Design Specification

TEAM_#9

2014312860_전민용

2015310422_민세원

2014312822_김호승

2017313120_정태우

2015313526_이나래

2012311142_김도연

목차

1. Preface

1.1 Objective

Preface에서는 본 문서의 예상되는 독자들과 문서의 전반적인 구조, 그리고 각 부분의 역할에 대하여 제시한다. 그리고 버전 관리 정책, 버전 변경 기록, 그리고 문서의 변경사항들과 그에 대한 근거들을 서술한다.

1.2 Readership

본 문서의 독자로는 각 서브 시스템을 개발하는 소프트웨어 엔지니어, 시스템을 설계하는 아키텍쳐와 개발에 참여하는 클라이언트 엔지니어, 고객 기술 지원을 위한 서비스 팀 등 본 문서에서 소개할 시스템과 관련된 모든 구성원들을 독자로 정의한다. 개발 과정에 외부업체를 이용하게 된다면, 해당 업체에 관련되는 모든 구성원 역시 독자에 포함된다.

1.3 Document Structure

A. Preface

Preface에서는 본 문서의 예상되는 독자들과 문서의 전반적인 구조, 그리고 각 부분의 역할에 대하여 제시한다. 그리고 버전 관리 정책, 버전 변경 기록, 그리고 문서의 변경사항들과 그에 대한 근거들을 서술한다.

B. Introduction

Introduction에서는 본 문서에서 시스템을 설계할 때 사용된 모든 종류의 다이어그램과 도구(툴)에 대하여 사용자 관점으로 서술한다.

C. System Architecture

System Architecture에서는 팀에서 개발하고자 하는 시스템에 대하여 전반적으로 서술한다. 먼저 시스템의 전체적인 구조를 설명하고, 시스템을 Block diagram으로 나타내어 각각의 관계와 실제 사용 과정을 Package diagram과 Development diagram을 이용하여 설명한다.

D. User Management System

실제 사용자가 사용하며 회원 가입과 로그인 기능을 통해 시스템을 이용하는 도중에

발생하는 데이터를 처리하는 사용자 관리 시스템의 설계를 설명한다. Class diagram, Sequence diagram, State diagram을 통해 User Management System의 구조를 표현하고 설명한다.

E. Survey System

추천시스템이라는 것은 사용자의 선호도에 따른 맞춤형 서비스이기 때문에, 사용자가 처음 서비스를 이용할 때 필요한 사용자의 선호도를 조사(Survey)를 진행한다. 사용자가 입력한 데이터를 처리하고 만약 수정 사항이 발생하였을 때, 이를 처리하는 Survey system의 설계를 설명한다. Class diagram, Sequence diagram 과 State diagram을 통하여 Survey System의 구조를 표현하고 설명한다.

F. Profile Customizing System

사용자가 My profile 탭에서 사용자 정보를 커스터마이징(Customizing)하기 위해, 필요한 데이터베이스에 접근하여 해당 정보를 출력하며 필요한 경우, 수정을 하는 시스템의 설계를 설명한다. Class diagram, Sequence diagram, State diagram을 통해 Profile Customizing System의 구조와 사용자와의 상호작용을 표현 및 설명한다.

G. Search System

User가 Product를 검색할 때, 구동되는 시스템의 설계를 설명한다. 검색을 할 때, 어떠한 검색 필터를 이용하는 지에 따라, 그 결과값들이 상이하게 나올 수 있다. 따라서 그것을 case로 설정하여 Class Diagram, Sequence Diagram, State Diagram으로, Search System의 구조와 'Shoose'와의 상호 작용을 표현하고 설명한다. 검색 필터는 크게, Product Search와 Category Search로 나뉜다.

H. Recommend System

사용자가 실제 상품을 리스트로 추천해주는 시스템을 설명한다. 사용자가 넘겨주는 상황 정보에 따라 추천해줘야 하는 정보가 달라지기 때문에 이를 Class diagram, Sequence diagram, State Diagram을 통해 Recommend System의 구조를 표현하고 설명한다.

I. Tournament System

사용자가 선호하는 제품을 Tournament 방식으로 고를 수 있는 시스템을 설명한다. Recommend System에서 넘어온 제품 리스트를 랜덤하게 매칭하여서 최종 선호 제품을

고르거나 도중에 list로 받기 때문에 이를 case로 나누어 Class diagram, Sequence diagram, State Diagram을 통해 Tournament System의 구조를 표현하고 설명한다

J. Link System

사용자가 해당 제품을 사기를 원하거나 마음에 든 제품을 파는 쇼핑몰에서 다른 제품들을 보기 원할 때, 그 쇼핑몰로 연결해주는 시스템이다. Class diagram, Sequence diagram, State Diagram을 통해 Link System의 구조를 표현하고 설명한다

K. Protocol Design

Protocol Design에서는 Subsystem들이 상호작용하는 프로토콜에 대해 서술한다. 프로토콜의 기본 형식은 JSON을 기본으로 하며, 통신하는 메시지(Message)의 형식과 용도, 의미를 설명한다.

L. Database Design

Database Design은 요구사항 명세서에서 기술한 데이터베이스 요구사항을 바탕으로 수정사항을 수정하여 다시 요구사항을 작성하였다. 요구사항을 바탕으로 ER Diagram을 작성하고, 이를 이용하여 Relational Schema를 작성하고, Normalization을 통해 Redundancy와 Anomaly를 제거한 후 마지막으로 SQL DDL을 작성한다.

M. Testing Plan

Testing Plan은 시스템이 의도한 방향으로 실행되는지를 확인하고 시스템 내부의 결함을 찾기 위해 testing 한다. 이러한 testing을 설계 단계에 계획한다. 이 때 Testing Plan에서는 Testing Policy 뿐만 아니라 여러 test case에 대해 기술한다.

N. Develop Plan

Develop Plan에서는 개발 계획에 대해 서술한다. 시스템의 개발 계획과 실제 개발 흐름에 대해서는 Gantt chart를 활용하여 서술한다.

O. Index

Index에서는 본 문서의 도표, 다이어그램 및 삽입 이미지들의 인덱스를 표기한다. 이를 통해 원하는 정보가 문서의 몇 페이지에 있는지 알 수 있다.

1.4 Version of the Document

A. Version Format

해당 문서의 Version은 0.1부터 시작하여 <Major number>.<Minor number>의 형식으로 표현된다.

B. Version Management Policy

'Design Specification (설계 명세서)' 가 수정될 때마다 버전을 업데이트 한다.
새로운 파트가 추가 되거나 문서의 구성이 변경될 경우에는 Major number를, 이미 완성된 파트를 변경, 보완 하는 경우에는 Minor number를 변경한다.

C. Version Update History

Version	Modified Date	Explanation
0.1	2018.10.31	설계 명세서 각 부분에 들어갈 내용에 대한 초안 작성 및 분담
1.0	2018.11.07	Preface, Introduction 초안 작성, 문서 세부사항 추가 논의 및 개발 스케쥴 정리
2.0	2018.11.09	System Architecture, User Management, Protocol Design, Search System, Recommend System, Tournament System, Link System 초안 작성
3.0	2018.11.10	Database, Profile Customizing System 작성
3.1	2018.11.10	Introduction 수정 및 추가
4.0	2018.11.11	Test plan 작성, Profile Customizing System 수정
5.0	2018.11.11	Index 및 목차 작성, 문서 통합

Table 1 Version of the Document

2. Introduction

2.1 Objective

Introduction에서는 본 문서에서 시스템을 설계할 때 사용된 모든 종류의 다이어그램과 도구(툴)에 대하여 사용자 관점으로 서술한다.

2.2 Applied Diagram

A. UML

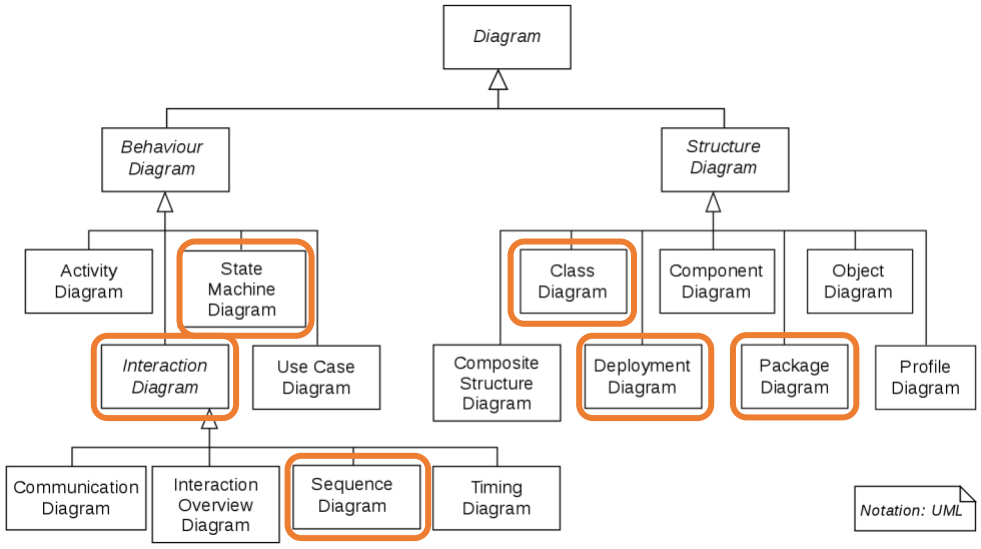

Figure 1 UML Diagrams Hierarchy

UML (Unified Modeling Language)는 소프트웨어 설계 단계에서 사용되는 통합 모델링 언어라는 뜻으로, 소프트웨어의 구조를 시각적인 모델로 표현하는 언어이다. 1997 년말에 OMG (Object Management Group)에 의해 표준 모델링 언어로 선정되었고, 2005 년 ISO (International Organization for Standardization)에 의해 국제 표준으로 선정되었다.

UML 의 궁극적인 목적은 시스템의 설계 및 구현 등 개발 과정 상에 이해당사자들 간의 의사소통을 원활하게 이루어지게 하기 위함에 있다. 시스템의 복잡성으로 인해 표준적인 표기법이 필요했고, 기존에 객체지향 분석/설계 방법론에 표준이 존재하지 않았기 때문에

등장하게 된 모델링 언어이다. 즉, 객체 지향 설계를 위한 표준 설계 언어이자, 하나의 시스템을 모델로 표현하기 위한 모든 산출물을 규정, 시각화, 상세화, 문서화하는 모델링 언어이다.

또한 UML 은 개발할 시스템의 유형, 개발 방법론, 개발에 사용되는 모든 프로그래밍 언어와 도구에 관계없이 적용될 수 있는 일반적인 표현 방법이기 때문에, 본 문서에서도 UML 을 사용하여 시스템 설계를 하였다.

B. Package Diagram

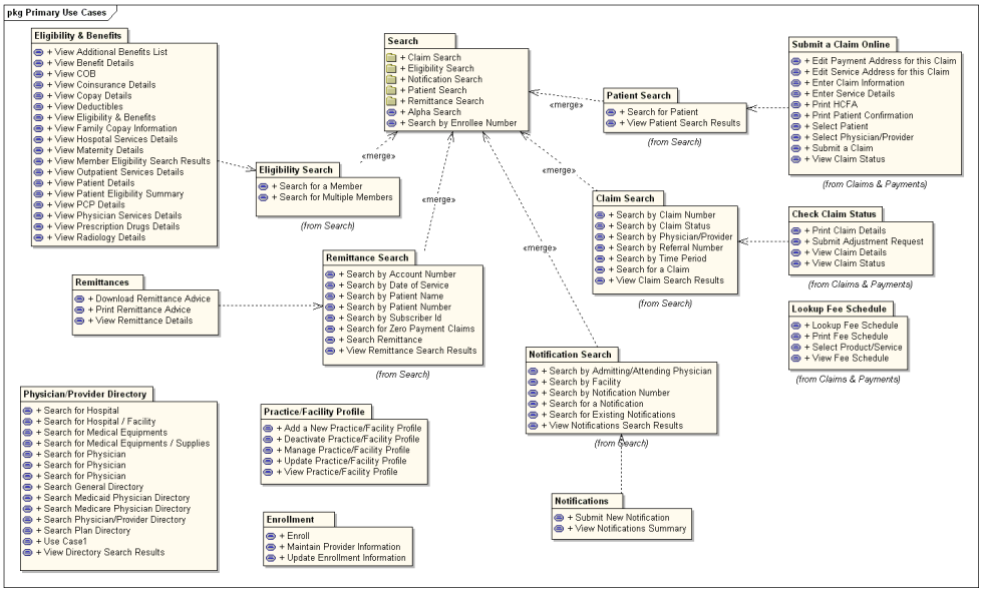

Figure 2 Package Diagram Example

시스템을 이해하기 위해서는 추상적인 개념들을 관련있는 요소들 끼리 묶는 수단이 필요한데, 이렇게 시스템 이해의 목적으로 클래스들을 그룹화하는 것을 UML 에서는 'Package' 라고 하고, 이를 통해 나타낸 것을 Package Diagram 이라 한다. Package 내부의 모든 클래스들은 개념적, 기능적, 변화적, 관리적 측면에서 유사한 성질을 가지고, Package 의 대상은 클래스에만 국한되는 것이 아니라 Use case, Activity diagram 등의 여러 모델 요소들과 다른 패키지들도 다 포함 할 수 있다. 하지만 각 구성 요소들은 오직 하나의 Package 에만 포함될 수 있다.

Package 의 내부의 요소들은 밀접한 관련성을 가지고 (높은 응집도), 다른 Package 의 요소들과는 약한 의존관계를 가진다 (독립적). Package Diagram 은 이러한 Package 들과 Dependency 라는 두가지의 요소로 표현된다. Dependency 는 시스템 내의 서로 다른 Package 들 사이의 관계를 표현하는 것이다. Package Diagram 은 이렇게 시스템의 모델 요소들을 그룹화하고 시각적으로 제공함으로써 복잡한 시스템 구조를 이해하는데 도움을 제공할 수 있다.

C. Deployment Diagram

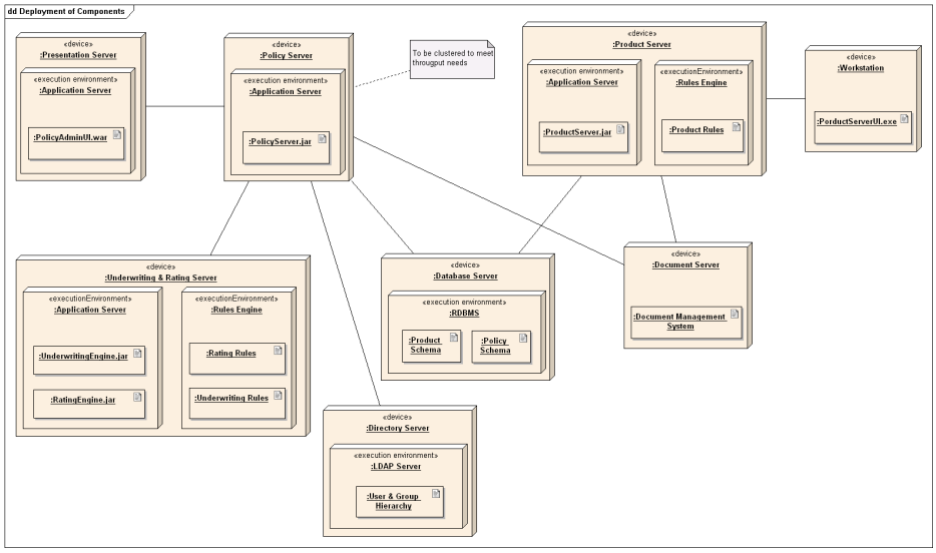

Figure 3 Deployment Diagram Example

Deployment Diagram 은 UML 중 에서 시스템의 물리적인 아키텍쳐를 모델링한 것으로, 시스템의 소프트웨어와 하드웨어 컴포넌트간의 관계 및 처리의 물리적 분배를 표현한다. 일반적으로 설계의 마지막 단계 또는 개발 구현 단계에서 작성된다.

Deployment Diagram 은 분산 시스템에서 node 의 물리적 배열과 각 node 에 저장되는 artifact 및 artifact 가 구현하는 component 나 기타 요소를 표시한다. 여기서 node 는 시스템의 런타임 환경을 지원하는 기타 장치 뿐 아니라, 컴퓨터, 센서 및 프린터와 같은 하드웨어 자원를 나타내고, artifact 는 실행파일, 라이브러리, 소프트웨어 구성 요소, 문서 및 데이터베이스와 같은 소프트웨어 시스템의 물리적인 엔티티를 나타내는 모델요소이다.

Component Diagram 이 시스템의 component 와 artifact 를 정의한다면, Deployment Diagram 은 하드웨어 자원을 정의하고, 정의한 component 와 artifact 가 어떤 하드웨어 자원에 탑재되어 실행될지 배치된 시스템을 표시한다는 점이 구별된다.

D. Class Diagram

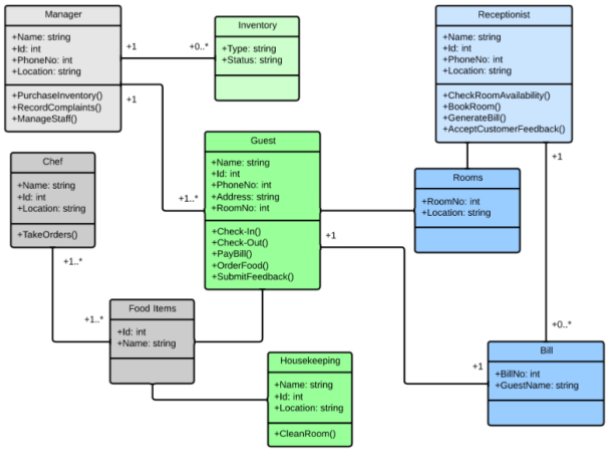

Figure 4 Class Diagram Example

Class Diagram 도 마찬가지로 UML 의 한 종류이자 객체 지향 모델링의 기본 모델로, 시스템의 클래스 내부의 정적인 내용이나 Class 사이의 관계를 표기하는 다이어그램이다. 시스템의 일부 또는 전체의 구조를 나타낼수 있고, Class 들 간의 관계와 의존관계를 쉽고 명확하게 파악할 수 있게 한다.

Class Diagram 에서의 'Class'는 주요 요소들, 시스템 내의 상호작용, 시스템에서 구현해야할 클래스들을 모두 나타낸다. 각각의 Class 는 이름, 속성, 실행할 수 있는 작업 이렇게 3 개의 구획을 가진다. Class Diagram 은 시스템 설계시에 여러 class 들이 식별되고 그룹화 되어지기 때문에 이 사이의 정적 관계를 결정하거나 상속 관계등을 명확히 표현할 수 있게 한다.

E. State Diagram

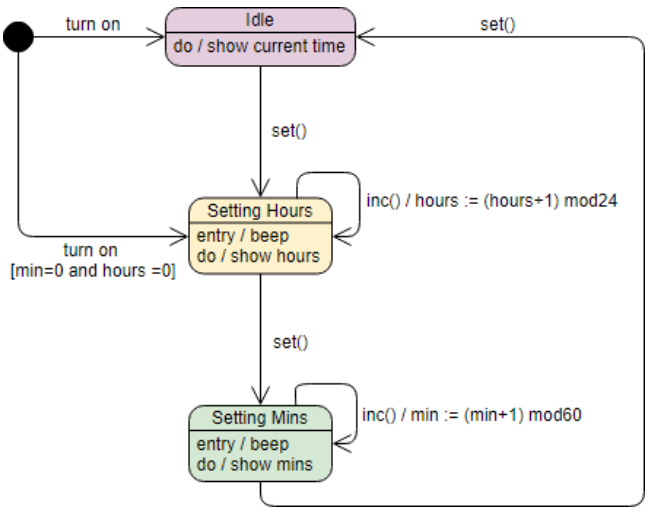

Figure 5 State Diagram Example

State Diagram 은 전체 시스템 혹은 한 객체에 대해서 인스턴스 사건(Event)에 따른 반응과 생명주기동안의 상태 변화를 모델링한 다이어그램이다. 주로 하나의 클래스 내부의 구성요소나 객체가 외부 Event 에 반응하는 동적인 상황을 기술하기 위한 것이기 때문에, Sequence Diagram 과 같이 동적인 모델로 분류된다.

State Diagram 을 통해 Event 에 대한 객체의 반응, 상태 변화를 상세하게 분석할 수 있고, 객체가 가질 수 있는 상태의 종류나 개수에 대한 정보를 얻을 수 있다. 또한 객체의 속성과 수행해야 하는 동작을 정의하고 검증할 수 있다.

