빅데이터를 사용하는
통합형 e-commerce

Petkage

Design Specification

#Team2

2016310696 김소희

2014310251 김준형

2014312601 김태경

2015314070 안영민

2018314848 한승희

내용

1. Preface

1.1 Objective

preface에서는 문서의 전반적인 구조를 설명하고 독자를 설정하는데 목표를 두고 있다. 개괄적인 목적과 개요를 설명하며 본문을 읽기 전 독자들에게 문서의 흐름을 알려주게 된다.

1.2 Readership

Petkage에 관한 문서로써 독자는 개발에 참여한 엔지니어, 아키텍처들을 포함한 기술자 들과 Petkage에 대한 관심을 가지고 있는 이해관계자들 또한 독자로 설정할 수 있다.

1.3 Structure of Design specification

A. Preface

Preface에서는 본 문서의 독자를 정의하며, 문서의 세부사항들과 각 사항들의 목적을 기술한다.

B. Introduction

Introduction에서는 Petkage를 만들기 위하여 필요한 개발 툴들을 소개하고 있다. UML과 Database 그리고 이해를 돕기 위해 만들어진 diagram을 예시로 설명한다.

C. System Architecture

System Architecture: System Architecture에서는 시스템의 개요와 세부 기능에 대해 서술하고 있다. System architecture에서는 block diagram 사용하였다. 세부 시스템에서는 class diagram과 state diagram 그리고 sequence diagram을 통해 전반적인 흐름과 관계에 대해 서술하였다. 이후 Petkage를 이루는 각각의 system 들은 별도의 대 제목으로 분류하여 각 시스템의 설계를 설명한다.

D. Protocol Design: Protocol design

Protocol Design: Protocol design에서는 전체적인 시스템의 동작 하에서 하위 시스템들이 따르는 소통의 규칙, 문법, 의미, 소통 동기화 및 에러 회복 방식 등에 관한 프로토콜에 대해 서술한다.

E. Database Design: Database design

Database Design: Database design은 요구사항 명세서를 기반으로 한 세부적인 데이터베이스 설계를 기술한다. ER Diagram을 통해 개괄적인 Entity 간의 관계를 기술하고, 이를 통해 Relational Schema을 작성한 뒤, 최종적으로 SQL DDL을 작성한다.

F. Testing Plan

Testing plan는 Petkage 개발 과정에서 user 및 admin 시나리오를 바탕으로 한 다양한 테스트 계획을 세우고 실행해보는 과정을 기술하였다. 오류를 사전에 발견하기 위한 작업으로 test case에 대한 세분화가 진행된다.

G. Development Plan

Development Plan에서는 실제 시스템 개발을 위해 필요한 개발 환경과 개발 언어, 패키지, 외부 API(공공기관)등에 대하여 서술한다. 또한 개발 일정에 대한 내용도 표로 서술한다.

H. Index

Index에서는 문서를 주제별 흐름에 맞춘 인덱스들을 정리하여 보기 편리하게 정리한다.

I. Reference

Reference는 참고문헌과 사이트 목록에 대한 출처를 밝힌다.

2. Introduction

2.1 Objective

Introduction에서는 시스템의 설계에 사용한 UML(Unified Modeling Language)과 데이터베이스모델의 다양한 다이어그램작성을 위해 사용한 개발 툴을 소개한다. Petkage에서 사용할 UML Diagram은 Deployment Diagram, Class Diagram, Sequence Diagram, State Diagram 4가지와, 데이터베이스를 표현할 ER Diagram의 예시를 소개한다. 이에 추가적으로 diagram을 만드는데 이용한 applied tool에 대해서도 소개한다.

2.2 Applied Diagram

A. ER diagram

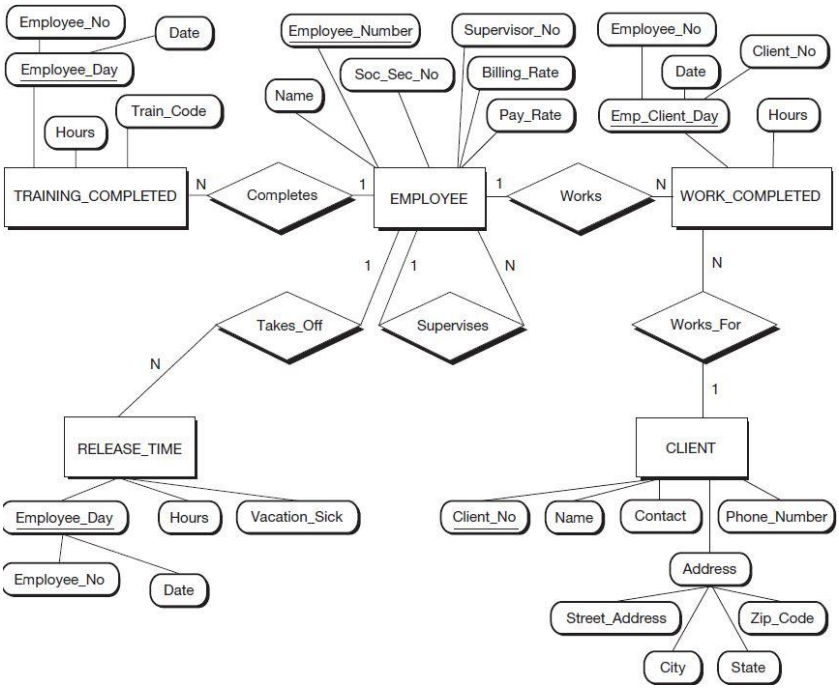

Figure1: Example of ER Diagram

ER Diagram은 각 개체가 가지고 있는 속성, 개체 사이의 관계를 나타낸다. 이 다이어그램은 데이터베이스를 설계할 때 사용된다. 개체들 간의 관계를 표현하는 것이 ER Diagram의 주요

한 특징이다. 개체는 분리된 물체 하나를 표현하는 것이다. Relationship은 두개 이상의 개체에서 나타나는 상호작용을 의미한다. ER Diagram을 바탕으로 각 subsystem의 관계를 기술한다.

B. Sequence Diagram

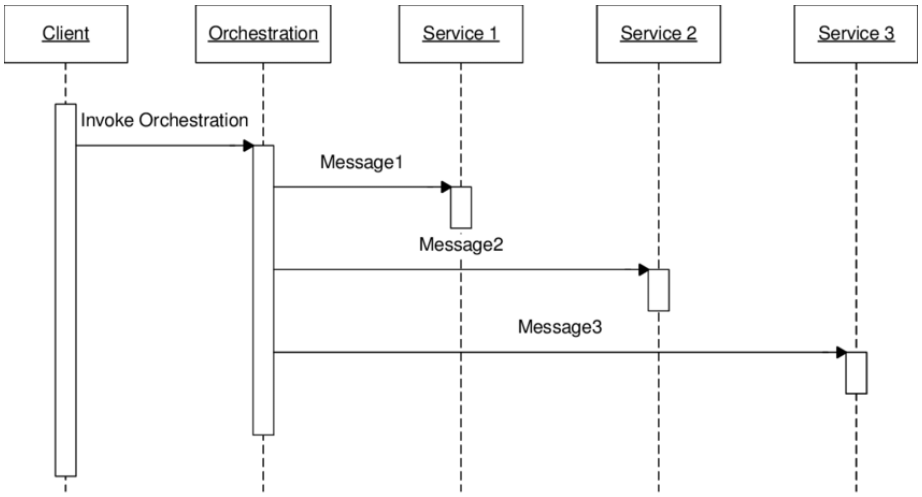

Figure2: Example of Sequence Diagram

Sequence diagram은 사용자들과 시스템들의 상호작용을 나타낸다. 순차 다이어그램은 Objects 간의 동적 상호작용을 시간의 순서로 모델링한다. Sequence Diagram의 목표는 상호작용 간의 sequence를 보이는 것으로 Use Case 간에서 프로세스들이 정확하게 만들어지고 작동하는지 보여준다. Sequence Diagram에는 한 객체간에서 일어나는 순서를 정확하게 기술해야 한다.

C. Deployment Diagram

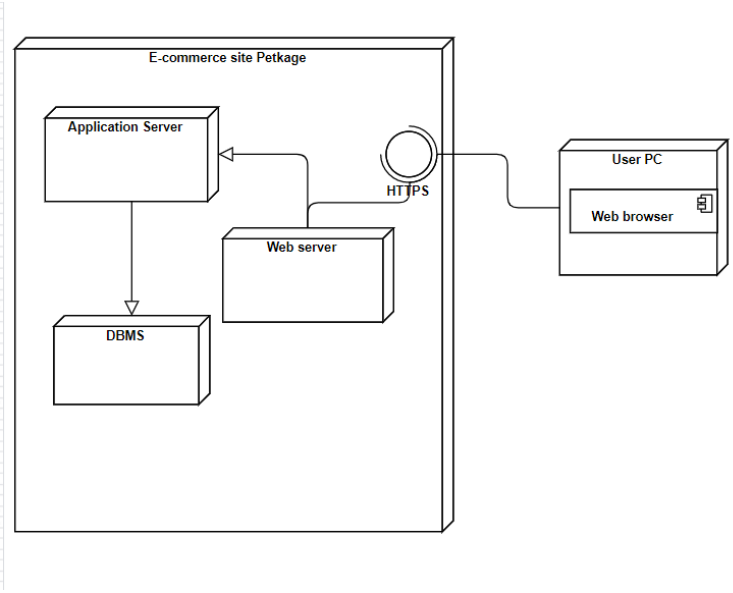

Figure3: Example of Deployment Diagram

Deployment diagram은 시스템을 구성하는 HW 자원 간의 연결 관계를 표현하고, HW 자원에 대한 SW 컴포넌트의 배치 상태를 표현한 다이어그램이다. 컴포넌트(SW)가 어떤 HW자원에 탑재되어 실행될지도 정의하며, 하드웨어 자원의 물리적 구성을 정의한다. Deployment diagram의 구성요소로는 노드와 컴포넌트 그리고 relationship이 있다.

D. Class diagram

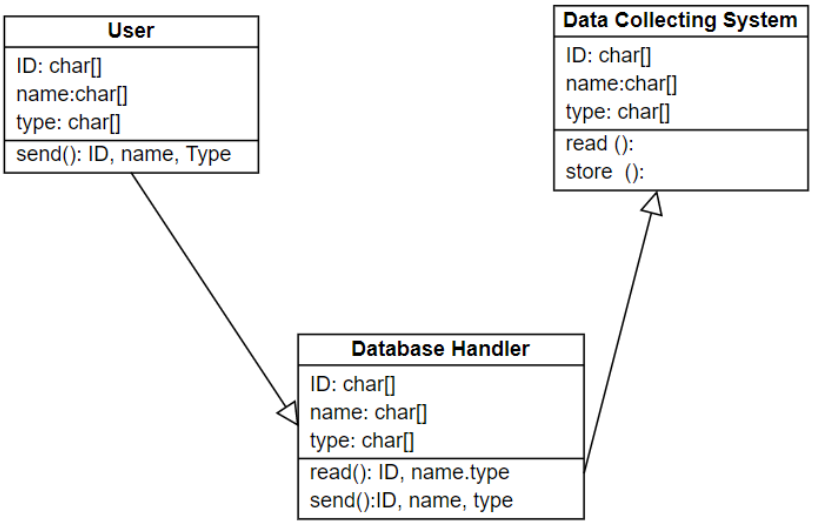

Figure4: Example of Class Diagram

클래스 다이어그램은 클래스 내부의 정적인 내용이나 클래스 사이의 관계를 표기하는 다이어그램으로 시스템의 일부 또는 전체의 구조를 나타낼 수 있다. 클래스다이어그램은 클래스들의 관계를 쉽게 보고, 의존관계를 쉽게 파악하게 해준다.

E. State Diagram

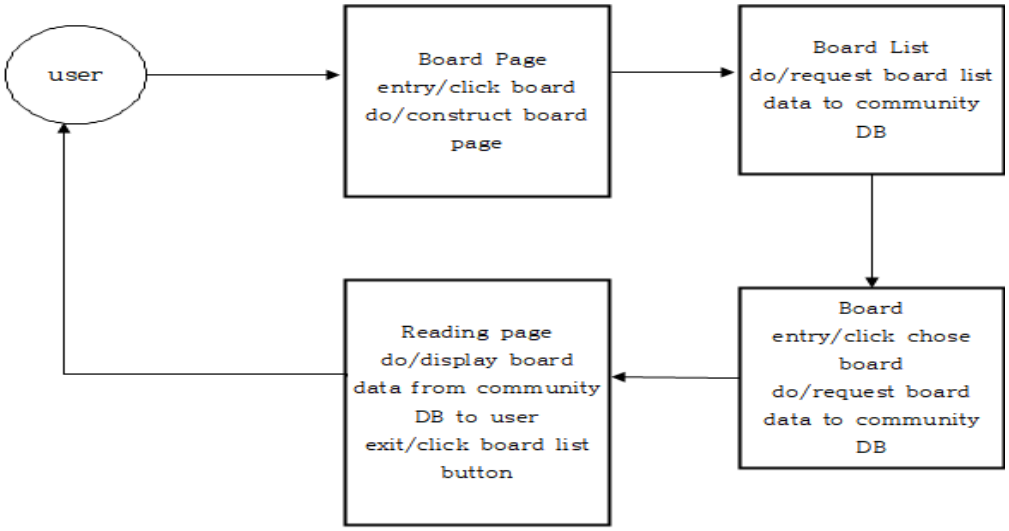

Figure5: Example of State Diagram

State Diagram은 객체 상태와 함께 객체 상태 변화를 유발하는 이벤트와 동작(Action/Activity)도 함께 정의한다. 이러한 요소가 정의된 state diagram을 통해 "객체 O는 이벤트 E에 의해 상태 S로 변화하고 그 상태에서 A라는 행위를 한다"라는 식의 분석을 수행할 수 있다.

F. Sequence diagram

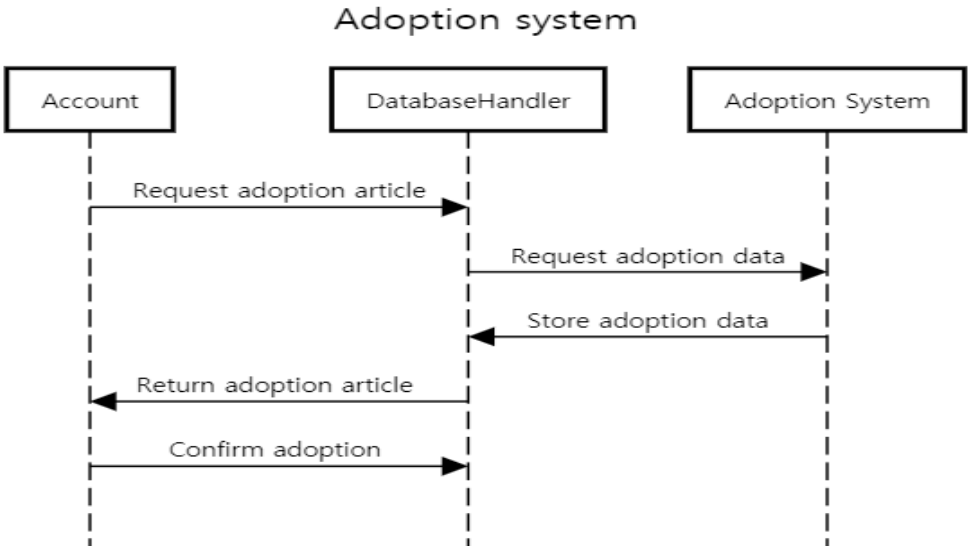

Figure6: Example of Sequence Diagram

Sequence Diagram은 객체간의 동적 상호작용을 시간적 개념을 중심으로 모델링하는 과정, 다이어그램의 수직방향이 시간의 흐름을 나타낸다. Sequence diagram에서는 객체의 오퍼레이

션과 속성을 상세히 정의해야 한다. 객체 간 상호작용을 정의하는 과정에서 객체들이 가져야 하는 오퍼레이션과 속성을 구체적으로 정의해야 하며, 객체는 다른 객체가 의뢰하는 일을 처리하는 것으로 나타낸다.

2.3. Applied Tool

A. Draw.io

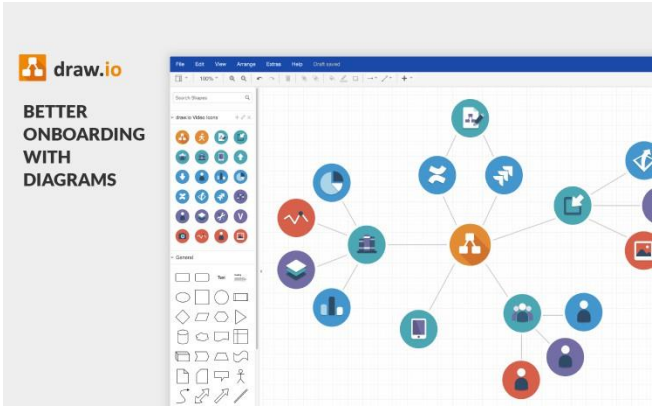

Figure7: Draw.io

현재 그림으로 첨부된 Diagram의 일부는 draw.io를 이용했다. 웹으로 손쉽게 접근할 수 있으며 제공되는 UML의 가짓수도 다양하다. Google Drive와 연동하여 사용하기 쉽고 공유를 지원하여 팀원간의 원활한 상호작용과 소통을 가능하게 한다.

B. Sequencediagram.org

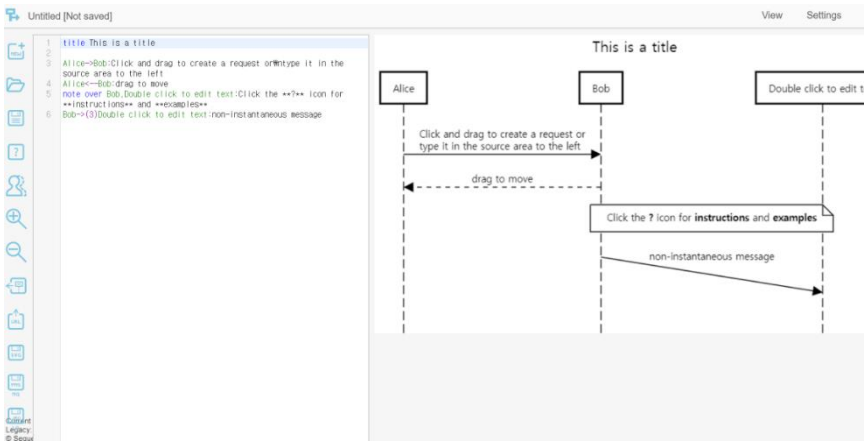

Figure8: Sequencediagram.org

현재 그림으로 첨부된 Sequence Diagram의 일부는 Sequencediagram.org를 이용했다. 웹으로 쉽게 접근이 가능하며, 무료로 이용 가능한 서비스이다. 텍스트 인풋으로 Sequence Diagram을 만들어 보다 직관적이고 빠른 이해를 가능하게 한다.

C. sparxsystems.com

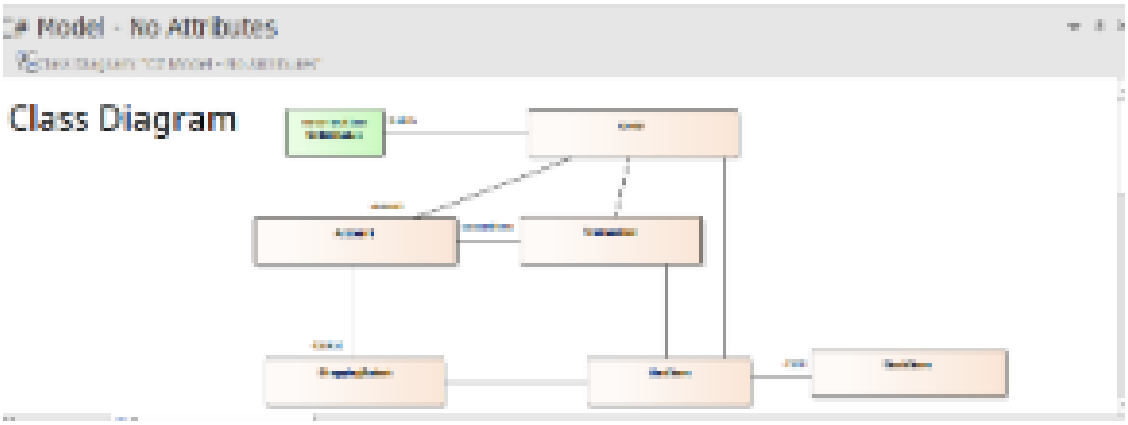

Figure 9: Sparxsystems.com

Class Diagram을 구현할 때 사용한 도구는 enterprise architect이다. 클래스 간의 상속과 관계를 표현하고 도식화하기에 개발자들이 편리하게 이용 할 수 있다. 그 외에도 도식화에 관련된 기타 diagram을 표현할 수도 있다.

