Requirement Specification

오늘도거래완료

학번	이름
2016311322	옥영민
2016313286	곽위
2015312681	안효준
2017313260	이재민
2015312516	한종운
2015313697	황영준

목차

1. Preface

1.1 Objective

Preface에서는 본 문서의 예상 독자를 설정하고 그에 따른 문서의 구성을 설명하고 문서의 전체 구조와 구조별 설명이 이루어진다.

1.2 Readership

본 문서의 예상 독자는 소프트웨어 개발자이며, 여기에는 소프트웨어 엔지니어, Architecture, Database Manager, 서버 운영자 및 추후 기술 지원을 위한 고객 서비스 팀 등이 이에 해당된다.

1.3 Document Structure

A. Preface

Preface에서는 본 문서의 예상 독자에 대해서 설명하고 전반적인 문서의 구조와 각 부분에 대한 설명이 이루어진다.

B. Introduction

Introduction에서는 시스템의 디자인을 표현하기 위해서 사용된 Diagram의 종류와 이 Diagram들을 그리기 위해 사용한 도구에 대해 설명하고, 본 문서에서 설명하는 시스템이 다루게 될 범위에 대해서 설정한다.'

C. System Architecture - Overall

System Architecture - Overall에서는 시스템의 전체 구조에 대해서 간략한 설명이 이루어지며, Fronted와 Backend의 구조에 대해서도 간단하게 설명한다.

D. System Architecture - Fronted

Fronted의 구조와 이에 해당되는 서비스의 서브 시스템에 대한 설명이 이루어진다.

E. System Architecture - Backend

Backend의 구조와 이에 해당되는 서비스의 서브 시스템에 대한 설명이 이루어진다.

F. Protocol Design

Fronted와 Backend 간의 Interaction부터 시스템의 각 Component 사이에서 발생하는 모든 Interaction을 위한 인터페이스와 프로토콜의 구조가 어떻게 설계되는지 설명하고, 인터페이스와 프로토콜이 어떤 환경과 언어를 활용하여 구현할지 설명한다.

G. DB Design

본 시스템에서 활용되는 데이터를 저장하고 관리하고 활용하기 위한 Database의 구조를 설명하는 단계로 데이터의 속성과 데이터베이스의 스키마, 데이터베이스 간의 관계에 대해서 작성한다.

H. Testing Plan

Verification과 Validation을 위한 Testing Plan을 작성하고 이에 대해서 설명한다.

I. Development Plan

시스템을 개발하는데 사용될 개발 언어와 도구, 라이브러리, 플랫폼 등의 개발 환경에 대해서 설명하고 개발 일정에 대해서 기술한다.

J. Index

Index에는 본 문서를 작성하는데 사용한 표, 다이어그램, 그림 등의 색인이 포함된다.

2. Introduction

2.1 Objective

Introduction에서는 본 문서를 작성하는데 사용된 다양한 Diagram과 이를 그리기 위한 도구들에 대해서 설명하고 시스템의 범위에 대해서 설정한다.

2.2 Applied Diagram

A. UML

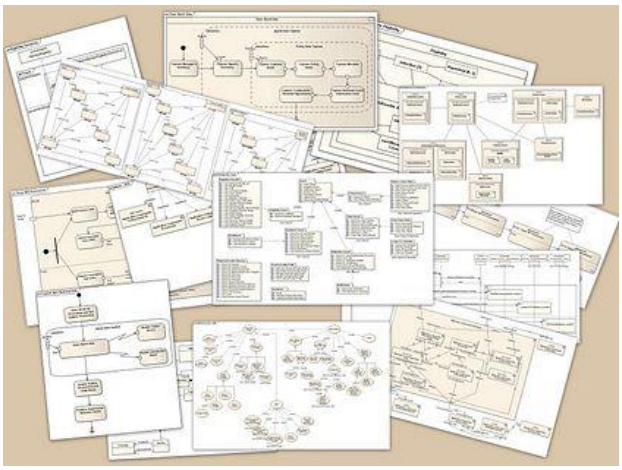

Figure 2.1 Unified Modeling Language

The Unified Modeling Language(UML) is a general purpose and developmental modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

B. Context Diagram

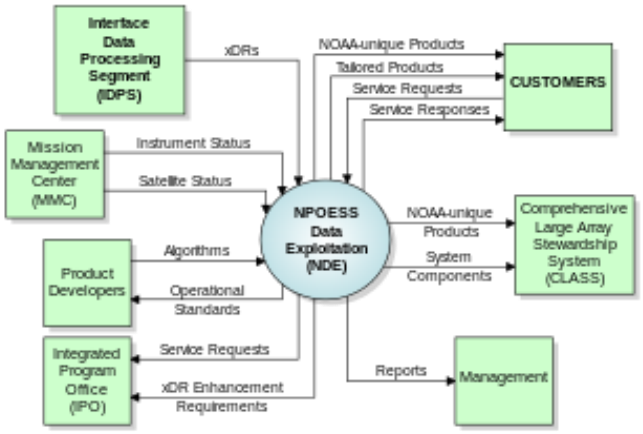

Figure 2.2 Example of Context Diagram

The context diagram is used to establish the context and boundaries of the system to be modelled: which things are inside and outside of the system being modelled, and what is the relationship of the system with these external entities.

C. Use-case Diagram

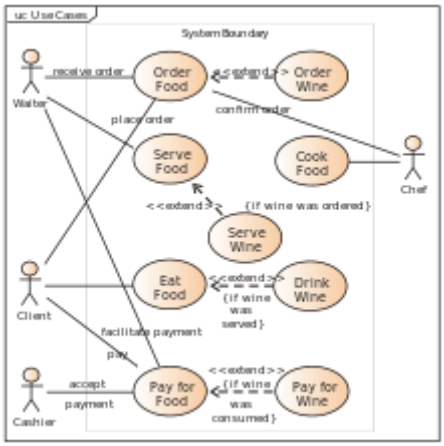

Figure 2.3 Example of Use-case Diagram

Use-case Diagram were developed originally to support requirements elicitation and now incorporated into the UML. Each Use-case represents a

discrete task that involves external interaction with a system. Actors in a Use-case Diagram may be people or other systems.

D. State Diagram

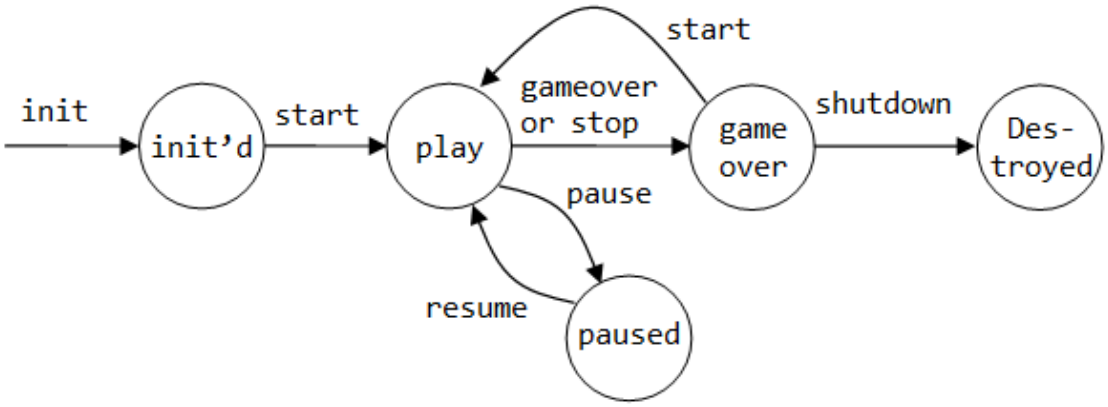

Figure 2.4 Example of State Diagram

A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. This diagram the behaviour of the system in response to external and internal events. And it show the system's responses to stimuli so are often used for modelling real-time systems.

E. Activity Diagram

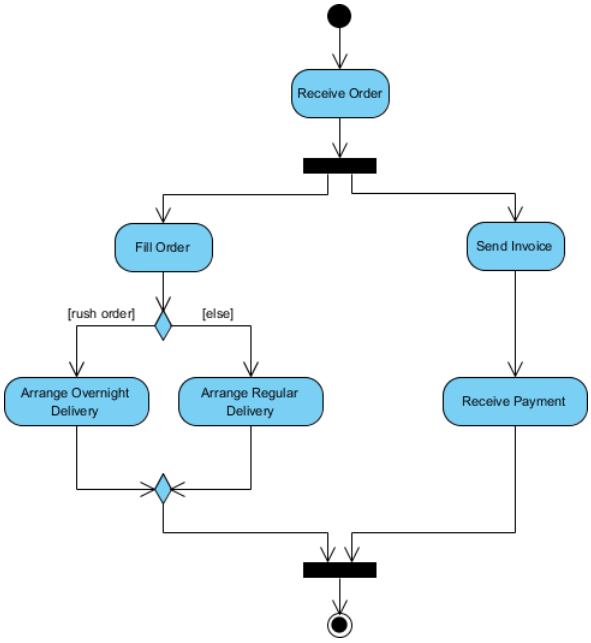

Figure 2.5 Example of Activity Diagram

Activity diagram is used to describe dynamic aspects of the system. We can represent the graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

