SSHSHOP

Design Specification

Team 7

2015310874 강재현

2017311902 김태규

2013311305 유재현

2015312871 차봉건

2016313856 성아영

목차

1. Preface

A. Objective

Preface 에서는 본 Design Specification 에 관한 전반적인 설명이 포함된다. 문서의 예상독자를 서술하고, 전체적인 문서의 구조와 각 부분의 역할, 그리고 이 문서가 어떤 과정을 거쳐 수정되어 왔는지를 설명한다.

B. Readers

이 문서의 예상 독자는 실질적으로 시스템을 개발하고, 유지보수 하는 사람들이라고 가정한다. 소프트웨어 엔지니어, 시스템구조 설계자 등 시스템 개발과 관련된 모든 사람을 포함한다.

C. Document Structure

i. Preface

Preface 단계에서는 본 문서 목적성, 본 문서에 관한 예상독자, 그리고 전체적인 문서의 구조와 각 부분의 역할, 그리고 이 문서가 어떤 과정을 거쳐 수정되어 왔는지를 설명한다.

ii. Introduction

Introduction 단계에서는 본 문서를 작성하는 데 사용된 다이어그램과 도구들에 관해 설명하고, 본 프로젝트가 다루는 시스템의 범위에 관해 설명한다.

iii. System Architecture – Overall

System Architecture 에서는 전체적인 시스템 구조를 기술한다. 기술 방법은 ADL 중 Weave 를 사용한다. System 을 구성하는 각 sub-system 을 하나의 component 로 생각하여 components 와 그들의 관계를 설명한다.

iv. System Architecture – Frontend

전체 시스템 중에서 사용자와의 상호작용을 담당하는 프론트엔드 시스템의 구조, 프론트엔드를 구성하는 컴포넌트의 구성과 작동 방향, 다른 컴포넌트와의

관계를 기술한다. 프론트엔드의 컴포넌트의 기준은 각 기능을 담당하는 페이지 군을 의미한다.

v. System Architecture – Backend

이 부분에서는 전체 시스템 중 사용자와의 상호작용을 담당하는 frontend 를 제외한 Backend 시스템과 각 서브시스템의 구조에 대해 설명한다.

vi. Protocol Design

네이버 로그인 인증 요청은 앱에 네이버 로그인 화면을 띄우는 역할을 한다.

vii. Database Design

System Requirement Specification 을 통해서 도출된 각 sub-system 및 service 들의 요구사항을 기반으로 하여 데이터베이스 설계를 기술한다. ER Diagram 을 통해서 Entity 와 Entity 간의 관계를 기술하고 Relational Schema 를 만든다. SQL DDL 은 사용하기로 결정한 WSGI framework 인 Django 에서 application 별로 별도 처리하기 때문에 기술하지 않는다.

viii. Testing Plan

Testing Plan 은 시스템 내부에 결함이 있는지 확인하고, 의도한대로 실행이 되는지 확인하기 위한 것이다. 이를 위해, test policy 에 맞게 여러 test 들을 설계한 후, 각각의 test case 들에 대하여 test 를 진행하여야 한다. 이 때, Testing Policy 에서는 testing 에 대한 단계적 접근을 설명하며, Test Case 는 각 Sub-System 에서 사용자를 판별하고 그에 따라 시스템에 기대하는 입,출력 동작을 서술한다.이번 장에서는, 이러한 testing policy 및 test case 들에 대해서 기술한다.

ix. Development Plan

이번 단락에서는 구현 단계에서 사용하는 각종 프레임워크 및 라이브러리 등을 다루고 향후 개발 일정과 진행 방향을 설명한다.

x. Index

2. Introduction

A. Objective

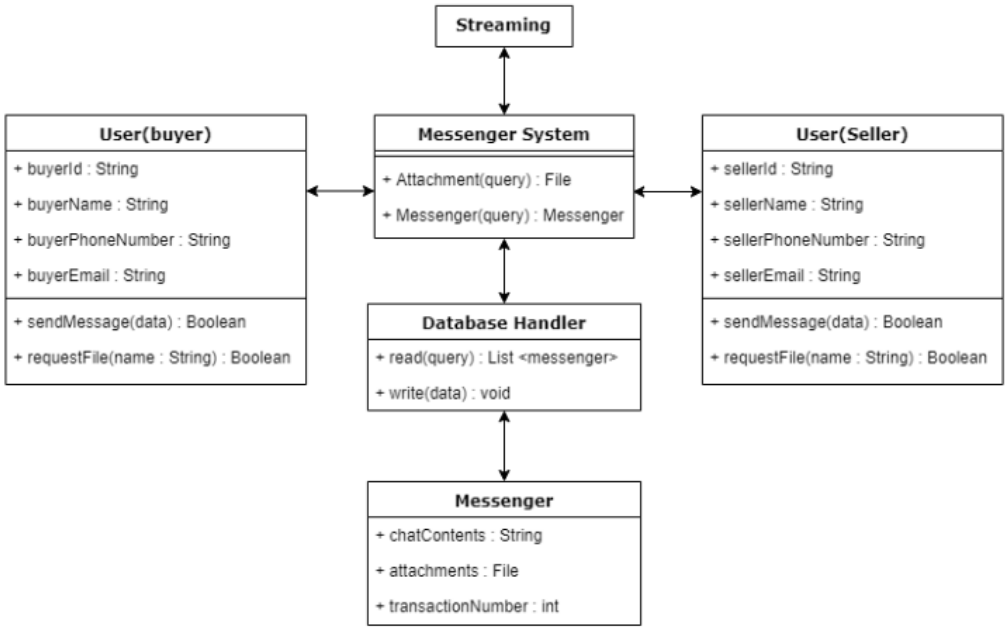
Intoduction 단계에서는 본 문서를 작성하는 데 사용된 다이어그램과 도구들에 관해 설명하고, 본 프로젝트가 다루는 시스템의 범위에 관해 설명한다.

B. Applied Diagram

i. UML

UML 은 Unified Modeling Language 의 약자로 소프트웨어 공학에서 사용되는 표준화된 범용 모델링 언어를 뜻한다. UML 은 객체지향 소프트웨어 집약 시스템을 개발할 때 산출물을 명세화, 시각화, 문서화할 때 주로 쓰인다. UML 을 사용하면 다른 사람들과의 의사소통 또는 설계 논의가 쉽고, 전체 시스템의 구조 및 클래스의 의존성 파악이 용이하며, 유지보수를 위한 설계문서로 쓰일 수 있다는 장점이 있다. 본 문서에서는 Class Diagram, Sequence Diagram, Component Diagram 등을 사용해 시스템의 구조를 시각화 한다.

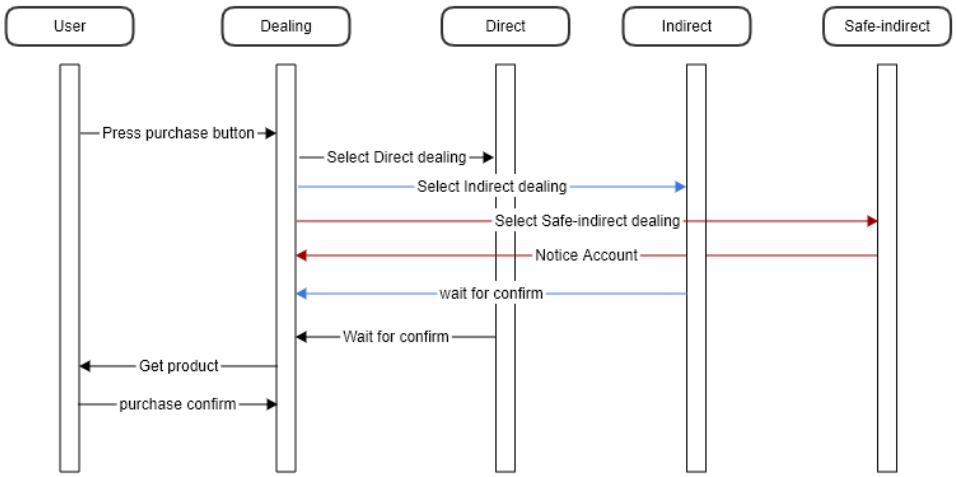
ii. Class Diagram



[그림 1] Example of Class Diagram

Class Diagram 은 UML 의 일종으로 시스템의 클래스, 속성, 방법, 그리고 객체 간의 관계를 보여줌으로써 시스템의 구조를 설명하는 정적구조 다이어그램이다. 본 문서에서는 Frontend, Backend 시스템이 어떻게 구성되어 있는지 서술할 때 주로 사용해 시스템 구조를 설명하고자 하였다.

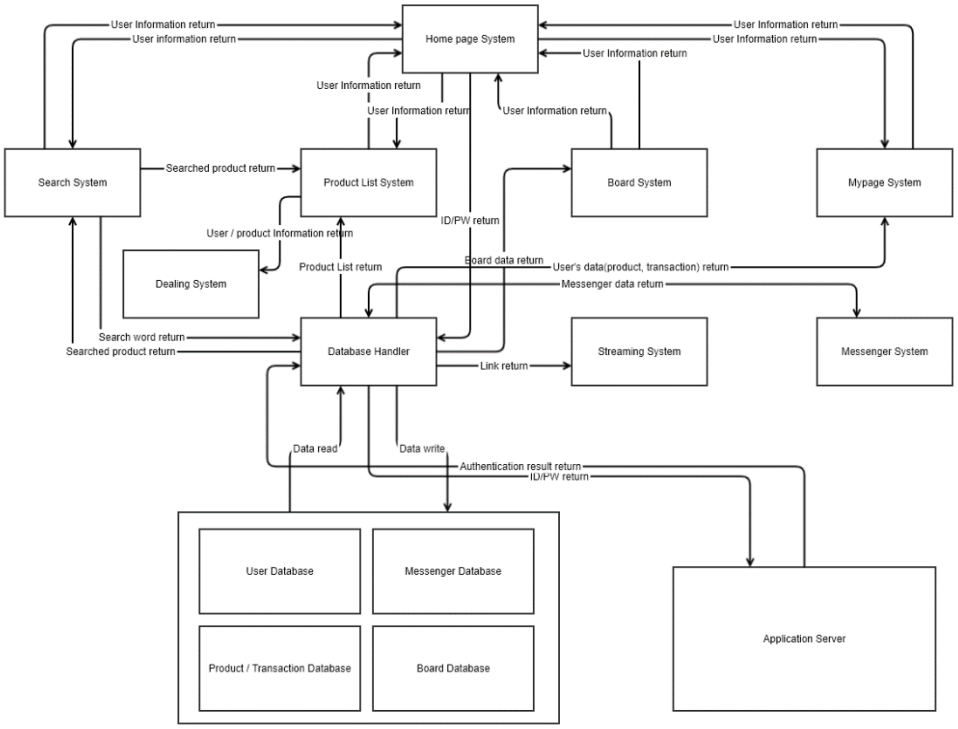
iii. Sequence Diagram



[그림 2] Example of Sequence Diagram

Sequence Diagram 은 시스템 구성요소들 사이에서 어떻게 상호작용이 일어나는지 그 순서를 기술한다. 시스템과 시스템을 사용하는 사용자, 그리고 시스템 구성요소 간의 상호작용을 나타낸다. 역시 본 문서에서는 Frontend 와 Back-end 에서 User 와 System 을 구성하는 요소 간의 상호작용을 나타내기 위해 본 Diagram 을 사용하였다.

