중년들의 패션친구

2013312262 권영재

2015311777 최민진

2016311589 오정호

2012312989 정정환

2016313475 오현지

2016313354 위응주

목차

1. Preface

1.1. Objective

Preface에서는 본 문서의 독자를 정의하고, 구조를 소개한다. 구조를 소개할 때는 각 목차의 목적을 서술한다. 또, 문서의 버전에 대해 버전 포맷, 버 전 관리 정책, 버전 업데이트 기록으로 나누어서 서술한다.

1.2. Readership

본 문서의 독자는 다음과 같다. 시스템을 직접 개발하는 소프트웨어 엔지니어, 시스템을 설계하는 아키텍처와 개발에 참여하는 모든 구성원을 독자로 정의한다. 만약, 시스템을 개발할 때 외주 업체를 이용한다면, 해당 업체에 서 개발에 관련되는 모든 구성원 역시 독자에 포함한다. 즉, 본 문서의 독자는 본 문서에서 소개하는 시스템의 개발 및 유지 보수에 관련된 모든 구성 원이다.

1.3. Document Structure

A. Preface

Preface에서는 본 문서의 독자를 정의하고, 구조를 소개한다. 구조를 소개할 때는 각 목차의 목적을 서술한다. 또, 문서의 버전에 대해 버전 포맷, 버전 관리 정책, 버전 업데이트 기록으로 나누어서 서술한다.

B. Introduction

Introduction에서는 본 문서에서 시스템의 설계할 때 사용하는 모든 종류의 다이어그램 및 툴에 관해 서술한다.

C. User Management System

회원 가입과 로그인 과정에서 발생하는 데이터의 처리를 진행하는 사용자 관리 시스템의 설계를 설명한다. Class diagram, Sequence diagram과 State Diagram을 통해 User Management System의 구조를 표현하고 설명한다.

D. Friends Management System

User가 친구들을 추가할 수 있도록 친구들의 리스트를 보여주고, 친구를 nickname을 통해 검색하고, 그런 과정들을 보여준다.

E. Post Management System

사용자의 게시물을 관리하는 시스템의 설계를 설명한다. Post Management System의 경우, 실제 게시물의 내용을 관리하는 Content Management System과 게시물의 댓글을 관리하는 Comment Management System, 다시 하위 2개 시스템으로 나뉜다. Class diagram, Sequence diagram과 State Diagram을 통해 Post Management System의 구조를 표현하고 설명한다.

F. Display System

실제 사용자가 사용하며 탭을 클릭하였을 때 발생하는 데이터를 처리하고 사용자의 화면에 보여주는 Display 시스템을 설명한다. Class diagram, Sequence diagram과 State Diagram을 통해 Display 시스템의 구조를 표현하고 설명한다.

G. Extract preferred posts system

사용자가 등록한 관심 태그를 가진 친구들의 게시글을 추출하는 시스템의 설계를 설명한다. Class diagram, Sequence diagram과 State Diagram을 통해 Extract preferred posts system의 구조를 표현하고 설명한다.

H. Protocol Design

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

I. 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.

J. Testing Plan

시스템이 의도한 방향으로 실행되고 시스템 내부의 결함을 찾기 위해 testing을 한다. 이를 위해 설계단계에 미리 계획한다. 이 때 Testing Plan에서는 Testing Policy와 여러 Test Case에 대해 기술한다.

K. Development Environment

Development Environment에서는 개발자의 환경에 대해 설명한다. 사용한 프로그래밍 언어와 IDE에 대해 서술한다.

L. Develop Plan

Develop plan에서는 개발 계획에 대해 서술한다. 이 때, Gantt chart를 이용하여 개발 계획과 실제 계발 흐름에 대해 서술하고자 한다.

M. Index

Index에서는 본 문서의 표, 다이어그램 및 사진들의 인덱스를 표시한다. 이를 통해 원하는 정보가 문서의 몇 페이지에 있는 지 알 수 있다.

1.4. Version of the Document

A. Version Format

버전 번호는 Major number와 Minor number로 이루어져 있으며, (Major number).(Minor

number)의 형태로 표현한다. 문서의 버전은 0.1 부터 시작한다.

B. Version Management Policy

설계명세서를 수정할 때 마다 버전을 업데이트한다. 다만 변경 간의 간격이 1 시간 이내일 때에는 버전 번호를 업데이트하지 않고, 하나의 업데이트로 간주한다. 이미 완성된 파트를 변경할 때에는 Minor number를 변경하며, 새로운 부분을 추가하거나 문서의 구성이 예전에 비해 괄목할 변화가 있을 경우 Major number를 변경한다.

C. Version Update History

Version Modified Date Explanation

Version	Modified date	Explanation
0.1	2018.10.29	설계 명세서 문서 8개의 장에 대한 문서 목차 작성. Preface, Introduction, System Architecture 작성 및 완성.
1.0	2018.10.31	Sub System을 5개로 구분하여 각각을 문서 목차에 반영. Develop Plan, Database Design, Development Environment 초안 작성.
1.1	2018.11.05	Database Design, Develop Environment 수정사항 반영
2.0	2018.11.07	Sub System 작성 및 수정. Protocol Design 작성 완료.
3.0	2018.11.09	Database Design 수정 사항 반영. Sub System 추가 사항 반영.
3.1	2018.11.10	Testing Plan, Develop Plan, Appendices 작성 완료.

Table1 version update history

2. Introduction

2.1. Objective

본 문서에서 사용하는 다이어그램이나, 시스템을 설계할 때 필요한 툴들에 대해서 소개한다.

2.2 Applied Diagram

A. UML

UML이란 Unified Modeling Language의 약자로서 1997년 OMG에서 표준으로 채택한 통합 모델링 언어이다. 즉 모델을 만드는 표준언어라고 볼 수 있다. 모델이라는 것은 어떤 것을 실제로 만들 때 이렇게 만들면 잘 작동할지를 미리 검증해보는 과정이라고 할 수 있다. UML은 구조 다이어그램 7개, 행위 다이어그램 7개로 14종류의 다이어그램이 있다. 시스템의 개념, 관계 등의 측면에서 요소들을 나타내고, 각 요소들의 정적인 면을 보기 위한 것이고, 행위 다이어그램은 각 요소들 혹은 요소들간의 변화나 흐름, 주고받는 데이터 등의 동작을 보기 위한 것으로, 클래스 다이어그램은 구조 다이어그램에 속한다.

따라서 프로그램이 어떻게 작동하는지를 명확하게 표현하고, 개발자와 client간의 소통을 명확하게 함으로써 client의 요구사항에 어긋나지 않도록 유의해야한다. 따라서 UML은 다른 다이어그램들보다 시스템의 동작과 기능을 표현하기 쉽고, 가시적으로 표현이 잘되어 있다. 또한 개발자와 의뢰인이 아니더라도, 하나의 시스템을 만들기 위해 여러 팀들이 필요하듯이, 여러 팀들 사이에서의 충분한 소통 및 시스템 설계 과정의 표준화들이 필요하다. 그렇기 때문에 UML이 이러한 표준화 수단을 제공해준다.

UML은 객체 지향 설계를 위한 표준언어이고, 소프트웨어 개발의 산출물들을 시각화, 상세화, 구축, 문서화하는 특징을 가지고 있다. 이 때 시각화를 하는 데 있어 여러 상징적인 symbol들이 사용된다. 이 때 각 symbol들에 대한 명확한 정의가 필요하다. 그래야 개발자들 사이에 오해 없는 원활한 의사소통이 이루어질 수 있다. UML은 또한 C++, visual basic, java 등과 같은 다양한 프로그래밍 언어로 표현할 수 있다. 즉 이 말은 UML로 상세화된 설계모델은 프로그래밍 소스 코드로

변환하여 구현이 가능하다는 뜻이다. 반대로 코드로 되어 있는 것을 다시 UML로 표현하기가 쉬워지므로 역공학 (reverse engineering) 을 가능하게 해준다.

B. Package Diagram

클래스와 같은 여러 모델 요소들을 그룹화 하고, 패키지를 구성하여 패키지 사이들의 관계를 표현한 diagram이다. 패키지 라는 것은 class의 집합이라 볼 수 있는데, use case, acitivty diagram 들을 담을 수도 있다. 또한 다른 패키지도 패키지에 담길 수 있다. 이 때 패키지를 구성하고 구분하는 기준은 어러 사람이 동의할 수 있는 형태로 구성되어야 하고, 패키지의 구성과 이름, 체계는 개발자들이 쉽게 이해하고, 사용할 수 있게 해야한다.

이 때 같이 담기는 클래스들의 기준은 개념적, 관리적, 혹은 기능적 측면에서 유사한 면을 가져야 한다. 그렇기 때문에 다른 패키지에 있는 클래스들과는 약한 의존관계를 가진다.

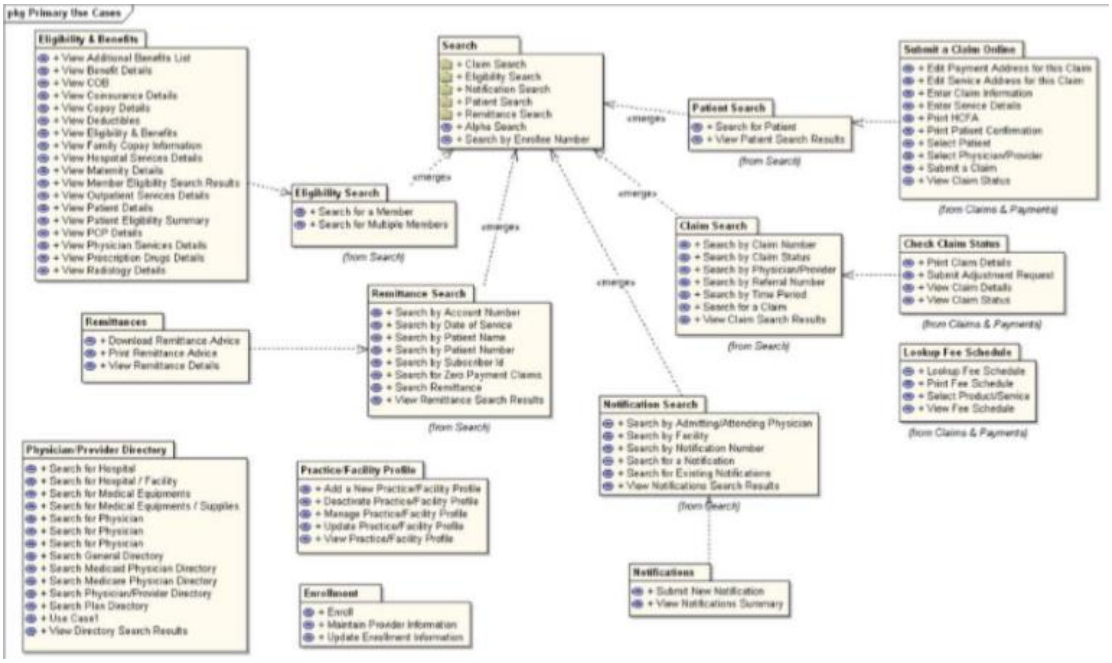


diagram1. package diagram

C. Deployment Diagram

Deployment diagram이란 그 프로그램의 물리적인 것들이 어떻게 배치되어있는지를 나타낸 diagram 이다. 물리적이라 함은 네트워크, 하드웨어, 소프트웨어의 실행파일 (.exe) 와 같은 것들을 말한다. 그래서 이렇게 시스템을 구성하는 하드웨어간의 연결관계를 표현하고, 리소스에 대한 소프트웨어 컴포넌트들의 배치 상태를 표현한 것이다.

Deployment diagram에서 직육면체의 의미는 한 노드를 의미하는데, 노드라는 것은 시스템에서 처리 능력을 가진 장치의 단위이다. 따라서 deployment diagram을 작성하기 위해서는 모든 설계가 마무리 되고 난 후에 진행된다. 왜냐하면 노드의 단위가 정의되고, 컴포넌트가 정의 되고, 하드웨어의 사양아 확정되기 때문이다.

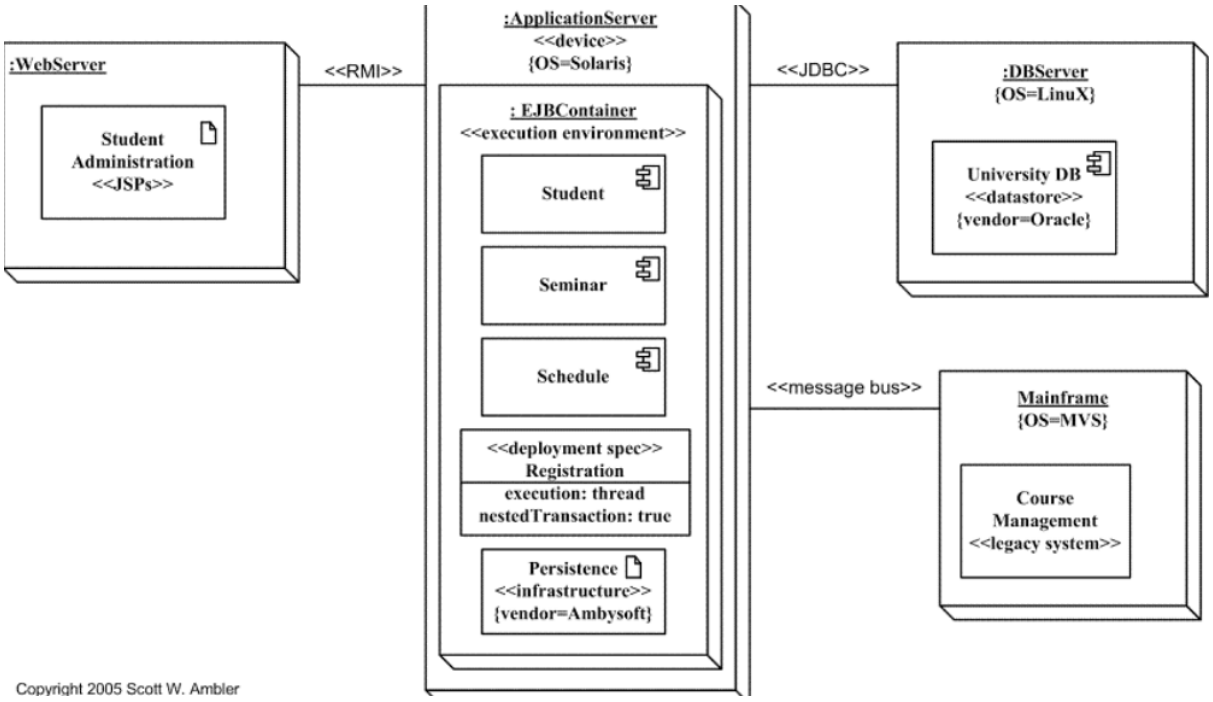


diagram2. deployment diagram

D. Class Diagram

Class diagram은 객체지향 설계에서 가장 널리 사용되는 다이어그램이다. 왜냐하면 class 라는 것이 객체지향 프로그래밍에서 가장 기본이 되는 단위가 되고, 그 class들을 기준으로 시스템을 표현하는 다이어그램이기 때문이다.

시스템의 정적인 상태인 논리적인 구조를 표현하는데, 그 논리적인 구조라는 것이 시스템의 상속, 연관 등의 관계를 symbol들을 통해서 명확하게 표현할 수 있다. 또한 이러한 class diagram을 통해서 코드생성의 직접적인 원인이 되기 때문에 프로그래밍 개념과 같은 의미의 표현을 통해 도식화 한다.

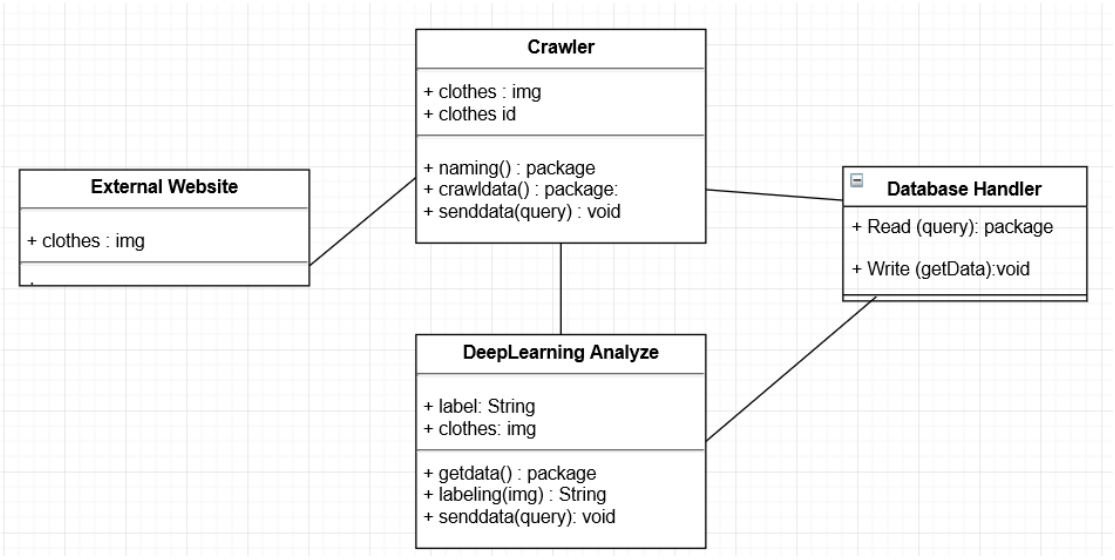


diagram3. class diagram

E. state diagram

State 다이어그램은 객체지향 모델에서 클래스의 인스턴스 (event)를 기반으로 하여 시스템의 전체적인 작동을 상세히 기술 하는 UML 다이어그램이다. State diagram은 상태의 변화에 의한 동작 또는 하나의 상태에서 다른 상태로 변하게 되는 모습을 나타낸다.

State diagram은 어떤 이벤트에 반응하는 객체를 기술하기 위해서 사용된다. 예를 들어 on/off 버튼을 누르는 이벤트가 발생했을 때, 전자레인지와 같은 객체가 켜지거나 꺼지는 것이 이러한 이벤트에 반응하여 상태가 변화한다고 볼 수 있고,

이렇게 변화하는 상태의 과정을 보이는 것이 state diagram이다. 따라서 state diagram에서는 시스템 내의 객체가 가질 수 잇는 상태가 어떤 것이 있는지에 대해, 또, 각 상태를 나타낼 때 특정 이벤트에 대해 어떤 반응을 보일지를 기술한다.

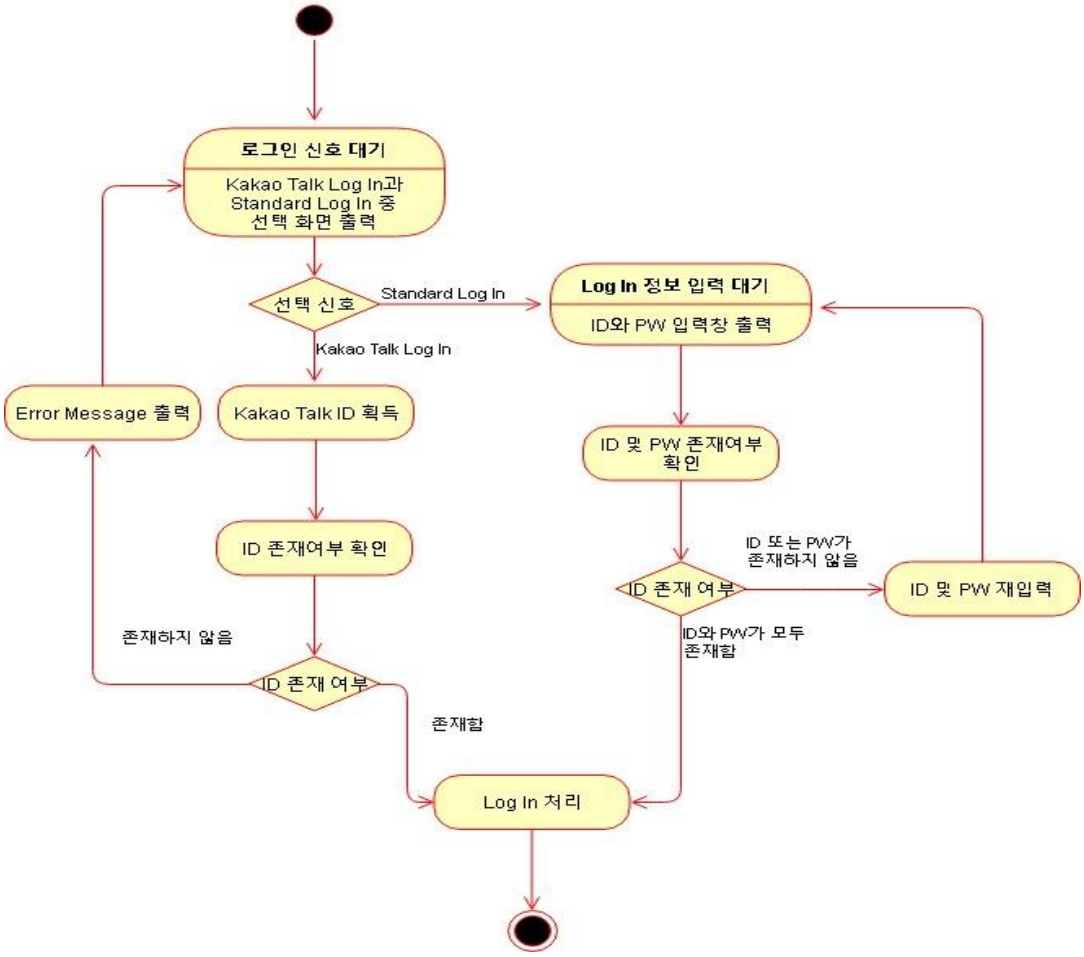


diagram4. state diagram

F . Sequence Diagram

시스템이 어떤식으로 작동되는지를 보여주는 측면에서는 activity 다이어그램과 비슷하다. 그렇지만, sequence diagram에서는 각 컴포넌트들이 주고 받는 메시지의 흐름을 표현하는데, 이 때 actor를 중심으로 재구성한 다이어그램이다. 그래서 actor에서 actor로 어떤 데이터 흐름이 발생하고, 메시지의 흐름이 발생하는지가 표현이 되는데 이것이 시간순차적으로 발생하는 것이다.

