샥 Shark

중고 물품 거래 플랫폼

— Design Specification

목차

1 Preface

해당 문서의 예상 독자, version history를 다룬다.

1.1 Readership

Readership

사용자를 예상 독자로 상정한다.
일반적인 사용자가 따르게 될 시스템의
구성요소를 자연어 중심으로 표현하여,
사용자가 보다 쉽게 이해하고,
요구사항을 구체화할 수 있도록 돕는다.

1.2 Version history

2020-05-24, Version 1.0
첫번째 버전입니다.
1조 전원 참여

2020-06-14, Version 1.1
양식을 수정한 버전입니다.
1조 전원 참여

2020-06-21, Version 2.0
샥딜 구매자 대기열과 관련한 내용을 추가한 버전입니다.
해당 내용은 5.5 SharkDeal 부분에 실었습니다.
1조 전원 참여

1.3 Document Structure

◯ Introduction

이 장에서는 본 시스템의 설계에 사용된 다이어그램과 도구를 소개하고, 시스템의 개발 배경이 되는 문제의식과 이 시스템이 해당 문제들을 어떻게 해결할 것인지를 기술한다.

◯ Overall System Architecture

이 장에서는 시스템과 각 서브시스템의 구조를 개괄적으로 서술하고 서브시스템과의 연결과 전체적인 기능을 보여주는데 초점을 맞추었다.

◯ System Architecture - Frontend (Mobile Language)

이 장에서는 기존까지 서술하지 않았던 모바일 개발환경에서 이용되는 언어에 대한 기술이 이뤄진다. 모바일의 경우 Kotlin과 Swift로 개발하게 되기 때문에, Python에 기반한 서버에서 어떻게 정보를 받아와 프론트 엔드의 여러 함수들을 구현할 것인지 사용자에게 보여줄 내용이 담겨있다.

◯ System Architecture - Backend

사용자와 직접적인 접촉을 하는 프론트 엔드를 제외한 백엔드에 대한 내용을 담는다. 백엔드 시스템과 각 서브 시스템들이 연결된 구조를 보여준다. 주로 서버가 돌아가는 모형과 여러 서브 시스템을 표현한다.

◯ Protocol Design

프론트엔드와 서버 시스템 간 상호작용이 어떤 방식으로 구동될지에 대한 내용을 담는다. 어플리케이션에서 어떤 방식으로 서버와 내용을 주고받을지에 대한 내용을 보여준다. 또한 통신상황에서 동일한 아이디로 인해 혹은 다른 권한으로 인해 발생하는 에러를 방지하기 위해 각 서브시스템이 어떤 양식을 가지는지, URL과 관련된 내용은 어떻게 처리하는지, 에러 시 어떤 메시지를 발생시킬지를 보여준다.

◯ Database / SQL

관계형 데이터베이스 다이어그램과 이를 표현한 SQL을 서술한다. 또한, 각 데이터베이스의 값들의 속성과 자료형을 나타낸다.

◯ 읽기에 앞서

<u>개발문서</u>

실제로 개발을 하면서 작성한 문서가 아니기 때문에 실제로 어플리케이션을 만들 때는 사용하지 못하거나 추가해야 할 함수나 오브젝트가 있을 수 있다.

<u>네트워크와 프로토콜</u>

네트워크와 프로토콜에 대한 지식이 전무하고 실제 개발을 목표로 하고 있지 않기

때문에 정확하고 명확한 설명이 이뤄지기는 쉽지 않을 것이다. 그럼에도 만약 실제로 개발을 한다면 무엇을 사용할지에 충분히 고려하여 작성하고자 하였다.

어플리케이션과의 연결

Python은 데이터 분석 언어의 최강자라 할 수 있지만 반대로 모바일과 웹환경에서는 좋지 못한 지원을 보인다. 어플리케이션과 웹을 주로 하게 되는 소셜커머스를 개발하면서 동시에 데이터분석까지 이뤄져야 하는 목표를 달성하기로 한다. 또한, 웹서버나 프로토콜에 대해서도 지식이 전무하기 때문에 쉬운 툴을 사용하기로 한다.

이런 목표를 달성하기 위해 최대한 쉬운 툴을 사용할 것이다. 그로 인해 실제로 많이 사용되지 않는 방법이 쓰일 수도 있다. 이 점을 감안해주시길 바란다.

2 Introduction

이번 장에서는 본 시스템의 설계에 사용된 다이어그램과 도구를 소개하고, 시스템의 개발 배경이 되는 문제의식과 이 시스템이 해당 문제들을 어떻게 해결할 것인지를 기술한다.

2.1. Applied Diagram

◯ Sequence Diagram (Event Trace Diagram)

Sequence Diagram 은 오브젝트 간 상호작용을 시간순으로 나타낸 도표이다. 사전에 작성한 시나리오를 바탕으로 다이어그램을 작성해 필요한 오브젝트와 기능을 식별하였다.

작성한 Sequence Diagram은 이후 Relation Diagram과 Class Diagram을 작성할 때에도 사용되며, 다른 다이어그램에서 보여주기 힘든 상호작용의 시간 순 진행을 보여주고, 추상적인 하나의 기능을 여러 단계로 나누어 생각할 수 있게 돕는다.

◯ Entity-Relation Diagram (ER Diagram)

Entity-Relation Diagram은 여러 Entity 간의 관계와 각 Entity의 간단한 세부사항을 보여주는 다이어그램으로, 시스템에 어떤 Entity가 필요하며, 여러 Entity들이 어떤 관계를 가질지를 식별하기 위해 작성한다.

Entity는 이전에 작성한 시나리오와 다른 다이어그램을 통해 식별해 나가며 다이어그램에 기술하며, 식별된 Entity는 필요한만큼 세분화하여 작성한다.

◯ Class Diagram

Class Diagram은 시스템의 class, attribute, method와 object 간의 포함관계를 보여주는 다이어그램으로 시스템의 전체적인 구조를 서술하기 위해 작성한다.

가장 상위 Entity인 시스템을 위에 기술하며, 아래로 갈수록 직접적으로 실행되는 각각의 operation들을 기술한다.

2.2. Applied Tool

◯ Diagrams.net (Draw.io)

Diagrams.net은 브라우져 기반 다이어그램 어플리케이션으로, 기본 템플릿과 여러 다이어그램 convention에 부합하는 도형, 기호 등을 제공하며, 자동적으로 다이어그램의 구성요소의 배치를 돕는다.

이 문서에 사용된 다이어그램은 작성의 편의와 형식의 통일을 위해 모두 위의 툴을 사용하여 작성되었다.

2.3. System Overview

본 시스템, '샥'은 '차세대 이커머스 플랫폼'이라는 키워드를 대주제로 삼아 기획되었다. 팀은 회의를 통해 이커머스 분야에서 중고거래 시장이 성장하고 있으며, 특히 모바일 환경에서의 중고거래 시장의 성장이 주목할 만하다고 판단하여 이를 소주제로 삼았다.

조사 결과, 모바일 중고거래 플랫폼들은 몇 가지 차별점을 갖긴 하나, 공통적으로 '신뢰도', '거래 과정의 번거로움' 그리고 이러한 문제를 야기하는 근본적 문제인 '거래 과정과 이에 대한 기존의 사용자 경험'에 있어 개선이 이루어질 필요가 있다고 판단하였으며, 이런 개선이 시장의 Needs를 충족할 수 있을 것이라 판단하였다.

본 시스템은 기존 중고거래 플랫폼의 사용자나 새로운 사용자도 쉽게 적응할 수 있도록 직관적인 사용자 경험을 목표로 기획하였으며, 이를 위해 기존 중고거래 과정에, 본 시스템이 개입하는 방식으로서 'Tinder' 등의 소개팅 어플이 선보인 방식을 도입해 거래 과정에서 직접적으로 여러 매물 혹은 구매자와 연락해야 하는 번거로움을 줄이고, 사용자가 원하는 매물 혹은 구매자에게 접근할 확률을 더욱 높이고자 하였다. 또한, 기존에 존재하는 '호텔스 컴바인', '스카이 스캐너', '다나와' 등의 여러 경쟁 플랫폼의 가격을 한눈에 볼 수 있게 해주는 가격비교 방식과 추가적인 시기별 평균가격 차트를 도입하여, 허위 매물에 대한 자정작용과 공정한 거래를 촉진하고자 하였으며, 이를 통해 시스템의 신뢰도를 높이고자 하였다.
마지막으로 본 시스템의 중심이 되는 위의 두가지 방식을 지원하기위해 개별 상품과 그 옵션, 그리고 중고 매물과 거래 자체의 여러 특징을 카테고리화 하여, 일종의 Tag로 사용하는 방식을 시스템에 도입하였다.

본 시스템의 사용자들은 이를 통해, 기존에는 '여러 구매자의 동시다발적 연락' 이었던 사용자 경험에서, '시스템이 매칭한 구매 확률이 높은 매물/구매자를 한 명 씩 자동으로 연결' 하는 사용자 경험으로 한층 편리하고, 직접 발품을 팔고, 거래를 위해 물품의 세부사항을 또다시 확인해야하는 수고를 덜어낸 중고 거래를 경험할 수 있을 것이며, 이외에도 여러 다른 중고 플랫폼의 물품을 검색해보거나 시기별 가격을 비교하는 행위를 통해 해당 물품에 대한 지식을 빠르게 습득하여 판매자와 구매자 모두 조금 더 합리적이고 안전한 거래를 경험할 수 있을 것으로 기대한다.

3 System Architecture – Overall

이번 장에서는 본 시스템의 전체적인 구조를 설명한다. 이는 각 서브시스템의 세부적인 설명에 들어가기에 앞서 전체적인 흐름을 짚어 이해를 돕기 위한 것이다.

3.1. System Organization

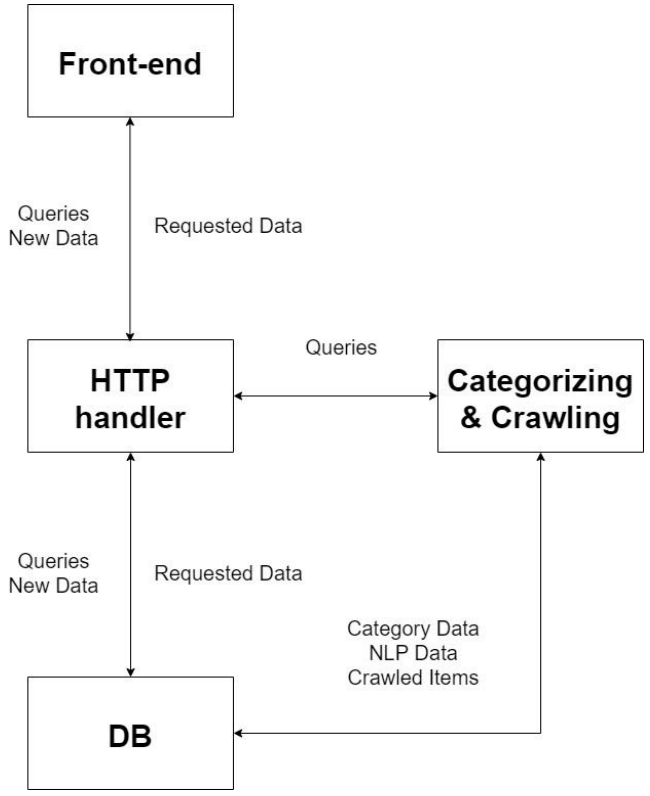

Diagram 1. System Organization

본 시스템은 크게 Front-end와 Back-end (Web-server, DB)로 이루어져 있다. Front-end는 서버에 정보를 입력하거나 받아오는 역할을 수행한다. Web-server는 Application의 서버인데, Handler와 주요 기능에 대한 Controller를 통해 Front-end와 DB의 상호작용을 돕는다. DB는 총 10개로 이루어져 있으며, 이에 포함된 정보들이 시스템의 주요 기능 수행 시 호출되어 사용된다.