3. System Architecture

3.1 Objective

System Architecture에서 만들고자 하는 시스템에 대한 전체적인 개요와 각각의 모듈들의 시스템 기능을 서술한다. 전체 시스템은 Block Diagram을 활용하여 표현하였고 서브 시스템의 관계, 실제 배포 형태는 Block Diagram, Deployment Diagram을 사용해 표현하였으며 세부적인 시스템의 설계는 Communication Diagram, Sequence Diagram, State Diagram을 사용하여 표현하였다.

3.2 System Organization

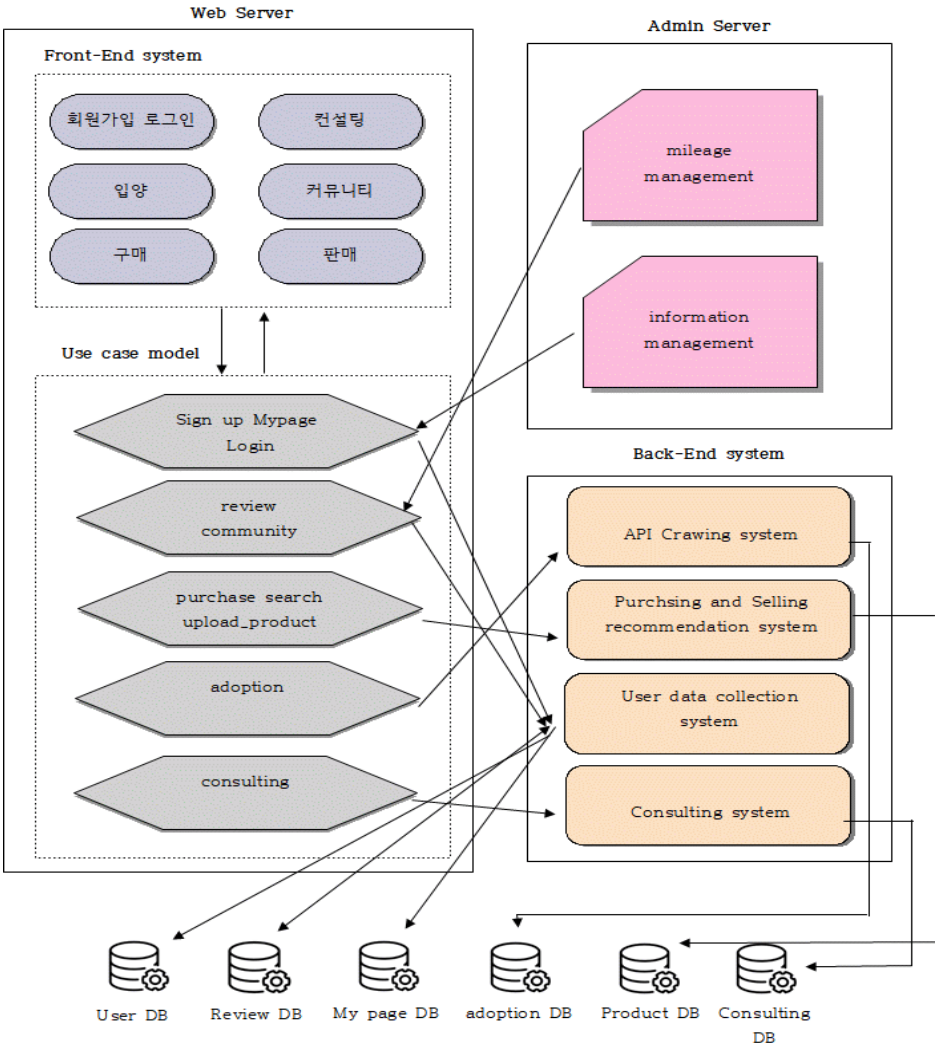

Diagram 1: Whole Map of Petkage Architecture

3.3 Package Diagram

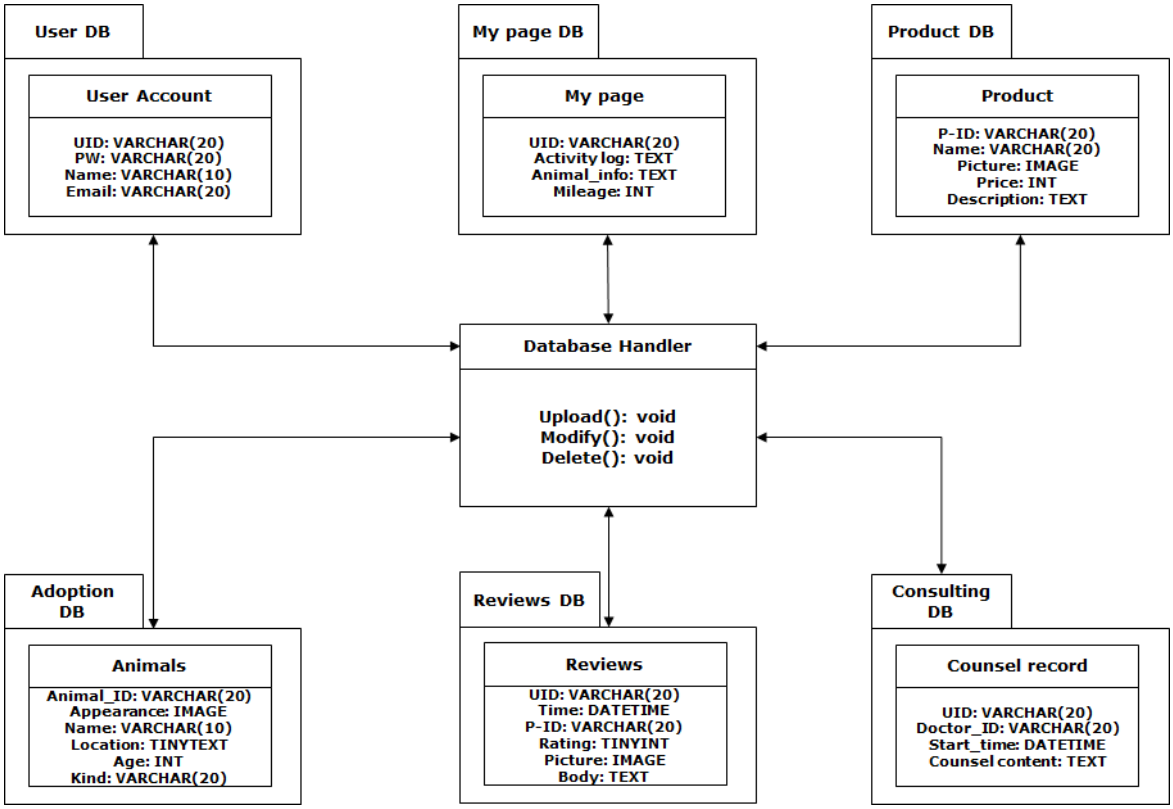

Diagram 2: Package Diagram for Petkage

3.4 Deployment Diagram

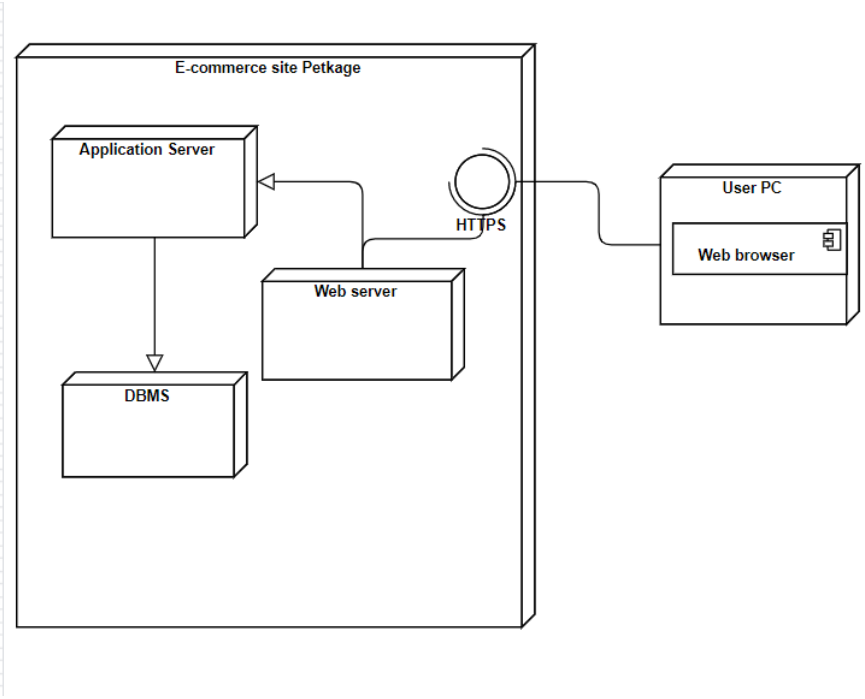

Diagram 3: Deployment Diagram for Petkage

이상으로 3개의 Diagram으로 각 시스템을 설명할 것이다. 각각의 시스템들은 하나의 큰 단원들로 구분해 세부적으로 분석할 것이며, 각각에 대해 핵심적인 기능들, 구성 요소 등에 대해 서술한다.

4. User Management System

4.1 Objective

사용자는 회원 가입 후 로그인을 한 후에 이 시스템을 활용할 수 있다. 이 시스템은 전체적으로 어떻게 User Management System이 다른 시스템과 상호작용하는지에 대해 서술한다. User Management 시스템은 유저의 개인 공간에 대한 데이터를 담고 있으며, 유저에 대한 정보와 활동로그, 개인 반려동물 정보와 마일리지, 아이디로 구성되어 있다.

4.2 Class Diagram

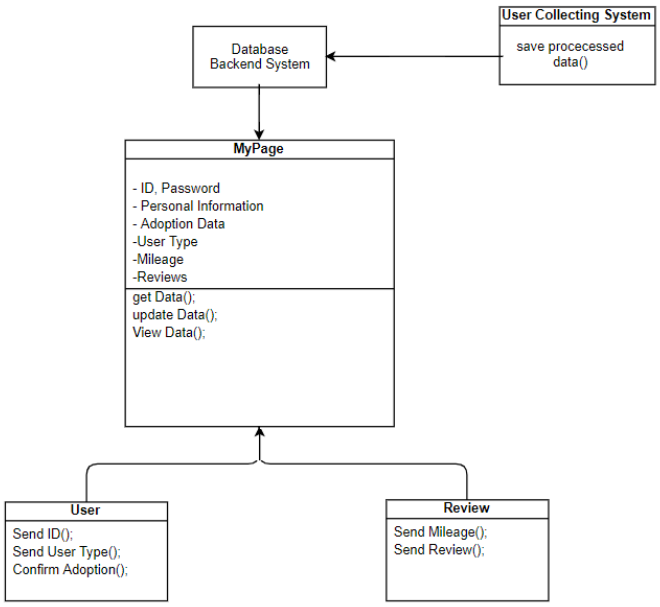

Diagram 4: Communication diagram for User Management System

A. My page

A.1. Attributes

ID: characteristic user identification(char)

Personal Info: which includes email, name, location…(char)

Password: characteristic user password (char)

Adoption Data: characteristic of adopted animal including Type (char)

User Type: Divided into three groups (Customer, Seller, PetDoctor) (char)

Mileage: point of Mileage which includes date and log (int)

Reviews: Check Past Review(char)

A.2. Operation

Get data: Get Data from DB

Update Data: Update Data from User

View Data: Show data to User

B. User

Send Id: Send ID to verify User

Send User type: Send User type to my page

Confirm Adoption: Confirms he adopted animal or not

C. Review

Send Mileage: Send Mileage to Mypage and save it

Send Review: Send Review data to My page

4.3 Sequence diagram

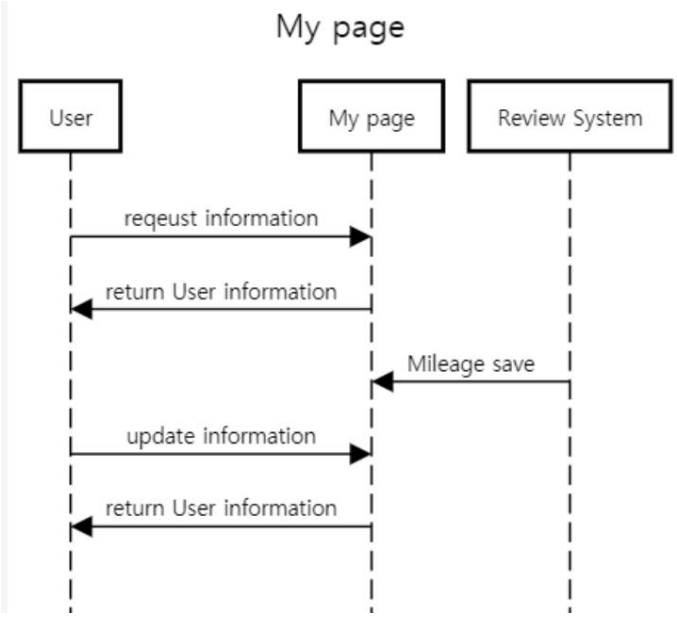

Diagram 5: Sequence diagram for User Management System

4.4 State Diagram

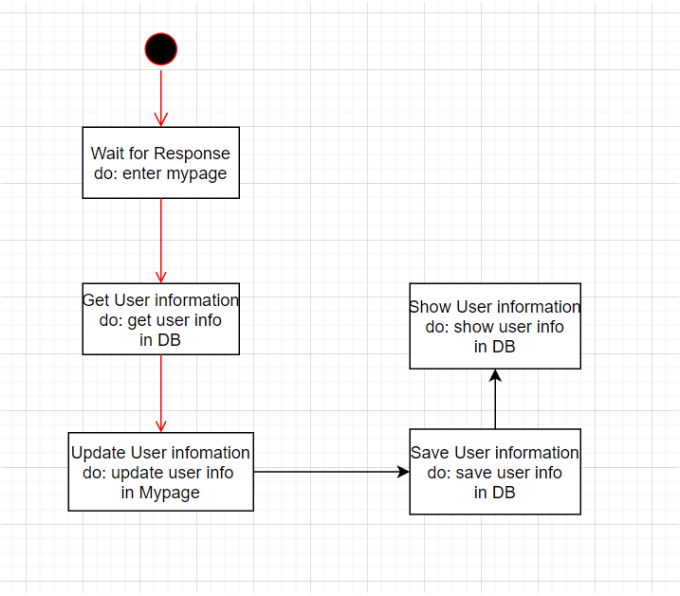

Diagram 6: State Diagram for User Management System

5. Adoption System

5.1 Objective

사용자는 회원 가입 이후 로그인을 한 후에 이 시스템을 사용할 수 있다. 이 부분은 입양 시스템 과정에서 데이터가 어떻게 만들어지고 어떤 방식으로 보여지는지 서술한다. Class Diagram, Sequence Diagram 그리고 State Diagram이 포함될 것이다. 이 세 부분이 Adoption System을 설명한다.

5.2 Communication Diagram

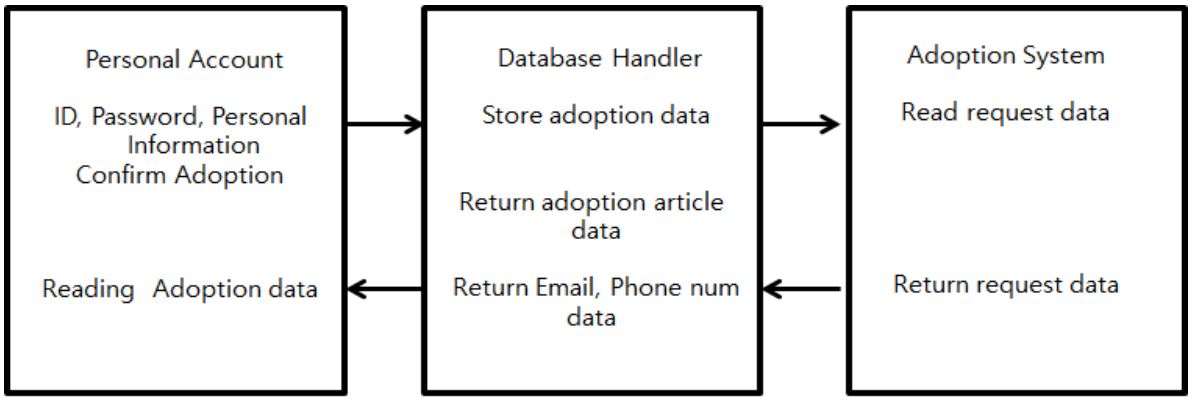

Diagram 7: Communication Diagram for Adoption System

A. Personal Account

ID: characteristic user identification(char)

Password: user password(char)

Operation

Reading: user read articles about adoption

Confirm: user confirms whether he adopts or not

B. Database Handler

Operation

Return data: Return data from database

Save(store) data: Storing data to database

C. Adoption System

Read data: Read data from Database

Return data: Return data from API system

5.3 Sequence Diagram

A. Read

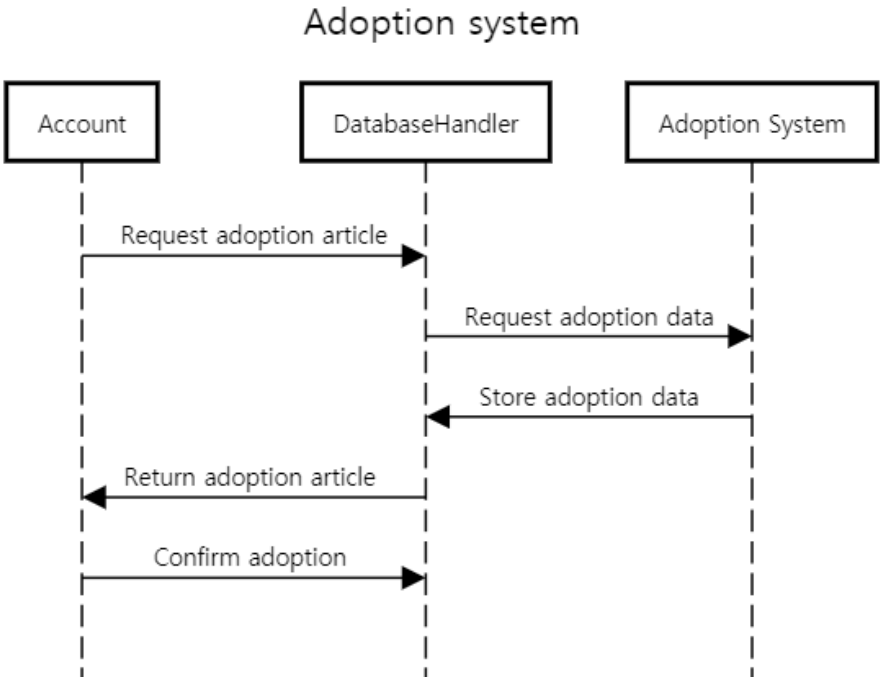

Diagram 8: Sequence Diagram for Adoption (Read)

B. Confirm

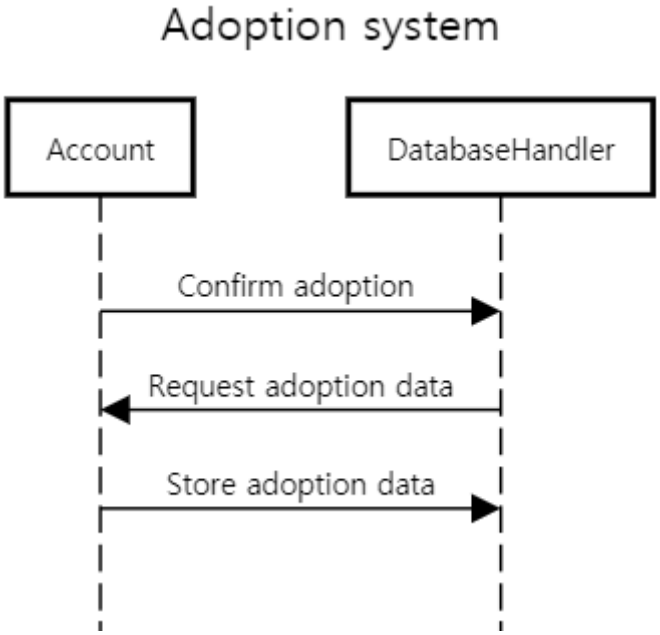

Diagram 9: Sequence Diagram for Adoption (Confirm)

C. Save (Store)

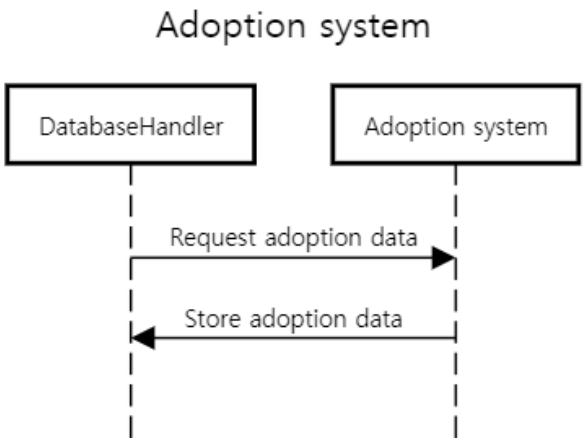

Diagram 10: Sequence Diagram for Adoption (Save)

5.4 State Diagram

A. Read

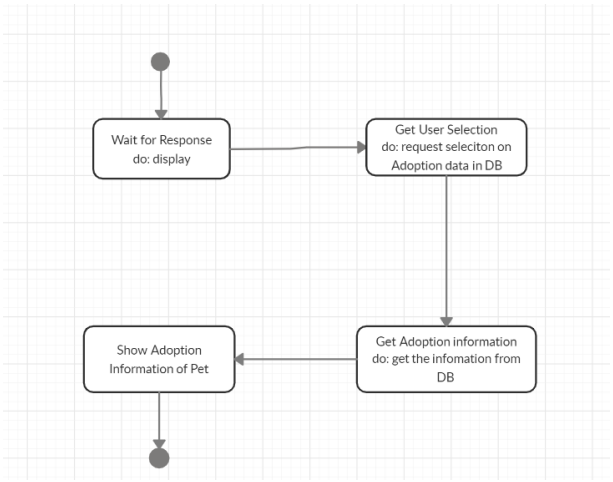

Diagram 11: State Diagram for Read in Adoption

B. Confirm

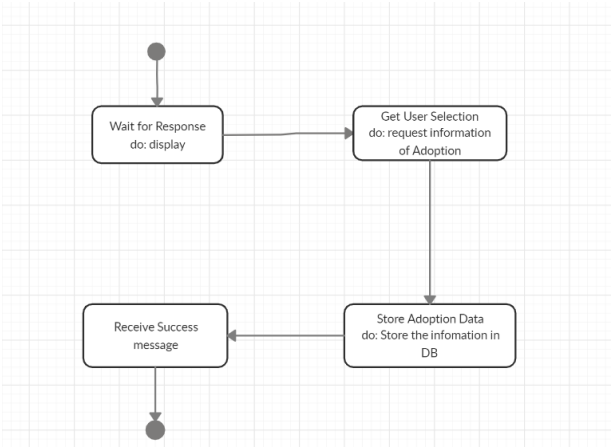

Diagram 12: State Diagram for Confirm in Adoption

6. User Data Collecting System

6.1 Objective

사용자가 Petkage를 이용하며 입력하는 다양한 정보들을 구분해서 저장하고 가공해두는 시스템이다. 구매 및 검색, 리뷰, 상담 게시글 등에서 온 데이터이며, 해당 데이터를 쉽게 사용 가능하도록 전처리(Preprocessing) 해 전체 시스템에서 활용할 수 있는 표준화된 데이터로 변환한다. 저장된 데이터들은 Recommendation System에서의 구매 및 검색 데이터, Consulting System에서 사용하는 게시글 데이터 등에 활용되며 다양한 알고리즘 및 시스템에 활용된다.