F. Class Diagram

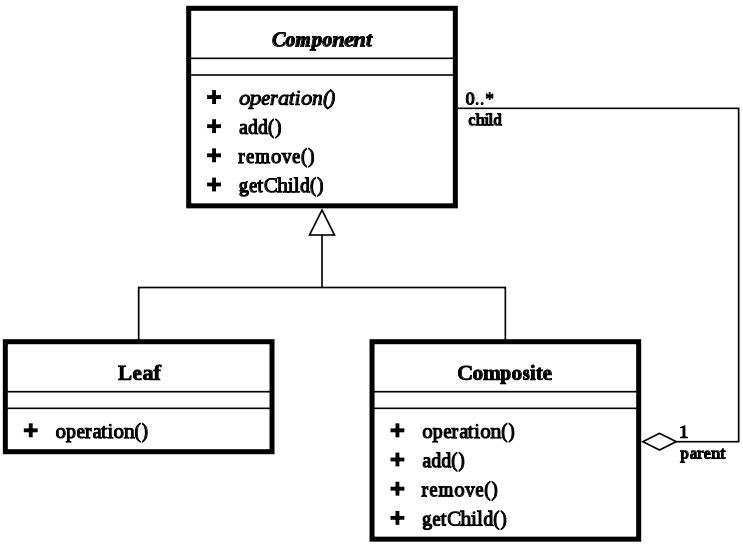

Figure 2.6 Example of Class Diagram

Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes. An object class can be thought of as a general definition of one kind of system object. An association is a link between classes that indicates that there is some relationship between these classes.

2.3 Applied Tool

A. PowerPoint

Figure 2.7 PPT

Microsoft사에서 제공하는 PowerPoint는 PPT를 제작하기 위해 사용되는 프로그램이지만 PPT를 만들기 위해 제공되는 다양한 도형과 지시선들을 사용자가 원하는대로 자세하게 설정할 수 있기 때문에 Diagram을 그리는데에도 유용하게 사용할 수 있다.

B. Draw.io

Figure 2.8 draw.io

Draw.io는 온라인 웹사이트를 통해서 이용할 수 있는 다이어그램을 그리기 위한 프로그램으로서 PowerPoint와 달리 용도가 다이어그램을 그리는 것에 맞춰져 있기 때문에 다이어그램을 만드는데 불필요한 과정들이 없고 세부 설정 또한 다이어그램에 최적화되어 있기 때문에 추가적인 조작 없이 쉽고 빠르게 다이어그램을 만들 수 있다.

2.4 Project Scope

본 시스템은 기존 당근 마켓의 예약금 관련 문제와 매너온도의 낮은 신뢰성으로 발생하는 여러가지 문제점을 해결하기 위해서 구매자와 판매자를 시스템 내에서 확정짓고, 거래 후에 진행되는 평가에 대한 신뢰성을 높여 거래를 활발하게 만드는 것이 본 시스템에서 원하는 개선 방향이다. 따라서 핵심적인 기능은 평가와 신고 기능이며, 해당 기능을 위한 사전 준비와 이후에 발생하는 상황을 위한 서브시스템들에 대하여 설계하였다.

시스템은 크게 사용자로부터 이벤트를 받는 Fronted Architecture와 이벤트와 그에 대한 결과를 처리하는 Backend Architecture로 구성되어 있다. Fronted Architecture를 통해 사용자는 개인 정보 수정부터, 동네 설정, 검색, 카테고리 설정, 물품 등록, 구매, 판매, 판매를 위한 Chatting, 거래에 대한 평가와 신고 등의

중고거래를 위한 활동들을 진행한다. Fronted Architecture에서 발생한 이벤트들은 Web Application을 통해 Backend Architecture로 전달되며, 여기에서 이벤트에 필요한 처리가 이루어진 후, 시스템의 Database에 결과가 반영된다. 이 중에서 평가나 신고와 관련된 사항에 대해서는 우선적으로 운영자에게 전달되며, 내역에 따라 제재가 결정된다. 결정된 내역은 Backend Architecture를 통해 다시 Database의 정보가 수정되고, Fronted Architecture에게도 전달되어 신고자와 대상자에게 제재가 진행되었다는 알림과 대상자에 대한 제재가 진행된다.

3. System Architecture - Overall

3.1 Objective

본 챕터에서는 System Architecture의 시스템의 전체구조와 시스템을 Fronted Architecture와 Backend Architecture의 큰 두 부분으로 나누어 각 컴포넌트에 대해 간단하게 설명한다.

3.2 System Organization

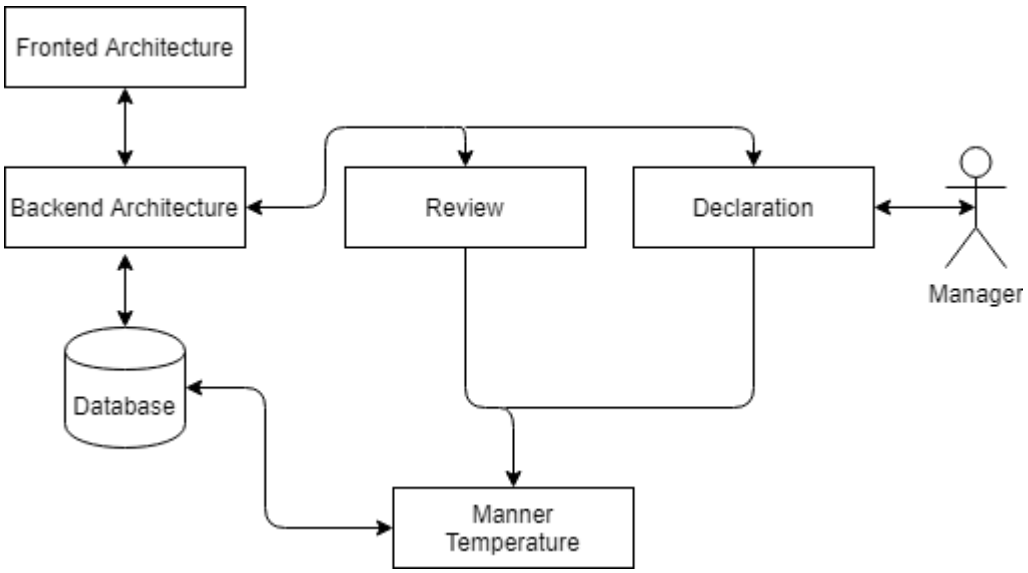

Diagram 3.1 Overall System Organization

사용자는 Fronted Architecture를 통해 시스템과의 상호작용을 하게 된다. Fronted Architecture를 통해 입력받은 이벤트들은 Backend Architecture를 통해 처리되며, 이에 대한 결과는 다시 Fronted Architecture를 통해 사용자에게 전달되거나, Database의 갱신에 이용된다. 이 중 이 시스템의 핵심 기능인 Review와 Declaration 기능은 마찬가지로 Fronted Architecture를 통해 Backend Architecture에서 처리되며, 둘 다 매너온도(Manner Temperature)의 값을 갱신하는데 사용된다. 이 중에서 신고 기능은 Manner Temperature에 반영되기 전에 Manager를 통해 검증을 받게 되고, 이에 따른 결과는 매너 온도 반영과 Backend Architecture를 통해 사용자에게 전달되고, Database의 갱신에 이용된다.