3.2. System Architecture – Front-end Application

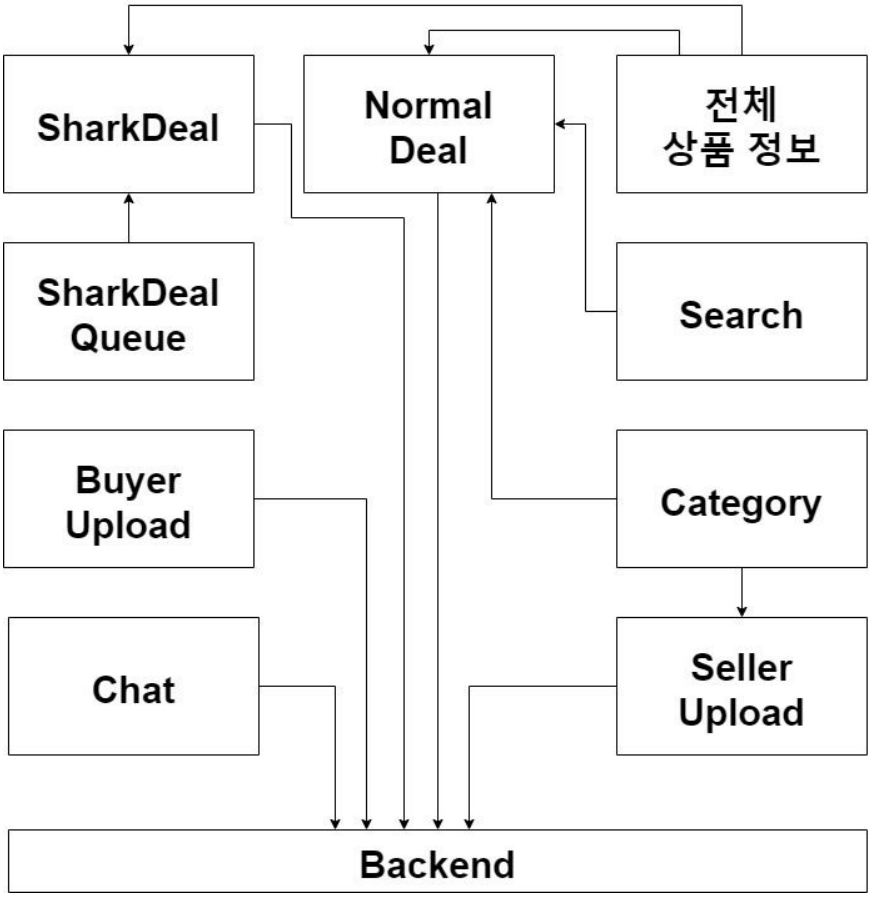

Diagram 2. System Architecture – Front-end

프론트엔드는 서버에 정보를 입력하고 받아오는 방식으로 작동한다. 사용자가 application을 통해서 여러가지 기능을 요청하면 그에 따른 호출이 이뤄지고 호출을 통해 기능이 작동하도록 한다. 서버에서 많은 작업을 수행하면서도, 어플리케이션에 정보를 전달하여 어플리케이션이 직접 기능들을 호출하도록 만들어서 어플리케이션에 높은 자유도를 준다. 이를 위해 개발에 JavaScript, Kotlin, Swift를 사용하며, 최종적으로 HTTP Handler에게 데이터를 전달한다.

3.3. System Architecture – Back-end

◯ System Architecture – Back-end ~ Web-server

크게 client가 볼 수 있는 창인 Front-end 적인 측면과 Application Server, 그리고 이에 관여되는 database로 나누어 정리할 수 있다. Application 내에서는 웹 서비스로 작용하고자 할 경우 HTTP Request Handler가 필요하며 사용자 정보, 채팅과 같은 주요 기능에 대한 Controller가 포함된다. 이와 같이 시스템 내부에서의 Controller, Handler를 정의할 수 있으며 우리 플랫폼의 경우 외부 사이트에서의 가격 정보도 크롤링 해와야 하므로 이를 위한 시스템을 두 개로 정의할 수 있다. 외부 사이트와 우리 플랫폼 내에서 발생한 데이터는 Heroku Web Server와 Hadoop을 이용하여 관리 가능하다.

○ System Architecture – Back-end ~ DB

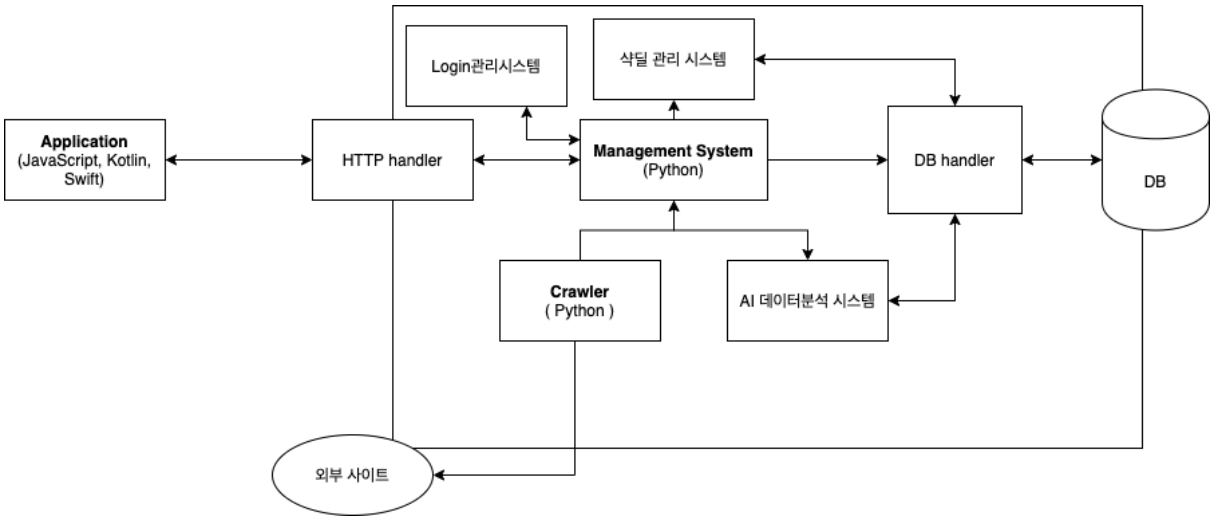

Diagram 3. System Architecture – Back-end

총 9개의 데이터베이스에 포함되어 있는 정보가 시스템의 주요 기능 수행 시 관여한다.

구매자 정보 데이터베이스와 판매자 정보 데이터베이스는 사용자 정보 데이터베이스로부터 조건에 맞는 (구매자인지 판매자인지) 정보를 불러온다. 외부 사이트 크롤링 정보 데이터베이스는 외부 사이트 각각의 특징을 파악하여 그에 맞춘 크롤링 방법에 대한 정보를 포함한다.

구매자와 판매자 정보 데이터베이스에는 모두 사용자에 대한 개인 정보와 과거 거래 이력, 신고 이력, 블랙리스트 포함 여부와 같은 정보를 담는다. 전체 상품 정보는 상품의 현재 상태 (샥딜 대기 중, 거래 완료 등)을 나타내며 이 결과가 Front-end에도 반영이 되도록 한다.

샥딜 큐는 판매자와 그가 올림 품목, 구매 희망자 정보를 담고 있으며 이를 통해 판매자에게 대기열에 있는 구매희망자 한 명씩을 매칭해줄 수 있다.

4 System Architecture - Frontend

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

4.1 Shark deal

◯ Class Diagram

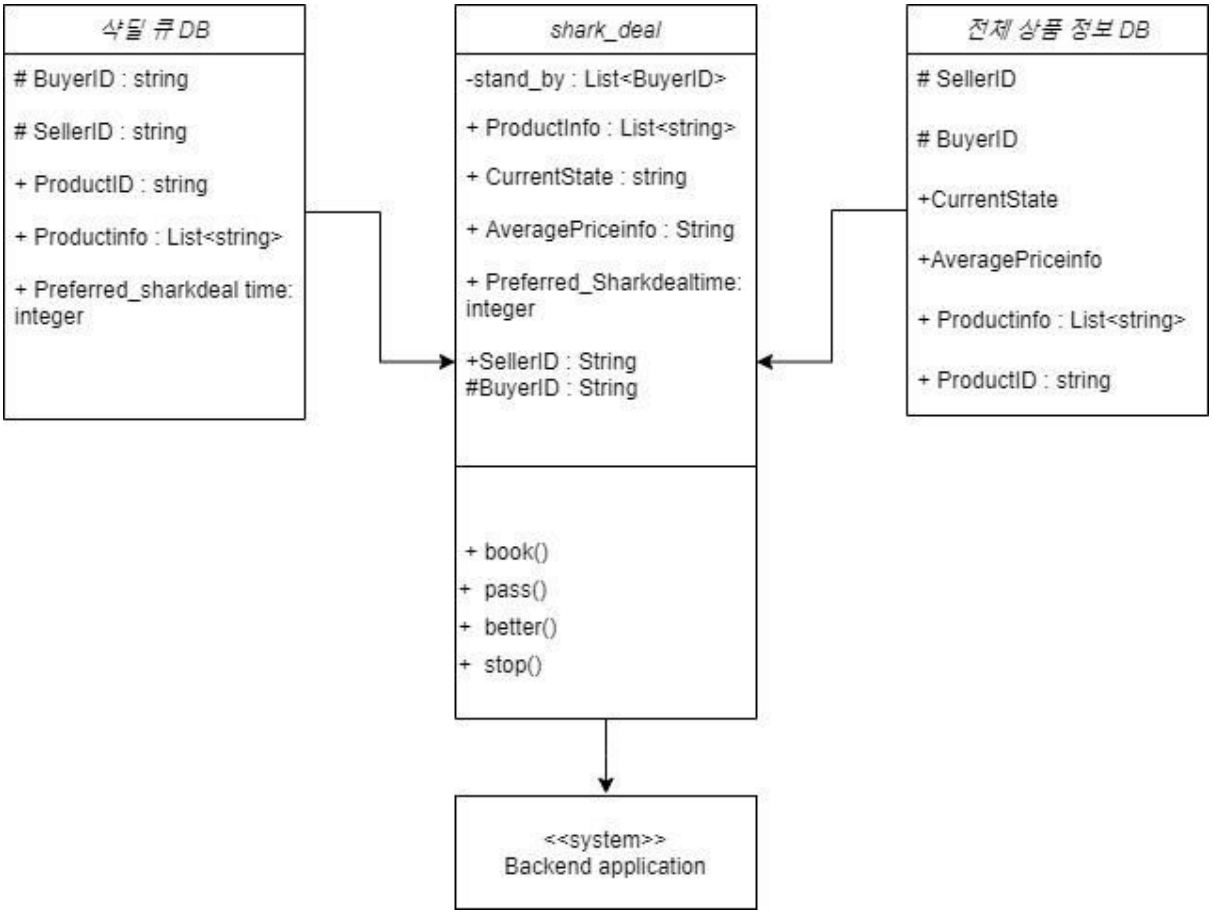

Diagram 4. Shark_deal - Class Diagram

◯ shark_deal - 샥딜 알림 객체

Attribute

stand_by - 물건 구매를 희망한 구매자들의 매칭(판매자와 대화) 대기열

ProductInfo - 물건에 대한 정보

CurrentState - 현재 물건의 거래 상태(ex. 판매완료, 구매희망자와 대화..)