6.2 Class Diagram

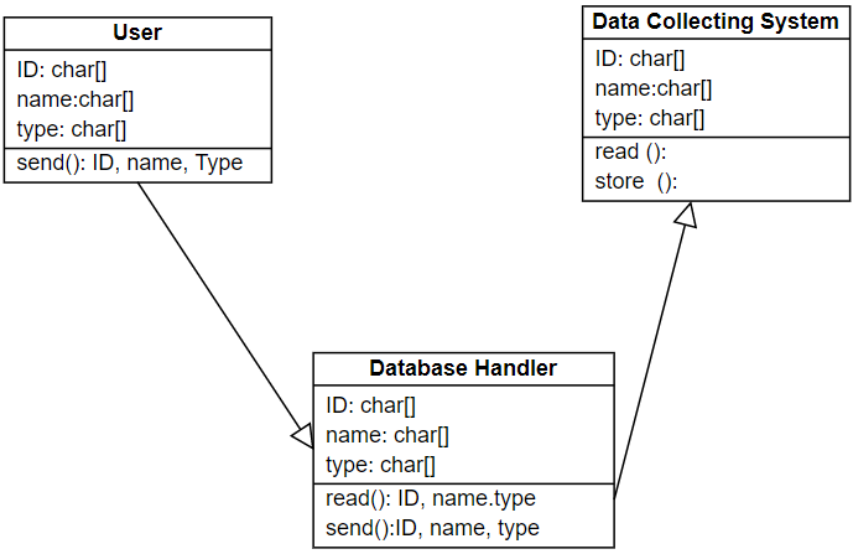

Diagram 13: Class Diagram for User Data Collecting System

A. User

ID: characteristic user identification(char)

Name: Name of Adopted Pet(char)

Type: Type of adopted pet(char)

Send (): send data through database Handler

B. Database Handler

Read (): read data and verify whether data is true or not

Send (); send data to user collecting system

C. Data Collecting System

Read (): System gets data

Store (): System save information in DB

6.3 Sequence Diagram

A. Adoption System

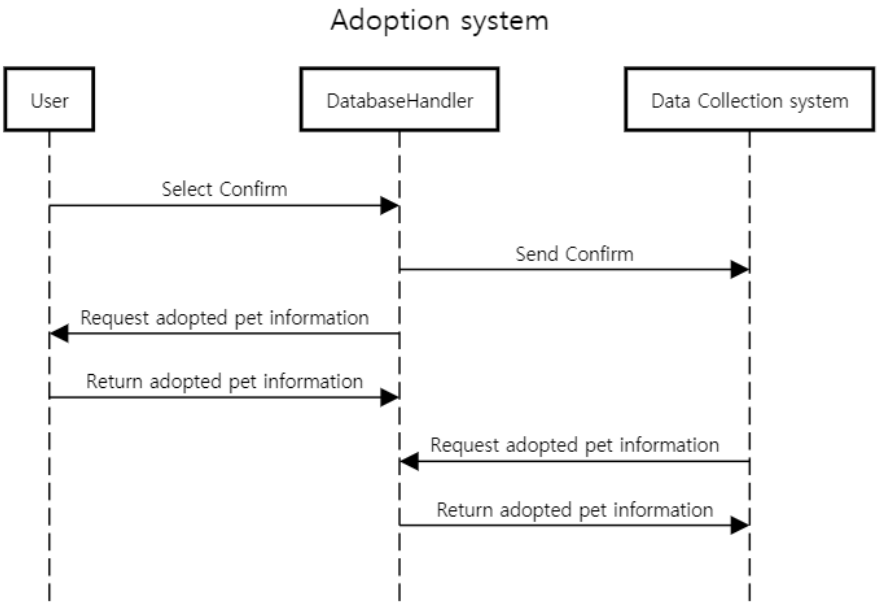

Diagram 14: Sequence Diagram for User Data Collecting System (Adoption)

B. Recommendation System

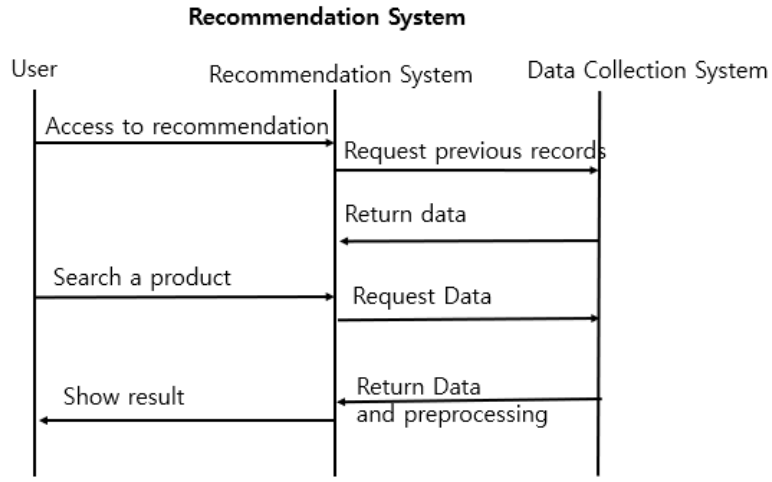

Diagram 15: Sequence Diagram for User Data Collecting System (Recommendation)

6.4 State Diagram

A. Adoption System

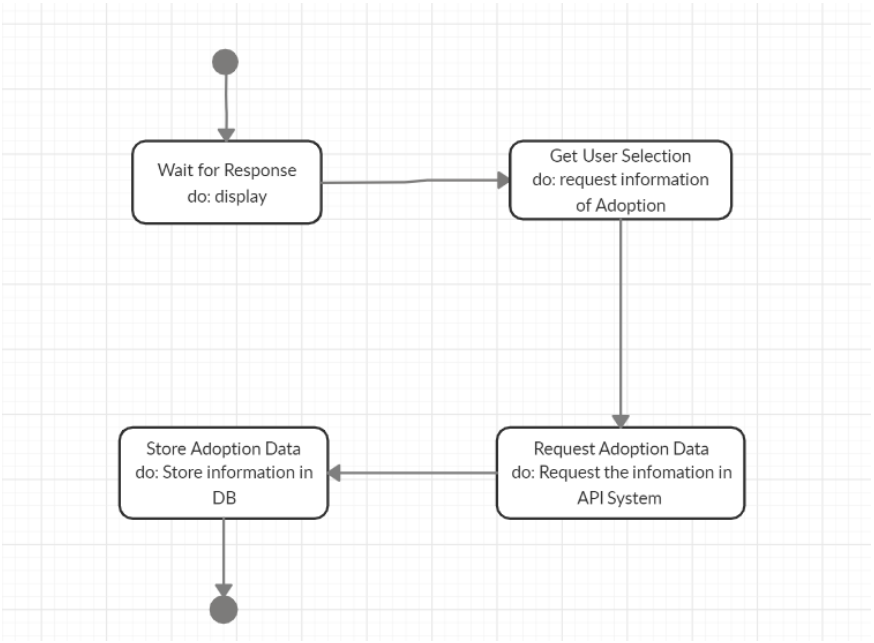

Diagram 16: State Diagram for User Data Collecting System (Adoption)

B. Recommendation System

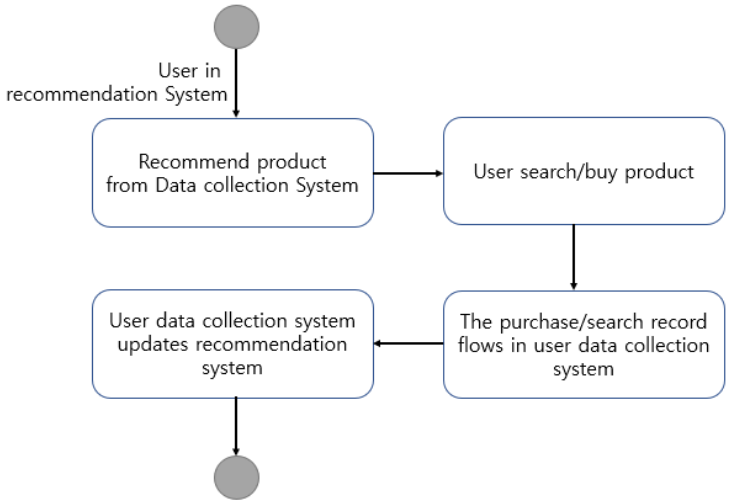

Diagram 17: State Diagram for User Data Collecting System (Recommendation)

7. Search System

7.1 Objective

이 시스템은 유저가 Petkage에 접속 한 후, 상품을 구매하거나 유기동물을 분양하기 위해서 사용자들이 검색하는 상품 혹은 동물을 찾아주는 Search 기능으로, 반려용품, 분양, 반려상담 어디서든 활용될 수 있으며 사용자가 원하는 정보를 빠르게 찾고 싶거나 불필요한 정보를 제외하고 필요한 정보만 보고 싶을 때 사용한다. 사용자가 원하는 검색 키워드를 입력하고 해당 키워드에 해당하는 결과를 출력하는데, 이때 해당 결과가 여러 개일 경우 시간 순으로 정렬하여 출력한다. 분양의 경우, 가까운 지역구에 대한 정보가 먼저 출력되게 된다.

7.2 Search Class Diagram

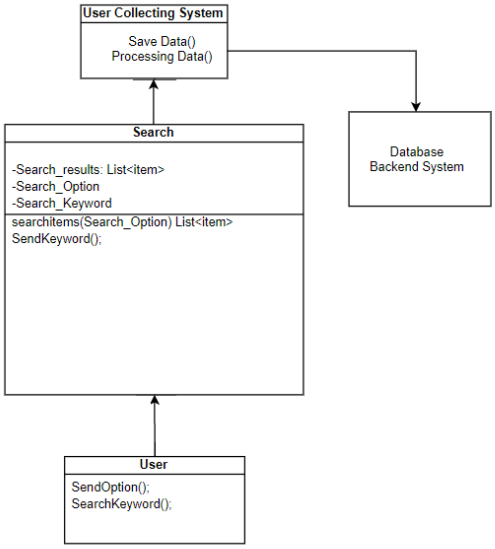

Diagram 18: Class Diagram for Search

A. Search

Serach_result: 검색 후 결과 목록 (list)

Search_option: 검색 옵션, 결과가 여러 개일 경우 시간순이나 공간순으로 정렬(int, char)

SearchKeyword: 유저가 검색한 키워드(char)

Searchitems(option): 해당 조건으로 상품을 검색한다

SendKeyword(): 사용자가 검색한 키워드들을 User collecting System으로 전달함

B. User

SendOption(): 사용자가 설정한 옵션을 Search System에 전달

SearchKeyword(): 사용자가 찾고 싶은 상품을 검색하여 Search System에 전달

C. User Collecting System

Processdata (): Search에서 전달받은 키워드를 배열의 형태로 정리

Savedata (): Search에서 전달받은 값들을 데이터베이스로 전달

7.3 Sequence Diagram

A. View

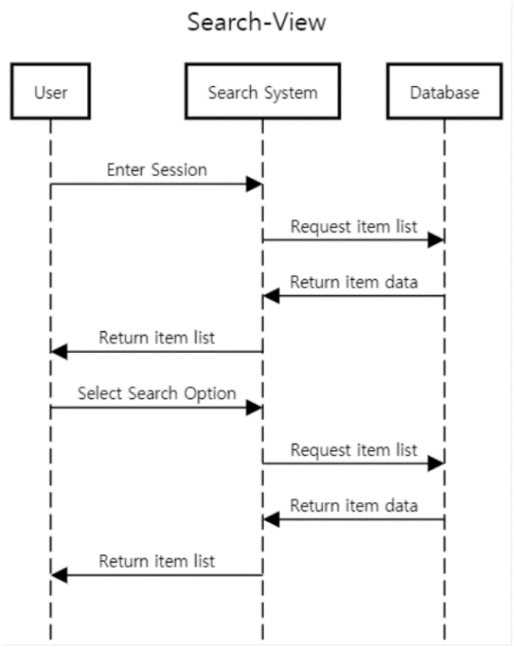

Diagram 19: Sequence Diagram for View

B. Search

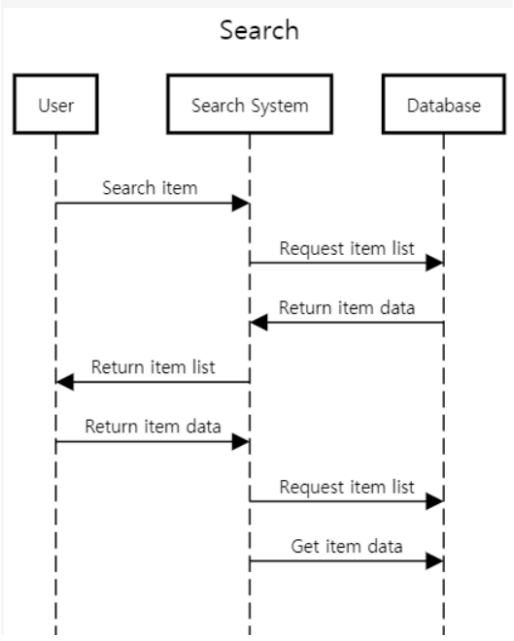

Diagram 20: Sequence Diagram for Search

7.4 State Diagram

A. view

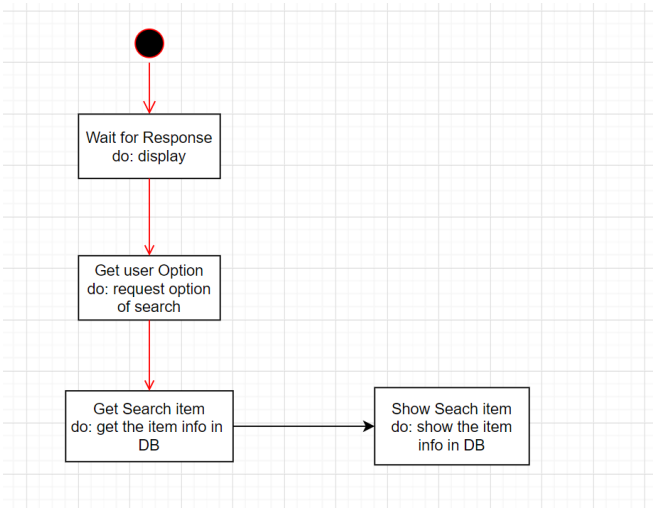

Diagram 21: State Diagram for View

B. Search

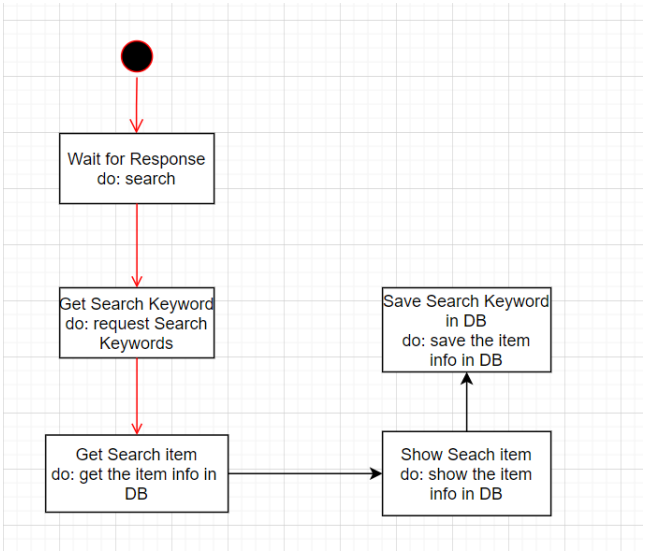

Diagram 22: State Diagram for Search

8. Recommendation System

8.1 Objective

일반사용자(구매자)에게 리뷰 평점이 높은 순, 판매량, 그리고 Petkage에서 제공하는 빅데이터를 활용한 방법 이렇게 세 가지로 제품을 추천하는 시스템이다. 사용자의 검색 기록, 구매 기록 및 리뷰한 상품 등의 전반적인 정보를 수집한 후, 개별 사용자에게 가장 구매할 법한 상품을 상단의 상품 구매 페이지에 띄어 준다.