그리고 보통 use case diagram과 같이 표현된다. 이는 use case를 상세히 표현하는데 sequence diagram을 사용하는 것이 유리하기 때문이다. 왜냐하면 actor 즉

컴포넌트 별로 상호작용을 명확하게 보이기 때문에 각 컴포넌트들 간의 속성이나 행동들을 명확화할 수 있다. Sequential diagram에서는 객체의 동작과 속성을 상세히 정의한다. 이 과정에서 객체 간의 상호작용을 정의할 때, 각 객체들이 가져야하는 오퍼레이션과 속성을 구체적으로 정의된다.

Sequential diagram은 시스템의 동적인 측면을 모델링하기 위한 용도로 사용됙, 객체들과 그 사이의 교류 및 메시지를 가시화한다. 객체, 메시지, 회귀 메시지, 제어블록등으로 구성된다.

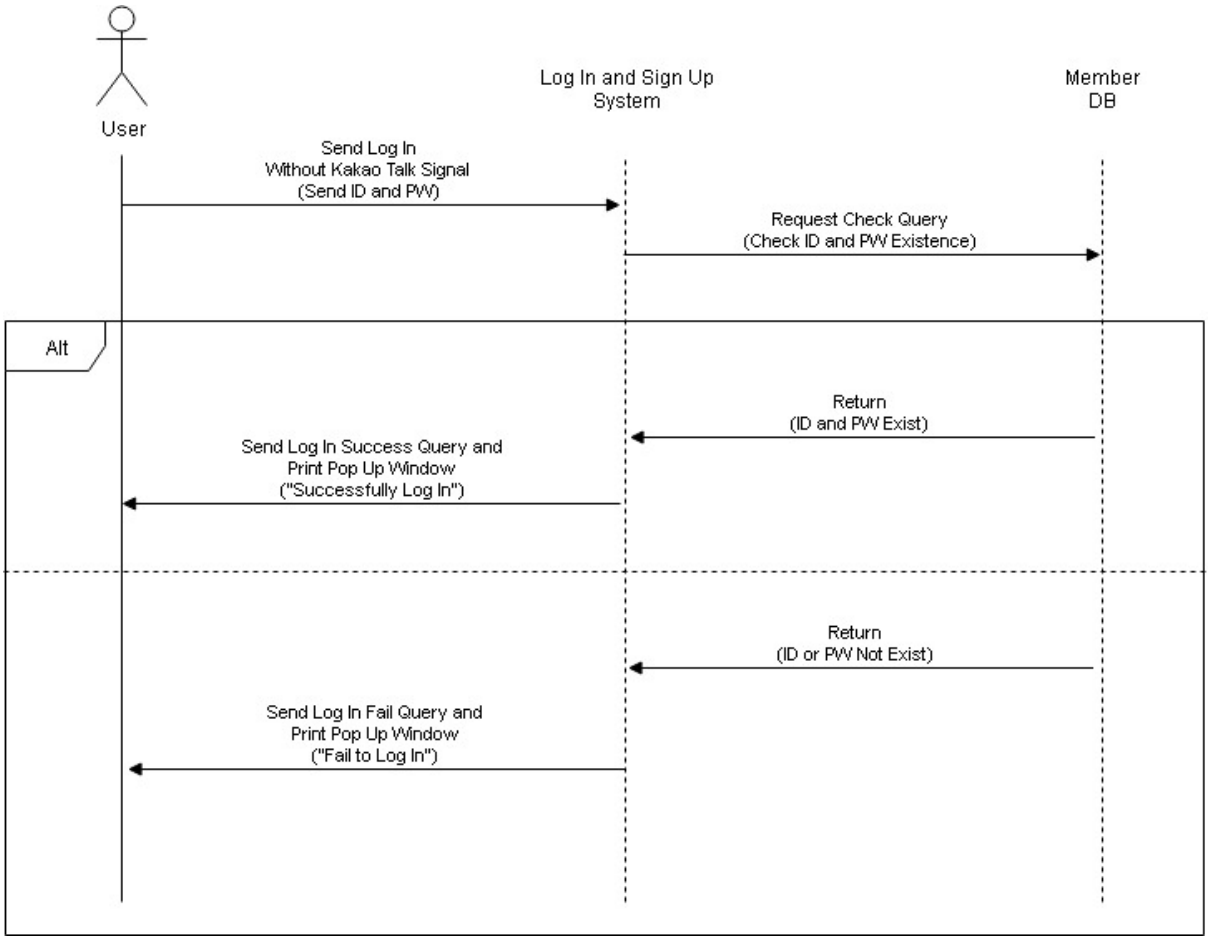
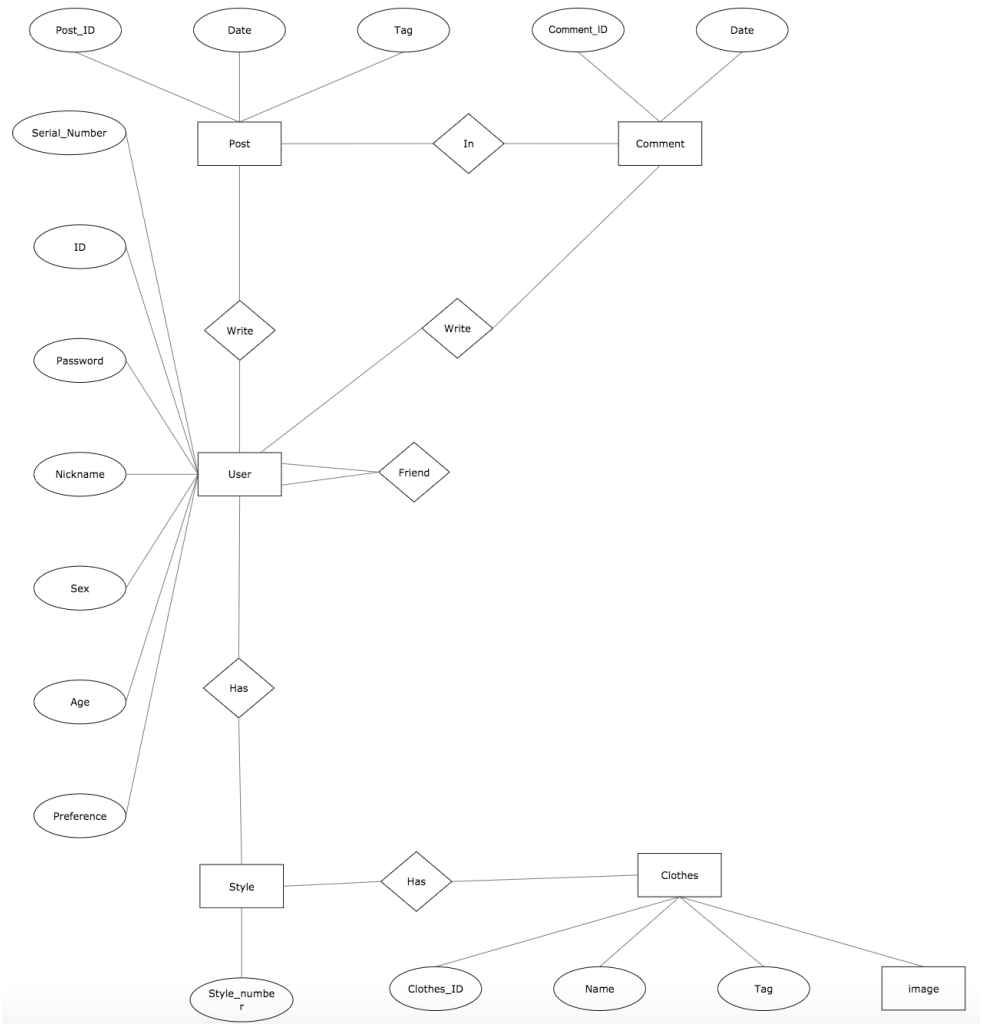


diagram5. Sequence diagram

G. ER diagram

데이터베이스에서 각 개체들의 관계를 표현하기 위해서 사용하는 다이어그램이다. 앞서 살펴본 다이어그램들과 같이 UML에서 정의되는 다이어그램은 아니다. 데이터 베이스에서 저장되는 데이터의 구조 및 관계를 표현해서 그릴 수 있는데 그러한 방법으로써, entity – relationship 이다. 그래서 ER 모델이라는 명칭이 붙게된 것이다.

ER 모델에서 Entity라는 것은 현실 세계의 객체로서 유형 또는 무형의 정보를 대상이다. 따라서 entity 이려면 속성을 가지고 있어야하고, 이러한 속성들을 가진 인스턴스가 발생될 수 있어야한다. 또한 entity의 종류에는 행위 entity 도 포함이 된다. Relation 이라는 것은 두 개이상의 entity 사이의 연관성을 기술 한 것이다.



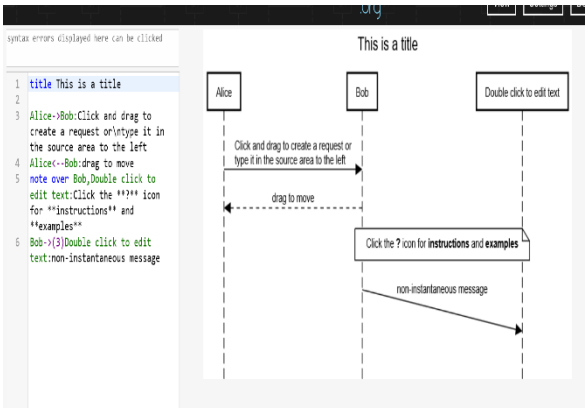
2.3 Applied Tool

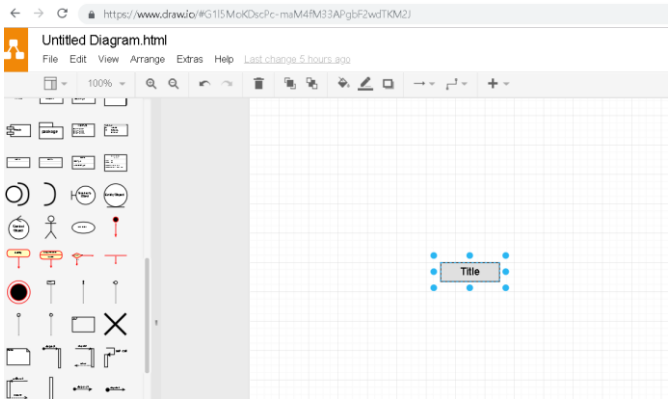
A. Diagram 표현

본 문서에서 표현하고 있는 다이어그램들은 "www. Draw.io " 라는 사이트에서 제공하는 그림 tool 들을 이용하고 있다. Class diagram이나 state 다이어그램들에 필요한 그림들을 어느정도 모듈화 되어 있어서 일일이 다시 그릴 필요 없이 내용적인 부분만 수정, 변경해주면 되므로 매우 편리하다. 또한 그린 다이어그램들은 자신이 등록한 구글 계정의 드라이브로 바로 연동이 되기 때문에 이미지를 수정, 생성하기가 매우 편리하다.

Sequential diagram 과 같은 경우에는 https://sequencediagram.org/ 의 사이트를 통해 작업하였다. 이것 또한 그림의 형식이 어느정도 정해져있어서 원하는 방향과 원하는 actor로 내용을 수정하기만 하면 되므로 쉽게 그림을 그릴 수 있도록 제공하기에 사용하였다.

ER diagram과 같은 경우에는 https://www.smartdraw.com/ 을 통해서 작업하였다.





B. Visual Studio code

본 프로젝트에서 딥러닝 분석을 진행하기 위해서 파이썬 프로그램을 사용하는데 이 때, 에디터로써 visual studio code를 사용한다. 디버그 뷰와 터미널 창을 같이 띄워서 실행 후 실행결과를 바로바로 확인할 수 있기 때문에 편리하다. 또한 아나콘다와 같은 프로그램에 비해서 가볍다. 그래서 딥러닝 분석하기에 적합한 에디터라고 생각하여 사용하게 되었다.

c. Firebase

firebase라는 플랫폼은 여러가지 API들을 패키지화하여 제공하기 때문에 본 프로젝트와 같이 개발할 시간이 부족할 때 더없이 적합한 플랫폼이다. 이것 자체가 서버의 역할도 하고 database도 연동이 되어 쉽게 개발을 진행할 수 있다.

2.4 Project Scope

중년들의 패션친구라는 프로그램은 일종의 패션 커뮤니티이다. "스타일 셰어"라는 어플리케이션처럼 user들이 자신들이 입었거나, 구매했던 옷들, 또는 관심있는 옷에 대해서 사진과 함께 글을 남기면 다른 user들이 그런 게시물들을 보고 정보를 얻어가거나 코디할 때 참고 가능하도록 해주는 어플리케이션이다.

그렇지만 "스타일 셰어" 를 비롯한 기존의 패션 커뮤니티들은 그냥 모든 게시물에 대한 정보들이 여과없이 다 게시된다. 남성 user의 경우에는 남자 스타일 코디에 관심이 많을텐데도 불구하고, 남자 user의 페이지에도 계속해서 여성 게시물만 많이 게시되었다. 즉 사용자의 취향이나 사용자의 특징을 고려하지 않았다.

그래서 "중년들의 패션친구" 어플리케이션은 user가 어떤 취향인지를 반영하여, 그 user와 비슷한 취향을 가진 다른 user를 추천해주고, 친구 사이가 되면 (상대방을 follow하게 되면) 상대방의 페이지에 가서 상대방의 스타일들을 참고해볼 수도 있고, 알림 탭에서는 자신의 친구들 중에서 자기가 관심있어 하는 분야의 게시물이 올라오면 볼 수 있도록 하였다.

또한 "중년들의 패션친구"에서는 그 focus가 중년에 있다. 따라서 중년들이 사용하기 쉽도록 스타일을 고르면 의사에 따라 자동으로 게시물을 업로드할 수 있다. 그리고 스타일을 고를 수 있도록 게시되는 사진들 또한 중년들에게 인기가 많은 인터넷 쇼핑몰의 사진들을 주로 사용하였다.

그래서 "중년들의 패션친구" 프로그램에서는 크게 User Management system, 취향분석 시스템, 친구관리 시스템, 게시물 관리 시스템, 크롤링 & 태킹 시스템 , 페이지 디스플레이 시스템으로 6가지의 sub system들로 이루어져있다.

1) User management system은 사용자가 처음 이용하기 위해 회원가입을 하거나, 회원 가입을 완료한 다음이라면 사용자가 로그인 하는 경우를 다루는 시스템이다. 그래서 그 하위의 기능으로서 카카오톡 계정을 이용한 회원가입과 카카오톡 계정이 없는 상태에서 회원가입, 그리고 로그인도 마찬가지로 카카오톡을 이용한 로그인과 그렇지 않은 로그인 이렇게 4가지

의 하위 기능들로 이루어져있다.

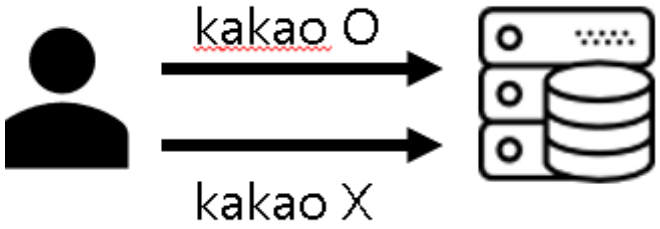
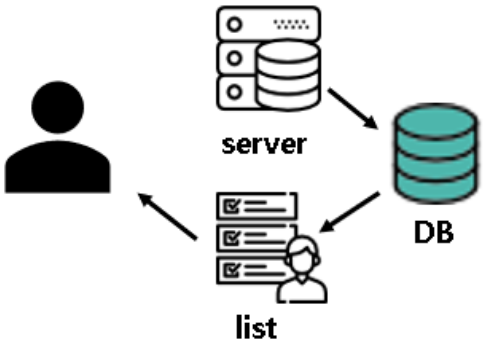
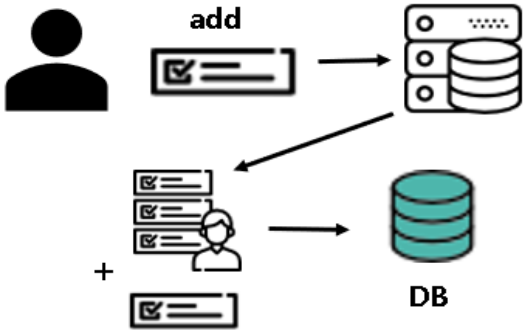


Figure 1. User management

2) Friend management system은 "중년들의 패션친구" 가 친구를 단위로 게시물을 보여주고 글을 남기면서 커뮤니티를 형성하기 때문에 이 때 친구와 관련된 작업들을 통칭하는 시스템이다. 따라서 그 하위 기능으로서는 친구를 추가하고, 추천된 친구들의 목록을 보여주고, 친구를 검색하는 3가지의 기능으로 이루어져있다.



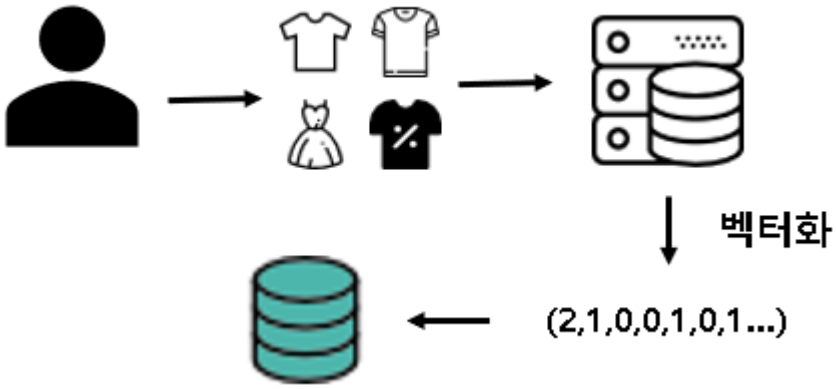


Search Friends Add Friends

3) 취향분석 시스템은 user가 어떤 취향을 가지고 있는지를 파악해보는 시스템이다. 그래서 이 시스템의 결과로서는 취향을 반영하는 vector가 산출된다. 이 시스템은 크게 2가지의 경우로 나뉜다.

취향을 반영하는 vector가 존재하는 경우와 존재하지 않은 경우로 나뉜다. 존재하지 않은 경우는 처음에 회원가입한 회원의 경우로써, 초반에 몇 장의 사진들을 제공하고, 자신의 취향에 맞는 사진들을 선택해서 취향벡터를 생성한다. 두 번째의 경우에는 이미 벡터가 생성되어 있는 경우이기

때문에 "스타일메이커"탭에서 고른 사진들을 기준으로 기존에 존재하는 취향벡터를 수정하게 된다.



4) 크롤링 & 태깅 시스템은 사진들을 외부 인터넷 쇼핑몰 사이트에서 가져오고, 그 사진들을 인식하여 우리가 가진 label로 분류해내는 작업을 통칭한다. 여기서 태깅 (tagging) 이라 함은 태그를 붙인다는 의미이다.

5) 페이지 디스플레이 시스템은 게시물들을 화면에 출력해내는 시스템이다. 페이지 구성상 크게 2가지가 있는데, 한 경우는 친구나 user의 페이지를 출력하는 경우이고, 다른 경우는 스타일 메이커 탭을 출력하는 경우이다.

친구나 user의 페이지는 자신들이 고른 옷들을 모아놓은 style 이나, 자신들이 지금까지 올린 게시물들을 볼 수 있다. 스타일 메이커라는 텝에서는 새로 크롤링되어 온 옷들이 게시가 되고, 사용자들은 그 탭에서 옷들을 고를 수 있다.

3. System Architecture

3.1 Objective

중년들의 패션친구의 전반적인 시스템에 대해 서술하고, 좀 더 세부적으로 그 구조에 대해서 서술한다.

3.2 System Oraganization

A. User Management System

User Management System은 사용자의 가입 정보 처리에 대한 시스템이다. 사용자의 회원가입에 대한 정보를 관리하는 Sign Up Subsystem과 로그인 기능을 시행하는 Log In Subsystem의 2개의 Subsystem으로 구성되어 있다.

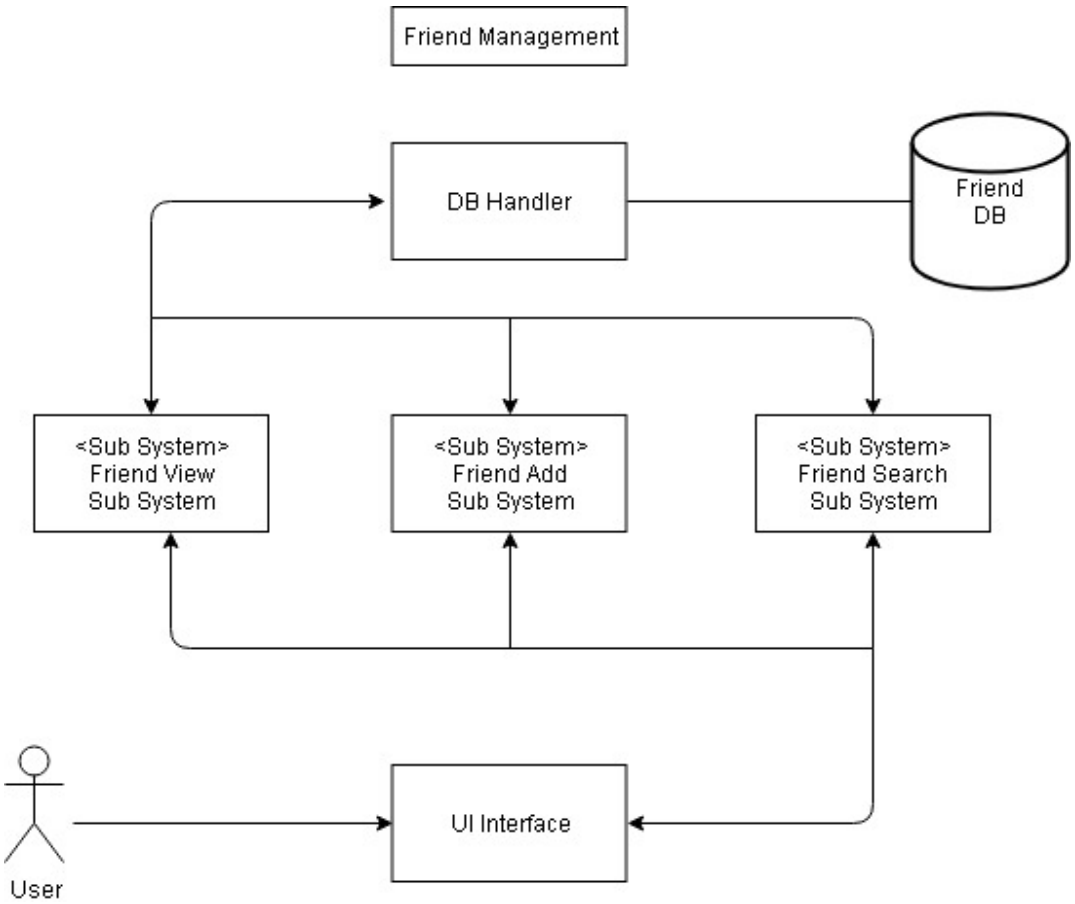


Diagram6 User Management System Architecture

B. Friends Management System

Friend Management System은 사용자의 Friend를 관리하는 시스템이다. 사용자의 Friend List를 보여주는 Friend View Subsystem과 사용자가 새로운 친구를 추가할 수 있게 해주는 Friend Add Subsystem, 그리고 사용자가 Nickname으로 새로운 Friend를 찾을 수 있게 해주는 Friend Search Subsystem의 3개의 Subsystem으로 구성되어 있다.