3.3 Fronted Architecture

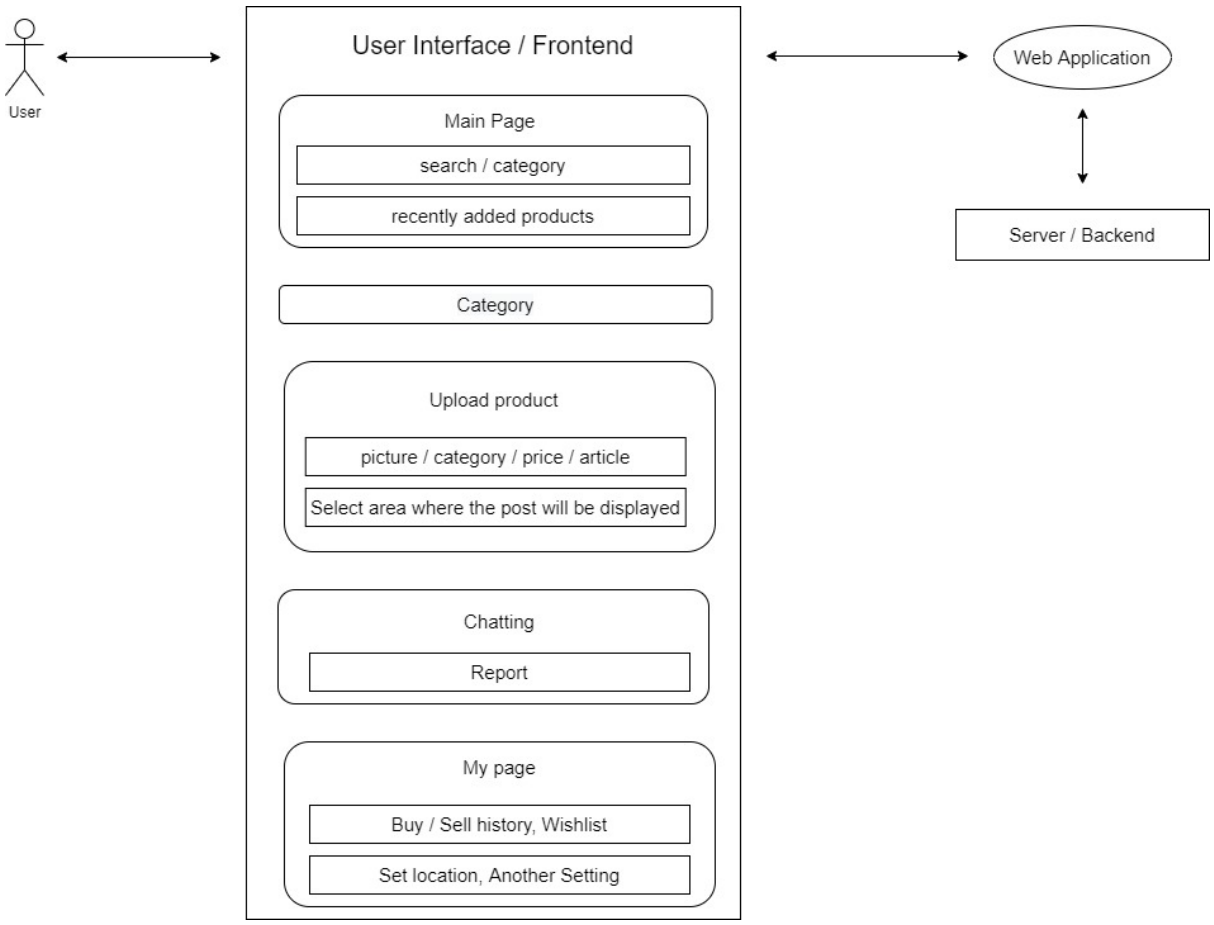

Diagram 3.2 Fronted Architecture

Fronted Architecture는 사용자와의 상호작용을 담당하는 Architecture로 로그인, 로그아웃, 동네 설정 등의 개인정보 수정, 물품 등록, 검색, 거래를 위한 채팅 등의 거래 관련 활동과 거래 완료 후의 평가와 신고 등의 사용자가 이용할 수 있는 시스템의 거의 모든 시스템이 Fronted Architecture를 통해 이루어지며 이 데이터들은 Web Application을 통해 Backend Architecture로 전달되어 처리된다.

3.4 Backend Architecture

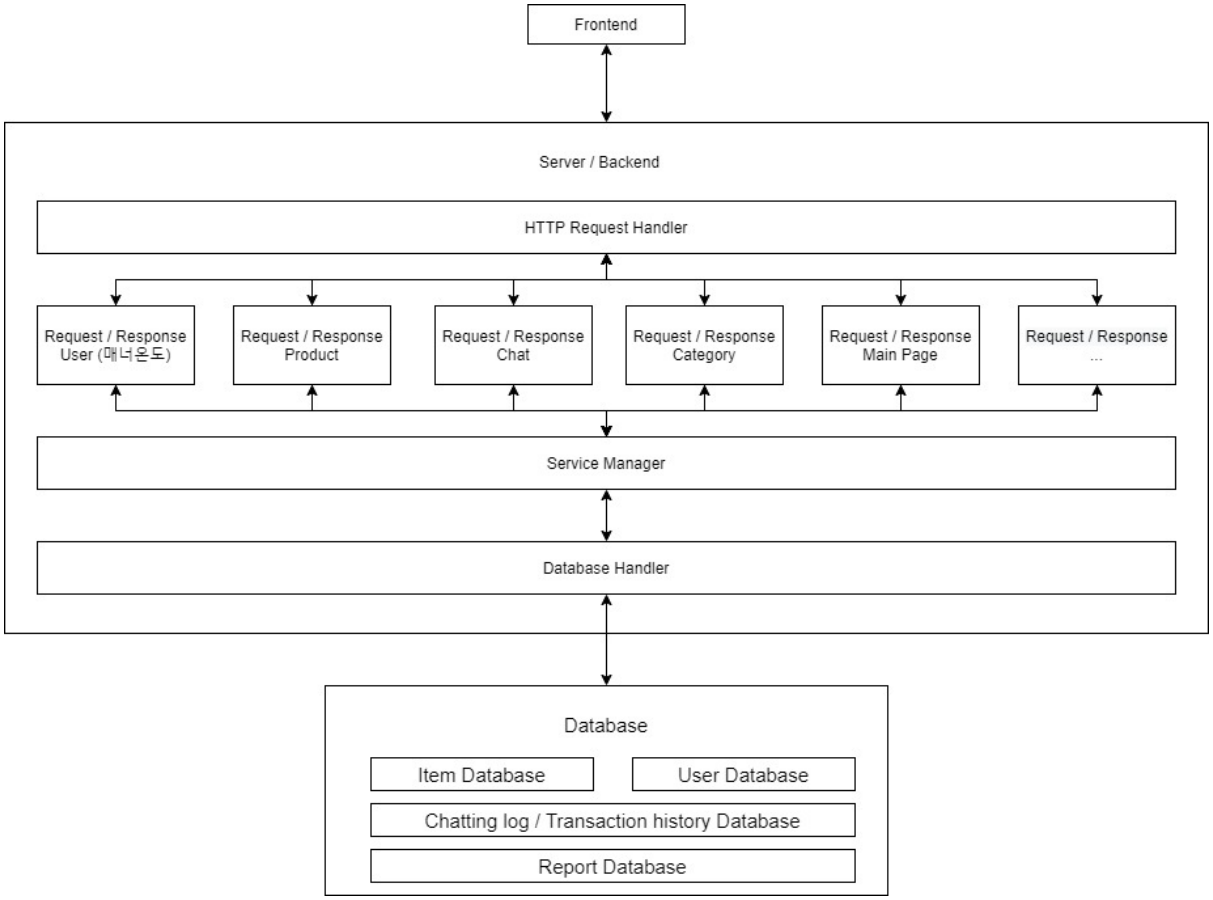

Diagram 3.3 Backend Architecture

앞의 Fronted Architecture로부터 Web Application을 통해 전달되어진 데이터들을 Backend Architecture는 HTTP Request Handler를 통해 받아들인다. Backend로 들어온 데이터들은 데이터의 종류에 따라 알맞은 Event Handler를 통해 처리되며, 이 중 신고와 같은 몇몇 기능의 경우에는 Manager의 검토를 추가로 받게 된다. 시스템의 특성상 대부분의 이벤트들을 데이터베이스의 수정과 연관되어 있으며, 따라서 Backend에서 처리되어진 데이터들은 Database Handler를 통해 Database를 갱신시키게 된다. 갱신이 성공적으로 이루어지면 다시 Backend로 신호가 돌아오고 이는 다시 Web application을 통해 Fronted Architecture를 통해 수정된 사항이 최종적으로 사용자에게 보여지게 된다.

4. System Architecture - Frontend

4.1 Objective

전체 시스템 구조 중 사용자와의 상호작용을 담당하는 Frontend 시스템의 구조, 컴포넌트의 구성과 관계에 대해 서술하는 내용이다.

4.2 Subcomponents

A. Main page

1. class Diagram

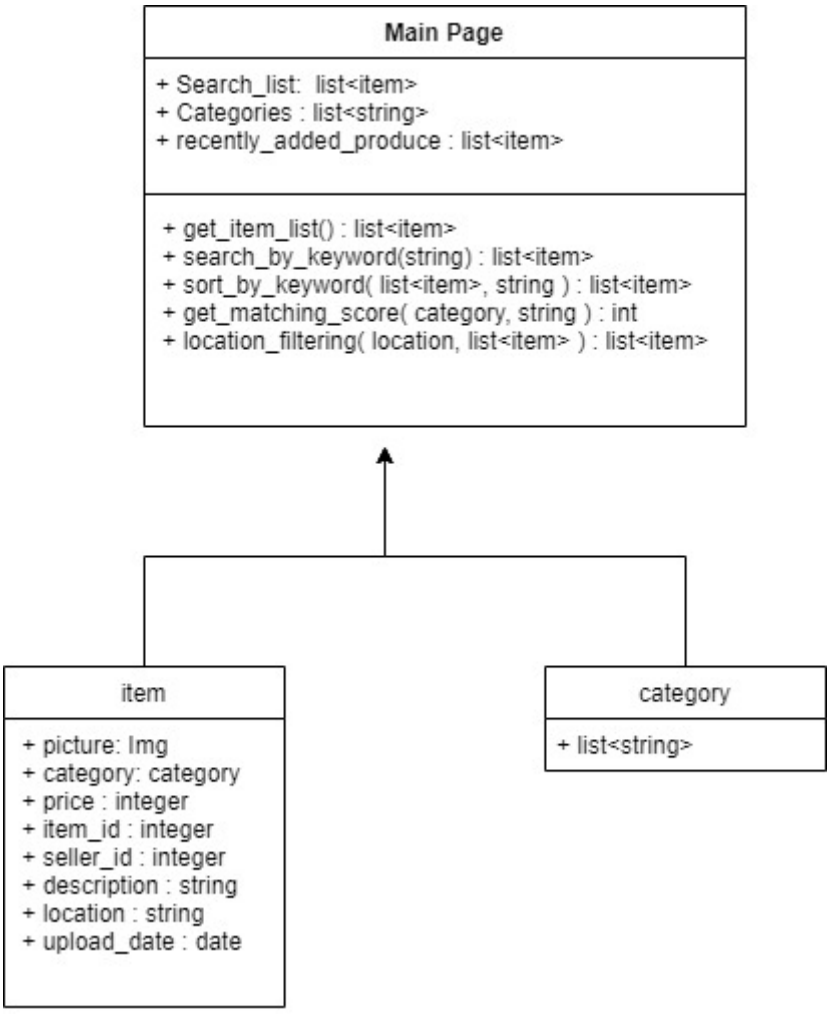

Diagram 4.1 Main Page

1. Main page - 메인 페이지 객체

A. attributes

+ search_list: 전체 item list 중 검색 조건에 맞는 item list 목록
+ Categories: item에 대해 검색 조건이 되는 항목의 모음
+ recently_added_product: 전체 item에 대해 upload 시간 순서로 정렬된 item list