8.2 Communication Diagram

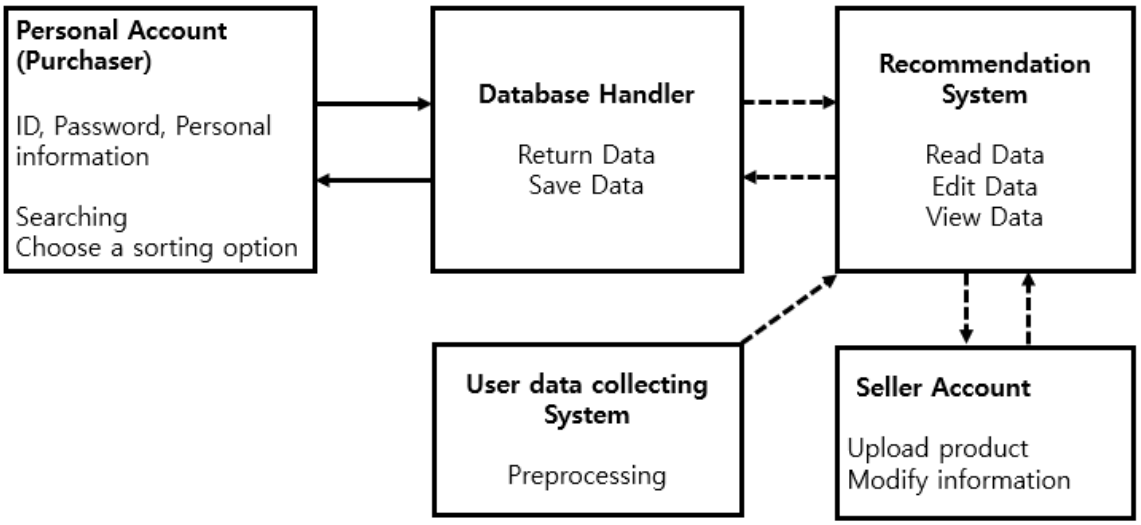

Diagram 23: Communication Diagram for Recommendation System

A. Personal/Seller Account

ID: characteristic user identification(char)

Password: user password(char)

-Searching: user search an item with specific keywords

-Choose a sorting option: user choose one of three sorting options; Rating, Purchased amount, Petkage Bigdata sorting

-Upload product/Modify information: seller upload its items which includes modifiable contents like pictures, texts, videos about them

B. Database Handler

Return Data: Reading data from database

Save Date: Saving data to database

C. Recommendation System

Read Data: System can read data

Edit Data: System can edit data

View Data: System can edit data

D. User data collecting System

Preprocessing: Providing Recommendation system with its preprocessed data

8.3 Sequence Diagram

A. User gets recommendation

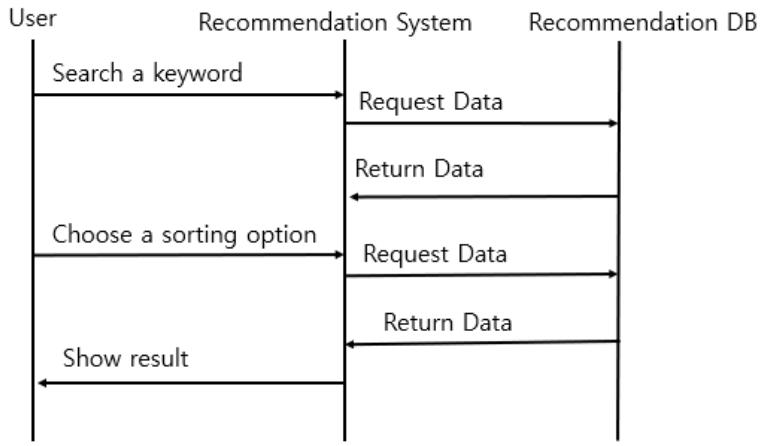

Diagram 24: Sequence Diagram for Recommendation

B. When user buy a product

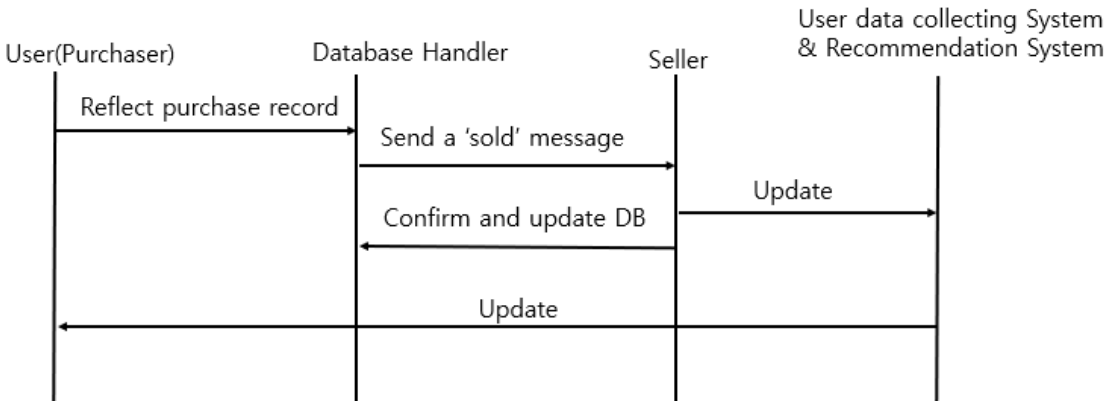

Diagram 25: Sequence Diagram for Buying

8.4 State Diagram

A. Recommendation system works

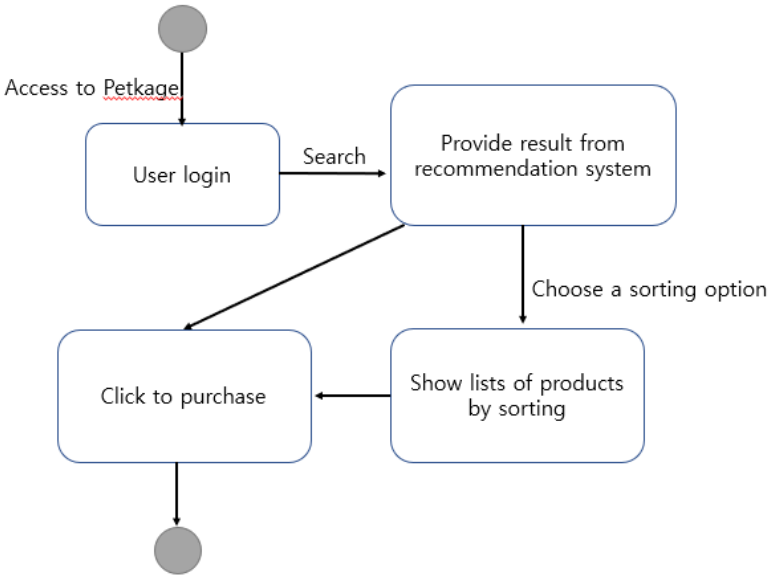

Diagram 26: State Diagram for Recommendation

B. Change in recommendations after purchase

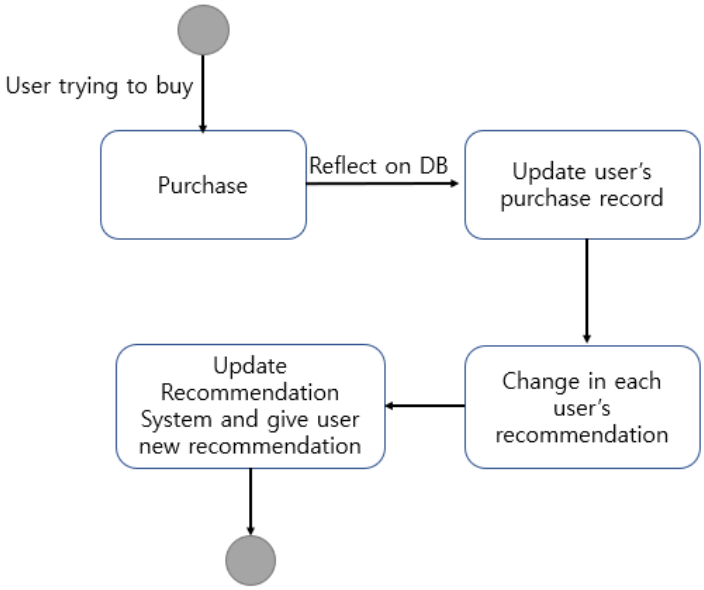

Diagram 27: State Diagram for Changing Recommendation

9. Review & Rating System

9.1 Objective

목표: 구매가 완료된 상품들을 바탕으로 상품에 대한 후기와 사진, 그리고 판매자에 대한 평가를 할 수 있다. DB에 쌓인 후기는 이후 사용자의 추천시스템에 적용되는 것뿐만 아니라 타 사용자의 상품 추천과 구매에 이용되기도 한다.

9.2 Communication Diagram

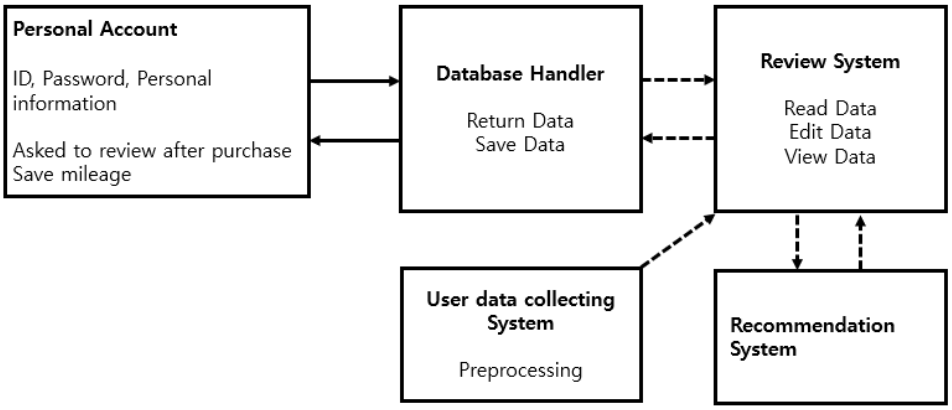

Diagram 28: Communication Diagram for Review &Rating

A. Personal/Seller Account

ID: characteristic user identification(char)
Password: user password(char)
-Asked to review after purchase: user is asked to leave comments on products
-Save mileage: Review & rating provides user with certain mileage

B. Database Handler

Return Data: Reading data from database
Save Date: Saving data to database

C. Review System

Read Data: System can read data
Edit Data: System can edit data
View Data: System can edit data

9.3 Sequence Diagram

A. Writing Review

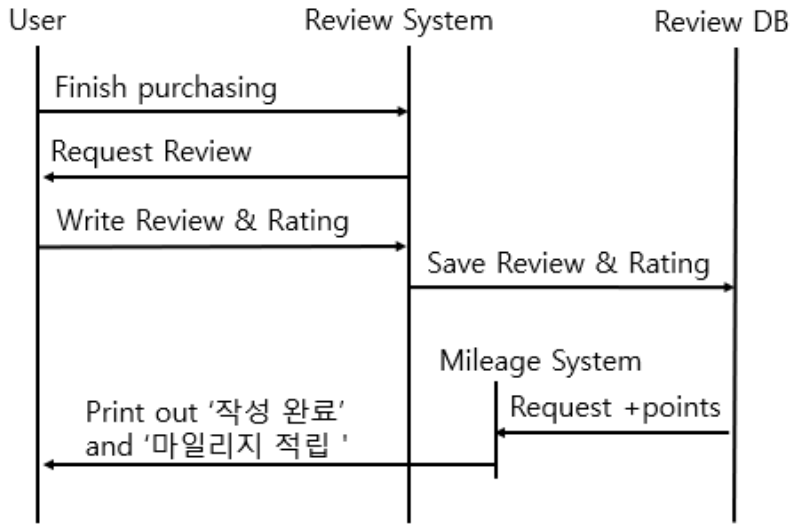

Diagram 29: Sequence Diagram for Writing Review

B. Reflecting reviews on Recommendation System

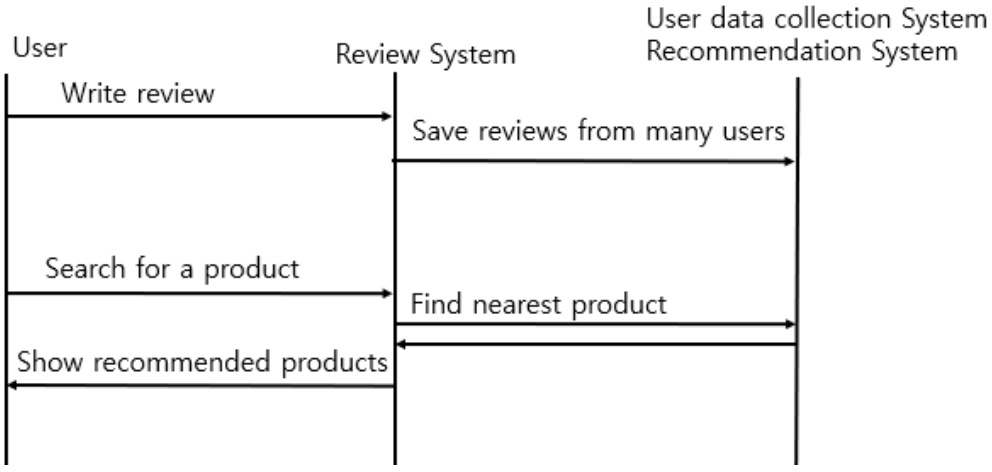

Diagram 30: Sequence Diagram for Reflecting Review

9.4. State Diagram

A. Review& Rating

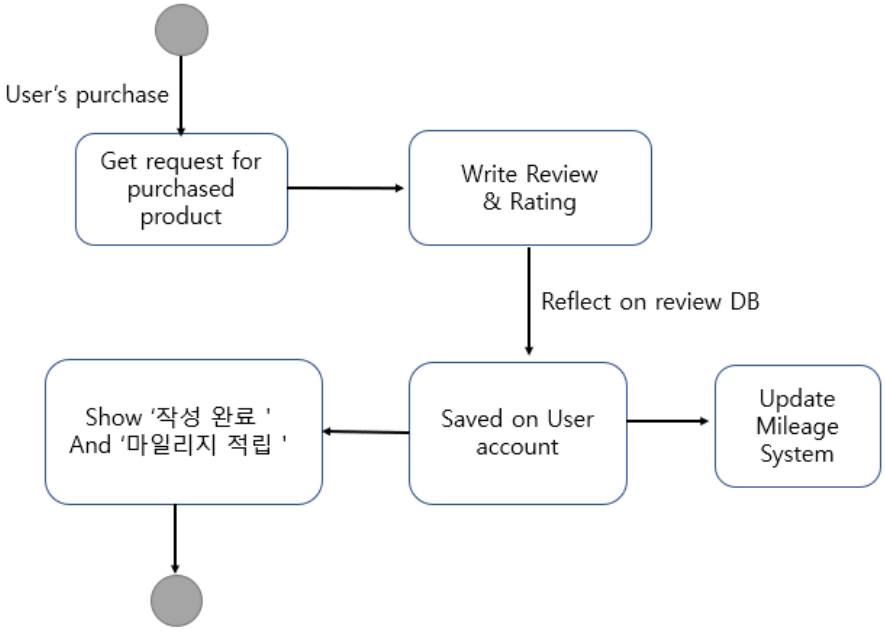

Diagram 31: State Diagram for Review &Rating

10. Community System

10. 1 Objective

community system은 Petkage 이용자들이 게시글을 올릴 수 있는 게시판이며 이용자들은 이곳에서 반려동물에 대한 지식(건강, 식사, 놀이) 등 다양한 분야의 내용을 글을 쓰며 공유 할 수 있는 공간을 만드는 것을 목표로 한다.

10.2 Communication Diagrams

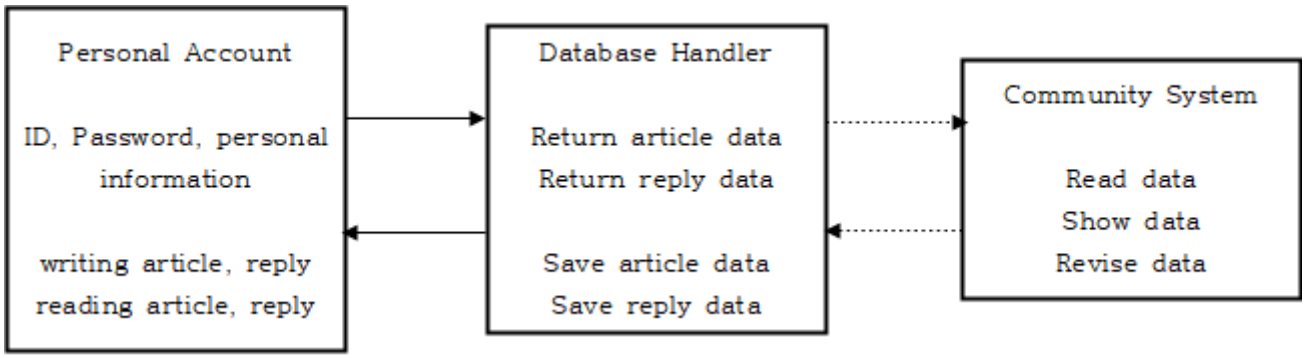

Diagram 32: Communication Diagram for Community System

A. Personal Account

ID: characteristic user identification

Password: user password

Writing: user make articles and reply in board

Reading: user read articles and reply from board

B. Database Handler

Return data: Reading data from database

Save Data: Saving data to database

C. Community System

Read Data: System read data from database

Revise Data: System revise data from user input

Show Data: System show saved data to user

10.3 Sequence Diagram

A. Read system

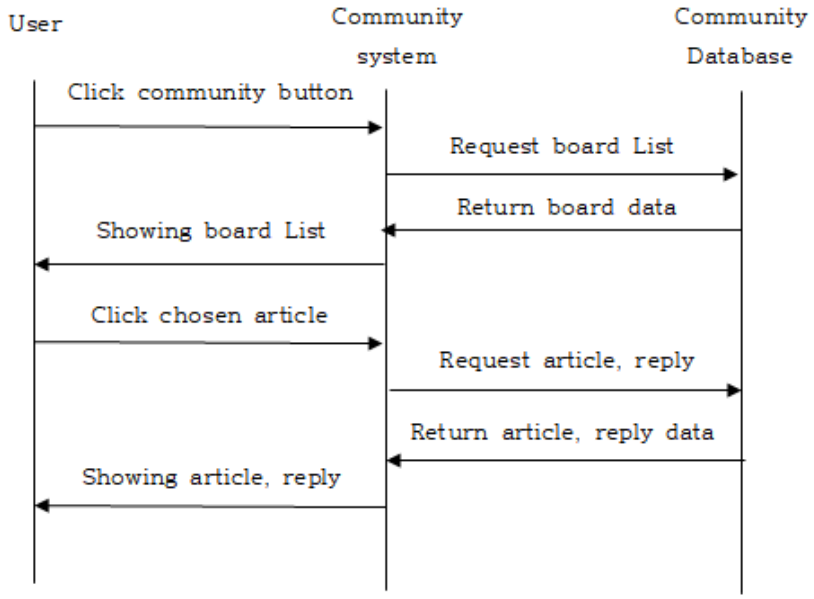

Diagram 33: Sequence Diagram for Read (Community)

B. Revise System

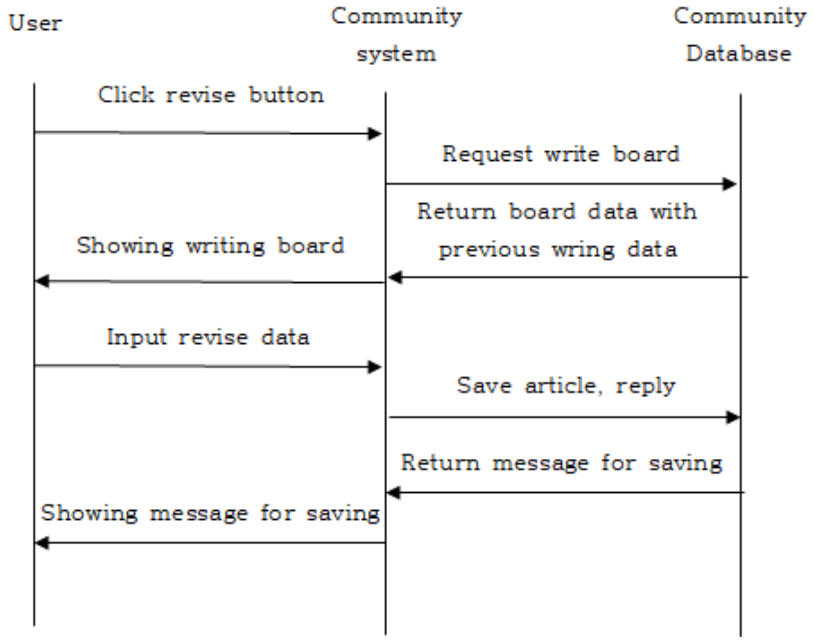

Diagram 34: Sequence Diagram for Read (Revise)

C. Writing system

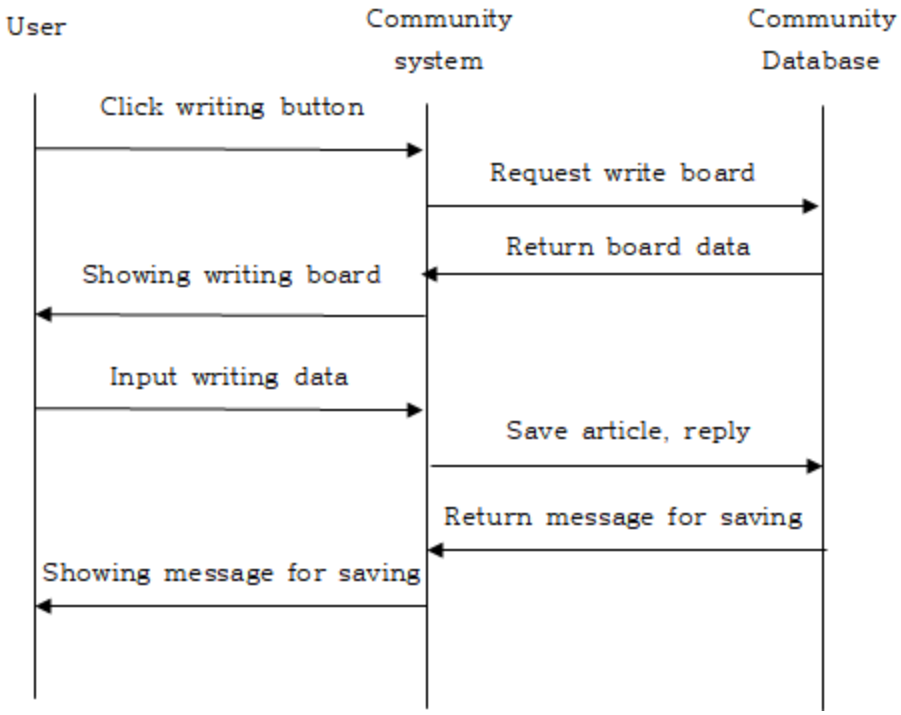

Diagram 35: Sequence Diagram for Read (Writing)