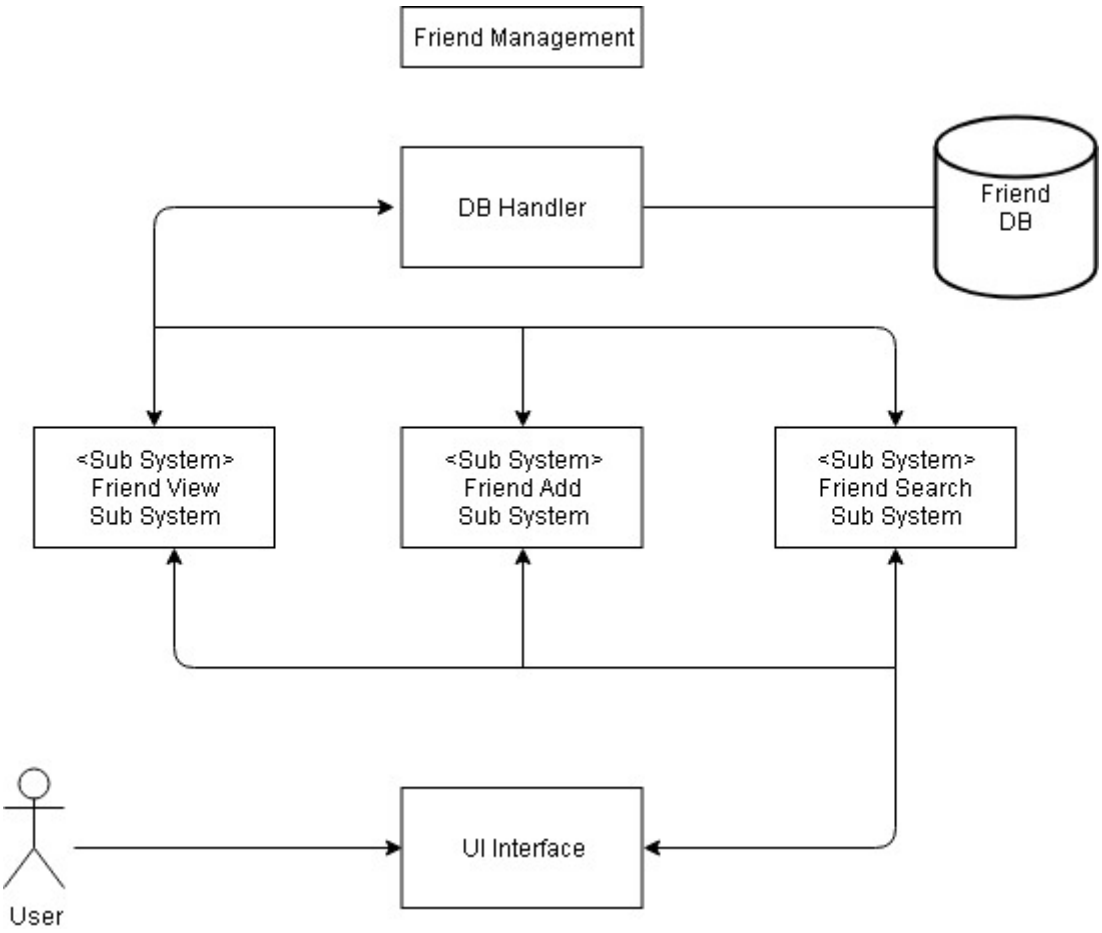


Diagram7 Friend Management System Architecture

C. 취향분석 시스템

취향분석 시스템은 user의 패션 취향을 분석해서 취향을 담고 있는 취향 벡터를 산출해내는 과정을 말한다. 취향분석 시스템의 경우에는 기존에 벡터가 존재하는 경우에, 벡터를 업데이트 시키는 기능과, 다른 user들과의 유사도를 계산하는 기능, 그리고 유사도를 바탕으로 친구를 추천해주는 기능으로 구성되어 있다.

이것은 취향을 파악하기 위해 그 정보들이 DB에 저장되는 그러한 구조를 보여준다.

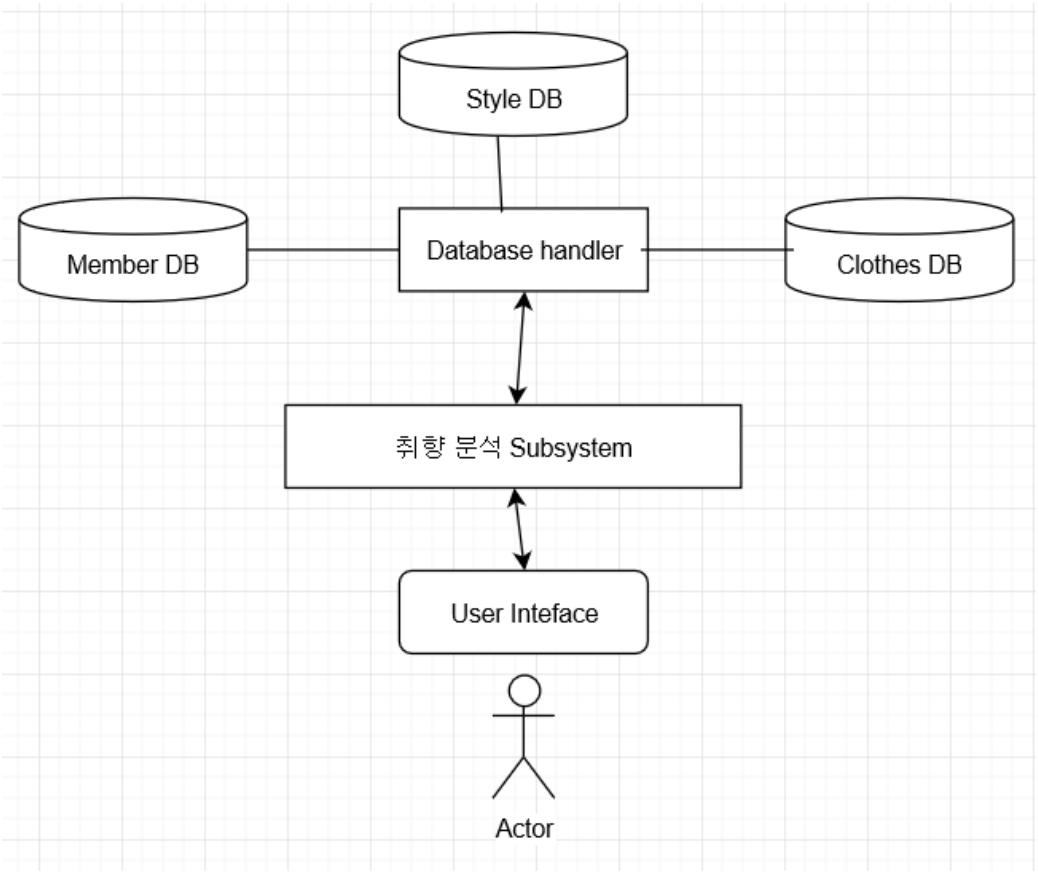


diagram 8 취향분석 system architecture

그리고 유사도를 기반으로 하여 친구를 추천해주는 구조를 뜻한다.

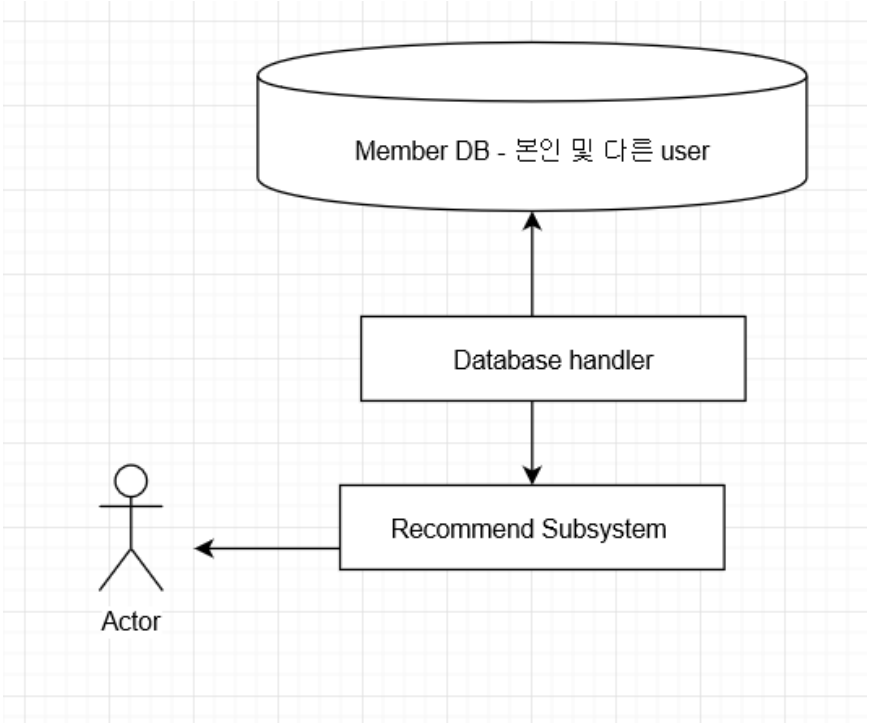


diagram9 친구추천 system archtecture

D. Crawling & Tagging system

외부 인터넷 쇼핑몰에서 게시되어 있는 사진들을 중년들의 패션친구 어플에서 가져오는 과정이다. 중년들의 패션친구에서는 이렇게 중년들을 대상으로 하는 쇼핑몰에서 게시되어 있는 사진들을 가져와야, 사용자들의 연령에 맞는 취향분석이 이루어질 수 있기 때문에 이 크롤링 작업이 필요하다

또한 단순히 사진만을 가지고 오면 그 사진에 대한 정보가 없기 때문에 그 사진에 대한 특징을 tag로 하여 그 tag를 붙이는 작업을 한다. 이 때 tag를 붙이는 작업은 딥러닝 모델을 통해서 옷을 인식해서 이 옷이 어떤 특징을 가지고 있는지가 분류된다.

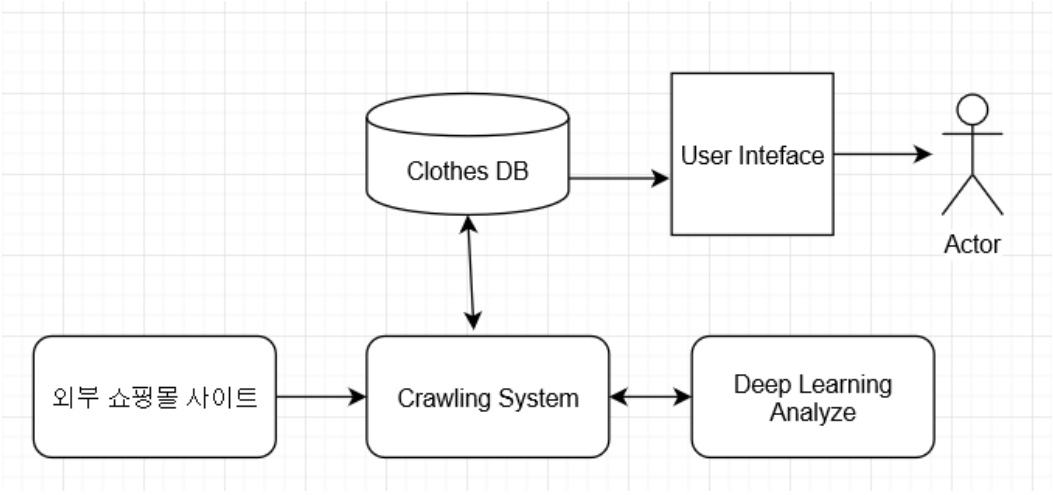


diagram10 크롤링 system archtecture

E. 게시물 관리 System

Post Management System의 경우, 실제 게시물의 내용을 관리하는 Content Management System과 게시물의 댓글을 관리하는 Comment Management System, 다시 하위 2개 시스템으로 나뉜다.

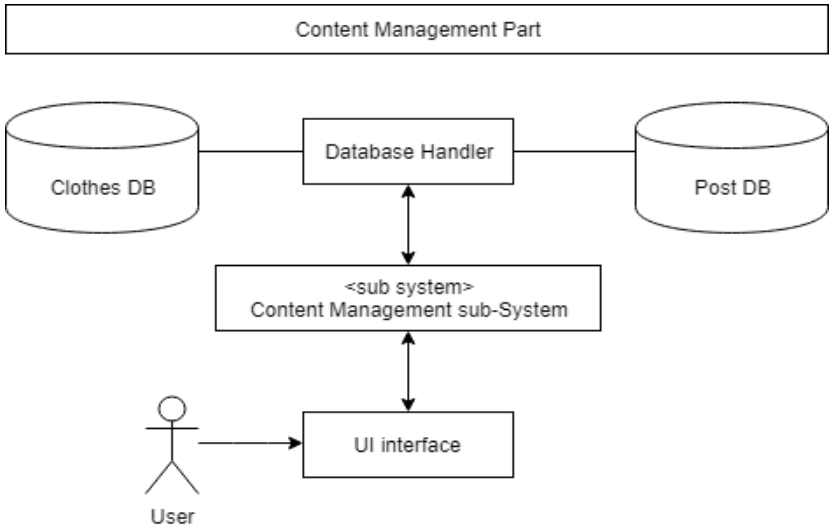


diagram11 content management system archtecture

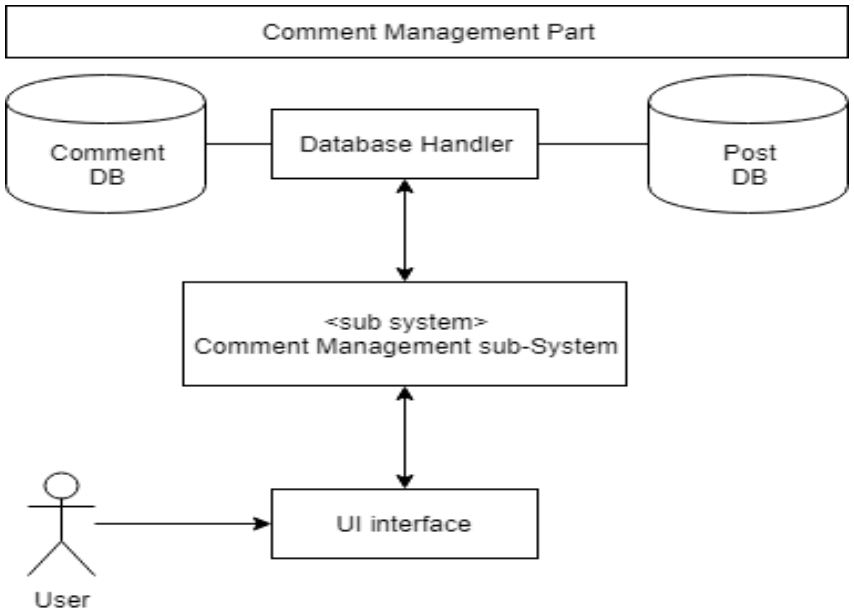


Diagram12 Comment Management system architecture

F. Extract preferred post system

사용자가 관심있어 하는 태그를 등록한다. 그리고 그 태그를 포함하고 있는 친구들의 게시물이 뜬다. 이 관심 tag를 기반으로 게시물이 display 되는 시스템을 말한다.

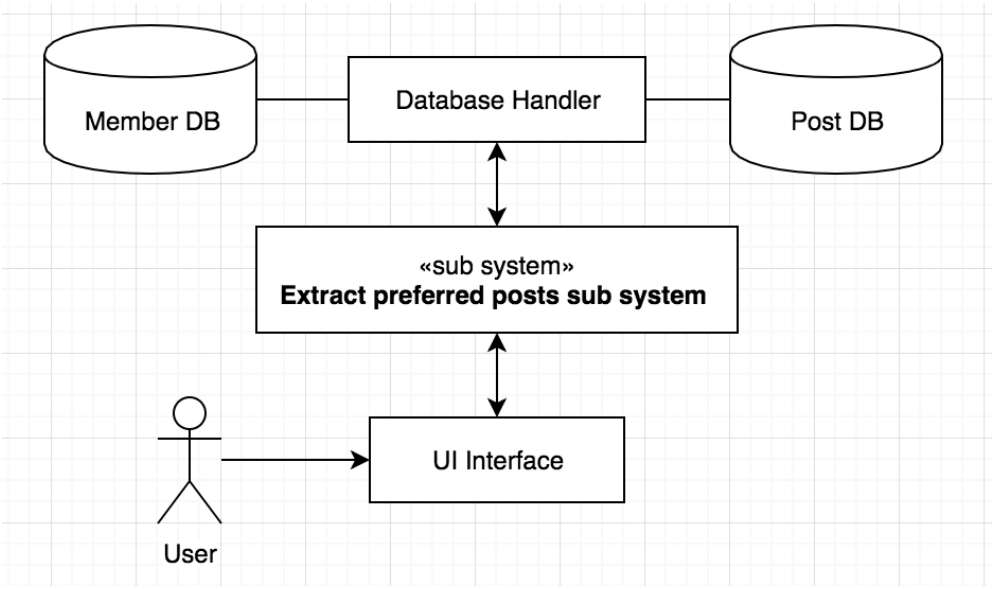


Diagram Extract preferred post system

G. Page Display System

Display System의 경우, 시스템 사용자에게 발생하는 Display에 관한 시스템으로, My page와 Friend page를 보여주는 경우(Case 1)와 Style Maker를 보여주는 경우(Case 2)가 있기 때문에, 그 쓰임 용도에 두 가지의 case를 가진다.

Case 1 Part

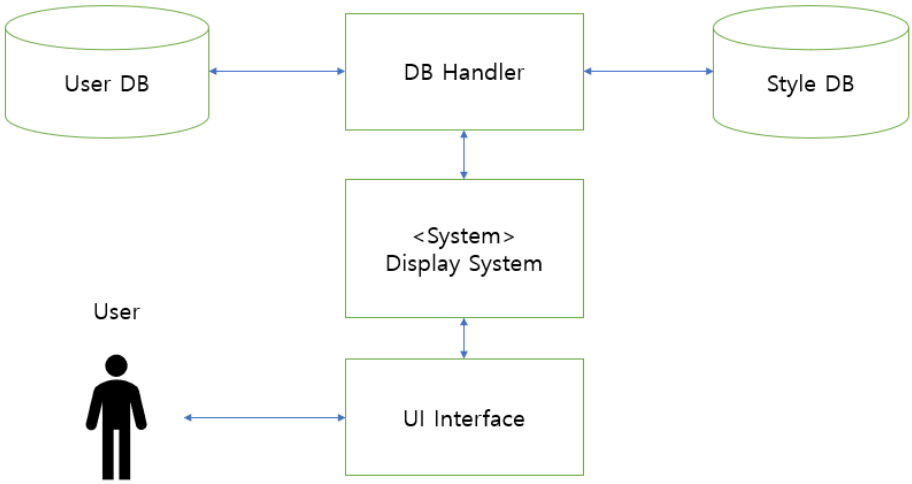


Diagram14 page display – case 1

Case 2 part

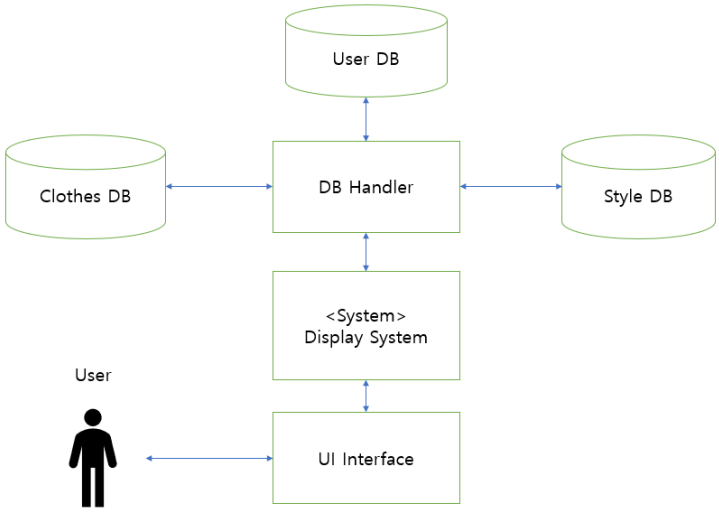


Diagram15 page-display-case2

4. User Management System

4.1 Objectives

회원 가입과 로그인 과정에서 발생하는 데이터의 처리를 진행하는 사용자 관리 시스템의 설계를 설명한다. Class diagram, Sequence diagram과 State Diagram을 통해 User Management System의 구조를 표현하고 설명한다.