AveragePriceinfo - 크롤링을 통해 얻은 물건의 평균 거래가격

Preferred_Sharkdealtime - 판매자가 설정한 샥딜 제한시간

SellerID - 판매자 트랜잭션 아이디

BuyerID - 구매자 트랜잭션 아이디

Method

book - 샥딜 알림을 통해 알게된 물건에 대한 구매 신청(예약)

pass - 샥딜 알림을 옆으로 넘겨서 현재 물건이 아닌 다른 물건 알림 받을 대기 상태로 전환

better - ProductInfo의 물건 정보들 중에서 마음에 드는 옵션을 선호한다고 피드백 할 수 있다. 클릭하여 표시를 할 경우 글자의 색깔이 바뀌고 구매자의 취향 정보에 영향을 준다.

stop - 제한 시간이 지나면 작동한다. 해당 물건에 대해 거래가 성사된 경우 구매를 희망한 사용자들에게 아쉬움을 전한 메시지가 전달되며 샥딜 관련 DB에서 물건의 정보가 사라진다. 혹은 거래가 성사되지 않은 경우 해당물건은 샥딜 정보에서 사라지며 일반 딜에서 거래가 진행된다.

◯ 샥딜 큐 DB - 샥딜 핵심 정보 객체

Attribute

BuyerID - 구매자 트랜잭션 아이디

SellerID - 판매자 트랜잭션 아이디

ProductID - 물건 트랜잭션 이름

Productinfo -물건 정보

Preferred_sharkdeal time - 판매자가 설정한 샥딜 제한시간

◯ 전체 상품 정보 DB

Attribute

ProductInfo - 물건에 대한 정보

CurrentState - 현재 물건의 거래 상태(ex. 판매완료, 구매희망자와 대화..)

AveragePriceinfo - 크롤링을 통해 얻은 물건의 평균 거래가격

4.2 Normal_deal

○ Class Diagram

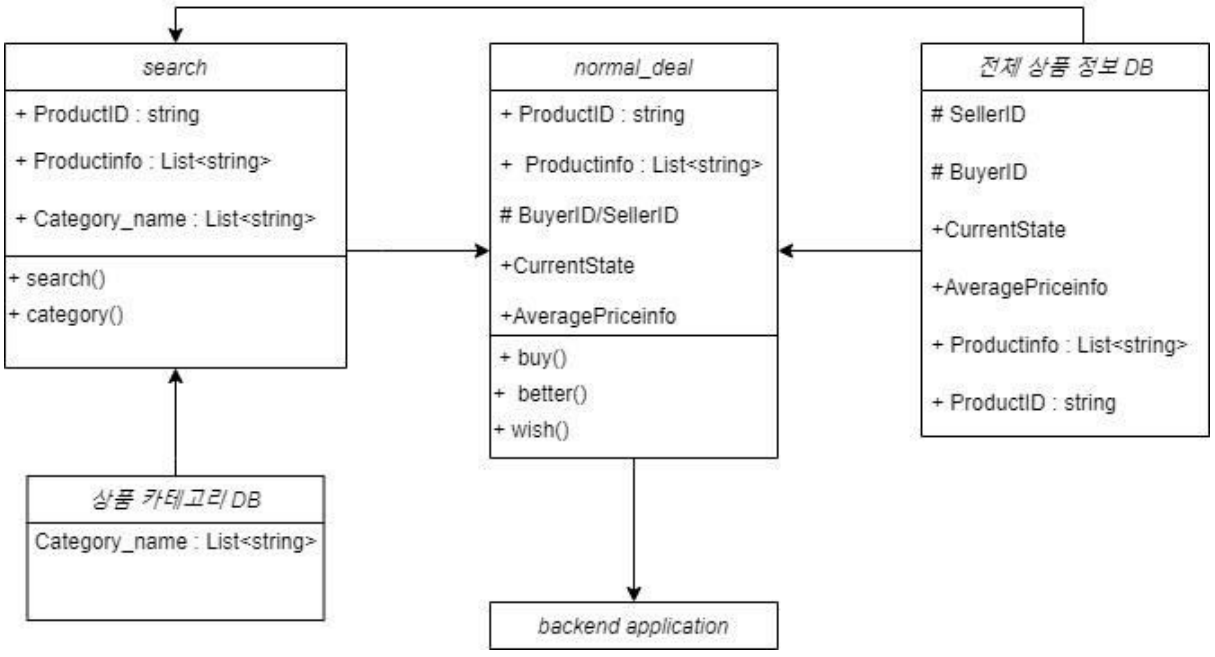

Diagram 5. Normal_deal Class Diagram

○ normal_deal - 일반 딜 객체

Attribute

ProductInfo - 물건에 대한 정보

CurrentState - 현재 물건의 거래 상태(ex. 판매완료, 구매희망자와 대화..)

AveragePriceinfo - 크롤링을 통해 얻은 물건의 평균 거래가격

SellerID - 판매자 트랜잭션 아이디

BuyerID - 구매자 트랜잭션 아이디

Method

buy - 구매자는 구매 의향을 판매자에게 전달한다.

better - ProductInfo의 물건 정보들 중에서 마음에 드는 옵션을 선호한다고 피드백 할 수 있다. 클릭하여 표시를 할 경우 글자의 색깔이 바뀌고 구매자의 취향 정보에 영향을 준다

wish - 물건을 장바구니에 담으면 취향 정보에 영향을 주고, 이후 물건을 찾기가 편하다

◯ search - 일반 딜 하위클래스인 검색 객체

Attribute

ProductID - 물건 이름

Productinfo - 물건 정보

Category_name - 검색에 도움을 주는 카테고리이다

Method

Search - 검색어를 입력하면 물건 이름과 물건 정보와 관련된 사항들이 출력된다

category - 카테고리를 설정하고 검색하면 더 상세히 물건을 찾을 수 있다

◯ 상품 카테고리 DB - 검색의 하위클래스(생략)

4.3. Chat

◯ Class Diagram

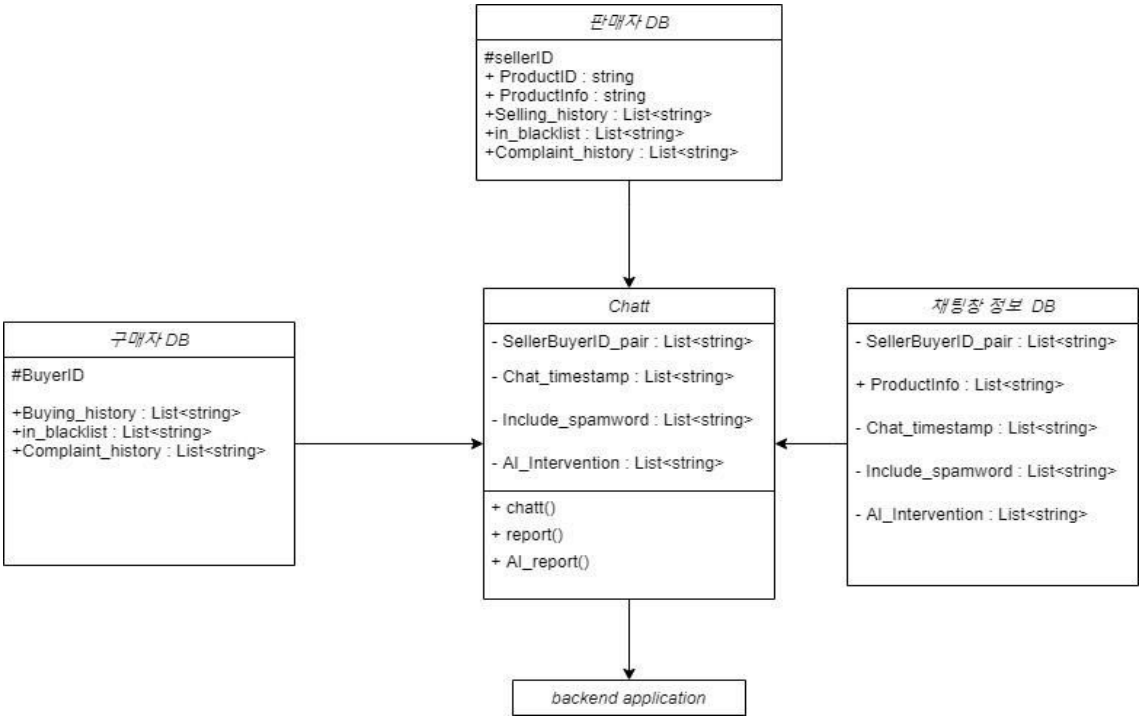

Diagram 6. Chat Class Diagram

◯ Chat - 채팅 객체

Attribute

SellerBuyerID_pair - 판매자와 구매자가 짝지어진 리스트

Chat_timestamp - 시간별로 채팅한 기록

Include_spamword - 잘못된 단어를 사용하는 사용자를 잡기위한 단어 리스트

AI_Intervention - 이상행동을 자동적으로 잡기위한 AI

Method

Chat - 대화참여자는 서로 말을 전달할 수 있다

report - 비정상적인 대화를 하는 상대방은 신고할 수 있다

◯ 채팅창 정보 DB - 채팅 관련 정보를 가지고 있는 DB (생략)

