Software Design Specification

Sungkyunkwan University

Software Engineering 2020 Spring

Team #3

2016310978 강승룡

2017313172 고성현

2017313114 김지우

2017314749 유민종

2014312692 이병건

2014313003 최현우

Contents

1. Preface

이 단원은 본 문서가 예상하는 독자들, 문서의 구조, 그리고 각 단원에 대해 설명한다. 그리고 문서의 버전과 각 버전에서 만들어진 변경 사항에 대해 요약한다.

1.1. Expected readership

본 문서는 다음과 같은 독자들에게 제공될 것을 기대하고 만들어졌다.

주 예상 독자는 본 시스템의 개발자(Developer)들이다. 시스템 개발자는 크게 Front-end와 Back-end 개발자로 나누어진다.

주 예상 독자는 본 시스템을 유지·보수하는 개발자들이다.

전체 예상 독자는 본 웹 어플리케이션(Web application)의 개발과 유지·보수에 참가하는 모든 개발자들이며, 그 외에 관련된 stakeholders이다.

1.2. Document structure

본 문서는 Preface, Introduction, Overall system architecture, Subsystem architecture – Front-end, Subsystem architecture – Back-end, Protocol design, Database design, Testing plan, Development plan, Index의 10개의 단원으로 이루어졌다. 각 단원에 대한 설명은 다음과 같다.

A. Preface

이 단원은 본 문서가 예상하는 독자들, 문서의 구조, 그리고 각 단원에 대해 설명한다. 그리고 문서의 버전과 각 버전에서 만들어진 변경 사항에 대해 요약한다.

B. Introduction

이 단원은 제안한 시스템의 설계에 사용된 다이어그램, 툴에 대해 설명하고 개발 범위에 대해 설명한다.

C. Overall system architecture

이 단원은 본 시스템의 전체 구조에 대해 설명하고 각 시스템 간 연관 관계를 나타낸다. 그리고 간략하게 각 시스템을 설명한다.

D. Subsystem architecture – Front-end

이 단원은 Front-end 시스템의 구조와 subcomponent 간의 구성도 및 각 요소의 역할 등을 여러 종류의 다이어그램을 통해 설명한다.

E. Subsystem architecture – Back-end

이 단원은 Back-end 시스템의 구조와 subcomponent 간의 구성도 및 각 요소의 역할 등을 여러 종류의 다이어그램을 통해 설명한다.

F. Protocol design

이 단원에서는 본 웹 어플리케이션의 여러 시스템들 간 communication을 위해 필요한 여러 가지 프로토콜(protocol)들을 기술한다. 또한, 프로콜을 사용하기 위한 데이터 포맷(data format)을 설명한다.

G. Database design

이 단원에서는 실질적인 데이터베이스 설계를 기술한다. ER diagram을 통해 개체 간 관계와 개체 속성에 대해 설명하고, Relational schema와 데이터 정의와 접근 양식을 기술한다.

H. Testing plan

이 단원에서는 본 웹 어플리케이션의 테스트에 있어 사용되는 테스트 방법 및 전체 프레임워크(Framework)에 대해 설명한다.

I. Development plan

이 단원에서는 본 시스템 개발에 사용되는 프로그래밍 언어(programming language), IDE(Integrated development environment), 프로젝트 관리 툴(project management tool)에 대해 설명한다. 또한, 시스템 개발 일정을 설명한다.

J. Index

이 단원은 그림 인덱스, 표 인덱스, 다이어그램 인덱스를 포함한다.

1.3. Version history

Version	Modified date	Explanation
0.1	2020.05.21	본 문서의 10가지 단원에 대해 목차 작성
1.0	2020.05.23	Preface, Introduction, Overall system architecture, Subsystem architecture – Front-end, Subsystem architecture – Back-end, Protocol design, Database design, Testing plan 작성
2.0	2020.05.24	Development plan 작성, Sub-system architecture - Back-end 내용 수정
3.0	2020.05.24	문서 보완
3.1	2020.05.24	Front-end view state diagram 추가

Table 1. Document Version history

2. Introduction

이번 챕터에서는 제안한 시스템의 설계에 사용된 다이어그램, 툴에 대해 설명하고 개발 범위에 대해 설명한다.

2.1. Applied Diagram

A. Use case Diagram

유스 케이스 다이어그램은 사용자, 그리고 사용자가 수반한 다른 유스 케이스 간의 관계를 보여주는 사용자-시스템 간 상호작용의 표현이다. 유스 케이스 다이어그램은 각기 다른 종류의 시스템 사용자와 각기 다른 유스 케이스를 식별할 수 있으며 다른 유형의 다이어그램이 수반되기도 한다. 유스 케이스는 원이나 타원으로 표현된다.

B. Class Diagram

클래스 다이어그램은 UML의 구조 다이어그램으로써 클래스, 내부 구성 요소 및 클래스 간의 관계를 도식화하여 시스템의 특정 모듈이나 일부 또는 전체 구조를 나타낸다.

클래스 다이어그램을 통해 문제 도메인의 구조를 나타낼 수 있고, 실제 소프트웨어의 설계 혹은 구현을 위한 용도로 사용되기도 한다. 사용 목적에 따라 소스코드와의 관계, 형식 등이 달라질 수 있다.

C. Sequence Diagram

시퀀스 다이어그램은 시간 순서로 정렬된 객체 상호작용을 보여준다. 시나리오 기능을 수행하는데 필수적인 객체들 간에 교환되는 일련의 메세지들과 시나리오에 수반되는 객체와 클래스를 표현한다.

D. ER Diagram

ER 다이어그램은 Entity-relationship model을 표현하는 것으로 개체, 속성, 관계성을 표현하여 데이터베이스를 모델링하는 방식이다. 주로 데이터베이스를 설계할 때 사용하는 다이어그램이다.

E. State Diagram

스테이트 다이어그램은 사건이나 시간에 따라 시스템 객체 상태 변화를 표현한 그램으로 단일 객체의 상태를 나타낸다. 시스템의 변화를 잡아내기 위하여 사용하며 분석가, 설계자, 개발자 들이 시스템 내의 객체 행동을 이해하는데 도움을 준다. 클래스 다이어그램이 정적인 모습을 보여주는데 반해, 상태 다이어그램은 각 객체의 동적인 상황을 보여준다. 이를 통해 시스템 내의 객체의 행동을 이해하는데 도움을 준다.

2.2. Used Tools

A. Draw.io

Figure 1. Draw.io

draw.io는 플로우 차트, 프로세스 다이어그램을 그리기 위한 온라인 드로잉 도구로써 여러가지 도형과 관계 표시자들을 제공한다. 본 문서에서 다이어그램의 대부분이 이것을 사용하여 제작되었다.

B. Powerpoint

Figure 2. Powerpoint

Powerpoint는 발표용 자료 제작 프로그램으로 기본적인 도형, 아이콘 등 그림을 그리는데 필요한 다양한 도구를 제공한다.

2.3. Project scope

본 시스템은 사용자가 상품의 가격 변동을 확인하고 상품의 가격에 변동을 주는 정보를 직접 찾고 확인해야 했던 기존의 서비스를 개선하기 위해 적절한 뉴스를 제공함으로써 사용자의 구매 선택을 도와주는 시스템이다. 본 시스템은 상품의 가격 변동의 열람, 가격 변동과 관련된 뉴스 제공을 핵심으로 하고 있으며 해당 기능을 위한 서브 시스템으로 구성되어 있다.

Front-end system은 시스템과 사용자와의 상호작용을 담당하며 사용자의 요청을 Back-end system에 전달하고, Back-end system의 응답과 데이터베이스의 정보를 사용자에게 전달하는 역할을 담당하고 있다.

Back-end system은 본 시스템이 제공하고자 하는 핵심 기능을 위한 서브 시스템으로 구성되어 있다. 정기적으로, 혹은 사용자의 요청이 발생할 경우 Crawling system이 상품의 관련 뉴스와 가격을 가져오면 News analysis system이 관련 뉴스를 선별하여 데이터베이스에 전달한다.

3. Overall system architecture

이 단원은 본 시스템의 전체 구조에 대해 설명하고 각 시스템 간 연관 관계를 나타낸다. 그리고 간략하게 각 시스템을 설명한다.

3.1. System Organization

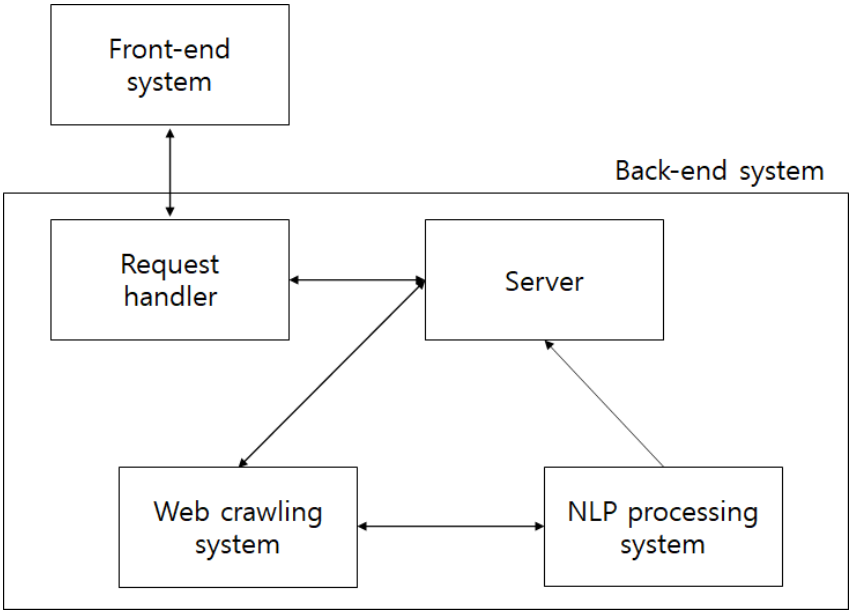

Diagram 1. Overall System Organization

본 시스템은 웹 어플리케이션(Web application)이다. 따라서 client-server 모델이다. Front-end system은 사용자와 상호작용을 하기 위한 시스템이다. Back-end system은 정보 전송을 하기 위한 시스템이다. 그리고 Web crawling system은 웹 상에서 뉴스 정보를 수집하기 위한 시스템이다. NLP processing system은 수집된 정보를 유의미하게 분류하는 시스템이다.

Back-end system은 정해진 시간 간격마다 Web crawling system과 NLP processing system을 호출한다. Web crawling system은 네이버 등의 뉴스 플랫폼에서 본 웹 어플리케이션에서 제공하는 상품의 뉴스 정보를 받아온다. 이렇게 받아온 뉴스 정보를 NLP

processing system에서 여러 가지 모델들을 사용해 가격, 신 제품, 프로모션, 업계 동향의 4가지 카테고리로 구분한다. 그렇게 구분된 정보와 뉴스 정보를 데이터베이스에 저장한다. 데이터베이스에 저장된 정보를 통해 Front-end system에서 사용자에게 서비스를 제공하게 된다. Front-end에서 제공하는 시스템에는 Main page, Product page, API page, My page, Login page가 있다.

A. Front-end system

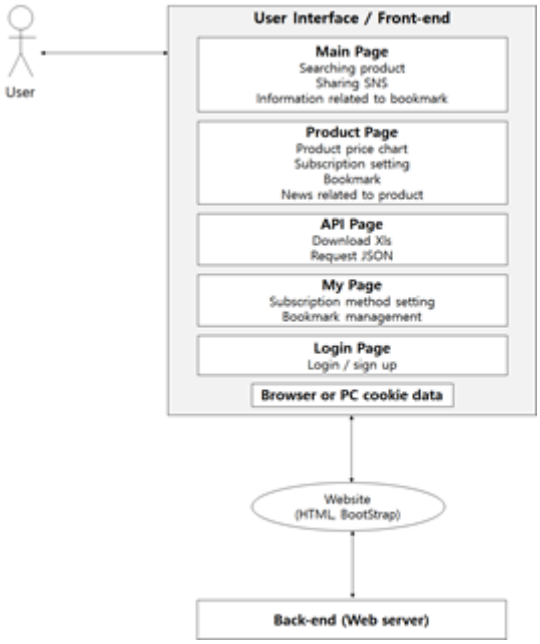

Diagram 2. Front-end Organization

Front-end system은 사용자와 상호작용하기 위해 제공되는 시스템이다. Main page, Product page, API page, My page, Login page를 제공한다. Main page는 사용자가 본 웹 어플리케이션(Web application)에 접속하면 가장 먼저 보이는 페이지이다. Product page는 사용자가 상품의 검색을 통해 얻을 수 있는 페이지로 상품의 가격 차트, 북마크, 관련된 뉴스 등의 정보를 제공한다. API page는 사용자가 본 웹 어플리케이션에서 얻을 수 있는 정보를 재가공할 수 있도록 제공하는 page이다. My page는 사용자가 알람 설정 및 북마킹 관리를 편리하게 할 수 있도록 하는 page이다. Log in page는 사용자가 본 웹 어플리케이션의 맞춤 서비스를 사용할 수 있게끔 하는 로그인과 회원가입 서비스를 제공하는 page이다. 필요에 따라 브라우저에서 Cookie를 사용할 수 있다.