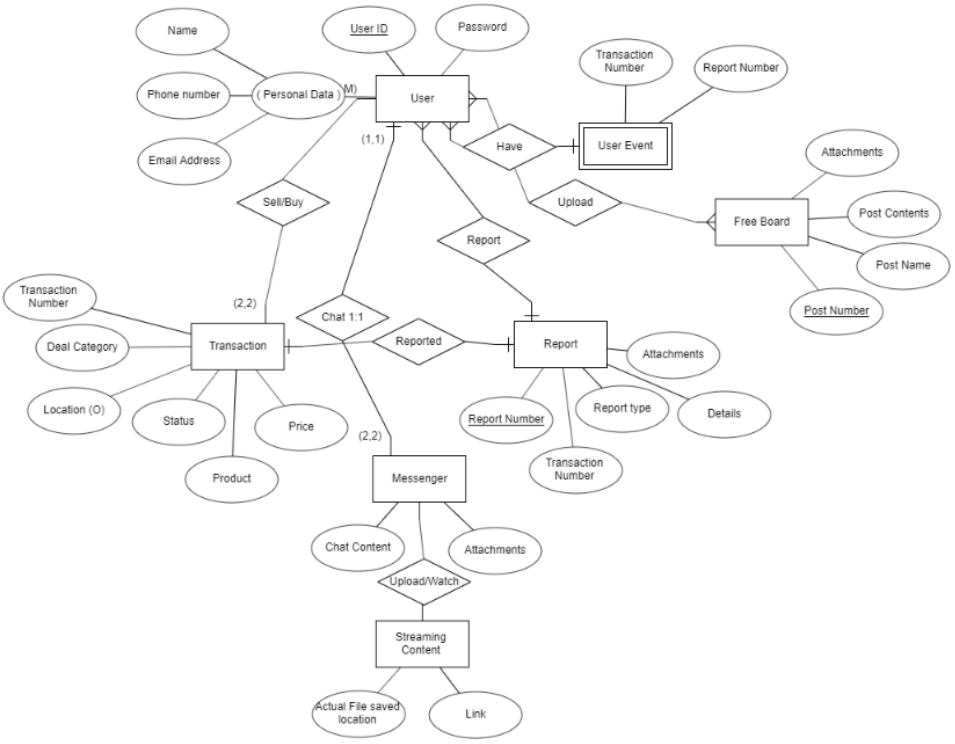
iv. Weave



[그림 3] Example of Weave

Weave 는 Component 사이의 관계를 상세히 서술한 ADL 이다. 전반적인 System architecture 을 서술하기 위해 Weave 를 사용하였다.

v. ER Diagram



[그림 4] Example of ER Diagram

ER Diagram 은 요구사항으로부터 알아낸 정보들을 개체, 속성, 관계성으로 기술하튼 데이터 모델을 말한다. 본 문서에서 데이터베이스의 개체, 속성 간의 관계를 표현할 때 DR Diagram 을 사용해서 표현한다.

C. Applied Tool

i. Power Point

[그림 5] Logo of PowerPoint

파워포인트는 프레젠테이션을 목적으로 개발된 소프트웨어이다. 하지만 기본적인 그래픽 툴을 제공하고 다루기가 쉽기 때문에 Diagram 을 그릴 때 이용하였다.

ii. ERDPlus

[그림 6] Logo of ERD

ERD 는 엔티티 관계 다이어그램, 관계 스키마 등 데이터베이스 모델링을 용이하게 도와주는 툴이다. 데이터베이스 ER Diagram 을 작성할 때 본 프로그램을 이용하였다.

iii. Gliffy

[그림 7] Logo of Gliffy

Gliffy 는 HTML5 클라우드 기반 앱을 통한 다이어그램 제작 소프트웨어이다. 온라인 상에서 UML, 평면도, 벤다이어그램, 플로우차트 등 다양한 종류의 도표를 만드는데 사용되는 툴이다. 이를 이용해 Class Diagram, Sequence Diagram 등을 작성하였다.

iv. Draw.io

[그림 8] draw.io

Draw.io 는 온라인 모델링 툴로 기본적으로 제공하는 도형과, 템플릿이 풍부하다. 따라서 Gliffy 와 함께 draw.io 를 이용해서 frontend, backend 대부분의 Diagram 을 작성하였다.

D. Project Scope

'SSH SHOP'은 커지는 중고거래 시장에 발맞춰 여러가지 기능을 제공하기 위해 등장한 서비스이다. 중고거래는 판매자, 구매자에게 많은 이점이 있어 자주 이용되어 왔지만 개인대 개인의 거래인 만큼 사기 위험성을 배제할 수 없었다. 우리는 이를

보완하기 위해 판매자와 구매자 사이에 번호를 교환하지 않고 연락할 수 있는 메진서 시스템, 비디오 스트리밍 기능을 통해 구매하는 물건을 좀더 신뢰성 있게 확인할 수 있는 기능, 구매자와 판매자 사이에서 직접적으로 돈이 오고가는 대신 중간에 관리자를 두어 사기위험성을 낮추고자 한 Indirect-dealing system, 사기위험성이 높은 품목에 한해 구매 페이지에서 알림을 제공하는 서비스, 사기를 당했을 때 신고를 원활하게 할 수 있는 신고게시판과 같은 기능을 제공하고자 하였다.

먼저 Frontend System 은 시스템과 사용자와의 상호작용을 담당한다. Backend System 은 Frontend 에서 오는 여러 요청에 응답한다. 이 때 Backend 를 구성하는 다양한 Subsystem 을 호출하는 역할을 한다. Messenger system 은 구매자가 구매할 물품을 클릭한 뒤 실행시킬 수 있다. Streaming system 은 Messenger system 이 호출하는 system 으로 동영상을 통해 물품 확인을 도와주는 역할을 한다.

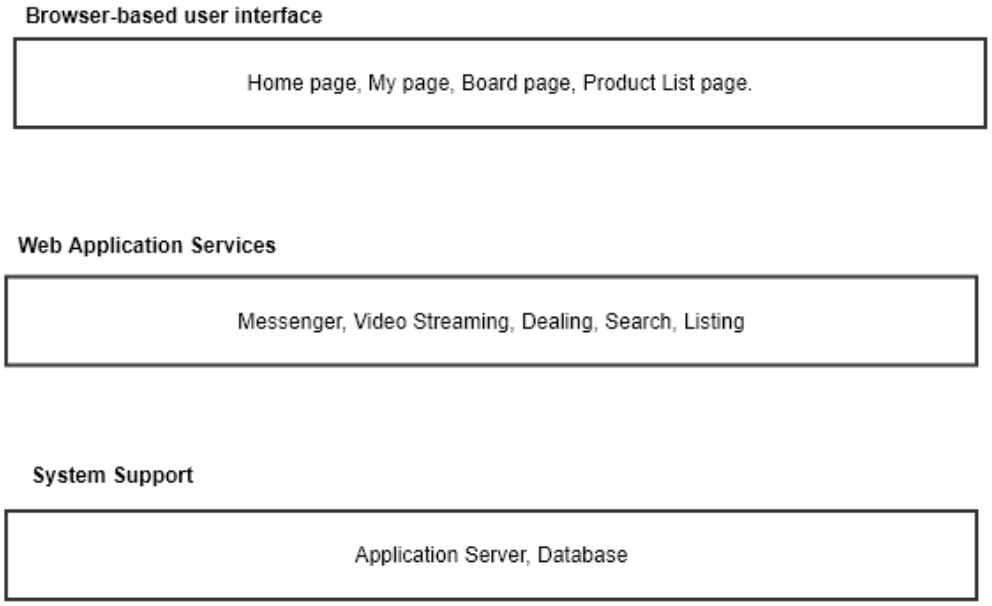
3. System Architecture – Overall

A. Objectives

System Architecture 에서는 전체적인 시스템 구조를 기술한다. 기술 방법은 ADL 중 Weave 를 사용한다. System 을 구성하는 각 sub-system 을 하나의 component 로 생각하여 components 와 그들의 관계를 설명한다.

B. Generic System Model

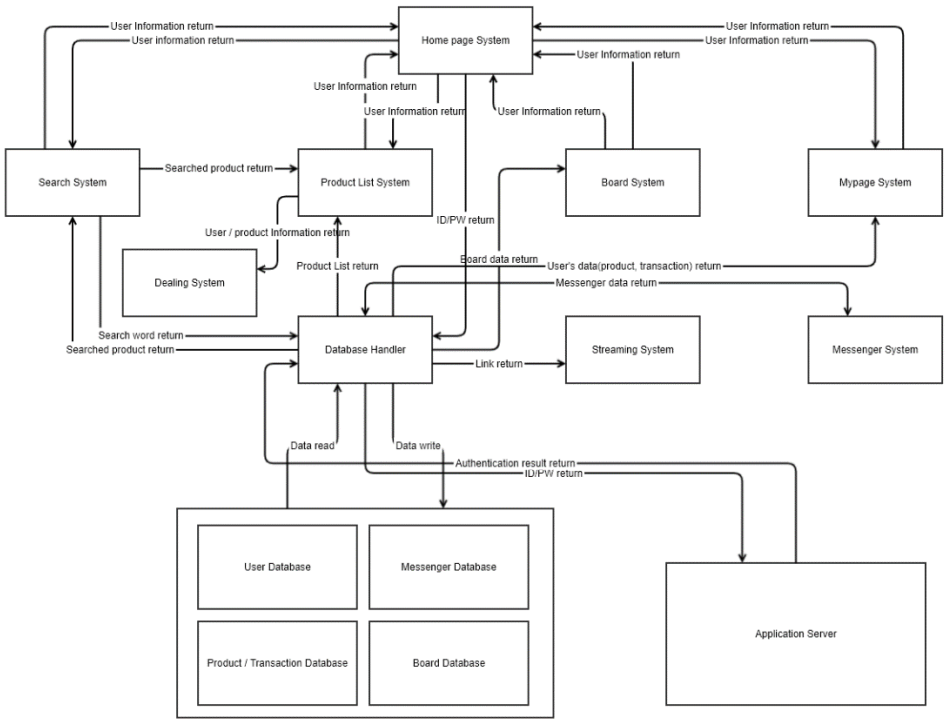
System 의 전체적인 model 에 있어서는 Generic system model 중 하나인 layered architecture model 을 기반으로 한다. 맨 위의 layer 에는 User 가 접하는 User Interface 부분(frontend), 맨 아래의 layer 는 Backend 에서 처리하는 DB 와 Server 가 존재한다. 인접한 layer 는 서로 상호작용하여 layer 의 각 component 마다 주어진 기능을 수행한다.



[그림 9] Generic system model

C. System Organization

ADL(Architecture Description Language) 중 Weave 를 통해 각 component(sub-system)들의 관계를 나타내었다. 앞서 말한 Layered architecture 를 고려하여, 위쪽은 User Interface, 중간 부분은 core business logic/application functionality, 아래층은 System Support 기능을 수행하는 component 를 배치하여 나타내었다.