◯ 판매자 DB/ 구매자 DB

Attribute

in_blacklist - 블랙리스트에 오른 사용자는 채팅을 이용할 수 없다

Complaint_history - 불만이 많이 누적될 시에 블랙리스트에 기록된다

4.4. seller_Upload

◯ Class Diagram

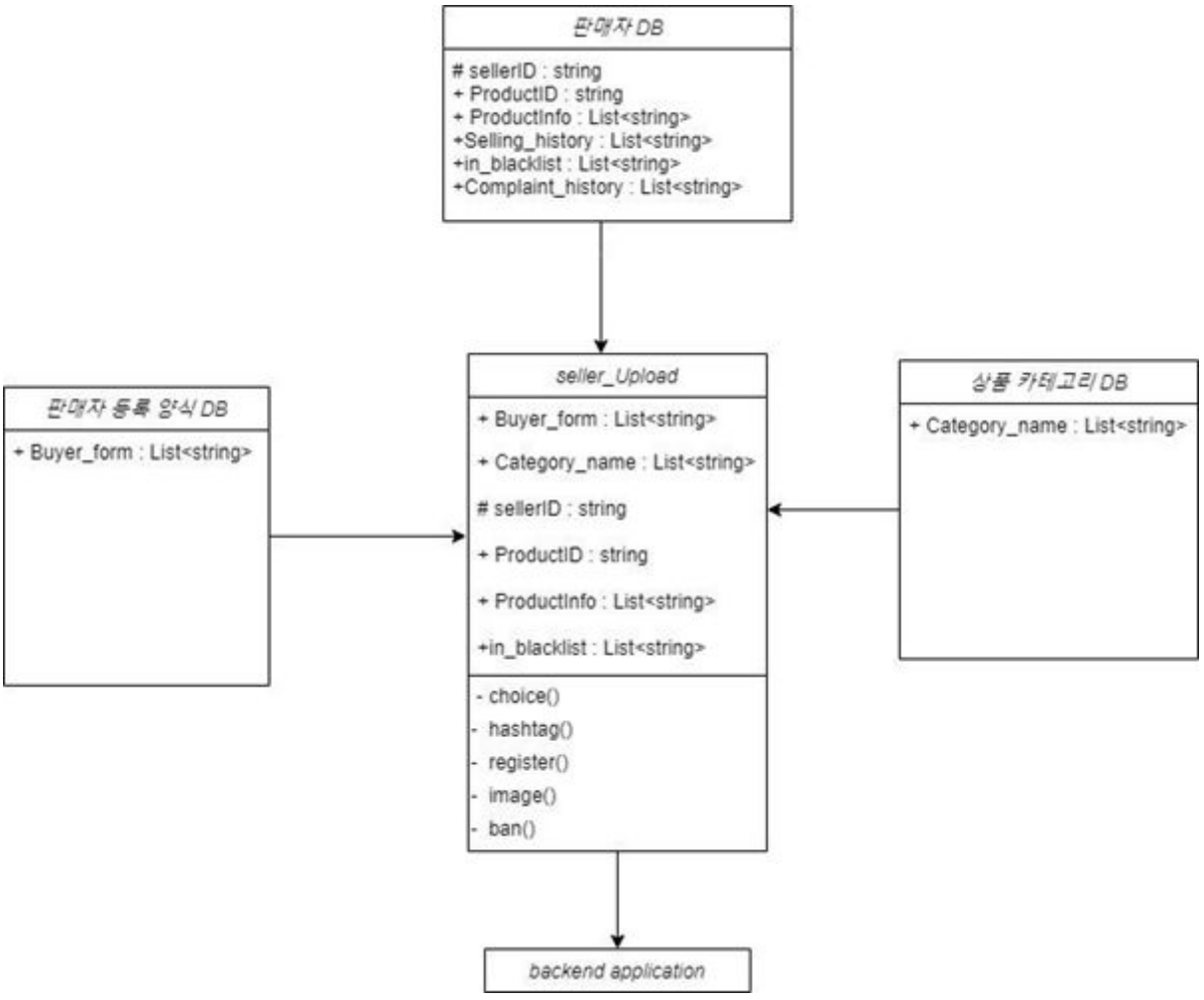

Diagram 7. seller_Upload Class Diagram

○ seller_Upload - 판매자 물건 등록 객체

Attribute

seller_form - 물건을 등록할 때의 양식으로, 미리 만들어져있고 판매자가 남은 부분을 등록하면 된다

Category_name - 물건에 따라 카테고리가 정해진다. 카테고리는 클릭해서 선택한다.

ProductInfo - 기존에 올린 정보에 수정을 할 수도 있다

in_blacklist - 블랙리스트에 오른 사용자는 물건 등록을 할 수 없다

Method

choice - 카테고리를 선택하면 해당 카테고리에 대해 자신이 입력하고 전에 선택한 카테고리와 관련된 것들이 다음 카테고리에 나타난다. (ex. 핸드폰 → 브랜드→ 기종)

hashtag - 해시태그를 통해서 검색이 용이하게 만들 수 있다

register -샥딜이나 일반딜에 등록할 수 있다

image - 이미지를 따로 가져와서 등록할 수 있다

ban - 블랙리스트에 등록된 회원은 등록할 수 없다

4.5. Buyer_Upload

○ Class diagram

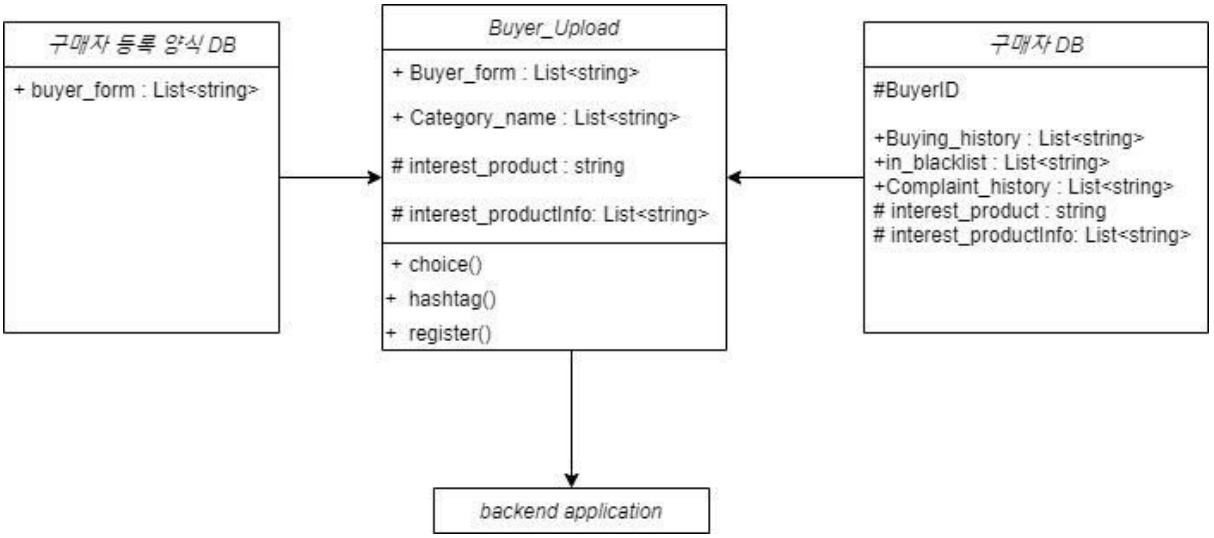

Diagram 8. Buyer_Upload Class Diagram

○ Buyer_Upload - 구매자 위시리스트 조건 등록 객체

Attribute

Buyer_form - 위시리스트의 기본적인 틀은 만들어져 있다. 선택할 수 있는 여러 카테고리가 있고 카테고리 선택과 입력에 따라서 위시리스트가 만들어진다

Category_name - 물건에 따라서 카테고리가 정해진다. 카테고리는 클릭해서 선택한다

interest_product - 구매자가 관심을 가지는 물건이다. 이 정보도 샥딜에 영향을 준다

interest_productInfo - 관심있는 물건에 대한 정보이다. 이 정보도 샥딜에 영향을 준다

Method

choice - 카테고리를 선택하면 해당 카테고리에 대해 자신이 입력하고 전에 선택한 카테고리와 관련된 것들이 다음 카테고리에 나타난다 (ex. 핸드폰 → 브랜드→ 기종)

hashtag - 해시태그를 통해서 자신이 원하는 특별한 조건을 입력할 수 있다

register - 등록을 통해 기존 위시리스트에서 바꾸어 적용할 수 있다

5 Back-end Architecture

5.1. Objectives

사용자와 직접적인 접촉을 하는 프론트 엔드를 제외한 백엔드에 대한 내용을 담는다.
백엔드 시스템과 각 서브 시스템들이 연결된 구조를 보여준다.
주로 서버가 돌아가는 모형과 여러 서브 시스템을 표현한다.

5.2. Overall Architecture

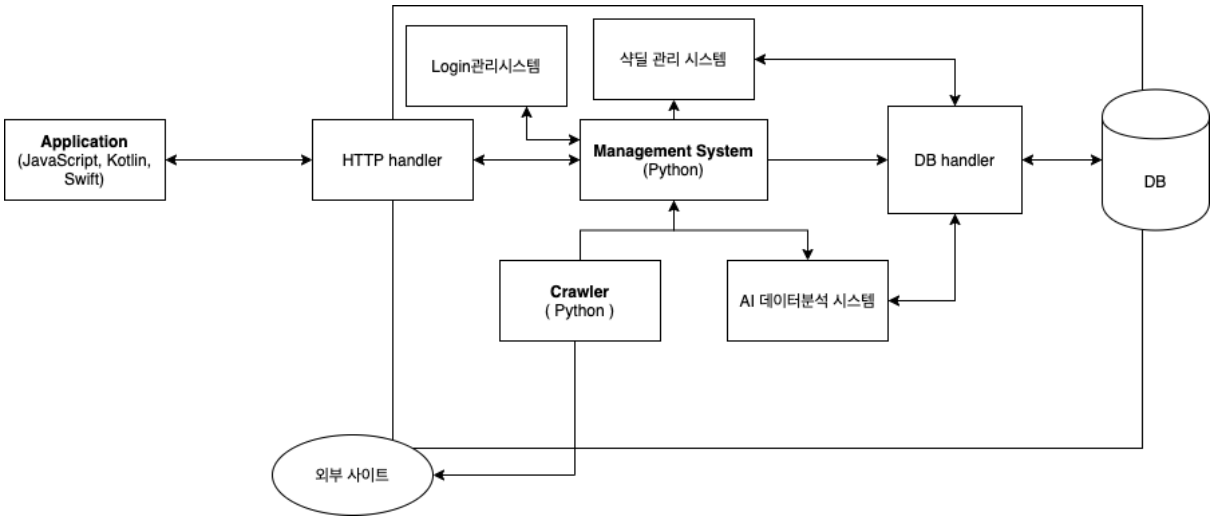

Diagram 9. Overall Back-end Architecture

개괄적인 백엔드의 구조이다. HTTP핸들러를 통해 시스템을 총괄하는 시스템에서 어플리케이션이 정보를 보내고 받을 수 있도록 설계하였다. 매니지먼트 시스템을 통해 각 크롤러와 AI분석 시스템으로 DB에 어떤 내용을 사용하고 어디에 기록해야 하는지 장소가 주어진다.

DB에 기록하는 부분은 주어진 장소를 활용하여 직접 기록하는 방식으로 한다.
크롤러의 경우 정보를 모아서 DB에 주기적으로 저장하는 역할을 한다. 상시적으로 크롤러가 작동하면 페이지를 로딩하는데 많은 시간이 소비되기 때문에 주기적으로 데이터베이스에 저장하는 역할을 담당한다.