B. Back-end system

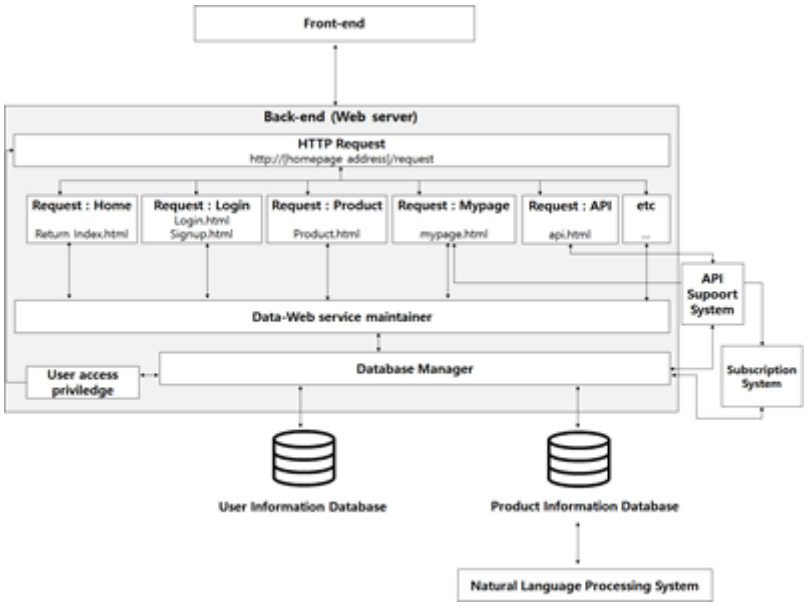

Diagram 3. Back-end Organization

Back-end system은 정보 전송을 하기 위한 시스템이다. 또한, 일정 시간 간격마다 자동으로 호출되어 데이터베이스를 업데이트 한다. 다시 말해, Front-end에서 사용자에게 제공될 정보들을 가공 및 관리하는 system이다. 데이터베이스는 Django의 캡슐화된 함수를 사용하여 처리할 수 있다. Data-Web service maintainer는 HTTP request를 다룬다. 즉, HTTP request 종류에 맞추어 해당하는 html 파일, css 파일 등을 선택하고, 그 외 필요한 내용을 채운다. NLP processing system과 Web crawling system과 연관되어 있다.

C. Web crawling system

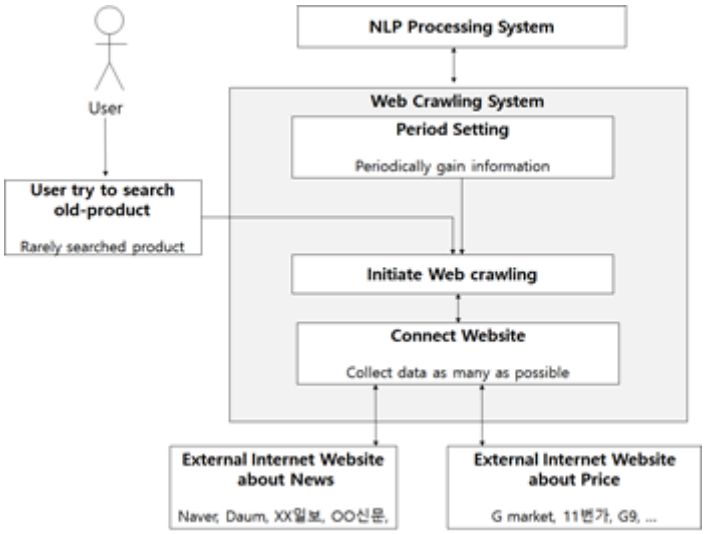

Diagram 4. Web Crawling System

Web crawling system은 Back-end system이 주기적으로 호출될 때 같이 호출되어 웹 상에 있는 뉴스 정보와 상품 가격 정보를 크롤링하는 시스템이다. 뉴스 정보는 네이버 뉴스 등의 플랫폼에서 가져오고, 상품 가격 정보는 G market 등의 플랫폼에서 가져온다. NLP processing system과 연관되어 있다.

D. NLP system

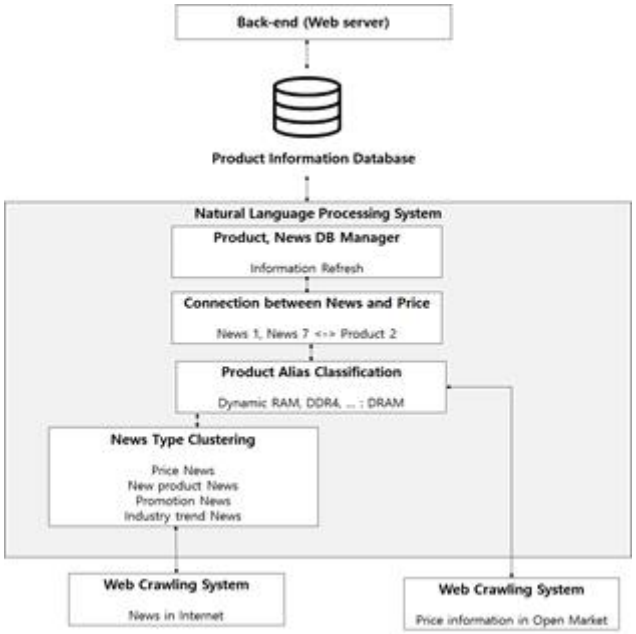

Diagram 5. NLP system

NLP system은 수집된 뉴스 정보를 분류하는 시스템이다. 분류는 가격에 영향을 주는 뉴스, 신 제품을 홍보하는 뉴스, 프로모션을 광고하는 뉴스, 업계 동향을 이야기하는 뉴스로 구분된다. NLP system은 여러 가지 NLP 모델을 사용하여 개발된다. TextRank, LSTM, KoBERT가 모델의 예시이다. TextRank는 'Textrank: Bringing order into text. In Proceedings of the 2004 conference on empirical methods in natural language processing (Mihalcea, R., & Tarau, P., 2004)에서 제시한 모델 구조를 따 왔다. LSTM은 pytorch 라이브러리에서 공식적으로 구현된 모델을 사용한다. KoBERT는 STKBrain 팀에서 github 상에서 제공하는 프로젝트를 사용한다. KoBERT는 Apache 2.0 라이선스를 가진다.

4. Subsystem architecture - Front-end

Front-end의 architecture, subcomponent간의 구성도 및 각 요소의 역할 등을 나열한다. 여러 종류의 diagram이 이 부분을 기술하는 데 사용 될 것이다.

4.1. Subcomponents of front-end

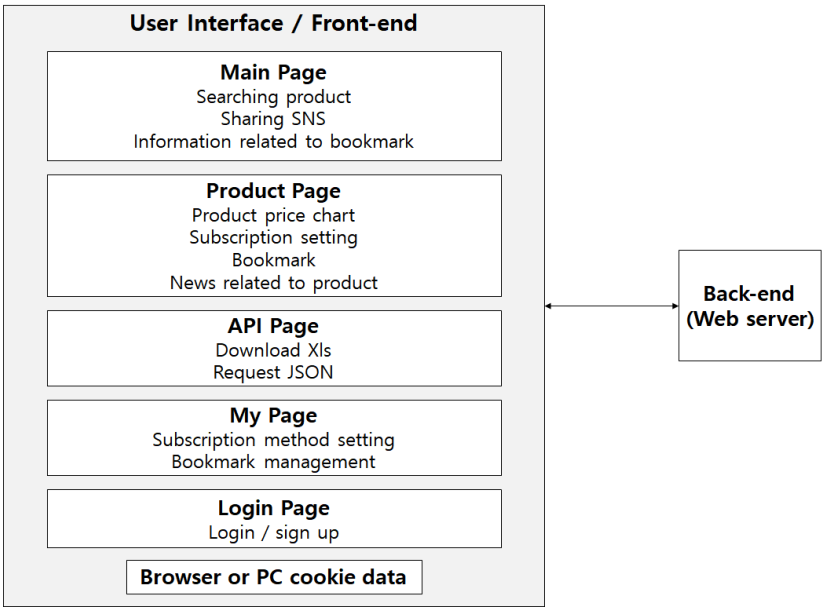

Diagram 6. Front-end Subcomponents

Front-end의 subcomponent 구성도는 위와 같다.

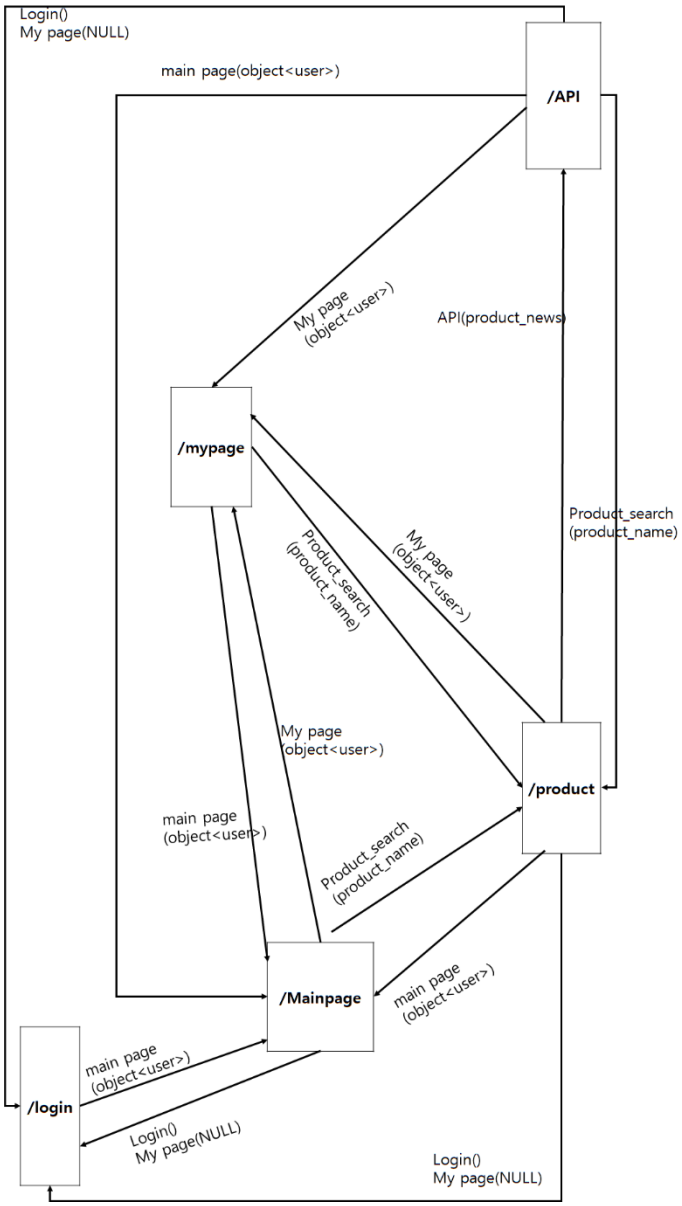

Diagram 7. Front-end state diagram

위 그림 front-end state diagram은 front-end의 홈페이지 객체간에 state diagram을 나타낸 것이다.

front-end의 전체적인 class diagram은 다음과 같다. 유저가 상호작용할 수 있는 가시적인 부분을 객체로 인식한다.

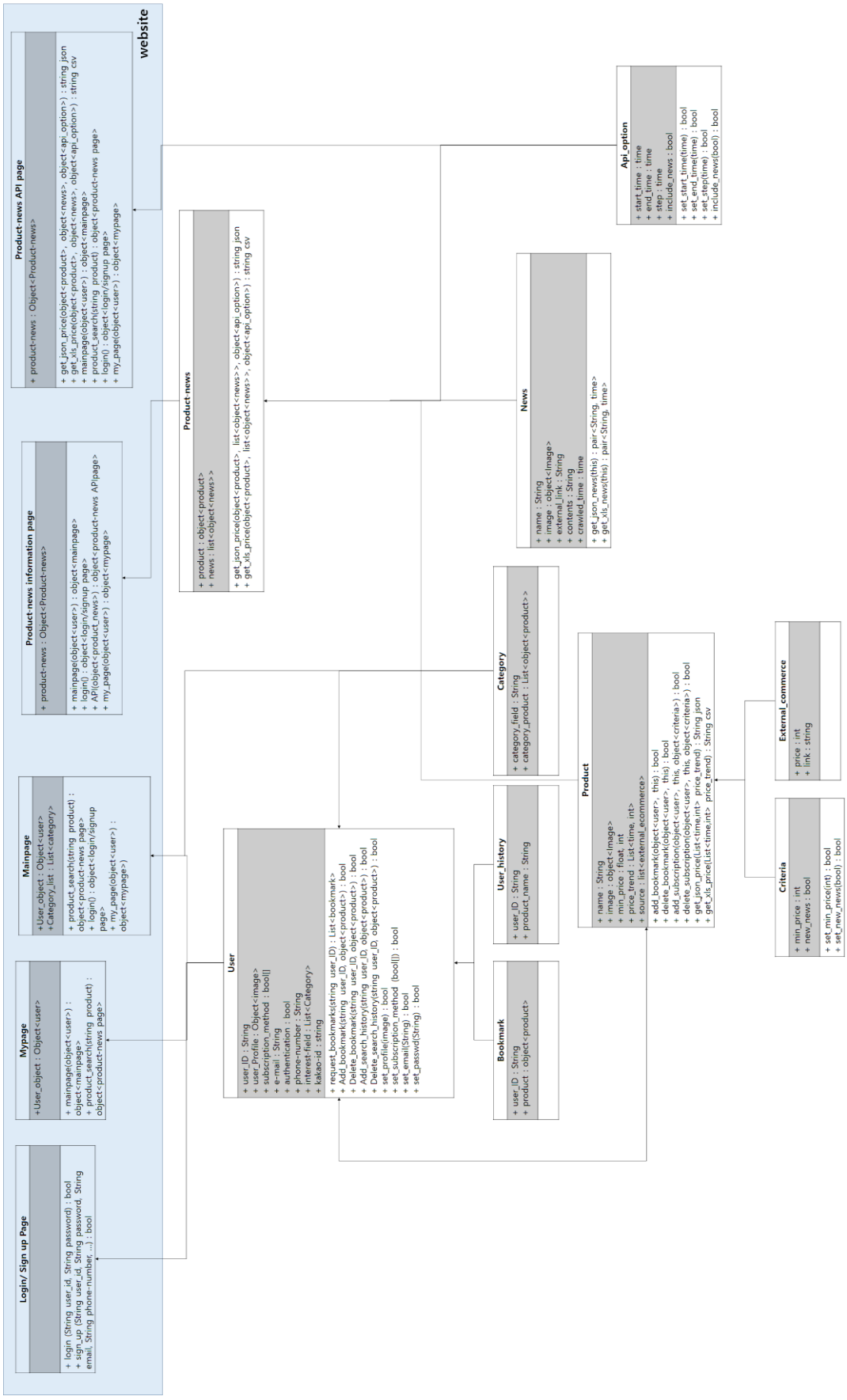

Diagram 8. Front-end Class Diagram

아래부터는 위의 class diagram을 page별로 분리하여 나타낸 것이다.