[그림 10] System Organization

4. System Architecture – Frontend

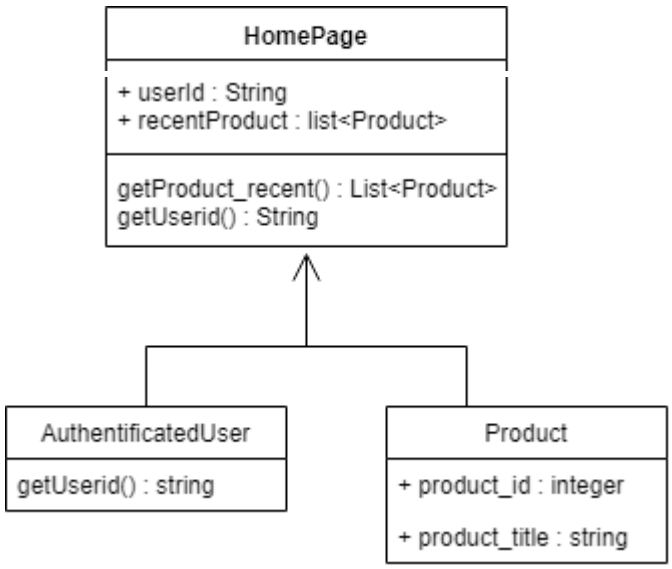
A. Objective

전체 시스템 중에서 사용자와의 상호작용을 담당하는 프론트엔드 시스템의 구조, 프론트엔드를 구성하는 컴포넌트의 구성과 작동 방향, 다른 컴포넌트와의 관계를 기술한다. 프론트엔드의 컴포넌트의 기준은 각 기능을 담당하는 페이지 군을 의미한다.

B. Subcomponents

i. Homepage System

1. Class Diagram



[그림 11] homepage class diagram

2. Homepage 객체

A. Homepage

- userId : 현재 접속한 유저 아이디
- recentProduct : 최근에 업로드된 물품 정보

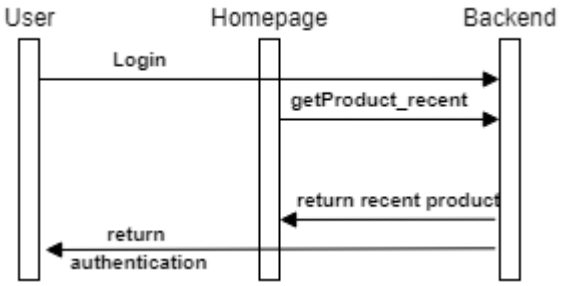
B. AuthentificatedUser

- getUserid( ) : 현재 접속한 유저의 정보를 가져오는 메소드

C. Product

- product_id : 물품이 가지고 있는 고유 아이디 정보
- product_title : 사용자가 물품을 올릴 때 설정한 제목

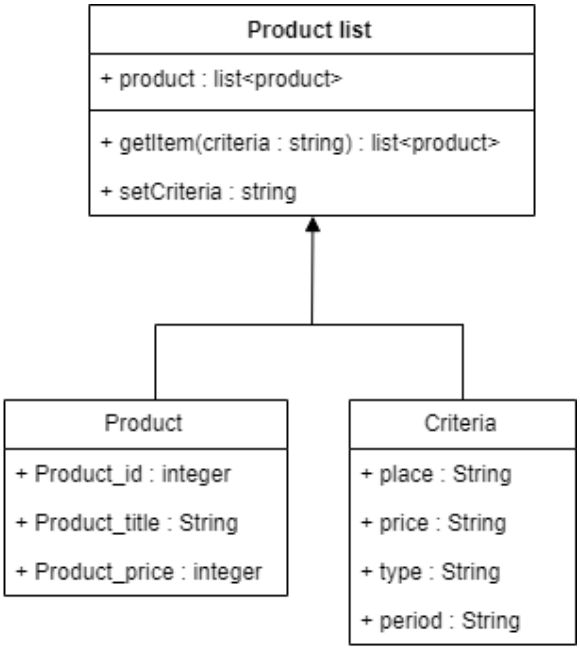
3. Sequence Diagram



[그림 12] Homepage Sequence Diagram

ii. Product List System

1. Class Diagram



[그림 13] Product List System Class Diagram

2. Product List 객체

A. Product list

- Product : 리스트 형식으로 제공되는 제품의 정보

- setCriteria : 보여줄 제품의 조건
- getItem(criteria : string) : 조건에 맞는 제품만 추린 제품을 보여주는 메소드

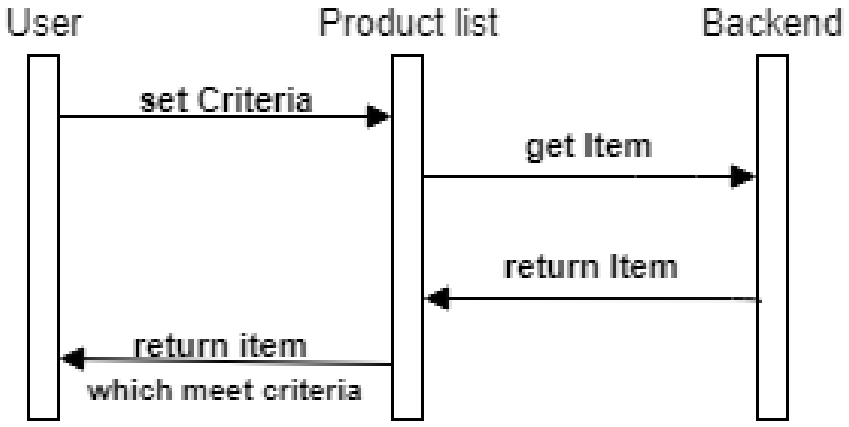
B. Product

- Product_id : 제품이 가지고 있는 고유 아이디 정보
- Product_title : 사용자가 물품을 올릴 때 설정한 제목
- Product_price : 사용자가 물품을 올릴 때 설정한 가격

C. Criteria

- Place : 물품 거래 위치
- Price : 물품 가격 범우
- Type : 물품 종류
- Period : 물품 사용기간 정보

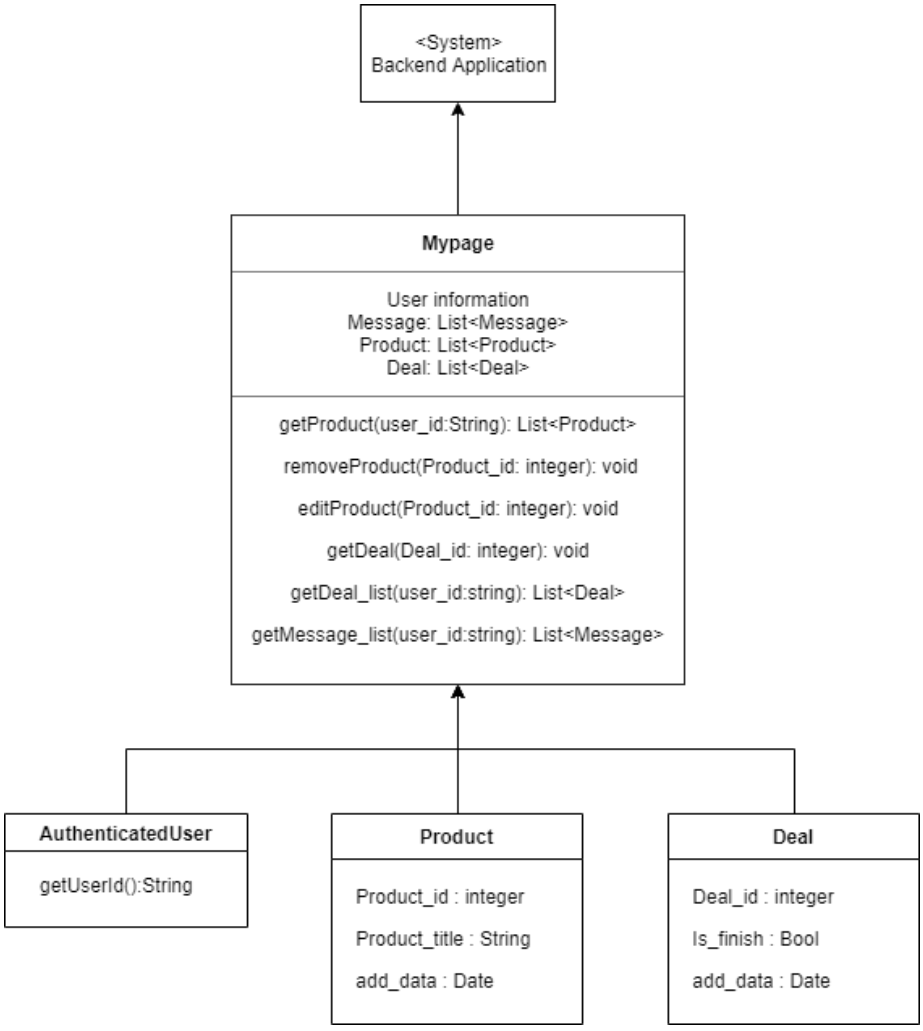
3. Sequence Diagram



[그림 14] Product List System Class Diagram

iii. Mypage System

1. Class Diagram



[그림 15] Mypage system class diagram

2. Mypage 객체정보

A. Mypage

i. Attributes

- User information(이름, 주소, 휴대폰 번호)
- Message : 메신저 창
- Product : 물품 목록

- deal:(진행중인 거래 목록

ii. Method

- removeProduct(Product_id: integer): 사용자가 올린 물품을 삭제하는 메소드
- editProduct(Product_id: integer): 사용자가 올린 물품을 수정하는 메소드
- getProduct(user_id:string): 사용자가 올린 물품 목록을 가져오는 메소드
- getDeal(Deal_id: integer): 특정 거래 상황을 가져오는 메소드
- getDeal_list(user_id:string): 사용자가 진행중인 거래 목록을 가져오는 메소드
- getMessage_list(user_id:string): 사용자가 진행중인 메시지 목록을 가져오는 메소드