4.2 Class Diagram

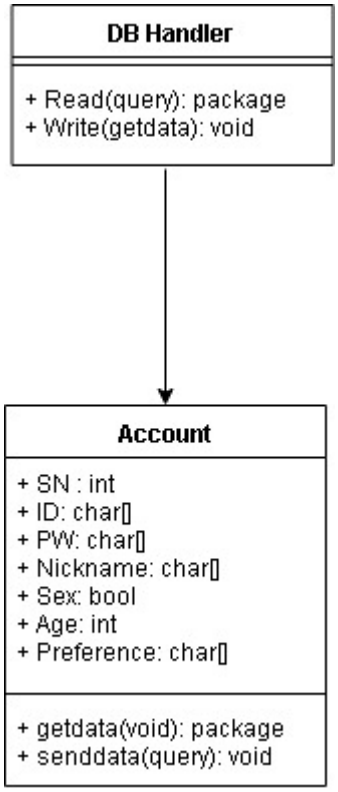


Diagram16 User Management system Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+package Read(query): 해당되는 DB에서 원하는 데이터를 읽어온다.
+void Write(package): 해당되는 DB에 데이터를 저장한다.

B. Account

B.1. Attributes

+ID: 계정의 ID
+PW: 계정의 Password
+Nickname: 계정의 닉네임
+Sex: 계정의 성별
+Age: 계정의 나이
+Preference: 계정의 취향정보

B.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.
+void senddata(package): 해당되는 DB에 데이터를 보낸다.

4.3 Sequence Diagram

A. Sign Up With Kakao

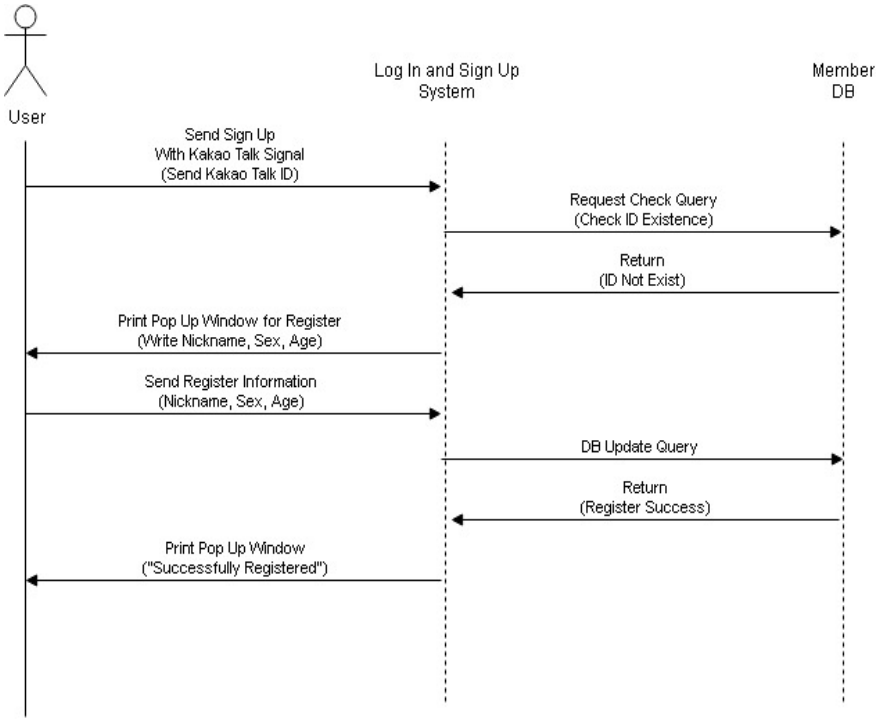


Diagram 17 User Management System Sequence Diagram 1

B. Sign Up Without Kakao

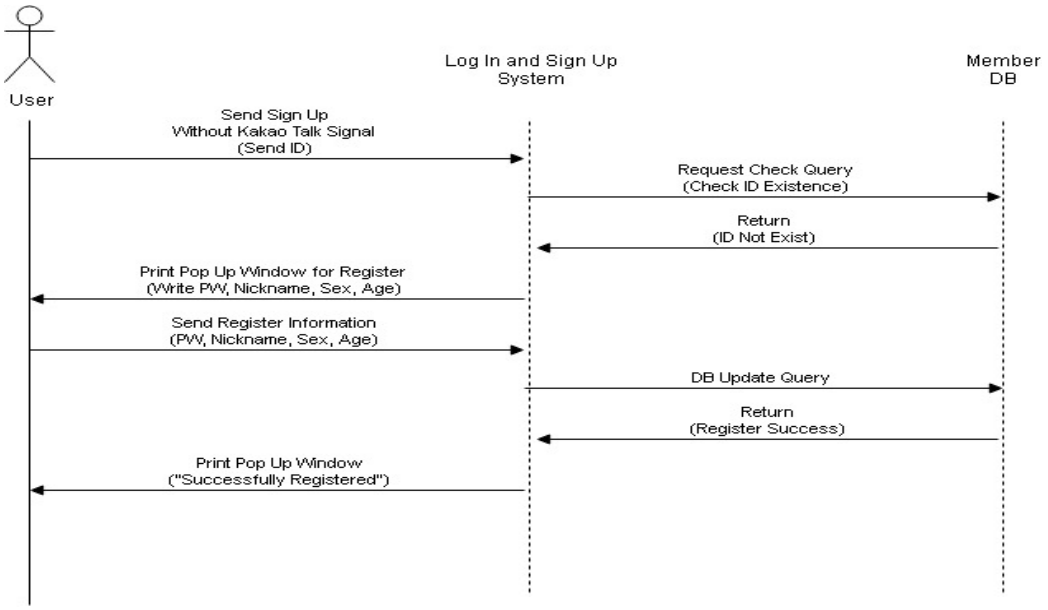


Diagram 18 User Management System Sequence Diagram 2

C. Log In With Kakao

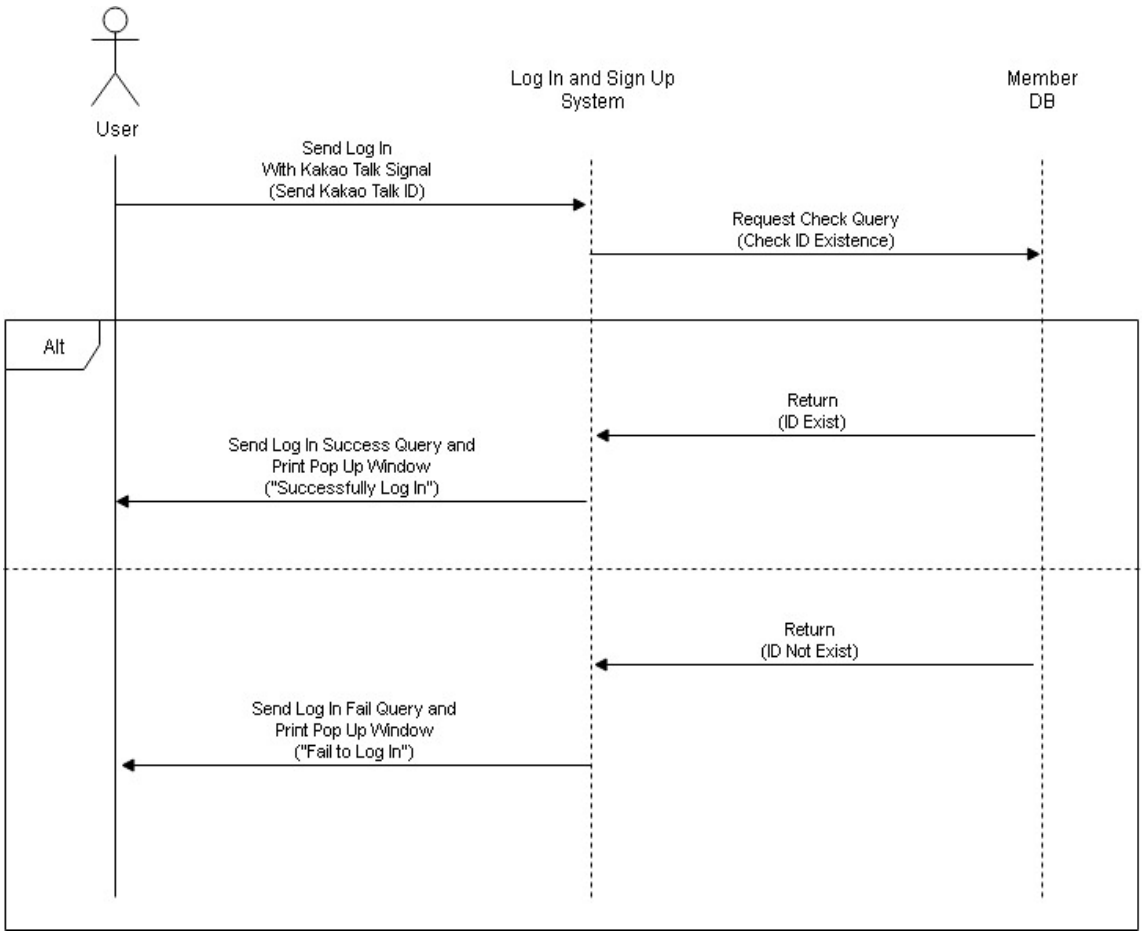


Diagram19 User Management System Sequence Diagram 3

D. Log In Without Kakao

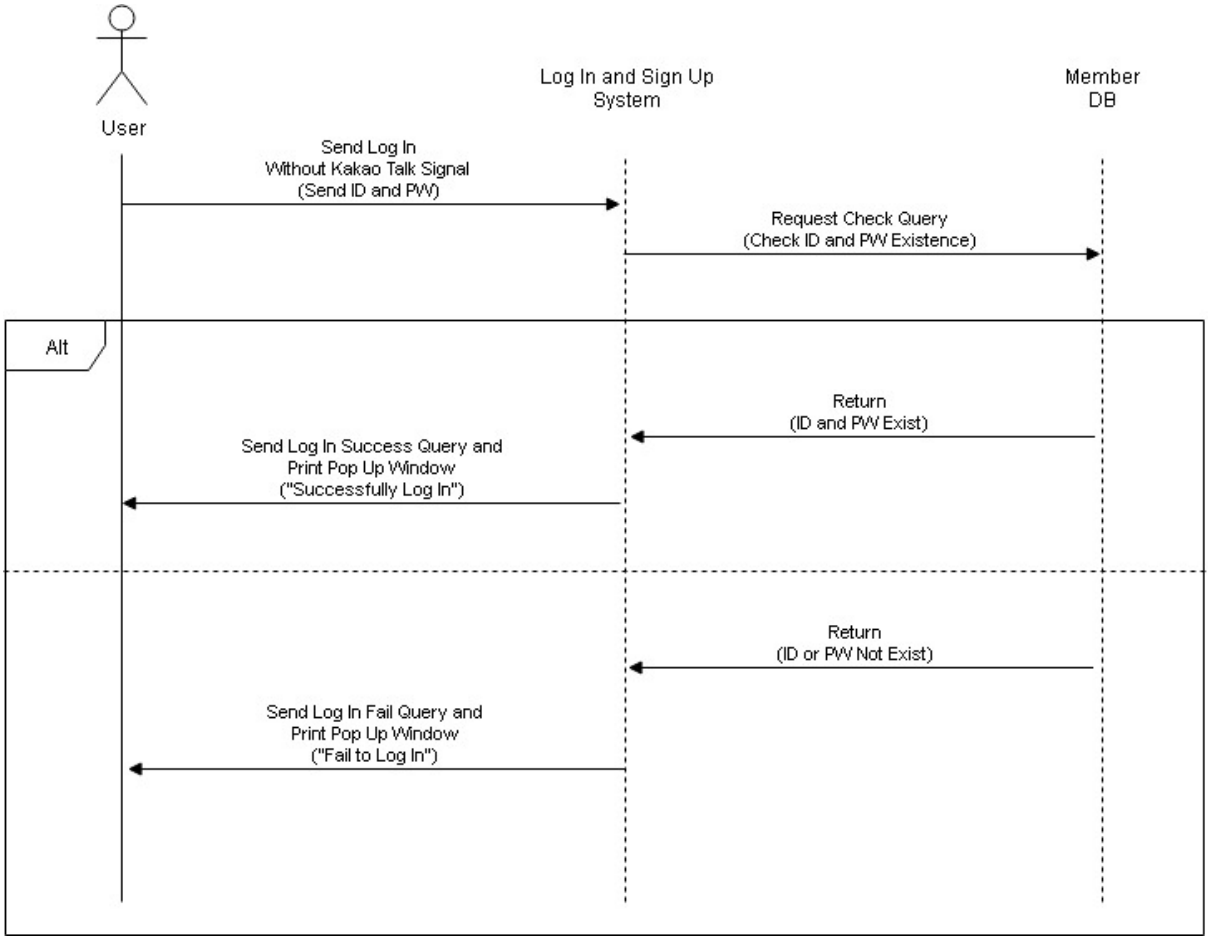


Diagram 20 User Management System Sequence Diagram 4

4.4 State Diagram

A. sign up

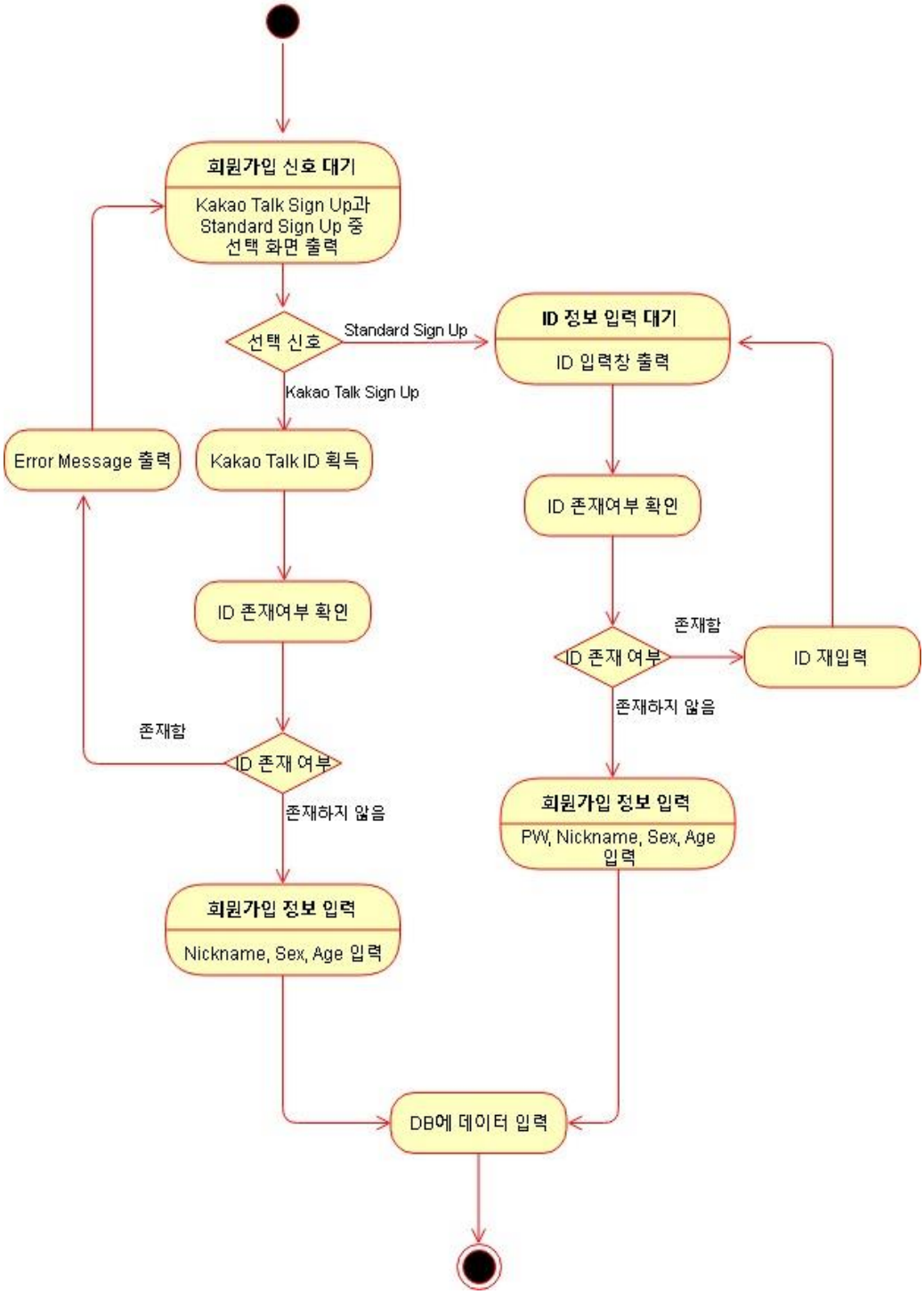


Diagram21 User Management System State Diagram 1

B. Log In

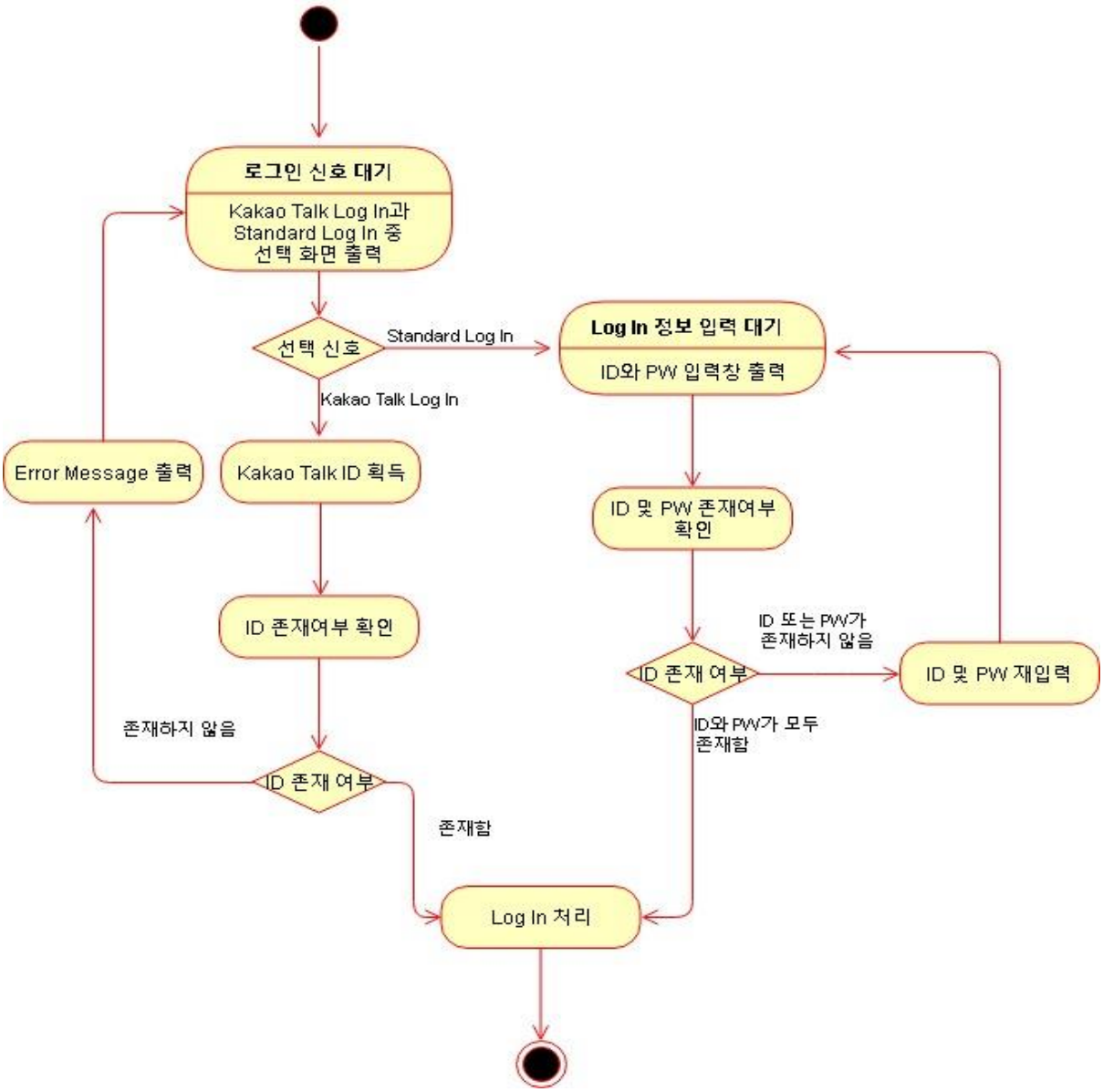


Diagram22 User Management System State Diagram 2

5. Friend Management System

5.1 Objectives

Friend List 표시, Friend의 추가, Nickname을 통한 Friend의 검색과 같은 기능을 수행하는 친구 관리 시스템의 설계를 설명한다. Class diagram, Sequence diagram과 State Diagram을 통해 User Management System의 구조를 표현하고 설명한다.

5.2 class diagrams

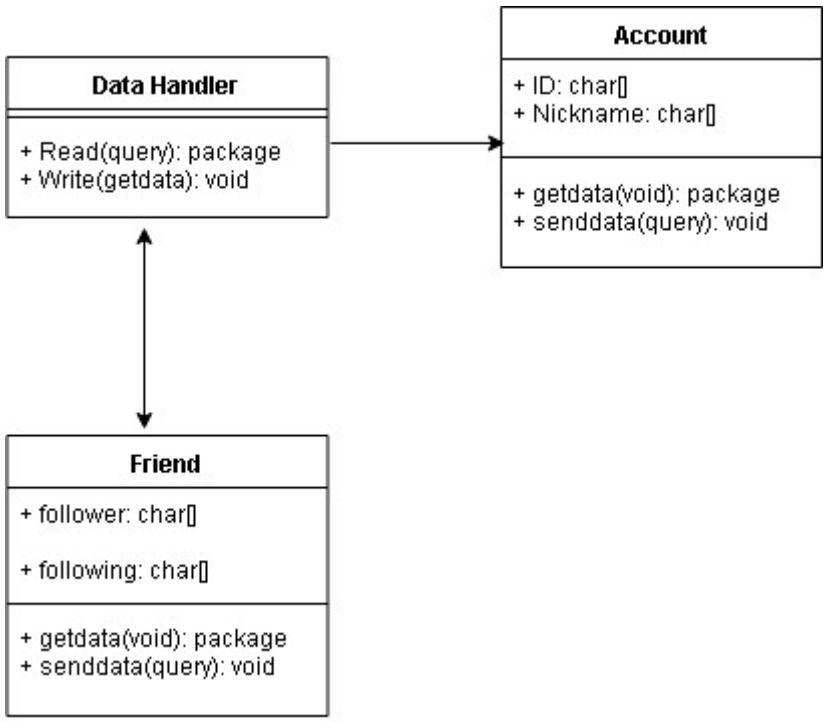


Diagram23 Friend Management System Class Diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+package Read(query): 해당되는 DB에서 원하는 데이터를 읽어온다.
+void Write(package): 해당되는 DB에 데이터를 저장한다.