A. Main

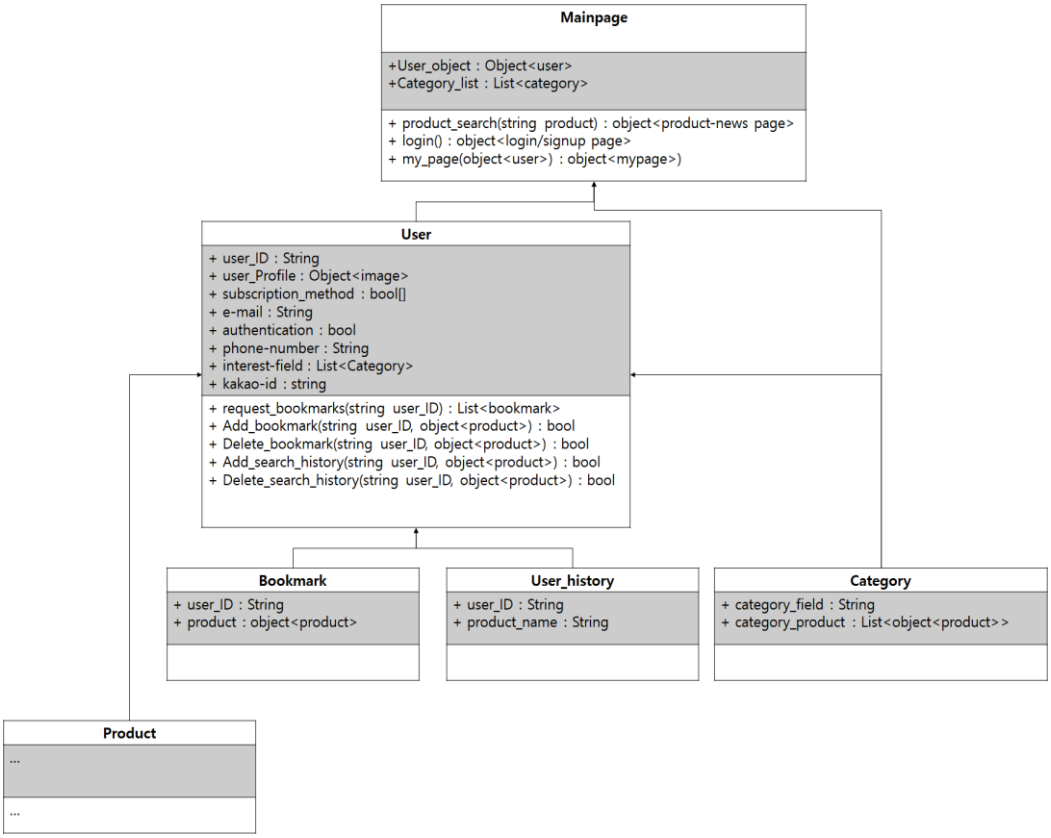

Diagram 9. Front-end Main Page Class Diagram

A-1. Mainpage Object

Attribute

+ User Object : Object<user>

User의 로그인 상태, 정보들이 담긴 object이다. 로그인 과정을 통해서 현재 사용자의 user information을 담아둔다. user object의 자세한 attribute와 method는 밑에서 기술한다. user object는 대부분의 페이지에 걸쳐 전반적으로 사용된다.

+ Category_list : List<category>

물품의 카테고리를 분류한 것을 main page에서 나열하기 위한 attribute이다.

Method

+ product_search(String product_name) : object<product-news page>

product name을 인자로 넘겨주어 product를 search한다. 반환은 product-news page object로, product-news page를 보여준다.

+ login() : object<login/signup page>
  login을 시도할 경우 login/signup page를 반환 받는다.

+ mypage() : object<mypage>
  mypage 보기를 시도할 경우 mypage를 반환 받는다.

A-2. User Object

Attribute

+ user_ID : String
  User의 실질적인 login ID를 의미하는 attribute이다.

+ User_Profile : Object<image>
  user의 profile imag를 갖고 있는 attribute이다.

+ subscription_method(bool[])
  user가 설정한 subscription method에 대한 정보를 가지고 있는 attribute이다. 다양한 method가 존재함과 더불어 future에 method 추가의 여지가 있으므로 array형태로 갖고 있다.

+ e-mail : String
  user가 설정한 e-mail을 갖고 있는다. 이 e-mail은 subscription 수단으로서 활용될 수 있다.

+ authentication : bool
  user가 인증하였는지 여부를 나타내는 attribute이다.

+ phone-number : String
  user의 phone number를 String 형태로 갖고 있는다. 이 전화번호를 통해 subscription 수단으로서 활용될 수 있다.

+ kakao-id : String

user의 kakao-id를 String 형태로 갖고 있는다. 이는 subscription 수단으로서 활용될 수 있다.

Method

+ request_bookmarks(string user_ID) : List<object<bookmark>>
  user_ID를 사용하여 user_ID와 연동된 bookmark object를 List형태로 받아온다. 이를 통해 main 화면에서 bookmark를 구현할 수 있다.

+ Add_bookmark(string user_ID, object<product>) : bool
  main화면에서 사용될 method는 아니지만, 다른 page에서 사용 할 수 있는 method이다. user_ID와 product를 넘겨주어 bookmark에 추가할 수 있다.

+ Delete_bookmark(string user_ID, object<product>) : bool
  main화면에서 사용될 수 있는 method로, user_ID와 삭제할 product를 넘겨주어 bookmark에서 삭제할 수 있다.

+ Add_search_history(string user_ID, object<product>) : bool
  user_ID와 함께 검색한 product object를 활용하여 특정 user의 history를 만들 수 있다. 검색 history는 추후에 검색을 할 때 사용될 정보이다.

+ Delete_search_history(string user_ID, object<product>) : bool
  Add search history와 원리는 비슷하나 삭제한다는 차이가 있다.

A-3. Bookmark Object

Attribute

+ user_ID : String
  bookmark는 user_ID와 product를 연결해 줌으로써 가능하다. 그 때, user ID를 저장하기 위한 attribute이다.

+ product : object<product>
  user ID에 연결되기 위한 product attribute 이다.

A-4. User history

Attribute

+ user_ID : String

user history는 user_ID와 product_name을 연결해 줌으로써 가능하다. 그 때, user ID를 저장하기 위한 attribute이다.

+ product_name : String
  user ID에 연결되기 위한 product name attribute이다.

A-5. category

Attribute

+ category_field : String
  사용자에게 category 정보를 보여주기 위한 class로, category field는 product들의 범주를 의미한다.

+ category_product : List<object<product>>
  해당 범주에 포함될 product들을 나열한 attriute이다.

B. Product-news information page

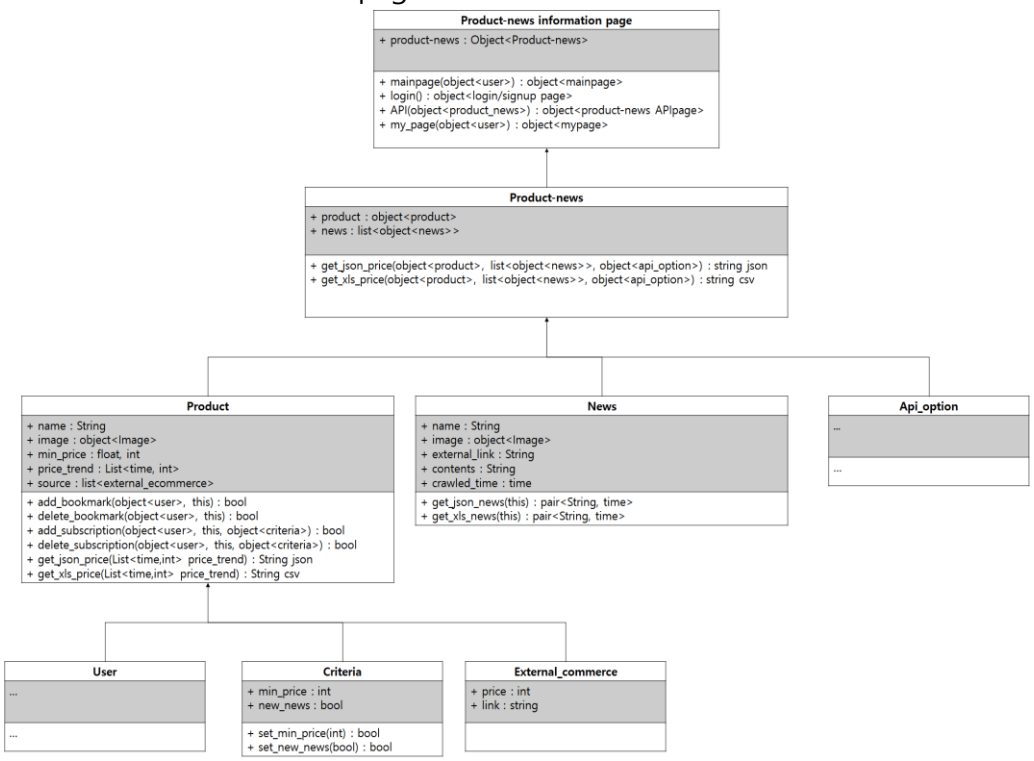

Diagram 10. Front-end Product-news Information Page Class Diagram

B-1. Product-news information page Object

Attribute

+ product-news : Object<Product-news>
  해당 page에서 product와 news의 관계를 표현하기 위한 object attribute이다.

method

+ mainpage(object<user>) : object<mainpage>
  object<user>를 담아서 main_page를 띄워준다.

+ login() : object<login/signup page>
  main page와 동일

+ API(object<product-news>) : object<product-news API page>
  현재 검색한 product-news를 인자로 하여 API page로 전송한다. API page에서는 이 인자를 활용하여 JSON 파일이나 xls 파일을 생성할 수 있다.

+ mypage() : object<mypage>
  main page와 동일

B-2. Product-news Object

Attribute

+ product : Object<product>
  product와 news 사이에서 relation을 나타내기 위한 요소 중 product attribute이다.

+ news : List<Object<news>>
  하나의 product에 여러 news가 관련되어 있을 수 있다. 따라서 news object를 list 형태로 관리해야 한다.

method

+ get_json_price(object<product>, List<object<news>>, object<api_option>) : String json
  현재 가지고 있는 product object와 news object를 활용하여 API 요청을 할 수 있다. 이 경우 json을 string으로 돌려받는다. api_option object를 통하여 사용자가 보다 더 세밀하게 옵션을 선택할 수 있다. 자세한 내용은 아래에서 기술한다.

+ get_xls_price(object<product>, List<object<news>>, object<api_option>) : String csv
  get_json_price와 유사하지만 xls형태로 돌려 받는다.

B-3. Product Object

Attribute

+ name : string
  product의 이름이다.

+ image : object<Image>
  product와 관련된 image를 저장하는 attribute이다.

+ min_price : float, int
  product의 최저가를 나타낸다. 단, 가격이 질량 단위와 같은 단위가 존재하면 float을 사용할 수도 있다.

+ price_trend : list<time, int>
  product의 가격 추이를 저장한 attribute이다. 시간과 함께 저장하여 언제 그 가격을 가졌는지도 알 수 있도록 한다.

+ source : list<external_ecommerce>
  product price에 대한 외부 크롤링 해온 결과를 저장한다. 이를 활용하여 사용자가 홈페이지로 이동하고 싶은 경우 링크를 연결해줄 수 있다.

method

+ add_bookmark(object<user>, this) : bool
  현재 로그인 한 user 객체와 이 method를 사용하는 product 개체를 전송하여 bookmark에 추가한다.

+ delete_bookmark(object<user>, this) : bool
  현재 로그인 한 user 객체와 이 method를 사용하는 product 개체를 전송하여 bookmark에서 제거한다.

+ add_subscription(object<user>, this, object<criteria>) : bool

현재 로그인 한 user 객체와 이 method를 사용하는 product 객체, 그리고 사용자가 지정한 세부 가격, 새 소식 등의 criteria object까지 전송하여 subscription product로 등록한다.

+ delete_subscription(object<user>, this, object<criteria>) : bool
  add_subscription과 마찬가지로 argument들을 전송하며, delete에서는 subscription product 목록에서 제거한다.

+ get_json_price(list<time, int> price_trend) : String json
  product 객체에 있는 price_trend를 활용하여 함수를 호출하고, json 형태의 string을 돌려받는다.

+ get_xls_price(list<time, int> price_trend) : String csv
  product 객체에 있는 price_trend를 활용하여 함수를 호출하고, xls 형태의 string을 돌려받는다.

B-4. News Object

Attribute

+ name : string
  뉴스의 제목이다.

+ image : object<image>
  뉴스와 관련된 이미지이다. 없을 수도 있다.

+ external_link : string
  뉴스의 근본적인 웹사이트 주소를 저장한다.

+ contents : string
  뉴스의 내용을 저장한다. 상황에 따라 일부만 저장할 수 있다.

+ crawled_time : time
  해당 뉴스 object를 가져온 시간을 기록한다.

Method

+ get_json_news(this) : pair<String, time>

현 object를 넘기어 API에서 활용할 수 있도록 한다.

+ get_xls_news(this) : pair<String, time>
  현 object를 넘기어 API에서 활용할 수 있도록 한다.

B-5. criteria

Attribute

+ min_price : int
  사용자가 알림을 받기 위해 product가 도달해야 할 가격을 의미한다.

+ new_news : bool
  만약 true라면 새 소식이 등록될 때 사용자에게 알림을 준다.

method

+ set_min_price(int) : bool
  min_price 값을 설정한다. 음수일 수 없다.

+ set_new_news(bool) : bool
  new_news 값을 설정한다.

B-6. external_ecommerce

Attribute

+ price : int
  외부 commercial 사이트의 가격을 저장한다.

+ link : string
  외부 commercial 사이트의 주소를 저장한다.