F. Sequence Diagram

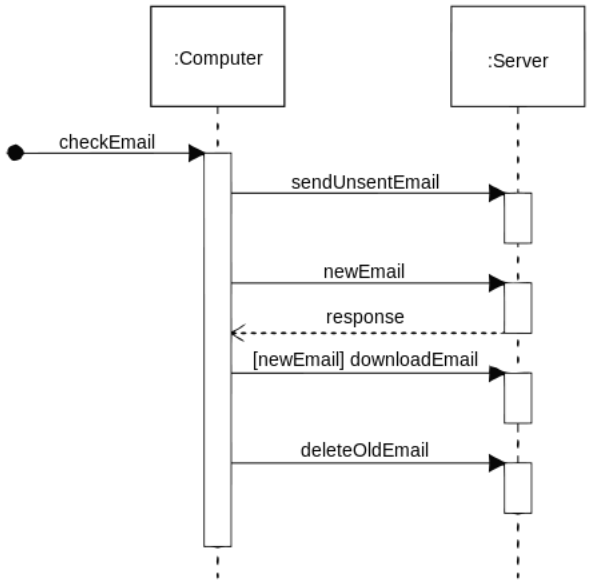

Figure 6 Sequence Diagram Example

Sequence Diagram 은 UML 의 interaction diagram 의 일종으로, 시스템내의 객체간의 상호작용을 시간 순으로 표현하는 다이어그램이다. 특히 객체 및 클래스와, 그 사이에서 기능을 수행하기 위해 교환되는 메세지들의 sequence 를 표현하기 때문에, 객체간의 관계와 속성, 동작을 명확히할 수 있다. 일반적으로 논리 측면에서 Use case 의 구현과 관련되어 많이 사용되고, Event Diagram 이라고 표현되기도 하는 동적인 측면의 시스템 모델링 과정이다.

Sequence Diagram 은 평행한 수직선으로서 동시에 살아있는 서로 다른 프로세스 혹은 객체를 나타내고, 수평 화살표로 그 사이 교환되는 메세지를 발생 순서대로 묘사한다. 이를 통해 간단한 런타임 시나리오를 만들 수도 있다.

G. Data Flow Diagram

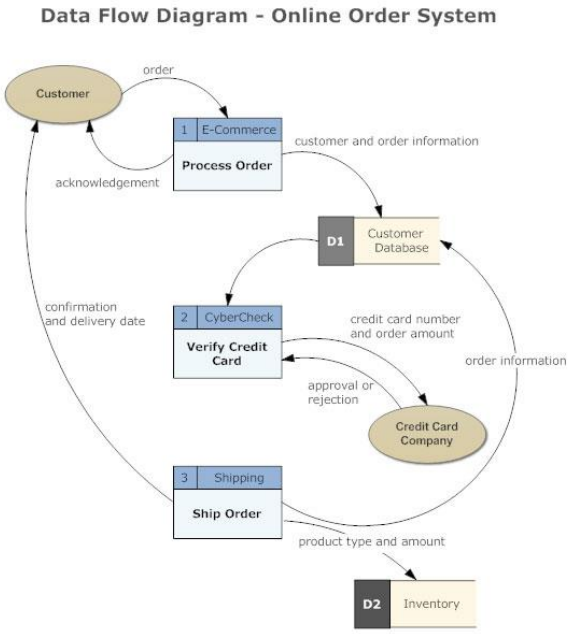

Figure 7 Data Flow Diagram Example

Data Flow Diagram (DFD) 은 입/출력 측면에서 정보 시스템의 데이터 처리 방식을 시각화하여 모델링한다. DFD 는 시스템에 어떤 종류의 정보가 입력되고 시스템으로 출력되는지, 데이터가 시스템을 통해 어떻게 진행되는지 그리고 데이터가 저장 될 위치를 보여준다. 시각화에 사용되는 기호는 표기법에 따라 모양에 차이가 존재하기도 하지만, 구성요소는 Process, Data store, Data flow, External entity 으로 이루어져있다.

DFD 의 특징은 그림 중심의 표현이고, 제어(Control)의 흐름보다 자료(Data)의 흐름에 중심을 두고 분석하기 위한 다차원적 다이어그램이라는 점이다.

H. Entity-Relationship Diagram

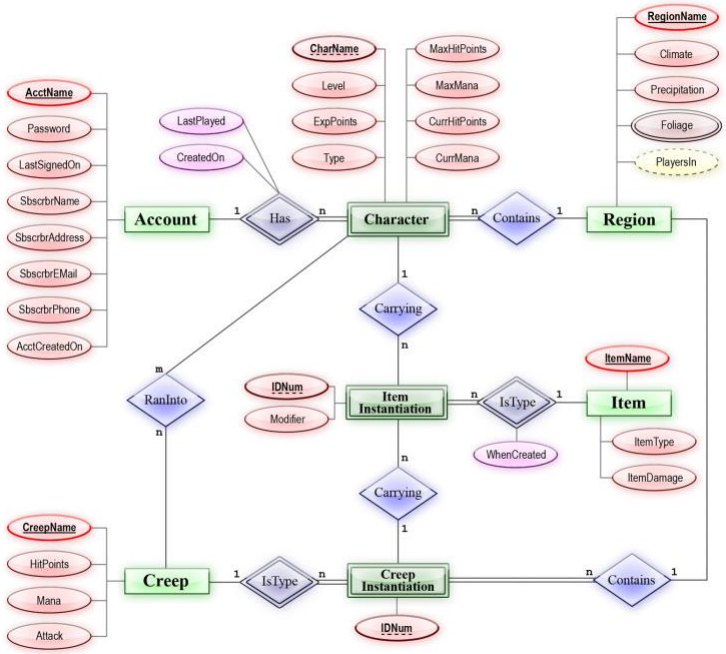

Figure 8 ER Diagram Example

Entity-Relationship Diagram (ERD) 은 데이터 베이스의 구조화된 데이터에 대한 일련의 표현으로, UML 에 속하는 다이어그램은 아니다. 데이터베이스에 저장된 데이터의 구조 및 수반되는 제약조건들을 다양한 기법으로 정의하고 모델링할 수 있다.

ERD 는 Entity 와 Relationship 으로 구성되어있다. Entity(개체) 는 현실세계에 객체로써 유무형의 정보대상으로 존재하면서 하나 이상의 속성으로 구성되어 구별될 수 있는 것을 의미하고, 같은 속성을 갖는 개체들의 집합을 개체타입이라고 한다. Relationship(관계) 는 여러 개체간에 존재하는 연관성으로써 역시 하나 이상의 속성을 가질 수 있고, 같은 관계들의 집합을 관계타입이라고 한다. 이 외에도 ERD 는 일대일 관계, 다중 관계, 제약조건 등의 풍부한 표현이 가능하고, 데이터들에 대해 단순하면서도 모호하지 않고 정확한 이해를 도울 수 있다.

2.3 Applied Tool

A. Cloud-based diagramming SW (Draw.io)

Figure 9 'draw.io' logo

본 문서는 다양한 UML diagram 을 사용해 시스템 설계 구조를 표현할 것이다. 이와 같은 diagram 작성을 위해 Cloud-based diagramming software 인 "Draw.io"를 주로 사용하였다. 이 tool 은 쉽고 편리하게 접근가능한 무료 소프트웨어이다. 또한 Web 기반이며, Cloud 에 연동가능하기 때문에 본 프로젝트와 같은 팀 작업에서 작업 상황 공유가 가능하다는 장점이 있다.

B. Android Studio

Figure 10 'Android Studio' logo

Android Studio 는 안드로이드의 공식 통합 개발 환경 (IDE)이다. 안드로이드용으로 제작된 IntelliJ 기반의 tool 이며, 안드로이드 개발의 가속화를 제공할 수 있고, 풍부한 코드 편집과 디버깅, 테스트 등 안드로이드 앱 개발에 필요한 맞춤형 도구를 제공하고 있다.

C. Amazon Web Service (AWS)

Figure 11 'AWS' logo

Amazon Web Service (이하 AWS)란, Amazon 에서 제공하는 클라우드 서비스로, 대표적인 클라우드 컴퓨팅 플랫폼이다. AWS 는 인프라 구축에 필요한 다양한 서비스를 제공하고 있고, 비교적 간편하게 서버를 구축 할 수 있다. AWS 는 웹 서비스 뿐만아니라 모바일 앱 개발을 위한 서비스도 제공하고 있고, 안정적이기 때문에, 안드로이드 어플리케이션의 백엔드 서버로 활용하기 좋다.

2.4 Project Scope

'Shoose' (Choose your Shoes) 시스템은 신발 추천 AI 시스템으로, 사용자의 개인 선호도에 맞게 신발을 추천해주고, 최저가 온라인 쇼핑사이트로 연결해주는 안드로이드 어플리케이션 시스템이다. 'Shoose' 시스템은 크게 7 가지의 서브 시스템으로 구성된다.

7 개의 서브시스템들에는 사용자 관리와 개인 선호도 관리를 위한 User management System, Survey System, Profile customizing System 과, 신발 추천을 위한 Search System, Recommend System, Tournament System, 온라인 쇼핑몰로 연결을 해주기 위한 Link System 이 있다.

- User Management System

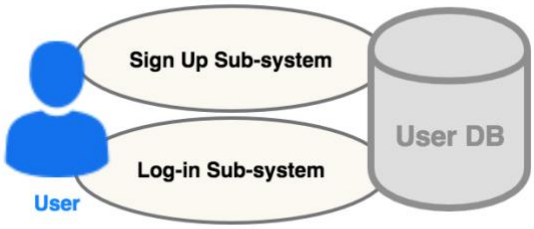

Figure 12 User Management System Architecture

User management System은 user를 관리하기 위한 시스템이다. User management System은 다시 2개의 하위 시스템으로 구성되어진다. 하위 시스템에는 회원가입 기능을 제공하는 Sign Up Sub-system과 로그인 기능을 제공하는 Log-in Sub-system이 있다.

- Survey System

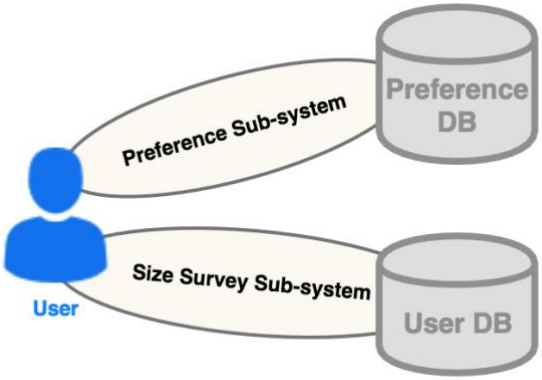

Figure 13 Survey System Architecture

Survey System은 user의 개개인의 선호도를 수집, 관리하기 위한 시스템이다. 신규 유저가 회원가입 이후에 의무적으로 수행하게 되는 시스템으로, 다시 2개의 하위시스템으로 구성된다. 하위시스템에는 user의 preference를 조사하는 Preference Survey Sub-system과 user의 신발 size를 조사하는 Size Survey Sub-system이 있다.

- Profile Customizing System

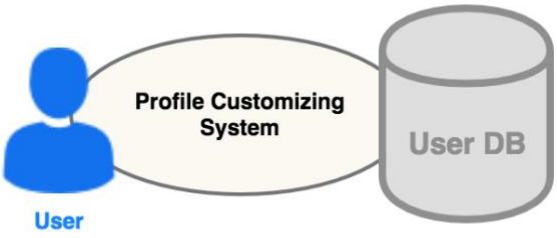

Figure 14 Profile Customizing System Architecture

Profile Customizing System은 user의 profile의 변경 요청을 수행하기 위한 시스템이다. Profile Customizing System은 user의 요청이 있을 때마다 초기 의무 survey를 통해 구축된 user profile을 자유롭게 수정할 수 있도록 하는 System이다.

- Search System

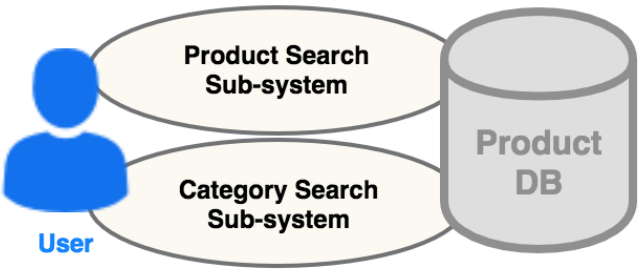

Figure 15 Search System Architecture

Search System은 user가 제품을 검색할 수 있게 하는 시스템이다. Search System은 사용자가 원하는 검색 방법에 따라 2개의 하위시스템으로 구분된다. 하나는 제품명을 검색하여 제품을 찾는 Product Search Sub-system이고, 다른 하나는 카테고리를 통해 제품을 찾는 Category Search Sub-system이다.

- Recommend System

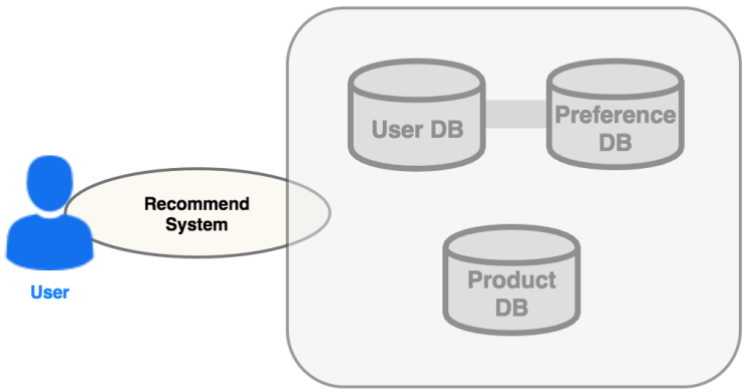

Figure 16 Recommend System Architecture

Recommend System은 user에게 제품을 추천해주는 시스템이다. Recommend System는 Survey system을 통해 추출한 user의 Preference DB와 User DB를 이용하여 Product DB으로부터 사용자가 선호할 만한 신발 제품들을 선정하여 추천하는 System이다.

- Tournament System

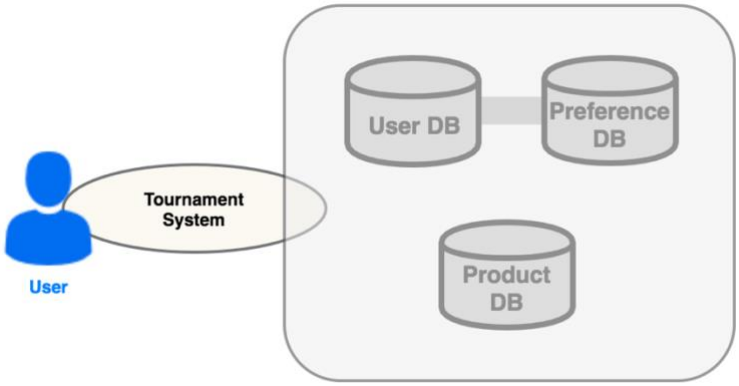

Figure 17 Tournament System Architecture

Tournament System은 user가 원하는 특정 상황에 대한 제품 추천을 위한 시스템이다. user가 Tournament를 요청할 때 실행되며, user의 Preference DB와 User DB를 토대로 적절한 상품 16개를 Product DB에서 추출하여 user에게 Tournament형식으로 선택할 수 있게 하는

시스템이다. 최종 승리 제품은 user의 Preference DB에 반영된다.

- Link System

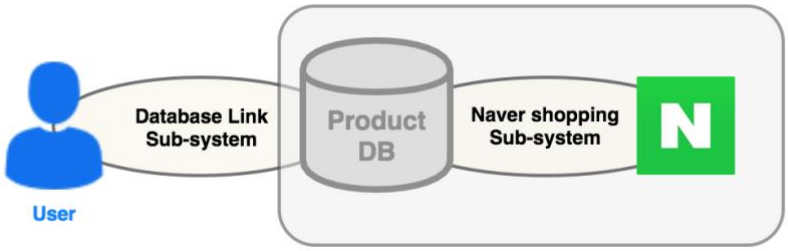

Figure 18 Link System Architecture

Link System은 user가 마음에 드는 상품을 구매하고자 할때 온라인 쇼핑몰로 연결시켜주기 위한 시스템으로, 2개의 하위시스템으로 구성된다. 하위시스템에는 Naver 쇼핑 API를 통해 온라인 쇼핑몰 링크를 찾는 Shopping Sub-system과 저장된 link를 가져오는 Database Link Sub-system이 있다.

3. System Architecture

3.1 Objectives

System Architecture에서는 우리 팀에서 개발하고자 하는 시스템에 대해 전반적으로 서술한다. 시스템의 전체적이 구조를 설명하고, 시스템을 Block diagram으로 나타낸다. 그리고 각각의 관계와 실제로는 어떻게 사용되는 지를 Package diagram과 Deployment diagram으로 사용하여 설명한다. 각 시스템에 대한 modular decomposition은 4~10장, 총 7개의 장에 걸쳐 설명한다.

3.2 System Organization

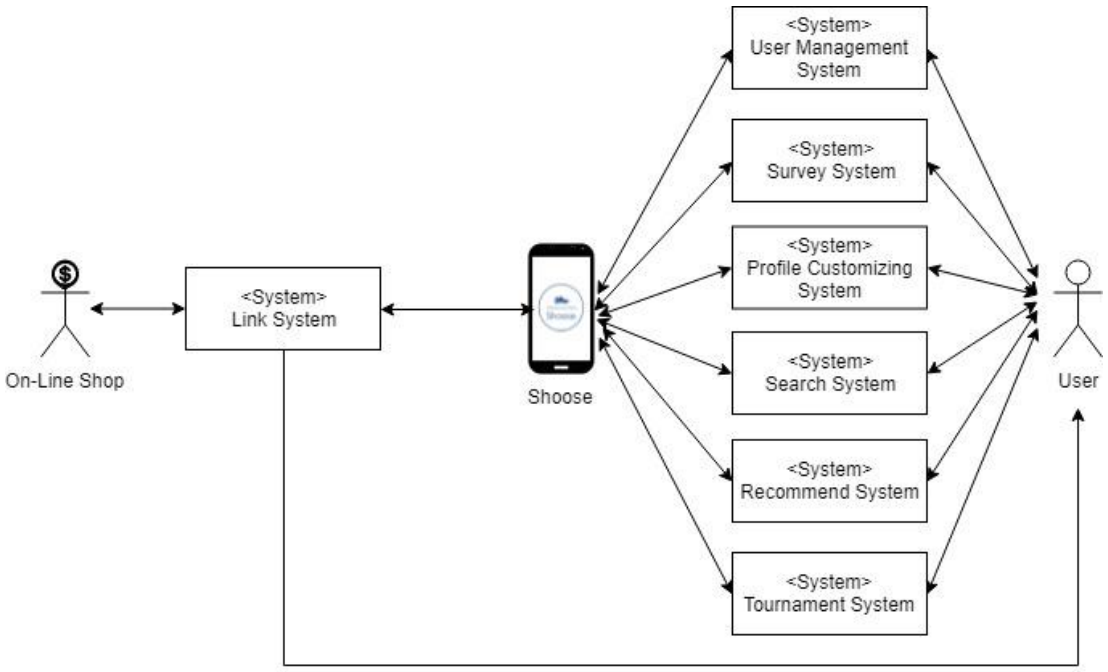

Figure 19 System Organization Block Diagram

"Shoose" App은 Client-Server Model을 사용하여 구현된다. Server는 Client에게 Shoose의 특색 있는 서비스를 제공하기 위해, 총 7개의 시스템을 가지고 있다. 각각의 시스템은 MySQL을 이용한 데이터베이스에 정보들을 저장하고 변경하며, user의 요청이나 시스템의 특성에 따라 데이터베이스에서 정보를 불러온다. 위의 diagram은 Shoose를 구성하는 시스템들과 그 시스템들이 Shoose, User, On-Line Shop과 어떠한 관계를 지니고 있는 것인지 확인하기 위하여 구성된 것이다. 따라서 System에 대한 자세한 설명, 예를 들어 데이

터베이스나 시스템의 functions에 대한 내용들은 생략하였다.

7개의 시스템은 각각 서버를 이용하여 user의 request에 따라 데이터베이스를 이용하여 적절한 response를 한다. 그에 대한 user의 반응을 다시 server로 전송하여 데이터베이스를 새롭게 갱신한다. User Management System, Survey System, Profile Customizing System, Search System, Recommend System, Tournament System은 Shoose와 User사이에서 상호작용하는 시스템이며, Link System은 On-Line Shop, Shoose, User가 서로 연동하는 시스템이다.

A. User Management System

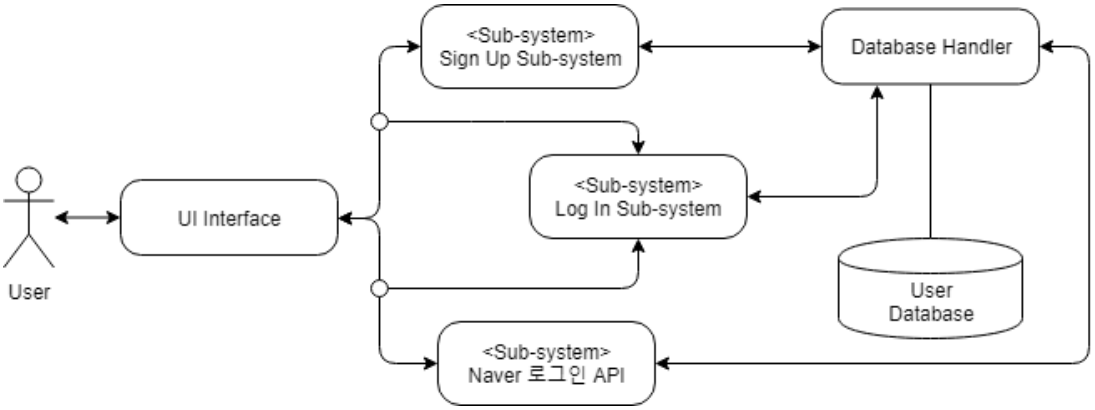

Figure 20 Sign Up/Log-in Sub-system

User management System의 경우, user의 이용과 관련된 시스템이다. User의 회원가입을 관리하는 Sign Up Sub-system과 로그인 기능을 제공하는 Log-in Sub-system으로, 총 2개의 하위 시스템으로 구성되어 있다.