B. AuthenticatedUser : 인증 유저 객체

C. Product : 물품 객체

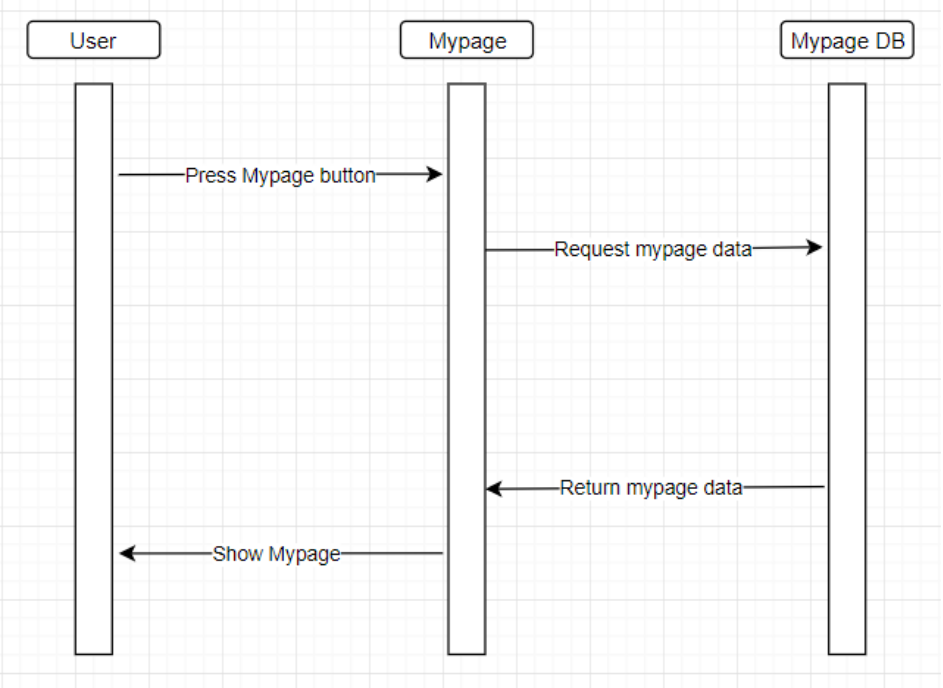
i. Attributes

- Product_id : 물품 id
- Product_title : 물품을 올린 제목
- add_data : 추가한 일자

D. Deal : 물품거래 객체

i. Attributes

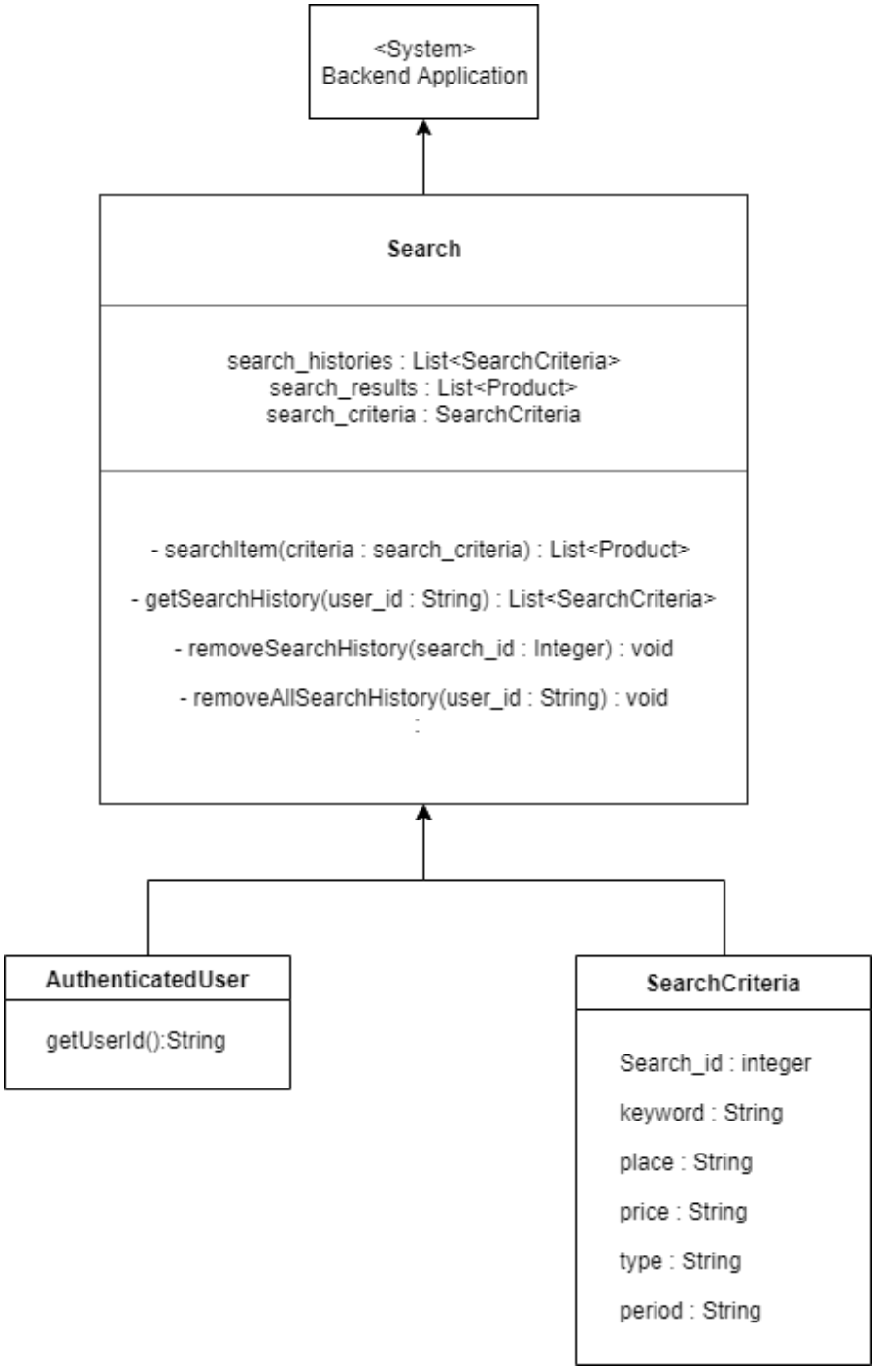
- Deal_id : 거래 id
- add_data : 추가한 일자
- Is_finish : 종료된 거래인지 아닌지 확인



[그림 16] Mypage system sequence diagram

iv. Search System

1. Class Diagram



[그림 17] Search System Class Diagram

2. Search System 객체정보

A. Search – 검색페이지 객체

i. Attributes

- search_histories : 사용자의 검색 기록
- search_results : 사용자의 검색 결과
- search_criteria : 현재 검색 조건

ii. Method

- searchItem(criteria : search_criteria) : 현재 검색 조건으로 물품을 검색하는 메소드
- getSearchHistory(user_id : String) : 사용자의 검색 기록을 가져오는 메소드
- removeSearchHistory(search_id : Integer) : 사용자의 특정 검색 기록을 지우는 메소드
- removeAllSearchHistory(user_id : String) : 사용자의 전체 검색 기록을 지우는 메소드

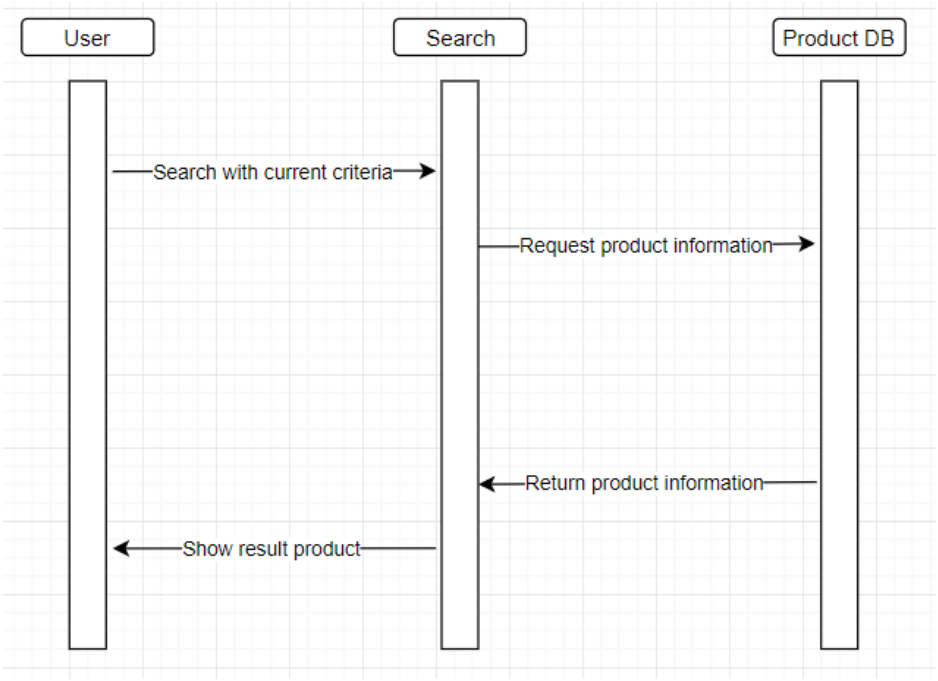
B. AuthenticatedUser : 인증 유저 객체

C. Search Criteria : 검색 조건 객체

i. Attributes

- Search_id : 검색 기록 id
- keyword : 검색할 때 쓰인 키워드
- place : 물품 거래 위치
- price : 물품 가격 범위
- type : 물품 종류
- period : 물품 사용시간

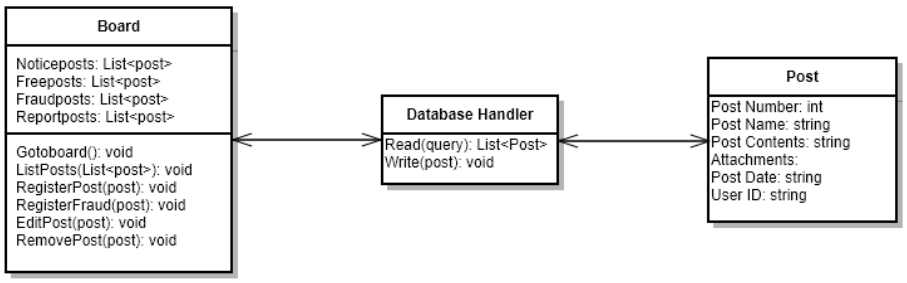
3. Sequence Diagram



[그림 18] Search System Sequence Diagram

v. Board System

1. Class Diagram



[그림 19] Board System class diagram

2. Board System 객체정보

A. Board: User 가 접근할 수 있는 Board page 에 대한 class 이다. 공지사항, 자유게시판, 신고게시판의 정보를 가지며, 각 게시판으로 이동하거나, 게시물을 등록/수정/삭제가 가능하다.

i. attributes

- Noticeposts: 공지사항의 게시물들이 list 형태로 저장된 data.
- Freeposts: 자유게시판의 게시물들이 list 형태로 저장된 data.
- Fraudposts: 신고게시판의 게시물들이 list 형태로 저장된 data.

ii. methods

- Gotoboard(): 공지사항, 자유게시판, 신고게시판으로 이동한다.
- ListPosts(List<post>): 각 게시판에 대한 게시물들을 listing 하여 보여준다.
- RegisterPost(post): post 게시물을 해당하는 게시판(공지사항, 자유게시판)에 등록한다. 게시물은 해당하는 List<post>에 추가된다.
- RegisterFraud(post): post 게시물을 신고게시판에 등록한다. 여기서는 List<post>에 바로 추가되는 것이 아니라 Reportposts 에 추가되어 관리자가 심사하여 Fraudposts 에 추가한다.
- EditPost(post): post 게시물이 자신의 것인지 확인하고, 자신의 게시물이 맞다면 게시물을 수정한다.
- RemovePost(post): post 게시물을 삭제한다.

B. Database Handler: 데이터베이스에서 게시물 자료들을 읽어오거나, 게시물의 내용을 등록, 수정할 수 있도록 DB 에 접근하는 클래스.

i. attributes

- 해당사항 없음.

ii. methods

- Read(): Board DB 에서 post 들을 list 형태로 읽어온다.
- Write(post): Board DB 에 post 내용을 추가한다.