B. methods

+ get_item_list(): 현재 게시판에 upload된 모든 item의 목록을 가져와 list 형태로 반환한다.
+ search_by_keyword(string): item의 list에 대해 주어진 검색어와 유사도를 판단해 유사도가 높은 item 목록을 가져온다.
+ sort_by_keyword(list<item>, string): 주어진 검색어에 대해 유사도가 높은 순서로 item list를 정렬한다.
+ get_matching_score(category,string): 주어진 검색어와 카테고리의 유사도를 계산한다.
+ location_filtering(location, list<item>): item list에서 주어진 위치와 인접한 위치의 item을 조회한다.

2. item - 판매 상품 객체

+ picture: 상품의 이미지 파일
+ category: 상품이 속한 카테고리
+ price: 상품의 가격
+ item_id: 상품이 DB에 등록될 때 생성되는 ID번호
+ seller_id: 상품 판매자의 user_id

+ description: 상품에 대한 판매자의 기술
+ location: 상품 거래를 희망하는 지역
+ upload_date: 상품이 게시판에 등록된 날짜

3. category

+ category: 전체 상품의 카테고리를 저장해 놓은 배열

B. Upload product

1. Class Diagram

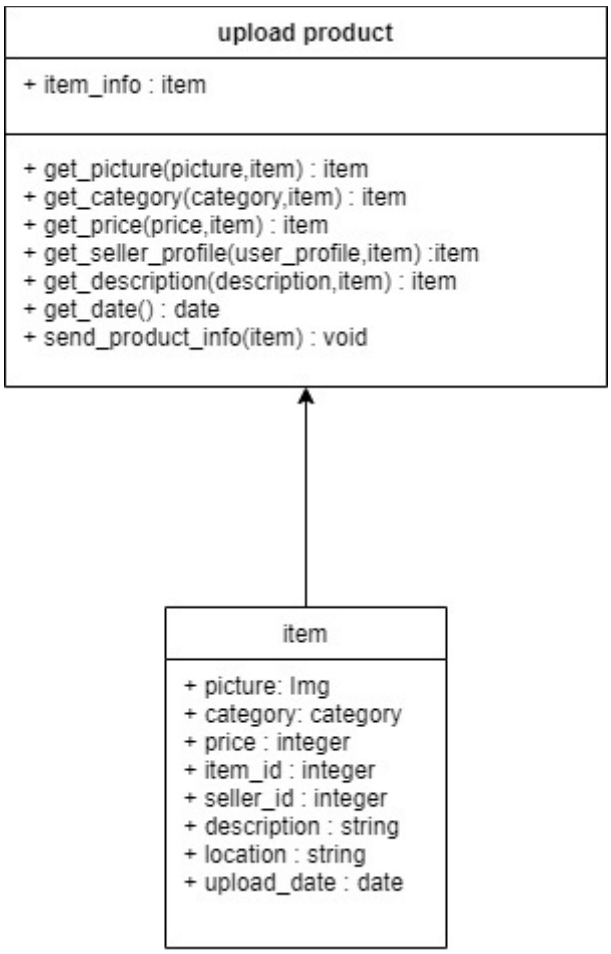

Diagram 4.2 Upload

1. upload product - 상품을 등록 객체

A. attributes

+ item_info: 상품의 객체

B. methods

+ get_picture(picture,item): item 객체의 상품의 이미지 파일을 등록
+ get_category(category,item): item 객체의 상품의 카테고리 설정
+ get_price(price,item): item 객체의 가격 설정
+ get_seller_profile(user_profile,item): item 객체에 판매자의 id, location 설정
+ get_description(description,item): item 객체에 판매자의 설명 추가
+ get_date(): item 객체가 DB에 등록될 때 시간을 item 객체에 저장
+ send_product_info(item): DB에 item 객체 정보 전달