C. API

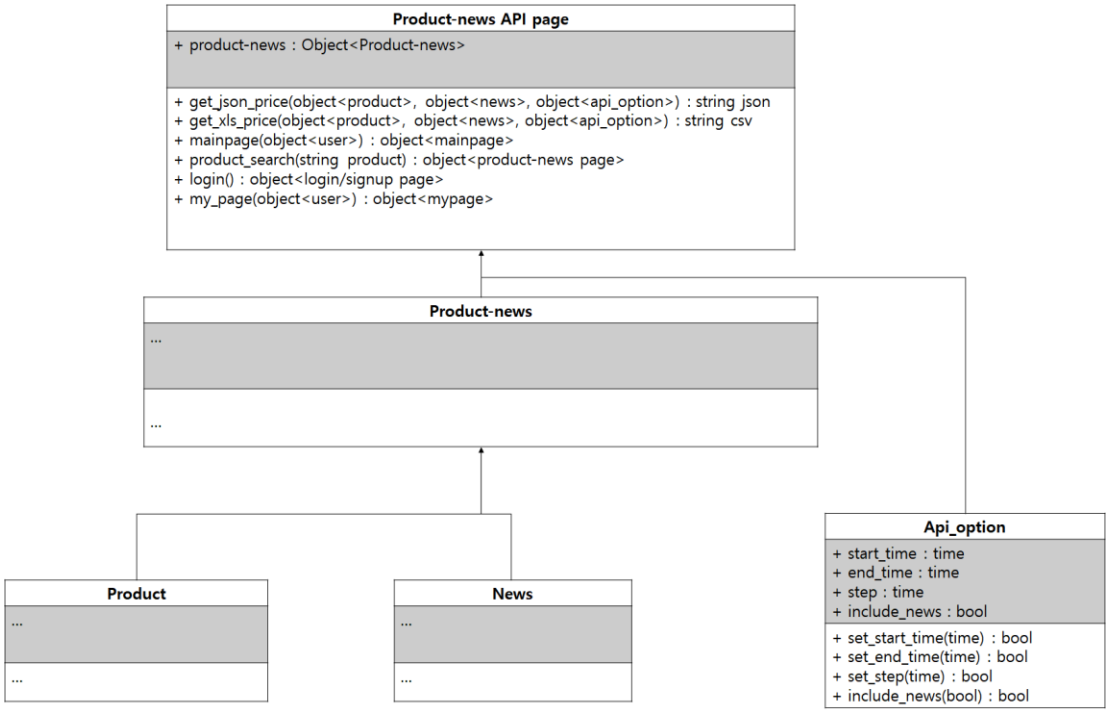

Diagram 11. Front-end API Page Class Diagram

C-1. API page

Attribute

+ product-news : Object<product-news>

product-news object를 갖고 있으며, 이를 사용하여 API를 요청한다.

method

+ get_json_price(object<product>, object<news>, object<api_option>) : String json

product-news page에서 설명한 바와 같다.

+ get_xls_price(object<product>, object<news>, object<api_option>) : String csv

product-news page에서 설명한 바와 같다.

+ mainpage(object<user>) : object<mainpage>
+ product_search(string product) : object<product-news page>
+ login() : object<login/signup page>
+ my_page(object<user>) : object<mypage>

main page와 product-news page에서 설명한 바와 같다.

C-2. api_option

Attribute

+ start_time : time
  API 정보를 받고 싶은 기간의 시점이다.

+ end_time : time
  API 정보를 받고 싶은 기간의 종점이다.

+ step : time
  API 정보에서의 시간 간격이다. 만약 10분이라면 10분단위의 데이터를 제공한다.

+ include_news : bool
  API 정보에 News를 포함할지 정할 수 있다.

Method

+ set_start_time(time) : bool
  start_time을 설정한다.

+ set_end_time(time) : bool
  end_time을 설정한다.

+ set_step(time) : bool
  step를 설정한다.

+ include_news(bool) : bool
  include_news value를 설정한다.

D. MyPage

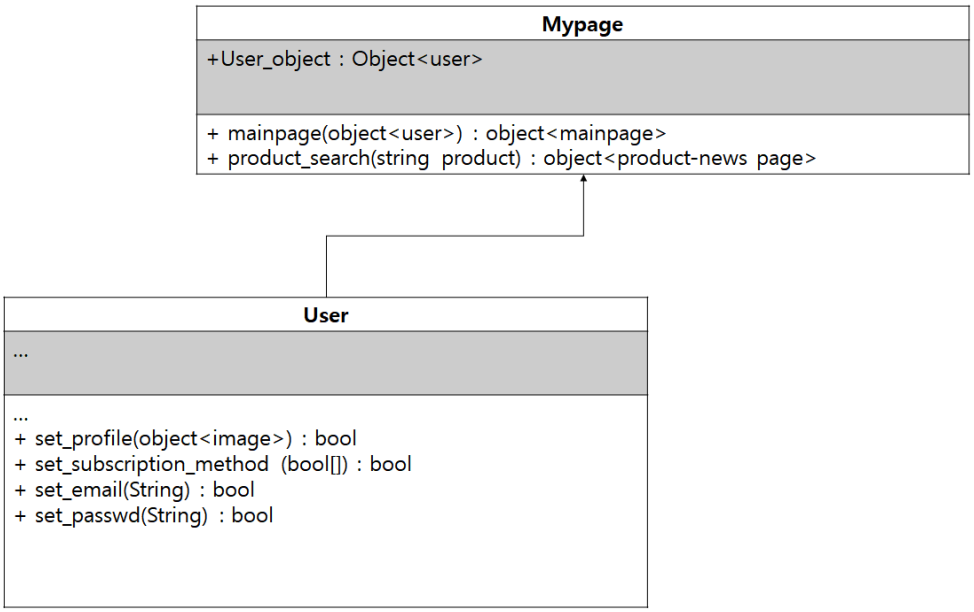

Diagram 12. Front-end Mypage Class Diagram

D-1. Mypage Object

Attribute

+ User_object : Object<user>

my page이므로 user object는 반드시 갖고 있어야 한다.

Method

+ mainpage(object<user>) : object<mainpage>
+ product_search(string product) : object<product-news page>

main page와 product-news information page에서 기술한 바와 같다.

D-2. User Object

main page에서 기술한 user object의 method에 비하여 추가되는 method가 존재한다.

Method

+ set_profile(object<image>) : bool

image를 사용하여 profile을 설정한다.

+ set_subscription_method(bool[]) : bool

subsciprion method는 다양하게 존재한다. 따라서 bool array를 통해서 각 method에 대한 activation을 설정할 수 있다.

+ set_email(string) : bool
  email을 설정할 수 있다.

+ set_passwd(string) : bool
  비밀번호를 변경할 수 있다.

E. Login/sign up Page

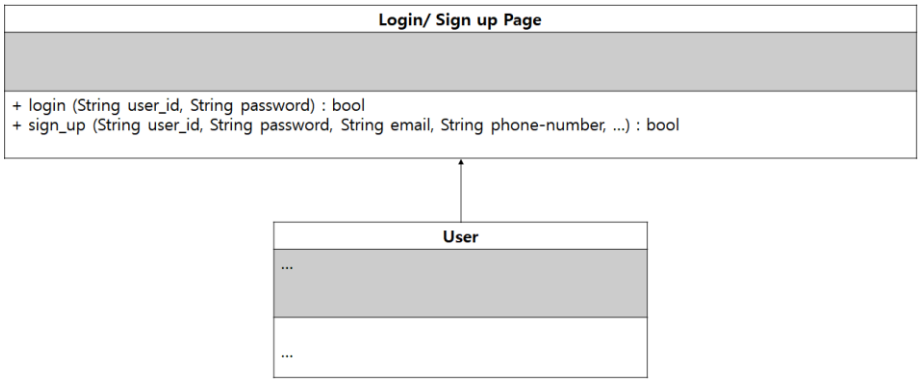

Diagram 13. Front-end Login/Sign up Page Class Diagram

E-1. Login/ Sign up page object

method

+ login(String user_id, String password) : bool
  로그인을 시도한다.

+ sign_up (String user_id, String password, String email, String phone-number, ...) : bool
  회원 정보들을 입력하여 회원가입을 시도한다.

E-2. User Object

기존에 언급된 것과 똑같다.

5. Subsystem architecture- Back-end

이 단원은 Back-end 시스템의 구조와 subcomponent 간의 구성도 및 각 요소의 역할 등을 여러 종류의 다이어그램을 통해 설명한다.

5.1. Subcomponents

Django back-end는 데이터베이스 접근자, 데이터 가공자, 데이터 생산자, 그리고 데이터 제시자로 역할을 나눌 수 있고 데이터 흐름은 데이터 접근자/생산자->데이터 가공자(들)->데이터 접근자/제시자 순서로 이어진다. 접근자와 제시자는 Django에서 주어진 것을 이용하면 되고 개발 범위에 있는 것은 데이터 생산자와 가공자로, 각각 server, data collector라고 이야기할 수 있다. 구조는 다음과 같다.

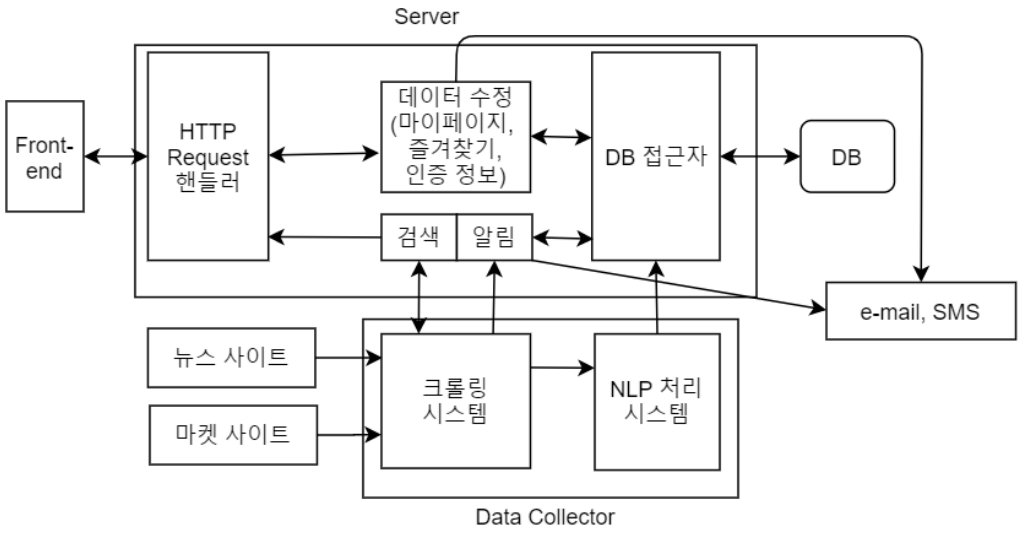

Diagram 14. Back-end Subcomponents Interaction

A. Server

각각의 view(컨트롤러)는 http request 핸들러에서 request를 받는 과정과 db 접근자를 이용하는 과정을 포함한다. 검색, 즐겨찾기 제거/토글(제거는 즐겨찾기 모음에서 하고, 토글은 검색 결과창에서 한다), JSON 데이터 생성, 가격 변동 알림, 데이터 수정(알림 설정, 인증 정보 생성/수정/삭제, 개인 정보 수정) view가 있으며 각각에 대하여 필요한 경우 sequence diagram 또는 data flow diagram으로 설명한다.

A-1. 인증 정보 생성(회원 가입)

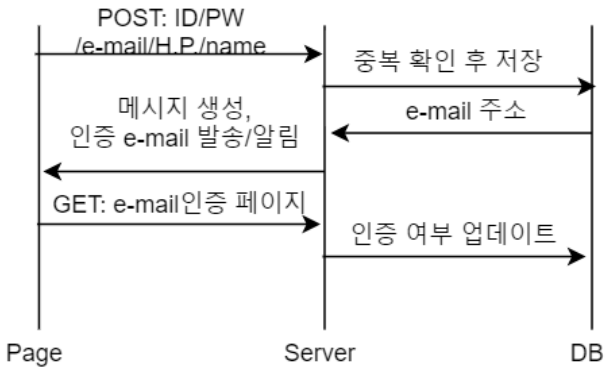

Diagram 15. Back-end Signing Up View Sequence Diagram

인증 여부가 False인 경우에도 중복 확인에 포함된다.

A-2. 로그인

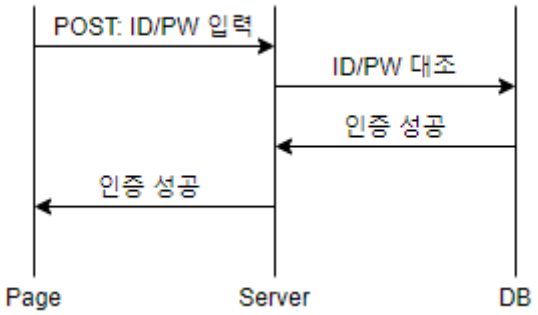

Diagram 16. Back-end Signing In View Sequence Diagram

E-mail 인증 여부가 False인 경우 로그인은 할 수 없다.

A-3. 인증 정보 수정

id에 등록된 e-mail로 무작위로 재설정된 pw를 보내는 view와 로그인 상태에서 pw를 변경하는 view가 있다.

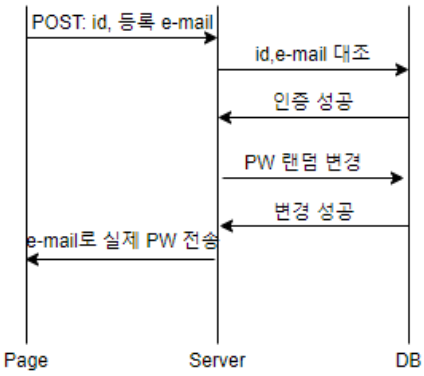

Diagram 17. Back-end PW Change E-mail View Sequence Diagram

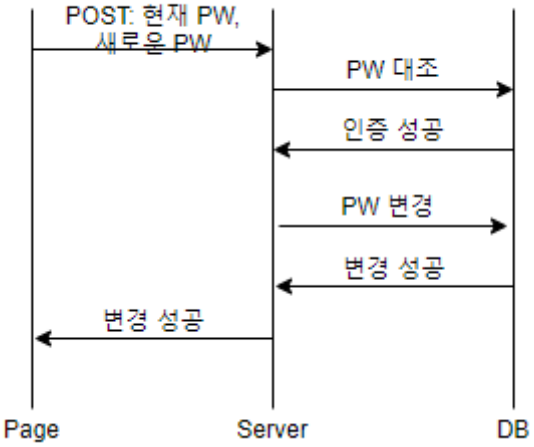

Diagram 18. Back-end PW Change View Sequence Diagram

A-4. 인증 정보 삭제(회원 탈퇴)

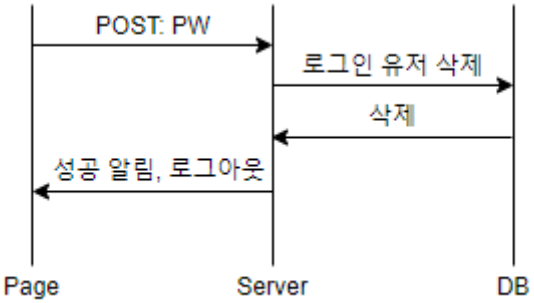

Diagram 19. Back-end Withdrawal View Sequence Diagram

A-5. 개인 정보 수정(알림 수단 설정, 이름 수정, 이미지 변경), 즐겨찾기 토글/제거, 알림 설정

Front-end에서 실제로 작성해야 하는 양식은 다르지만 거치는 과정을 나타내면 같다.