C. Post: 게시물 1 개를 표현하는 class 이다.

i. attributes.

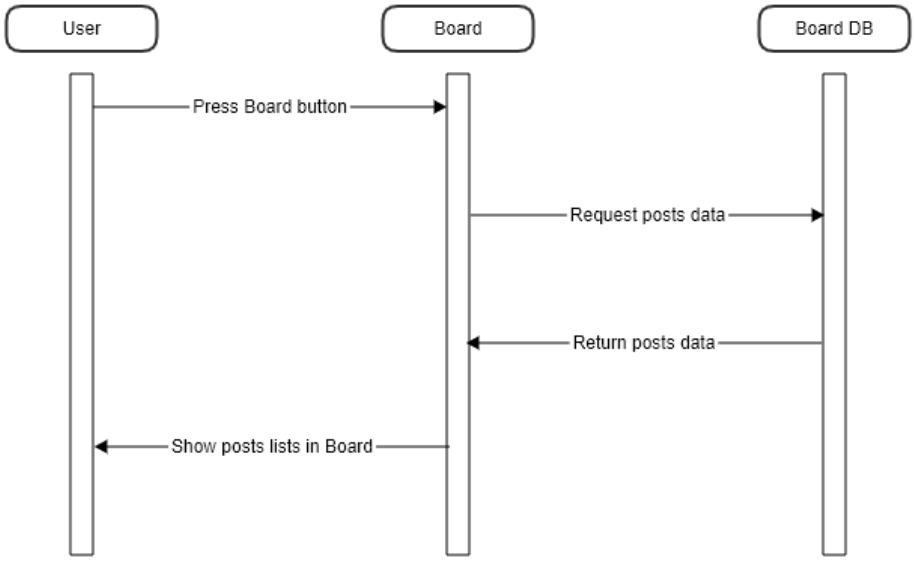
- Post Number: 게시물 번호.
- Post Name: 게시물 제목.
- Post Contents: 게시물 내용.
- Attachments: 첨부파일.
- Post Date: 게시물 게시 일자.
- User ID: 게시하는 User ID.

ii. methods.

- 해당사항 없음.

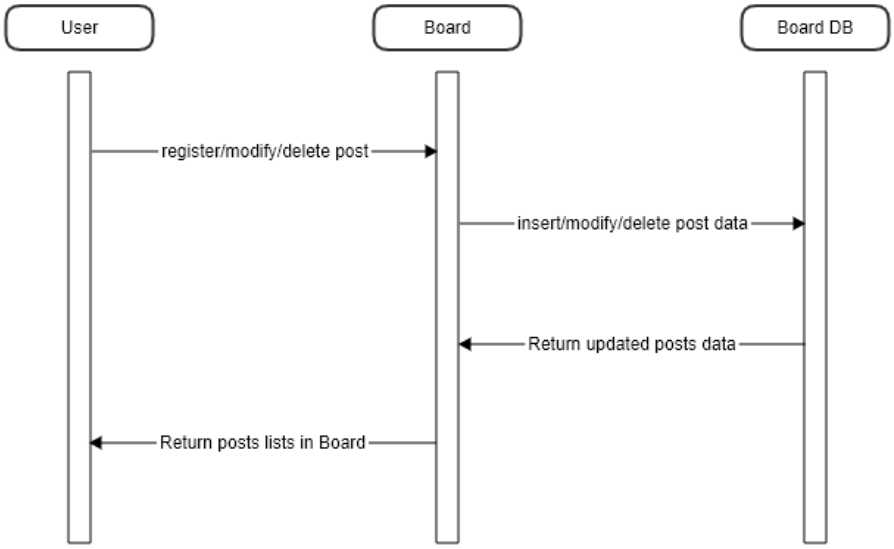
3. Sequence Diagram

A. List posts in Board sequence. (게시물 list sequence)



[그림 20] Board System Sequence Diagram _1

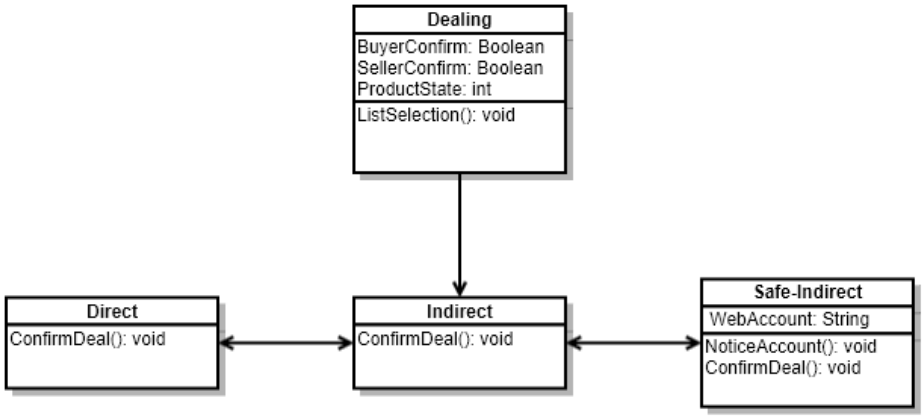
B. register/modify/delete post sequence



[그림 21] Board System Sequence Diagram _2

vi. Dealing System

1. Class Diagram



[그림 22] Dealing System Class Diagram

2. Dealing System 객체정보

A. Dealing: 3 가지 거래 방법에 따라 각 화면으로 이동할 수 있게 한다.

i. A. attributes

- BuyerConfirm: 구매자의 구매 확정을 나타내는 data.
- SellerConfirm: 판매자의 판매 확정을 나타내는 data.
- ProductState: 상품의 판매 상태를 나타내는 data. (판매중, 거래중, 판매완료로 나누어짐)

ii. B. methods

- ListSelection(): Direct/Indirect/Safe-indirect dealing 중 하나를 선택할 수 있게 한다.

B. Direct: 직거래 방법이다. 구매자/판매자는 messenger 로 소통하여 거래를 진행하고, 구매/판매 확정이 이루어지면 거래가 끝나게 된다.

i. A. attributes

- 해당사항 없음.

ii. B. methods

- ConfirmDeal(): 구매자와 판매자의 확정이 이루어지면 ProductState 를 판매 완료로 바꾼다.

C. Indirect: Indirect 거래 방법이다. 구매자/판매자는 messenger 로 소통하여 입금/배송을 진행하고, 구매/판매 확정이 이루어지면 거래가 끝나게 된다.

i. attributes

- 해당사항 없음.

ii. methods

- ConfirmDeal(): 구매자와 판매자의 확정이 이루어지면 ProductState 를 판매 완료로 바꾼다.

D. Safe-Indirect: Safe-Indirect 거래 방법이다. 구매자는 WebAccout 계좌로 입금, 판매자는 구매자에게 배송을 진행하고 구매/판매 확정이 이루어지면 거래가 끝나게 된다.

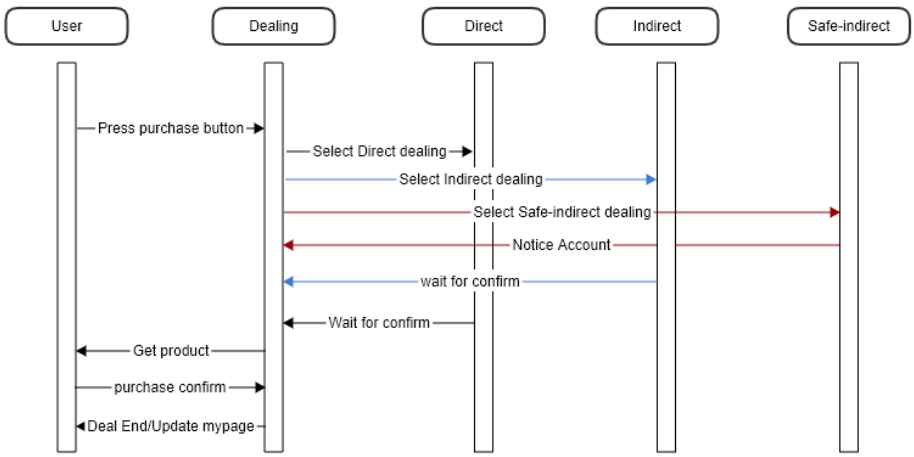
i. attributes

- WebAccount: 관리자의 계좌번호 data.

ii. methods

- NoticeAccount(): Safe-Indirect 를 진행하는 경우 관리자의 계좌번호를 화면에 띄운다.
- ConfirmDeal(): 구매/판매 확정이 이루어지면 ProductState 를 판매 완료로 바꾼다.

3. Sequence Diagram



[그림 23] Dealing System Sequence Diagram

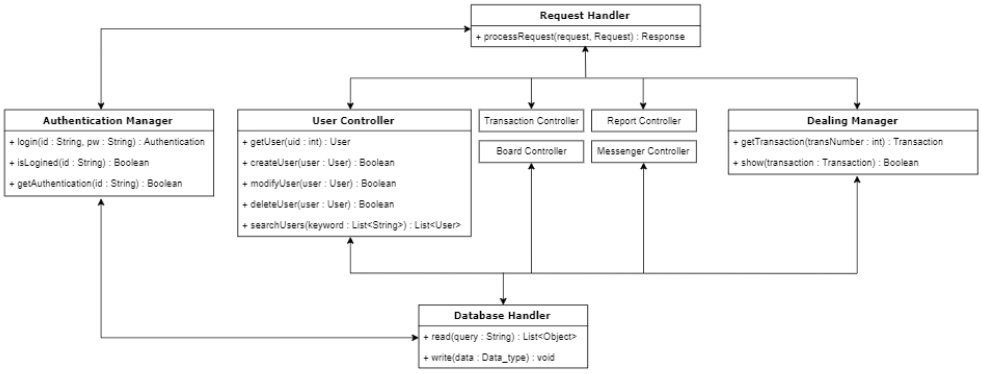
5. System Architecture – Backend

A. Objective

이 부분에서는 전체 시스템 중 사용자와의 상호작용을 담당하는 frontend 를 제외한 Backend 시스템과 각 서브시스템의 구조에 대해 설명한다.