2. item - 판매 상품 객체

C. Chatting

1. Class Diagram

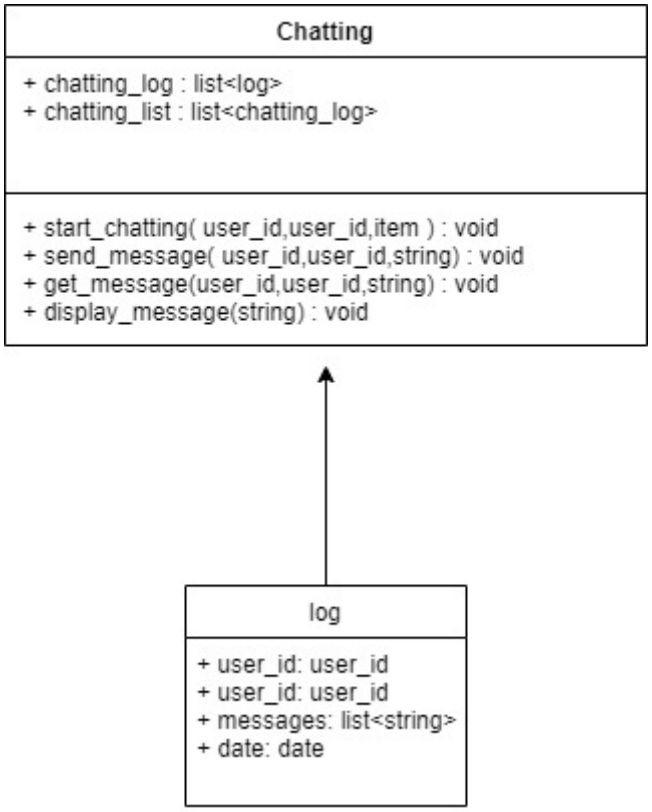

Diagram 4.3 Chatting

1. chatting - 채팅 시스템 객체

A. attributes

+ chatting_log: 채팅 내용을 기록해 놓은 목록

+ chatting_list: 사용자의 채팅 기록을 모아놓은 목록

B. methods

+ start_chatting(user_id,user_id,item): 판매자와 사용자 사이의 상품 거래에 대한 채팅 log 생성

+ send_message(user_id,user_id,string): 사용자의 다른 사용자에 대한 채팅 전송

+ get_message(user_id,user_id,string): 다른 사용자로 부터 채팅 내용 수신

+ display_message(string): 사용자의 채팅 내용 표시 및 log에 기록

2. log - 채팅 내용 객체

+ user_id: 발신자, 혹은 수신자의 user id

+ message: 발신, 혹은 수신 내용

+ date: 발신, 수신 시간

D. My page

1. Class Diagram

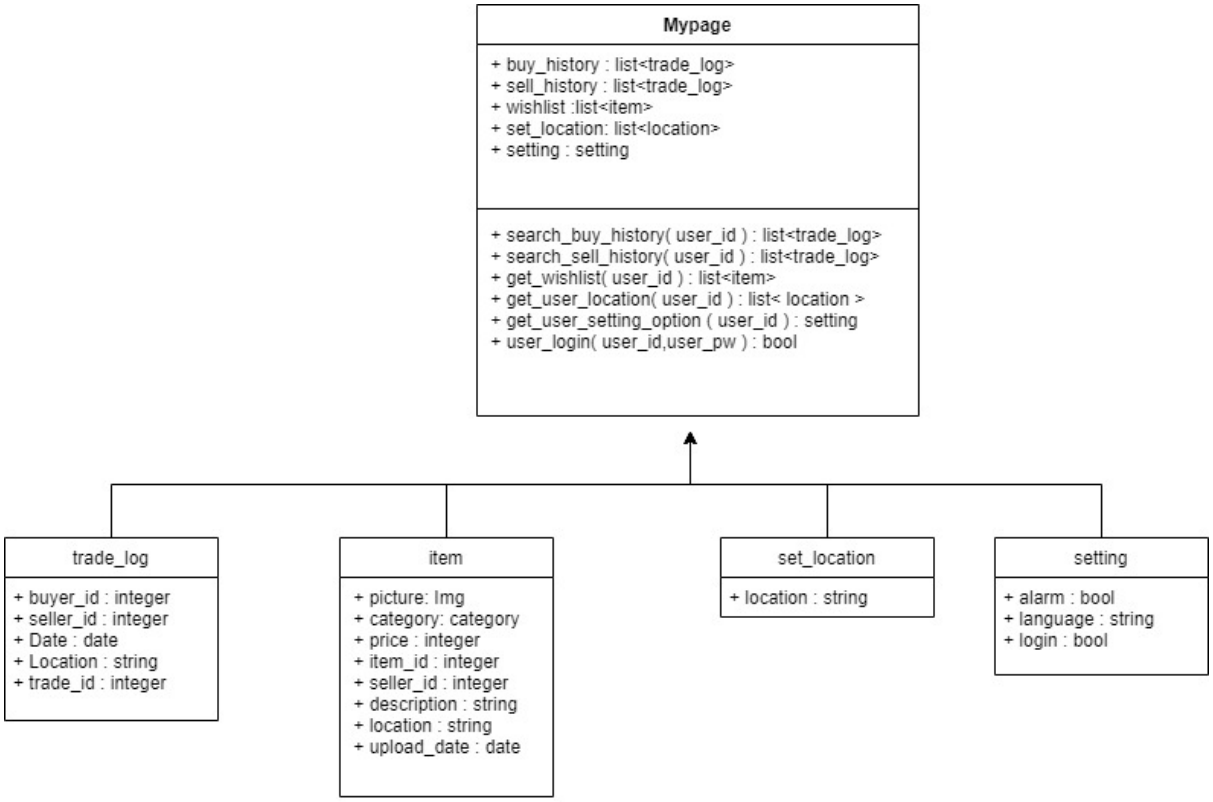

Diagram 4.4 My page

1. Mypage – 사용자의 정보를 담은 객체

A. attributes

+ buy_history: 사용자의 물품 구매 목록

+ sell_history: 사용자의 물품 판매 목록

+ wishlist: 사용자가 관심있는 물품 목록

+ set_location: 사용자의 동네 위치 정보

+ setting: 사용자의 시스템 설정

B. methods

+ search_buy_history: 전체 trade_log 중 사용자가 구매자인 trade_log 검색

+ search_sell_history: 전체 trade_log 중 사용자가 판매자인 trade_log 검색

+ get_wishlist: 전체 item 중 사용자가 관심 표시 한 item 검색

+ get_user_location: 사용자의 회원 가입 시 자신의 동네로 설정한 위치 정보 설정

+ get_user_setting_option: 사용자의 시스템 설정

+ user_login: 사용자 로그인 기능

2. trade_log - 거래 기록

+ buyer_id: 구매자의 user_id

+ seller_id: 판매자의 user_id

+ Date: 거래가 발생한 날짜

+ Location: 거래가 이루어진 동네 정보

+ trade_id: 거래의 고유 ID

3. item - 판매 상품 객체

4. set_location - 사용자의 동네 정보

5. setting - 사용자의 시스템 설정

+ alarm: 새로운 알람 표시

+ language: 사용자가 시스템을 사용할 언어

+ login: login 상태 정보

5. System Architecture - Backend

5.1 Objectives

전체 시스템 중 frontend를 제외한 backend 시스템과 서브시스템의 구조에 대해 설명한다.

5.2 Overall Architecture

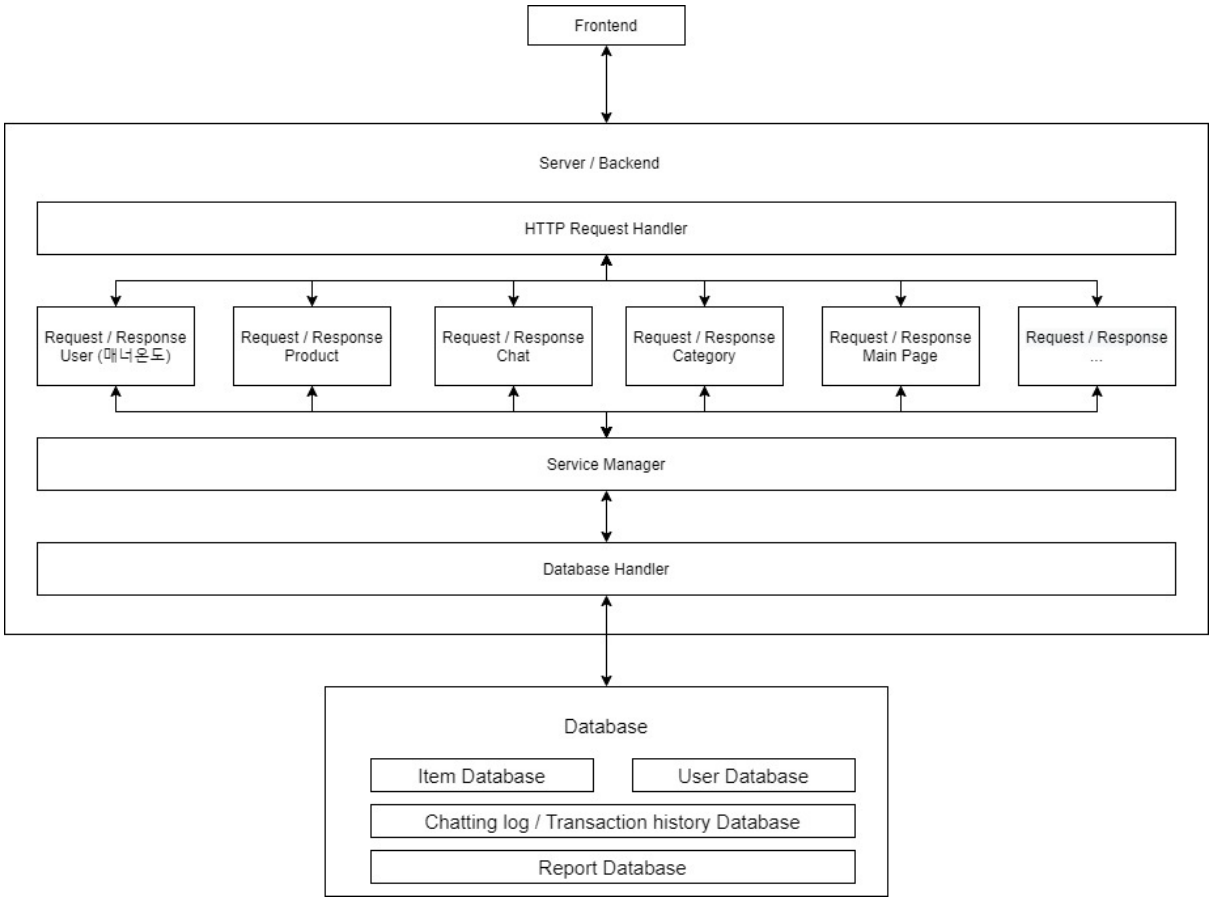

Diagram 5.1 Overall Architecture

Frontend 와의 직접적인 통신을 하는 server에서 database를 호출하는 구조이다.

5.3 Subcomponents

A. Sever

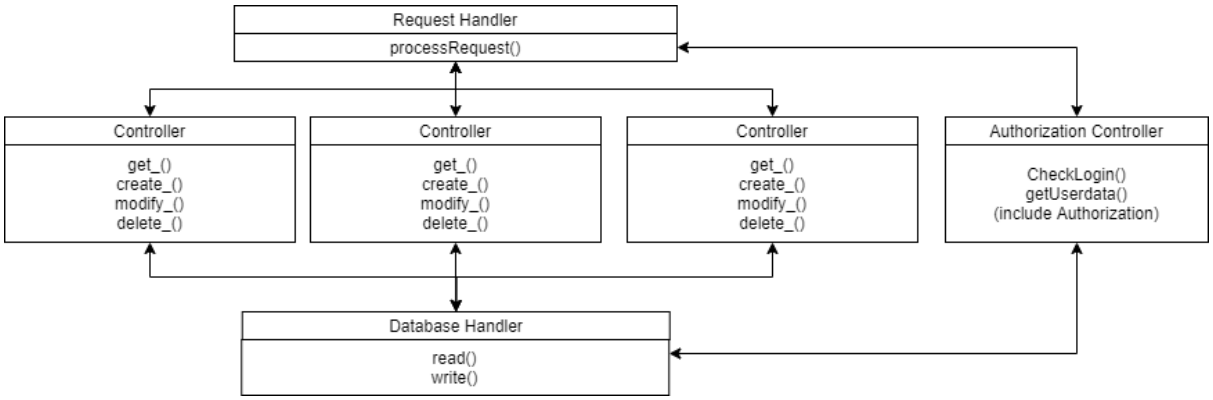

Diagram 5.2 Server

1. Request Handler : Manage requests come from frontend

(1) processRequest() : 프론트엔드에서 들어온 요청을 각 컨트롤러로 전송하고 프론트엔드에 응답하는 함수

2. controllers

(1) get_(): 정보를 가져오는 함수
(2) create_(): 정보를 생성하는 함수
(3) modify_(): 정보를 수정하는 함수
(4) delete_(): 정보를 삭제하는 함수

3. Authorization Controller : 권한 관리 컨트롤러

(1) Checklogin(): 로그인이 되어있는지 확인하는 함수
(2) getUserdata(): 사용자의 권한(admin, normal user) 과 사용자 정보를 불러오는 함수

4. Database Handler

(1) read(): 데이터베이스에서 정보를 읽도록 요청한다.
(2) write(): 데이터베이스에 수정을 할 수 있도록 요청한다.