B. Survey System

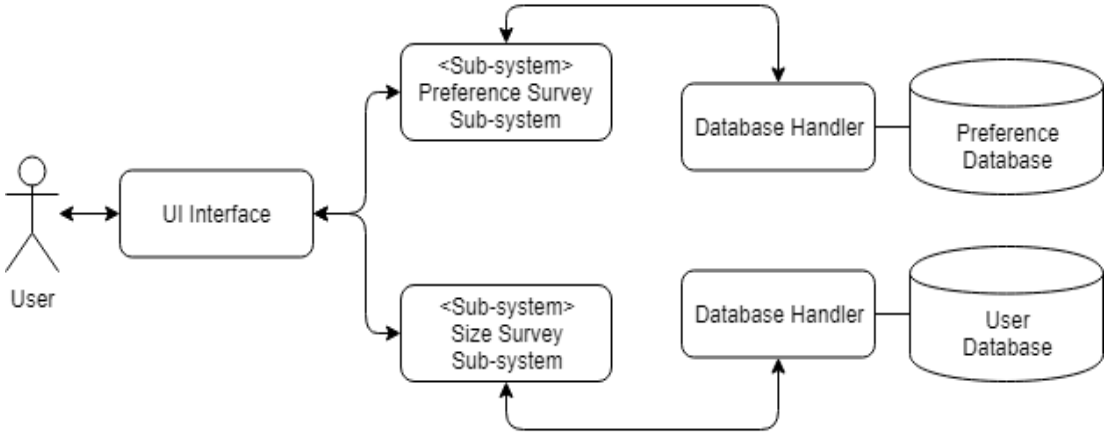

Figure 21 Survey System Block Diagram

Survey System의 경우에도 user와 관련된 시스템이다. 신규유저에게 회원가입 후, 가장 먼저 의무적으로 하도록 하는 시스템이다. User의 preference를 조사하는 Preference Survey Sub-system과 user의 size를 조사하는 Size Survey Sub-system으로, 총 2개의 하위 시스템으로 구성된 Survey System이다.

C. Profile Customizing System

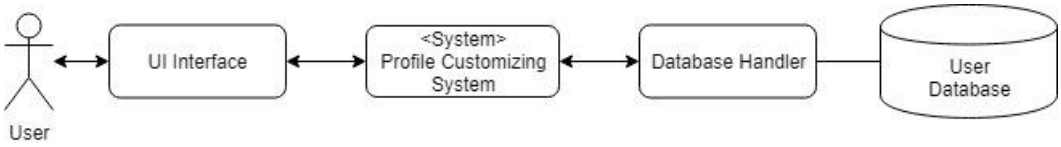

Figure 22 Profile Customizing Block Diagram

신규 User가 초기 앱 구동 시, 의무적으로 받은 초기 조사를 통하여 갖춰진 user의 profile을 user의 요청이 있을 때마다 자유롭게 수정할 수 있도록 하는 System이다.

D. Search System

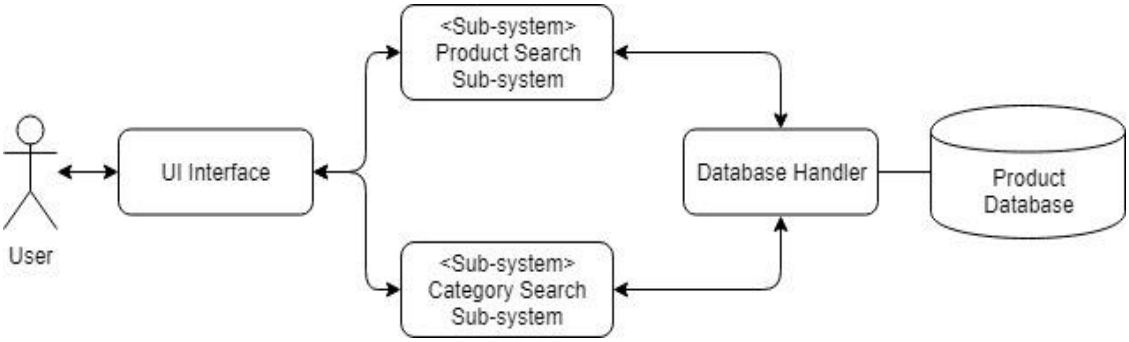

Figure 23 Search System Block Diagram

Search System은 User가 'Shoose'에게 request를 하면 실행되는 시스템이다. 제품명을 이용하여, Product를 찾는 Product Search Sub-system과 category를 통하여, Product를 찾는 Category Search Sub-system으로, 총 2개의 Sub-system으로 구성되어 있다. User의 request에 따라, Product나 Category Search Sub-system을 통하여, user에게 response를 한다.

E. Recommend System

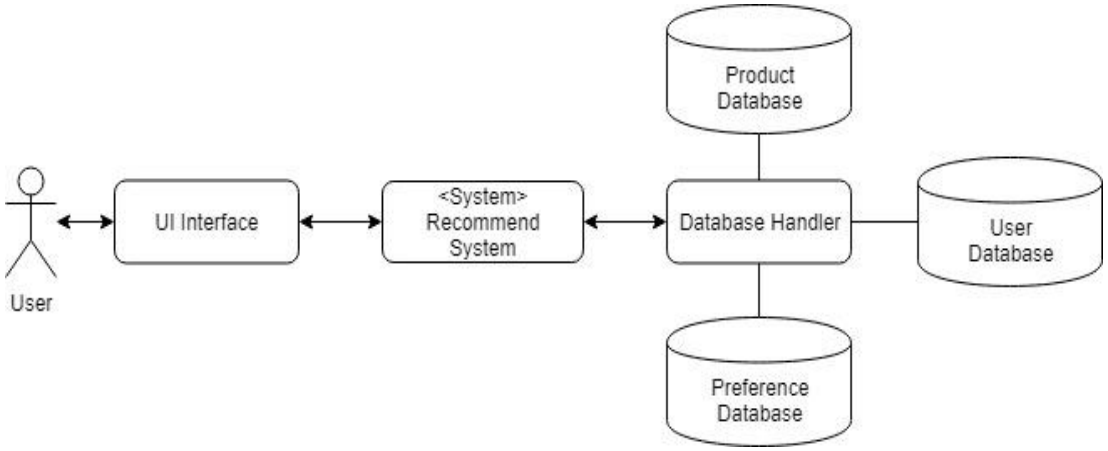

Figure 24 Recommend System Block Diagram

Recommend System은 user에게서 survey system을 통하여 추출해 낸, Preference DB와, User DB의 size를 이용하여 Product DB에서 적절한 product를 user에게 추천하는 system 이다.

F. Tournament System

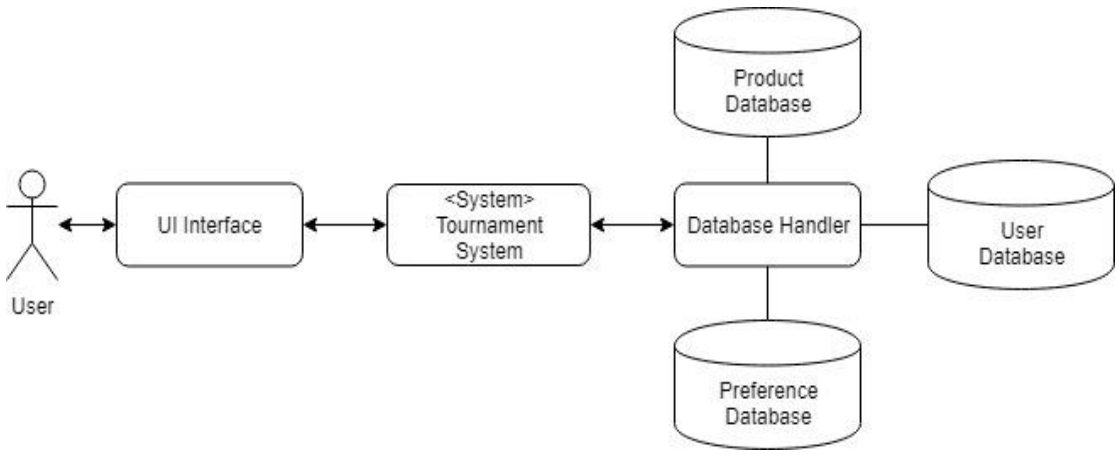

Figure 25 Tournament System Block Diagram

Tournament system은 user의 request가 있을 때, 실행되는 system이다. User가 Tournament를 시행하기를 원한다면 'Shoose'는 Tournament System을 구동한다. User DB의 size 항목과, Preference DB를 통하여 적절한 제품 16개를 Product항목에서 추출해낸다. 16개의 제품으로 토너먼트 경기를 한 후에, 최종 승리한 제품을 user의 Preference DB에 추가한다.

G. Link System

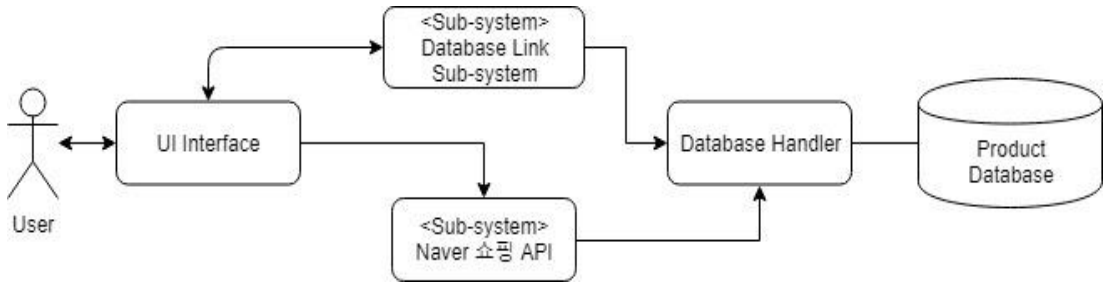

Figure 26 Link System Block Diagram

Link System은 'Shoose'가 user에게 search, recommend하거나 tournament를 통하여 결정된 product들을 user가 구매하기를 원할 때, 해당 사이트로 넘어가게 해주는 system이다. Naver 쇼핑 API를 이용하는 Sub-system과 Database에서 내장된 Link를 가져오는 Database Link Sub-system, 총 2개의 하위 시스템으로 구성된 시스템이다. Naver 쇼핑 API를 통하여 product를 판매하는 쇼핑몰의 링크를 Product DB에 내장하고, 주기적으로 링크가 유효한 지를 확인한다. User가 해당 링크로 넘어가기를 원한다면 Database Link

Sub-system을 통하여, user에게 링크를 제공한다.

3.3 Package Diagram

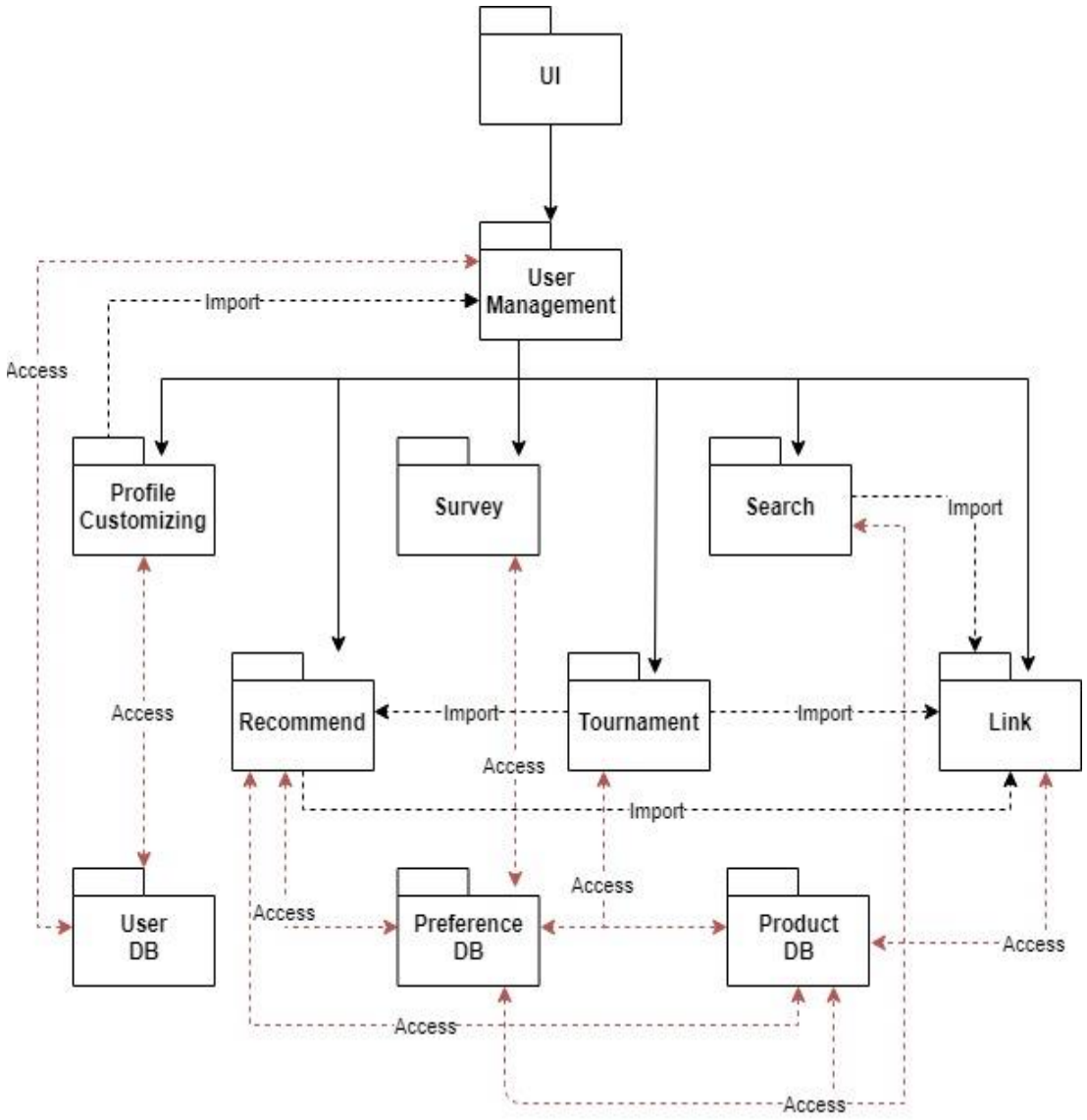

Figure 27 Package Diagram

Package들로만 간단하게 표현하였다. 각 Package 내의 Class, Component에 대해서는 4~10장에서 설명한다.

3.4 Deployment Diagram

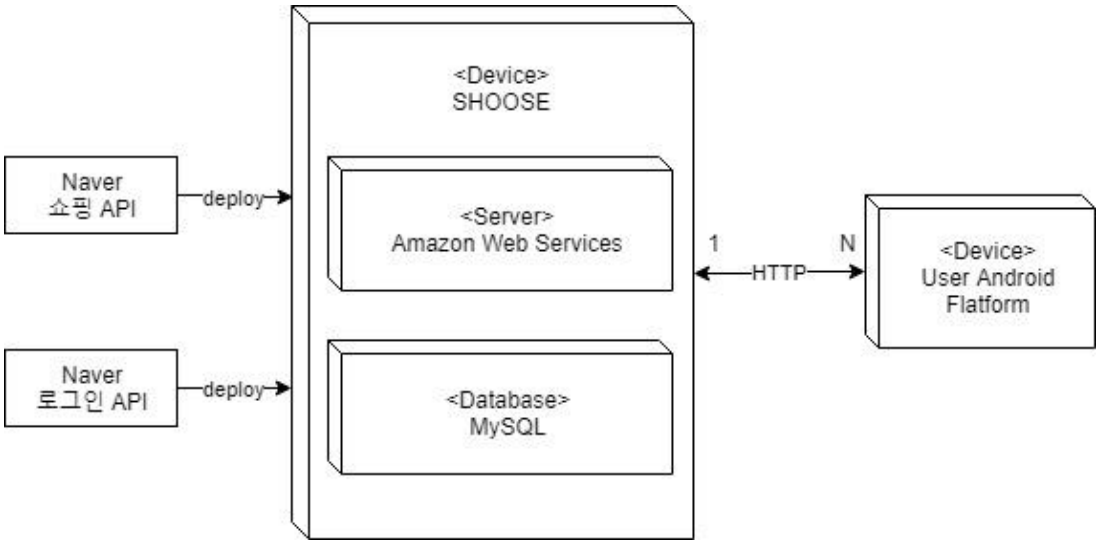

Figure 28 Deployment Diagram

4. User Management System

4.1 Objectives

실제 사용자가 사용하며 회원 가입과 로그인 기능을 통해 시스템을 이용하는 도중에 발생하는 데이터를 처리하는 사용자 관리 시스템의 설계를 설명한다. Class diagram, Sequence diagram, State diagram을 통해 User Management System의 구조를 표현하고 설명한다.

4.2 Class Diagram

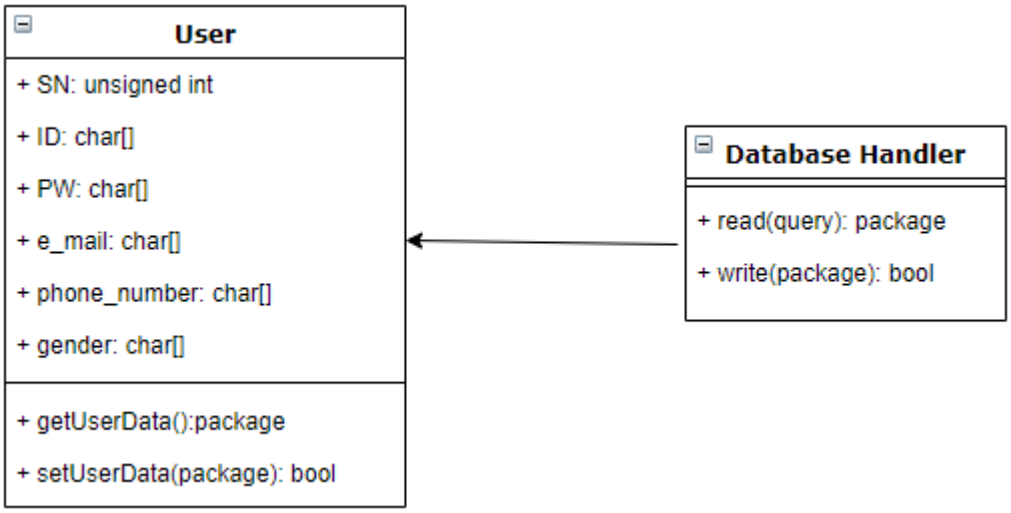

Figure 29 User Management System Class Diagram

A. Database Handler

A.1 Attributes

해당 사항 없음

A.2 Methods

+ package read(query): User DB에서 query에 해당하는 package(data)를 읽어온다.

+ bool write(package): User DB에 package(data)를 저장한다. 성공하면 true를, 실패하거나 오류가 발생하면 false를 return한다.

B. User

B.1 Attributes

+ SN: User 고유번호

+ ID: User ID

+ PW: User 비밀번호

+ e-mail: User e-mail

+ phone_number: User 전화번호

+ gender: User 성별

B.2 Methods

+ package getUserData(): DB로부터 User package(data)를 받는다.

+ bool setUserData(package): DB에 User package(data)를 보낸다. 성공하면 true를, 실패하거나 오류가 발생하면 false를 리턴한다.