AI데이터분석 시스템의 경우 미리 충분하게 모델을 학습하고 저장한다. 학습한 모델을 바탕으로 매니지먼트 시스템이 특정 데이터 분석이나 예측을 요청했을 때, 데이터로부터 자료를 로드하여 보여준다. 시간이 중요한 작업의 경우 사용자가 사용하고 있지 않을 때, 미리 데이터를 분석하여 사용자에게 미리 제공할 정보를 DB에 저장해둔다.

프로그램의 핵심 내용인 샥딜의 경우 샥딜을 관리하는 시스템을 통해 소비자들끼리의 샥딜 매칭을 진행한다.

5.3. Application Server

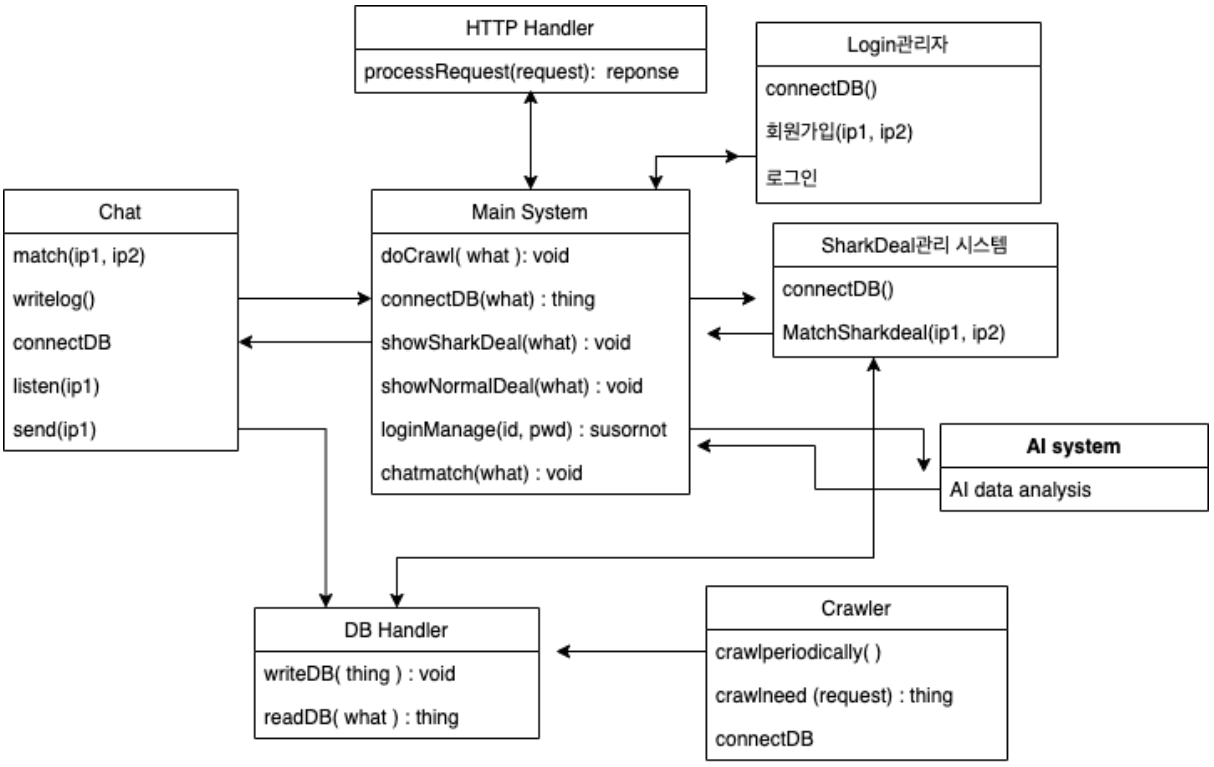

Diagram 10. Application Server

◯ HTTP Handler

processRequest(request): response → 어플리케이션에서 요청한 행동이나 정보, 입력을 처리하는 것이다. 들어온 정보를 처리해서 메인시스템에게 요청을 전달하여 적절한 기능을 수행할 수 있도록 한다

◯ Main System

doCrawl(what) : void → 특정 외부사이트의 내용이 크롤링되어야 할 때, 크롤러에게 해당 작업이 필요한 시점이전에 데이터베이스에 저장해 두도록 설정하는 작업이다

connectDB(what): thing → DB Handler와 연결을 하는 작업이다. DB핸들러에게 저장하도록 요청하면 DB핸들러가 저장하고 반대로 특정 정보를 읽어오라고 하였을 때, 읽어오는 작업까지 수행한다

showSharkDeal(what): void → 샥딜이 매칭되면 샥딜을 보여주기 위해 필요한 정보를 http핸들러에게 전달하는 역할을 한다

showNormalDeal(what): void → 일반딜에 대한 정보요청이 들어왔을 때, 데이터베이스에서 읽어와서 http핸들러로 정보를 넘긴다

loginManage(id, pwd): bool → 로그인이 성공했는지 확인하고, 아니라면 회원가입과 다른 계정에 따른 다른 권한을 부여하는 역할을 담당한다

chatmatch(what): void → 채팅을 하는 사람들의 ip를 채팅 오브젝트로 넘긴다

◯ Chat

match(ip1, ip2) → ip1, ip2 를 받아서 소켓연결을 하고 채팅기능을 시작하도록 한다

writelog(ip1, ip2, contents) → 두개의 아이피와 내용을 로그에 기록한다

connectDB() → 기록된 로그를 디비에 기록한다

listen(ip) → ip에서 내용이 들어올 때까지 기다리다 listen을 통해 소켓을 받는다

send(ip) → ip를 통해 내용을 전송한다

◯ DB Handler

writeDB() → DB에 내용을 기록한다

readDB() → DB에서 특정 조건에 따른 내용을 읽어온다

◯ Crawler

Crawlperiodically → 필요할 때 크롤링하면 시간이 오래걸리기 때문에 중간중간 DB에 필요한 기록들을 만들고 분석해놓는다.

crawlneed → 미리 크롤링하지 못한 데이터들을 요청이 있을 때, 추가적으로 크롤링한다.

connectDB → db에 크롤링한 것들을 저장한다.

◯ SharkDeal관리시스템

connectDB → db에 크롤링한 것들을 저장한다.

MatchSharkDeal → 샥딜의 물건과 인물에 대한 리스트를 만들고 매칭한다.

◯ Login관리자

connectDB → db에 크롤링한 것들을 저장한다.

회원가입 → 회원가입을 지원한다.

로그인 → 로그인을 기록하고 접근권한을 부여한다.

5.4. AI Analysis System

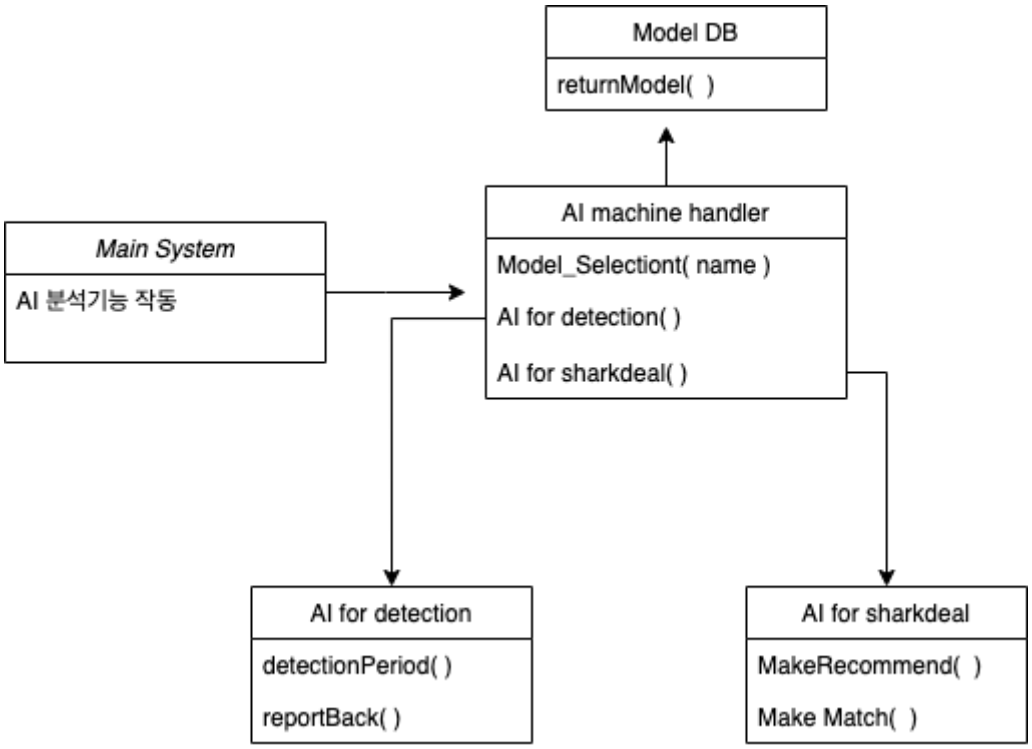

Diagram 11. AI Analysis System

◯ AI Machine Handler

필요한 기능이 들어오면 핸들러가 판단하여 각각의 서브클래스인 분야별 AI에 모델과 정보를 주고 리턴값을 다시 반송한다.

Model_Selection(name) → model 필요한 모델을 호출하는 기능이다.

AI for detection( ) → 주기적으로 일정부분의 데이터를 분석하거나 특정 신고가 들어왔을 때, AI로 이상거래 감지

AI for Sharkdeal( ) → 상품매칭과 추천 상품등을 개인의 선호에 맞춰서 추천해주는 기능을 호출한다.

◯ AI for Detection(info): info

들어온 정보를 분석하여 이상 거래인지, 이상 게시글인지를 파악하여 이상이 있는 듯한 정보를 관리자에게 전달하여 판단하도록 한다.

detectionPeriod( ) → 주기적으로 들어온 정보를 순회하며 학습한 모델을 기반으로 이상 거래를 탐지한다.

reportBack( ) → 이상한 거래를 감지하였으면 해당 거래 정보와 인물정보를 리턴한다.

◯ AI for Sharkdeal

상품을 추천하거나 유관상품을 찾아주는 알고리즘을 통해 특정 인물에게 어떤 상품이 추천되어야 하는지를 찾고 샥딜을 매칭한다.

MakeRecommend( ) → 특정인물에게 어떤 상품이 맞는지 찾는다

MakeMatch( ) → 샥딜로 올라온 상품과 해당 상품을 구매하고자 하는 사람들의 목록에서 순위를 정하고 매칭하는 기능을 담당한다.

◯ Model DB

모델의 저장소이다. 매번 학습을 시킬 수 없기 때문에 여러 분야의 상품과 유저, 채팅, 게시글 등의 다른 정보를 가지고 학습한 여러 개의 모델을 저장해 놓고 handler가 특정 모델을 요구할 때, 필요한 모델을 내보내 주는 역할을 하는 데이터베이스 관리기다.