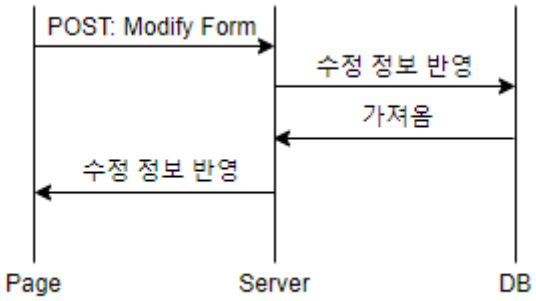

Diagram 20. Back-end Changing Data View Sequence Diagram

이미지 변경에 대하여서는 외부 이미지만을 사용하는 것으로, 정보 반영 양식을 통일한다.

A-6. 검색

검색어 입력에서 결과 출력까지의 과정 사이에는 크롤링과 NLP 처리가 포함된다. 그것 대해서는 B절에서 다룬다.

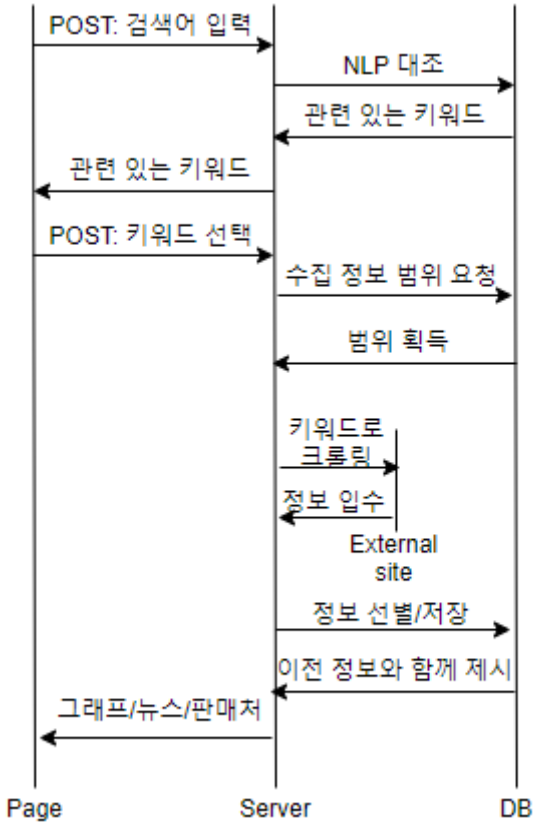

Diagram 21. Back-end Searching View Sequence Diagram

A-7. 상시 가격 수집 및 알림

일정 간격으로, 각 키워드에 대하여 다음을 반복한다.
반복은 서버 측에서 at 등을 등록하여 수행 가능하다.

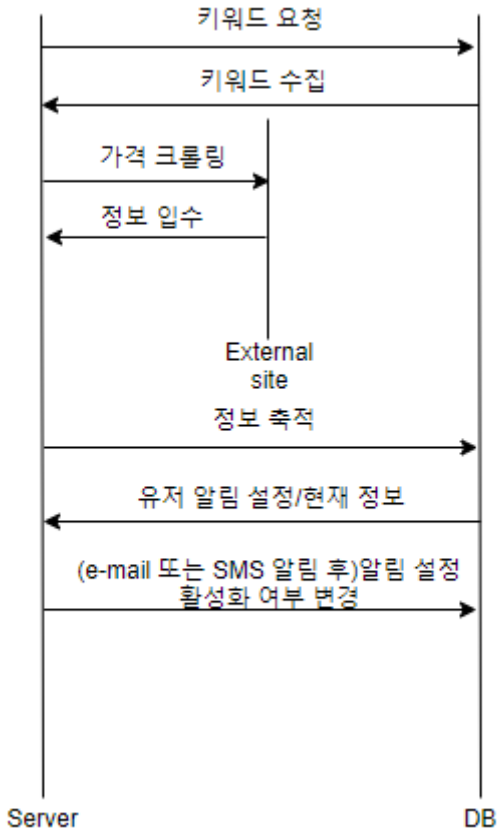

Diagram 22. Back-end Collect/Alarm View Sequence Diagram

A-8. 데이터를 JSON/xls로 변환

그래프는 javascript를 통해 그리게 된다. 이는 페이지에서 처리하므로, 서버에서는 JSON과 xls파일을 제공하는 것을 담당한다.

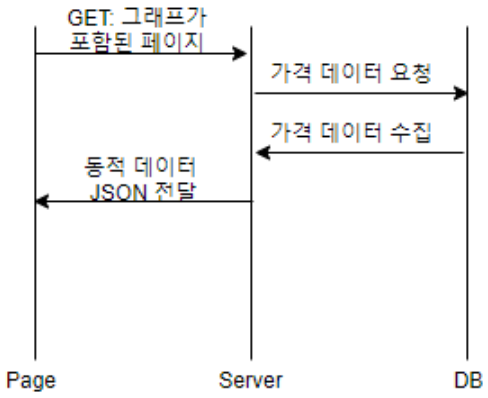

Diagram 23. Back-end data Transforming View Sequence Diagram

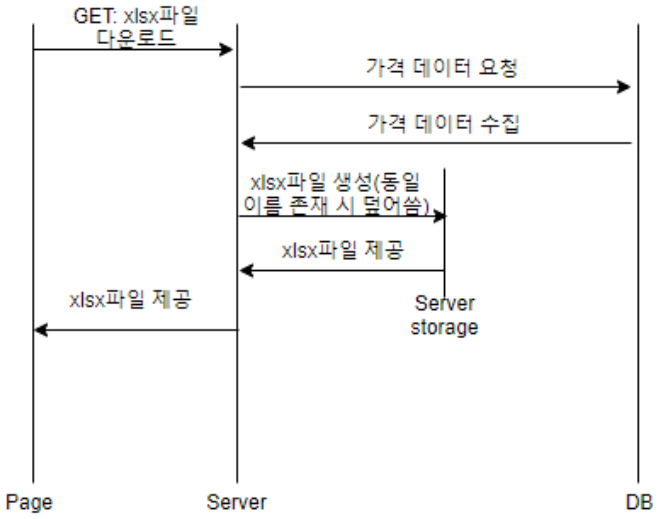

Diagram 24. Back-end Data Transforming View Sequence Diagram

Server Storage는 DB 외의 서버 측 공간을 말한다.

B. Data Collector

Data Collector는 Crawling system과 NLP 처리 시스템으로 이루어져 있으며, 상품의 가격, 뉴스 정보를 모아서 처리하는 기능을 수행한다.

B-1. 상품의 가격 수집

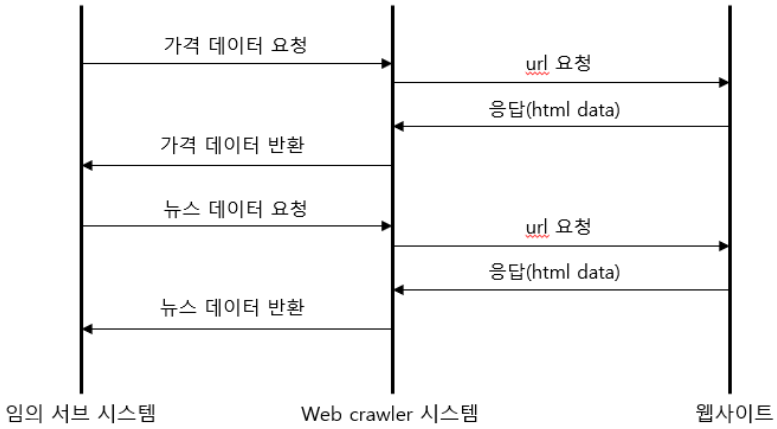

Diagram 25. Back-end Web Crawling System Sequence Diagram

B-2. 상품 관련 뉴스 수집

B-2.1) Class diagram

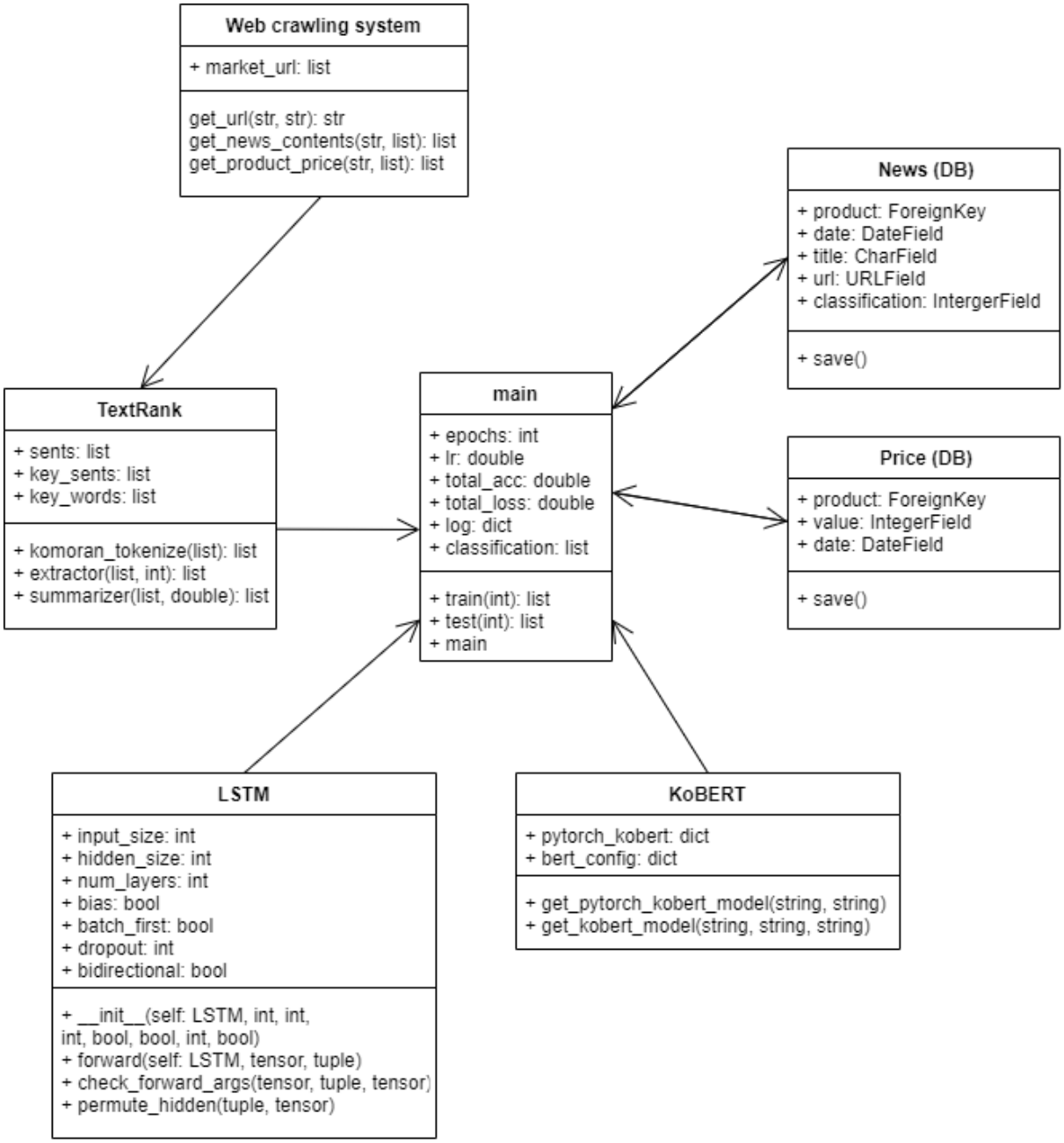

Diagram 26. Web Crawling System Class Diagram

(1) Web crawling system

Attribute

+ market_url: list

웹 크롤링을 하기 위한 목표 웹 사이트 url들의 리스트이다.

Method

+ get_url(str, str): str

url을 가져오기 위한 메소드이다.

+ get_news_contents (str, list): list
  News 정보를 가져오기 위한 메소드이다.

+ get_product_price (str, list): list
  가격 정보를 가져오기 위한 메소드이다.

(2) TextRank

Attribute

+ sents: list
  TextRank를 적용시키기 위한 문장들의 리스트이다.

+ key_sents: list
  TextRank가 적용되어 요약된 문장들의 리스트이다.

+ key_words: list
  TextRank가 적용되어 뽑아낸 키워드들의 리스트이다.

Method

+ komoran_tokenize(list): list
  코모란을 적용시킨 tokenizer를 만드는 메소드이다.

+ extractor(list, int): list
  키워드를 뽑아내는 메소드이다.

+ summarizer(list, double): list
  요약 문장을 생성하는 메소드이다.

(3) LSTM

Attribute

+ input_size: int
  Input x의 예상되는 features의 숫자이다.

+ hidden_size: int
  Hidden state h의 features의 숫자이다.

+ num_layers: int
  Recurrent layer의 숫자이다.

+ bias: bool
  False라면 layer가 bias weight를 사용하지 않는다. Default 값은 True이다.

+ batch_first: bool
  True라면 input과 output tensor가 (batch, seq, feature)의 형태로 주어진다. Default 값은 False이다.