4.3 Sequence Diagram

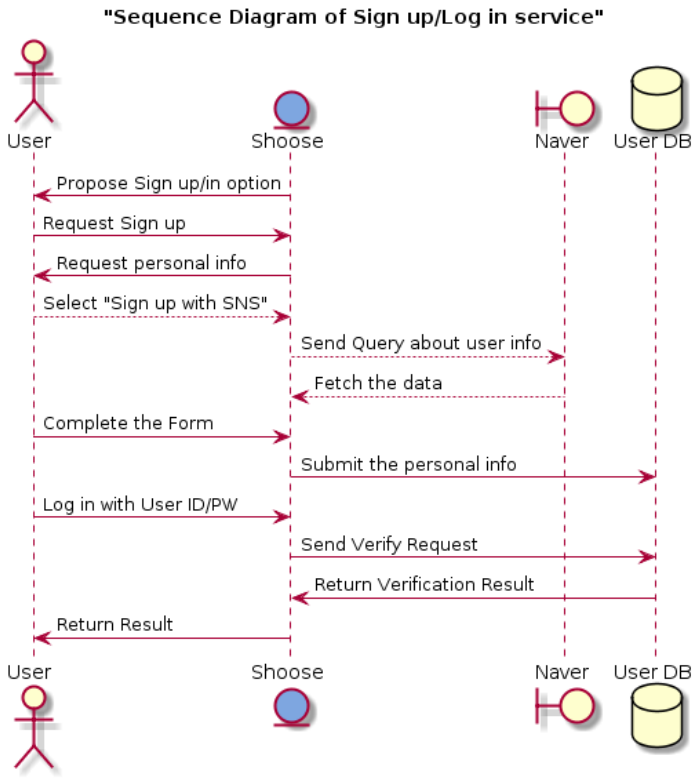

Figure 30 User Management System Sequential Diagram

4.4 State Diagram

A. Sign up

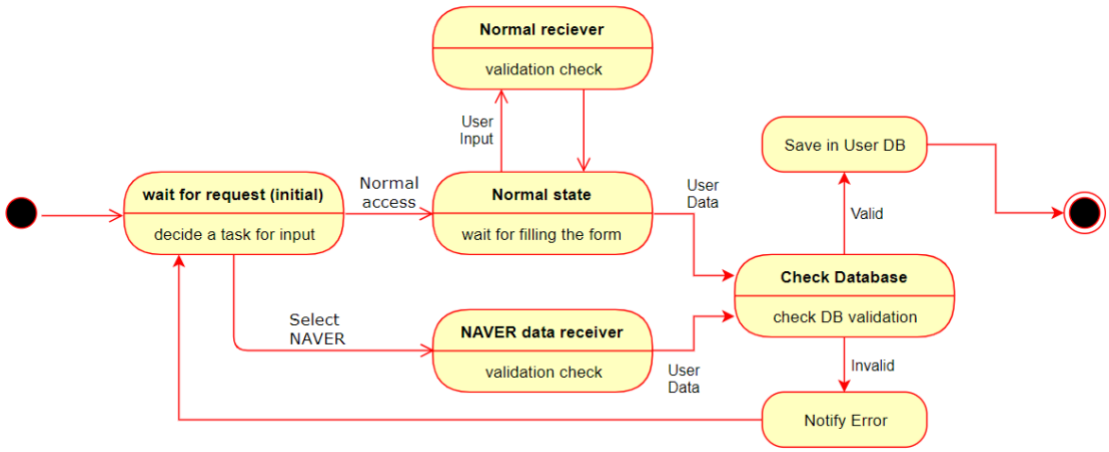

Figure 31 User Management System State Diagram 1

B. Log in

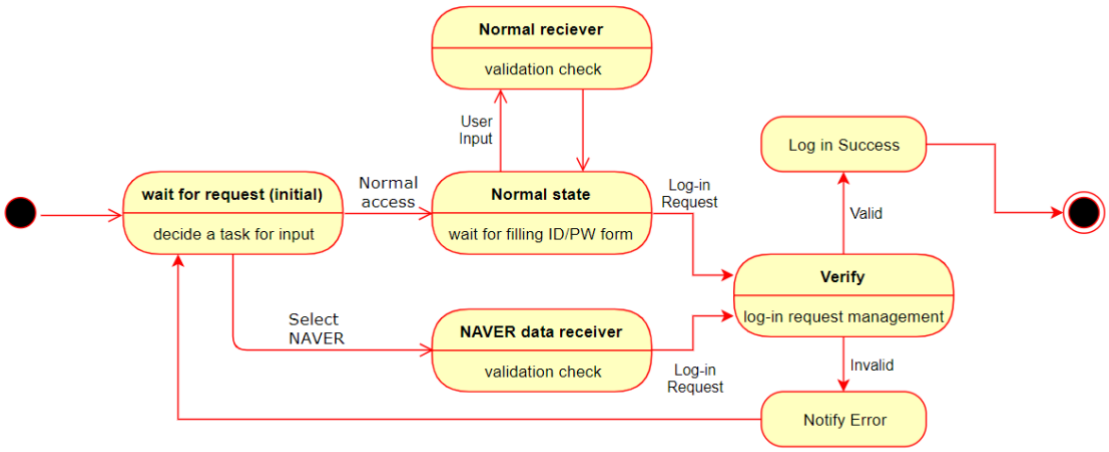

Figure 32 User Management System State Diagram 2

5. Survey System

5.1 Objectives

추천시스템이라는 것은 사용자의 선호도에 따른 맞춤형 서비스이기 때문에, 사용자가 처음 서비스를 이용할 때 필요한 사용자의 선호도를 조사(Survey)를 진행한다. 사용자가 입력한 데이터를 처리하고 만약 수정 사항이 발생하였을 때, 이를 처리하는 Survey system의 설계를 설명한다. Class diagram, Sequence diagram 과 State diagram을 통하여 Survey System의 구조를 표현하고 설명한다.

5.2 Class Diagram

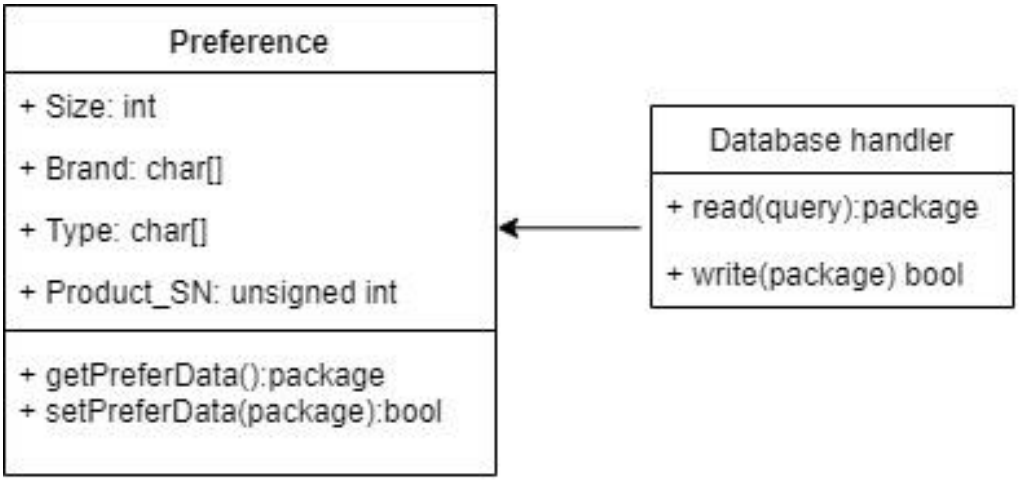

Figure 33 Survey System Class Diagram

A. Database Handler

A.1 Attributes

해당 사항 없음

A.2 Methods

+package read(query): Preference DB에서 query에 해당하는 package(data)를 읽어온다.

+bool write(package): Preference DB에 package(data)를 저장한다.

B. Preference

B.1 Attribute

+Size: 선호하는 신발 size 값

+Brand: 선호하는 Brand명

+Type: 선호하는 type

+Product_SN: 선호하는 제품의 고유번호 값

B.2 Methods

+package getdata(): DB로부터 data를 받는다.

+void sendData(query): DB로 data를 보낸다.

5.3 Sequence Diagram

A. Survey User's Preferences

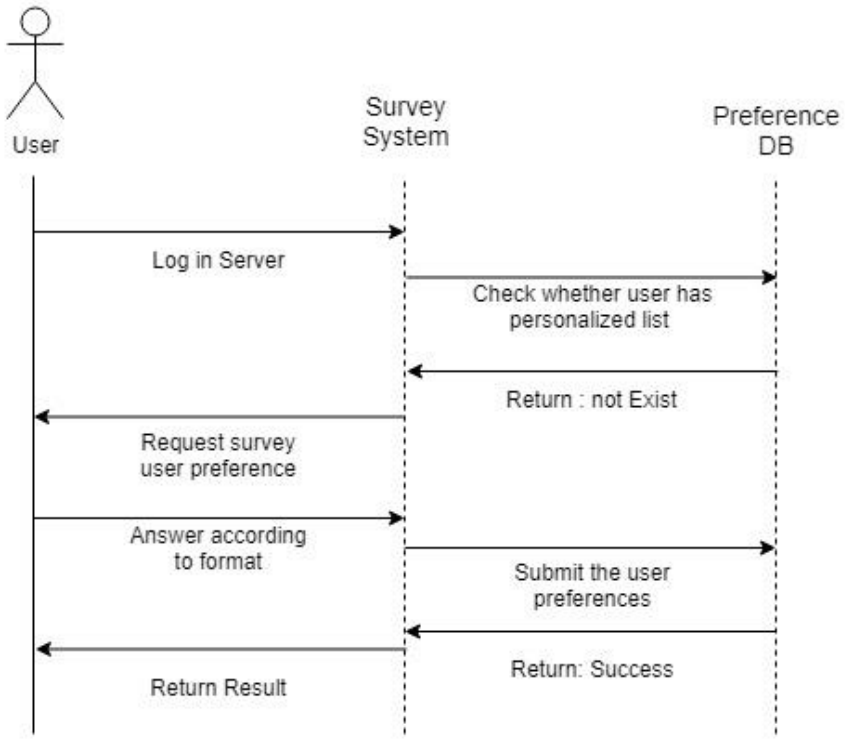

Figure 34 Survey User's Preferences Sequential Diagram

B. Edit Personalized list(Preferences)

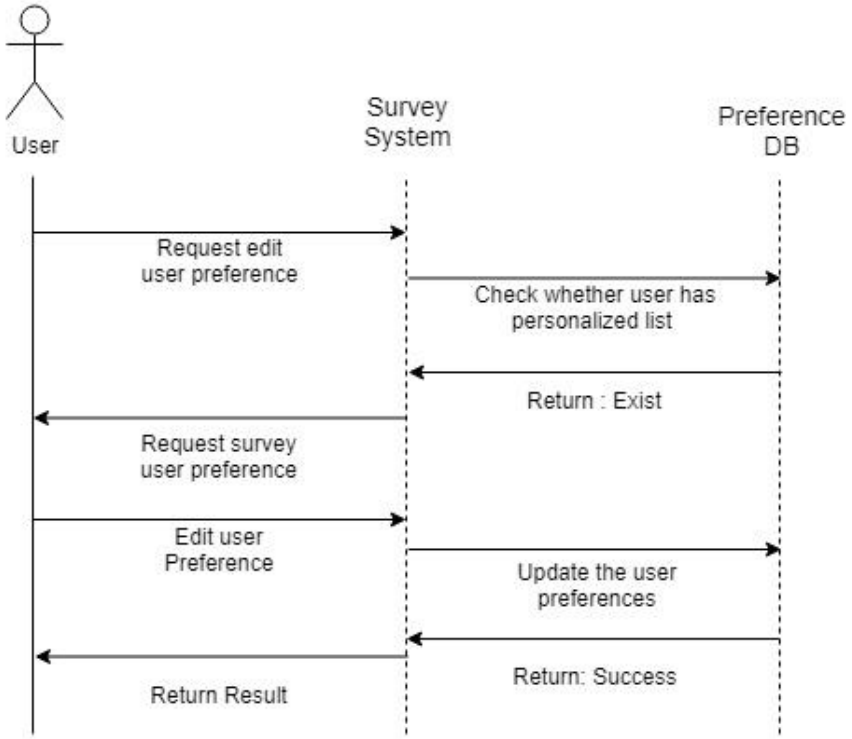

Figure 35 Edit User's Personalized list Sequential Diagram

C. Survey/Edit User's Preferences (Personalized list)

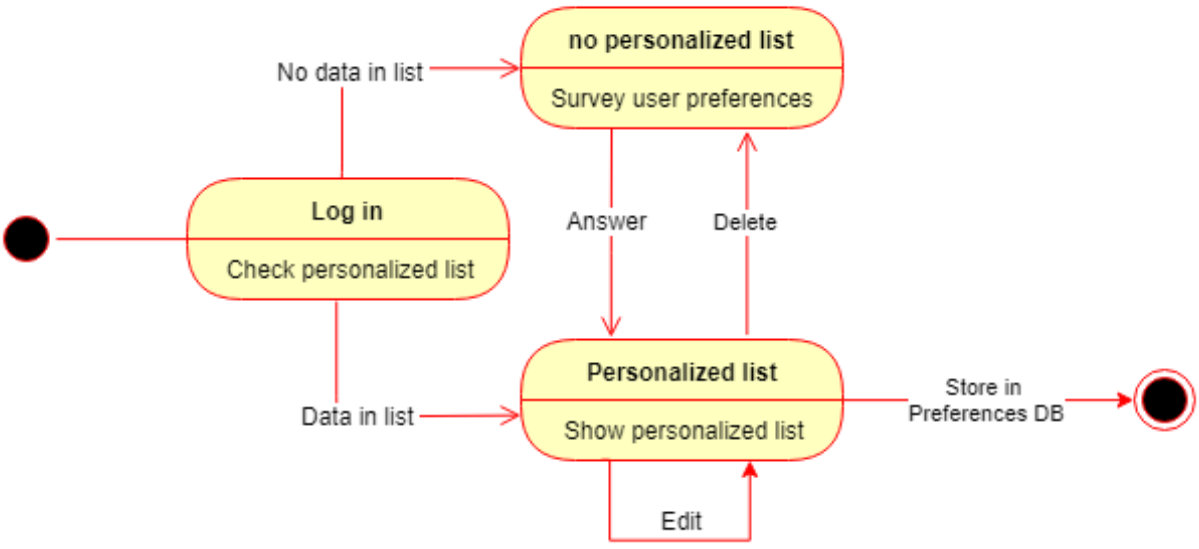

Figure 36 Survey System State Diagram

6. Profile Customizing System

6.1 Objectives

사용자가 My profile 탭에서 사용자 정보를 커스터마이징(Customizing)하기 위해, 필요한 데이터베이스에 접근하여 해당 정보를 출력하며 필요한 경우, 수정을 하는 시스템의 설계를 설명한다. Class diagram, Sequence diagram, State diagram을 통해 Profile Customizing System의 구조와 사용자와의 상호작용을 표현 및 설명한다.

6.2 Class Diagram

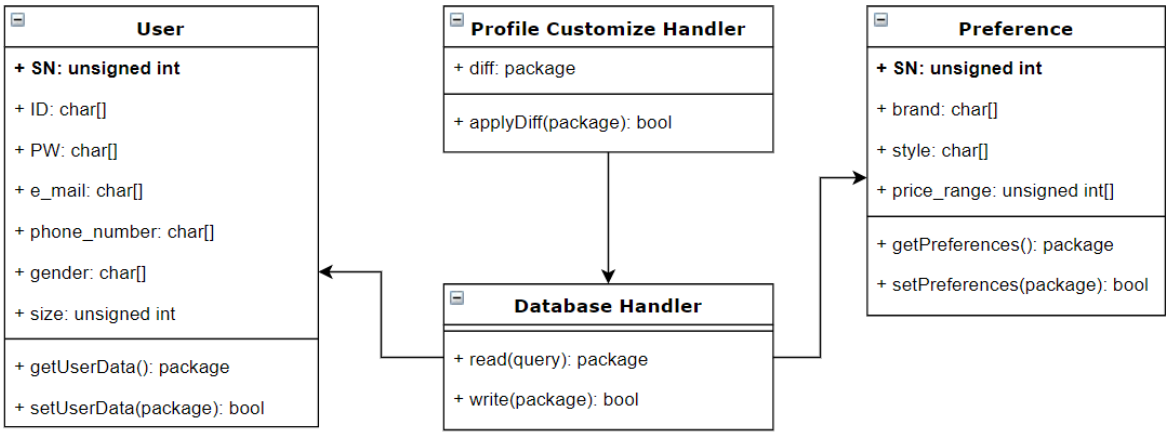

Figure 37 Profile Customizing System Class Diagram

A. Profile Customize Handler

A.1 Attributes

+ diff: Shoose app의 My profile 탭에서 사용자가 변경한 profile 구성요소들

A.2 Methods

+ bool applyDiff(package): Database Handler의 write()를 통해 User/Preferences DB에 변경사항을 반영한다.

B. Database Handler

B.1 Attributes

해당 사항 없음

B.2 Methods

+ package read(query): User/Preferences DB에서 query에 해당하는 package(data)를 읽어온다.

+ bool write(package): User/Preferences DB에 package(data)를 저장한다. 성공하면 true를, 실패하거나 오류가 발생하면 false를 return한다.

C. User

C.1 Attributes

+ SN: User 고유번호

+ ID: User ID

+ PW: User 비밀번호

+ e-mail: User e-mail

+ phone_number: User 전화번호

+ gender: User 성별

C.2 Methods

+ package getUserData(): DB로부터 User package(data)를 받는다.

+ bool setUserData(package): DB에 User package(data)를 보낸다. 성공하면 true를, 실패하거나 오류가 발생하면 false를 리턴한다.

D. Preference

D.1 Attributes

+ SN: User 고유번호

+ brand: User가 선호하는 브랜드

+ style: User가 선호하는 스타일

+ price_range: User가 선호하는 가격대

D.2 Methods

+ package getPreferences(): DB로부터 Preferences package(data)를 받는다.

+ bool setPreferences(package): DB에 Preferences package(data)를 보낸다. 성공하면 true를, 실패하거나 오류가 발생하면 false를 리턴한다.

6.3 Sequence Diagram

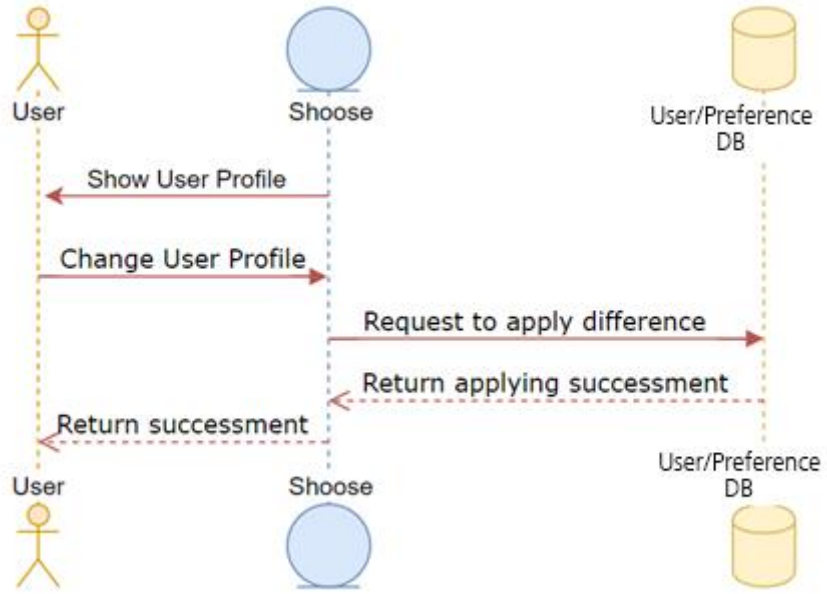

Figure 38 Profile Customizing System Sequence Diagram

6.4 State Diagram

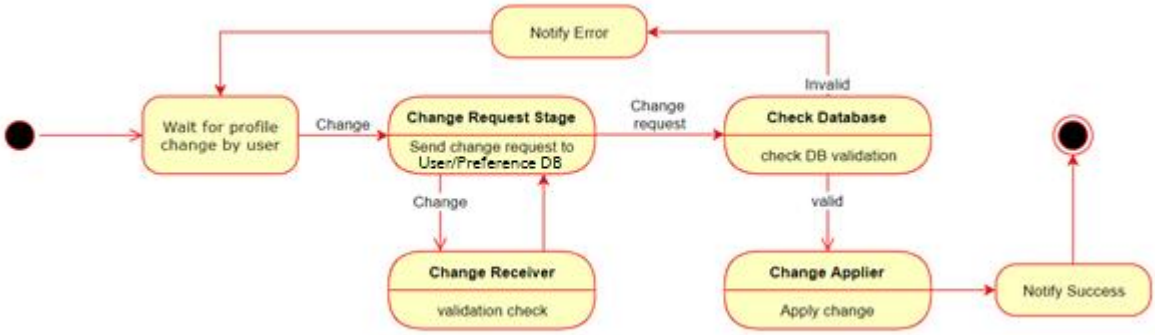

Figure 39 Profile Customizing System State Diagram

7. Search System

7.1 Objectives

User가 Product를 검색할 때, 구동되는 시스템의 설계를 설명한다. 검색을 할 때, 어떠한 검색 필터를 이용하는 지에 따라, 그 결과값들이 상이하게 나올 수 있다. 따라서 그것을 case로 설정하여 Class Diagram, Sequence Diagram, State Diagram으로, Search System의 구조와 'Shoose'와의 상호 작용을 표현하고 설명한다. 검색 필터는 크게, Product Search와 Category Search로 나뉜다.