B. Report

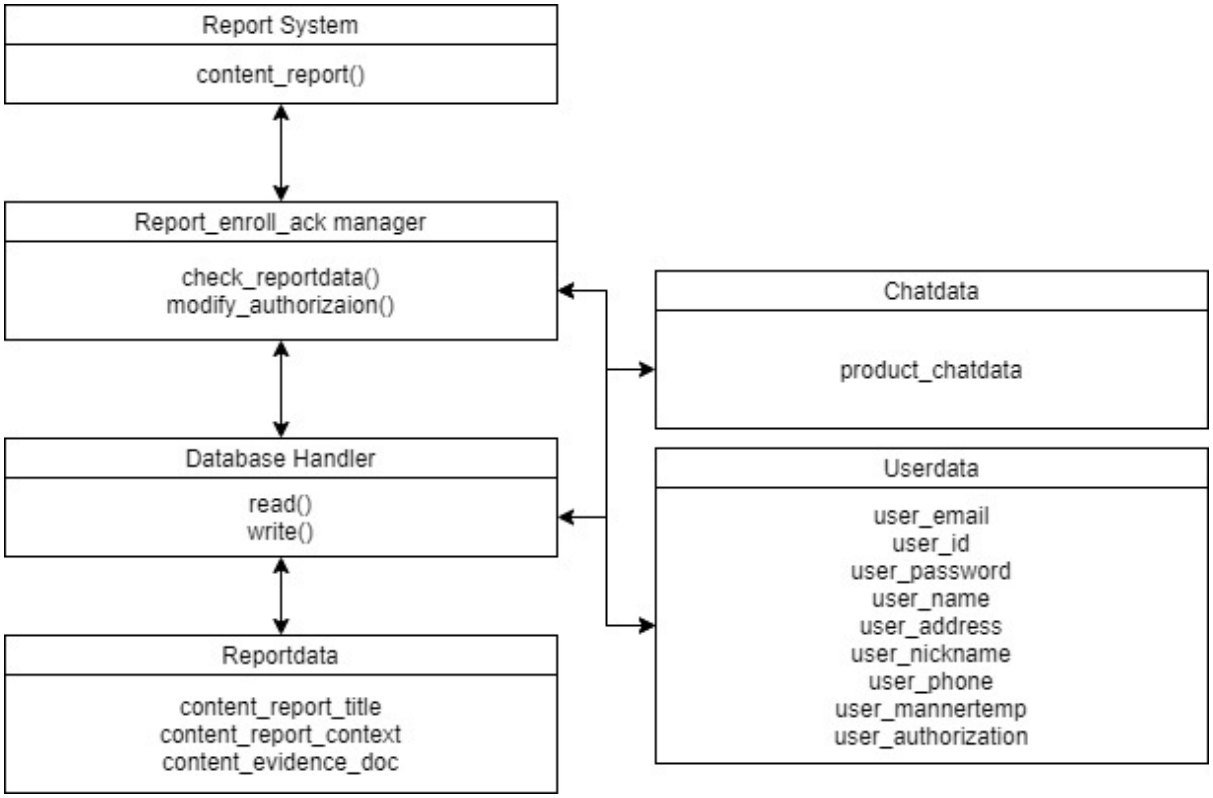

Diagram 5.3 Report

1. Report System: 사용자가 신고 사유, 첨부파일을 전송하고 다른 함수들을 call한다.

   (1) Report_enroll_ack manager: reportdata가 합당한지 판단후 사용자의 authorization을 수정한다.

2. Database handler

   (1) read(): 데이터베이스에서 정보를 읽도록 요청한다.
   (2) write(): 데이터베이스에 수정을 할 수 있도록 요청한다.

C. Review

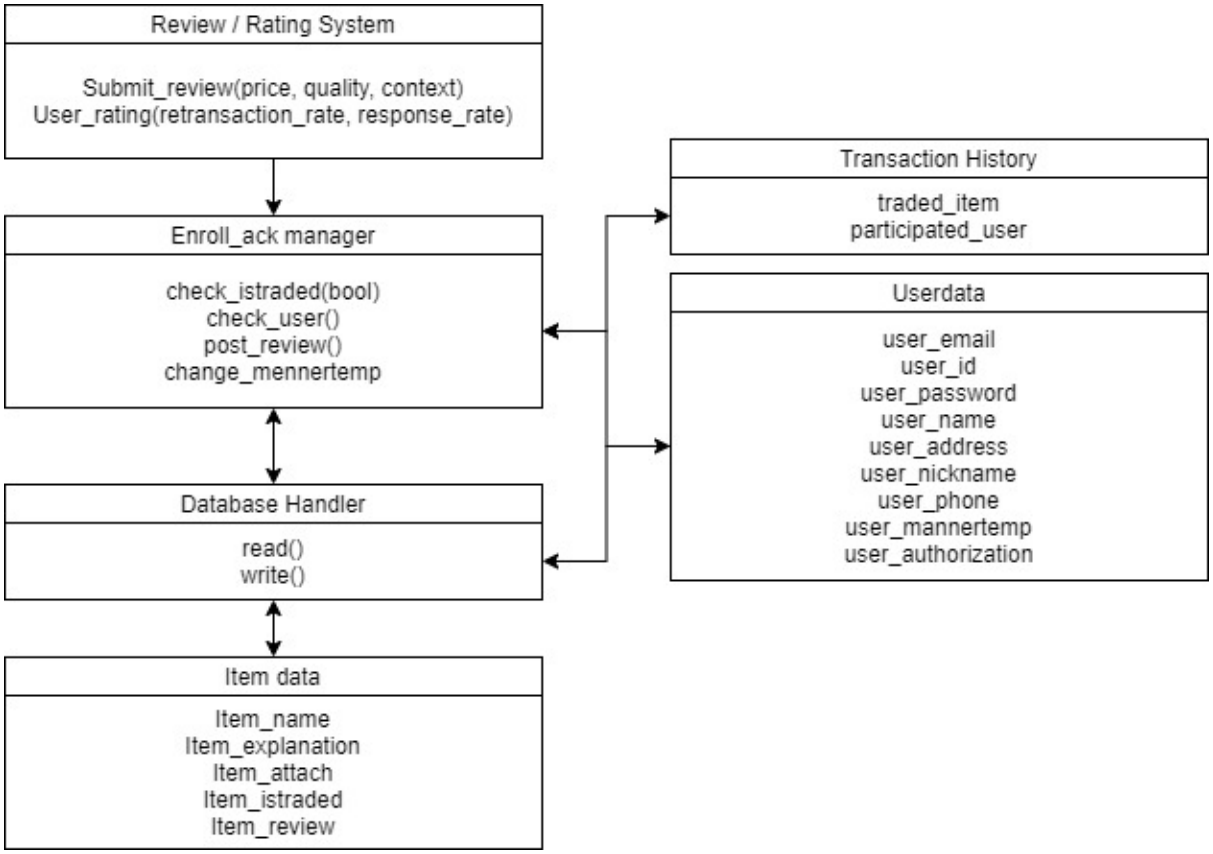

Diagram 5.4 Review

1. Review / Rating System

(1) Submit_review: 가격, 품질, 설명에대한 후기를 작성한다. 이는 상품 후기에 등록된다.

(2) User_rating: 재거래희망률, 응답률에 대한 평가를 한다. 이는 사용자의 매너온도에 영향을 끼친다.

2. Enroll_ack manager

(1) check_istraded(bool): 거래가 완료된 상품인지 확인한다.

(2) check_user(): 거래에 참여한 이용자가 맞는지 확인한다.

(3) post_review(): 상품에 리뷰데이터를 등록하도록 요청한다.

(4) change_mannertemp(): 사용자의 매너온도를 변경하도록 요청한다.

3. Database Handler

(1) read(): 데이터베이스에서 정보를 읽도록 요청한다.

(2) write(): 데이터베이스에 수정을 할 수 있도록 요청한다.

6.Protocol Design

6.1 Objectives

당근마켓에서 시스템과 시스템을 구성하는 컴포넌트 사이에서 발생하는 모든 Interaction을 위한 인터페이스와 프로토콜의 구조가 어떻게 설계되는지 설명한다. 그리고 인터페이스와 프로토콜을 어떤 환경과 언어를 활용하여 구현할 지 설명한다.

6.2 JSON

Interaction을 위한 프로토콜로 HTTP 프로토콜을 사용할 것이며, 컴포넌트들 사이에서 발생하는 요청/응답 프로토콜을 위한 데이터 포맷으로, 대표적으로 XML과 JSON이 있다. 우리는 단순하면서도 확장성이 좋은 JSON을 사용할 것이다.

JSON은 "키-값 쌍"으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다. 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합하다.

XML

```
<empinfo>
   <employees>
      <employee>
         <name>James Kirk</name>
         <age>40></age>
      </employee>
      <employee>
         <name>Jean-Luc Picard</name>
         <age>45</age>
      </employee>
      <employee>
         <name>Wesley Crusher</name>
         <age>27</age>
      </employee>
   </employees>
</empinfo>
```

JSON

```
{  "empinfo" :
      {
            "employees" :  [
            {
                  "name" : "James Kirk",
                  "age" : 40,
            },
            {
                  "name" : "Jean-Luc Picard",
                  "age" : 45,
            },
            {
                  "name" : "Wesley Crusher",
                  "age" : 27,
            }
                        ]
      }
   }
```

Figure 6.1 XML과 JSON의 포맷의 차이점

그림에서 보는 것과 같이. JSON은 XML보다 단순한 형식을 가지고 있기 때문에, 이용하기 좀 더 쉽다. 또한, 프로그래밍 언어에 독립적이고, 많은 프로그래밍 언어에서 JSON의 parser를 지원하고 있기 때문에 JSON을 사용하는 것이 용이하다고 할 수 있다.