B. Friend

B.1. Attributes

+follower: 친구로 등록한 주체가 되는 유저의 ID
+following: 친구로 등록된 유저의 ID

B.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.
+void senddata(package): 해당되는 DB에 데이터를 보낸다.

C. Account

C.1. Attributes

+ID: 계정의 ID
+Nickname: 계정의 닉네임

C.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.
+void senddata(package): 해당되는 DB에 데이터를 보낸다.

5.3 Sequence Diagrams

A . Friend View

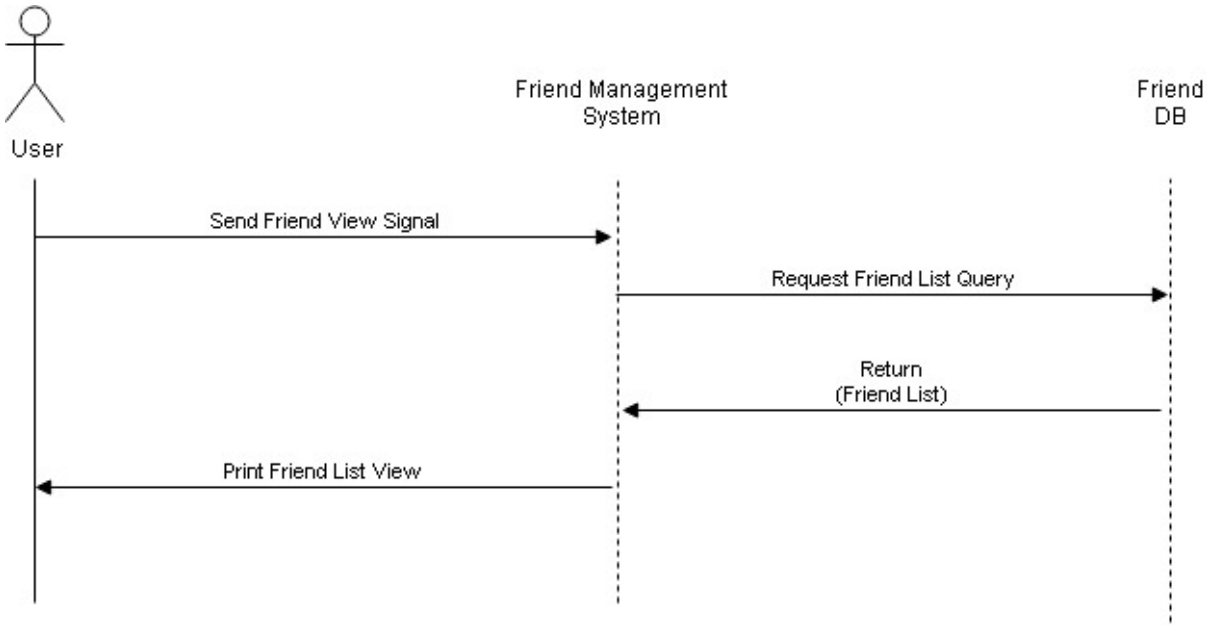


Diagram 24 Friend Management System Sequence Diagram 1

B. Friend Add

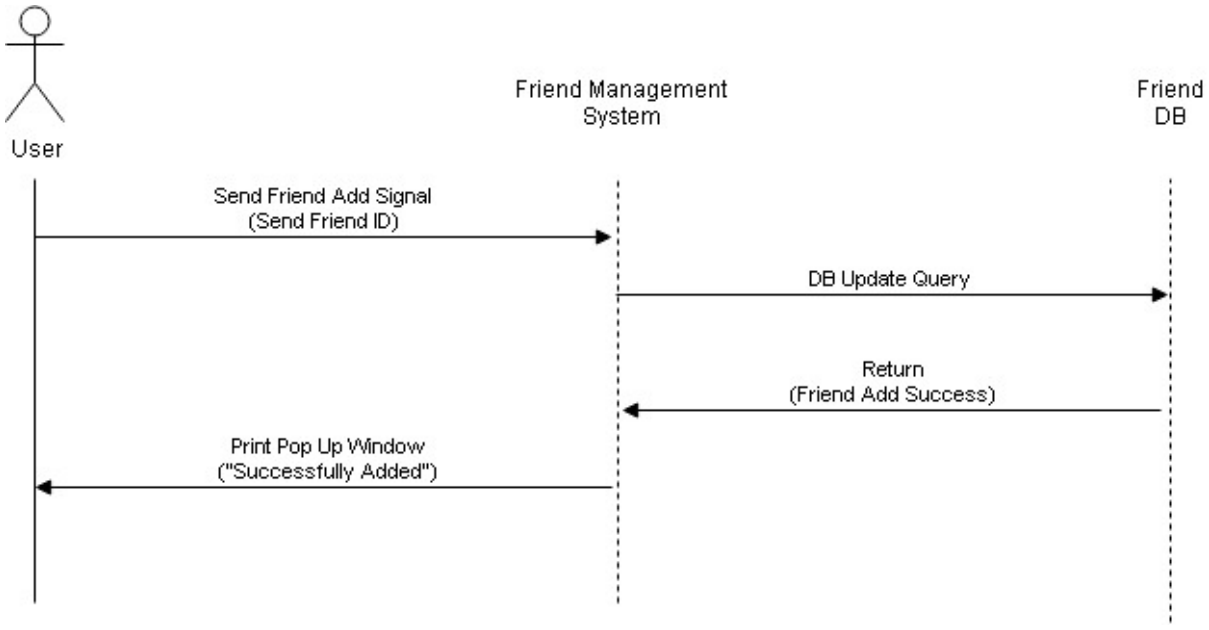


Diagram25 Friend Management System Sequence Diagram 2

C. Friend Search

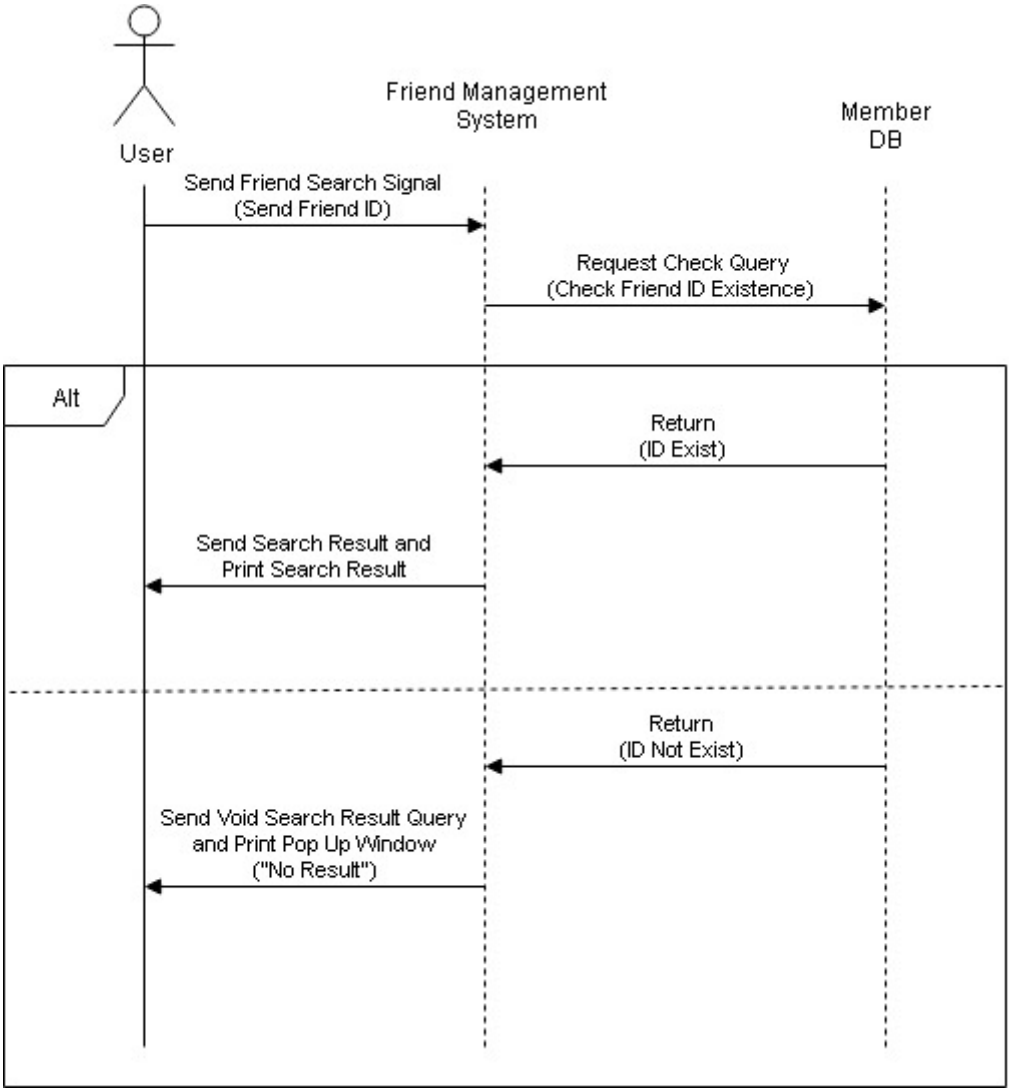
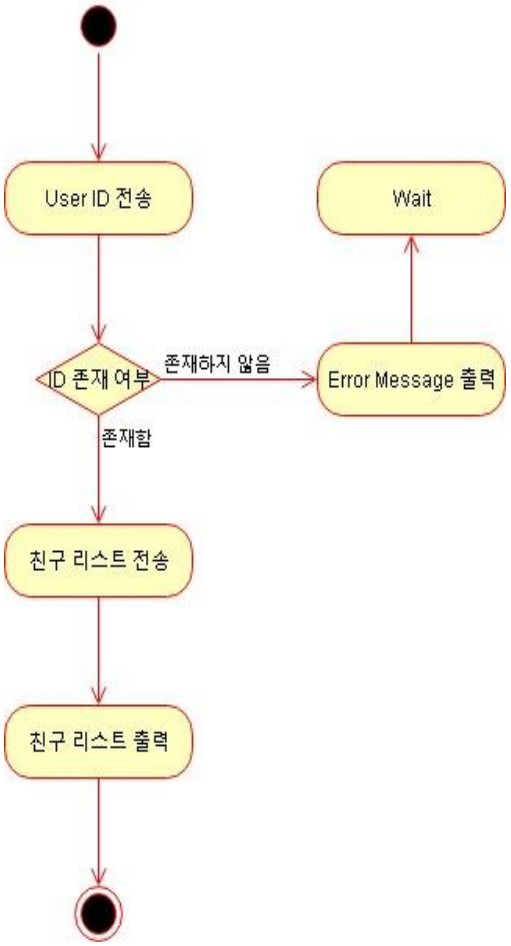


Diagram26 Friend Management System Sequence Diagram 3

5.3 State Diagram

A. Friend View



B. Friend Add

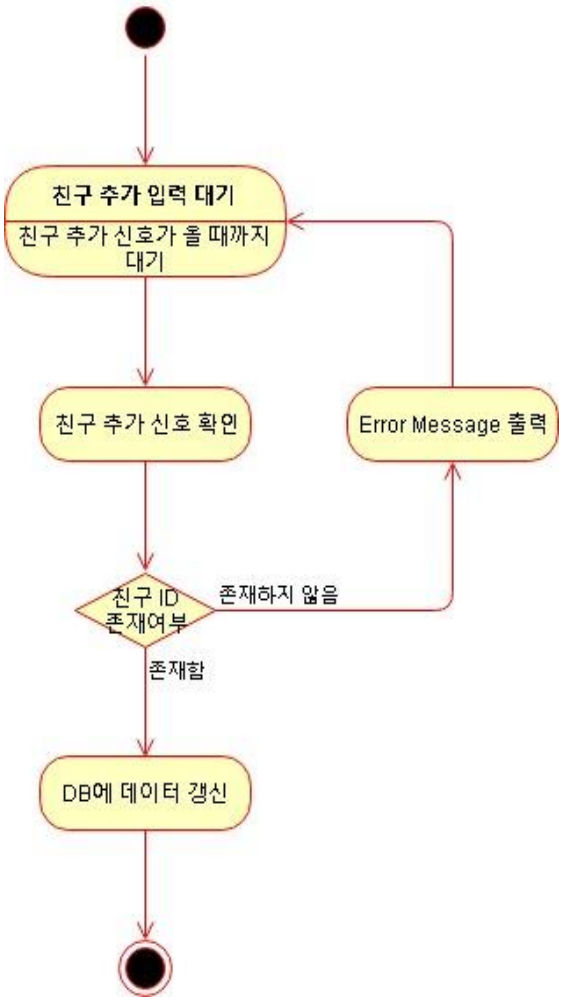


Diagram 27 Friend Management System State Diagram 1

Diagram 28 Friend Management System State Diagram 2

C. Friend Search

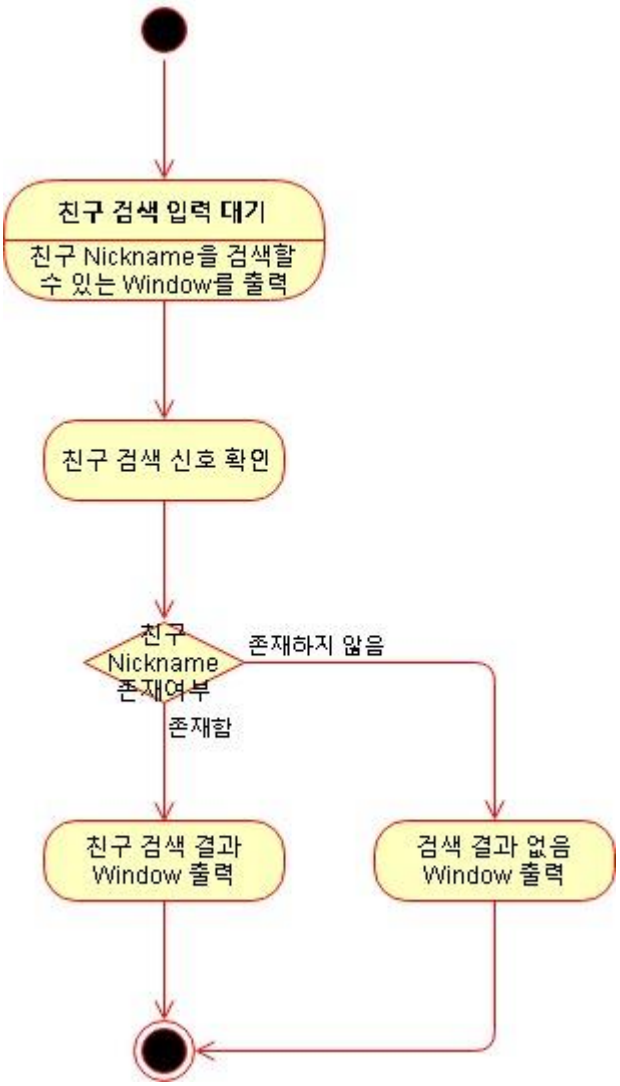


Diagram 29 Friend Management System State Diagram 3

6. 취향분석 시스템

1. Objectives

중년들의 패션친구라는 어플리케이션에서 핵심기능인 취향이 유사한 다른 user들 추천에 관한 과정을 설명한다. 취향분석 시스템의 경우에는 크게 2가지 경우로 나눠볼 수 있다. 취향 벡터를 처음에 생성하고 난 후, 업데이트 수정에 관한, 즉 취향 벡터를 관리하는 부분과 취향벡터를 기준으로 친구를 추천하는 부분이다. 이렇게 2가지 경우를 class diagram과 state diagram, sequential diagram으로 표현하였다.

1. Class diagram

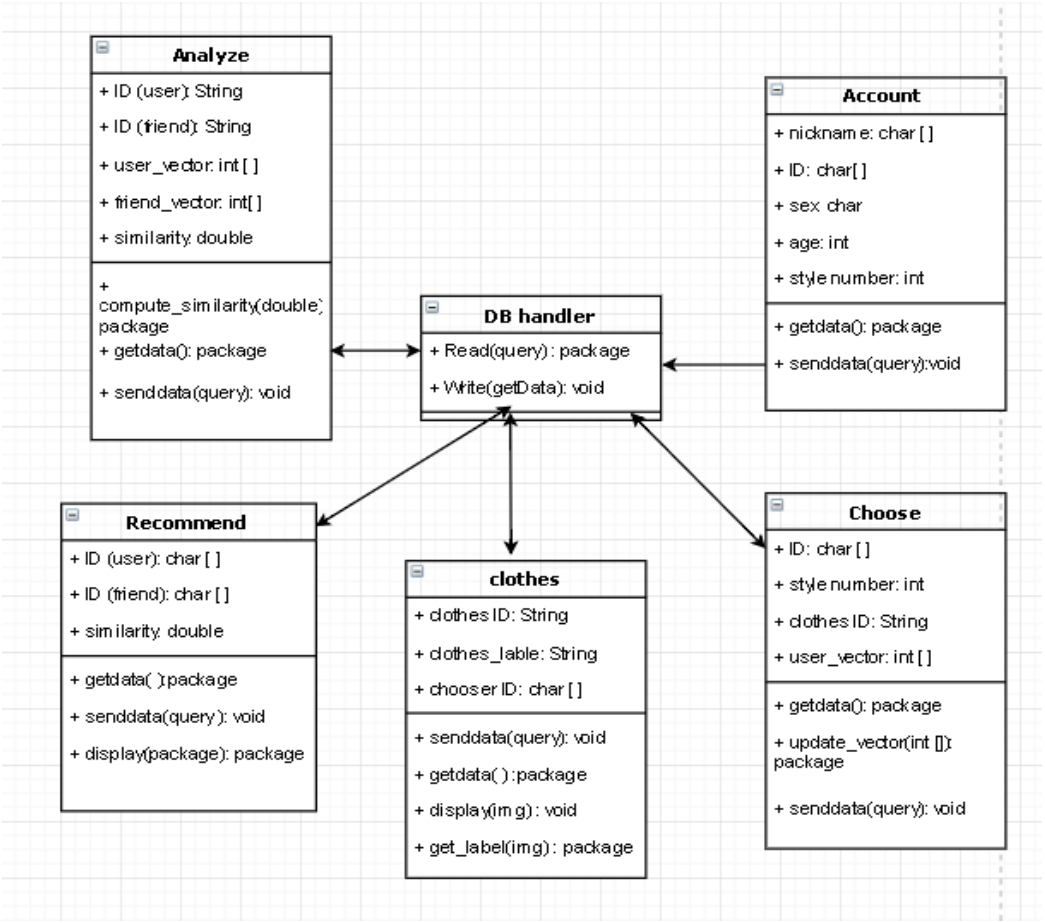


Diagram 30. 취향분석 class diagram

A. DB Handler

A .1 Attribute

해당 사항 없음

A .2 Methods

+package read (query) : 해당되는 DB에서 원하는 데이터를 읽어온다.

+ void write(data) : 해당되는 DB에 데이터를 저장한다.

B. Analyze

B.1 Attribute

+ ID (user) : 사용자 계정 ID
+ ID (other) : 다른 친구들 계정 ID
+ preference_vector : 사용자의 취향을 반영하는 벡터
+ other_vector : 다른 사용자의 취향을 반영하는 벡터
+ similarity : 사용자와 다른 사용자의 벡터의 유사도

B.2 Methods

+ package compute_similarity
  : 서로 다른 두 벡터 간의 유사도를 계산한다.
+ package getdata() : 해당되는 DB에서 정보를 가져온다.
+ void senddata (data) : 해당되는 DB에 정보를 보내준다.

C. Recommend

C.1 Attribute

+ ID (user) : 사용자 계정 ID
+ ID (friend) : 다른 사용자들의 계정

+ similarity : 사용자와 다른 사용자 벡터 간의 유사도

C.2 Methods

+ package getdata() : 해당되는 DB에서 데이터를 얻어온다.
+ void senddata(package) : 해당되는 DB에 데이터를 보낸다.
+ package display : 일정 유사도 이상의 사람들을 띄워준다.

D. Clothes

D.1 Attribute

+ clothes ID : 게시되는 해당 옷의 아이디
+ clothes_label : 해당 옷의 특징을 반영하는 label
+ chooser ID : 그 옷을 고른 사람들의 ID

D.2 Methods

+ void senddata (query) : 해당되는 DB에 데이터를 보낸다.
+ package getdata() : 해당되는 DB에서 데이터를 얻어온다.
+ package getlabel(img) : img의 label만을 가져온다.

E. Choose

E.1 Attribute

+ ID (user) : 사용자의 계정
+ Style Number : 그 사용자에게서 해당되는 style 번호
+ clothes ID : 선택하려는 옷의 ID
+ preference_vector : 해당 user의 취향벡터

E .2 Methods

+ package getdata():
+ void senddata(query)

+ update_vector

F. Account

F .1 Attribute

+ SN : 해당 user의 계정 고유번호
+ ID : 해당 user의 ID
+ nickname : 해당 계정의 닉네임
+ sex : 해당 user의 성별
+ age : user의 나이
+ preference_vector : user의 취향벡터

F .2 Methods

+ package getdata(): 해당되는 DB에서 데이터를 얻어온다.
+ void senddata(query): 해당되는 DB에 데이터를 보낸다.