10.4 State Diagrams

A. Read

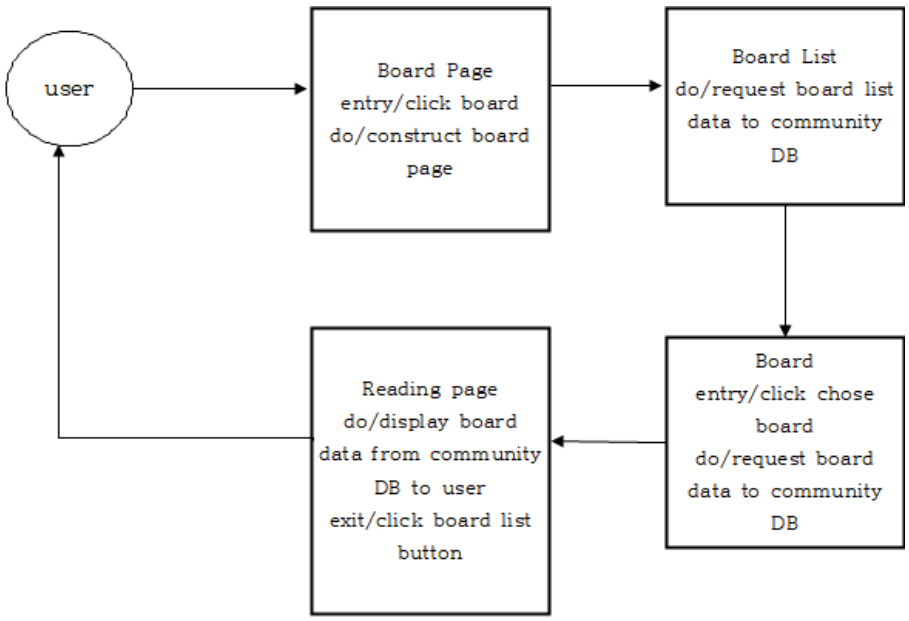

Diagram 36: State Diagram for Read (Community)

B. Write

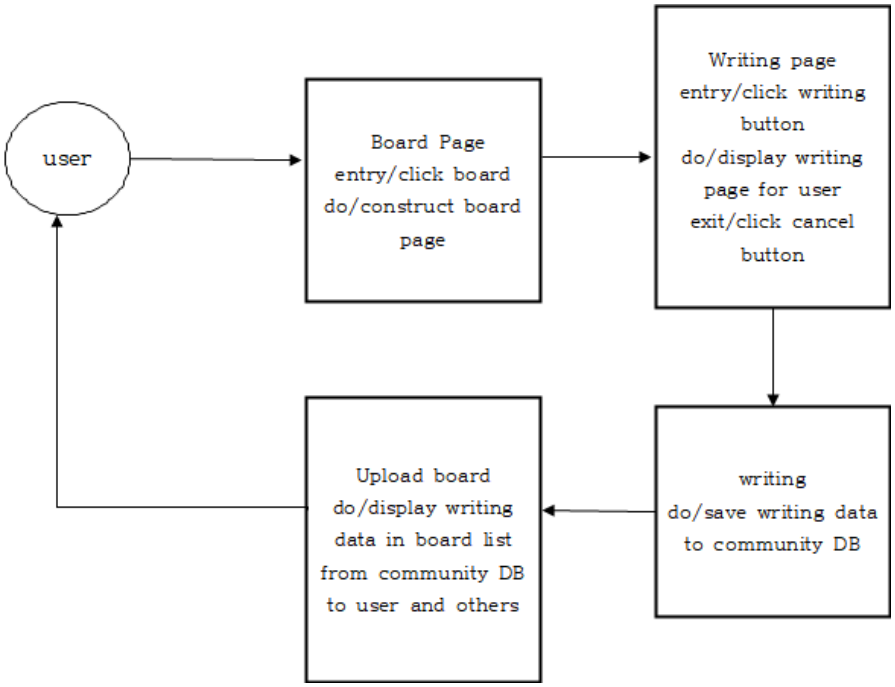

Diagram 37: State Diagram for Write (Community)

C. Revise

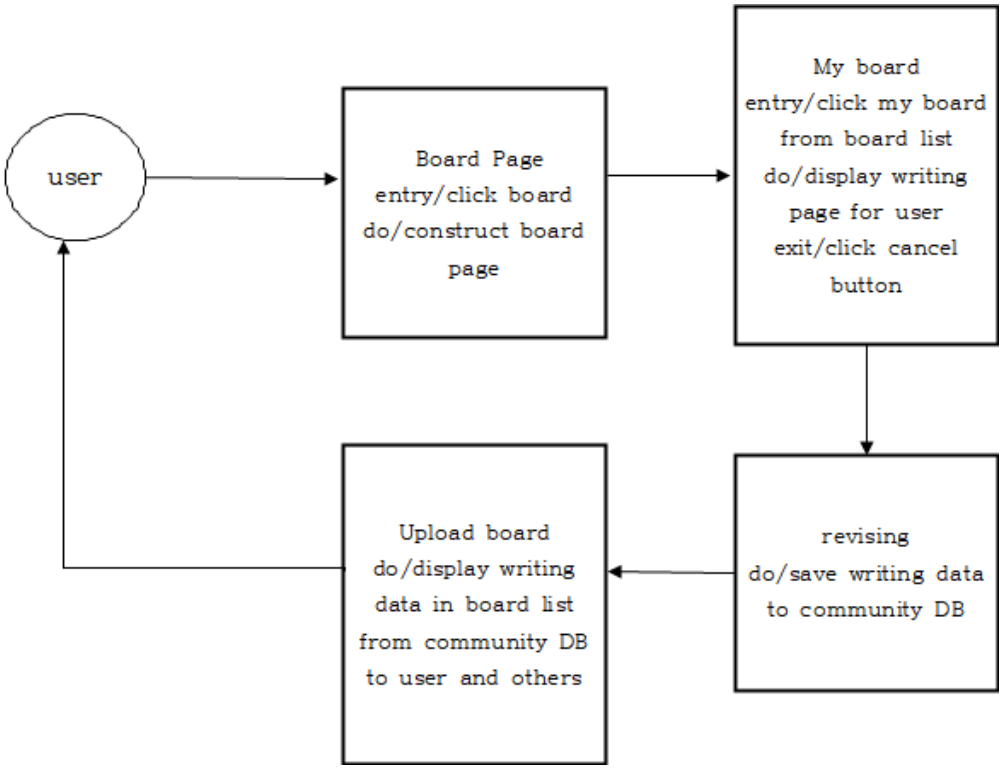

Diagram 38: State Diagram for Revise (Community)

11. Consulting System

11.1 Objective

Consulting system은 Petkage 이용자들이 반려동물 전문의에게 1:1 채팅 서버를 이용하여 반려동물의 건강 상태를 확인, 질문할 수 있게 만들었으며 이용자들이 전문의에게 보내는 텍스트, 사진, 동영상을 통해 내원 여부를 결정할 수 있게 해준다.

11.2 Communication Diagram

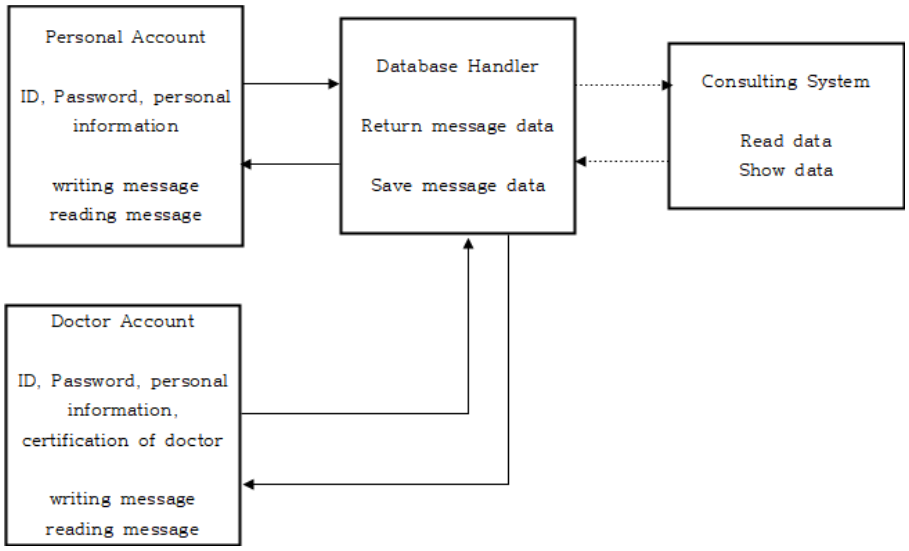

Diagram 39: Communication Diagram for Consulting System

A. Personal Account

ID: characteristic user identification

Password: user password

messaging: user make message in chatting system

Reading: user read message in chatting system from doctor

B. Doctor Account

ID: characteristic doctor identification

Password: doctor password

Certification of doctor: doctor must be certificated themselves by license etc.

messaging: doctor make message in chatting system

Reading: doctor read message in chatting system from user

C. Database Handler

Return Data: Reading data from database

Save Data: Saving data to database

D. Consulting System

Read Data: System read data from database

Show Data: System show saved data to user, doctor

11.3 Sequence Diagram

A. Read System: User

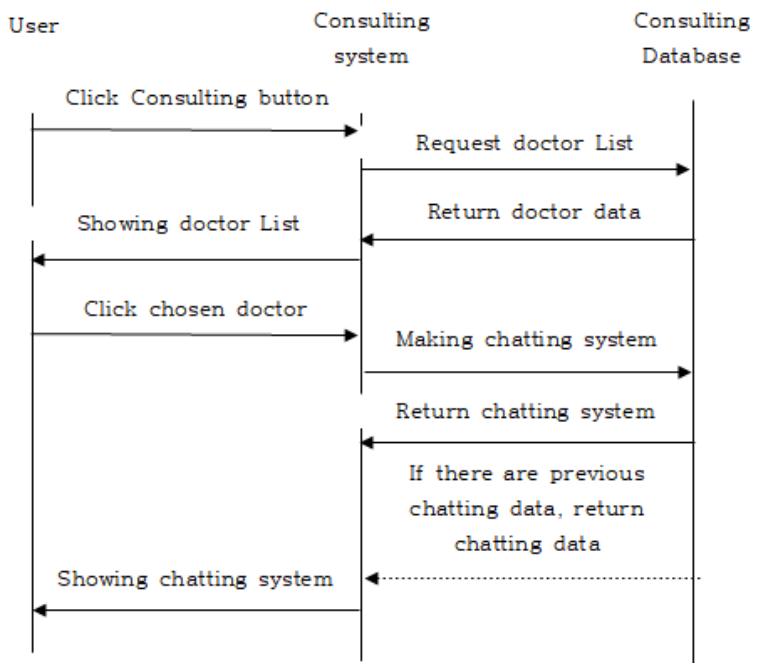

Diagram 40: Sequence Diagram for Read (Consulting)

B. Read system: Doctor

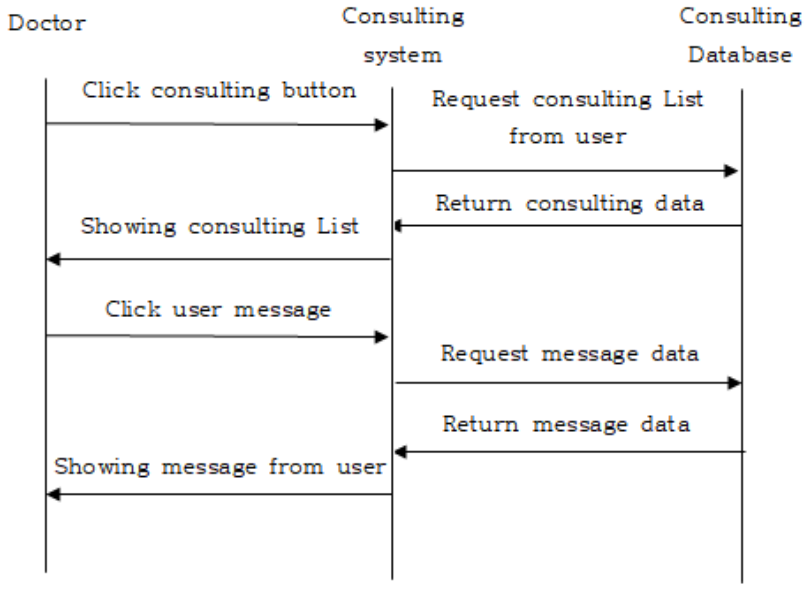

Diagram 41: Sequence Diagram for Read Doctor (Consulting)

C. Write

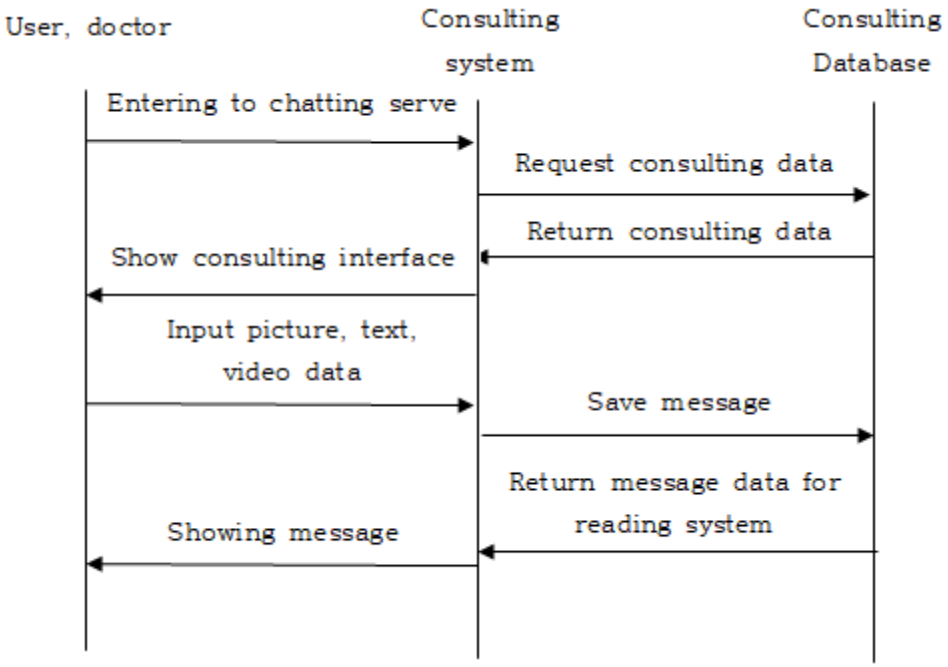

Diagram 42: Sequence Diagram for Write (Consulting)

11.4 State Diagram

A. Write: User

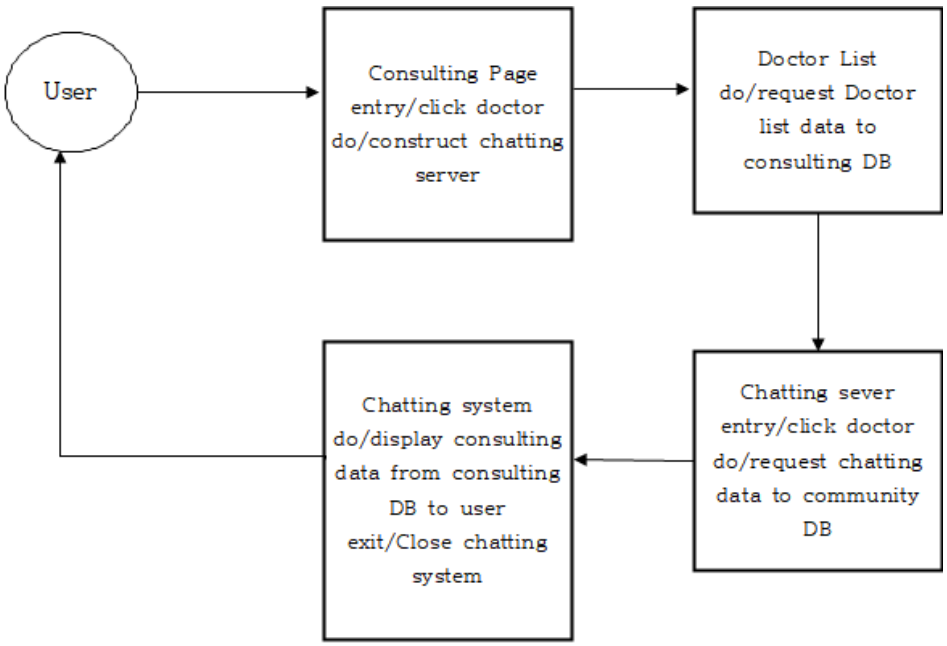

Diagram 43: State Diagram for Writing User (Community)

B. Read: Doctor

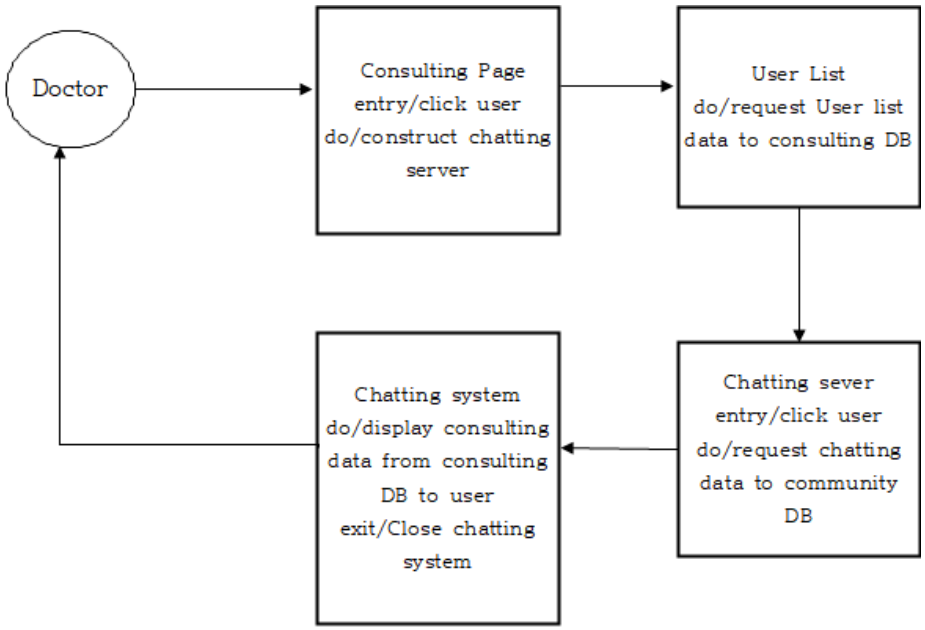

Diagram 44: State Diagram for Read Doctor (Community)

C. Write

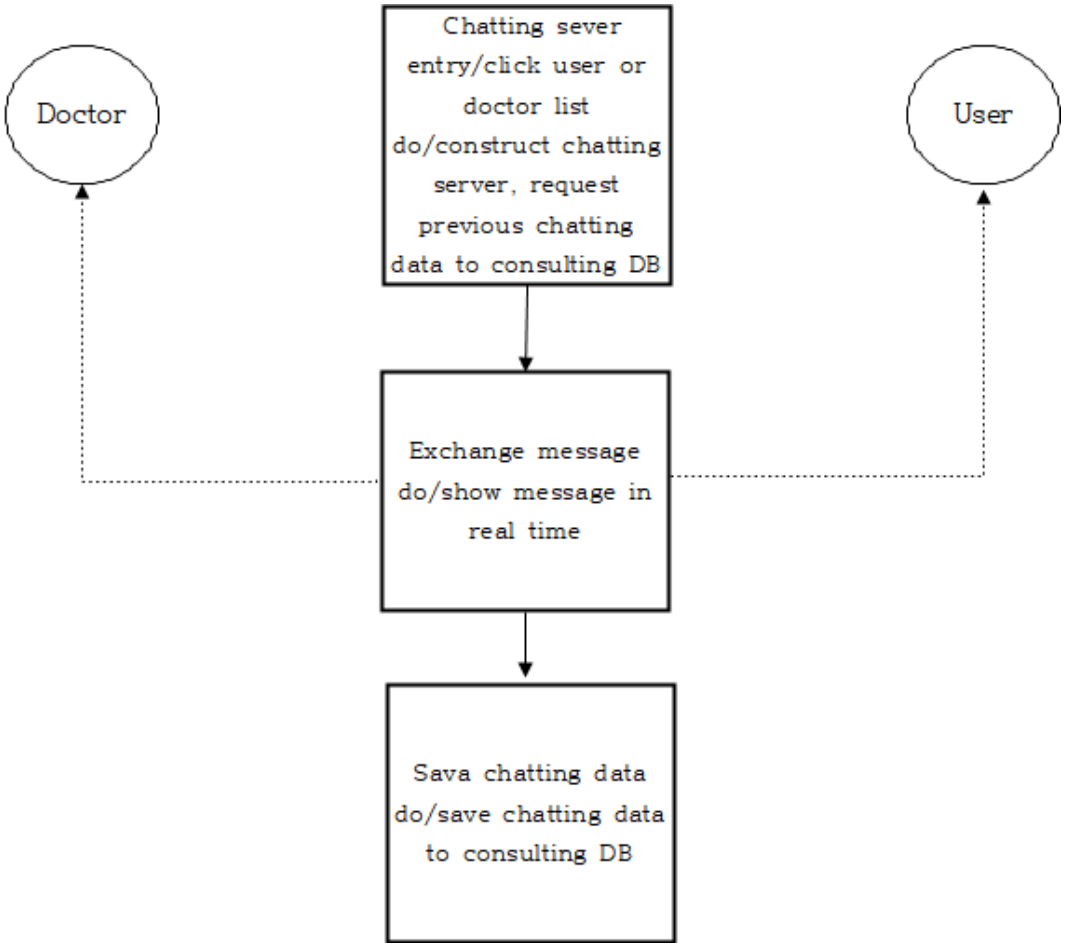

Diagram 45: State Diagram for Write (Community)

12. Protocol Design

12.1. Objectives

Communication Protocol은 시스템 내부의 두 개 이상의 개체 간에 정보를 전달하기 위해 소통하는 시스템 상의 규칙을 의미한다. 이번 단원에서는 Petkage 내의 다양한 개체 간에 어떠한 프로토콜 구조가 사용되는지, 그리고 어떻게 정의되어 있는지 기술한다.