7.2 Class Diagram

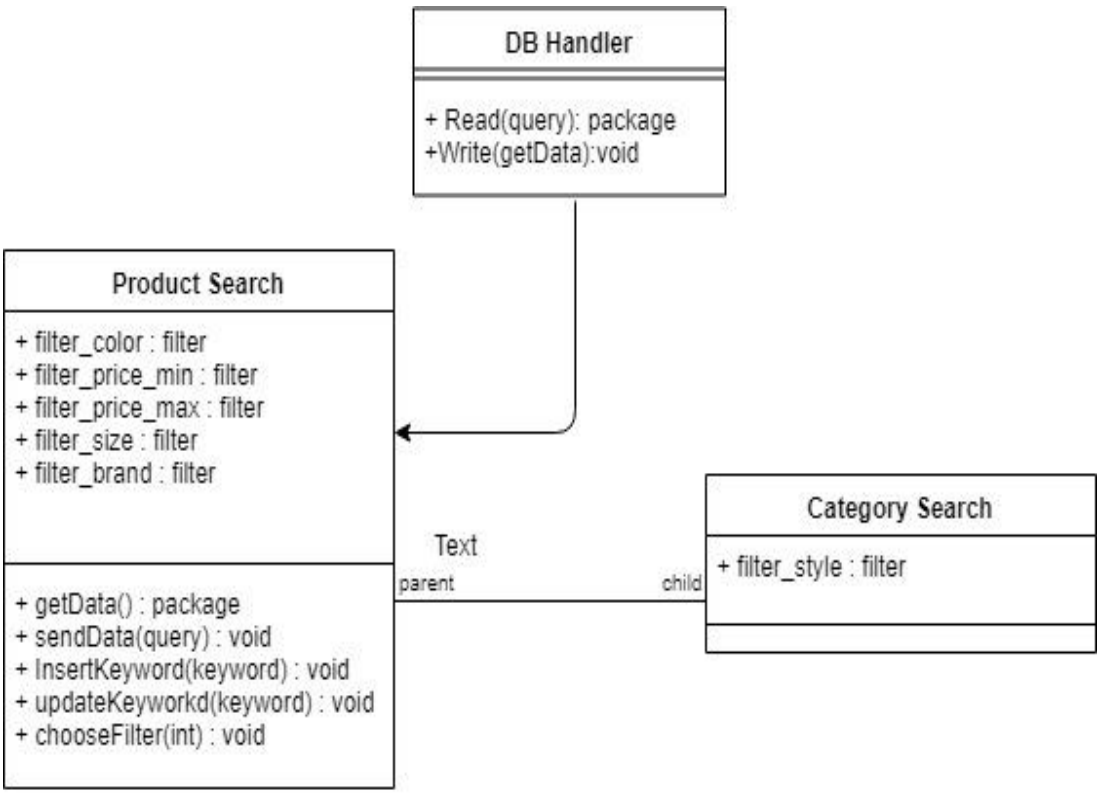

Figure 40 Search System Class Diagram

A. DB Handler

A.1 Attributes

(해당 사항 없음)

A.2 Methods

+ Read(query) : package → 해당되는 DB에서 원하는 data를 읽어온다

+ Write(getData) : void → 해당되는 DB에 data를 저장한다.

B. Product Search

B.1 Attributes

+ filter_color : filter → 색 정보 값

+ filter_price_min : filter → 가격 최솟값 정보

+ filter_price_max : filter → 가격 최댓값 정보

+ filter_size : filter → user의 사이즈 정보

+ filter_brand : filter → brand 정보

B.2 Methods

+ getData() : package → DB로부터 data를 받는다

+ sendData(query) : void → DB로부터 data를 보낸다.

+ insertKeyworkd(keyword) : void → keyword를 입력 받는다.

+ updateKeyworkd(keyword) : void → keyword를 수정한다.

+ chooseFilter(int) : void → filter를 선택한다.

C. Category Search

C.1 Attributes

(부모 class로부터 상속받음)

+ filter_style : filter → style 정보

C.2 Methods

(부모 class로부터 상속받음)

7.3 Sequence Diagram

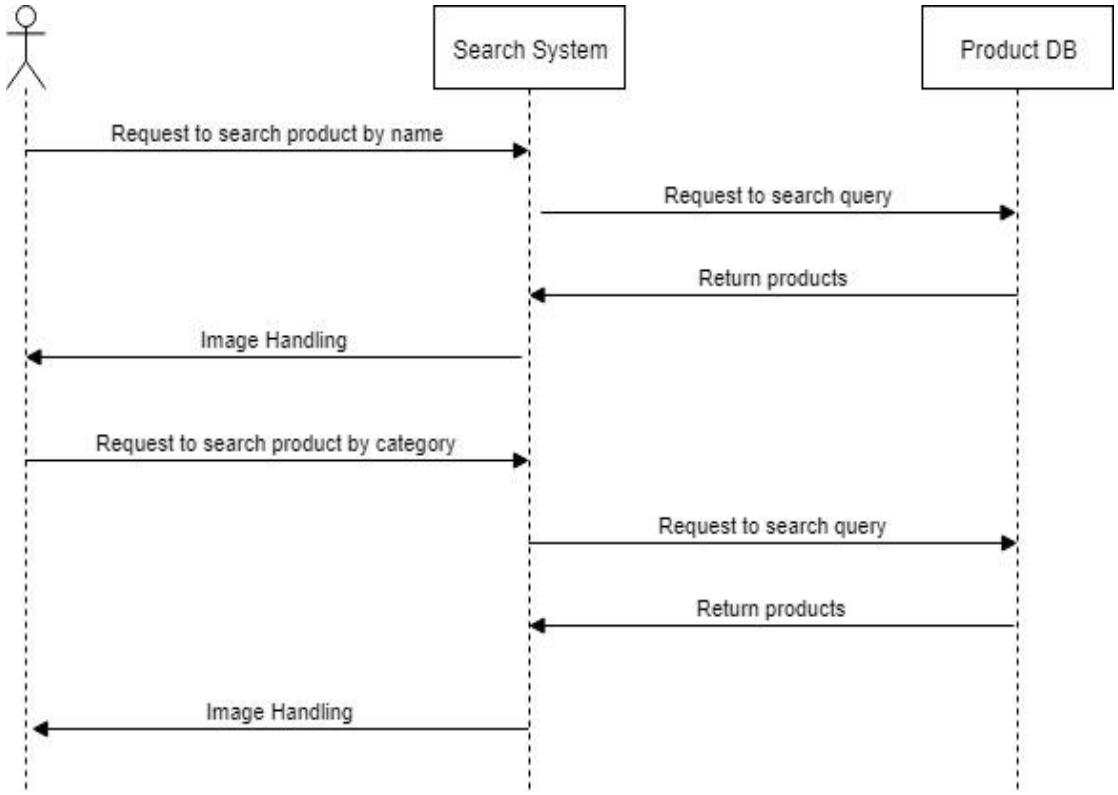

Figure 41 Search System Sequence Diagram

7.4 State Diagram

A. Product Search

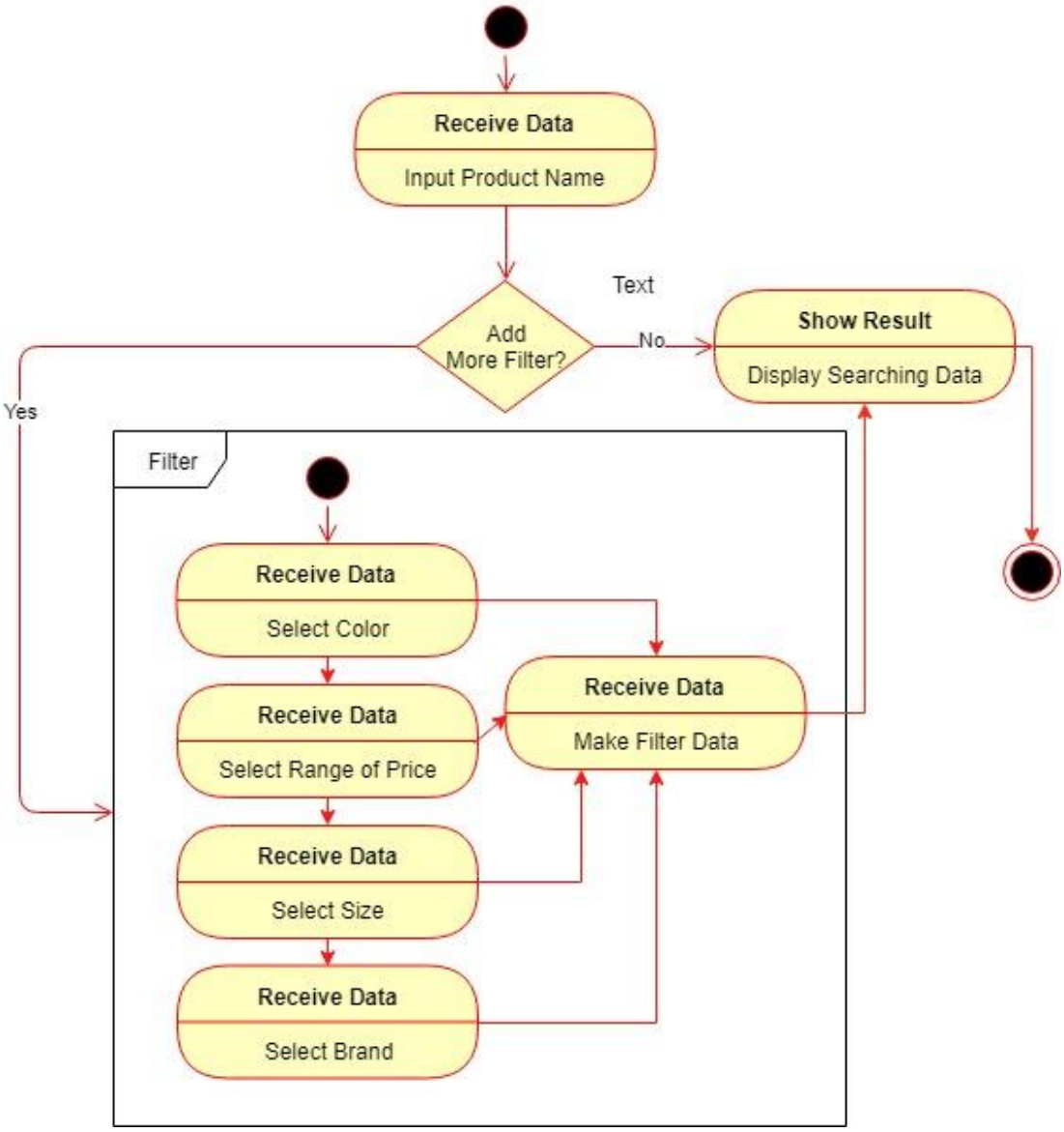

Figure 42 Product Search State Diagram

B. Category Search

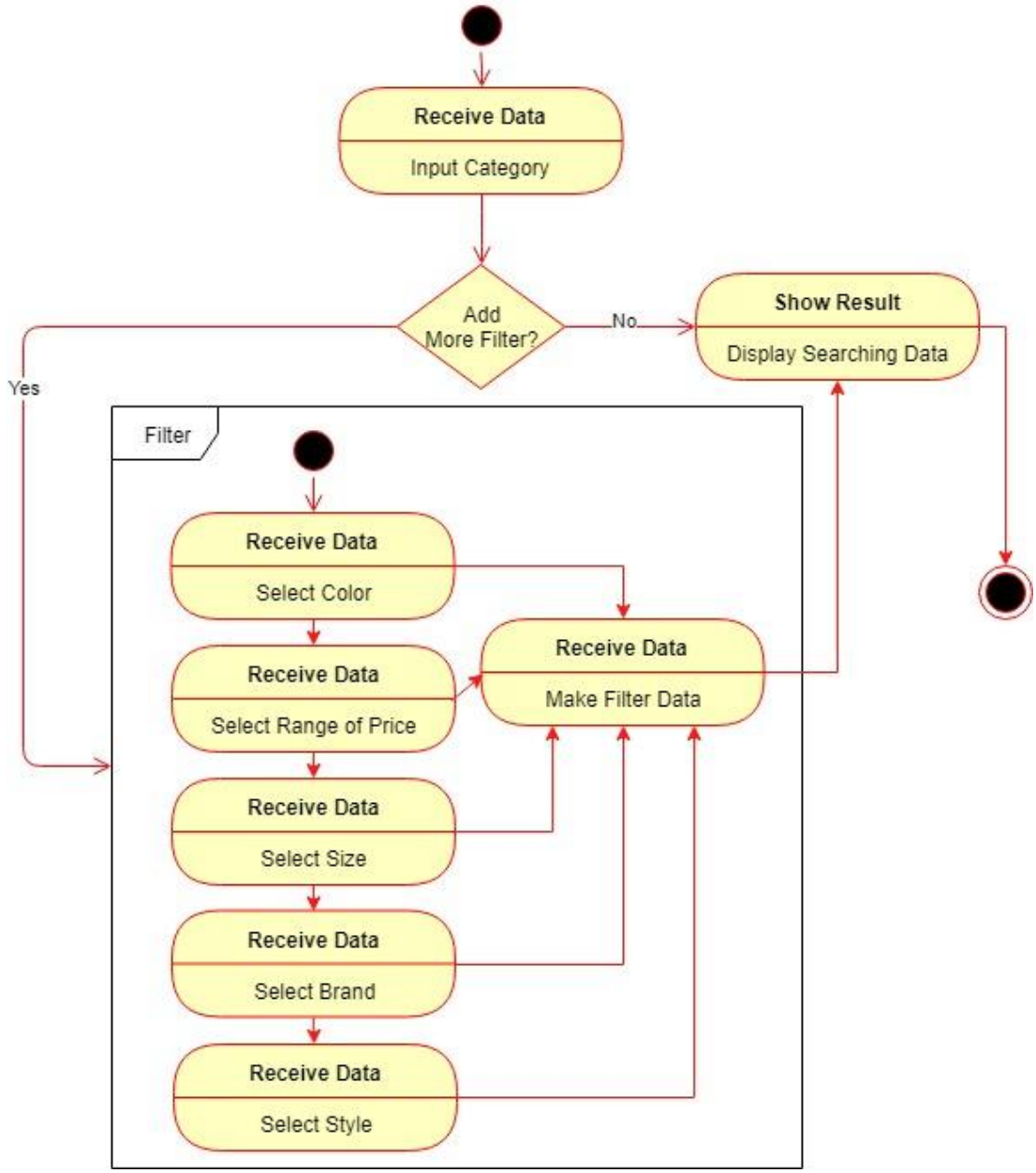

Figure 43 Category Search State Diagram

7.5 Search Filter

User가 원하는 제품을 수월하게 찾기 위해 'SHOOSE'가 제공하는 서비스이다. 우선적으로 user는 제품을 찾을 때, 제품명으로 찾거나 신발 종류, 카테고리에 따라 찾을 수 있다. 또한 'SHOOSE'는 보다 상세한 필터들을 제공한다. 제품명으로 찾을 때에는 '색깔', '가격 범위', '사이즈', '브랜드', 4가지 세부 필터를 추가 제공한다. 카테고리에 따라 제품을 찾을 때에는 앞의 4가지 세부 필터를 기본으로 제공하고, 추가적으로 '스타일' 세부 필터를 제공한다. 시스템은 user가 원하는 필터 정보에 따라 DB에서 해당 정보에 해당하는 모든 제품들을 user에게 보여준다.

시스템은 카테고리로 '슬리퍼&샌들', '스니커즈&슬립온', '플랫&로퍼', '워커&부츠', '힐&펌프스', 총 5가지 기본 카테고리를 제공한다.

'색깔' 세부 필터에서 시스템은 빨간색, 주황색, 황금색, 노란색, 연두색, 초록색, 하늘색, 파란색, 남색, 흰색, 회색, 그리고 검은색, 총 12가지 기본 색을 제공한다. User는 제공하는 색 중에서 찾고자 하는 색을 선택하고, 시스템은 판매자가 미리 업로드한 제품의 색과 일치하는 제품들을 사용자에게 보여준다.

'가격' 세부 필터에서 user는 가격대를 자유롭게 선택하고, 시스템은 그 범위 안으로 들어가는 모든 제품들을 user에게 보여준다.

'사이즈' 세부 필터에서 시스템은 Survey System을 통하여 알아낸 user의 발 사이즈를 DB에서 불러와, 최대한 user의 발에 적합한 사이즈가 있는 제품들을 user에게 보여준다.

'스타일' 세부 필터는 user가 Category Search를 할 때, 사용 가능한 세부 필터이다. 시스템은 '심플베이직', '럭셔리', '빈티지', '오피스슈즈', '스쿨슈즈', '캠퍼스슈즈', 총 6개의 기본 스타일을 제공한다. User는 제공하는 스타일 중에서 원하는 스타일을 선택한다. 그러면 시스템은 DB에서 미리 설정해둔 정보를 이용하여, user가 원하는 스타일에 맞는 제품들을 보여준다.

8. Recommend System

8.1 Objectives

사용자가 실제 상품을 리스트로 추천해주는 시스템을 설명한다. 사용자가 넘겨주는 상황 정보에 따라 추천해줘야 하는 정보가 달라지기 때문에 이를 Class diagram, Sequence diagram, State Diagram을 통해 Recommend System의 구조를 표현하고 설명한다.

8.2 Class Diagram

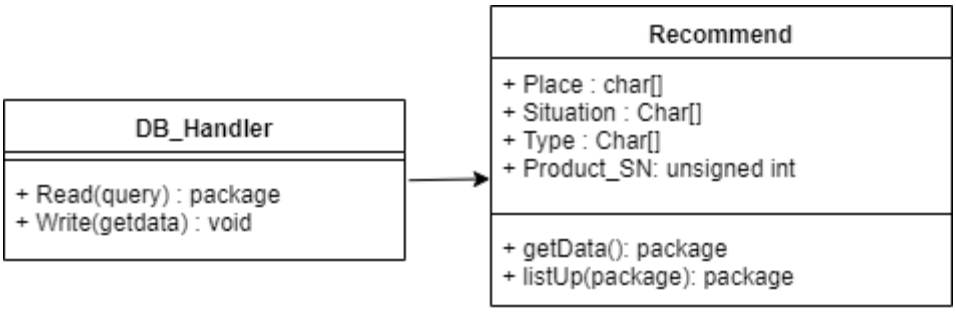

Figure 44 Recommend System Class Diagram

A. DB Handler

A.1 Attributes

해당 사항 없음.

A.2 Methods

+ package Read(query): 해당되는 DB에서 원하는 data를 읽어온다.

+ void Write(getdata): 해당되는 DB에 data를 저장한다.

B. Recommend

B.1 Attributes

+ Place: 장소 정보

+ Situation: 상황 정보

+ Type: 원하는 상품 타입 정보

+ Product_SN: 선호하는 제품의 고유번호 값

B.2 Methods

+ package getData(): DB로부터 data를 받는다.

+ package listUp(package): db로 받은 정보들을 여러 상황에 알맞게 추천해준다.