+ dropout: int
  0이 아닌 값이라면 마지막 layer를 제외한 각각의 LSTM layer의 outputs에 Dropout layer를 추가한다. Default 값은 0이다.

+ bidirectional: bool
  True라면 bidirectional LSTM이 된다. Default 값은 False이다.

Method

+ __init__(self: LSTM, int, int, int, bool, bool, int, bool)
  초기화하는 메소드이다.

+ forward(self:LSTM, tensor, tuple)
  Forwarding 하는 메소드이다.

+ check_forward_args(tensor, tuple, tensor)
  Forwarding 할 때 forward arguments를 체크하는 메소드이다.

+ permute_hidden(tuple, tensor)
  Forwarding할 때 hidden을 permute하는 메소드이다.

(4) KoBERT

Attribute

+ pytorch_kobert: dict
  Pre-trained KoBERT 모델을 가져오기 위한 정보이다.

+ bert_config: dict
  Transformers 라이브러리의 BERT를 사용하기 위한 config 파일이다.

Method

+ get_pytorch_kobert_model(string, string)
  Pytorch의 Pre-trained KoBERT 모델을 가져오기 위한 메소드이다.

+ get_kobert_model(string, string, string)
  Pre-trained KoBERT 모델을 가져오기 위한 메소드이다.

(5) News (DB)

Attribute

+ product: ForeignKey
  뉴스와 관련된 상품명을 나타낸다.

+ date: DateField
  뉴스의 생성 날짜를 나타낸다.

+ title: CharField
  뉴스의 제목을 나타낸다.

+ url: URLField
  뉴스의 url을 나타낸다.

+ classification: IntegerField
  뉴스가 어떤 분류를 가지고 있는지 나타낸다.

Method

+ save()
  DB에 저장하기 위한 메소드이다.

(6) Price (DB)

Attribute

+ product: ForeignKey

가격과 관련된 상품을 나타낸다.

+ value: IntegerField
가격을 나타낸다.

+ date: DateField
가격의 날짜를 나타낸다.

Method

+ save()
DB에 저장하기 위한 메소드이다.

(7) main

Attribute

+ epochs: int
학습을 진행하기 위한 전체 epoch 수를 나타낸다.

+ lr: double
학습을 진행하기 위한 learning rate 값을 나타낸다.

+ total_acc: double
학습이 진행된 후 전체 accuracy를 나타낸다.

+ total_loss: double
학습이 진행된 후 전체 loss를 나타낸다.

+ log: dict
학습과 테스트 과정의 log를 나타낸다.

+ classification: list
학습 후 뉴스가 classification된 결과를 나타낸다.

Method

+ train(list): list
학습을 진행하는 메소드이다.

+ test(int): list
  테스트를 진행하는 메소드이다.
+ main
  학습과 테스트를 총괄하는 메소드이다.

B-2.2) Sequence diagram

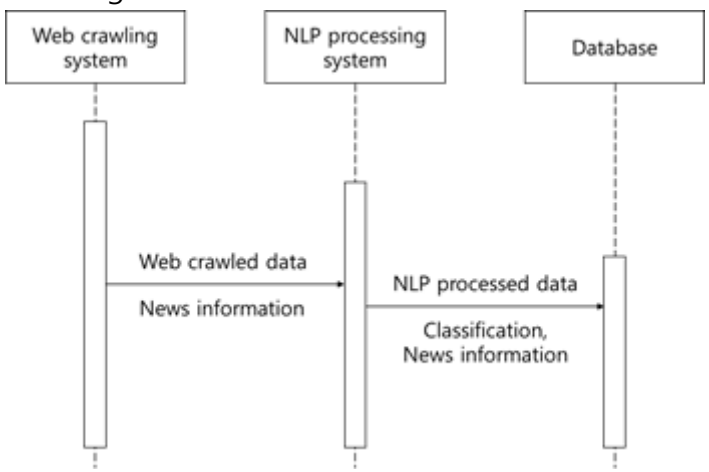

Diagram 27. Back-end Data Collecting System Sequence Diagram

6. Protocol design

Protocol Design 챕터에서는 newShop의 여러 System들 간의 communication에 있어 필요한 Protocol에 대해 기술한다. Protocol을 사용하기 위해서는 기본적인 data format이 필요하다. Data format에는 가장 널리 알려진 XML, JSON 이 있다. 웹과 서버 사이의 데이터 포맷은 장고에서 지원하는 템플릿을 이용할 것이며 API 제공 기능에서 JSON을 data format으로 사용할 것이다. 아래에서는 데이터 포맷과 System들 간의 Protocol에 대해 기술할 것이다.

6.1. JSON

JSON은 속성-값 쌍 이나 키-값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위한 개방형 표준 포맷이다. 비동기 AJAX 또는 XML을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 데이터를 주고 받을 때 자료를 표현하는 방법으로 잘 알려져 있으며 변수 값을 표현하는데 적합하다.

6.2. Django Template

Django에서 템플릿은 MVC Framework에서의 View와 비슷한 역할을 한다. Django에서 render 함수를 통해 페이지를 전달하고 render 함수의 파라미터로 request, template, Optional을 가진다. template은 해당 html 페이지 이며 Optional은 View에서 템플릿에 전달한 데이터를 Dictionary로 전달한다.. Dictonary의 Key는 템플릿에서 사용할 변수명이고 Value는 데이터의 내용을 담게 되어 서버의 데이터를 웹페이지로 전달하여 나타낸다.

6.3. Details

A. User

A-1. Signup

Request

Method	POST	
URL	/auth/signup	
Parameters	id	ID

	name	이름
	password	비밀번호
	email	이메일 주소
	phone	전화번호

Table 2. Sign Up Http Request

Response

Success Code	200 OK	
Failure Code	400 Bad Request(ID가 중복)	
Success Response Body	Url	이메일 인증 Url
Failure Response Body	message	Failure 원인에 대한 정보

Table 3. Sign Up Http Response

A-2. EmailAuth

Request

Method	GET	
URL	/auth/email/:id	
Parameters	-	-

Table 4. Email Authentication Http Request

Response

Success Code	200 OK
Failure Code	404 Not Found(해당 인증 만료)

Success Response Body	-	-
Failure Response Body	message	Failure 원인에 대한 정보

Table 5. Email Authentication Http Response

A-3. Login

Request

Method	POST	
URL	/auth/login	
Parameters	id	ID
	password	비밀번호

Table 6. Login Http Request

Response

Success Code	200 OK	
Failure Code	400 Bad Request(로그인 실패)	
Success Response Body	user data Object	Bookmark(가격, 뉴스), name, profile History등 User관련 Object
Failure Response Body	message	Failure 원인에 대한 정보 (ID 없음, ID/PW불일치, 만료)

Table 7. Login Http Response

A-4. My page

Request

Method	GET	
URL	/mypage/:id	
Parameters	-	-

Table 8. My Page Http Request

Response

Success Code	200 OK	
Failure Code	404 Not Found(로그인 정보 없음)	
Success Response Body	User_data	해당 유저의 정보
Failure Response Body	message	Failure 원인에 대한 정보

Table 9. My Page Http Response

A-5. Edit_user

Request

Method	POST	
URL	/mypage/edit	
Parameters	User_data	해당 유저의 변경 정보

Table 10. Edit User Http Request

Response

Success Code	200 OK
Failure Code	400 Bad Request

Success Response Body	-	-
Failure Response Body	message	Failure 원인에 대한 정보

Table 11. Edit User Http Response

A-6. Delete_user

Request

Method	POST	
URL	/mypage/delete	
Parameters	User_data	회원 삭제 확인에 필요한 비밀번호를 포함한 유저 정보

Table 12. Delete User Http Request

Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	-	-
Failure Response Body	message	Failure 원인에 대한 정보

Table 13. Delete User Http Response

B. Product

B-1. Get

Request

Method	POST

URL	/product/get	
Parameters	id	Product id

Table 14. Product Get Http Request

Response

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	product_list	Product id에 해당하는 Product Object List
Failure Response Body	-	-

Table 15. Product Get Http Response

B-2. Search

Request

Method	POST	
URL	/product/search	
Parameters	name	제품 이름 (검색키워드)

Table 16. Product Search Http Request

Response

Success Code	200 OK	
Failure Code	400 Bad request (해당 제품이 없음)	
Success Response Body	product_id	해당 키워드에 해당하는 제품 id

Failure Response Body	-	-

Table 17. Product Search Http Response

C. News

C-1. Get

Request

Method	POST	
URL	/news/get	
Parameters	id	Product id
	id	User id

Table 18. News Get Http Request

Response

Success Code	200 OK	
Failure Code	400 Bad request	
Success Response Body	news_list	사용자의 기간 설정에 맞춰진 원하는 제품의 뉴스 List
Failure Response Body	message	reason of failure

Table 19. News Get Http Response

D. Alarm

D-1. Set

Request

Method	POST	
URL	/alarm/set	
Parameters	heigh	heigh bound price
	low	low bound price
	id	User id
	id	Product id

Table 20. Alarm Set Http Request

Response

Success Code	200 OK	
Failure Code	400 Bad request	
Success Response Body	-	-
Failure Response Body	message	reason of failure

Table 21. Alarm Set Http Response

E. API

Request

Method	POST	
URL	/api	
Parameters	keyword	제품 키워드

Table 22. API Page Http Request

Response

Success Code	200 OK	
Failure Code	400 Bad request (키워드 해당 제품 없음)	
Success Response Body	product_list	키워드 관련 제품 Object (News, Price)
Failure Response Body	message	reason of failure

Table 23. API Page Http Response

7. Database design

이 장에서는 요구사항 명세서에서 작성한 데이터베이스 요구사항에 대하여 실질적인 데이터베이스 설계를 기술한다. ER Diagram 을 통해 개체 간 관계를 보여주고, 개체 속성의 역할을 명시하고, 실제 개발에 대하여 Relational Schema, 데이터 정의와 접근 양식에 대하여 기술한다.

7.1. ER diagram

테이블이 되는 모든 개체/관계는 PK로 등록 번호를 가진다.

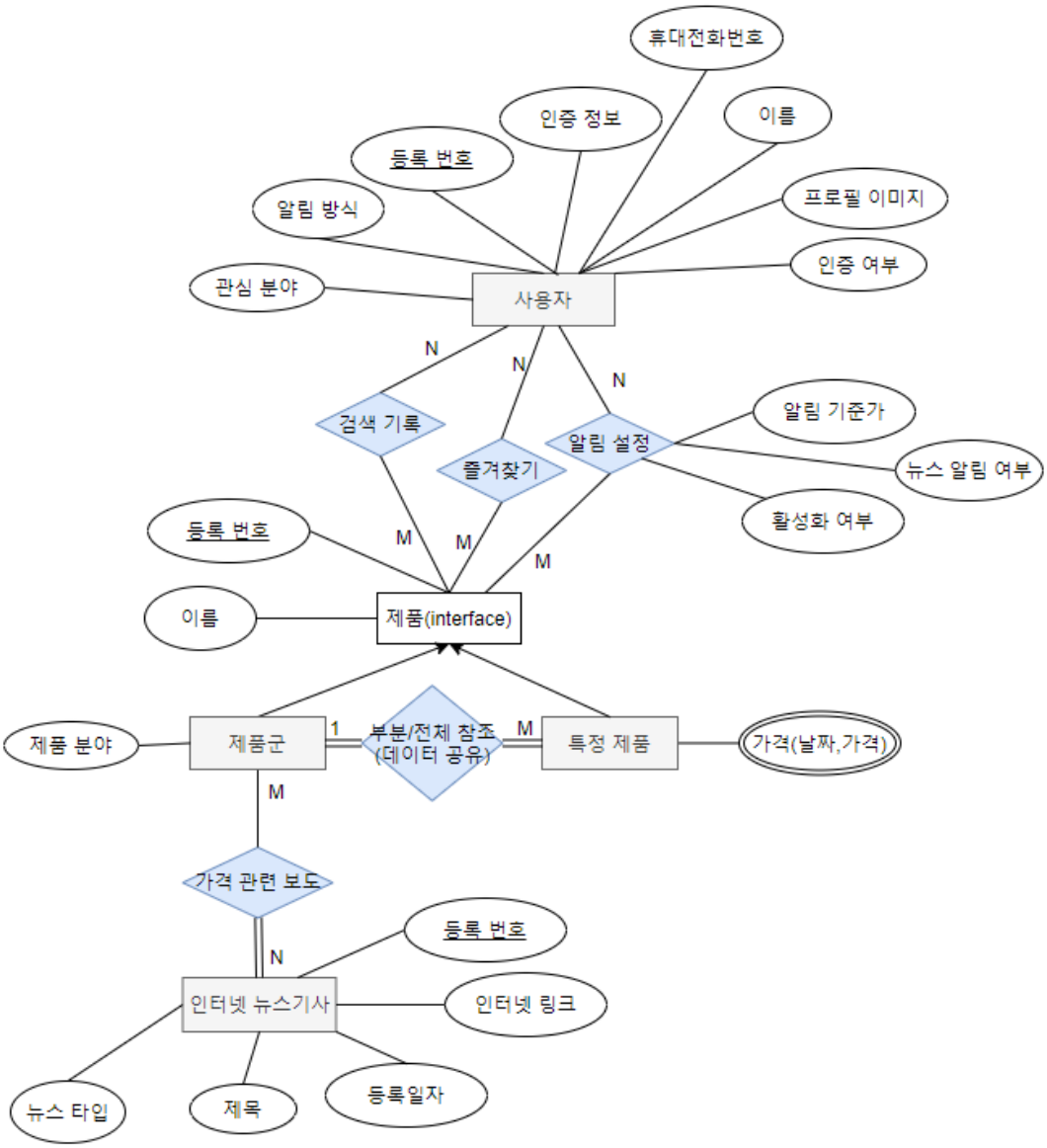

Diagram 28. ER Diagram for Database

A. Entities

A-1. 사용자

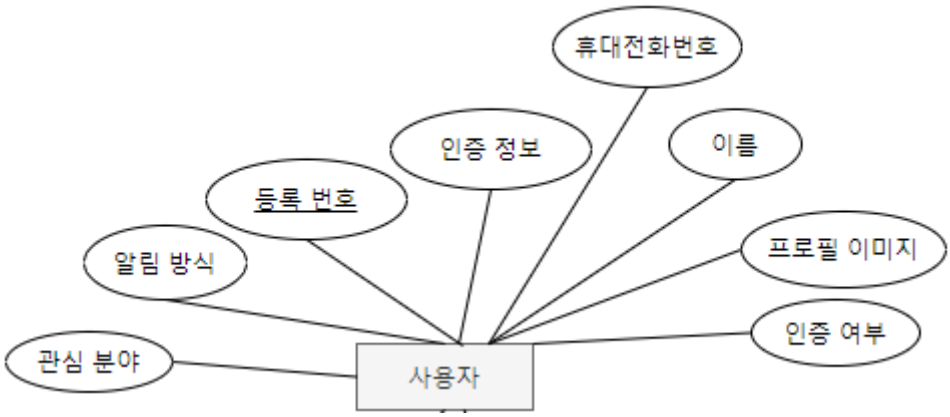

Diagram 29. User Attributes

사용자 인증 정보는 Django에서 제공하는 인증 방식을 사용한다. 즉 개체 '사용자'는 Django에서 제공하는 사용자의 확장형이라 볼 수 있으며, ID, PW 인증 정보와 인증 e-mail이 들어간다.