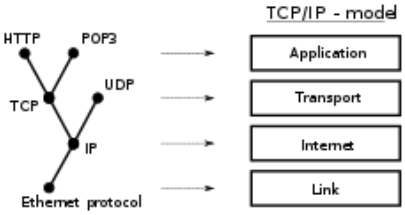

Figure 10: TCP/IP Model for Web Protocol

12.2. Protocol Overview

Petkage는 웹 어플리케이션이기 때문에, 웹 상에서 표준 통신 프로토콜인 TCP/IP 모델을 사용한다. 해당 모델에는 다양한 프로토콜이 존재하며, Petkage는 이 중 대표적으로 HTTP를 사용한다. 해당 프로토콜은 GET/HEAD/POST/PUT의 4가지 형식으로 구성되어 있는데, 로그인처럼 사용자의 정보를 서버에 넘기는 행위는 POST를 통해 서버에 전달되며, Recommendation System 처럼 내부적인 알고리즘을 통해 출력된 결과를 사용자에 넘기는 행위는 GET을 통해 전달된다.

Petkage 내부에서 전달되는 데이터는 숫자나 문자 같은 단일 형식이 아닌, 다양한 형식(ex. 아이디, 비밀번호, 리뷰 문서 등)으로 구성되어 있기 때문에, 다양한 형식을 효율적으로 HTTP 프로토콜로 전달하기 위한 통일된 형식이 필요하다. 이를 위해 XML 형식을 차용한다.

Figure 11: XML and its example

XML은 Extensible Markup Language은 W3C에서 개발된 다목적 마크업 언어로, 많은 종류의 데이터를 기술하는 데에 사용해 인터넷에 연결된 시스템끼리 쉽게 데이터를 전달할 수 있도록 도와준다. 이러한 특징은 우리 시스템의 다양한 데이터 형식을 간소화하고 일괄적으로 전달할 수 있다는 장점이 있다. 또한 구성 자체가 HTML과 유사하기에 개발에도 유용할 것이라 예상한다.

12.3. Protocol Details

시스템 간의 HTTP 통신은 크게 Request와 Response로 이루어지며, 해당 구조체들은 명령 라인(GET/POST 등), 헤더, 바디, 상태 코드, 오류 시 내용 등으로 구성된다. 모든 기능은 서버에 정보를 보내는 Request, 서버에서 처리를 한 후 정보를 받아오는 Response로 구성되기 때문에 해당 순서로 표를 작성했다. 다만 서술할 때 게시글 등록이나 리뷰 등록 등에 따라올 수 있는 글 수정/삭제 등의 부가적인 기능들은 자세히 서술할 필요가 없기 때문에 4.3.2의 Review Modify /Delete 부분에서만 자세히 다루도록 한다.

12.3.1. Authentication

로그인을 할 때 주의할 점은 단순히 일반 사용자 뿐 아니라 인증된 사용자 및 판매자도 존재한다는 점이다. 이를 구분해서 로그인을 할 때 서버에서 처리를 할 수 있는 구문이 필요하다.

*Login

Method	POST
URL	/auth/login/ (normal, vet, seller) ->일반 사용자(normal), 인증된 사용자(주로 동물병원 근무자기 때문에 vet), 판매자(seller)
Request Body	<user info> <id>: 사용자 아이디 <pw>: 사용자 패스워드 <specification> : 사용자 상세(normal/vet/seller)

If Success	200 OK
If Fail	400 Bad Request: 서버 내의 정보와 일치하지 않음
Success Body	<success>: 성공했다는 표시
Fail Body	<failed>: 실패했다는 표시 <log> <fail reason>: 인증 실패 사유

Table 1: Authentication

*Signup

Method	POST
URL	/auth/signup/ (normal, vet, seller) ->일반 사용자(normal), 인증된 사용자(주로 동물병원 근무자기 때문에 vet), 판매자(seller)
Request Body	<user info> <id>: 사용자 아이디 <pw>: 사용자 패스워드 <specification> : 사용자 상세(normal/vet/seller)

If Success	200 OK
If Fail	400 Bad Request: 서버 내의 정보와 중복됨
Success Body	<success>: 성공했다는 표시
Fail Body	<failed>: 실패했다는 표시 <log> <fail reason>: 등록 실패 사유

Table 2: Sign Up

12.3.2. Purchase Page & Recommendation System

애완용품 판매 페이지는 다양한 상품들이 존재하고, Recommendation System이 혼합되어 다양한 추천 상품들을 사용자에게 표시한다. 이에 따라 두 개의 서브 시스템을 연계해 다양한 프로토콜에 대해 서술할 것이다.

*Item Search

Method	POST
URL	/purchase/search: keyword
Request Body	<keyword>: 상품 검색에 사용한 검색어

If Success	200 OK
If Fail	404 Not Found: 해당 정보가 없음
Success Body	<search result>: 다수의 결과이므로 여러 개로 표시 <product id> <product name>
Fail Body	<failed>: 실패했다는 표시

Table 3: Purchase Page & Recommendation System

*Item View

Method	GET
URL	/purchase/product: id
Request Body	<product> <id>: 상품마다 주어지는 고유 id <name>

If Success	200 OK
If Fail	404 Not Found: 해당 정보가 없음
Success Body	<item info> <name> <price> <rating> <reviews>
Fail Body	<failed>: 실패했다는 표시

Table 4: Item View

*Item Purchase

Method	GET
URL	/purchase/user: id/product: id
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <product> <id>: 해당 상품의 id <price> <purchasing log>: 결제를 위한 정보들(금융거래법에 준함)

If Success	200 OK
If Fail	400 Bad Request: 결제 실패
Success Body	<purchase info> <user id> <item id> <purchasing log>: 결제 정보들(금융거래법에 준함)
Fail Body	<failed>: 실패했다는 표시 <fail log>: 실패 사유(결제 취소 001, 오류 002 등)

Table 5: Item Purchase

*Item Reviewing

Method	POST
URL	/purchase/product: id/review
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <review> <id>: 해당 리뷰에 주어지는 id <rating> <review>: 리뷰 내용

If Success	200 OK
If Fail	400 Bad Request: 리뷰 등록 실패
Success Body	<purchase info> <user id>
Fail Body	<failed>: 실패했다는 표시

Table 6: Item Reviewing

*Item Reviewing: Modification

Method	GET
URL	/purchase/product: id/review: id
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <review> <id> <modification>: 수정하는 평점, 내용 등

If Success	200 OK
If Fail	404 Not Found: 해당 리뷰가 없음
Success Body	<review> <id> <content>
Fail Body	<failed>: 실패했다는 표시

Table 7: Item Reviewing (Modification)

*Item Reviewing: Delete

Method	GET
URL	/purchase/product: id/review: id/delete
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <review> <id>

If Success	200 OK
If Fail	404 Not Found: 해당 리뷰가 없음
Success Body	<success>: 삭제 성공
Fail Body	<failed>: 실패했다는 표시

Table 8: Item Reviewing (Delete)

*Item Recommendation

Method	GET
URL	/purchase/search/recommendation
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id

If Success	200 OK
If Fail	400 Bad Request: 추천 상품 찾기 실패
Success Body	<search result>: 다수의 결과이므로 여러 개로 표시, Recommendation System의 결과임 <product id> <product name>
Fail Body	<failed>: 실패했다는 표시

Table 9: Item Recommendation

12.3.3. Adoption Page & API Crawling System

애완동물 분양 페이지는 API Crawling System을 통해 실시간으로 갱신되는 입양 보호 동물의 리스트를 사용자에게 보여주며, 애완동물 검색이나 입양 절차 등은 상품 구매 때와 비슷한 흐름으로 진행된다.

*Adoption Search

Method	POST
URL	/adoption/search: keyword
Request Body	<keyword>: 입양 검색에 사용한 검색어

If Success	200 OK
If Fail	404 Not Found: 해당 정보가 없음
Success Body	<search result>: 다수의 결과이므로 여러 개로 표시 <adoption id>: API로부터 추출한 해당 애완동물의 id <adoption name> <image src>
Fail Body	<failed>: 실패했다는 표시

Table 10: Adoption Search

*Adoption View

Method	GET
URL	/adoption/animal: id
Request Body	<adoption> <adoption id>

If Success	200 OK
If Fail	404 Not Found: 해당 정보가 없음
Success Body	<animal info> <name> <species>
Fail Body	<failed>: 실패했다는 표시

Table 11: Adoption View

*Adoption

Method	GET
URL	/adoption/user: id/animal: id
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <animal> <id>: API에서 추출한 해당 동물의 id <species> <adoption log>: 입양을 위한 정보들(동물보호법에 준함)

If Success	200 OK
If Fail	400 Bad Request: 입양 실패
Success Body	<adoption info> <user id> <animal id> <adoption log>: 입양을 위한 정보들(동물보호법에 준함)
Fail Body	<failed>: 실패했다는 표시 <fail log>: 실패 사유(입양 취소 001, 오류 002 등)

Table 12: Adoption

12.3.4. Consulting, Community & Consulting System

커뮤니티 및 애완동물 상담 페이지에서는 Consulting System이 곁들여져 다양한 게시글과 애완동물 상담 내용이 올라온다. 각 글들은 수정/삭제가 가능하다.

*Post Article

Method	POST
URL	/community/post: id
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <article> <id>: 해당 글에 주어지는 id <content>

If Success	200 OK
If Fail	400 Bad Request: 게시 실패
Success Body	<success>: 성공했다는 표시
Fail Body	<failed>: 실패했다는 표시

Table 13: Post Article

*Post Comment

Method	POST
URL	/community/post: id/comment: id
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <article> <id>: 해당 글에 주어지는 id <comment> <id>: 댓글에 주어지는 id <content>: 댓글 내용

If Success	200 OK
If Fail	400 Bad Request: 게시 실패
Success Body	<success>: 성공했다는 표시
Fail Body	<failed>: 실패했다는 표시

Table 14: Post Comment

*Post Consulting

Method	POST
URL	/consulting/post: id
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <consulting> <id>: 해당 글에 주어지는 id <content>: 게시글의 내용

If Success	200 OK
If Fail	400 Bad Request: 게시 실패
Success Body	<success>: 성공했다는 표시
Fail Body	<failed>: 실패했다는 표시

Table 15: Post Consulting

*Consulting Mediate

Method	GET
URL	/consulting/post: id
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <consulting> <id>: 해당 글에 주어지는 id <data collection> <content>: 게시글 내용 분석을 위한 텍스트 <user info>: 사용자의 성향 분석을 위한 텍스트

If Success	200 OK
If Fail	400 Bad Request: 내부 처리 오류
Success Body	<consulting info> <user info>: 유저에 대한 다양한 정보들 <consulting vet>: 상담에 응한 수의사 정보 <id>: 해당 사용자의 id
Fail Body	<failed>: 실패했다는 표시

Table 16: Consulting Mediate

*Consult

Method	GET
URL	/consulting/user: id/vet: id
Request Body	<user> <id>: 각 사용자마다 주어지는 회원 id <vet> <id>: 해당 상담사에게 주어지는 회원 id <consulting info>: 상담에 사용된 다양한 정보들, 차후 User Data Collection에 보내진다.

If Success	200 OK
If Fail	400 Bad Request: 내부 처리 오류
Success Body	<success>: 성공했다는 표시 <consulting info> <log>: 상담 로그 <user satisfaction>: 상담에 대한 만족도
Fail Body	<failed>: 실패했다는 표시

Table 17: Consult

12.3.5. User Data Collection

해당 시스템은 다양한 시스템의 기능에서 나오는 사용자 및 다양한 주체들의 텍스트/이미지 정보 등을 모아 저장한다. 이는 당장 Recommendation System에 사용되기도 하며, 차후 다양한 기능 확장을 위해 활용의 여지가 많다.

*Review Collection

Method	GET
URL	/purchase/product: id/review: all
Request Body	<product> <id>: 해당 제품의 id <reviews>: 여러 개의 리뷰를 모으기 때문에 다단 항목 <id> <rating> <content>

If Success	200 OK
If Fail	400 Bad Request: 내부 처리 오류
Success Body	<success>: 성공했다는 표시 <processed content> <text> <tokenize>: 토큰화 한 리뷰 텍스트 집합
Fail Body	<failed>: 실패했다는 표시

Table 18: Review Collection

*Consulting Collection

Method	GET
URL	/consulting/post: all
Request Body	<post>: 여러 개의 상담 글을 모으기 때문에 다단 항목 <id> <content>

If Success	200 OK
If Fail	400 Bad Request: 내부 처리 오류
Success Body	<success>: 성공했다는 표시 <processed content> <text> <tokenize>: 토큰화 한 상담 텍스트 집합
Fail Body	<failed>: 실패했다는 표시

Table 19: Consulting Collection

*Adoption Collection

Method	GET
URL	/adoption/user: all/animal: all
Request Body	<user id> <animal id> <adoption log>: 입양 시 등록했던 정보들

If Success	200 OK
If Fail	400 Bad Request: 내부 처리 오류
Success Body	<success>: 성공했다는 표시 <processed content> <tokenized log>
Fail Body	<failed>: 실패했다는 표시

Table 20: Adoption Collection

13. Database Design

13.1 Objectives

Database Design은 요구사항 명세서에서 작성한 데이터베이스 요구사항을 기반으로 하여 세부적인 데이터베이스 설계를 기술한다. ER Diagram을 통해 개괄적인 Entity 간의 관계를 기술하고, 이를 통해 Relational Schema을 작성한 뒤, 최종적으로 SQL DDL을 작성한다.

13.2 ER diagram

개체(Entity)는 분리된 물체 하나를 표현한다. 개체는 사각형으로 표현되며, 관계(Relationship)는 다이아몬드로 표현된다. 개체나 관계는 특성(Attributes)을 가질 수 있으며, 이 특성들은 관계 집합에 실선으로 연결된 타원형으로 표현한다. 관계는 두개 이상의 개체들의 연관관계를 표현한다. 모든 개체는 고유하게 식별되는 특성 집합을 가지고 있어야 하며, 최소한의 고유 식별 특성 집합은 개체의 기본 키(Primary Key)라 불린다. 전체적인 ER Diagram은 다음과 같다

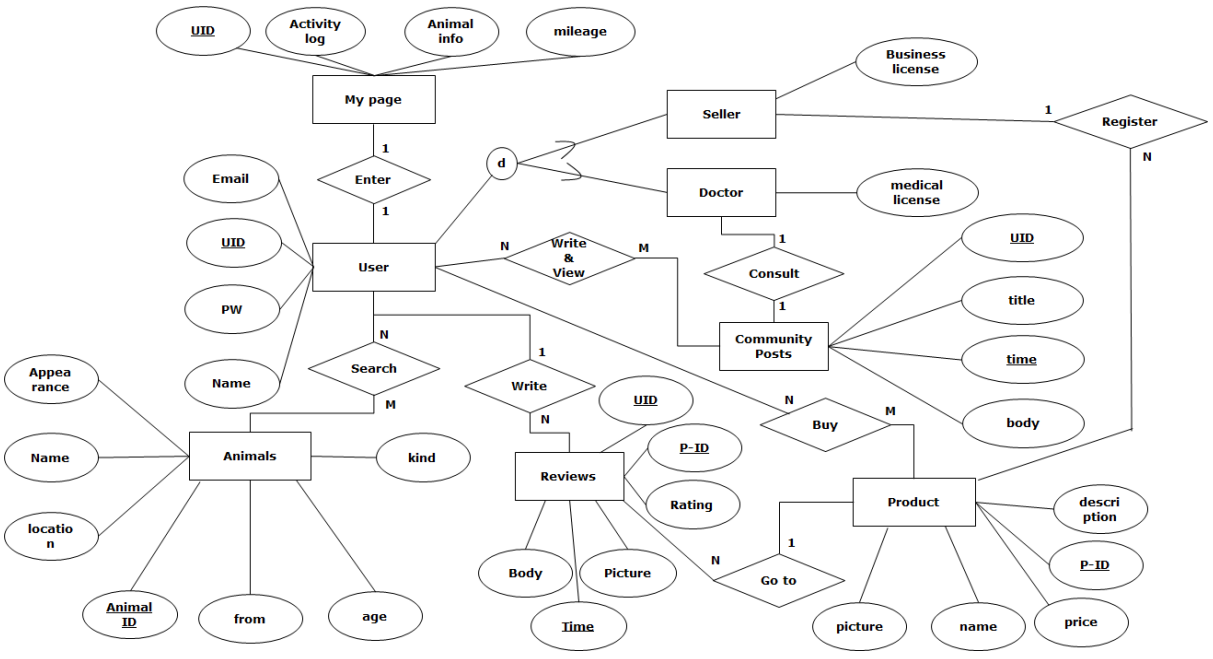

Diagram 46: ER Diagram for Petkage

13.3. Entities

13.3.1. User

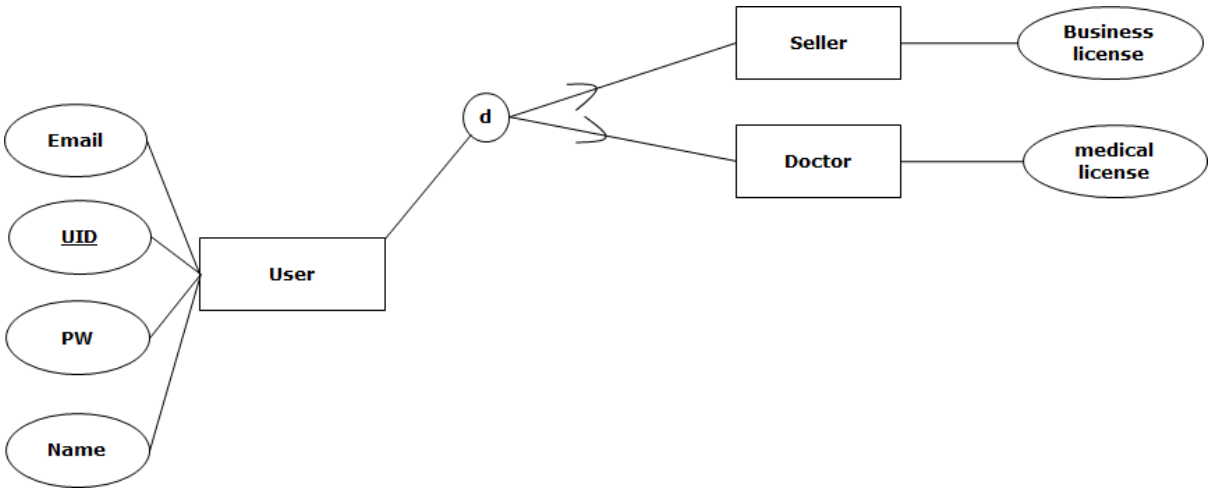

Diagram 47: ER Diagram (User)

User는 회원의 정보를 나타내는 데이터이다. 후보 키(Candidate Key)는 {UID(User ID), Email, Name} 이지만 편의상 Primary Key는 UID로 한다. 기본적으로 UID, PW, Name, Email의 속성을 갖고 있다. Seller와 Doctor는 Super-class인 User의 속성을 상속받고, 추가적으로 각각 Business license와 medical license의 속성을 갖는다. Super-class에 속한 모든 멤버들이 반드시 특정 Sub-class에 속할 필요는 없으며, 여기서는 일반 유저가 이에 해당한다.

13.3.2. My page

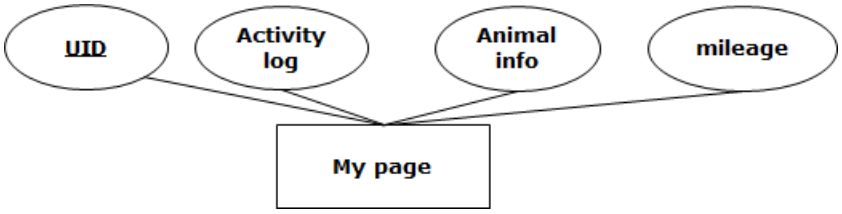

Diagram 48: ER Diagram (My page)

My page는 유저의 개인 공간에 대한 데이터를 담고 있다. Primary Key는 UID이고, 속성으로는 활동로그, 개인 반려동물 정보, 마일리지, 유저 아이디가 있다.