5.5. SharkDeal

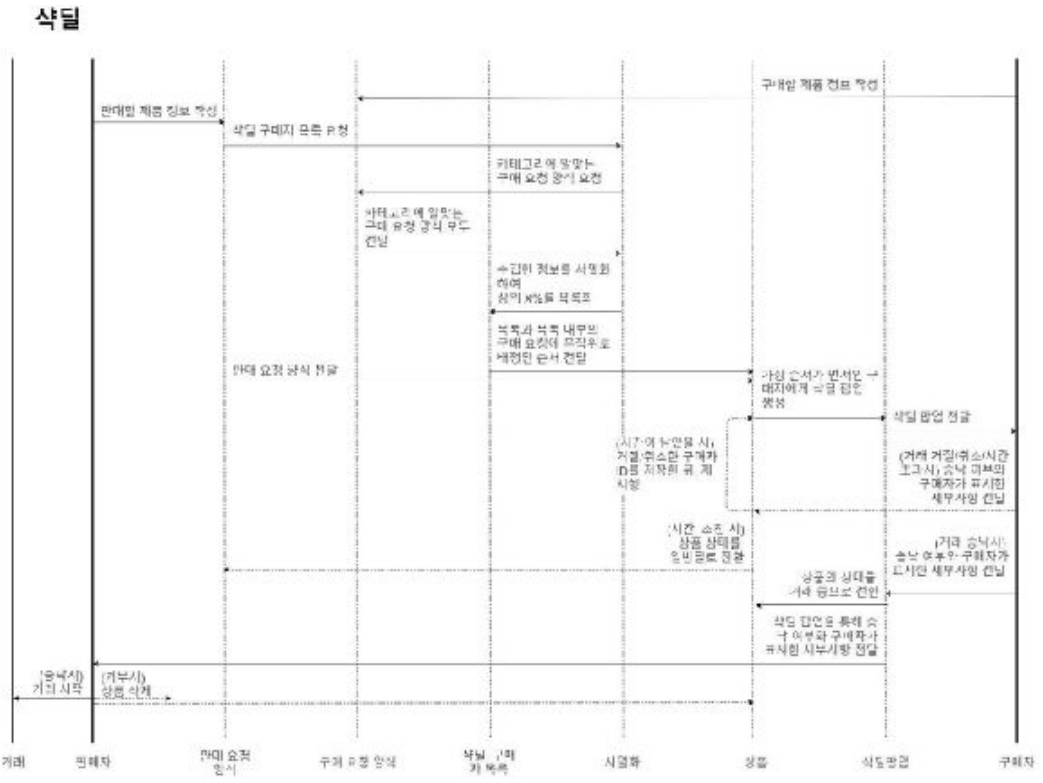

Diagram 12. SharkDeal Sequence Diagram

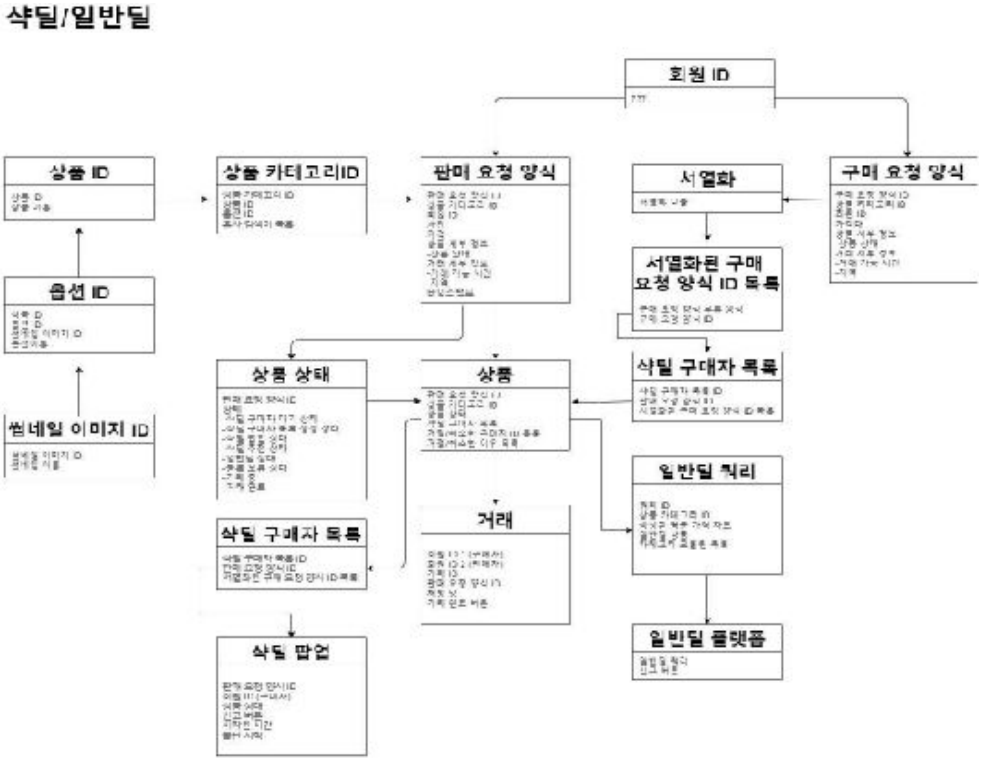

Diagram 13. SharkDeal /NormalDeal Class Diagram

위의 방법을 통해 샥딜을 수행한다. 각각의 과정을 서브클래스로 구성하여 상위 샥딜클래스를 구성한다.

○ SharkDeal 구매자 대기열 및 구매자-판매자 매칭 알고리즘

동일 품목을 구매하고자 희망하는 사용자들 중 판매자가 등록한 판매 조건에 부합하는 사용자들을 선별하고, 과거 거래이력 등의 조건에 따라 이들을 정렬하는 알고리즘

<u>구매자 선별</u>
구매자와 판매자는 관심 있는 물건을 위시리스트에 저장할 때/상품을 등록할 때 직거래여부, 가격, 성능, 물품의 외관, 4가지 기준에 따라 그 중요도를 매긴다.

직거래여부의 경우에는 0,1로 직거래 희망하지 않음/직거래 희망함으로 분류한다.
가격, 성능, 물품 외관의 중요도를 매길 때는 그 값들이 2차원 배열의 형태로 저장이 되는데, 구매자의 측면에서는 행의 값들이, 위시리스트에 저장한 물품들에 대해 매긴 선호도이며 열의 값들은 실제 해당 구매자의 과거 사용내역을 토대로 산출해낸 값들이다.

판매자 측면에서도 유사한 작동 원리로, 행/열 값들이 지정이 되는데 이들 중 행의 값은 구매자들이 해당 품목 옵션에 대해 평가한 결과에 따라 (ex. 가격이 너무 비싸요) 그 값이 수시로 변화할 수 있다.

다시 구매자의 측면으로 돌아가서, 만약 구매자가 2순위 중요도로 선정한 기준에 대해 1순위 중요도로 선정한 기준보다 실제 더 중요하게 생각하고 있음을 확인한다면 (앱 내의 사용 기록 전반을 토대로 판단), 해당 기준에 대한 열값을 0으로 초기화하고 기존의 행값에서 1을 내린 값을 저장한다. 가장 낮은 중요도로 지정한 것의 경우에는 그 행값에 1을 더한다.

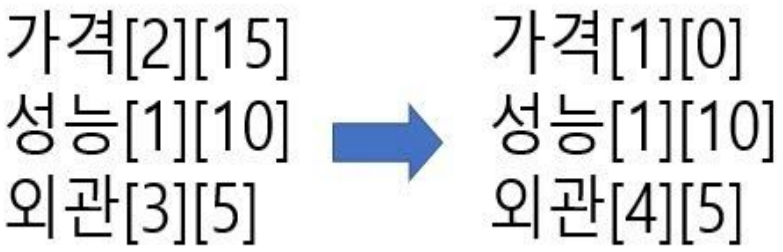

Figure 1. Column Value Change in Buyer Selection

여기에서 정렬된 구매자의 선호도 행 값과 판매자가 지정한 (또는 다른 구매자들로부터 수정된) 선호도의 행 값을 비교하여 이와 일치하는 구매자만 선별하여 대기열에 넣는다.

구매자 대기열 순서 정하기

우선, 직거래 희망 여부 같은 1차원 배열의 기준을 이용해 다시 구매자 선별 작업을 거친다. 그리고, 다시 동일품목 이전 거래 횟수, 거래 성실도, 판매자와의 거리, 카테고리 일치도를 가중치 기준으로 삼아 이를 토대로 구매자 대기열 내의 순서를 정한다.

동일품목에 대한 거래 내역을 고려하는 이유는 무분별한 사재기를 예방하기 위함이며, 선별 과정에서 사용한 카테고리 일치도를 다시 고려하여 공정하면서도 탄력적인 시스템 운용이 가능하도록 한다.

예를 들어, 위 4가지 기준에 대한 가중치를 다음과 같이 정한다면, W=[-10,0.5,-1,5], 그리고 특정 구매자 A가 해당 기준들에 대해 다음과 같은 값을 지니고 있다면, [2,80,25,3] 해당 사용자의 점수는 10이다. 이와 동일한 방식으로 모든 구매자의 점수를 매기고 점수가 높은 사람이 대기열 내에서 먼저 거래할 수 있도록 그 순서를 정렬한다.

6 Protocol

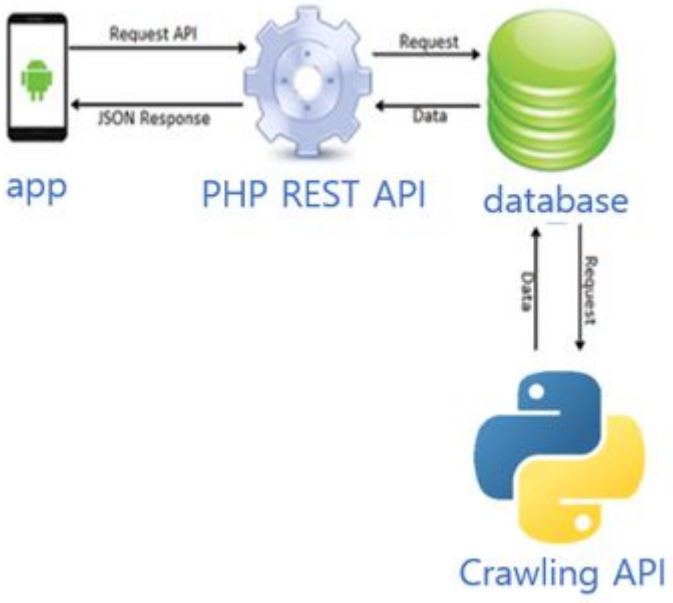

Figure 2. Overall Protocol Structure

프론트 앤드와 백앤드를 연결시켜주는 프로토콜은 기본적으로 HTTP를 통해 이루어진다.

APP 내의 HttpsURLConnection 클라이언트를 활용하는데, 여기에는 app에서 네트워크 통신에 필요한 TLS, IPv6 그리고 연결 풀링을 지원한다.

6.1. Rest API

Representational State Transfer API의 약자로, URL을 통해 데이터를 주고받는 간편한 네트워킹 방식이다. 사이트 내에 세션과 캐시같은 별도의 정보를 저장할 필요가 없기 때문에 정보의 저장과 호출이 자유로운 편이다.