이름은 로그인한 유저(자신)가 누구인지 상단에서 보여주며, 프로필 이미지는 유저 개인 페이지에서만 사용된다.

인증 여부는 이메일 인증을 완료한 경우에 참이 되어 로그인을 허용한다.

관심 분야는 로그인했을 때 대문에서 보여줄 여러 제품군의 소식을 결정한다. 문자열이며, 정의역은 한정되어 있다. null일 수 있다.

휴대전화번호는 가격 변동 SMS 알림에 사용되며, null일 수 있다.

A-2. 제품(제품군, 특정 제품)

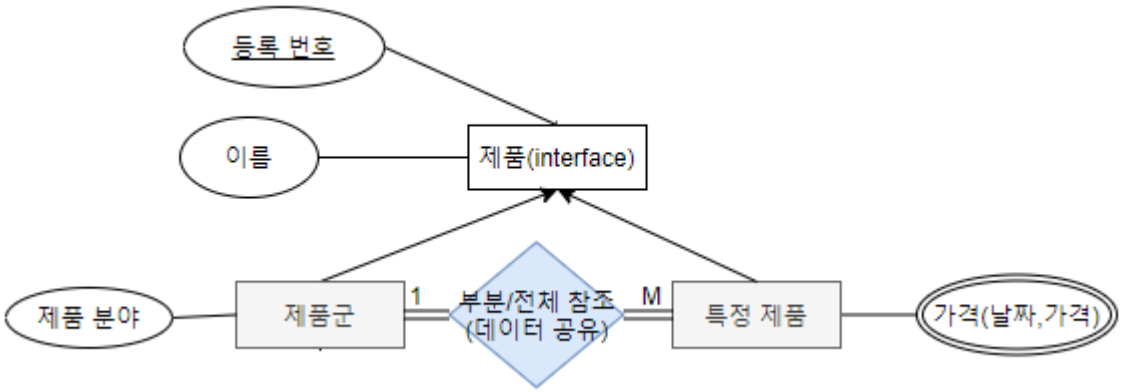

Diagram 30. Product Attributes

제품군과 특정 제품은 제품이라는 인터페이스의 파생 클래스다. 이는 정보(선호 상표)가 없는 사용자에게도 직관적으로 가격을 소개할 수 있게 하기 위함이다.

제품 분야는 여러 제품군의 공통 영역이며(ex: 제품군: 250GB SSD, 제품 분야: 컴퓨터 부품) 사용자의 관심 분야와 정의역을 공유한다.

가격은 마켓 사이트에서 가격을 가져온 것을 말하며 날짜, 가격의 쌍으로 구성되어 있다.

부분/전체 참조(데이터 공유) 관계에 대해서는 후술되어 있다.

A-3. 인터넷 뉴스 기사

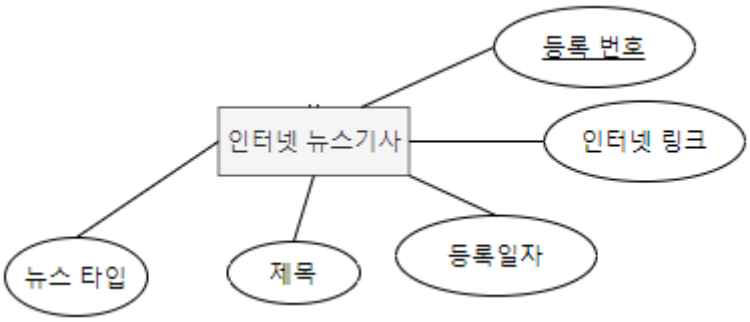

Diagram 31. News Attributes

사이트에서는 인터넷 뉴스 기사의 제목 및 등록일자만을 띄우며, 그것을 누르면 해당 사이트(링크)로 이동하도록 한다.

등록 일자 중 최신을 조사하여 그 이후의 기사만을 크롤링한다.

B. Relations

B-1. 검색 기록, 즐겨찾기, 알림 설정

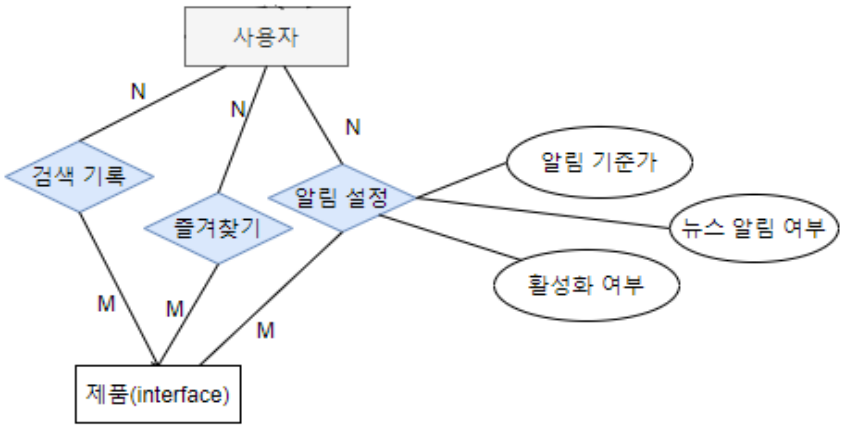

Diagram 32. User-Product Relations

검색 기록, 즐겨찾기는 사용자가 검색어를 입력하지 않고 정보에 접근할 수 있게 한다.

알림 설정은 알림 기준가, 활성화 여부로 표시된다. 기준가는 가격이 내려가면 알려주고, 그 후 다시 일정 수준 이상 올라가면 알려주는(null 가능) 기능이다. 활성화 여부의 경우 한 번 알려 주면 꺼진다.

이 관계들은 테이블로 표현될 것이며, 등록 번호가 포함된다.

B-2. 제품군과 특정 제품의 관계

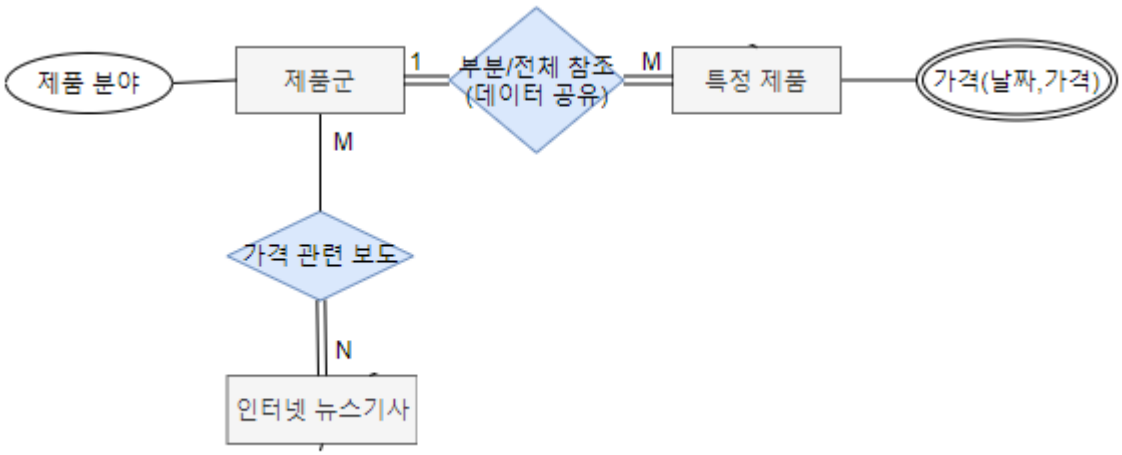

Diagram 33. Product Subclasses' Relations

'제품군'과 '특정 제품'은 모두 '제품'으로 분류된다. 즉 사용자의 정보 검색에서 동등하게 분류되는데, 실제 관계는 특정 제품이 제품군에 포함되지만 제품군과 특정 제품이 가격, 뉴스에 대하여 데이터를 각각 쓰는 것은 공간 효율성이 매우 떨어진다. 그리고 이미 한 크롤링을 또 해야 하는 경우가 생겨 시간 효율성도 매우 떨어지기 때문에, 특정 제품과 제품군 각각을 통해 서로의 데이터를 읽을 수 있도록 하는 것이 적절하다.

그 모든 데이터의 접근은 '제품'클래스에 정의되어야 하며(이는 데이터베이스 수준에서 정의될 수 있는 부분은 아니다), 속성이 아닌 메소드를 상속하는 것으로 해결이 가능하다.

7.2. Relational schema

모든 테이블이 pk로 '등록 번호'를 쓰는 이유는 Django에서 제공하는 pk라는 키를 사용하는 것이 가장 편하기 때문이다.

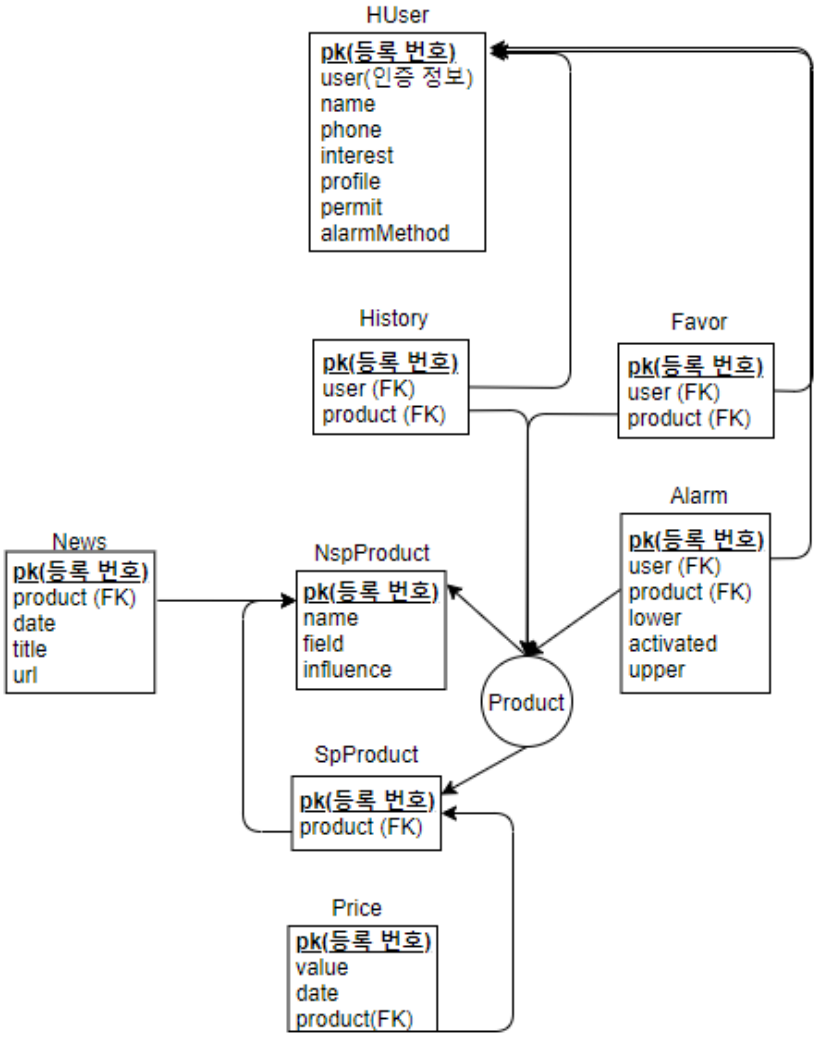

Diagram 34. Relational Schema

NspProduct는 Non-specific 제품(제품군)을 뜻하고 SpProduct는 specific 제품을 뜻한다.

7.3. Data definition and access form

데이터 정의와 접근에 대한 질의어를 Django 함수로 캡슐화 한 것이 있으므로 실제 개발을 위해 그 사용에 대하여 기술한다. 결과는 MYSQL에서의 테이블이기 때문에 SQL로 관리하는 것도 가능하다.