6.2 Sequence Diagram

3.1) 취향 분석

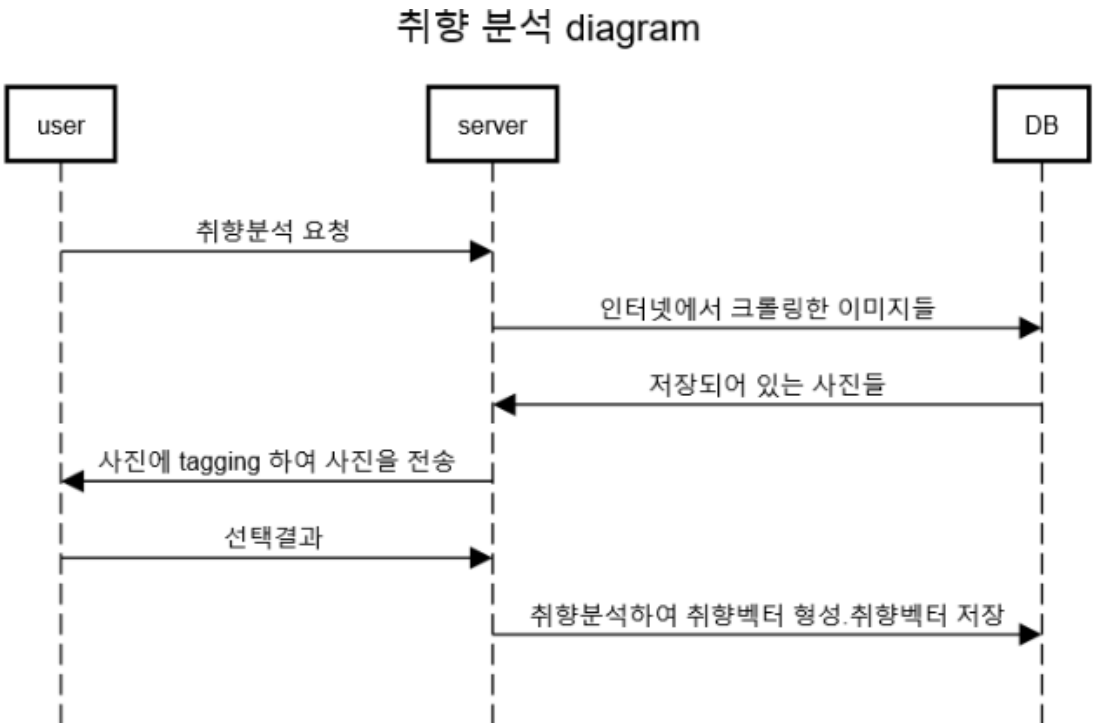


Diagram 31 취향분석 sequence diagram

3.2) 취향 업데이트 (view new clothes를 통한)

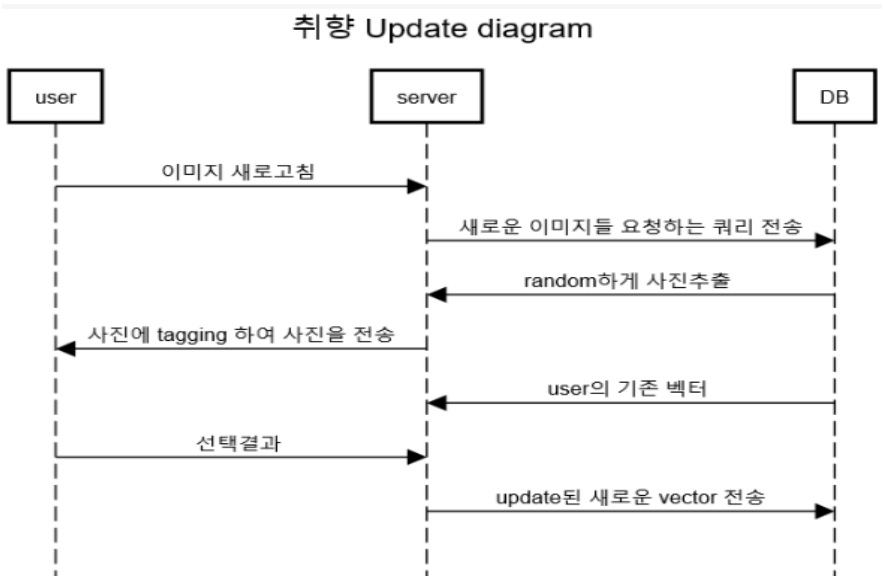


Diagram 32. 취향 update sequence diagram

3.3) 친구 추천

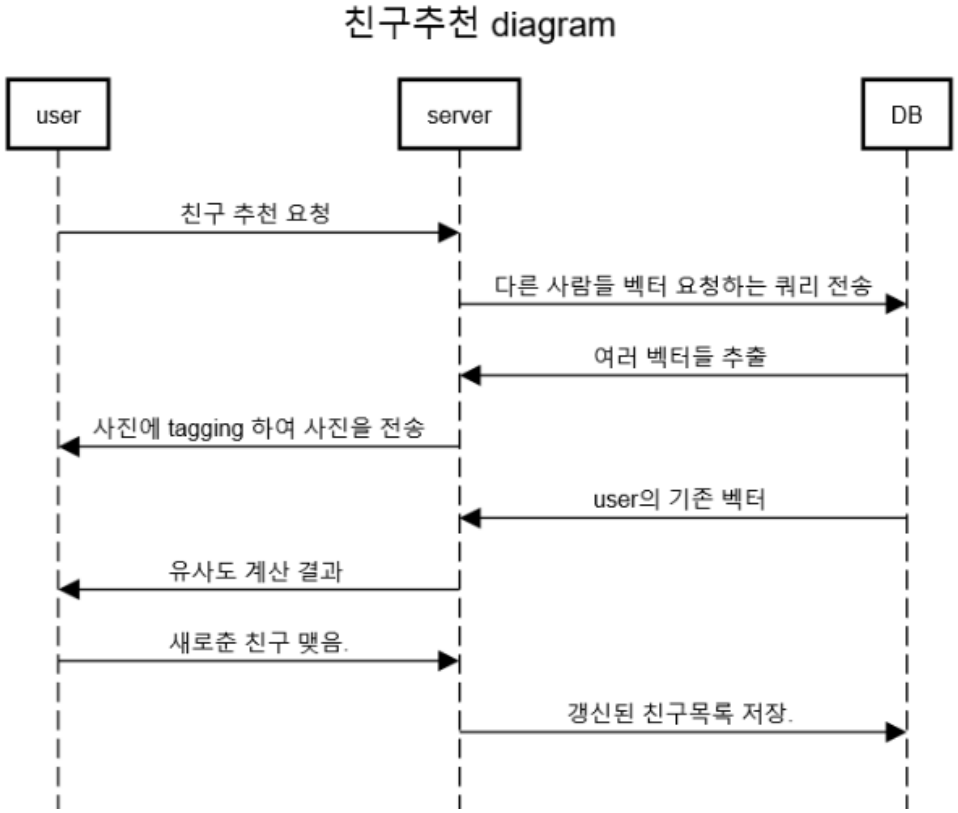


Diagram 33. 친구추천 sequence diagram

6.3 State Diagram

취향도출 및 업데이트

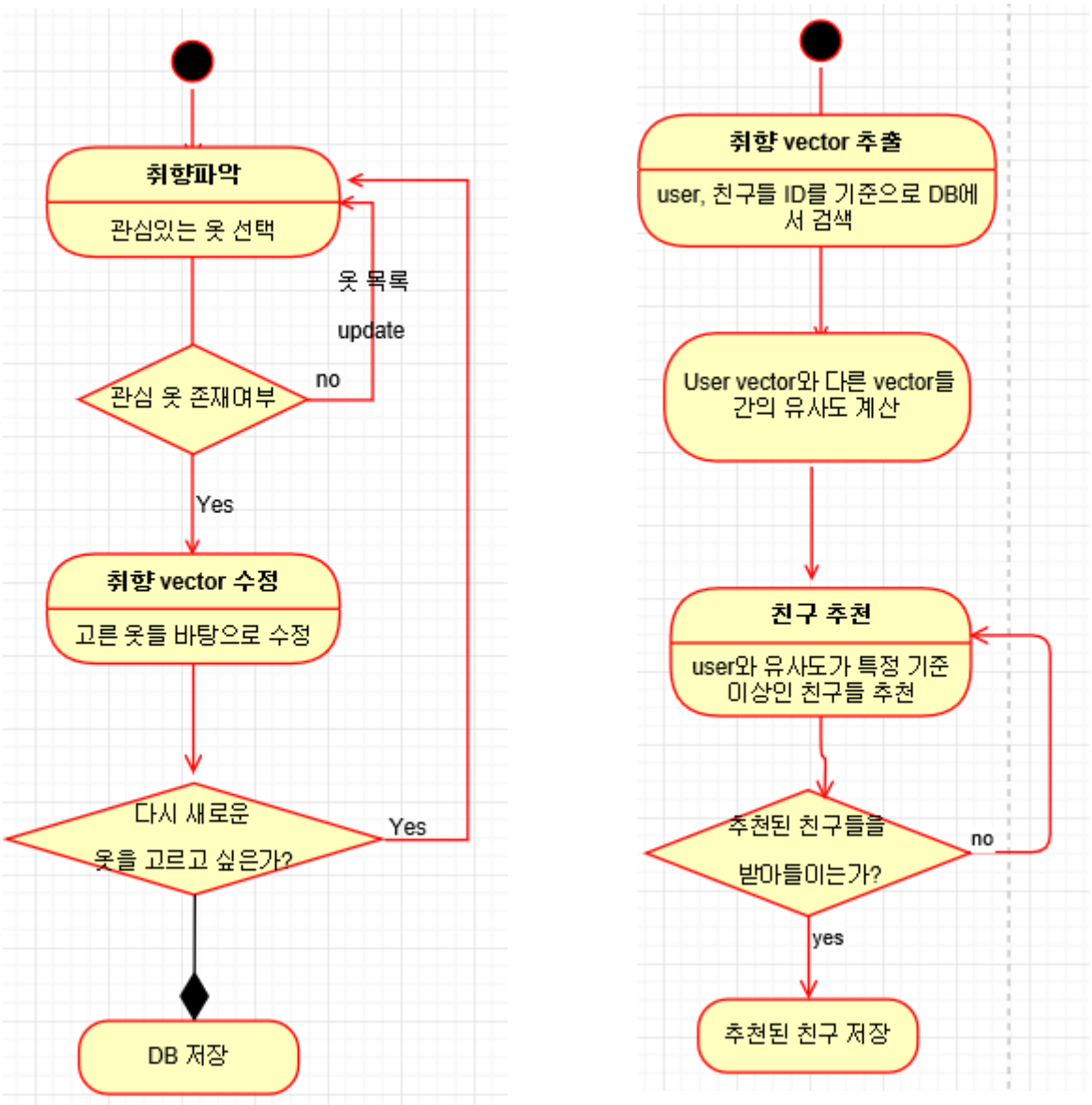


diagram 34. 취향도출 및 update state diagram

7. Crawling and Tagging System

7.1 Objectives

중년들의 패션친구에서는 중년들에게 맞춤인 옷을 제공하기 위해서 "꽃중년", "마담 4060" 과 같은 사이트에서 제공되는 옷 이미지를 크롤링하여 가지고 온다. 중년들의 옷 취향분석과 새로운 옷을 고를 때 이미지를 제공하기 위해서 crawling system이 그 일을 한다.

그리고 이 때 외부에서 가져오는 사진들은 이 옷이 어떤 옷인지에 대한 정보 즉 label이 없다. 그 옷의 종류와 스타일의 정보를 파악하기 위해서 DeepLearning analyze 시스템에서 그 옷에 대해 분석을 진행한다. 그 결과 그 옷의 종류 및 여러 특징들을 라벨화된다. 이 단계에서는 이러한 특징들이 label이라고 사용하였는데, 나중 단계에서는 이러한 라벨들이 검색의 기준이 되고, SNS적으로 친숙한 용어인 tag를 사용하였다.

7.2 Class Diagram

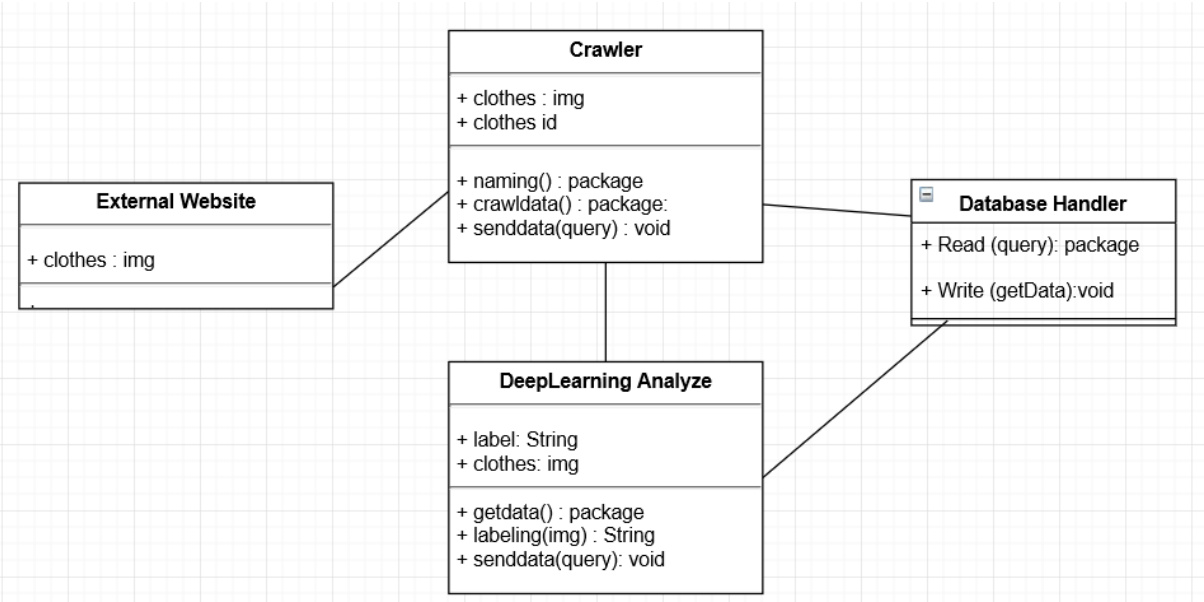


Diagram 35 crawling and tagging class diagram

A. Crawler

A .1 Attribute

+ clothes : 크롤링의 대상이 되는 옷 이미지
+ clothes ID : 해당 옷 이미지의 아이디 (옷 식별자)

A .2 Methods

+ package naming() : 해당 옷의 ID를 부여하는 작업
+ package crawldata(): 외부 사이트로부터 옷 데이터를 가져오는 작업
+ void senddata(query) : 해당 DB로 데이터를 전송

B. External Website

B .1 Attribute

+ clothes : 외부 사이트에서 게시되어 있는 옷 이미지들

B .2 Methods

해당 사항 없음

C. DeepLearning Analyze

C .1 Attribute

+ label : 딥러닝 분석의 결과로 산출되는 해당 옷의 라벨
+ clothes : 분석 시에 input으로 사용되는 옷 이미지
+ clothes ID : 해당 옷의 ID

C .2 Methods

+ package getdata () : 해당 DB로부터 옷을 받아옴
+ package labeling (img) : 이미지를 분석하여 라벨을 해당 이미지에 부여하는 과정

D. Database Handler

D .1 Attribute

해당 사항 없음

D .2 Methods

+ package Read(query) : 해당되는 DB에서 원하는 데이터를 읽어 온다.

+ void Write(getData) : 해당되는 DB에 데이터를 저장한다.

7.3 Sequential Diagram

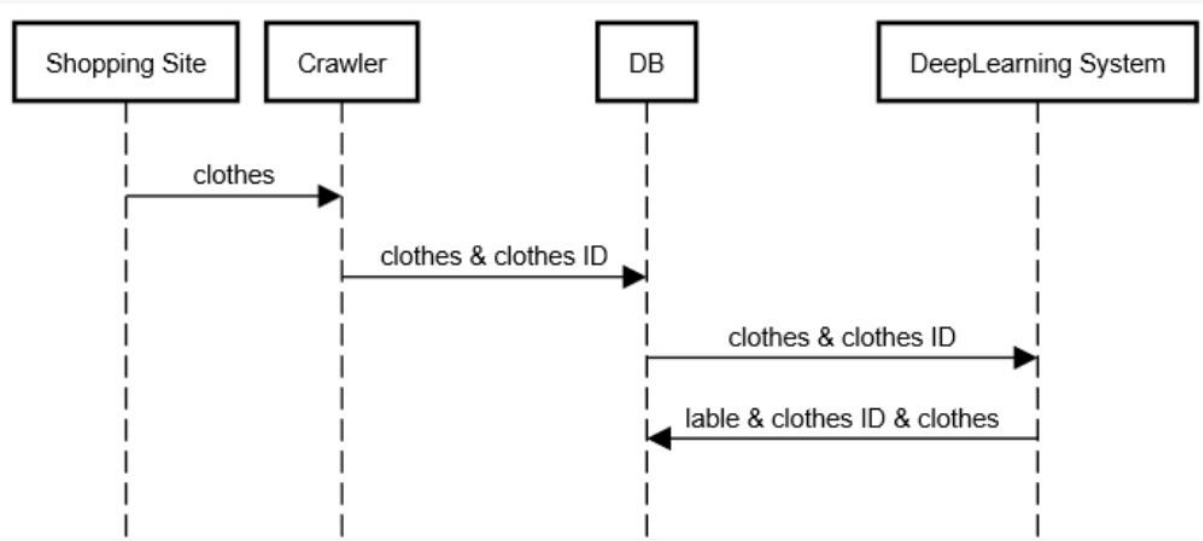


Diagram 36 crawling and tagging sequence diagram

7.4 State Diagram

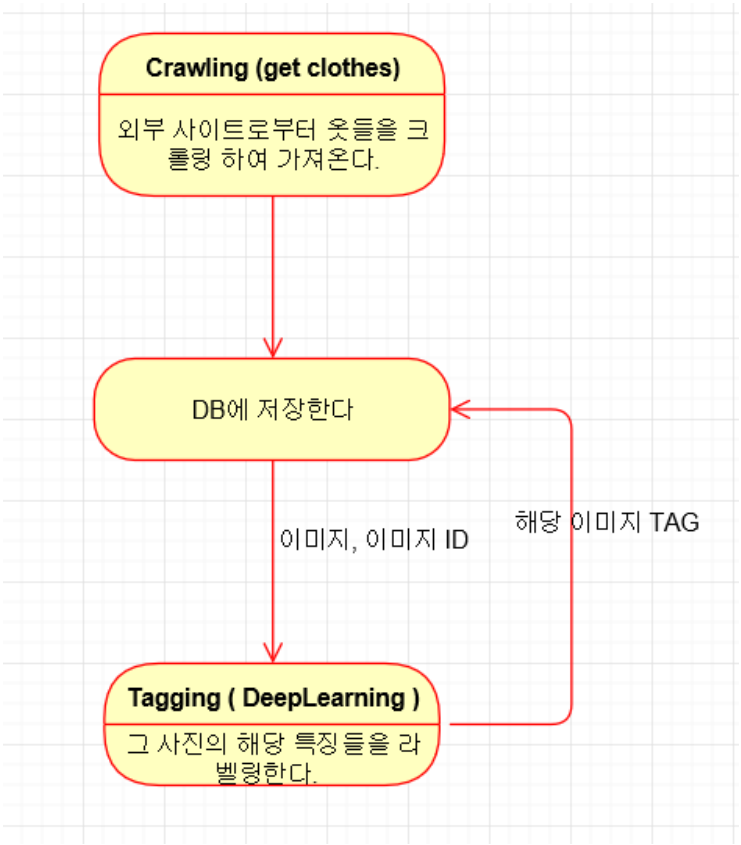


Diagram 37 crawling and tagging state diagram

8. Post Management System

8 .1. Objectives

사용자의 게시물을 관리하는 시스템의 설계를 설명한다. Post Management System의 경우, 실제 게시물의 내용을 관리하는 Content Management System과 게시물의 댓글을 관리하는 Comment Management System, 다시 하위 2개 시스템으로 나뉜다. Class diagram, Sequence diagram과 State Diagram을 통해 Post Management System의 구조를 표현하고 설명한다.

8.2. Class Diagram

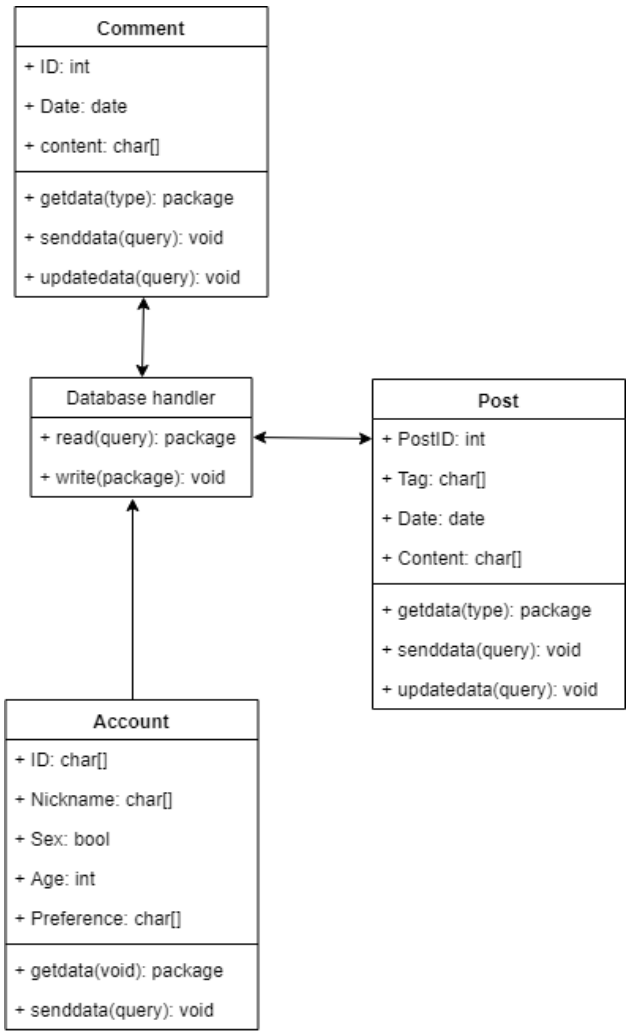


Diagram 38 Post management class diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+package read(query): 해당되는 DB에서 원하는 데이터를 읽어온다.

+void write(package): 해당되는 DB에 데이터를 저장한다.

B. Account

B.1. Attributes

+ID: 계정의 ID

+Nickname: 계정의 닉네임

+Sex: 계정의 성별

+Age: 계정의 나이

+Preference: 계정의 취향정보

B.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.

+void senddata(package): 해당되는 DB에 데이터를 보낸다.

C. Post

C.1. Attributes

+PostID: 게시물의 ID. 고유번호 값.

+Tag: 게시물의 태그정보

+Date: 게시물의 게시일

+Content: 게시물의 내용

C.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.

+ void senddata(package): 해당되는 DB에 데이터를 보낸다.

+void updatadata(): 게시물의 내용을 수정한다.