6.3 Details

a. 회원 가입 인증

-요청

Attribute	Value
user_phone	전화번호
user_identification_number	인증 번호

Table 6.2 회원 가입 인증 요청

-응답

Attribute	Value
Signup_Authorization	회원 가입 허가

Table 6.3 회원 가입 인증 응답

b. 회원 가입

-요청

Attribute	Value
user_email	이메일 주소
user_password	비밀번호
user_phone	전화번호
user_name	이름
user_address	주소
user_nickname	닉네임 설정

Table 6.4 회원 가입 요청

-응답

Attribute	Value
signup_ack	회원 가입 승인

Table 6.5 회원 가입 응답

c. 로그인

-요청

Attribute	Value
user_phone	전화번호
user_identification_number	인증번호

Table 6.6 로그인 요청

-응답

Attribute	Value
signin_ack	로그인 승인

Table 6.7 로그인 응답

d. 내 동네 설정

-요청

Attribute	Value
user_gps_authorization	gps 허가 설정
user_address	주소

Table 6.8 내 동네 설정 요청

-응답

Attribute	Value
user_location_ack	내 동네 위치 정보 승인

Table 6.9 내 동네 설정 응답

e. 중고 거래 등록

-요청

Attribute	Value
user_nickname	닉네임
image_attachment	이미지 첨부 파일
content_title	글 제목

content_context	글 내용
content_categories	카테고리 선택
content_price	가격 입력
content_select_location	보여줄 동네 고르기

Table 6.10 중고 거래 등록 요청

-응답

Attribute	Value
content_enroll_ack	글 등록 승인

Table 6.11 중고 거래 등록 응답

f. 상품 평가

-요청

Attribute	Value
content_rate_price	가격 평가
content_rate_quality	품질 평가
content_rate_context	평가 내용

Table 6.12 상품 평가 요청

-응답

Attribute	Value
content_rate_enroll_ack	평가 등록 승인

Table 6.13 상품 평가 응답

g. 매너온도 평가

-요청

Attribute	Value
user_rate_retransaction	재거래희망률 평가
user_rate_responserate	응답률 평가

Table 6.14 매너온도 평가 요청

-응답

Attribute	Value
user_mannertemp_ack	매너온도 반영 승인

Table 6.15 매너온도 평가 응답

h. 신고

-요청

Attribute	Value
content_report_title	신고글 제목
content_report_context	신고글 내용
content_evidence_doc	증빙서류 첨부파일

Table 6.16 신고 요청

-응답

Attribute	Value
content_report_enroll_ack	신고글 등록 승인

Table 6.17 신고 응답

7. DB Design

7.1 Objectives

본 프로젝트의 Entity 간의 relationship을 개괄적으로 설명한다. 후에 자세한 설명을 덧붙여 구체적인 관계와 기능을 부연 설명한다. 이 과정을 아래의 Diagram을 통해 서술한다. 이를 통해 전체적인 프로젝트에 대한 청사진을 그려 보다 정확하게 기능의 종류를 정의 및 선택하고, 기능들이 어떻게 상호 연결되는지 알 수 있으며, 넓은 시야에서 프로젝트 진행상황을 볼 수 있다.

7.2 ER diagram

다이어그램에서 네모는 Entity를, 동그라미(타원형)은 Attribution을 나타낸다. Entity 혹은 Attribution 간의 짝대기는 상호관계를 나타내며, 짝대기에 화살표가 있는 경우 화살표를 향하는 쪽이 하위 관계에 있다는 것을 나타낸다. 점선은 해당 Entity 간 속성이 겹치는 경우를 나타낸다.

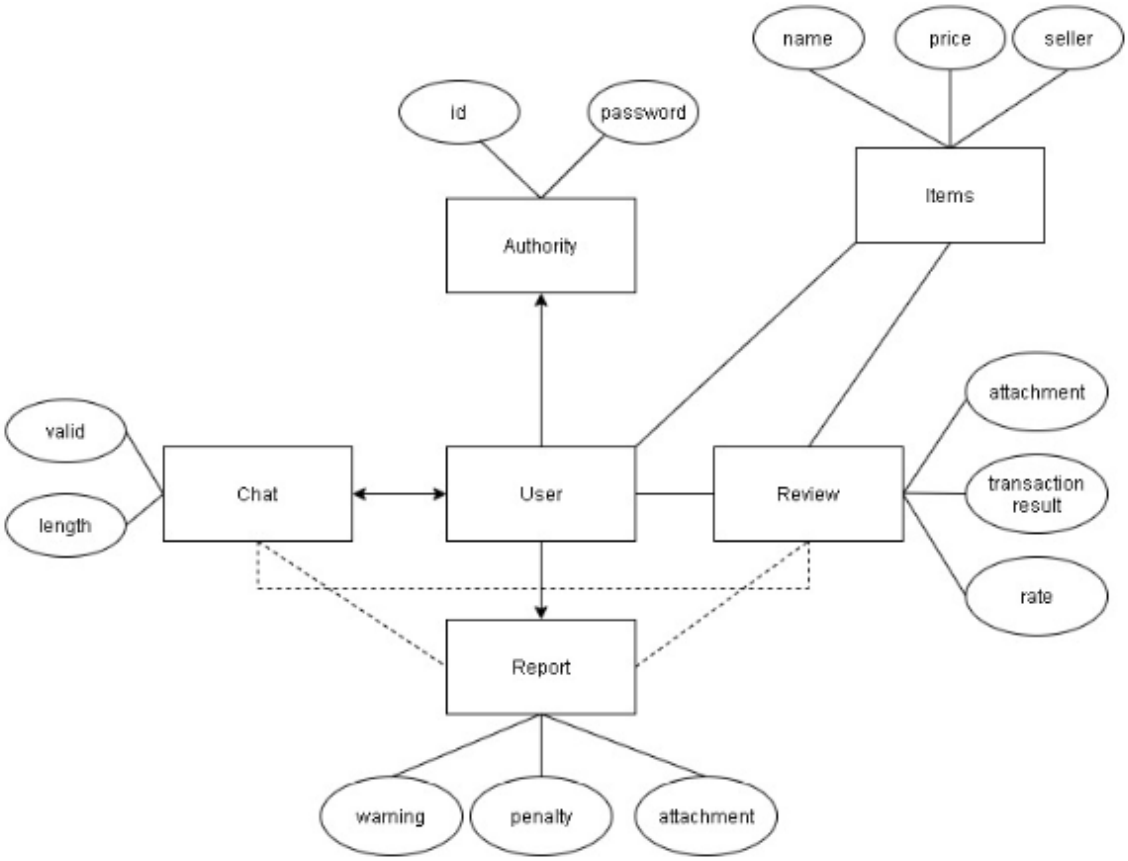

Diagram 7.1 Design specification

A. Entities

a. Authority

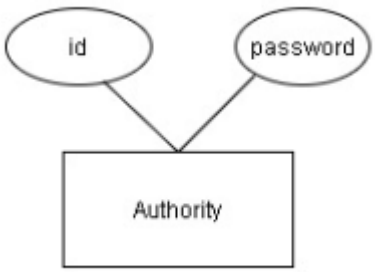

Diagram 7.2 Authority

b. Items

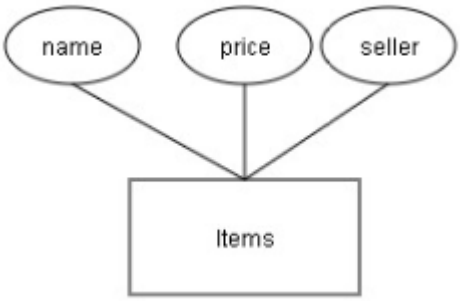

Diagram 7.3 Items

c. Review

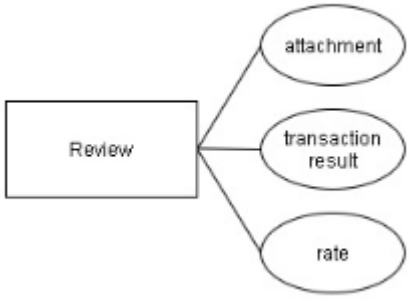

Diagram 7.4 Review

d. Report

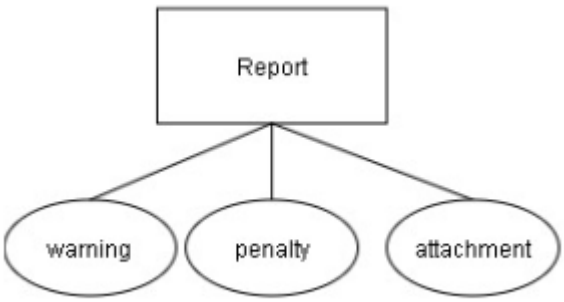

Diagram 7.5 Report

e. Chat

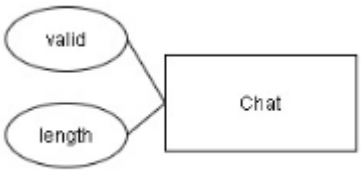

Diagram 7.6 Chat

B. Relations

7.3 Relational Schema

아래의 Relational Schema를 통해 Entity의 속성과 더불어 Operation을 알 수 있다. 보다 자세한 기능에 대한 설명을 통해 기능에 대한 이해를 높임과 동시에 코드의 유지보수 효율성을 증대시킬 수 있다.