B. Subcomponents

i. Application Server



[그림 24] Application Server

1. Request Handler : Frontend에서 넘겨준 내용을 확인하고 각 controller로 넘겨준 이후 돌아온 응답을 전송하는 객체

   - ProcessRequest(request: Request) : Response : 서버로 들어온 Frontend request를 각 Controller로 넘겨주고, 응답을 반환

2. 각 Contoller

   - getObject(object : object_type) : 각 Object를 가져옴
   - create
   - Object(object : object_type) : Object를 새로 만듬
   - modifyObject(object : object_type) : 존재하는 object에 대해 수정
   - deleteObject(object : object_type) : 존재하는 object를 삭제
   - searchObjects(keyword : List<String>) : 검색조건에 맞는 object를 검색하여 조건에 맞는 object 리스트를 반환

3. Authentication Manager : login 과정에서 인증을 수행

   - login(id : String, pw : String) : ID와 PW로 로그인하고, 인증 토큰을 발행
   - isLogined(id : String) : 해당 ID가 로그인 한 상태인지 확인
   - getAuthentication(id : String) : 해당 ID에 대한 사용자의 정보를 불러옴

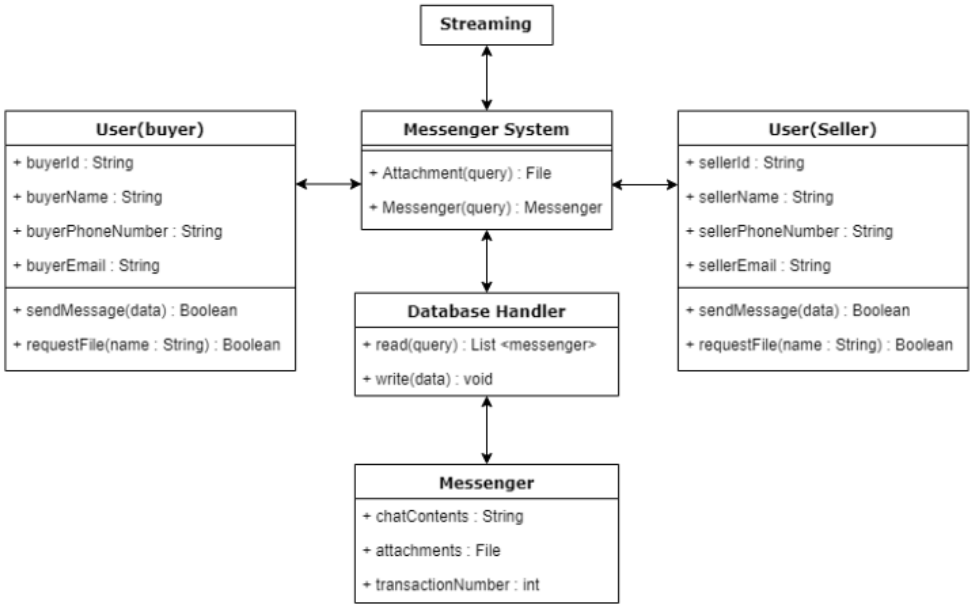
4. Dealing Manager : 결제 정보를 조회 및 보여줌

- getTransaction(transNumber : int) : input Transaction number에 해당하는 Transaction을 가져옴
- show(transaction : Transaction) : 해당 Transaction에 관한 정보를 출력

5. Database Handler
   - read(query : String) : Database에 해당 query를 수행
   - write(data : Data_type) : Database에 data를 저장

ii. Messenger System

1. Class Diagram



[그림 25] Messenger system class diagram

2. Class Description

A. Streaming : 연동되는 시스템인 Streaming 시스템

B. User : Messenger에 참여하는 사용자.
   - userId : 사용자의 ID
   - userName : 사용자의 이름
   - userPhoneNumber : 사용자의 전화번호

- userEmail : 사용자의 e-mail 주소
- sendMessage(data) : 상대방에게 전송하고자 하는 메시지를 전송
- Messenger(query) : DB handler에 현재 Messenger 정보를 불러오고 채팅을 수행한다.

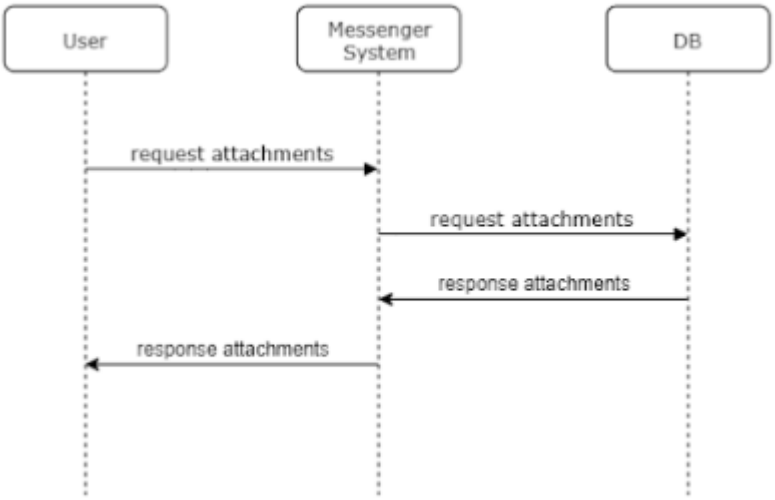
C. Messenger System : messenger를 관리하는 시스템. 열려 있는 Messenger를 조회하고 파일 탐색을 지원한다.
- Attachment(query) : 필요로 하는 파일을 확인하고, DB Handler를 통해서 실제 파일을 확인한 후 파일을 사용자에게 보여준다.
- Messenger(query) : 진행중인 Messenger 목록을 확인하고, query에 해당하는 Messenger를 불러온다.

D. Database Handler : 생성된 Messenger의 정보를 저장하도록 Database에 요청하고, Messenger System으로부터 들어온 request에 대한 response를 반환한다.
- read(query) : data request를 받아 Database에 해당 정보를 요청하고, 정보를 System에 반환한다.
- write(data) : 새로운 파일이나 생성된 Messenger를 Database에 저장한다.

E. Messenger : Database에 저장된 현재 생성된 Messenger들에 대한 정보를 나타낸다.

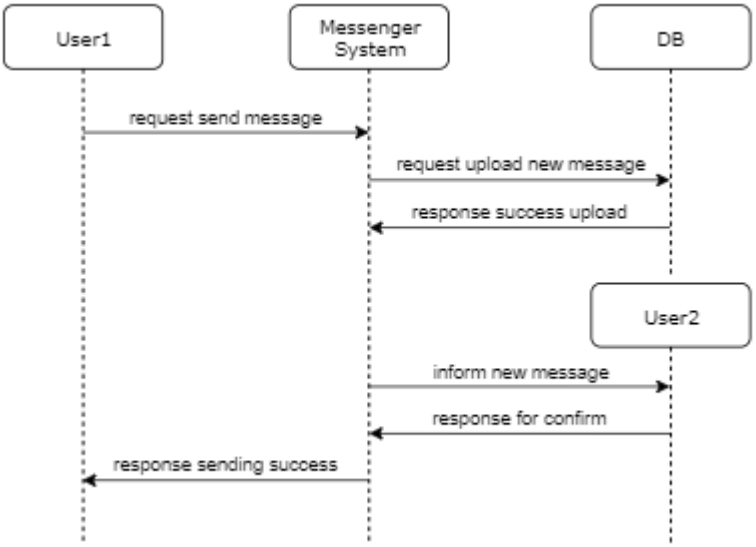
3. Sequence Diagram

A. Attachment request



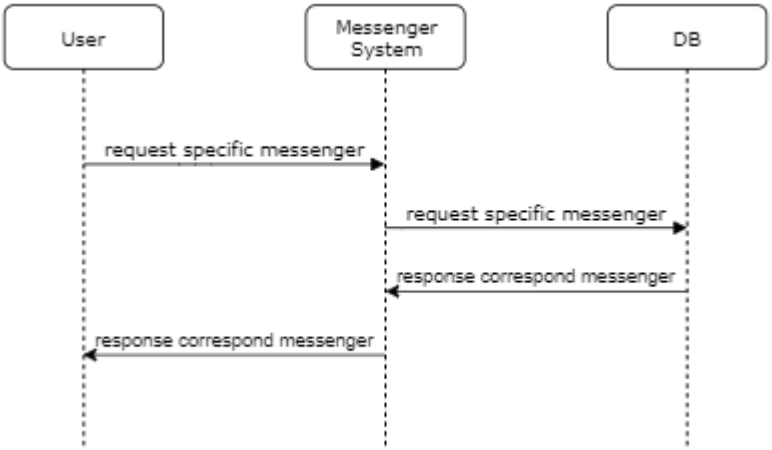
[그림 26] Messenger system Sequence diagram_1

B. Send message



[그림 27] Messenger system Sequence diagram_2

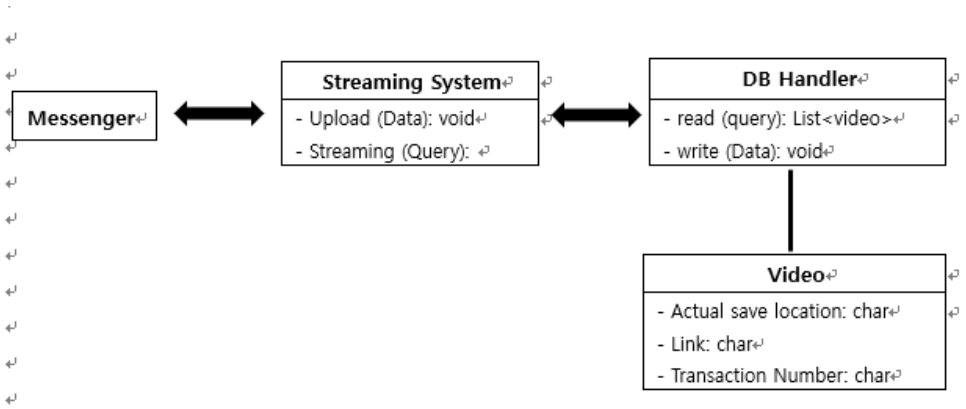
C. Bring messenger information



[그림 28] Messenger system Sequence diagram_3

iii. Streaming System

Messenger 에서 물품 정보를 공유하기 위해서 사용하는 동영상을 스트리밍하는 서비스이다.



[그림 29] Streaming system Class Diagram

1. Class Description

A. Messenger: 연동되는 시스템인 Messenger 시스템
B. Streaming System: 스트리밍을 통제하는 시스템. 비디오의 업로드, 스트리밍을 지원한다.
   i. Upload: Messenger에서 들어온 비디오 업로드 요청을 받고 DB Handler에 데이터 저장을 요청한다.
   ii. Streaming: 요청받은 영상을 확인하고, DB Handler를 통해서 실제

파일을 확인한 이후, 스트리밍을 진행한다.

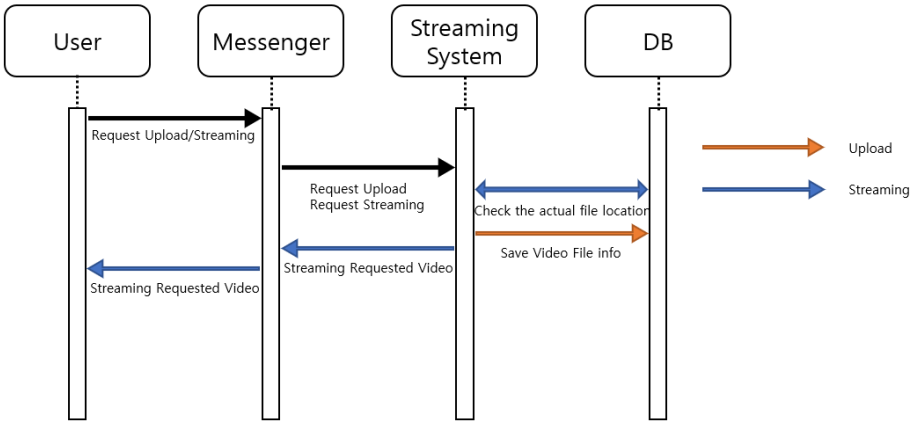
C. DB Handler: 실제 저장된 영상의 정보들을 저장하도록 DB에 요청하고, Streaming System에서 들어온 정보 요청을 반환한다.

i. read: Streaming System에서 들어온 데이터 요청을 받고 DB에서 정보를 요청, 받아서 System에 반환한다.

ii. write: Upload 요청을 통해서 저장되는 영상의 정보들을 DB에 저장한다,

D. Video: 서버에 저장되는 영상과 관련된 정보들로, DB에 저장되는 정보들을 나타낸다.

2. Sequence Diagram



[그림 30] Streaming System Sequence Diagram

6. Protocol Design

네이버 로그인 인증 요청은 앱에 네이버 로그인 화면을 띄우는 역할을 한다. 사용자가 네이버 회원 인증에 성공하면 API 로부터 받은 code 값을 이용하여 접근 토큰 발급 요청을 하게 된다. 요청의 결과로 접근 토큰(access token)을 받게 되고, 이 토큰이 만료되기 전까지 네이버 로그인을 유지하게 된다. 만약 토큰이 만료된다면, 접속 토큰을 발급 받을 때 같이 받았던 refresh_token 을 이용하여 접근 토큰을 다시 발급 받을 수 있다.