13.3.3. Animals

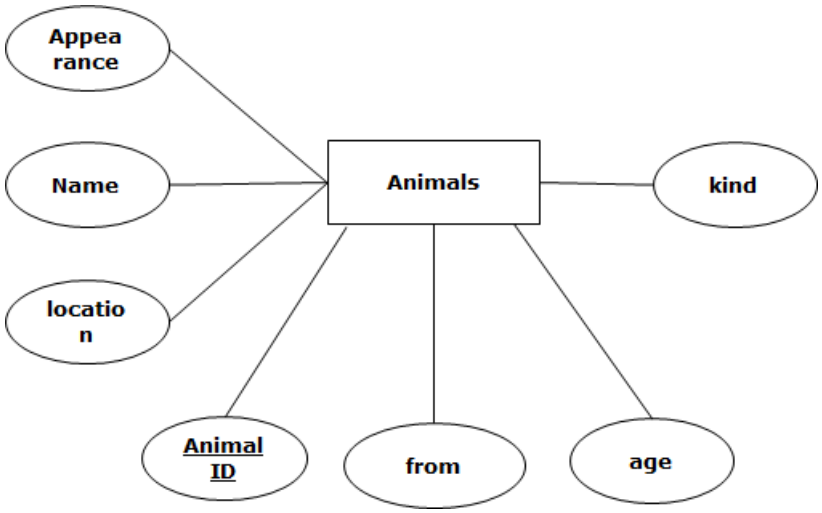

Diagram 49: ER Diagram (Animals)

Animals는 반려동물 분양/입양에 관심이 있는 유저들을 위한 반려 동물들의 데이터를 담고 있는 개체이다. 반려 동물의 특성상 Primary Key를 찾기 힘들기 때문에 인위적인 후보 키(Animal ID)를 Relation에 추가해 Primary Key로 한다. Appearance, Name, Location, Kind, Age, Animal ID, from의 속성을 갖고 있다.

13.3.4. Reviews

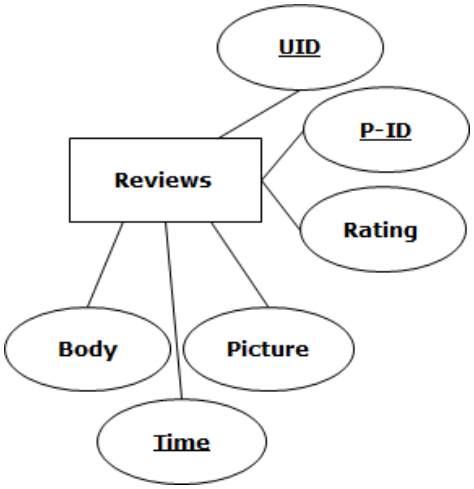

Diagram 50: ER Diagram (Reviews

유저가 상품을 구매하고 해당 상품에 대한 평가를 담은 데이터이다. Primary Key는 Time 이고, Time, Rating, Picture, Body, P-ID, UID의 속성을 갖고 있다.

13.3.5. Product

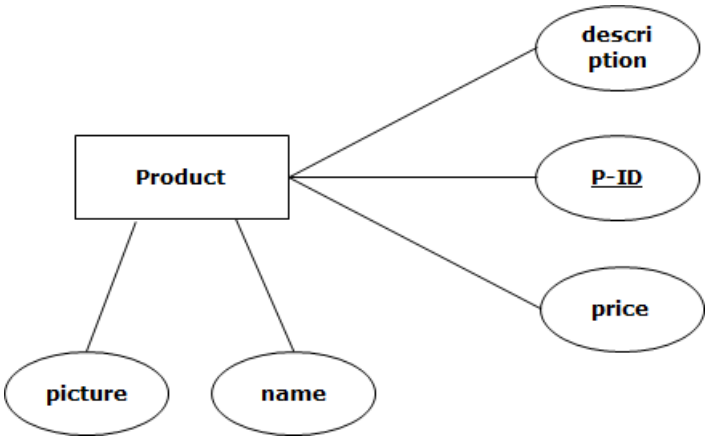

Diagram 51: ER Diagram (Product

Seller가 게시한 반려 동물 용품에 대한 정보이다. Primary Key는 P-ID이고, description, price, name, picture에 대한 속성을 갖고 있다. 추후에 다시 얘기하겠지만, description에는 유저가 작성한 review내용까지 포함한다.

13.3.6. Community posts

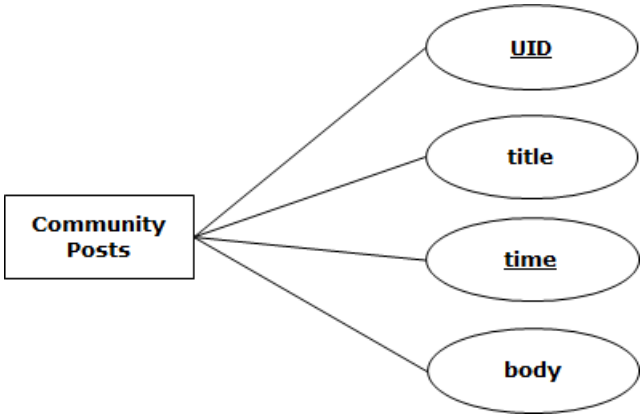

Diagram 52: ER Diagram (Community posts)

반려 동물을 키우는 사람들끼리 반려동물에 대한 정보와 고민 등을 담은 게시 글들의 정보이다. Primary Key는 time이고, UID, title, time, body의 속성을 갖고 있다.

13.4. Relationship

13.4.1. Enter (my page)

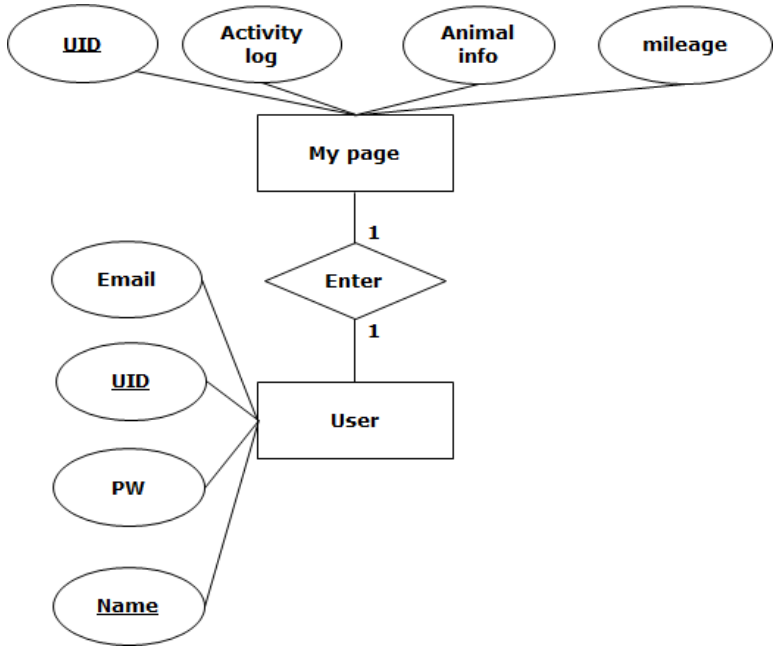

Diagram 53: ER Diagram (Enter my page)

유저가 개인공간인 my page에 접속하는 관계이다. 한 명의 유저는 하나의 my page만 접속할 수 있으며, 하나의 my page는 유저 한명에게만 공개된다 (one to one relation type)

13.4.2. Search (Companion animals)

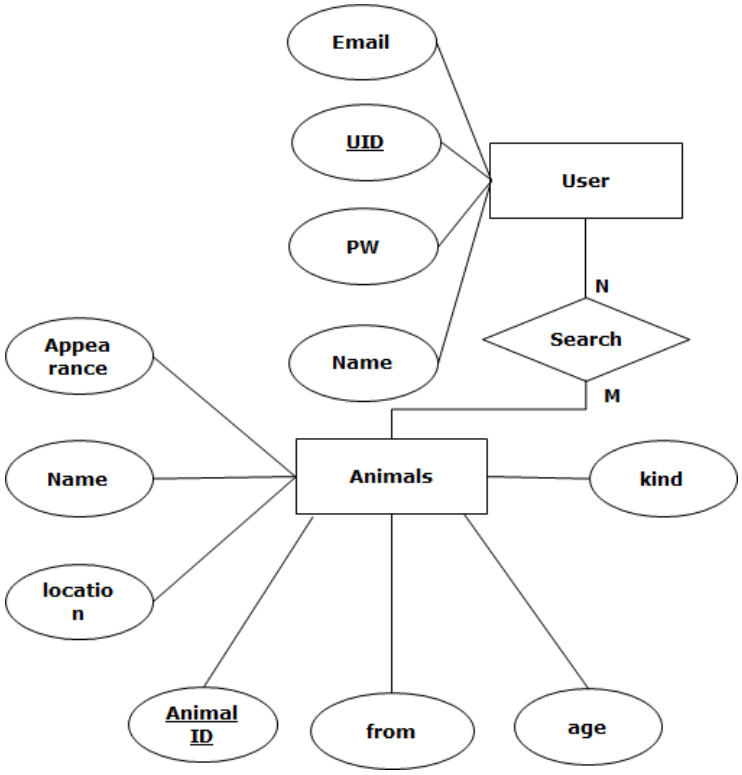

Diagram 54: ER Diagram (Search companion animals)

유저가 반려 동물을 분양 받기 위해 동물의 정보를 찾는 관계이다. 각 유저들은 여러 동물들의 정보를 확인할 수 있으며, 각 동물들의 정보는 여러 유저들에게 똑같이 공개된다. (many to many relation type)

13.4.3. Write (reviews)

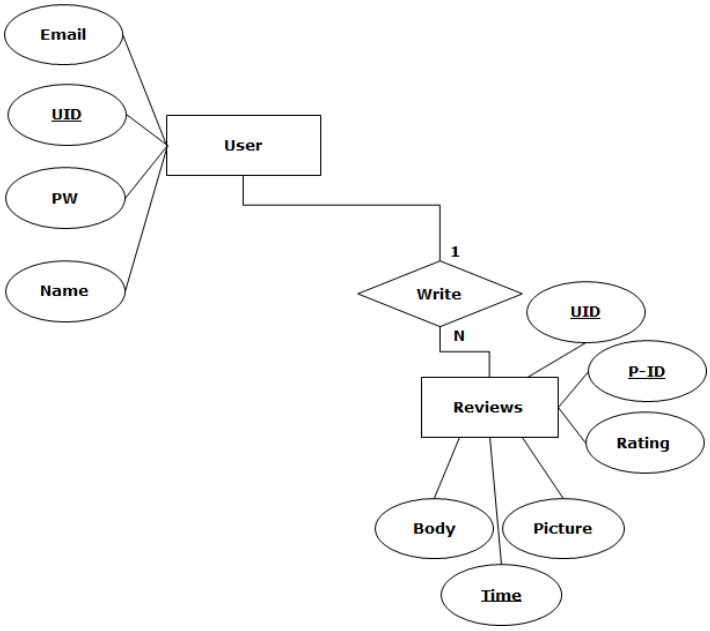

Diagram 55: ER Diagram (Write Reviews)

유저들이 상품을 구매하고 해당 상품에 대한 평가를 작성하는 관계이다. 각 유저들은 (구입한 기록이 있는) 상품들에 대해 여러 개의 평가를 남길 수 있지만, 하나의 리뷰는 1명의 유저에 의해 작성된다. (one to many relation type)

13.4.4. Write & View (posts)

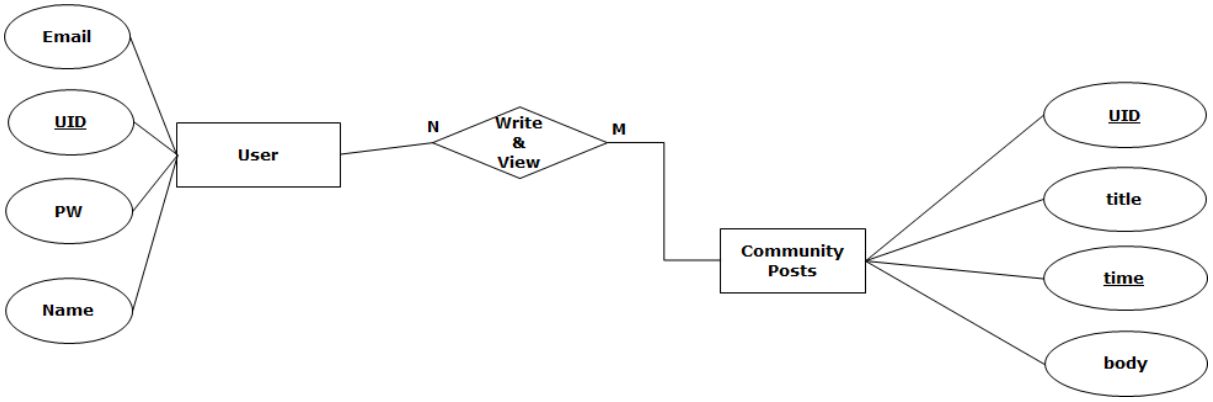

Diagram 56: ER Diagram (Write &View Posts)

유저들이 커뮤니티에서 반려동물과 관련된 글을 게시하고 정보를 공유하는 관계이다. 각 유저들은 여러 개의 게시글을 작성할 수 있으며, 각 게시글은 여러 유저들이 보고 확인할 수 있다 (many to many relation type)

13.4.5. Consult

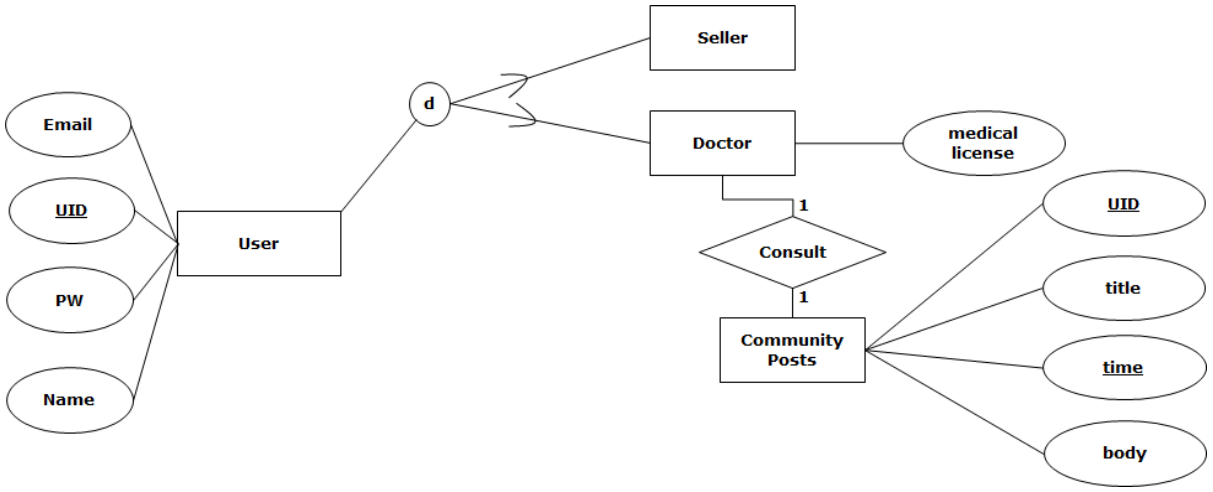

Diagram 57: ER Diagram (Consult

커뮤니티에서 고민을 해결할 수 없는 경우 인증 받은 전문의와의 상담을 통해 해결하 도록 도와주는 관계이다. 커뮤니티에서 각 유저는 한 명의 전문의와 상담할 수 있으며, 각 전문의는 한 명의 피상담인을 도와준다. (one to one relation type)

13.4.6. Buy

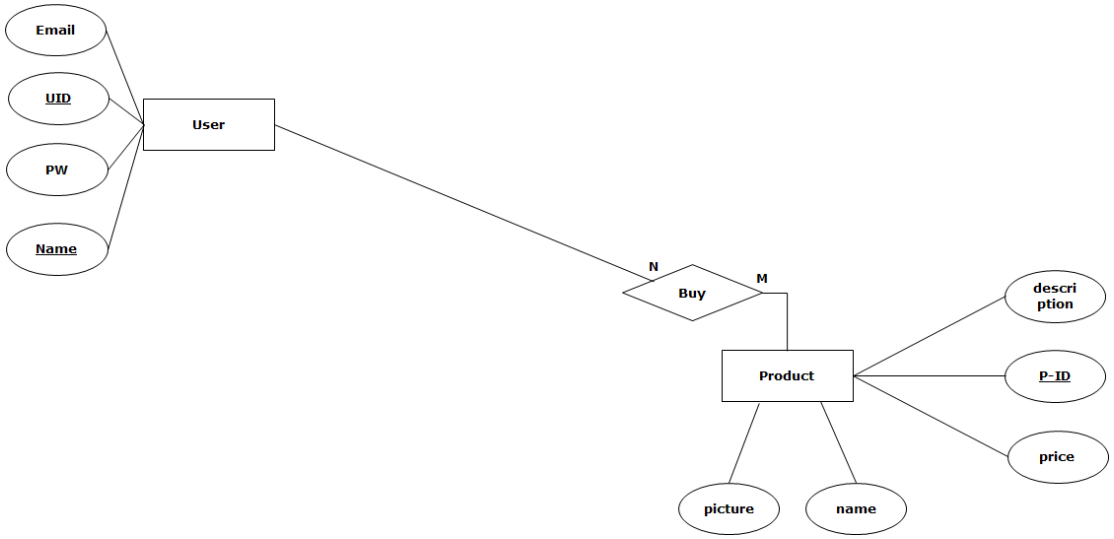

Diagram 58: ER Diagram (Buy)

유저가 원하는 반려동물 용품을 구입하는 관계이다. 각 유저는 원하는 만큼 여러 개의 상품을 구입할 수 있으며, 각 상품은 여러 유저들에게 판매될 수 있다. (many to many relation type)

13.4.7. Register

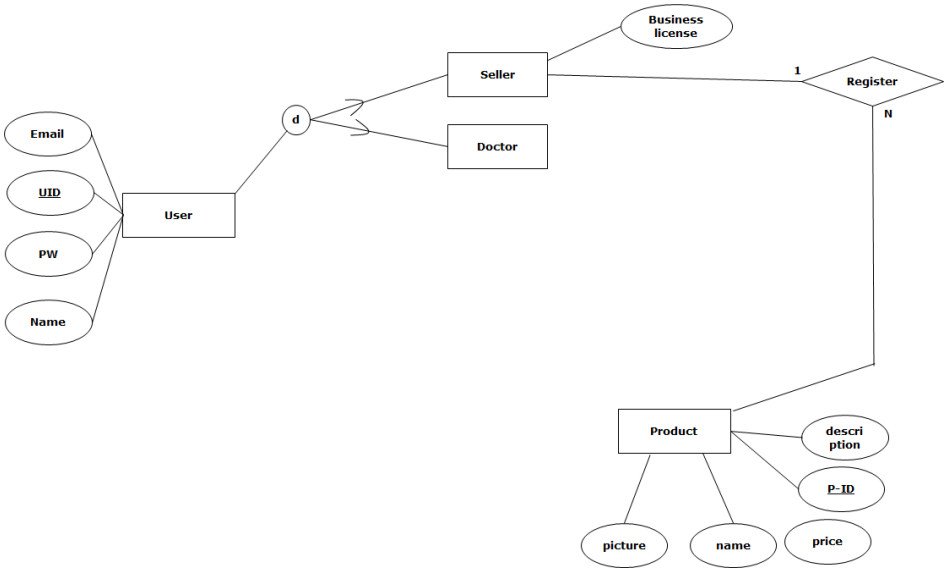

Diagram 59: ER Diagram (Register)

유저가 물품을 구입할 수 있게끔 상품을 등록하는 관계이다. 각 Seller들은 여러 상품을 등록할 수 있지만, 각 상품들은 한 명의 판매자에 의해 등록된다(one to many relation type)

13.4.8. (Reviews) Go to (product)

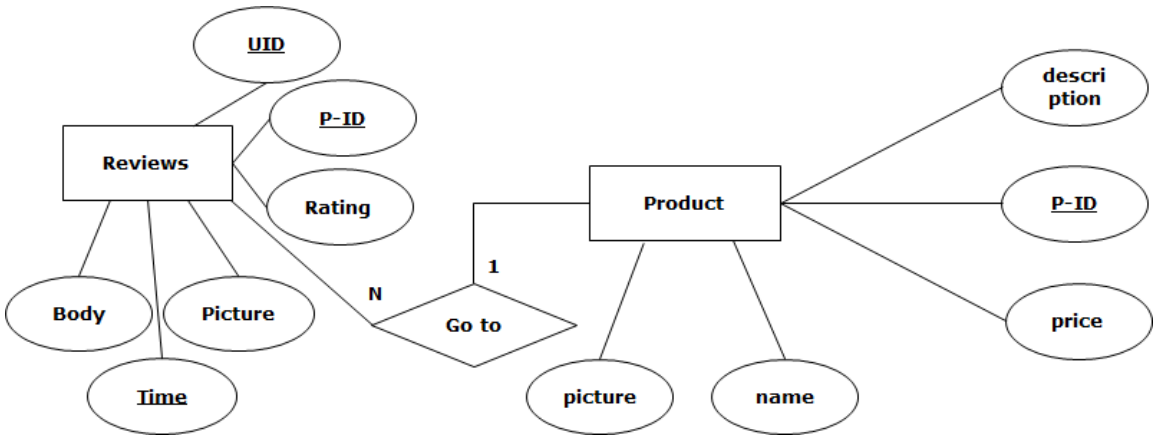

Diagram 60: ER Diagram (Review go to Product)

유저들이 작성한 리뷰가 상품의 정보로 들어가는 관계이다. 각 상품들에 대해 여러 리뷰가 작성될 수 있지만, 하나의 리뷰는 하나의 상품에 대한 정보만 담는다. (one to many relation type)

13.5. Relational schema

13.5.1. User

UID	PW	Name	Email

Table 21: Relational Schema (User)

Primary Key (PK): UID

Foreign Key (FK): None

Functional Dependency (FD): UID -> {PW, Name, Email}

Description: User에 대한 테이블이다. 모든 속성에 NULL값을 허용하지 않는다.