D. Comment

D.1. Attributes

+ID: 댓글의 ID. 고유번호 값.

+Date: 댓글의 게시일

+Content: 댓글의 내용

D.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.

+void senddata(package): 해당되는 DB에 데이터를 보낸다.

+void updatadata(): 게시물의 내용을 수정한다.

7.3 Sequence diagram

A. Content Management System

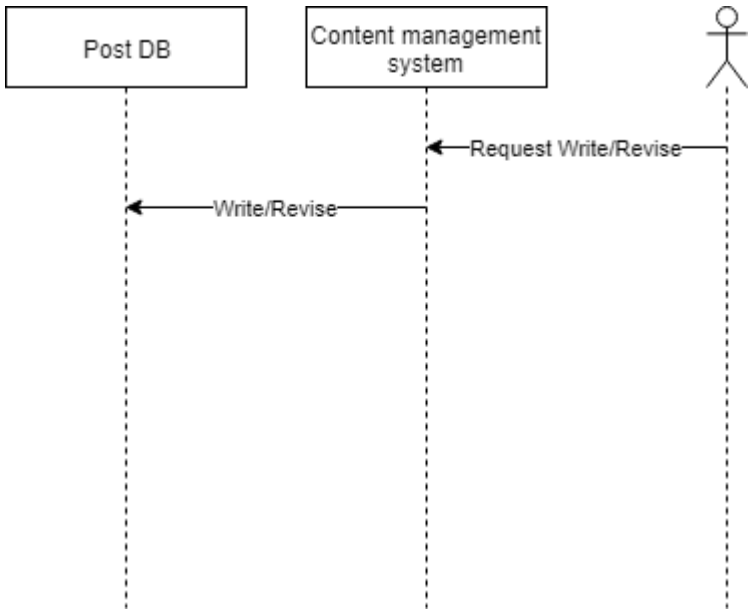


Diagram 39 Content management sequence diagram

B. Comment Management System

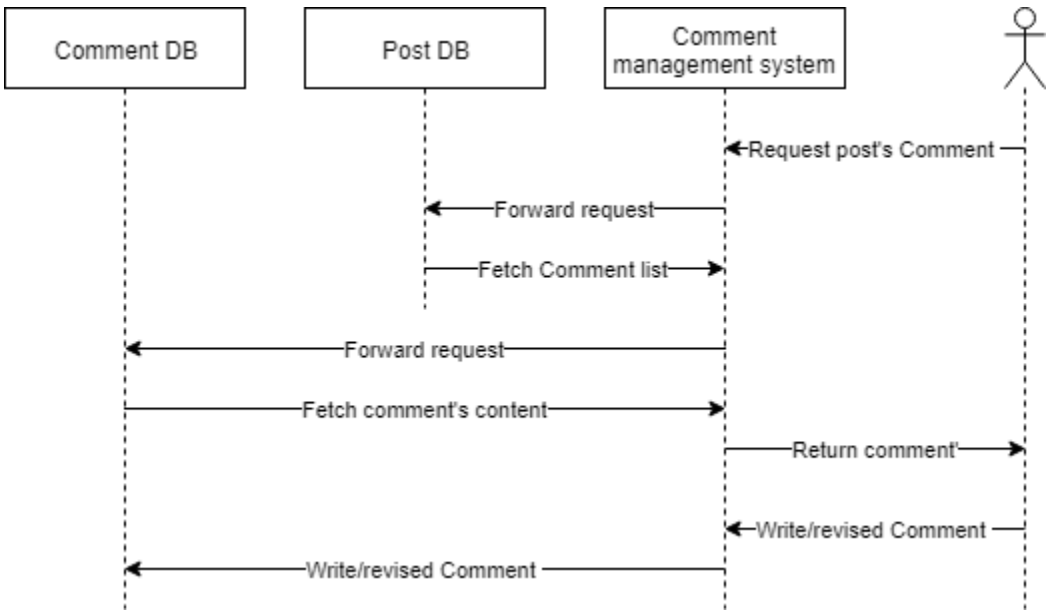


Diagram 40 comment management sequence diagram

7.4 State diagram

A. Content Management System

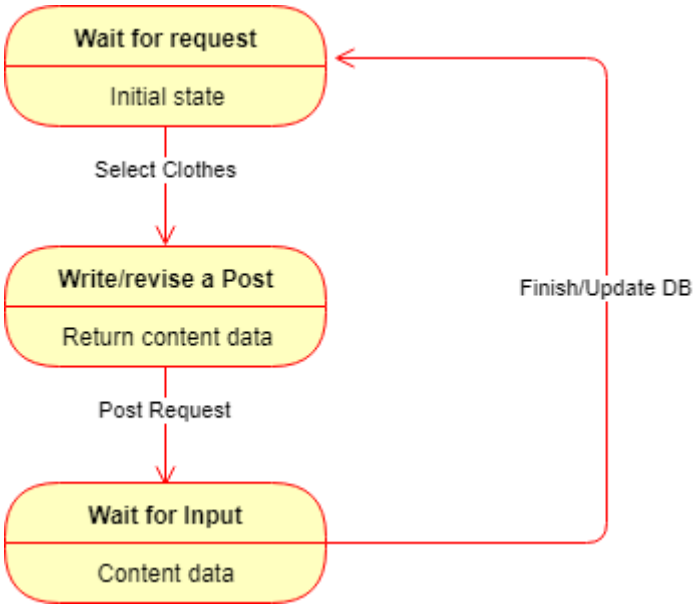


Diagram41 content management state diagram

B. Comment Management System

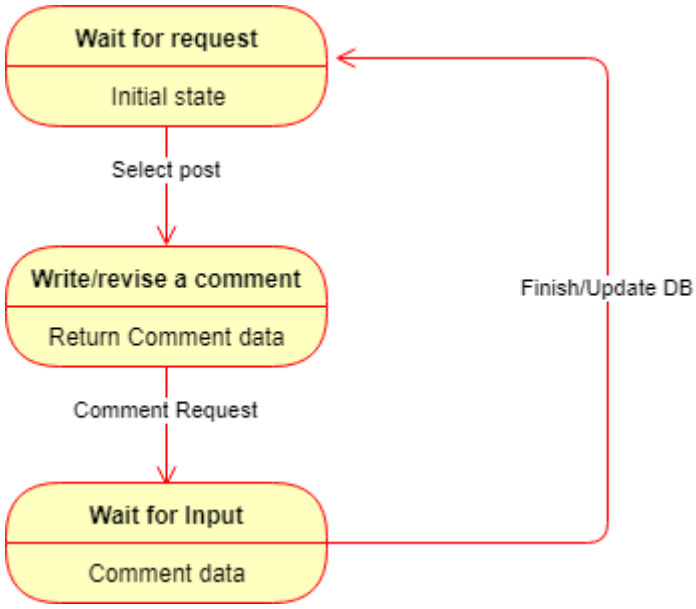


Diagram 42 content management state diagram 2

9. Display System

9.1. Objectives

실제 사용자가 사용하며 탭을 클릭하였을 때 발생하는 데이터를 처리하고 사용자의 화면에 보여주는 Display 시스템을 설명한다. Class diagram, Sequence diagram과 State Diagram을 통해 Display 시스템의 구조를 표현하고 설명한다.

9.2. Class Diagram

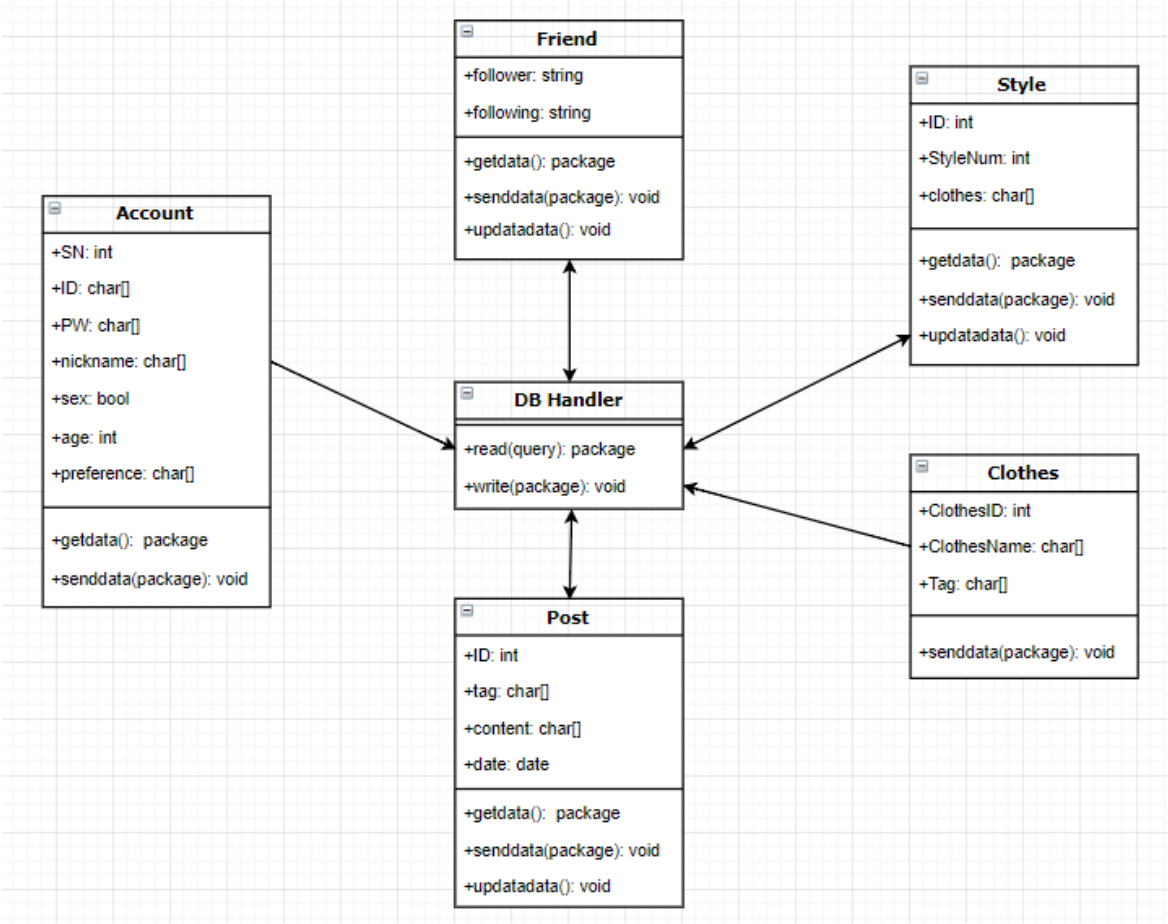


Diagram 43. Display system class diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+package read(query): 해당되는 DB에서 원하는 데이터를 읽어온다.

+void write(package): 해당되는 DB에 데이터를 저장한다.

B. Account

B.1. Attributes

+SN: 계정 고유번호
+ID: 계정 ID
+PW: 계정 비밀번호
+Nickname: 계정의 닉네임
+Sex: 계정의 성별
+Age: 계정의 나이
+Preference: 계정의 취향정보

B.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.
+void senddata(package): 해당되는 DB에 데이터를 보낸다.

C. Friend

C.1. Attributes

+follower: follow한 친구의 정보
+following: following한 자신의 정보

C.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.
+void senddata(package): 해당되는 DB에 데이터를 보낸다.
+void updatdata(): data를 업데이트한다.

D. Post

D.1. Attributes

+PostID: 게시물의 ID. 고유번호 값.
+Tag: 게시물의 태그정보
+Date: 게시물의 게시일
+Content: 게시물의 내용

D.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.
+void senddata(package): 해당되는 DB에 데이터를 보낸다.
+void updatadata(): 게시물의 내용을 수정한다.

E. Clothes

E.1. Attributes

+ClothesID: 옷의 ID
+ClothesName: 옷의 이름
+Tag: 옷의 태그 정보

E.2. Methods

+void senddata(package): 해당되는 DB에 데이터를 보낸다.

F. Style

F.1. Attributes

+ID: 계정 ID
+StyleNum: 스타일 번호
+clothes: 옷의 정보

F.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.
+void senddata(package): 해당되는 DB에 데이터를 보낸다.
+void updatadata(): 게시물의 내용을 수정한다.

9.3. Sequence Diagram

A. Tab my page & friend page

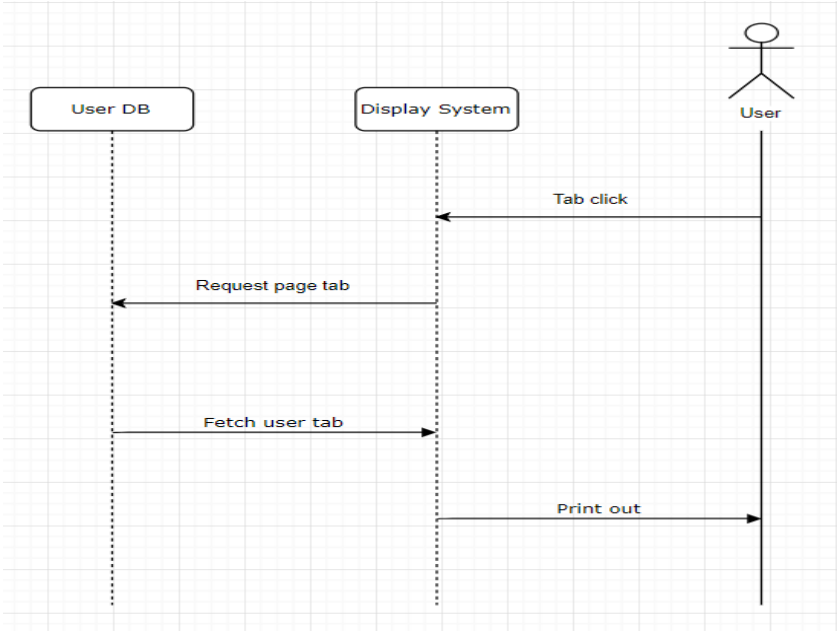


Diagram44. Tab my page & friend page sequene diagram

B. Tab style maker page

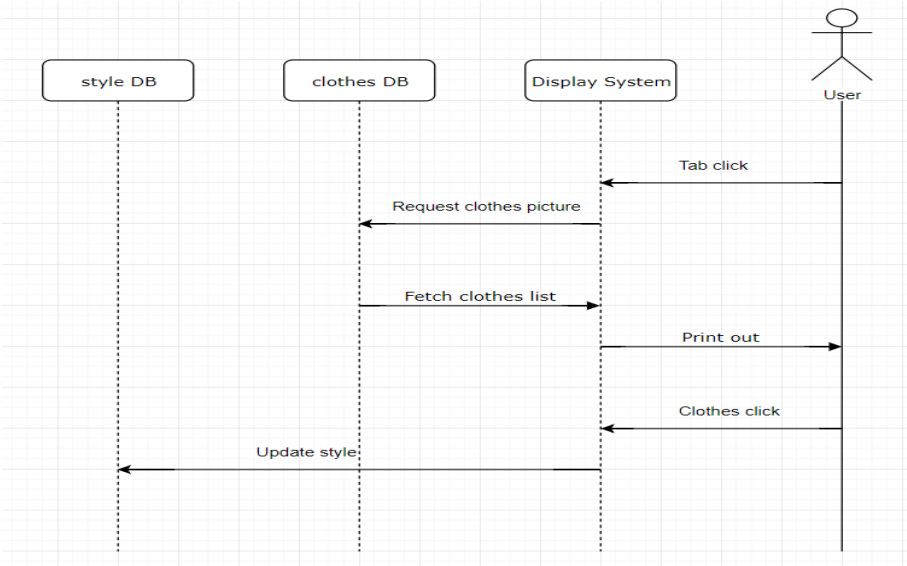
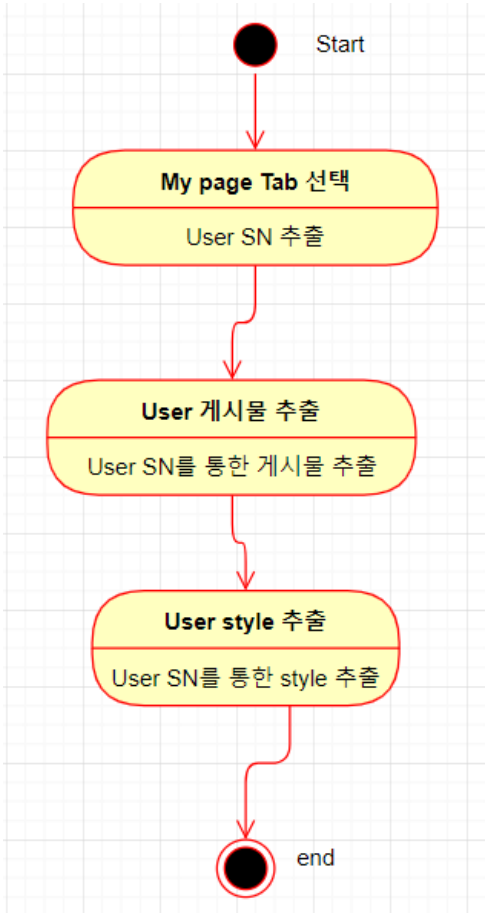


Diagram45 tab style marker page

9.4. State Diagram

A. Tab my page



B . Tab friend page

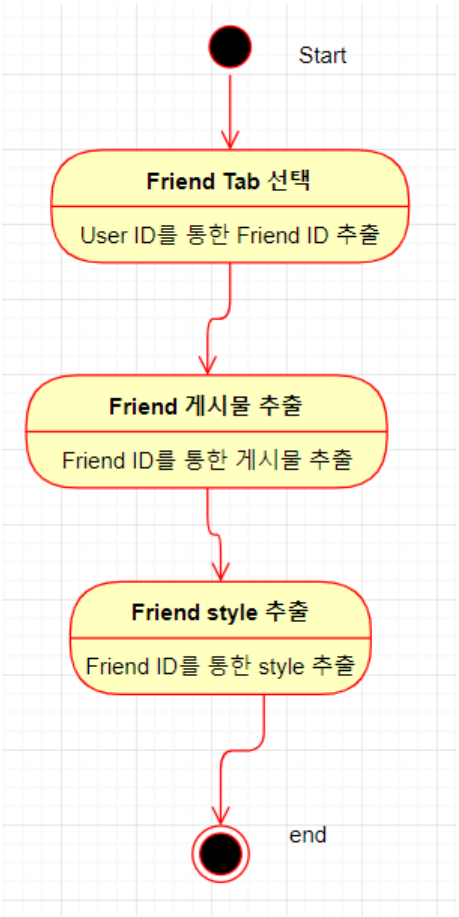


Diagram 46 tab my & friend page state diagram

c. Style Maker Page

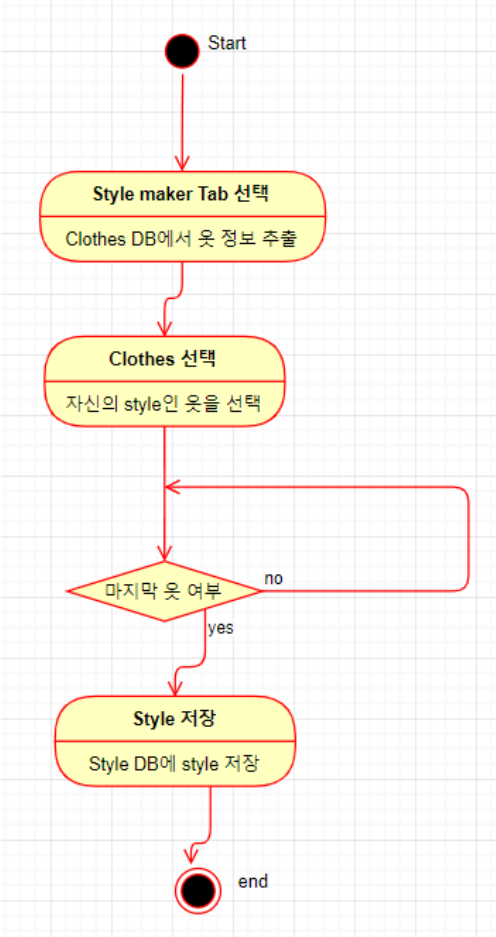


Diagram 47 style marker page state diagram

10. Extract Preferred Post System

10.1 Objectives

사용자가 등록한 관심 태그를 가진 친구들의 게시글을 추출하는 시스템의 설계를 설명한다. . Class diagram, Sequence diagram과 State Diagram을 통해 Extract preferred posts system의 구조를 표현하고 설명한다.

10.2 Class Diagram

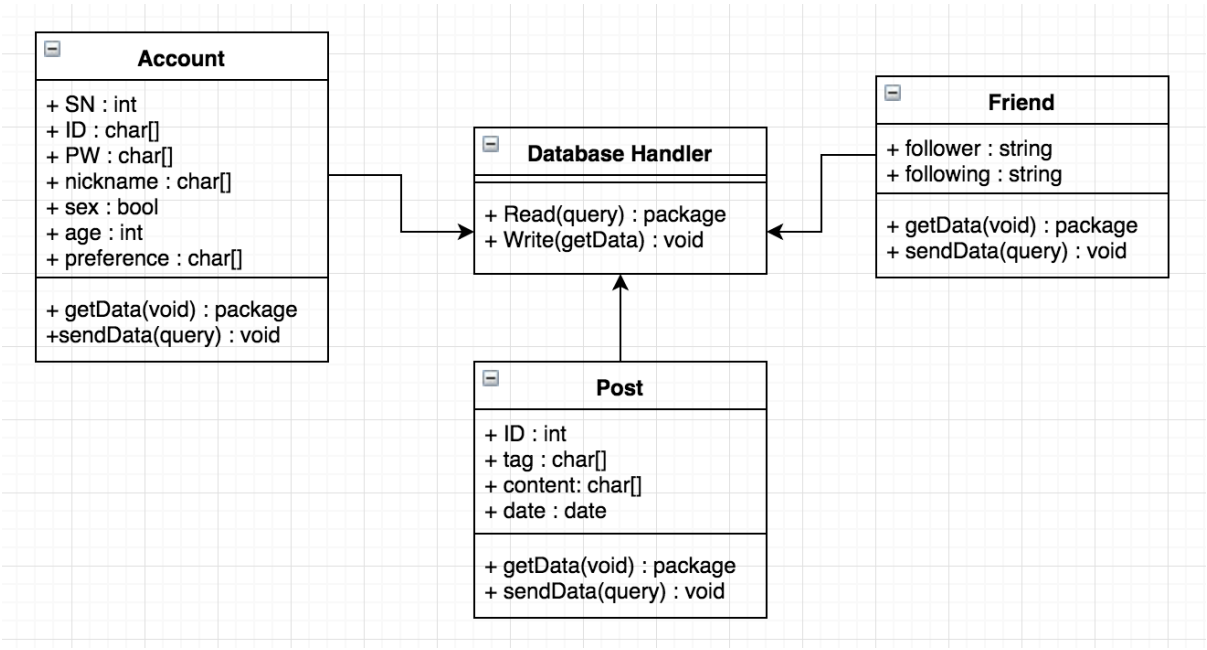


Diagram 48 Extract preferred page system class diagram

A. DB Handler

A.1. Attributes

해당 사항 없음.