크게 자원의 표현(resource representation)과 상태 전달 단계로 구분되며, 자원의 표현은 자료의 이름을 통해 구분되고, 상태 전달은 JSON 또는 XML을 통해 이루어진다. 앱과 데이터베이스를 연동하는 경우 주로 JSON이 활용된다.

METHOD	역할
POST	POST를 통해 해당 URI를 요청하면 리소스를 생성합니다.
GET	GET를 통해 해당 리소스를 조회합니다. 리소스를 조회하고 해당 도큐먼트에 대한 자세한 정보를 가져온다.
PUT	PUT를 통해 해당 리소스를 수정합니다.
DELETE	DELETE를 통해 리소스를 삭제합니다.

Table 1. Methods in Rest API

6.2. HTTP

HTTP(Hypertext Transfer Protocol)는 인터넷상에서 데이터를 주고 받기 위한 서버/클라이언트 모델을 따르는 프로토콜이다.

애플리케이션 레벨의 프로토콜로 TCP/IP위에서 작동하며, 하이퍼 텍스트 기반으로 데이터를 전송한다. HTTP : Rest api에서 HTTP URI를 통해 자원을 명시하고, 프론트 앤드와 백앤드에서 이루어지는 모든 data request와 return 또한 HTTP를 통해 이루어지기에, 해당 시스템에서 HTTP는 매우 중요한 역할을 담당하고 있다.

HTTP Method : 자료의 상태를 조작하는 method로 POST, GET, PUT, DELETE 총 4가지를 포함한다. 규격화된 정보의 형태와 조작 방식을 가지기 때문에 나타낼 수 있는 정보의 복잡성이 한정되는 단점과, 서버 자원의 쉬운 관리와 활용이라는 장점을 동시에 가진다.

6.3. JSON

◯ JSON
JavaScript Object Notation의 약자로, Name/ Value의 쌍 형태로 이루어진 경량의 DATA-교환 형식이다. 데이터 형태가 C나 JAVA에서 사용되는 텍스트 형식과 유사하기에 쉽게 활용할 수 있으며 언어로부터 독립적이기 때문에 다양한 언어로 개발되는 시스템에서 유용하다. 본 시스템은 app의 경우 JAVA와 Swift를 통해 개발되며, 웹 크롤링을 수행하는 API는 Python library를 활용하기 때문에 전반적인 데이터 통신을 언어와 독립적인 JSON을 통해 하는 것이 중요하다.

6.4. Detail

User
Method Post / Get / Put/ Delete
URI /users/
Parameters Code number
Identification
Password
Register date
Credit grade
Name
Birth date
Item
Method Post / Get / Put /Delete
URI /item/
Parameters Item number
Category
Options
Price

Post date
Shark deal time

7 Database Design

7.1. Objectives

Requirement specification에서 작성된 내용을 근거로 database design을 기술한다. 전체 entity들 간의 관계를 표현하기 위해 ER Diagram을 그려주었고, 이를 통해 relational schema, SQL DDL 명세를 작성한다.

7.2. ER Diagram

본 시스템에는 Seller, Buyer, WishList, Category, Product, Chat, Complaint, SharkDeal, PriceCrawling, 총 9개의 Entity가 존재한다. 각각의 Entity는 직사각형의 형태로 표현되고, Entity간 관계는 마름모꼴로 표현된다. 이때 특정 Entity가 다른 Entity와 가지는 1:N, N:N을 관계를 표현해주었고 각 Entity의 전체 참여/부분 참여 여부에 따라 이중선, 단일선으로 표현해주었다. 각 Entity가 가지는 Attribute은 타원형으로 표현되는데, 각 Entity를 식별하는 Primary Key는 그 라벨에 밑줄을 그어 표시하고, 여러 Attribute을 허용하는 경우에는 테두리를 이중으로 긋는다.

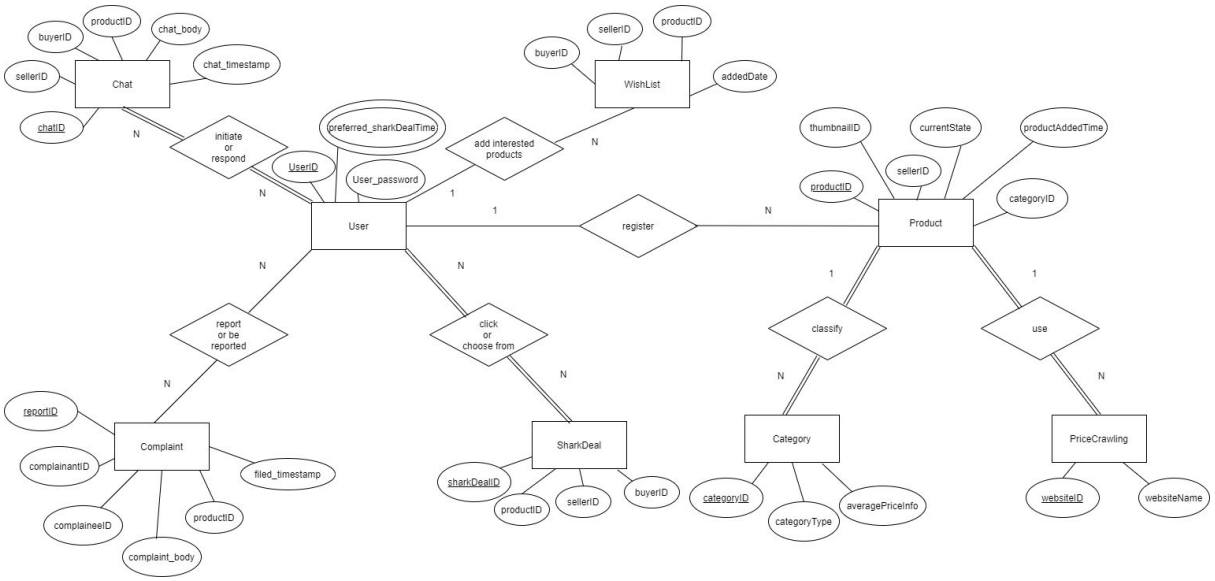

Diagram 14. ER Diagram

◯ Entity - User

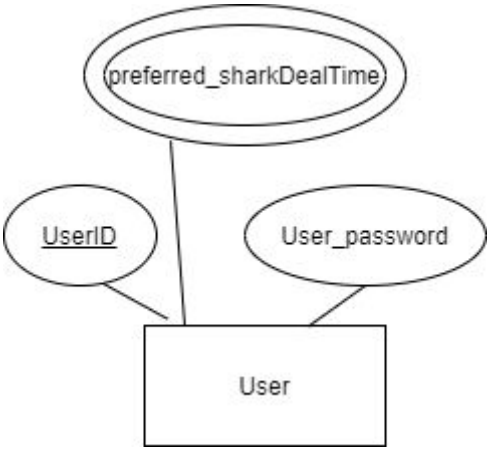

Diagram 15. User Entity

User Entity는 구매자, 판매자를 아우르는 사용자 전체에 대한 정보를 담고 있으며, UserID를 primary key로 갖고 있다. User_password, preferred_sharkDealTime을 기타 attribute으로 지닌다.

◯ Entity - WishList

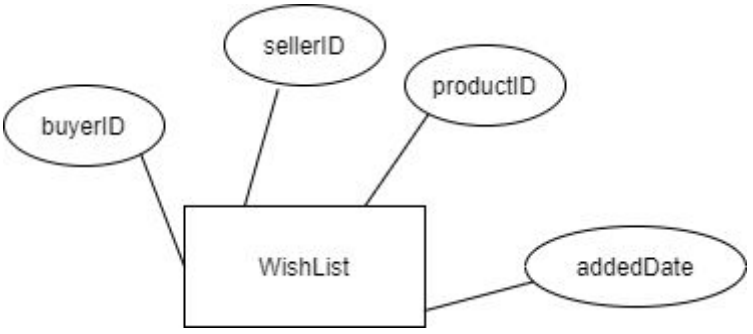

Diagram 16. WishList Entity

WishList Entity는 사용자 중 구매자의 구매희망목록을 담은 데이터베이스로, primary key는 존재하지 않으며, User entity와 Product Entity의 primary key를 조합한 composite key로 WishList를 가진다. 기타 속성으로는 addedDate가 있다.

○ Entity - Product

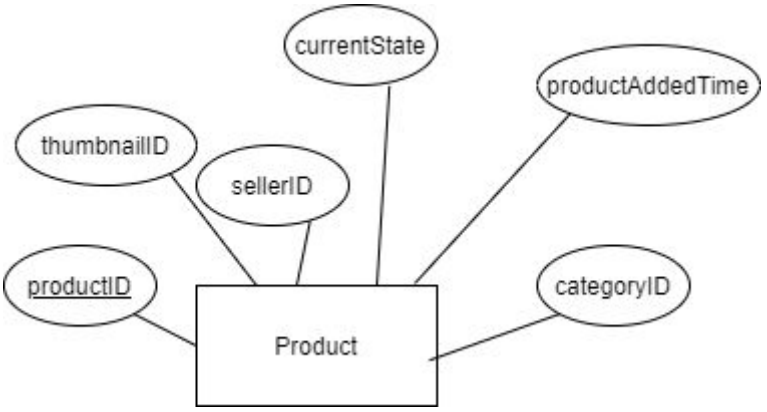

Diagram 17. Product Entity

Product Entity는 판매자가 등록한 상품에 대한 정보가 저장된다. Primary key는 productID이며, 이외에 품목 사진에 대한 아이디인 thumbnailID, 판매자 sellerID, 품목을 조건으로 분류하는 categoryID, 품목 등록 시간인 productAddedTime,
판매중/샥딜 대기중 등과 같은 물품 상태에 대한 정보인 currentState 정보가 포함된다.

○ Entity - Chat

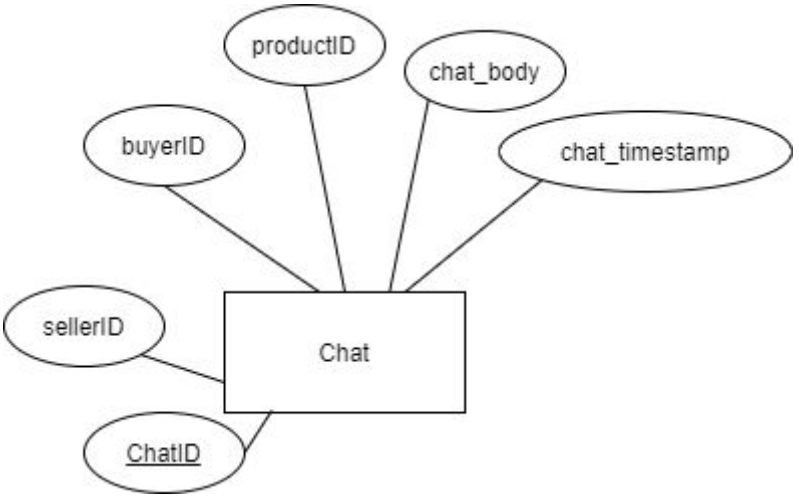

Diagram 18. Chat Entity

Chat Entity는 판매자와 구매자가 매칭되었을 때 일어나는 채팅에 대한 정보를 담고 있다. Primary key는 ChatID이며, sellerID, buyerID, productID 등의 정보가 저장되며 채팅 내용과 시간과 같은 정보를 역시 포함한다.