8.3 Sequence Diagram

A. Recommend System

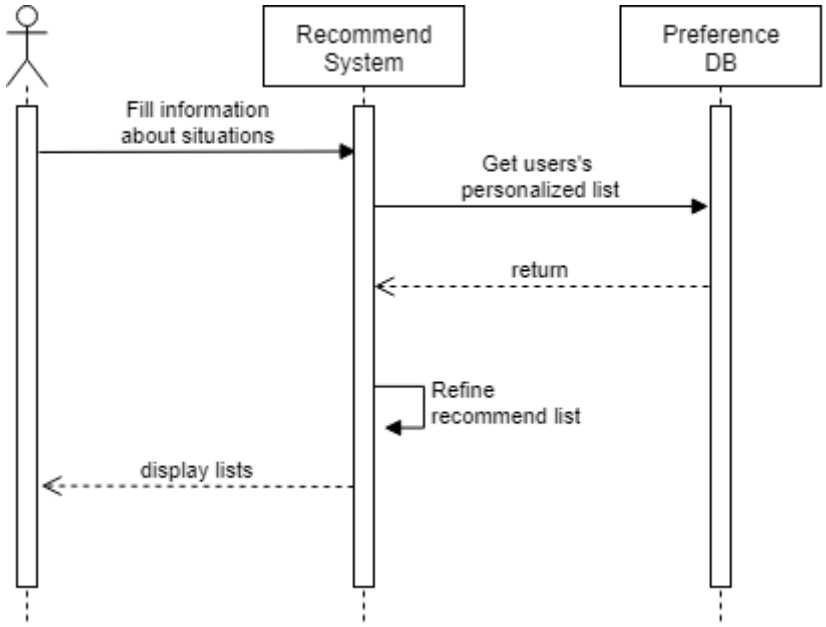

Figure 45 Recommend System Sequence Diagram

8.4 State Diagram

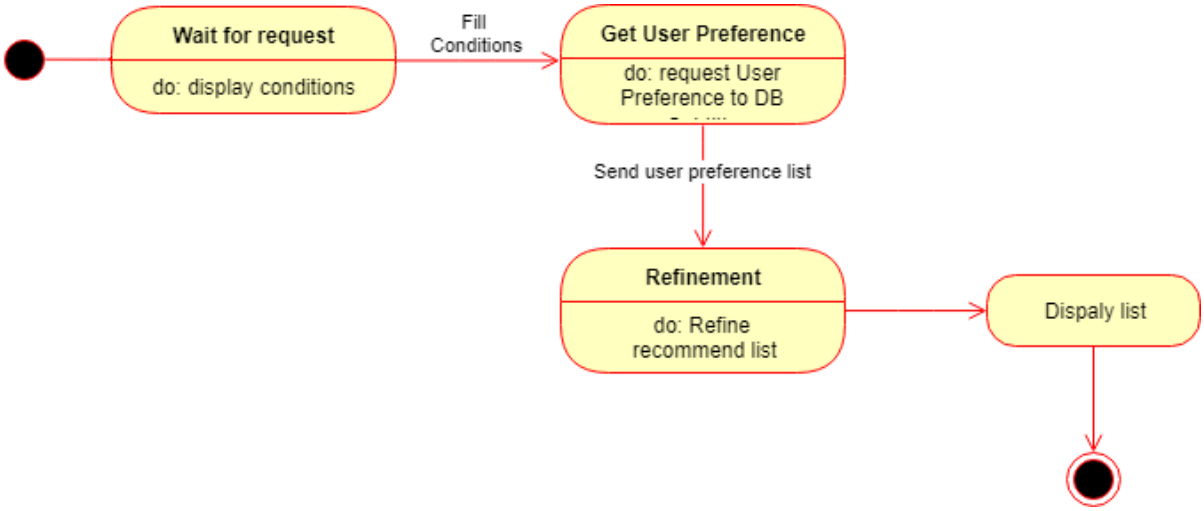

Figure 46 Recommend System State Diagram

9. Tournament System

9.1 Objectives

사용자가 선호하는 제품을 Tournament 방식으로 고를 수 있는 시스템을 설명한다. Recommend System에서 넘어온 제품 리스트를 랜덤하게 매칭하여서 최종 선호 제품을 고르거나 도중에 list로 받기 때문에 이를 case로 나누어 Class diagram, Sequence diagram, State Diagram을 통해 Tournament System의 구조를 표현하고 설명한다

9.2 Class Diagram

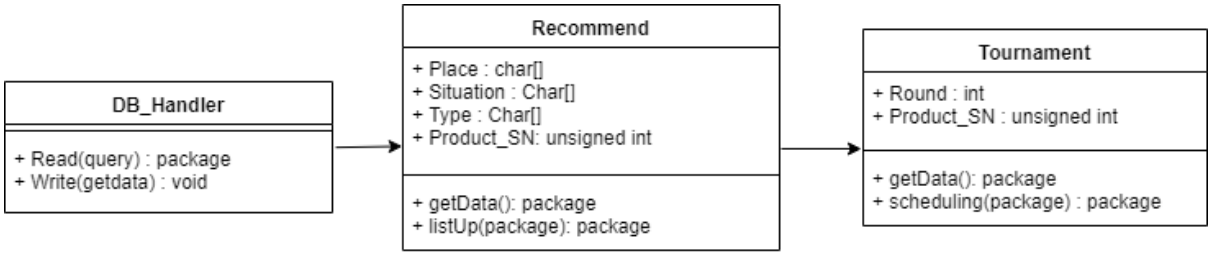

Figure 47 Tournament System Class Diagram

A. DB Handler

A.1 Attributes

해당 사항 없음.

A.2 Methods

+ package Read(query): 해당되는 DB에서 원하는 data를 읽어온다.

+ void Write(getdata): 해당되는 DB에 data를 저장한다.

B. Recommend

B.1 Attributes

+ Place: 장소 정보

+ Situation: 상황 정보

+ Type: 원하는 상품 타입 정보

+ Product_SN: 선호하는 제품의 고유번호 값

B.2 Methods

+ package getData(): DB로부터 data를 받는다.

+ package listUp(package): db로 받은 정보들을 여러 상황에 알맞게 추천해준다.

C. Tournament

C.1 Attributes

+ Round: 토너먼트 round 정보

+ Product_SN: 선호하는 제품의 고유번호 값

C.2 Methods

+ package getData(): DB로부터 data를 받는다.

+ package scheduling(package): Tournament list를 만들어준다.

9.3 Sequence Diagram

A. Normal Tournament System

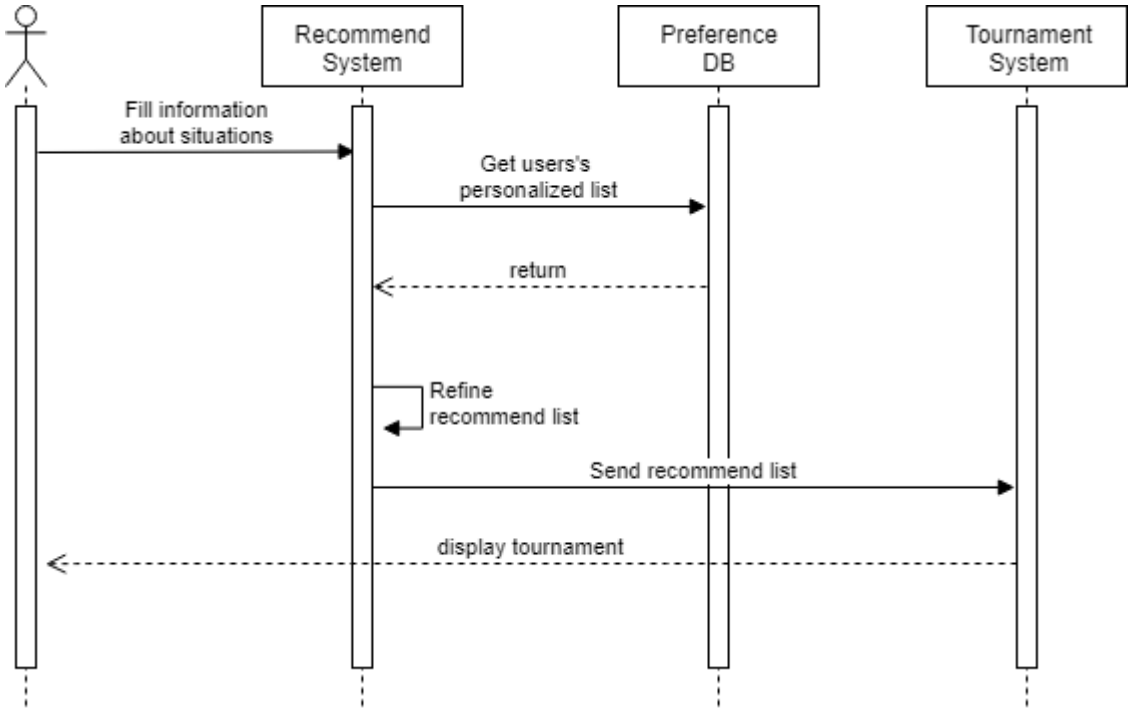

Figure 48 Tournament System Sequence Diagram 1

B. Close Tournament System

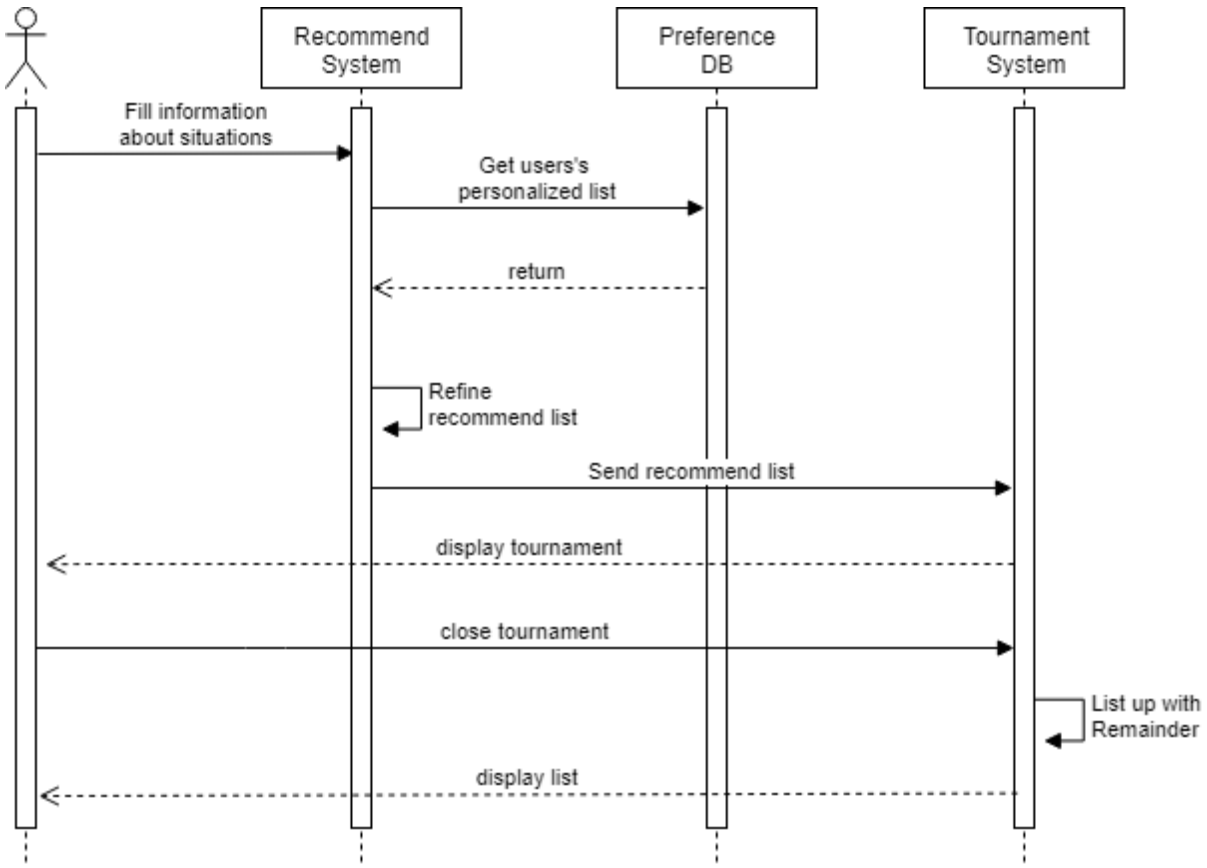

Figure 49 Tournament System Sequence Diagram 2

9.4 State Diagram

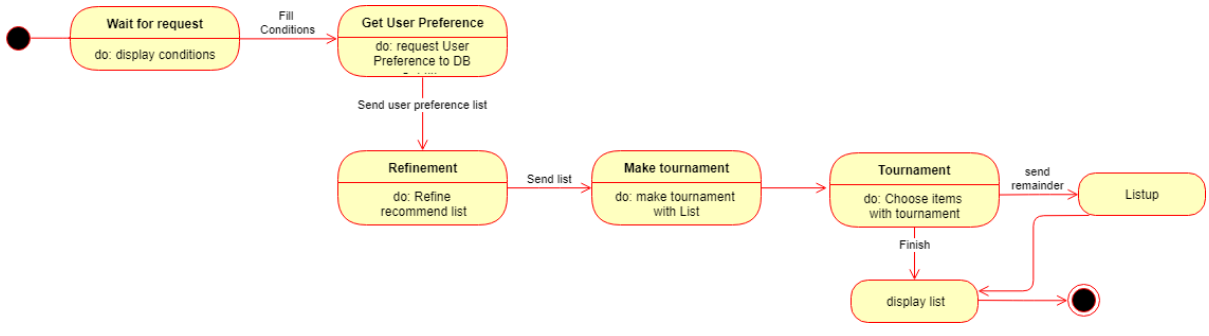

Figure 50 Tournament System State Diagram

10. Link System

10.1 Objectives

사용자가 해당 제품을 사기를 원하거나 마음에 든 제품을 파는 쇼핑몰에서 다른 제품들을 보기 원할 때, 그 쇼핑몰로 연결해주는 시스템이다. Class diagram, Sequence diagram, State Diagram을 통해 Link System의 구조를 표현하고 설명한다

10.2 Class Diagram

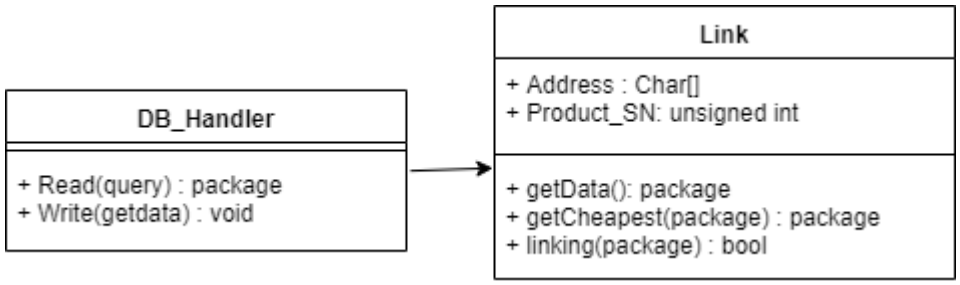

Figure 51 Link System Class Diagram

A. DB Handler

A.1 Attributes

해당 사항 없음.

A.2 Methods

+ package Read(query): 해당되는 DB에서 원하는 data를 읽어온다.

+ void Write(getdata): 해당되는 DB에 data를 저장한다.

B. Link

B.1 Attributes

+ Address: 쇼핑몰의 주소 정보

+ Product_SN: 선호하는 제품의 고유번호 값

B.2 Methods

+ package getData(): DB로부터 data를 받는다.

+ package getCheapest(package): 가장 저렴한 쇼핑몰의 link를 구한다.

+ bool linking(package): 사용자를 쇼핑몰로 연결해준다.

10.3 Sequence Diagram

A. Link System

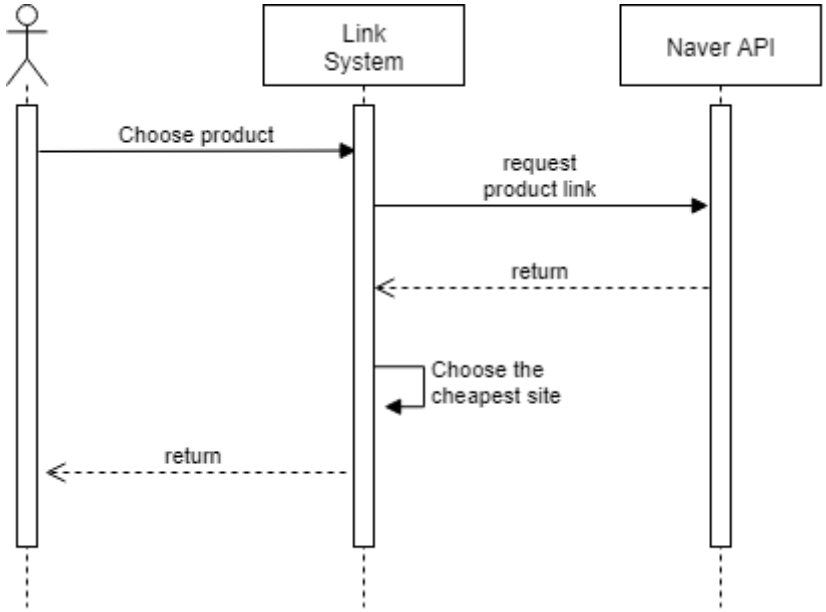

Figure 52 Link System Sequence Diagram

10.4 State Diagram

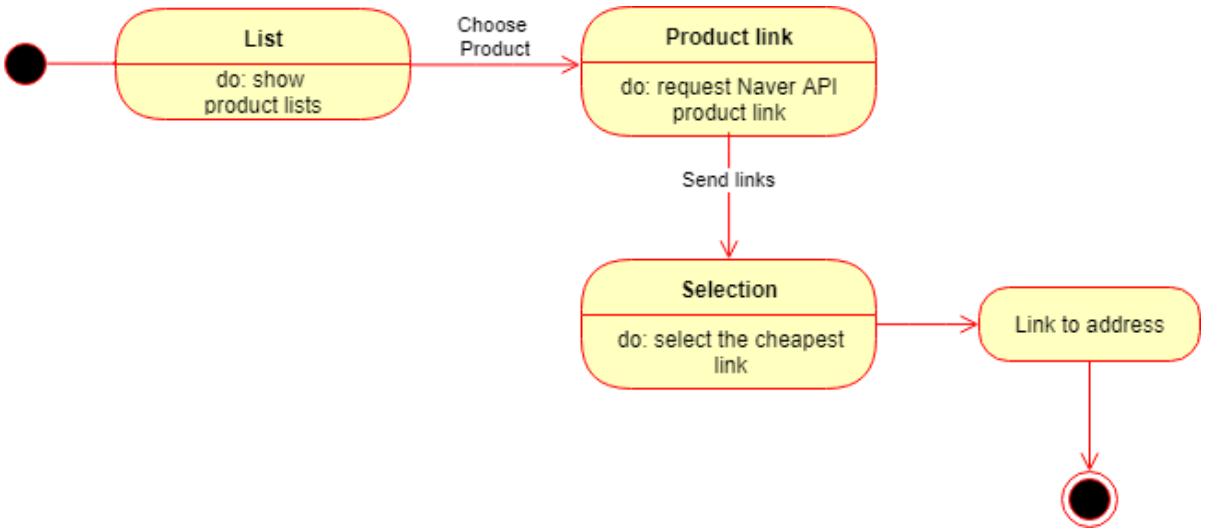

Figure 53 Link System State Diagram

11. Protocol Design

11.1 Objectives

Protocol Design에서는 Subsystem들이 상호작용하는 프로토콜에 대해 서술한다. 프로토콜의 기본 형식은 JSON을 기본으로 하며, 통신하는 메시지(Message)의 형식과 용도, 의미를 설명한다.

11.2 JSON

JSON은 JavaScript Object Notation의 약어로써 XML과 더불어 대표적인 데이터 교환 방식이다. 자바스크립트에 기반하여 만들어진 데이터 표현형식이지만 프로그래밍언어나 플랫폼에 독립적인 특성을 갖고 있어, 다양한 언어에서 JSON을 활용할 수 있다. 또한 기능이 적어 파싱(Parsing)이 빠르다는 장점을 가지고 있다. 자료의 종류에 큰 제한이 없으며, 컴퓨터 프로그램의 변수 값을 표현하는 데 적합하다. Attribute와 value의 쌍으로 표현되며, 거의 대부분의 자료형을 사용할 수 있다.

11.3 Protocol Description

A. Overview

HTTP 통신에서 client와 server사이에서 전송되는 메시지의 형태를 용도 별로 정의한다. Client에서의 요청(Request) 메시지와 Server에서의 응답(Response) 메시지메 구분한다. 해당절의 표는 캡션을 생략한다.

B. Login Protocol

B.1 Request

Attribute	Value
E-mail	사용자의 E-mail
Password	사용자의 Password

Table 2 Login Request

B.2 Response

Attribute	Value
Login_success	로그인 성공 여부

Table 3 Login Response

C. Sign up

C.1 Request

Attribute	Value
E-mail	사용자의 E-mail
Password	사용자의 Password
Name	사용자의 이름
Nickname	사용자의 닉네임
Age	사용자의 나이
Sex	사용자의 성별
Address	사용자의 주소
Call-number	사용자의 전화번호

Table 4 Sign up Request

C.2 Response

Attribute	Value
Up_success	회원가입 성공 여부

Table 5 Sign up Response

D. Product Post/Edit Protocol

D.1 Request

Attribute	Value
Product_SN	제품의 고유번호
Type	제품의 종류
Name	제품명
Price	제품의 가격
Size	제품의 크기
Material	제품의 소재
Color	제품의 색상
Brand	제품의 브랜드
Sell_URL	판매 사이트의 URL
Stock	제품의 재고
Img_URL	제품 이미지 URL

Comment_URL	상세 정보 이미지 URL

Table 6 Product Post/Edit Request

D.2 Response

Attribute	Value
Reg_success	제품 등록/수정 성공 여부

Table 7 Product Post/Edit Response

E. All Product View Protocol

E.1 Request

Attribute	Value
Product_SN	제품의 고유번호

Table 8 All Product View Request

E.2 Response

Attribute	Value
Type	제품의 종류
Name	제품명
Price	제품의 가격
Size	제품의 크기
Material	제품의 소재
Color	제품의 색상
Brand	제품의 브랜드
Sell_URL	판매 사이트의 URL
Rank	제품의 평점
Stock	제품의 재고

Table 9 All Product View Response

F. Product Search Protocol

F.1 Request

Attiribute	Value

Search_filter	Attribute	Value
	Search_scope	검색할 범위
	Search_word	검색할 단어

Table 10 Product Search Request

F.2 Response

Attiribute	Value	
Search_filter	Attribute	Value
	Type	제품의 종류
	Name	제품명
	Price	제품의가격
	Size	제품의 크기
	Material	제품의 소재
	Color	제품의 색상
	Brand	제품의 브랜드
	Sell_URL	쇼핑몰의 URL
	Rank	제품의 평점
	Stock	제품의 재고
	Img_URL	제품 이미지 URL
	Comment_URL	상세 정보 이미지 URL

Table 11 Product Search Response

G. Insert(Survey)/Edit preferences Protocol

G.1 Requeset

Attribute	Value
Size	사용자의 발 사이즈
Brand	사용자의 선호 브랜드
Type	사용자가 선호하는 스타일
Product_SN	사용자가 선호하는 제품

Table 12 Insert(Survey)/Edit preferences Request

G.2 Response

Attribute	Value

Reg_success	Preference 등록/수정 성공 여부

Table 13 Insert(Survey)/Edit preferences Response

H. Recommend Product View Protocol

H.1 Request

Attribute	Value
User_SN	사용자의 고유번호

Table 14 Recommend Product View Reqeust

H.2 Response

Attiribute	Value	
Search_filter	Attribute	Value
	Type	제품의 종류
	Name	제품명
	Price	제품의가격
	Size	제품의 크기
	Material	제품의 소재
	Color	제품의 색상
	Brand	제품의 브랜드
	Sell_URL	판매 사이트의 URL
	Rank	제품의 평점
	Stock	제품의 재고
	Img_URL	제품 이미지 URL
	Comment_URL	상세 정보 이미지 URL

Table 15 Recommend Product View Response

I. Tournament Protocol

I.1 Request

Attribute	Value
User_SN	사용자의 고유번호

Table 16 Tournament Request

I.2 Response

Attiribute	Value	
Tournament_filter	Attribute	Value
	Type	제품의 종류
	Name	제품명
	Img_URL	제품 이미지 URL

Table 17 Tournament Response

J. Link Shop Protocol

J.1 Request

Attribute	Value
Product_SN	구매할 제품의 고유번호

Table 18 Link Shop Request

J.2 Response

Attribute	Value
Sell_URL	해당 제품을 판매하는 쇼핑몰 URL

Table 19 Link Shop Response

12. Database Design

12.1 Objectives

Database Design은 요구사항 명세서에서 기술한 데이터베이스 요구사항을 바탕으로 수정사항을 수정하여 다시 요구사항을 작성하였다. 요구사항을 바탕으로 ER Diagram을 작성하고, 이를 이용하여 Relational Schema를 작성하고, Normalization을 통해 Redundancy와 Anomaly를 제거한 후 마지막으로 SQL DDL을 작성한다.