13.5.2. My page

UID	Activity log	Animal info	Mileage

Table 22: Relational Schema (My page)

Primary Key (PK): UID

Foreign Key (FK): UID -> User (UID)

Functional Dependency (FD): UID -> {Activity log, animal info, mileage}

Description: My page에 대한 테이블이다. Animal info와 Activity log에 NULL값을 허용한다. UID, mileage에 NULL값을 허용하지 않는다.

13.5.3. Animals

Animal_ID	Appearance	Name	Location	Age	Kind	From

Table 23: Relational Schema (Animals)

Primary Key (PK): Animal ID

Foreign Key (FK): None

Functional Dependency (FD): Animal ID -> {Appearance, Name, Location, Age, Kind, From}

Description: Animals에 대한 테이블이다. 모든 속성에 NULL값을 허용하지 않는다.

13.5.4. Reviews

Time	UID	P-ID	Rating	Body	Picture

Table 24: Relational Schema (Reviews)

Primary Key (PK): Time

Foreign Key (FK): UID -> User (UID), P-ID -> Product(P-ID)

Functional Dependency (FD): {Time, UID, P-ID} -> {Rating, Picture, Body}

Description: Reviews에 대한 테이블이다. 모든 속성에 NULL값을 허용하지 않는다.

13.5.5. Product

P-ID	Description	Price	Name	Picture

Table 25: Relational Schema (Product)

Primary Key (PK): P-ID

Foreign Key (FK): None

Functional Dependency (FD): P-ID -> {Description, Price, Name, Picture}

Description: Product에 대한 테이블이다. 모든 속성에 NULL값을 허용하지 않는다.

13.5.6. Community posts

Time	UID	Title	Body

Table 26: Relational Schema (Community Posts)

Primary Key (PK): Time

Foreign Key (FK): UID -> User (ID)

Functional Dependency (FD): {Time, UID} -> {Title, Body}

Description: Community posts에 대한 테이블이다. 모든 속성에 NULL값을 허용하지 않는다.

13.6. SQL DDL

13.6.1 User

```
CREATE TABLE User(
   UID VARCHAR(20) NOT NULL,
   PW VARCHAR(20) NOT NULL,
   Name VARCHAR(10) NOT NULL,
   Email VARCHAR(20) NOT NULL,
   PRIMARY KEY (UID),
   UNIQUE (Email)
   );
```

Table 27: SQL DDL (User)

13.6.2. My page

```
CREATE TABLE My_page(
   UID VARCHAR(20) NOT NULL,
   Activity_log TEXT,
   Amimal_info TEXT,
   Mileage INT NOT NULL DEFAULT 0,
   PRIMARY KEY (UID),
   FOREIGN KEY (UID) REFERENCES User(UID)
   ON DELETE CASCADE ON UPDATE CASCADE
);
```

Table 28: SQL DDL (My page)

13.6.3. Animals

```
CREATE TABLE User(
   Animal_ID VARCHAR(20) NOT NULL,
   Appearance IMAGE NOT NULL,
   Name VARCHAR(10) NOT NULL,
   Location TINYTEXT NOT NULL,
   Age INT NOT NULL,
   Kind VARCHAR(20) NOT NULL,
   From VARCHAR(40) NOT NULL,
   PRIMARY KEY (Animal_ID),
);
```

Table 29: SQL DDL (Animals)

13.6.4. Reviews

```
CREATE TABLE Reviews(
  UID VARCHAR(20) NOT NULL,
  Time DATETIME NOT NULL,
  P-ID VARCHAR(20) NOT NULL,
  Rating TINYINT NOT NULL,
  Picture IMAGE NOT NULL,
  Body TEXT NOT NULL,
  PRIMARY KEY (TIME),
  FOREIGN KEY (UID) REFERENCES User(UID)
  ON DELETE CASCADE ON UPDATE CASCADE
  FOREIGN KEY (P-ID) REFERENCES Product(P-ID)
  ON DELETE CASCADE ON UPDATE CASCADE
  );
```

Table 30: SQL DDL (Reviews)

13.6.5. Product

```
CREATE TABLE Product(
   P-ID VARCHAR(20) NOT NULL,
   Picture IMAGE NOT NULL,
   Name VARCHAR(20) NOT NULL,
   Price INT NOT NULL,
   Description TEXT NOT NULL,
   PRIMARY KEY (P-ID)
);
```

Table 31: SQL DDL(Product)

13.6.6. Community posts

```
CREATE TABLE Community_posts(
   Time DATETIME NOT NULL,
   UID VARCHAR(20) NOT NULL,
   Title VARCHAR(20) NOT NULL,
   Body TEXY NOT NULL,
   PRIMARY KEY (Time),
   FOREIGN KEY (UID) REFERENCES User(UID)
   ON DELETE CASCADE ON UPDATE CASCADE
);
```

Table 32: SQL DDL(Community Post)

14. Testing Plan

14.1. Objectives

이 문서는 Petkage 개발 과정에서 진행되는 다양한 테스트 계획을 모두 총망라한 Master Testing Plan에 관한 내용이다. 테스트 과정을 전반적으로 정리함으로써 Parallel Development에서 발생할 수 있는 다양한 일정 변경으로 인한 혼란을 사전에 방지하고, 개발이 될 때마다 표준화된 전략으로 서브 시스템을 평가하고 성능 측정할 수 있도록 가이드라인을 제시한다. Petkage MVC로 이루어진 큰 틀의 아키텍처 아래에 Front/Backend 구조가 존재하며, 해당 시스템들 내부에 다양한 Template(사용자에게 보여주는 페이지)와 Module(내부적인 기능들)이 존재하므로 각 기능들이 개발됨에 따라 발생할 수 있는 기능적인 문제, 연계로 인한 문제 등을 최대한 빠르고 효율적이게 처리해야 한다. 그러므로 하나의 표준화된 지침을 해당 챕터에서 잡아 혼란을 최소화하고 테스트에 표준화된 문서를 정립한다.

14.2. Tasks and Scope

테스트를 위해선 다양한 업무들이 존재한다. Petkage 개발에서는 크게 3가지의 테스팅 업무를 통해 테스트할 예정이다: Development Testing, Release Testing, User Testing이다.

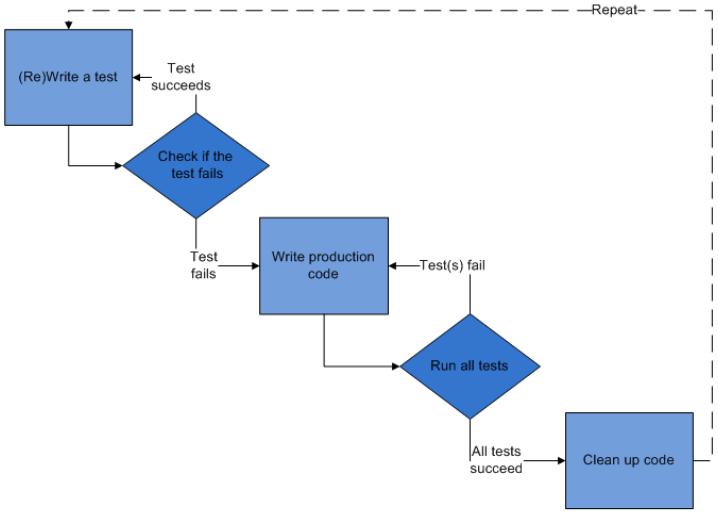

Figure 12: TDD

우선 Development Testing은 병렬 개발이 진행되면서 각 서브 시스템을 개발하거나 통합하는 과정에서 일어날 수 있는 잠재적인 문제점이나 오류 등을 해결하기 위해 코드를 분석하거나 내부 서버에서 돌려보는 테스팅 방법이다. 해당 테스트를 통해 개발자들은 간략하게 코드가 어떻게 작동하는지 시각적으로 이해할 수 있으며, 다양한 서브시스템이 연계됨에 따라 발생할 수 있는 Synchronizing 문제나 데이터 통일성 등의 문제를 발견할 수 있다. TDD(Test-driven Development)를 추구하는 Petkage에 유용한 테스트 방법이다.

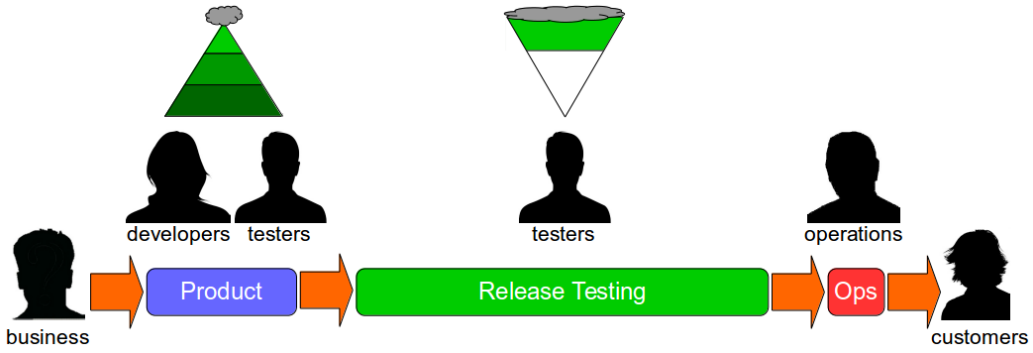

Figure 13: Release Testing

다음으로 Release Testing은 지속적으로 새로운 패치 버전을 배포함으로써 시스템에 존재하는 버그나 오류 등을 지속적이고 반복적으로 고치기 위한 테스팅이다. 배포 버전이 올라감에 따라 기존에 존재했던 오류나 문제 등을 해결하고, 더 완강히 연결된 하나의 아키텍처를 갖출 수 있다.

Figure 14: User Testing

마지막으로 User Testing은 게임에서 진행하는 알파/베타 테스트처럼 실험대상이 되는 고객의 표본을 구한 뒤, 배포된 Petkage의 버전을 직접 체험하면서 평가를 듣는 방식이다. 고객들은 시스템을 사용하는 가장 최전방의 이용자들이므로 제품에 대한 가장 솔직한 리뷰를 들을 수 있으며 개발에서 미처 발견하지 못한 요소들을 발견하는 데에 유용하다.

한편, Petkage에서 테스트해야 하는 Scope는 크게 3가지로 나뉘는데, Frontend에서 개발할 각각의 페이지, Backend에서 개발할 각각의 모듈들, 그리고 DB와 같은 기타 구성요소들이다.

-Frontend: 프런트 엔드의 경우 사용자에게 보여지는 다양한 페이지들을 개발하기 때문에 해당 페이지들의 디자인이 다양한 화면과 사양에서 제대로 표시되는지 확인해야 한다.

-Backend: 백 엔드는 시스템에 가장 핵심적인 모듈들이 존재하는 곳이기 때문에, 해당 모듈들이 그 어떤 경우에도 제대로 작동하는지, 결함 없이 작동하는지 등을 살펴봐야 한다.

-DB등의 구성요소: DB같은 경우 사용자들의 정보가 담기는 장소이므로 안정성이 매우 중요시된다.

한편, 시스템을 지원하는 다양한 Supporting Module의 경우 시스템을 운영할 때 결함 없이 제대로 작동하는지를 살펴봐야 한다.

14.3. Strategy

Petkage를 테스트하는 데에 가장 중요하게 생각해야 할 요소는 다음의 4가지이다: Efficiency, Dependability, Maintainability, 그리고 Acceptability. 각각은 시스템의 속도/보안성/유지보수성/수용성에 관한 내용이며, 좋은 소프트웨어가 갖춰야 할 필수 요소들이다. 하지만 테스트에서 이 모든 것을 완벽히 파악하기란 쉽지 않은 일이기 때문에, 해당 파트에서는 중점적으로 봐야 할 전략을 서술한다. 한편, Acceptability의 경우 우리 소프트웨어에 큰 위협 요소가 되지 않기 때문에 따로 전략을 세우기보단 시스템 성능에 최대한 집중하도록 한다.

우선, Efficiency의 경우 Speed와 Accuracy를 중심으로 측정한다. 웹 어플리케이션은 이미 고도로 발달된 네트워크를 기반으로 최대한 빠른 시간 내에 사용자에게 반응하는 반응형 웹이 대세이기 때문에, Petkage 또한 이에 맞춰 최대한 빠른 속도로 정확하게 사용자와 상호작용해야 한다. 이를 고려하여 페이지 테스팅을 할 때 지표를 설정해야 한다.

다음으로 Maintainability는 향후 System Evolution이나 Requirement Specification에서 발생할 수 있는 다양한 변동사항들을 중심으로 살펴봐야 한다. Petkage 자체는 MVC 모델이라는 간소화된 디자인 패턴을 차용하기 때문에 유지보수와 확장에 큰 문제가 없지만, 개발이 거듭될수록 이러한 특징이 퇴색되고 개발 모듈의 구성이 꼬일 수 있다. 이를 방지하기 위해 테스트 과정에서 모듈이나 패키지 디자인에 대한 표준화된 구성을 정의함으로써 추가적인 기능이 전체 시스템에 자연스럽게 녹아들 수 있도록 유도한다.

한편, Security의 경우 다양한 네트워크 환경에서 서버에 위협이 될 수 있는 다양한 해킹 위험이나 내부 서버에서 발생할 수 있는 오류 처리에 초점을 맞춰야 한다. 가령, DB같은 경우 하나의 DB만을 갖출 시 만일의 사태가 발생할 때 복구하고 처리하는 시간이 많이 들 뿐더러 비용 또한 만만치 않다. 그러므로 Hadoop의 삼중 복제 시스템처럼 데이터베이스 자체를 복제/저장하여 리스크 발생 시의 위협을 최소화 하거나, 별도로 Fault Tolerance 기능을 추가해 시스템을 서포트 하는 다양한 기능들이 항상 제기능을 할 수 있도록 신뢰도 지표를 갖춰야 한다.

이러한 테스트 전략은 Scrum이라는 반복 개발 관리 팀 시스템을 도입해 최적화한다. 테스트하는 과정에서 외부의 요구나 충격으로 인한 시스템 변동사항 없이 테스트 최적화에 몰두할 수 있도록 Scrum Master는 해당 테스트 팀을 보호해야 한다.

14.4. Environmental Requirements

테스트 환경을 위해서 팀 내부적으로 가상화 된 서버를 생성해 테스트한다. 이 때, Hypervisor (HV)를 동반한 반가상화를 통해 속도와 안정성을 갖춘 운영체제 위에서 해당 서버를 운영할 계획이다. 세부적인 하드웨어나 소프트웨어 명세서는 이후 단원에서 자세히 다루도록 한다.

14.5. Test Schedule

테스트 스케줄은 각각의 서브 시스템이 끝나는 대로 진행한다. 이는 각 서브 시스템의 개발이 끝난 이후 1~2주에 걸쳐 진행되며, 세부적인 일정은 이후 단원에서 자세히 다루도록 한다.

15. Development Plan

15.1. Objective

Development Plan에서는 실제 시스템 개발을 위해 필요한 개발 환경과 패키지, 외부 API(Application Programming Interface)등에 대하여 서술한다. 그리고 개발 일정에 대한 내용도 서술한다.

15.2. Development Environment

15.2.1. Bootstrap

Figure 15: Logo of Bootstrap

Bootstrap은 웹사이트를 쉽게 만들 수 있게 도와주는 HTML, CSS, JS 프레임워크이다. 하나의 CSS로 휴대폰, 태블릿, 데스크탑까지 다양한 기기에서 작동한다. 다양한 기능을 제공하여 사용자가 쉽게 웹사이트를 제작, 유지, 보수할 수 있도록 도와준다. Petkage의 프런트 엔드를 구축하는데 사용된다.

15.2.2. Django

Figure 16: Logo of Django

Django는 Python으로 작성된 오픈소스 웹 애플리케이션 프레임워크로, 모델-뷰-컨트롤러(MVC) 패턴을 따르고 있다. 현재는 장고 소프트웨어 재단에 의해 관리되고 있으며, 컴포넌트의 재사용성(reusability)과 플러그인화 가능성(pluggability), 빠른 개발 등을 강조하고 있다.

15.2.3. MySQL

Figure 17: Logo of MySQL

MySQL은 세계에서 가장 많이 쓰이는 오픈 소스의 관계형 데이터베이스 관리 시스템이다. 다중 스레드, 다중 사용자 형식의 구조 질의어 형식의 데이터베이스 관리 시스템으로서 오라클이 관리 및 지원하고 있으며, Qt처럼 이중 라이선스가 적용된다. Petkage에서 데이터베이스 구축을 위해 사용된다.

15.2.4. 동물보호관리시스템 유기동물 조회 서비스(외부 API)

Figure 18: Open API for adoption system

Petkage에서 동물 분양 기능에 활용될 동물들의 정보를 얻기 위하여 공공 긴급 유기동물 API(지자체 제공 및 오픈소스)를 사용할 것이다. 현재 사용 계획이 있는 API는 다음과 같다.

https://www.data.go.kr/data/15001096/openapi.do

15.2.5. R

Figure 19: Logo of R

R은 통계와 그래픽을 위한 프로그래밍 언어이자 소프트웨어 환경이자 프리웨어다. 통계 소프트웨어 개발과 자료 분석에 널리 사용되고 있으며, 패키지 개발이 용이해 통계 소프트웨어에 쓰인다. Petkage에서는 Recommendation System이 사용하는 Apriori 연관규칙에 사용될 것이다.

15.3. Schedule

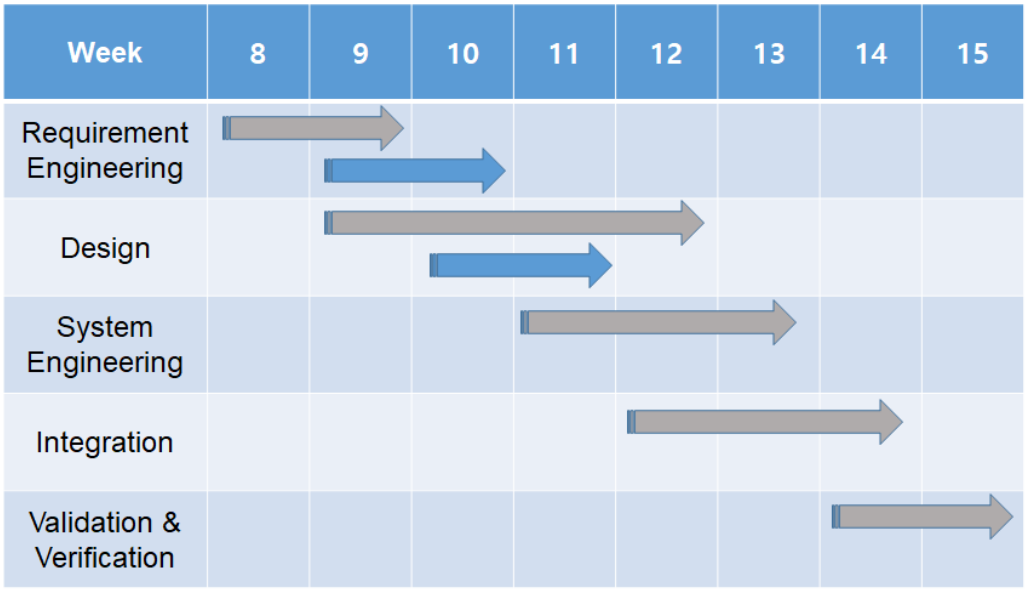

Figure 20: Schedule

회색으로 표시된 부분이 Proposal때 계획한 것이고, 파란색으로 표시된 부분이 현재까지의 개발 현황이다. 기존에 계획했던 것보다 Design측면을 빠르게 작업했지만, 12주차 등에도 작업할 여지가 있다. 실제 일정이 바뀌게 될 경우 차트를 갱신한다.

16. Index

16.1. Tables

16.2. Figures

16.3. Diagrams