A. Data definition

```python
from django.db import models
from django.conf import settings
import abc
class HUser(models.Model):
    user = models.OneToOneField(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, related_name='handle')
    name = models.CharField(max_length=50)
    phone = models.CharField(max_length=13,null=True)
    interest = models.CharField(max_length=50,null=True)
    # profile = models.ImageField(upload_to=None, height_field=None, width_field=None, max_length=None, null=True)
    permit = models.BooleanField(default=False)
    alarmMethod = models.IntegerField() #비트로 다룸 : ex) 2^0자리 이메일, 2^1자리 문자
class History(models.Model):
    #pk는 전체를 기준으로, 생성 순으로 부여되니 정렬 id는 따로 부여하지 않겠음
    user = models.ForeignKey("HUser", related_name='history', on_delete=models.CASCADE)
    product = models.ForeignKey("Product", on_delete=models.CASCADE)

class Favor(models.Model):
    user = models.ForeignKey("HUser", related_name='favor', on_delete=models.CASCADE)
    product = models.ForeignKey("Product",on_delete=models.CASCADE)

class Alarm(models.Model):
    user = models.ForeignKey("HUser", related_name='alarm', on_delete=models.CASCADE)
    product = models.ForeignKey("Product", related_name='alarm',on_delete=models.CASCADE)
    lower = models.IntegerField(default=0)
    reuse = models.BooleanField()
    upper = models.IntegerField()

class Product(models.Model):    #상표 없는 것과 있는 것의 공통 규약을 위한 추상 클래스
    name = models.CharField(max_length=100)
class NspProduct(Product): #상표 무관 product 키워드를 말함
    field = models.CharField(max_length=50,null=True)
    influence = models.CharField(max_length=100,null=True)
class SpProduct(Product):  #상표가 있는 specific product 키워드를 말함.
    product=models.ForeignKey("NspProduct",related_name='brand',on_delete=models.CASCADE)

class News(models.Model):
    product = models.ForeignKey("NspProduct", related_name='news', on_delete=models.CASCADE)
    date=models.DateField()
    title = models.CharField(max_length=200)
    url = models.URLField(max_length=200)

class Price(models.Model):
    product = models.ForeignKey("SpProduct",related_name='price', on_delete=models.CASCADE)
    value = models.IntegerField()
    date = models.DateField()
```

Figure 3. Classes for Tables

위와 같이, 정의하려는 테이블마다 python 클래스를 정의한 후(메소드는 테이블과 무관하다)
python manage.py makemigrations
python manage.py migrate
로 MYSQL 테이블을 만들 수 있다.

A-1. 데이터베이스 접근

다음은 원하는 테이블의 모든 데이터를 QuerySet 형태로 불러온다. Model.objects.all() (클래스 메소드)

QuerySet은 iterable 객체이며, 구성 항목은 Model의 객체이다.

다음은 쿼리셋에서 속성값에 조건을 주어 필터링한다. Queryset.filter(조건)

다음은 쿼리셋에서 속성값에 따라 정렬한다. Queryset.order_by(속성이름들(str)) 기본적으로 str 리터럴 앞에 -를 붙여 내림차순 정렬이 가능하다.

최종적으로 객체를 얻는 것은 [] 연산자 또는 Queryset.get(조건)을 사용한다.

Product는 추상 클래스이되, 테이블은 만들어야 Product를 참조하여 NspProduct와 SpProduct 객체들이 모두 포함된 테이블을 얻을 수 있다.

A-2. 데이터베이스 업데이트

위 클래스의 생성자를 통해 데이터를 만든 후, Model.save() 함수를 통해 데이터베이스에 저장할 수 있다.

접근을 통해 가져온 객체를 수정한 후, Model.save() 함수를 통해 데이터베이스에 업데이트할 수 있다.

8. Testing plan

Testing plan에서는 newShop 사이트의 테스트에 있어 사용되는 테스트 방법 및 전체 framework에 대해서 설명한다. 이 장에서는 Test strategy, Test Policy, Test case가 설명된다.

8.1. introduction

A. Audiences

Project 팀 멤버들은 이 장에서 설명되는 테스트 방식을 진행하고 팀원에게 테스트의 결과 및 권장 사항을 제공한다.

Project Manager는 전체 프로젝트 일정의 테스트 계획, 문서의 검토, 테스트의 성능 추적, 그리고 테스트 결과에 따른 책임을 진다. 본 newShop application project의 경우 Project 팀 멤버가 이 역할을 포함한다.

Stakeholder와 participants는 비즈니스가 테스트 결과와 일치하는지 확인하기 위해 User 테스트에 참여할 수 있다.

Technical Team 멤버들은 테스트 계획 및 결과물이 디자인과 일치하는지 확인하고 테스트 환경을 제공하며 결함 수정과 관련된 절차를 따른다. 본 newShop application project의 경우 Project 팀 멤버가 이 역할을 포함한다.

8.2. Test strategy

A. Test Objectives

이 테스트의 목적은 newShop application의 모든 기능적 요소들이 웹의 사양 및 기기의 사양에 따라 올바르게 작동하는지 확인하는 것이다. 각 기능별로 테스트 스크립트를 실행 및 검증하고, 기능 테스트의 심각도를 판별하여 심각도를 낮추는 방향으로 향후 수정을 진행한다. 이렇게 진행되는 테스트의 최종 결과는 다음 2가지로 구분된다.

A-1. Production이 준비된 web application newShop
A-2. 재사용될 수 있는 안정적인 test script set

B. Test Assumptions

B-1. Key assumptions

테스팅 과정에 필요한 모든 소프트웨어적, 하드웨어적 요소들은 갖추어져 있음을 가정한다.

테스트 주기에서 결함이 나타난 경우 수정을 거쳐 새로운 테스트 주기를 시작한다.

B-2. General assumptions

모든 테스트 결과와 과정은 프로젝트 팀 내에서 공유될 수 있다.

테스트의 설계 및 실행에 필요한 입력, 테스트 케이스 디자인, 테스트 환경 설계 및 검토, 문서화 등의 모든 활동은 프로젝트 팀 내에서 이루어진다.

테스트는 결과론적으로 이루어진다.

테스트를 진행하는 프로젝트 팀은 지식과 경험이 있거나 테스트 프로세스에 대한 교육을 받았다.

테스트 cycle 중에 결함을 발견하더라도 하나의 cycle을 완료한다.

테스트를 진행함에 있어 newShop application web 내부에서만 진행한다.
UAT 테스트는 최종 사용자에 의해 수행된다.

C. Test Principles

테스트를 오래 진행하지 못하는 환경이므로 시간을 단축시키는 것에 우선순위를 둔다.

테스트는 business objective, cost efficiency 및 품질을 충족시키는 데에 중점을 둔다.

테스트 프로세스는 필요에 따라 변경될 수 있고, 유연하다.

테스트 activity는 중복을 피하기 위해 테스트 이전단계에서 구축된다.

테스트 환경은 production 환경을 최대한 모방한다.

테스트는 명확하게 정의된 목표와 단계로써 이루어진다.

8.3. Test Policy

테스트 과정에 있어, 개발과정에서의 테스팅인 Development testing, 개발 후 production 직전의 release testing, production 이후의 테스트 과정인 User testing과정으로서 테스트를 구분한다. 여기서 Development testing이라는 표현은 좁은 의미로써 사용되지 않고, software testing의 전반적인 내용을 포함한다.

A. Development testing

Development testing 과정은 최소한 Unit Testing, Integration Testing, System testing 이 3가지 Level로 나누어진다. Unit Testing은 각 프로그램에 사용되는 함수나 모듈, 클래스의 testing을 의미하고, Integration Testing은 이러한 Unit들이 모여서 생긴 Components들을 테스트하는 것을 의미한다. 마지막으로 System testing은 Sub-system 단위로 테스트하는 것을 의미한다.

A-1. Compatibility testing

Purpose : newShop은 웹 application이므로, 호환성 테스트가 매우 중요하게 다뤄져야 한다. General한 Browser Software에서 올바르게 가동되어야 하며, OS를 고려해야할 수도 있다.

Scope : 일반적으로, High Level의 범위에서 testing이 진행된다. application으로써 시스템을 완전히 구현한 상황에서 진행되기 때문이다.

Method : Development Environment Version 확인, OS기능을 별도의 모듈 또는 라이브러리로 추상화.

A-2. Regression testing

Purpose : 코드를 고칠 일이 있다면, 즉 웹을 업데이트 할 일이 있다면 업데이트 이후에 생기는 결점에 관해서 testing하는 과정이다. 기능적으로 상당히 손실이 큰 편이므로 testing을 강하게 고려해야한다.

Scope : Low Level의 범위부터 High Level의 범위까지 전부 testing scope에 포함된다.

Method : 만약 새로운 시스템의 추가로 코드의 충돌이 발생했다면, 충돌이 어디에서 발생했는지를 확인하고, 모듈이나 함수를 바꾸거나 system과 component의 재구성을 고려한다.

A-3. Functional testing

Purpose : 자명하게, 제공되는 기능이 제 역할을 해야 하므로 Functional test가 필요하다.

Scope : Testing 자체는 High level에서 이루어진다. 하지만 경우에 따라 Low level에서의 검수가 필요할 수 있다.

Method : 제공하고자 하는 기능이 개발 환경에서 제 역할을 하는지 확인한다.

A-4. Non-functional testing

Purpose : 이는 scalability, performance, security와 같은 사용자의 작업에 눈에 띄는 관련이 없는 비기능적인 측면을 testing한다. 이가 부족하면 실행이 불안정해지거나 심할 경우 중단될 수도 있으므로 필요하다.

Scope : Low-level에서부터 High-level까지 전반적으로 testing이 이루어진다.

Method : 테스트하는 비기능적 요인에 따라 다양하다.

A-5. Usability testing

Purpose : 사용자가 이 어플리케이션을 쉽게 사용할 수 있을지 User interface 등을 테스트한다. User test와는 달리 test의 주체가 개발자이다.

Scope : High-level에서 testing이 이루어진다.

Method : 팀 내에서, 주어진 application interface내에서의 사용을 진행해본다. 이가 가질 수 있는 부정확성은 User test의 측면에서 해결한다.

B. Release testing

Purpose : Release testing과정은 User에게 새로운 버전이 배포되기 직전, 사용에 문제가 생기지 않았음을 확인하는 과정이다.

Scope : 일반적으로 High-level혹은 아예 application level에서 이루어진다. 하지만 결함을 해결하는 과정에서 low-level의 components를 수정해야할 수 있다.

Method : Pre-Alpha test, Alpha test, Beta test, gamma test...등의 acceptance testing으로 구성되어, 이 방식은 Alpha version app, Beta version app 등등을 기반으로 User test와 Development test 사이에서 testing이 이루어진다. 경우에 따라 End-User test의 형식으로 진행되기도 한다.

C. User testing

Purpose : 이 테스트는 비즈니스 logic을 검증하는 데에 중점을 둔다.

Tester : End-User

Method : Application User가 Application requirement를 제공하고, 시스템적으로 이를 조정하는 방법을 사용한다. 이 경우 사용자가 test 스크립트에 표시되지 않은 임의의 사항을 수행해볼 수 있다.

8.4 Test Case

여기서 제안되는 목록은 Development testing의 목록으로, 위 Development testing 유형 중 하나에 속하며, 특히 Functional testing에 속하는 경우가 많다.

User	Scenarios	Sub levels	Complexity	Number of test cases	Negative Test cases
Web User	Login page	Login	-	-	-
	My page	Add info	-	-	-
		Delete info	-	-	-
		Add photo	-	-	-
		Delete photo	-	-	-
		알림 기능 등록	-	-	-
		알림 기능 등록 해제	-	-	-
		검색 이력 관리	-	-	-
		Sign out	-	-	-
	Sign in Page	Sign in	-	-	-
	MainPage	Logout	-	-	-
		검색	-	-	-
		추천 카테고리 목록	-	-	-
		북마크한 상품 제안	-	-	-

		북마크한 상품관련 뉴스 제안	-	-	-
		SNS 공유	-	-	-
	Product Page	상품정보 확인	-	-	-
		상품 알림 등록	-	-	-
		상품 북마크 등록	-	-	-
		상품 북마크 등록 해제	-	-	-
		관련 뉴스 내용 확인	-	-	-
		관련 뉴스 리다이렉트	-	-	-
		가격에 영향 끼치는 요소 확인	-	-	-
		가격 차트 확인	-	-	-
		쇼핑 사이트 리다이렉트	-	-	-
	API Page	가격 차트 확인	-	-	-
		가격 json 다운로드	-	-	-

		가격 xls 다운로드	-	-	-
	General	HTML requirements	-	-	-
		Performance	-	-	-
		Accessibility	-	-	-
		Image display	-	-	-
		Compatibility	-	-	-
		NLP requirements	-	-	-
		Database requirements	-	-	-
		Crawling Requirements	-	-	-

Table 24. Test Cases

9. Development plan

이 단원에서는 본 시스템 개발에 사용되는 프로그래밍 언어(programming language), IDE(Integrated development environment), 프로젝트 관리 툴(project management tool)에 대해 설명한다. 또한, 시스템 개발 일정을 설명한다.

9.1 System environment

A. Django2.0

Figure 4. Django Logo

python으로 작성된 오픈 소스 웹 애플리케이션 프레임워크이다. 데이터베이스 관리와 동적 데이터 처리를 모두 python으로 작성할 수 있다는 특징이 있으며, 관리자 인터페이스, 사용자 인증 시스템 등 기본적인 자체 내장 애플리케이션이 있으며 다양한 외부 패키지를 이용할 수 있다. 팀원들이 모두 python에 익숙하기 때문에 Django를 선택하였다. Visual Studio Code 등 맞춤 린터를 제공하는 IDE를 사용하면 된다.

B. Pythonanywhere

Figure 5. Pythonanywhere Site Logo

pythonanywhere는 가상 환경을 통해 Django로 작성된 애플리케이션을 배포할 수 있다. 회원 등급이 있어 돈을 내고 구독하면 좋은 자원을 사용할 수 있지만 무료 버전은 자원이 크게 한정되는데, 팀원이 동시에 테스트를 하는 데에는 적당한 환경이기 때문에 2차적인 테스트 환경으로 활용한다.

C. sublime Text

Figure 6. Sublime Text Logo

sublime text는 source code를 작성할 때 사용할 수 있는 Tool이다. OS 별로 GUI 툴킷이 따로 있기에 오버헤드가 적다. 다른 Tool에 비해 가볍고 쉽게 사용할 수 있으며, 기능 추가도 간편한 편이다. 이를 통하여 html, css, js 등의 front-end를 작성할 것이다.

9.2. Schedule

A. NLP system schedule

~5/26 : TextRank class

~5/28 : main class

~5/30 : performance check on LSTM, KoBERT model

~5/31 : error checking, final code

B. Server development schedule

~5/26 : develop Authentication part

~5/28 : develop Database retrieving part

~5/31 : Integrate Front-end

~6/3 : Integrate NLP and Crawling & Test code

C. Front-end Template development schedule

~5/26 : Apply site style

~5/30 : Apply forms

~6/3 : Integrate Back-end, NLP and Crawling system

10. Index

이 단원은 그림 인덱스, 표 인덱스를 포함한다.

10.1. Figure Index

10.2. Table Index

10.3. Table Index