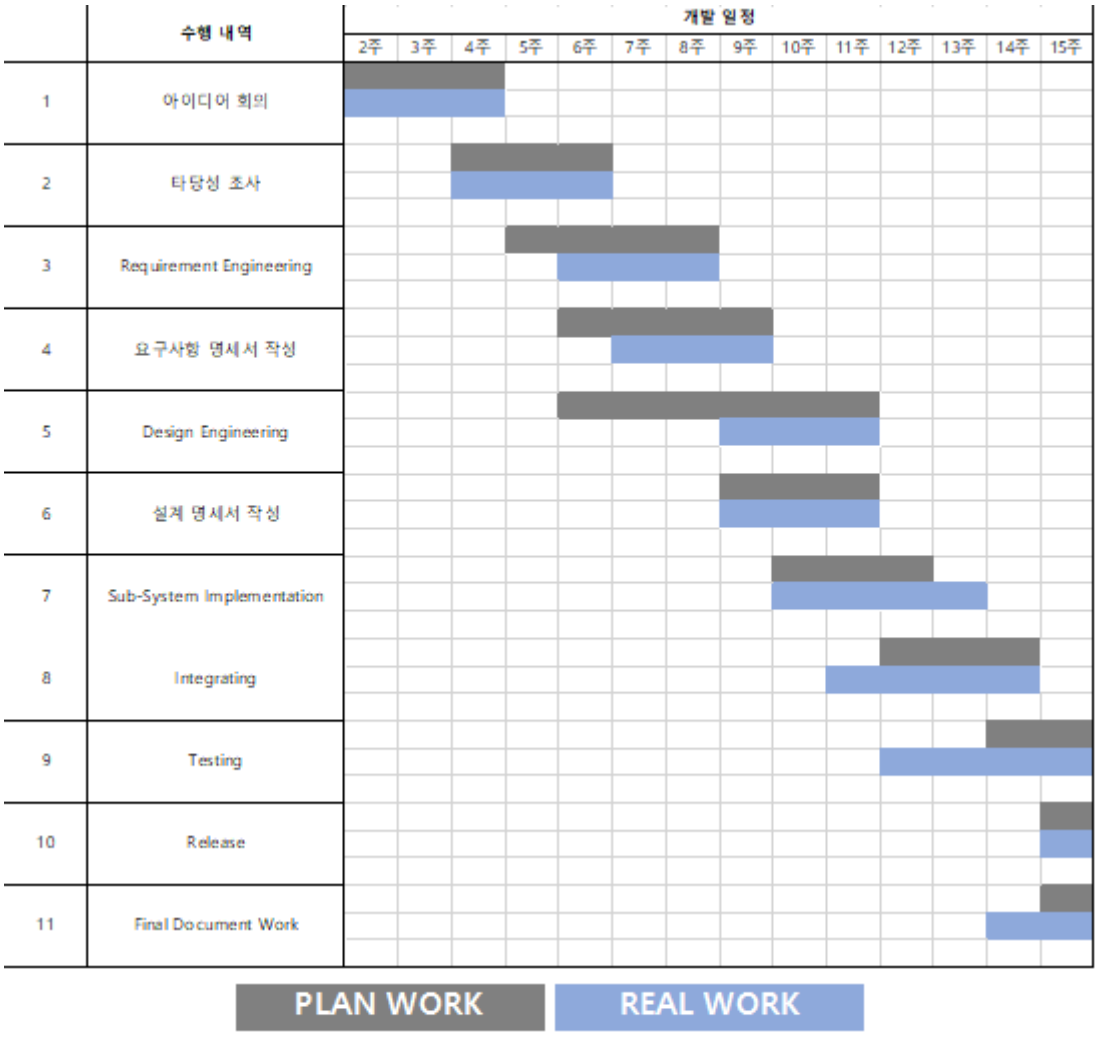

Table 2 Gantt Chart

12.3. Domain

AWS를 이용하여 서버를 구축한 뒤, 사용자의 편리하고 빠른 접속과 관리자의 편리한 관리를 위해 도메인을 설정할 예정이다. 도메인은 가비아(Gabia)를 통해 받고, 이름은 'jikbyeon'로 정하여 구입할 예정이다.

13. Index

13.1. Table

13.2. Figure

13.3 Diagram