A. 네이버 아이디로 로그인 인증 요청

- Request

Method	Post	
URL	https://nid.naver.com/oauth2.0/authorize	
Request Body	response_type	인증과정에 대한 내부 구분 값(ID, PW 등)
	client_id	App 등록시 발급받은 Client ID 값
	redirect_url	App 등록시 입력한 Callback URL(URL 인코딩 적용)
	state	Cross-site request forgery 공격을 방지하기 위해 app 에서 생성한 상태 토큰 값으로 URL 인코딩 적용한 값

[표 1] 네이버 아이디 인증요청_1

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	code	네이버 아이디로 로그인 인증에 성공하면 반환 받는 인증 코드
	state	Cross-site request forgery 공격을 방지하기 위해 app 에서 생성한 상태 토큰 값으로 URL 인코딩 적용한 값
Failure Response Body	state	Cross-site request forgery 공격을 방지하기 위해 app 에서 생성한 상태 토큰 값으로 URL 인코딩 적용한 값
	error	에러 코드
	error_description	에러 메시지

[표 2] 네이버 아이디 인증요청_2

B. 접속 토큰 발급 요청

- Request

Method	Post	
URL	https://nid.naver.com/oauth2.0/token	
Request Body	grant_type	인증과정 구분 값(발급 : authorization_code)
	client_id	App 등록시 발급받은 Client ID 값
	client_secret	App 등록시 발급받은 Client secret 값

	code	로그인 인증 요청 API 호출에 성공하고 리턴받은 인증 코드 값
	state	Cross-site request forgery 공격을 방지하기 위해 app 에서 생성한 상태 토큰 값으로 URL 인코딩 적용한 값

[표 3] 접속토큰 발급요청_1

- Response

Success Code	200 OK	
Failure Code	404 Not Found	
Success Response Body	access_token	접속 토큰, 발급 후 expires_in 파라미터에 설정된 시간이 지나면 만료
	refresh_token	갱신 토큰, 접근 토큰이 만료될 경우 접큰 토큰을 다시 발급받을 때 사용
	token_type	접근 토큰의 타입으로 Bearer 와 MAC 의 두가지를 지원
	expires_in	접근 토큰의 유효 기간(초 단위)
Failure Response Body	error	에러 코드
	error_description	에러 메시지

[표 4] 접속토큰 발급요청_2

7. Database Design

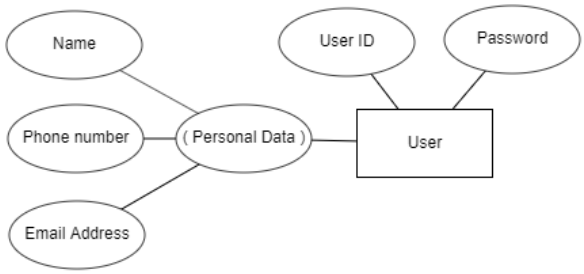
A. Objective

System Requirement Specification 을 통해서 도출된 각 sub-system 및 service 들의 요구사항을 기반으로 하여 데이터베이스 설계를 기술한다. ER Diagram 을 통해서 Entity 와 Entity 간의 관계를 기술하고 Relational Schema 를 만든다. SQL DDL 은 사용하기로 결정한 WSGI framework 인 Django 에서 application 별로 별도 처리하기 때문에 기술하지 않는다.

B. ER Diagram

본 단락에서는 각 Entity 와 Entity 의 내용들을 먼저 기술하고, 이후에 전체 ER Diagram 을 통해서 각 Entity 별 관계를 기술한다. ER Diagram 의 기술은 표준적인 ER Diagram 의 기술 방향을 따른다.

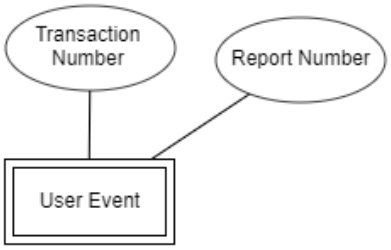
i. User



[그림 31] Database_user

사용자의 정보를 표현한다. 사용자의 개인정보와 인증에 사용할 user id, password 를 포함한다.

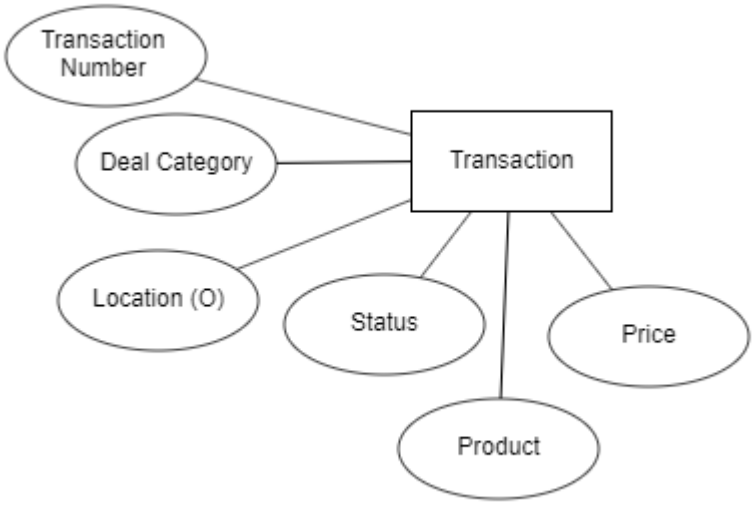
ii. User Event



[그림 32] Database_UserEvent

각 유저의 거래 내역과 신고 내역을 표현하는 Entity 이다. 거래 내역과 신고 내역이 담겨 있다.

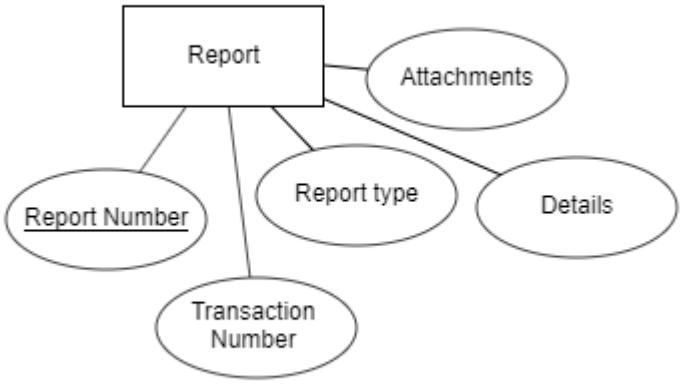
iii. Transaction



[그림 33] Database_transaction

시스템 내에서 진행된 모든 거래 정보가 담겨 있다. 거래 상태, 거래 당사자들(Seller/Buyer), 거래 종류, 직거래시 직거래 장소, 물품 종류, 가격이 담겨 있다. 거래 이전에는 Customer 가 None 으로 되어 있는 상태로 저장된다.

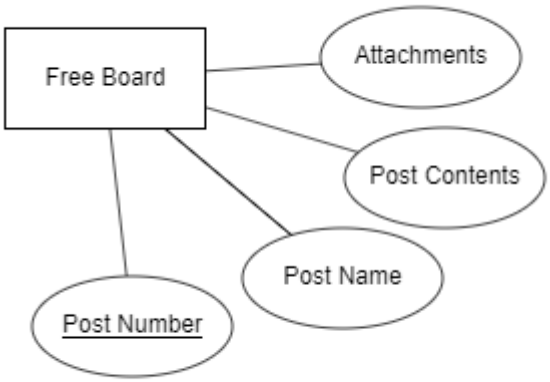
iv. Report



[그림 34] Database_Report

비정상 거래 발생시 거래 당사자가 신고한 내용을 표현하는 Entity 이다. 신고 유형, 신고 대상, 처리 상태, 내용 상세, 그리고 첨부 파일로 이루어져 있다.

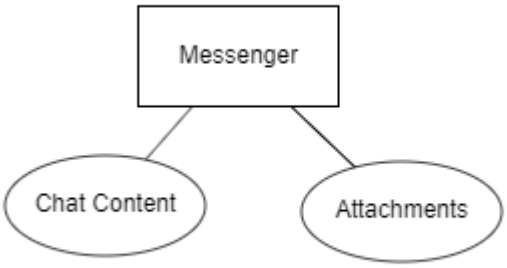
v. Free Board



[그림 35] Database_FreeBoard

자유 게시판을 저장하는 Entity 이다. 작성자, 제목, 내용으로 구성되어 있다.

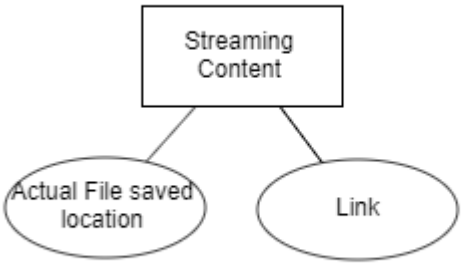
vi. Messenger



[Figure 36] Database_Messenger

거래 도중에 Seller 와 Customer 간 1:1 대화 내용을 표현하는 Entity 이다. 대화 참여자와 대화 내용, 대화간 첨부파일이 여기에 해당된다.