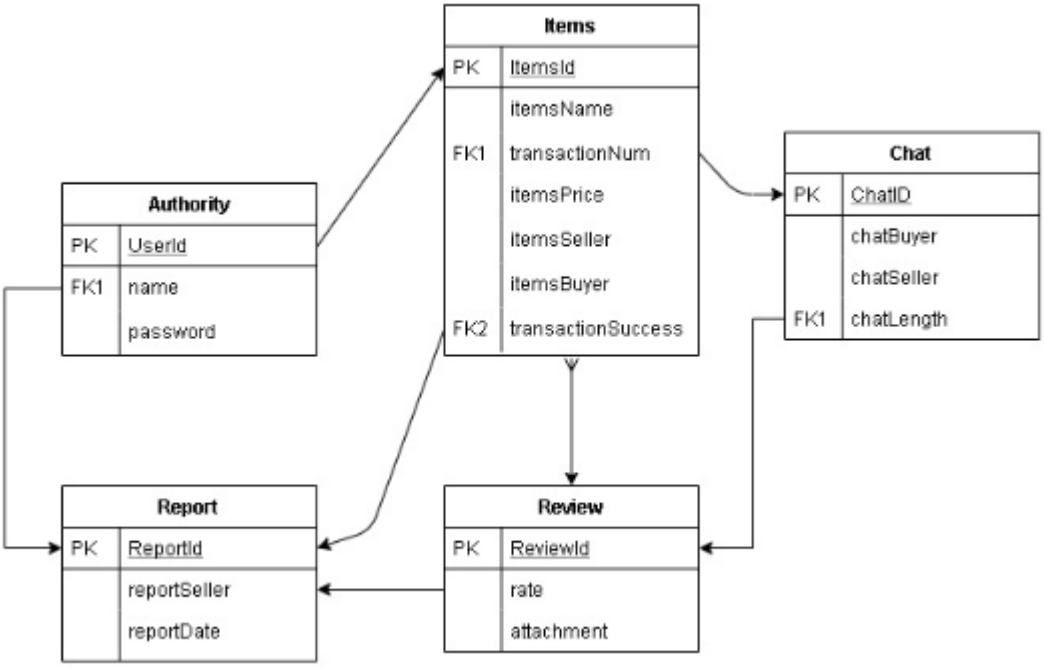

Diagram 7.7 Relational Schema

8.Testing plan

8.1 Objective

This part will carry out the test plan, describing the scope, methods, resources and progress of the test activities to be carried out; it is the application software assembly test and confirmation test of the entire information system. It determines the test project, the features to be tested, the test task, the person performing the task, and various possible risks. The test plan can effectively prevent planned risks and ensure the smooth implementation of the plan.

8.2 Testing Policy

A. Development Testing

Development testing is a very important part of the entire system. The quality of the test is related to the development of the product. Customers' needs for software quality, performance, and reliability must be achieved through testing. The testing process must follow the principles of rigor, perfection, and standardization. The main purpose of the test is to see if there will be bugs in the system operation, and then debug the bugs until the program runs perfectly. The fewer bugs, the lower the chance of system errors, and the user is more convenient and safe to use.

1. Reliability

   Software reliability refers to the probability of a software system operating without failure in a given environment and within a given time, and is an important component of software quality. We can use

software reliability evaluation techniques to test the reliability of the system.

Software reliability evaluation refers to the process of using statistical techniques to process software reliability testing and software failure data collected during system operation and evaluate software reliability. Implementing software reliability evaluation is also a perfection of the software testing process, which contributes to the reliability growth of the software product itself.

2. Security

We can operate the database to improve the security of the system. After the user registers an account, the system allows the user to identify their own identity or name in a certain way. Each time the user requests to enter the system, the system will check the identity of the login person, and then enter the system after passing the authentication. On this basis, we can add access control methods. Only users who have passed the user rights definition and have legal rights can access the database. All unauthorized personnel cannot enter the database to store or read data. Encrypt and protect the stored and transmitted content, so that people without authority cannot learn the data content.

3. Performance

We can judge the performance of the software by three aspects.

i. Response time.

ii. Throughput.

iii. Concurrent users

Response time refers to the time for the system to respond to the request. Intuitively, this indicator is very consistent with people's subjective perception of software performance, because it completely records the time that the entire computer system processes the request. Since a system usually provides many functions, and the processing logic of different functions is also very different, the response time of different functions is not the same, even the response time of the same function under different input data. Therefore, when discussing the response time of a system, we usually refer to the average time of all functions of the system or the maximum response time of all functions. Of course, it is often necessary to discuss the average response time and maximum response time for each function or group of functions.

Throughput refers to the number of requests processed by the system in a unit of time. For application systems without concurrency, throughput and response time are strictly inversely proportional. In fact, throughput is the reciprocal of response time. For a single-user system, the response time (or system response time and application delay time) can be a good measure of system performance, but for concurrent systems, usually need to use throughput as a performance indicator.

The number of concurrent users refers to the number of users that can be carried by the system and use the system functions normally. Compared with throughput, the number of concurrent users is a more intuitive but more general performance indicator.

B. Release Testing**[3]**

Release testing is a test to make the software be released smoothly. Reduce bug as much as possible during the testing phase to make the software more complete.

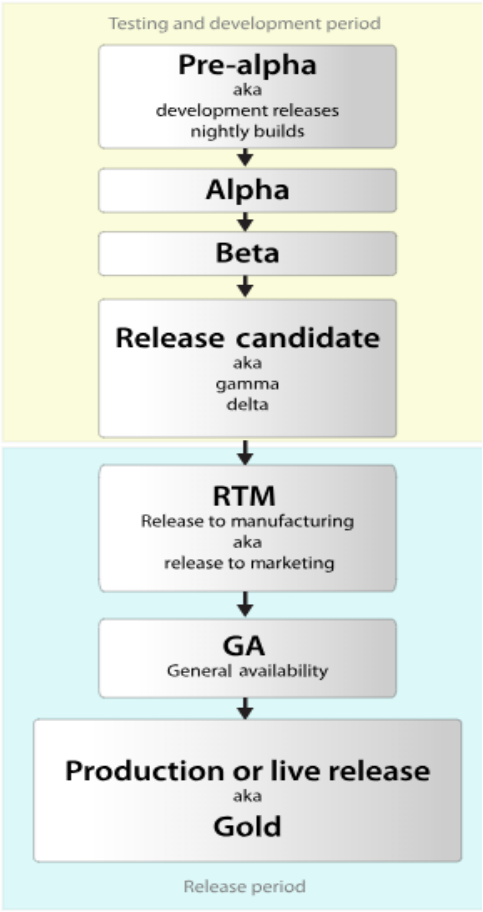

Figure 8.1 Release Testing

As shown in the above figure, Release testing includes development period and completion period. Development period includes Pre-Alpha, Alpha, Beta, Release candidate. Completion period includes RTM, GA, Release to web.

C. User testing

Find as many people as possible to conduct project experiments and conduct individual tests. Let everyone write their own feelings after use, and record their use process. Let them conduct their own experiments instead of letting us guide them. They must not have any help behavior. After testing, find out the shortcomings and correct them to improve the software.

9. Development Plan

9.1 Objectives

This part will introduce the technology and development procedures used in actual development and explain the development formation.

9.2 Frontend Environment

A. Bootstrap**[4]**

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Bootstrap is the sixth-most-starred project on GitHub, with more than 135,000 stars, behind freeCodeCamp and marginally behind Vue.js framework.

Figure 9.1 Bootstrap

9.3 Backend Environment

A. Django **[5]**

Django is a Python-based free and open-source web framework that follows the model-template-view (MTV) architectural pattern. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Figure 9.2: Django

B. SQLite**[6]**

SQLite is a relational database management system (RDBMS) contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, generally following PostgreSQL syntax. However, SQLite uses a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity. This means that one can, for example, insert a string into a column defined

as an integer. SQLite will attempt to convert data between formats where appropriate, the string "123" into an integer in this case, but does not guarantee such conversions, and will store the data as-is if such a conversion is not possible.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others.[8] SQLite has bindings to many programming languages.

Figure 9.3: SQlite

9.4. Schedule

Phase	Expected Date	Actual Date
Proposal	5/2	5/3
Requirements Documentation	5/12	5/13
Design Documentation	5/23	5/24
Code and Test result	6/6	-
Presentation	6/12	-

Table 9.4 Schedule

10. Index

10.1 Figure index

10.2 Table index

10.3 Diagram index

11.Reference

[1]https://www.1337pwn.com/json-vs-xml-format-use-api/

[2]https://ko.wikipedia.org/wiki/JSON

[3]Release testing

https://zh.wikipedia.org/wiki/%E8%BB%9F%E4%BB%B6%E7%89%88%E6%9C%AC%E9%80%B1%E6%9C%9F#/media/File:Software_dev2.svg

[4]BOOTSTRAP https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

[5]DJANGO https://en.wikipedia.org/wiki/Django_(web_framework)

[6]SQL https://en.wikipedia.org/wiki/SQLite