12.2 ER Diagram

Entity는 분리된 물체 하나를 표현한다. Entity는 사각형으로 표현되며, Relationship은 다이아몬드로 표현된다. Entity나 Relationship는 Attributes를 가질 수 있으며, 이 Attributes는 관계 집합에 실선으로 연결된 타원형으로 표현한다. Relationship은 두 개 이상의 Entity들의 연관 관계를 표현한다. 모든 Entity는 고유하게 식별되는 Attributes 집합을 가지고 있어야 하며, 최소 한의 고유 식별 Attributes 집합은 Entity의 기본 키(Primary Key)라 불린다.

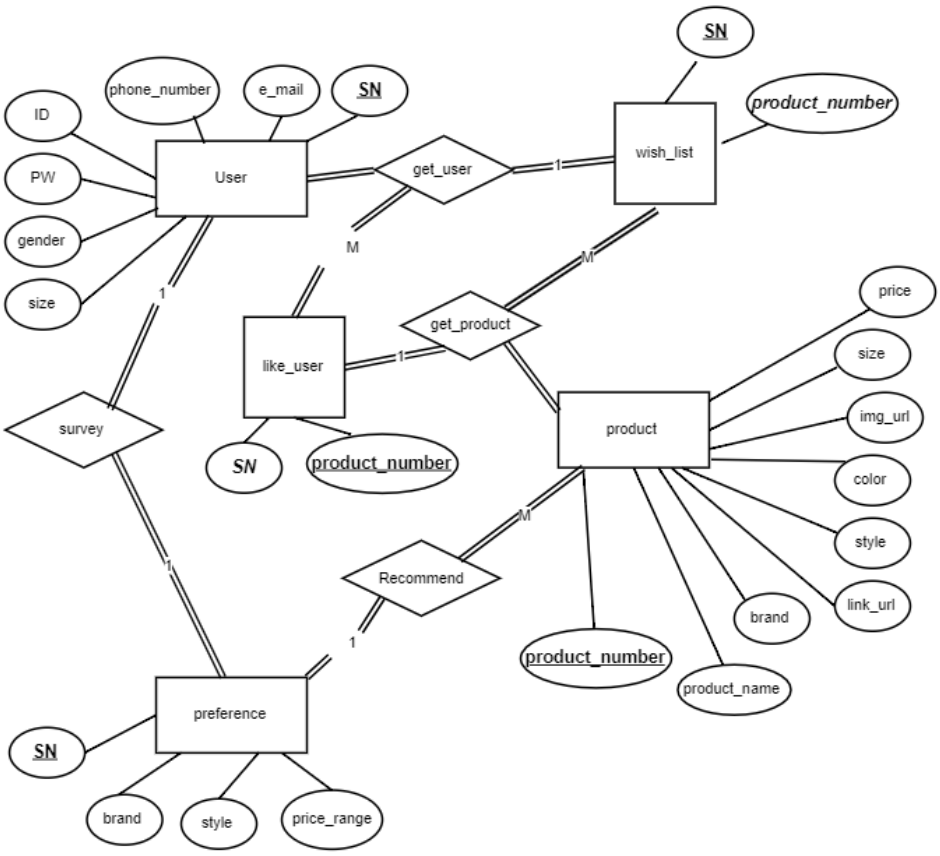

Figure 54 Overall ER Diagram

A. Entity

A.1 User

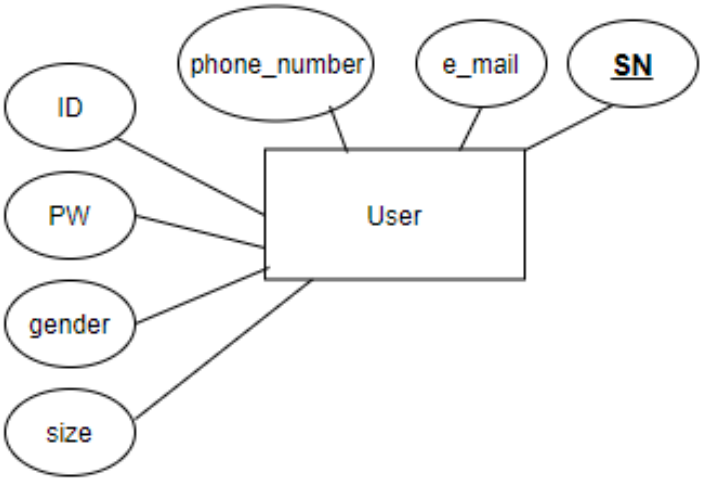

Figure 55 User Entity

User는 회원의 정보를 나타낸다. Key는 Serial number(SN)이고, e_mail, phone_number, ID, PW, gender, size의 속성을 가지고 있다.

A.2 Preference

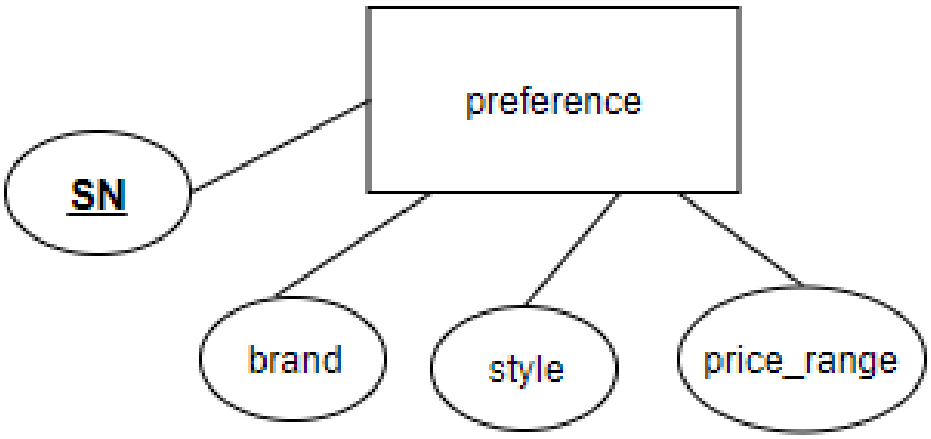

Figure 56 Preference Entity

Preference는 회원의 선호도에 대한 정보를 나타낸다. Key는 Serial number(SN)이고, brand, style, price_range의 속성을 가지고 있다.

A.3 Product

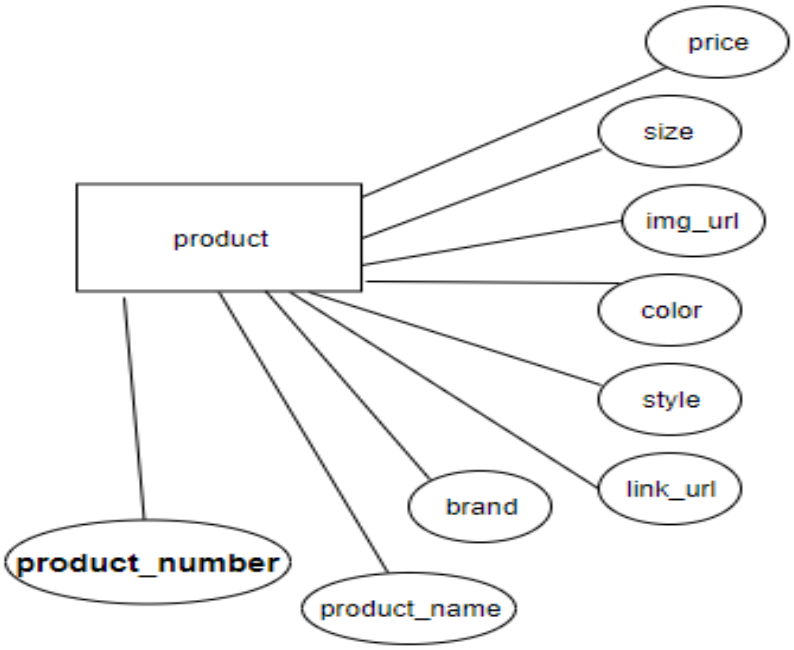

Figure 57 Product Entity

Products는 상품에 대한 정보를 나타낸다. Key는 product_number이고, product_name, brand, link_url, style, color, img_url, size, price에 대한 속성을 가지고 있다.

A.4 wish_list

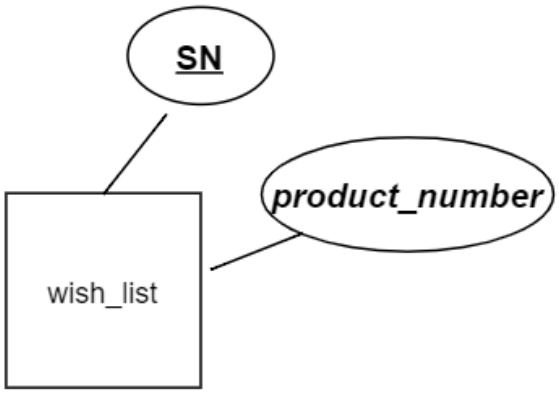

Figure 58 wish_list Entity

Wish_list는 user가 찜한 상품에 대한 정보를 나타낸다. Key는 user entity의 SN을 foreign key로 가져와서 사용하고, product로부터 product_number를 foreign key로 가져와서 속성에 대한 정보를 가지고 있다.

A.5 like_user

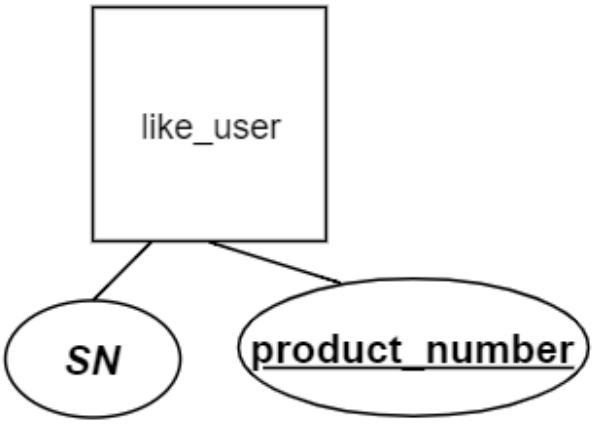

Figure 59 like_user Entity

Like_user entity는 상품에 대해 like, unlike를 누른 user에 대한 정보를 나타낸다. Key는 product의 product_number를 foreign key로 가져와서 사용하고, user의 SN을 foreign key로 가져와서 속성 정보를 나타낸다.

B. Relationship

B.1 Survey

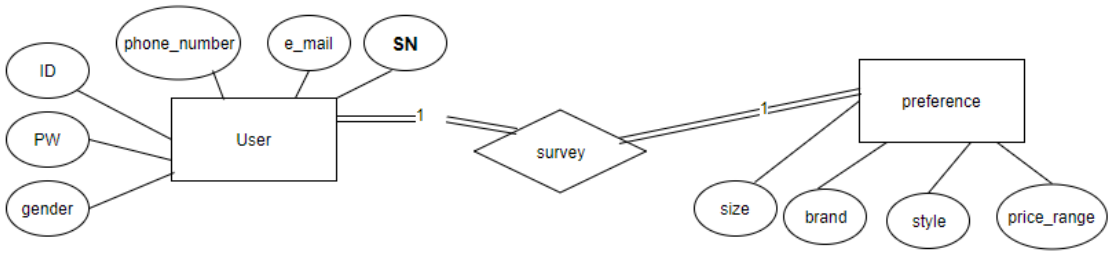

Figure 60 Survey Relationship

Survey Relationship은 survey를 통해서 user의 preference를 획득하는 관계로, 1명 user는 하나의 preference 집합을 가진다.

B.2 Wish_list

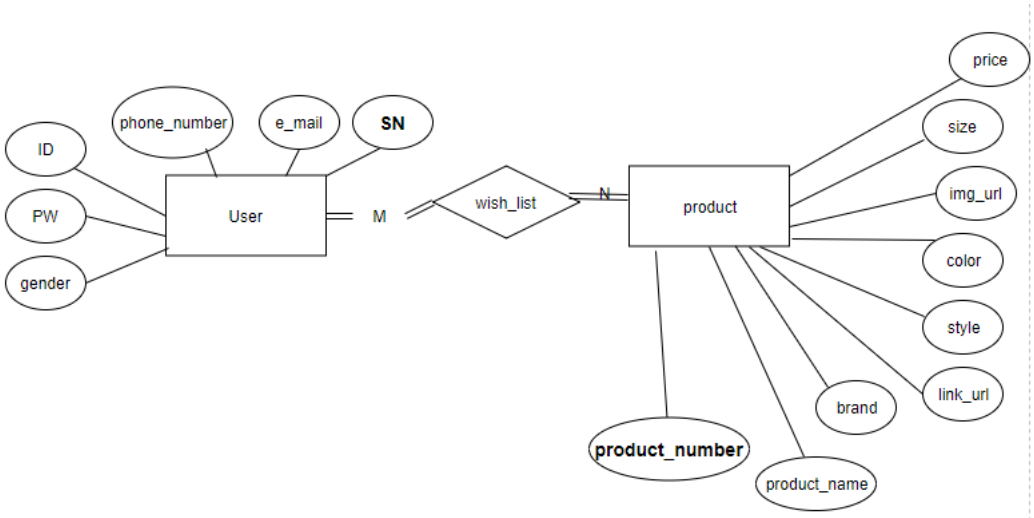

Figure 61 Wish_list Relationship

Wish_list Relationship은 user가 마음에 든 product를 wish list에 넣는 관계로, 1명의 user는 1개의 wish list를 가지고, 그 wish list는 여러 상품을 포함할 수 있다. 또한, 1개의 product는 여러 user의 wish list에 담길 수 있다.

B.3 Like

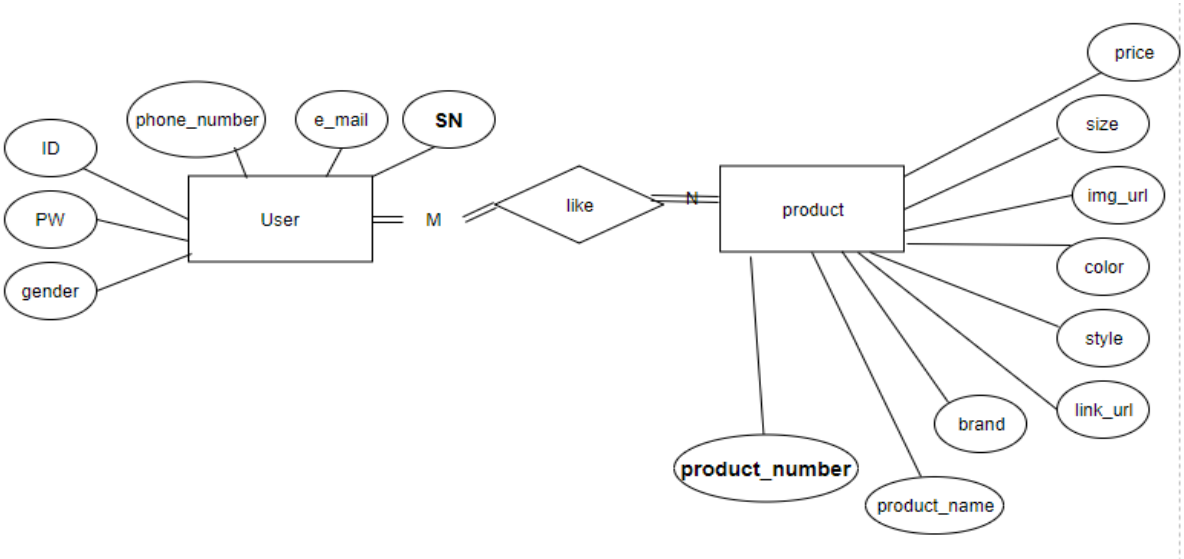

Figure 62 Like Relationship

Like Relationship은 user가 해당 product에 대해 직접 선호를 표시하는 관계로, 1명의 user는 각 상품 당 한 번의 평가를 할 수 있다. 또한 1개의 상품은 여러 명의 user로부터 평가 받을 수 있다.

B.4 Recommend

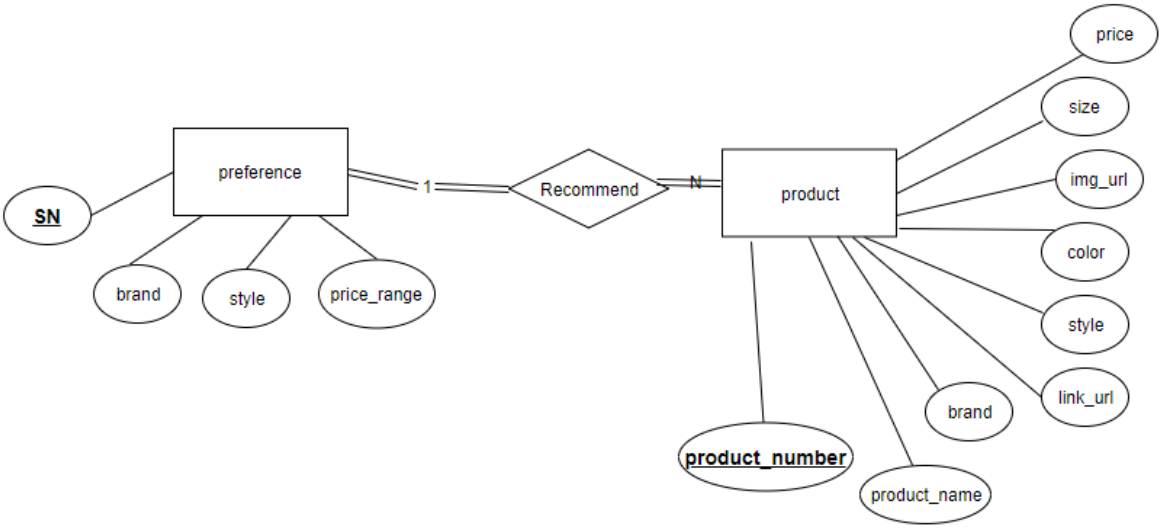

Figure 63 Recommend Relationship

Recommend Relationship은 user의 preference에 맞는 product를 추천해주는 관계로, 하나의 preference 집합으로 여러 개의 product를 추천 받을 수 있다.

B.5 get_wish_list

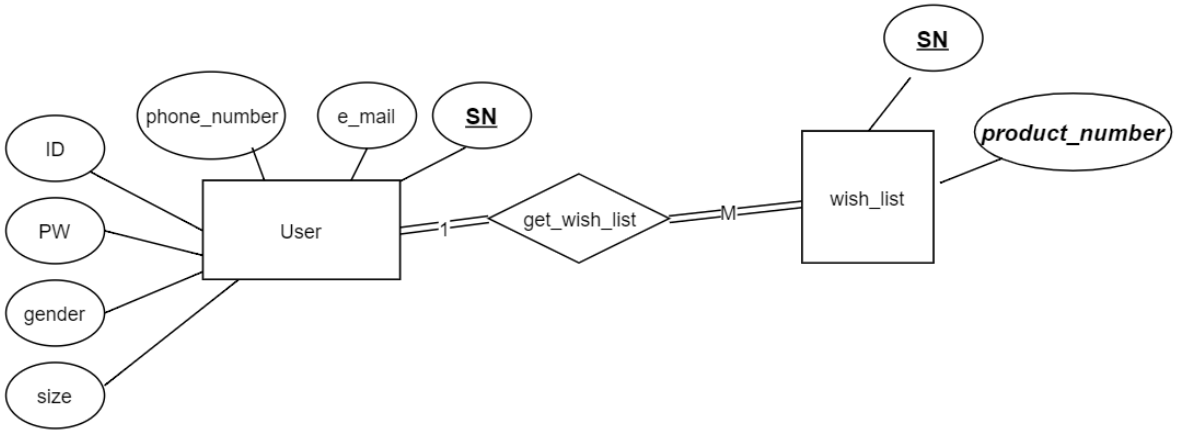

Figure 64 get_ wish_list Relationship

Get_wish_list relationship은 user의 wish_list를 가져오는 relationship으로, 한 명의 user는 하나의 wish_list 안에 여러 product를 넣을 수 있다.