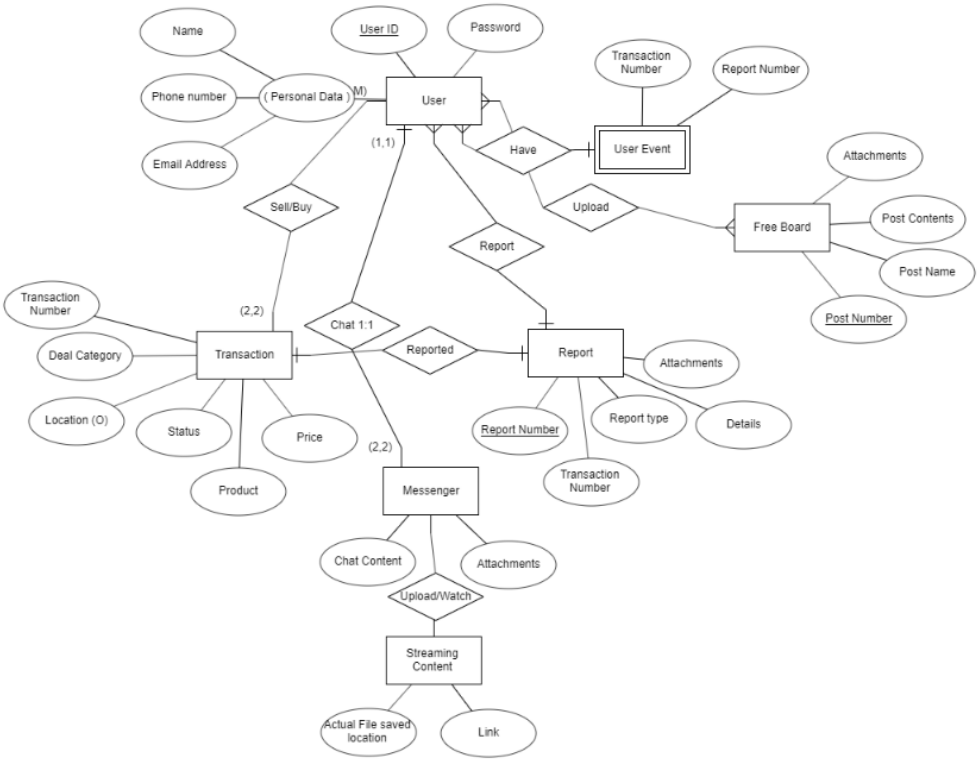
vii. Streaming Video



[그림 37] Database_StreamingVideo

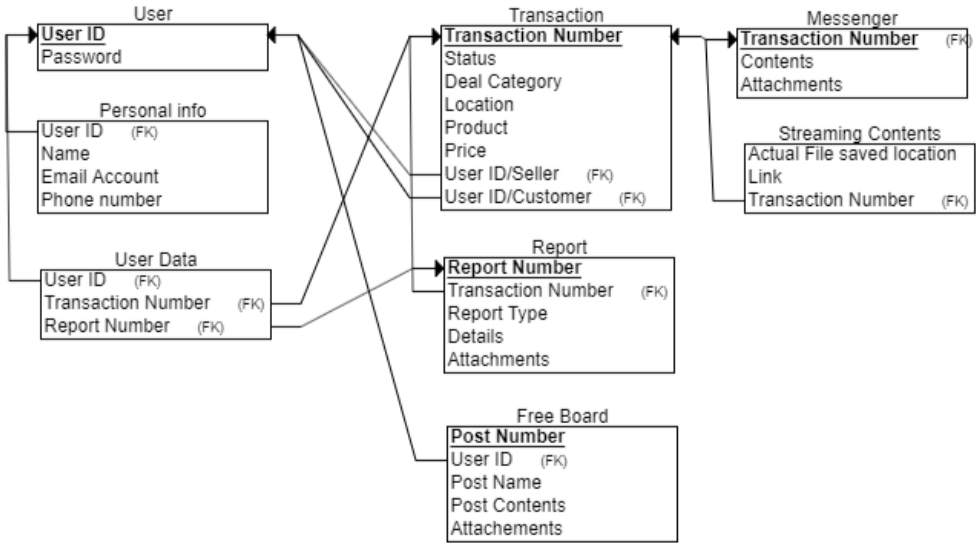
Messenger 등의 서비스에서 사용하는 영상들을 관리하는 Entity 이다. 공유 대상과 링크 URL, 실제 미디어의 저장 장소가 담겨 있다.

viii. Overall ER Diagram



[그림 38] Overall ER Diagram

C. Relational Schema



[그림 39] Relational Schema

8. Testing Plan

A. objective

Testing Plan 은 시스템 내부에 결함이 있는지 확인하고, 의도한대로 실행이 되는지 확인하기 위한 것이다. 이를 위해, test policy 에 맞게 여러 test 들을 설계한 후, 각각의 test case 들에 대하여 test 를 진행하여야 한다. 이 때, Testing Policy 에서는 testing 에 대한 단계적 접근을 설명하며, Test Case 는 각 Sub-System 에서 사용자를 판별하고 그에 따라 시스템에 기대하는 입,출력 동작을 서술한다.이번 장에서는, 이러한 testing policy 및 test case 들에 대해서 기술한다.

B. Testing Policy

i. Development Testing

Development testing 은 unit testing, component testing, system testing 으로 나뉜다. Unit testing 은 각각의 프로그램의 단위, 모듈, 클래스 들을 testing 한다. 이는 각 unit 을 개발하는 개발자들이 직접 testing 해야 하며, 검증되면 commit 한다.

Component testing 은 unit 들이 모여서 만들어진 component 를 testing 한다. 이 또한, 각 component 를 개발한 개발자들이 직접 testing 하며, unit 간의 protocol, component 의 interface 등을 testing 한다.

System testing 은 sub-system 단위로 testing 을 진행한다. 이 때, 각 sub-system 요소들을 incrementally integrating 하며 testing 을 진행한다. 각각의 sub-system 이 다 합쳐진 이후에는, 전체 system 에 대하여 최종 testing 을 한다.

ii. Release Testing

Release testing 은 해당 시스템이 고객의 요구사항을 충족시키면서 개발되었는지 확인하는 testing 이다. 이를 위해, Requirement Specification 을 참고하여, 이에 서술된 각각의 요구사항들에 대하여 모두 비교하며 testing 을 진행한다. 이는 보통 Development testing 이 끝난 후 개발팀에서 진행한다.

iii. User Testing

User testing 은 사용자들이 직접 참여하여 사용자 환경에서 시스템을 testing 하는 것이다. User testing 은 Alpha testing, Beta testing, Acceptance testing 이 있다. 보통 Release testing 이 끝난 후 User testing 을 진행하며, 효과적인 testing 을 위해서는 실제 구매자들을 대상으로 testing 을 진행하는 것이 좋다.

C. Test Case

i. Search System

1) 사용자는 원하는 물품에 대한 카테고리를 선택한 후, 검색어를 입력한다.

2) 사용자는 검색 버튼을 누른다.

2-1) 일치하는 물품이 없으면 물품이 없다는 알람을 띄운다.

3) 해당 물품을 띄워 준다.

ii. Mypage System

- product management

1) 사용자는 자기가 등록한 판매 물품 리스트를 확인한다.

2) 사용자는 올린 물품 내용을 삭제하거나 수정한다.

3) 만약 물품을 새로 등록하고 싶다면, 물품을 추가한다.

4) 새롭게 적용된 물품 리스트가 mypage 에 띄워진다.

- Deal management

1) 사용자는 자신의 거래 목록을 확인한다.

2) 진행중인 거래가 있다면, 눌러서 상세정보를 확인할 수 있다.

3) 완료된 거래를 확인하고 싶어도, 눌러서 상세정보를 확인할 수 있다.

9. Development plan

A. Objectives

이번 단락에서는 구현 단계에서 사용하는 각종 프레임워크 및 라이브러리 등을 다루고 향후 개발 일정과 진행 방향을 설명한다.

B. Frontend Development

i. Vue.js

[그림 40] logo of Vue.js

Vue.js 는 자바스크립트 기반의 프론트엔드 프레임워크로서, 기존의 다른 프론트엔드 프레임워크에 비해서 간편하기 시작하기 쉽다는 장점을 가지고 있다. 마찬가지로 사용 후보군에 속했던 React.js 는 JSX 를, Vue 는 template 를 기반으로 활용하기 때문에 웹 기반 지식 혹은 경험이 부족하더라도 기여하기 쉽다는 점을 가지고 있다. 또한 React 에 비해 더 발전된 서버 사이드 렌더링을 지원한다. 마지막으로 다른 라이브러리/프레임워크에 비해서 매뉴얼의 한글화가 잘 되어 있다. 따라서 Vue.js 를

기반으로 프론트엔드를 개발함으로서 프론트엔드를 배우는데 드는 시간을 최소화함과 동시에 Backend 로 사용할 Django 와의 원할한 연동을 노린다.

ii. Nuxt.js

[그림 41] Logo of Nuxt

React.js 의 Next.js 에서 영감을 받아서 개발된 Nuxt.js 는 Vue.js 의 각종 기능들을 편리하게 이용할 수 있도록 하고, 서버와 클라이언트 사이를 개발하는데 도움이 되는 기능들을 제공하는, Vue.js 와 같이 활용 가능한 프론트엔드 프레임워크이다. Nuxt.js 를 이용해 서버 사이드 렌더링 등 다양한 기능들을 구현할 계획이다.

iii. Node.js

[그림 42] Logo of Node js

Chrome V8 자바스크립트 엔진으로 빌드된 자바스크립트 런타임이다. 주로 서버사이드를 개발하기 위하여 활용되지만, 이번 프로젝트에서는 위의 Vue.js 와 Nuxt.js 를 원활하게 사용하기 위하여 사용할 예정이다.

C. Backend Development

i. Python

[그림 43] Logo of Python

Python 은 객체 지향 프로그래밍 언어로서 분야를 막론하고 활용도가 높게 사용되는 프로그래밍 언어이다. 배우는 기간이 적고, 다방면에서 활용 가능한 방대한 양의 라이브러리를 장점으로 가지고 있다. Flask, Django 등의 서버 사이드 웹 프레임워크가 python 을 기반으로 개발되었다.

ii. Django

[그림 44] Logo of django

Django 는 모델-뷰-컨트롤러 패턴을 따르고 있는 Python 기반의 오픈소스 웹 프레임워크이다. 고도의 데이터베이스 기반 웹사이트를 작성하는 데 있어 수고를 더는 것이 Django 의 주된 목표로, 컴포넌트의 재사용성과 플러그인화 가능성, 빠른 개발 등을 강조하고 있다. 10 년 이상 널리 활용된 웹 프레임워크로서 강력한 내장 애플리케이션과 배우기 쉬운 언어 특성, 쉽게 관련 자료를 구할 수 있다. 프로젝트에서는 Django 를 주요 Backend 프레임워크로 활용한다.

iii. Nginx

[그림 45] Logo of Nginx

Nginx 는 웹 서버 소프트웨어로, 가벼움과 높은 성능을 목표로 한다. 웹 서버, 리버스 프록시 등에 주로 활용하는 소프트웨어다. 프로젝트에서는 스트리밍 서비스와 웹 서비스 간의 로드 밸런싱에 활용할 예정이다.

D. Development Plan

Main Step	Expected	Actual
Project Proposal	10/9	10/9
Requirement Specification	10/27	10/27
Design Specification	11/10	11/13
Frontend Implementation	11/24	12/1
Backend Implementation – WCGI	11/24	12/1
Backend Implementation – Streaming Service	11/24	12/1
System integration	11/30	12/4
Acceptance test	12/8	12/8

[표 5] Development Plan

Design Specification 과정에서 시간이 더 소요되어 기존 계획에 비해 1 주 늦어졌다. Frontend 와 Backend 의 parallel development 를 통해서 기간을 단축할 예정이며, 실제 기간에 맞출 수 있도록, 또 integration 간의 문제가 발생하지 않도록 팀원들간 협업이 필요하다.

10. Index

A. 표

B. 그림 및 Diagram