○ Entity - SharkDeal

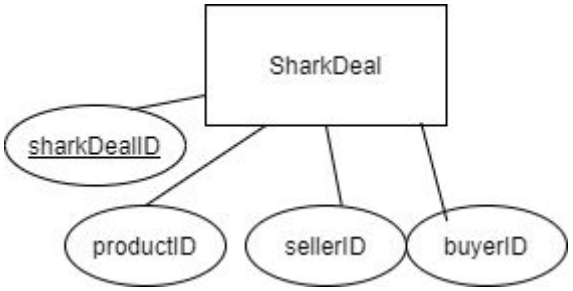

Diagram 19. SharkDeal Entity

SharkDeal Entity는 샥딜과 관련된 정보를 담고 있는 데이터베이스로, sharkDealID를 primary key로 보유하고 있으며 이외에 구매자정보, 판매자정보, 품목정보를 담고 있다.

○ Entity - Complaint

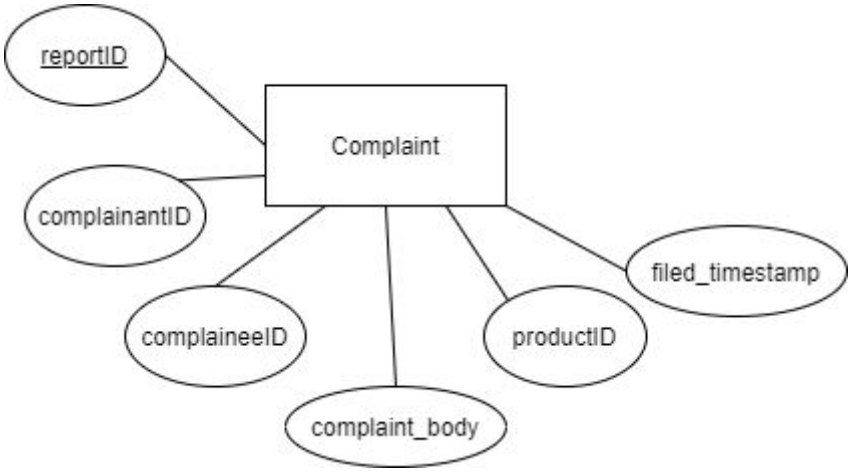

Diagram 20. Complaint Entity

Complaint Entity는 해당 신고 내용 고유의 ID를 primary key로 삼으며 신고자, 신고 대상자,품목, 신고 시간 및 내용을 기타 attribute으로 삼는다.

○ Entity - Category

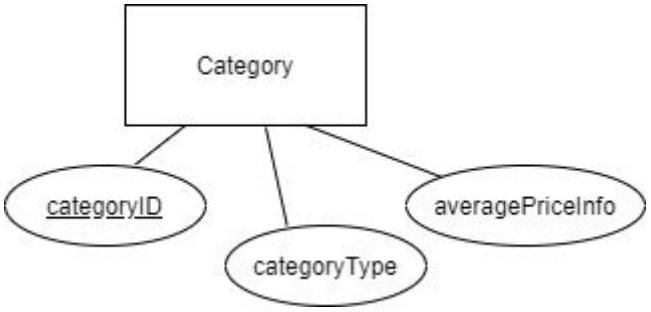

Diagram 21. Category Entity

Category Entity는 판매자가 품목을 등록할 때 저장되는 정보이며, 구매자가 해당 품목을 검색할 때 사용할 수 있는 키워드이다. 각 카테고리는 고유의 아이디값인 primary key를 가진다. 해당 카테고리 이름과 크롤링해온 품목의 평균 시세가 여기에 저장된다.

○ Entity - PriceCrawling

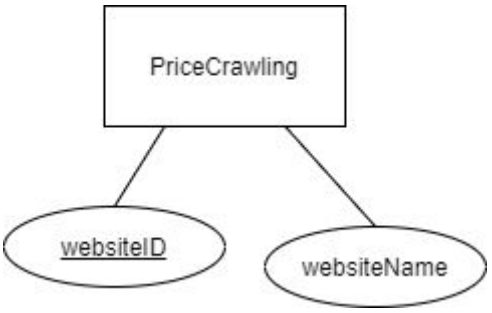

Diagram 22. PriceCrawing Entity

PriceCrawling Entity는 타 중고거래 사이트에 있는 동일 품목에 대한 평균 시세 가격 추이에 이용된다. Primary key는 타 중고거래 사이트 고유의 아이디 값이며 기타 attribute로 해당 웹사이트의 이름을 보유한다.

7.3. Relational Schema

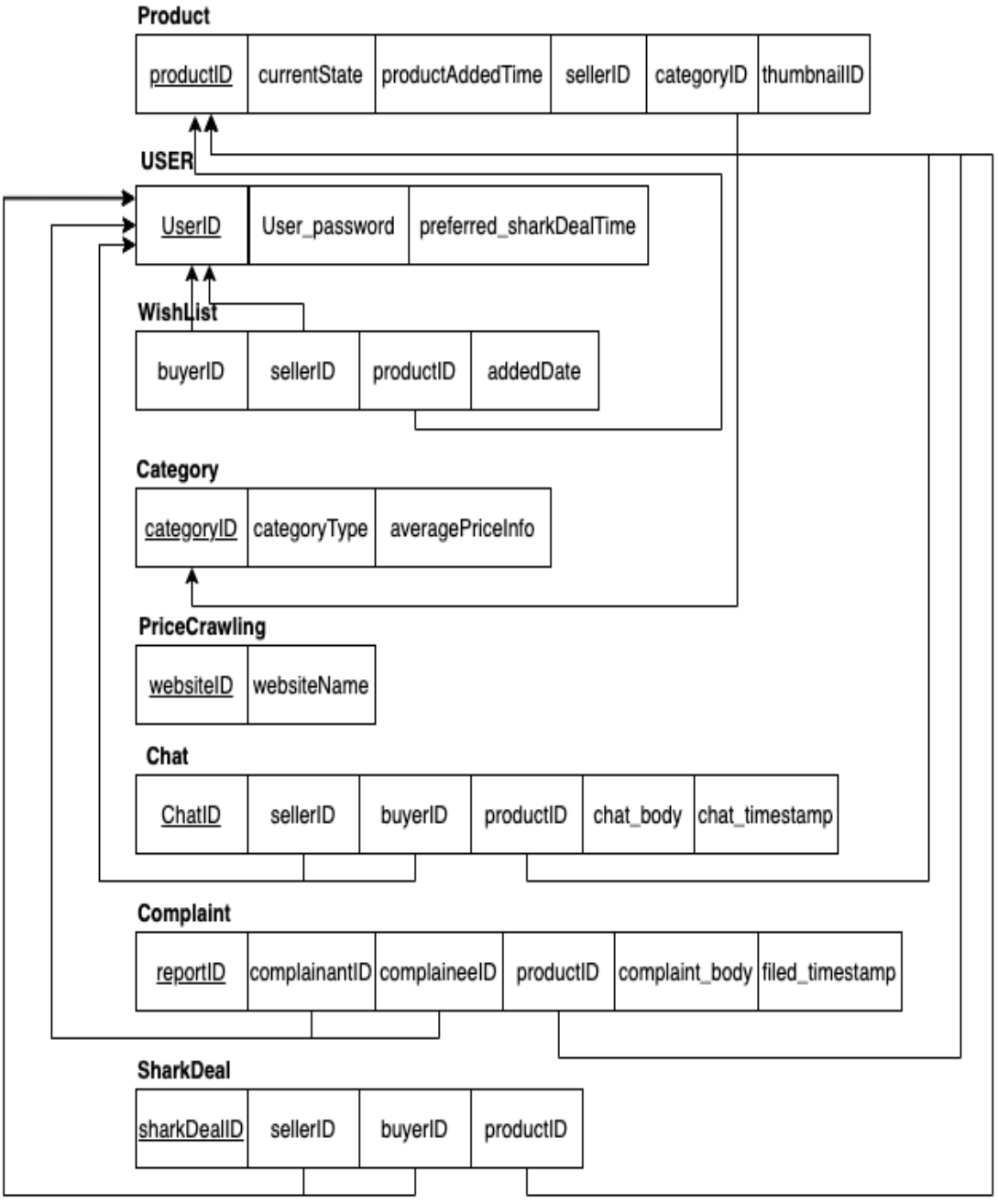

Diagram 23. Relational Schema

7.4. SQL DDL

○ Product

```
CREATE TABLE Product
(

        currentState VARCHAR(10) NOT NULL,
        productAddedTime TIMESTAMP WITH TIME ZONE NOT NULL,
        sellerID VARCHAR(20) NOT NULL,
        thumbnailID INT NOT NULL,
        PRIMARY KEY (productID),
        FOREIGN KEY (categoryID) REFERENCES Category(categoryID),

);
```

○ User

```
CREATE TABLE User
(

        User_password PASSWORD NOT NULL,
        preferred_sharkDealTime INTERVAL HOUR TO MINUTE NOT NULL,
        PRIMARY KEY (UserID)

);
```

○ WishList

```
CREATE TABLE WishList
(

        addedDate TIMESTAMP WITH TIME ZONE NOT NULL,
        PRIMARY KEY (buyerID, sellerID, productID),
        FOREIGN KEY (buyerID) REFERENCES User(UserID),
        FOREIGN KEY (sellerID) REFERENCES User(UserID),
        FOREIGN KEY (productID) REFERENCES Product(productID)

);
```

○ Category

```
CREATE TABLE Category
(

        categoryType VARCHAR(50) NOT NULL,
        averagePriceInfo INT NOT NULL,
        PRIMARY KEY (categoryID)

);
```

○ PriceCrawling

```
CREATE TABLE PriceCrawling
(

        websiteName CHAR NOT NULL,
        PRIMARY KEY (websiteID)

);
```

○ Chat

```
CREATE TABLE Chat
(

        chat_body VARCHAR(200) NOT NULL,
        chat_timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
        PRIMARY KEY (ChatID),
        FOREIGN KEY (sellerID) REFERENCES User(UserID),
        FOREIGN KEY (buyerID) REFERENCES User(UserID),
        FOREIGN KEY (productID) REFERENCES Product(productID)

);
```

○ Complaint

```
CREATE TABLE Complaint
(

        complaint_body VARCHAR(200) NOT NULL,
        filed_timestamp TIMESTAMP WITH TIME ZONE NOT NULL,
        PRIMARY KEY (reportID),
        FOREIGN KEY (complainantID) REFERENCES User(UserID),
        FOREIGN KEY (complaineeID) REFERENCES User(UserID),
        FOREIGN KEY (productID) REFERENCES Product(productID)

);
```

○ SharkDeal

```
CREATE TABLE SharkDeal
(

      PRIMARY KEY (sharkDealID),
      FOREIGN KEY (sellerID) REFERENCES User(UserID),
      FOREIGN KEY (buyerID) REFERENCES User(UserID),
      FOREIGN KEY (productID) REFERENCES Product(productID),

);
```

8 Index