B.6 get_like_user

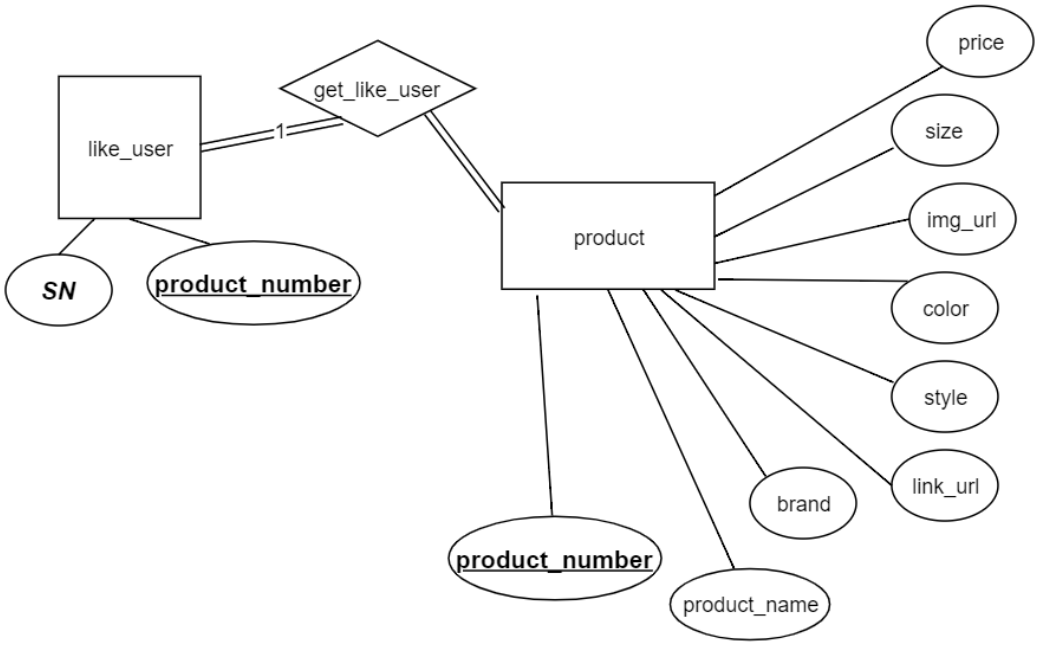

Figure 65 get_like_user Relationship

Get_ like_user relationship은 product에 대해 like나 dislike를 누른 user의 list를 가져오는 relationship으로, 하나의 product는 하나의 like_user에 관한 list를 가지며, like_user는 여러 명의 user 정보를 담고 있을 수 있다.

12.3 Relational Schema

A. User

- SN, e_mail, phone_number, ID, PW, gender, size, name

- PK(Primary Key) : SN

- FK(Foreign Key) : 없음.

- FUNCTION DEPENDENCY(FD) :

SN → { e_mail, phone_number, ID, PW, gender, size, name }

ID → { SN, e_mail, phone_number, PW, gender, size, name }

- Description : Entity User에 관한 테이블이다. 모든 속성에 Null을 허용하지 않는다.

B. Preference

- SN, brand, style, price_range

- PK(Primary Key) : SN

- FK(Foreign Key) : SN

- FUNCTION DEPENDENCY(FD) :

SN → { brand, style, price_range }

- Description : Entity Preference에 관한 테이블이다. 모든 속성에 Null을 허용하지 않는다. User의 PK인 SN을 FK로 가져온다.

C. Product

- product_number, product_name, brand, link_url, style, color, img_url, size, price

- PK(Primary Key) : product_number

- FK(Foreign Key) : 없음

- FUNCTION DEPENDENCY(FD) :

product_number → { product_name, brand, link_url, style, color, img_url, size, price }

- Description : Entity Product에 관한 테이블이다. 모든 속성에 Null을 허용하지 않는다.

D. Wish_list

- product_number

- PK(Primary Key) : product_number

- FK(Foreign Key) : product_number

- Description : Entity wish_list에 관한 테이블이다. 모든 속성에 Null을 허용하지 않는다.

E. Like_user

- SN

- PK(Primary Key) : SN

- FK(Foreign Key) : SN

- Description : Entity Like_user에 관한 테이블이다. 모든 속성에 Null을 허용하지 않는다.

12.4 SQL DDL

A. User

```
CREATE TABLE 'User' (
'SN' int(11) NOT NULL AUTO_INCREMENT,
'e_mail' varchar(50) NOT NULL,
'phone_number' int(11) DEFAULT NULL,
'ID' varchar(30) NOT NULL,
'PW' varchar(45) NOT NULL,
'gender' varchar(6) NOT NULL,
'size' int(6) NOT NULL,
'name' varchar(45) NOT NULL,
PRIMARY KEY('SN'),
UNIQUE KEY 'ID_UNIQUE' ('ID')
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

SN은 PRIMARY KEY이므로 NOT NULL이며 AUTO-INCREMENT이다.

B. Preference

```
CREATE TABLE 'Preference' (
'SN' int(11) NOT NULL AUTO_INCREMENT,,
'brand' varchar(30) NOT NULL,
'style' varchar(45) NOT NULL,
'price_range' int(10) NOT NULL,
PRIMARY KEY('SN'),
KEY 'user_idx'('SN'),
CONSTRAINT 'user_idx' FOREIGN KEY('SN') REFERENCES 'User'('SN') ON DELETE NO ACTION ON UPDATE NO ACTION,
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

SN은 PRIMARY KEY이므로 NOT NULL이며 AUTO-INCREMENT이다.

C. Product

```
CREATE TABLE 'Product' (
'product_number' int(11) NOT NULL AUTO_INCREMENT,
'product_name' varchar(45) NOT NULL,
'brand' varchar(30) NOT NULL,
'style' varchar(45) NOT NULL,
'price' int(10) NOT NULL,
'size' int(6) NOT NULL,
'color' varchar(15) NOT NULL,
'link_url' varchar(200) NOT NULL,
'img_url' varchar(200) NOT NULL,
PRIMARY KEY('product_number')
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Product_number는 PRIMARY KEY이므로 NOT NULL이며 AUTO-INCREMENT이다.

D. Wish_list

```
CREATE TABLE 'Wish_list' (
'product_number' int(11) NOT NULL AUTO_INCREMENT,,
PRIMARY KEY('product_number'),
KEY 'product_idx'('product_number'),
CONSTRAINT 'product_idx" FOREIGN KEY('product_number') REFERENCES
'Product'('product_number') ON DELETE NO ACTION ON UPDATE NO ACTION,
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Product_number는 PRIMARY KEY이므로 NOT NULL이며 AUTO-INCREMENT이다.

E. Like_user

```
CREATE TABLE ‘Like_user’ (

‘SN’ int(11) NOT NULL AUTO_INCREMENT,,

PRIMARY KEY(‘SN’),

KEY ‘user_idx’(‘SN’),

CONSTRAINT ‘user_idx” FOREIGN KEY(‘SN’) REFERENCES ‘User’(‘SN’) ON DELETE NO
ACTION ON UPDATE NO ACTION,

) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

SN은 PRIMARY KEY이므로 NOT NULL이며 AUTO-INCREMENT이다.

13. Testing Plan

13.1 Objectives

시스템이 의도한 방향으로 실행되는지를 확인하고 시스템 내부의 결함을 찾기 위해 testing 한다. 이러한 testing을 설계 단계에 계획한다. 이 때 Testing Plan에서는 Testing Policy 뿐만 아니라 여러 test case에 대해 기술한다.

13.2 Testing Policy

Shoose 시스템의 개발에서는 크게 3단계로 나누어 testing을 진행한다. Development testing, release testing, user testing으로 나눠진다.

A. Development testing

개발 과정에서 수행되는 testing으로, 다음과 같은 세부 testing으로 구성된다.

- Unit testing

  Subsystem을 개발 후, 제대로 동작하는지 확인한다.

- Integration testing

  Subsystem을 incremental하게 통합하면서 제대로 동작하는지 확인한다.

- System testing

  Subsystem을 모두 합친 후, 하나의 system으로 제대로 동작하는지 확인한다.

- Acceptance testing

  사용자의 정보를 이용하여 사용자의 요구사항이 시스템에 제대로 반영되었는지 확인한다.

B. Release testing

Release에 앞서서 최종적으로 확인한다. Requirement specification의 요구사항이 시스템에 제대로 반영되었는지 확인한다.

C. User testing

사용자가 실제 동작 환경에서 시스템을 확인한다.

13.3 Test Case

A. User Management System

A.1 Sign up for User with using NAVER

1) User: 'NAVER'을 이용하여 회원가입을 시도한다.

2) 시스템 동작: User DB에 있는 ID와 중복 여부를 확인한다.

2-1) (Sign-up 성공)

시스템 알림: "회원가입이 완료되었습니다."

시스템 동작: User DB에 새로 만든 ID/PW를 저장한다.

2-2) (Sign-up 실패)

시스템 알림: "이미 사용중인 ID입니다."

시스템 동작: Sign-up 시퀀스를 다시 시작한다.

A.2 Sign up for User without using NAVER

1) User: 'NAVER'을 이용하지 않고 회원가입을 시도한다.

2) 시스템 동작: User DB에 있는 ID와 중복 여부를 확인한다.

2-1) (Sign-up 성공)

시스템 알림: "회원가입이 완료되었습니다."

시스템 동작: User DB에 새로 만든 ID/PW를 저장한다.

2-2) (Sign-up 실패)

시스템 알림: "이미 사용중인 ID입니다."

시스템 동작: Sign-up 시퀀스를 다시 시작한다.

A.3 Log in for User with using NAVER

1) User: 'NAVER'을 이용하여 로그인을 시도한다.

2) 시스템 동작: 입력 받은 ID/PW가 User DB에 있는지 확인한다.

2-1) (Log-in 성공)

시스템 알림: "로그인이 완료되었습니다."

시스템 동작: User DB, wishlist DB에서 ID/PW에 맞는 데이터를 불러온다.

2-2) (Log-in 실패)

시스템 알림: "존재하지 않는 ID이거나, PW가 맞지 않습니다."

시스템 동작: Log-in 시퀀스를 다시 시작한다.

A.4 Log in for User without using NAVER

1) User: 'NAVER'을 이용하지 않고 로그인을 시도한다.

2) 시스템 동작: 입력 받은 ID/PW가 User DB에 있는지 확인한다.

2-1) (Log-in 성공)

시스템 알림: "로그인이 완료되었습니다."

시스템 동작: User DB, wishlist DB에서 ID/PW에 맞는 데이터를 불러온다.

2-2) (Log-in 실패)

시스템 알림: "존재하지 않는 ID이거나, PW가 맞지 않습니다."

시스템 동작: Log-in 시퀀스를 다시 시작한다.

B. Profile Customizing System

B.1 User's customization of 'Basic user information'

1) User: User DB에서 불러온 data를 수정한다.

2) 시스템 동작: 수정한 data를 User DB에 업데이트한다.

2-1) (Update 성공)

시스템 알림: "수정이 완료되었습니다."

시스템 동작: 초기 화면으로 돌아간다.

2-2) (Update 실패)

시스템 알림: "수정에 실패했습니다"

시스템 동작: Profile customizing 시퀀스를 다시 시작한다.

B.2 User's customization of 'User preferences'

1) User: preference DB에서 불러온 data를 수정한다.

2) 시스템 동작: 수정한 data를 preference DB에 업데이트한다.

2-1) (Update 성공)

시스템 알림: "수정이 완료되었습니다."

시스템 동작: 초기 화면으로 돌아간다.

2-2) (Update 실패)

시스템 알림: "수정에 실패했습니다"

시스템 동작: Profile customizing 시퀀스를 다시 시작한다.

C. Recommend System

C.1 Recommendation with situation

1) User: 상황에 관련된 정보를 입력한다.

2) 시스템 동작: 상황 정보가 형식에 맞는지 확인한다.

2-1) (Situation information: Appropriate Format)

시스템 알림: "추천 목록을 생성합니다."

시스템 동작: Preference DB에서 personalized list를 가져오고, 입력 받은 상황 정보를 반영하여 recommend list를 만들어 출력한다.

2-2) (Situation information: Wrong Format)

시스템 알림: "상황 정보가 형식에 맞지 않습니다."

시스템 동작: Recommendation with situation 시퀀스를 다시 시작한다.

D. Tournament System

D.1 Tournament with situation

1) User: 상황에 관련된 정보를 입력하고, 시스템에서 보여주는 tournament 선택지에 응답한다.

2) 시스템 동작: recommend system을 통해 recommend list를 만든다.

2-1) (Situation information: Appropriate Format)

시스템 알림: "Tournament를 시작합니다."

시스템 동작: recommend list를 가져와 tournament를 만들어 user에게 선택지를 제공, user가 선택한 제품들로 display list를 만들어 출력한다.

2-2) (Situation information: Wrong Format)

시스템 알림: "상황 정보가 형식에 맞지 않습니다."

시스템 동작: Tournament with situation 시퀀스를 다시 시작한다.

E. Link System

E.1 Link to cheapest shopping mall

1) User: 제품 목록에서 제품을 선택한다.

2) 시스템 동작: 해당하는 제품이 있는 쇼핑몰 목록을 NAVER API를 통해 요청한다.

2-1) (Product exists)

시스템 알림: "가장 저렴한 쇼핑몰로 이동합니다."

시스템 동작: 받아온 쇼핑몰 목록에서 가장 저렴한 가격을 가진 쇼핑몰 주소를 찾아 연결한다.

2-2) (Product not exists)

시스템 알림: "해당하는 제품을 판매하는 쇼핑몰이 존재하지 않습니다."

시스템 동작: Link to cheapest shopping mall 시퀀스를 다시 시작한다.

F. Search System

F.1 Product search

1) User: 제품명을 입력하여 search product를 요청한다.

2) 시스템 동작: 해당하는 제품이 Product DB에 있는지 확인하고 그 결과를 출력한다. 출력할 때, 세부 필터를 적용한다.

2-1) (Products exist)

시스템 알림: "검색이 완료되었습니다."

시스템 동작: 검색 결과를 출력한다.

2-2) (Products exist, but not appropriate filter)

시스템 알림: "세부 필터에 해당하는 제품이 존재하지 않습니다."

시스템 동작: Product search 시퀀스를 다시 시작한다.

2-3) (No products)

시스템 알림: "해당하는 제품이 존재하지 않습니다."

시스템 동작: Product search 시퀀스를 다시 시작한다.

F.2 Category search

1) User: 제품의 카테고리(범주)를 입력하여 search product를 요청한다.

2) 시스템 동작: 해당하는 제품이 Product DB에 있는지 확인하고 그 결과를 출력한다. 출력할 때, 세부 필터를 적용한다.

2-1) (Products exist)

시스템 알림: "검색이 완료되었습니다."

시스템 동작: 검색 결과를 출력한다.

2-2) (Products exist, but not appropriate filter)

시스템 알림: "세부 필터에 해당하는 제품이 존재하지 않습니다."

시스템 동작: Product search 시퀀스를 다시 시작한다.

2-3) (No products)

시스템 알림: "해당하는 제품이 존재하지 않습니다."

시스템 동작: Category search 시퀀스를 다시 시작한다.

G. Survey System

G.1 Preference survey: No previous data

1) User: survey의 format에 맞춰 답변을 작성한다.

2) 시스템 동작: 답변이 format에서 벗어나지 않는지 확인한다음 preference DB에 저장한다.

2-1) (Appropriate Format)

시스템 알림: "설문조사 참여에 감사합니다."

시스템 동작: 초기 화면으로 돌아간다.

2-2) (Wrong Format)

시스템 알림: "답변이 형식에서 벗어납니다."

시스템 동작: survey 시퀀스를 다시 시작한다.

G.2 Preference survey: Edit

1) User: My profile에서 preference DB의 data를 수정한다.

2) 시스템 동작: 수정한 data가 format에서 벗어나지 않는지 확인한다음 preference DB에 업데이트한다.

2-1) (Appropriate Format)

시스템 알림: "수정에 성공했습니다."

시스템 동작: 초기 화면으로 돌아간다.

2-2) (Wrong Format)

시스템 알림: "수정한 preference가 형식에서 벗어납니다."

시스템 동작: preference survey: Edit 시퀀스를 다시 시작한다.

14. Development Plan

14.1 Objectives

Develop Plan에서는 개발 계획에 대해 서술한다. 시스템의 개발 실제 흐름에 대해서는 Gantt chart를 활용하여 서술한다.

14.2 Gantt Chart

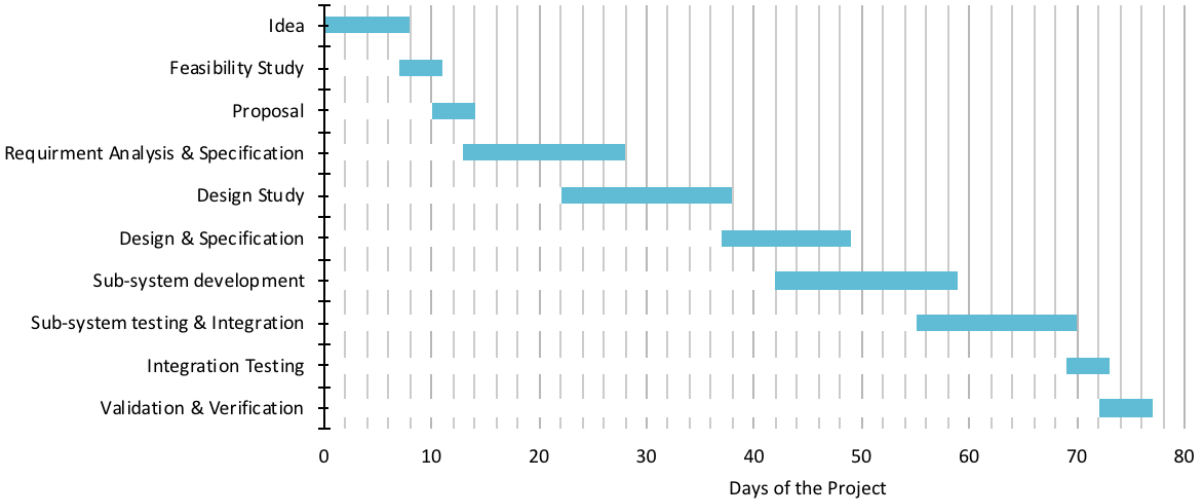

Figure 66 Gantt Chart

위의 Gantt chart는 개발 계획에 따른 실제 개발 상황을 나타내고 있다. 기존 계획했던 스케쥴에 크게 벗어나고 있지 않으나, 요구사항 명세서 작성에 생각보다 많은 시간이 소요되었다.

시스템 전체 과정에서는 각각의 단계에 대한 문서화를 통해 요구사항과 설계 명세를 명확히하려 노력하였고, 위의 정해진 일정외에도 지속적으로 변경사항을 반영하도록 한다. 이후 개발 일정 역시 계획에 맞게 최대한 지연되지 않을 수 있도록 노력할 예정이다.

15. Index

15.1 Table Index

15.2 Figure Index

16. References

1. UML
- https://en.wikipedia.org/wiki/Unified_Modeling_Language
- http://asfirstalways.tistory.com/95
- https://www.omg.org/

2. Package Diagram
- https://en.wikipedia.org/wiki/Package_diagram
- http://securesearch.tistory.com/161

3. Deployment Diagram
- https://en.wikipedia.org/wiki/Deployment_diagram
- https://www.ibm.com/support/knowledgecenter/ko/SS4JE2_7.5.5/com.ibm.xtools.modeler.doc/topics/cdepd.html

4. Class Diagram
- https://en.wikipedia.org/wiki/Class_diagram
- https://www.lucidchart.com/pages/uml-class-diagram

5. State Diagram
- https://online.visual-paradigm.com/tutorials/state-machine-diagram-tutorial/

6. Sequence Diagram
- https://en.wikipedia.org/wiki/Sequence_diagram

7. Data Flow Diagram
- http://cjmyun.tripod.com/Knowledgebase/DFD.htm
- https://www.smartdraw.com/data-flow-diagram/

8. Entity Relationship Diagram
- https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model
- http://jwprogramming.tistory.com/49

9. Android Studio
- https://developer.android.com/studio/features