A.2. Methods

+package read(query): 해당되는 DB에서 원하는 데이터를 읽어온다.

+void write(package): 해당되는 DB에 데이터를 저장한다.

B. Account

B.1. Attributes

+SN: 계정 고유번호

+ID: 계정 ID

+PW: 계정 비밀번호

+Nickname: 계정의 닉네임

+Sex: 계정의 성별

+Age: 계정의 나이

+Preference: 계정의 취향정보

B.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.

+void senddata(package): 해당되는 DB에 데이터를 보낸다.

C. Post

C.1. Attributes

+PostID: 게시물의 ID. 고유번호 값.

+Tag: 게시물의 태그정보

+Date: 게시물의 게시일

+Content: 게시물의 내용

C.2. Methods

+package getdata(): 해당되는 DB에서 원하는 데이터를 얻어온다.

+void senddata(package): 해당되는 DB에 데이터를 보낸다.

+void updatadata(): 게시물의 내용을 수정한다.

10.3 Sequence Diagram

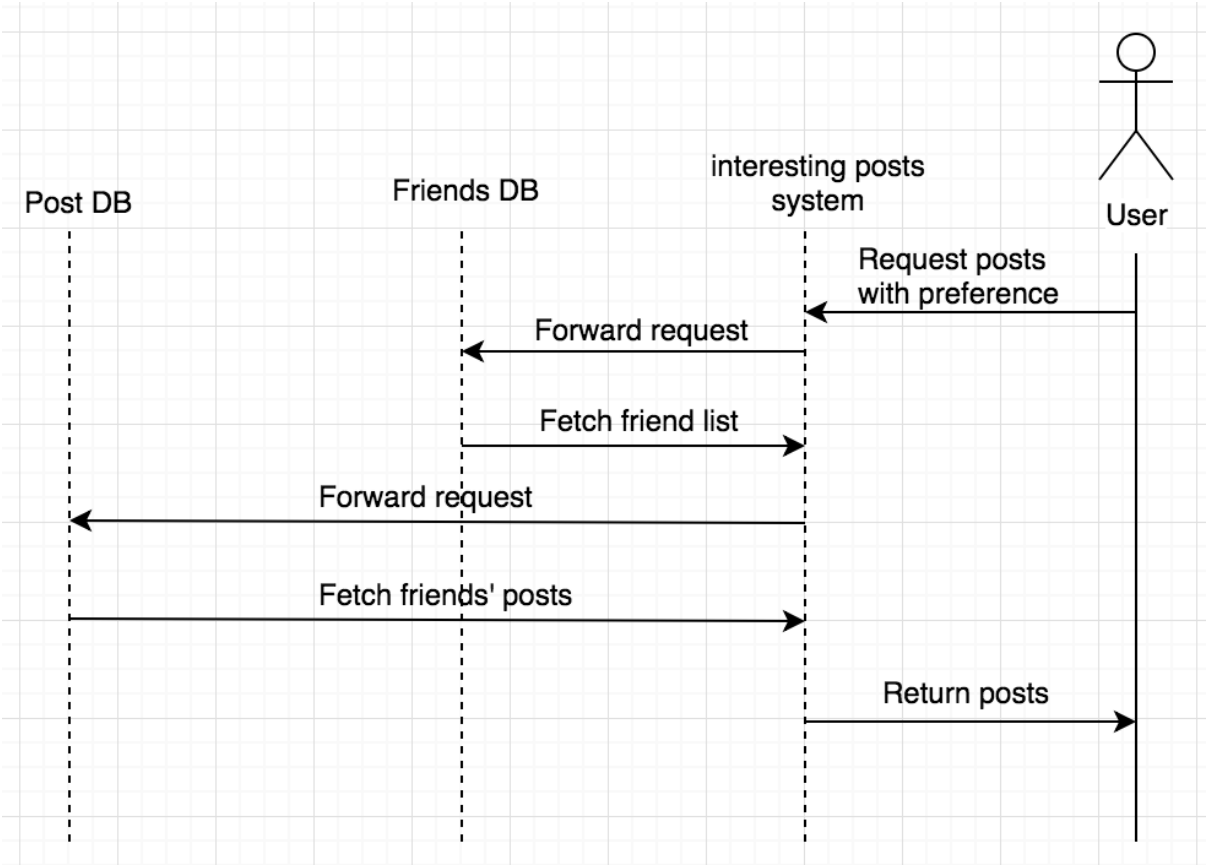


Diagram 49 extract preferred page sequence diagram

10.4 State diagram

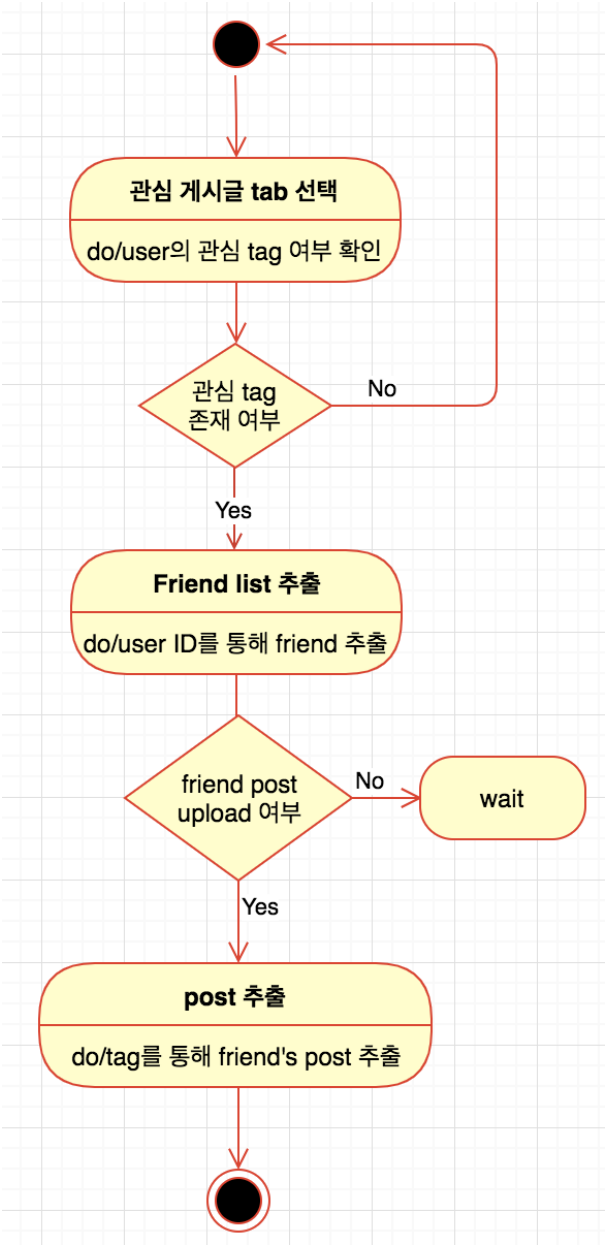


Diagram 50 Extract preferred page sequence state diagram

11. Protocol Design

11.1. Objectives

11.2. JSON

JSON은 JavaScript Object Notation의 약자입니다. XML 과 같이 텍스트를 기반으로 한 데이터 교환 시의 표준이다. 따라서 기계 뿐만이 아니라 사람도 읽기 편한 구조로 되어 있다. 따라서 다양한 프로그래밍 언어에 의해 parsing될 수 있고, HttpRequest 객체를 이용해서 서버로부터 데이터를 전송받을 수 있다.

JSON 자체가 XML의 대안으로서 좀 더 쉽게 데이터를 교환하고 저장하기 위해 고안된 것이다. 따라서 XML 보다 쉽고, 문법 및 구문이 간소화 되어있다. 예를 들어 XML에서는 종료태그를 사용하지만, JSON에서는 종료태그를 사용하지 않는다. 또한 JSON은 배열을 사용할 수 있고, 자바스크립트의 표준 함수인eval() 함수로 파싱된다. 이러한 특징들 덕에 JSON 데이터가 XML 데이터보다 더 빨리 읽고 쓸 수 있다. 모바일을 기반으로 하는 데이터 전송에 사용하기에 적절한 형식이다. 또한 AJAX를 사용하여 데이터를 주고 받을 때의 포맷으로서 사용된다.

11.3. Protocol Description

A. Overview

HTTP 통신에서 client와 server 사이에서 전송되는 메시지의 형태를 용도 별로 정의한다. Client에

서의 요청(request) 메시지와 server에서의 응답 (response) 메시지로 구분한다. 해당 절의 표는 캡션을 생략한다.

B. Community display protocol

a. Request

Attribute	Value
Tab	데이터 필드 값
SN	User의 고유번호

b. Response

Attribute	Value	
Title	글의 제목	
Content	글의 내용	
Writer	작성자	
Date	작성한 날짜	
Comment list	댓글 목록	
	Attribute	Value
	Content	글의 내용
	Writer	작성자

C. Style store protocol

a. Request

Attribute	Value
SN	User의 고유번호
ClothesID	옷의 ID
ClothesName	옷의 이름
Tag	옷의 태그 정보
StyleNum	Style의 숫자

b. Response

Attribute	Value
Reg_success	Style 저장 성공 여부

D. Content post protocol

a. Request

Attribute	Value
SN	User의 고유번호
User ID	User의 ID
ClothesID	옷의 ID
ClothesName	옷의 이름
Tag	옷의 태그 정보

b. Response

Attribute	Value
Reg_success	Content 게시 성공 여부

E. Content Post/Edit Protocol

a. Request

Attribute	Value
ID	사용자의 ID
content	게시물 내용

b. Response

Attribute	Value
reg_success	글 작성/수정 성공 여부

F. Extract preferred post Protocol

a. Request

ATTRIBUTE	**VALUE**
ID	User의 id값
FOLLOWING	User의 친구 id값
TAG	User가 지정한 선호 태그

b. Response

ATTRIBUTE	**VALUE**	
EXTRACT_RESULT	추출된 게시글의 목록	
	Attribute	*Value*

	Post ID	글의 고유 ID
	content	글의 내용
	Following	친구 ID
	date	작성 날짜

G. 취향 생성 및 update protocol

a) Request

Attribute	Value
User ID	사용자의 ID
User Preference vector	그 사용자의 취향 벡터 (Null vector 혹은 기존의 vector)
Clothes ID	사용자가 고른 옷 ID
Clothes Label	사용자가 고른 옷의 label 값

b) Response

Attribute	Value
User preference vector	업데이트 된 vector

H.친구 Recommend Protocol

a) Request

Attribute	Value
User ID	사용자의 ID
Friends ID	다른 사용자들의 ID
User preference	해당 사용자의 취향 벡터
Friends preference	다른 사용자들의 취향 벡터들
유사도 기준	친구를 추천하는 유사도 기준

b) Response

Attribute	Value
유사도	벡터들 간의 유사도 계산 결과값
Friends ID	특정 유사도 이상의, 추천된 친구들 ID

I. Crawling Protocol

a) Request

Attribute	Value
URL	크롤링하려는 target 사이트
Number of images	크롤링 요청하는 사진 수

b) Response

Attribute	Value
Images	크롤링 되어오는 사진들

J. Tagging Protocol

a) Request

Attribute	Value
Clothes ID	해당 옷의 ID
Clothes Image	해당 옷의 이미지

b) Response

Attribute	Value
Label	해당 사진의 특징을 반영하는 label 값

K. Friend View Protocol

a. Request

Attribute	Value
ID	사용자의 ID

b. Response

Attribute	Value
Follower	친구로 등록한 사용자의 ID
Following	친구로 등록된 사용자의 ID

L. Friend Add Protocol

a. Request

Attribute	Value
Friend_ID	추가하고 하는 친구의 ID
User_ID	사용자의 ID

b. Response

Attribute	Value
Add_Success	친구 추가 성공 여부

M. Friend Search Protocol

a. Request

Attribute	Value
Friend_Nickname	찾고자 하는 친구의 닉네임

b. Response

Attribute	Value
Friend_ID	탐색한 친구의 ID
Friend_Nickname	탐색한 친구의 닉네임
Search_Success	친구 탐색 성공 여부

N. Comment Post/Edit Protocol

a. Request

Attribute	Value
ID	작성자의 ID
Post ID	게시물의 ID
content	댓글 내용

b. Response

Attribute	Value
reg_success	댓글 작성/수정 성공 여부

12. Database Design

12.1 Objective

Database Design은 요구사항 명세서에서 기술한 데이터베이스 요구사항을 바탕으로 작성하였다. 요구사항을 바탕으로 ER Diagram을 작성하고, 이를 이용하여 Relational Schema를 작성하고, 마지막으로 SQL DDL을 작성한다

12. 2 ER diagram

개체는 분리된 물체 하나를 표현한다. 개체는 사각형으로 표현되며, 관계는 다이아몬드로 표현된다. 개체나 관계는 특성을 가질 수 있으며, 이 특성들은 관계 집합에 실선으로 연결된 타원형으로 표현한다.

관계는 두개 이상의 개체들의 연관 관계를 표현한다. 모든 개체는 고유하게 식별되는 특성 집합을 가지고 있어야 하며, 최소한의 고유 식별 특성 집합은 개체의 기본 키(Primary Key)라 불린다.

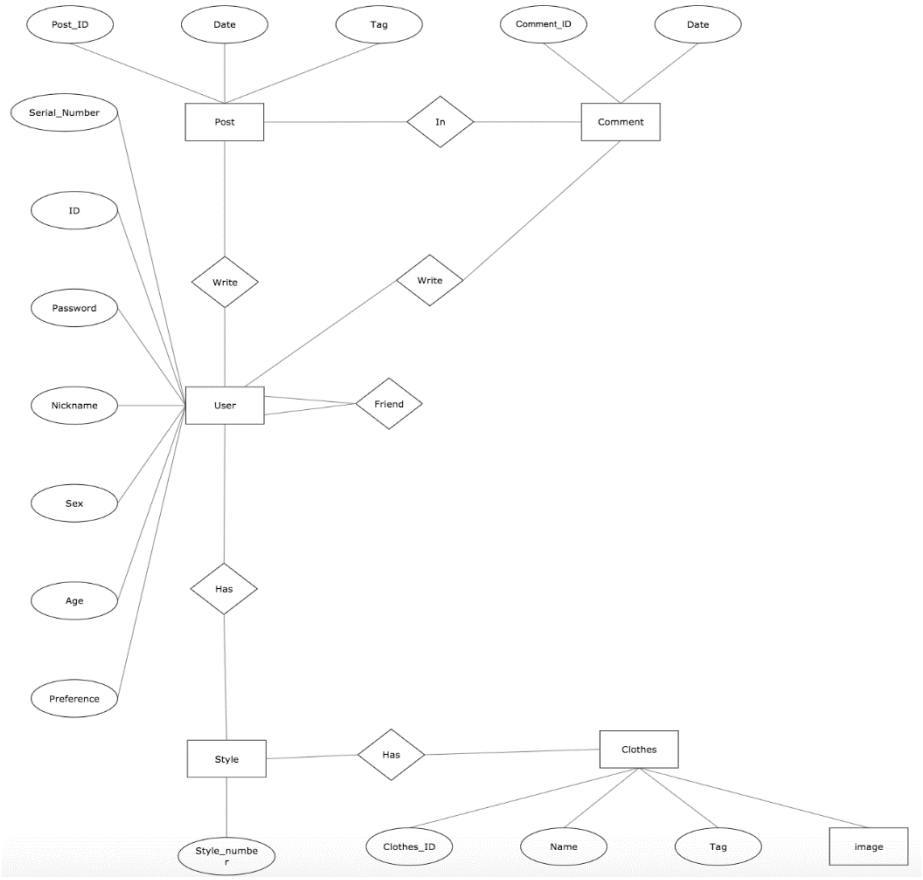


Diagram 51 overall ER Diagram

A. Entity

A.1. User

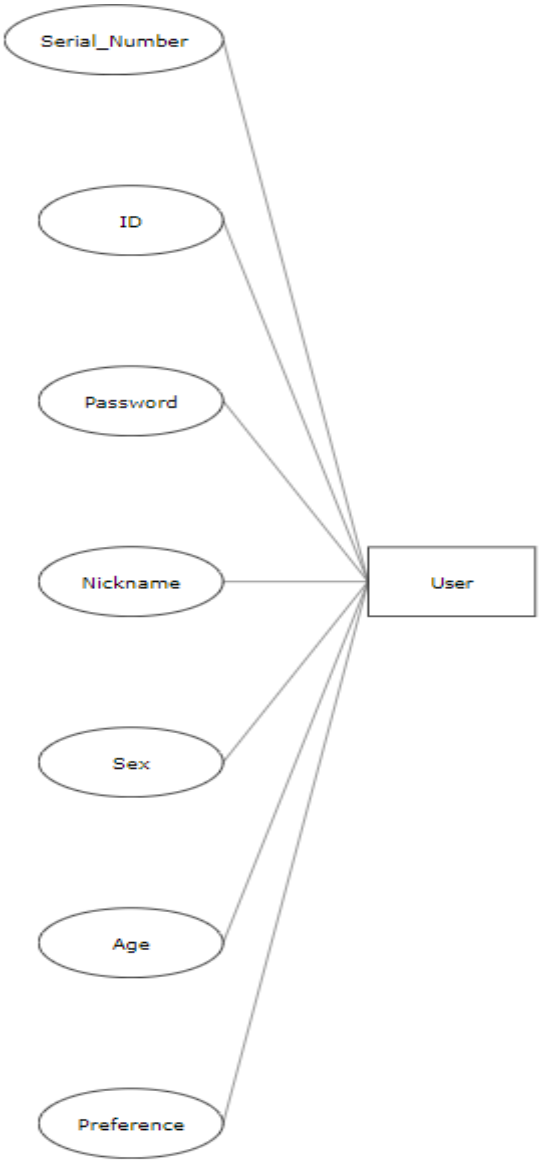


Diagram 52. entity - User

User는 사용자를 나타낸다. ID, Password, Nickname, Sex, Age, Preference의 속성을 가지고 있으며, Primary Key는 Serial_Number이다.

A.2. Style

Diagram 53 entity -style.

Style은 사용자의 스타일을 나타낸다. Style_number의 속성을 가지고 있으며 Primary Key는 Style_number와 User_ID이다.

A.3. Clothes

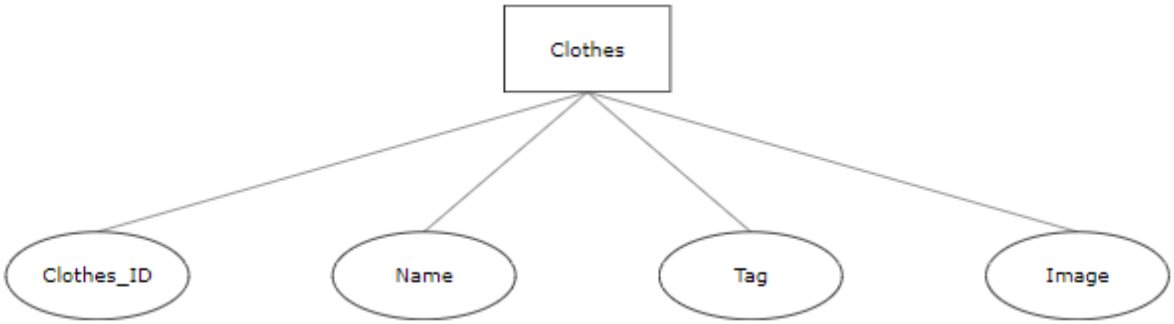


Diagram 54. Entity- clothes

Clothes는 옷의 정보를 나타낸다. Clothes_ID, Name, Tag, Image의 속성을 가지고 있으며 Primary Key는 Clothes_ID이다.

A.4. Post

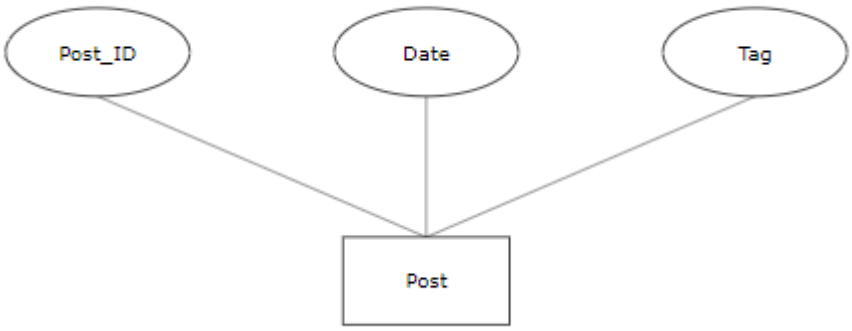


Diagram 55 entity - post

Post는 게시물을 나타낸다. Post_ID, Date, Tag의 속성을 가지고 있으며 Primary Key는 Post_ID이다.

A.5. Comment

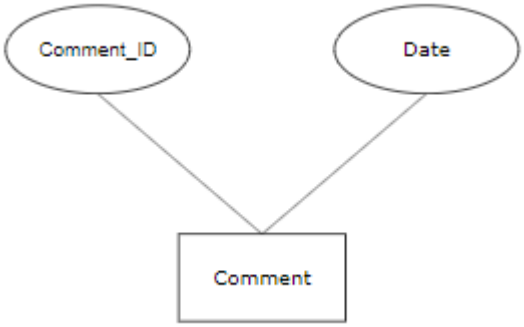


Diagram 56 entity- comment

Comment는 게시물의 달린 댓글을 나타낸다. Comment_ID, Date의 속성을 가지고 있으며 Primary Key는 Comment_ID이다.

B. Relationship

B.1. Write post

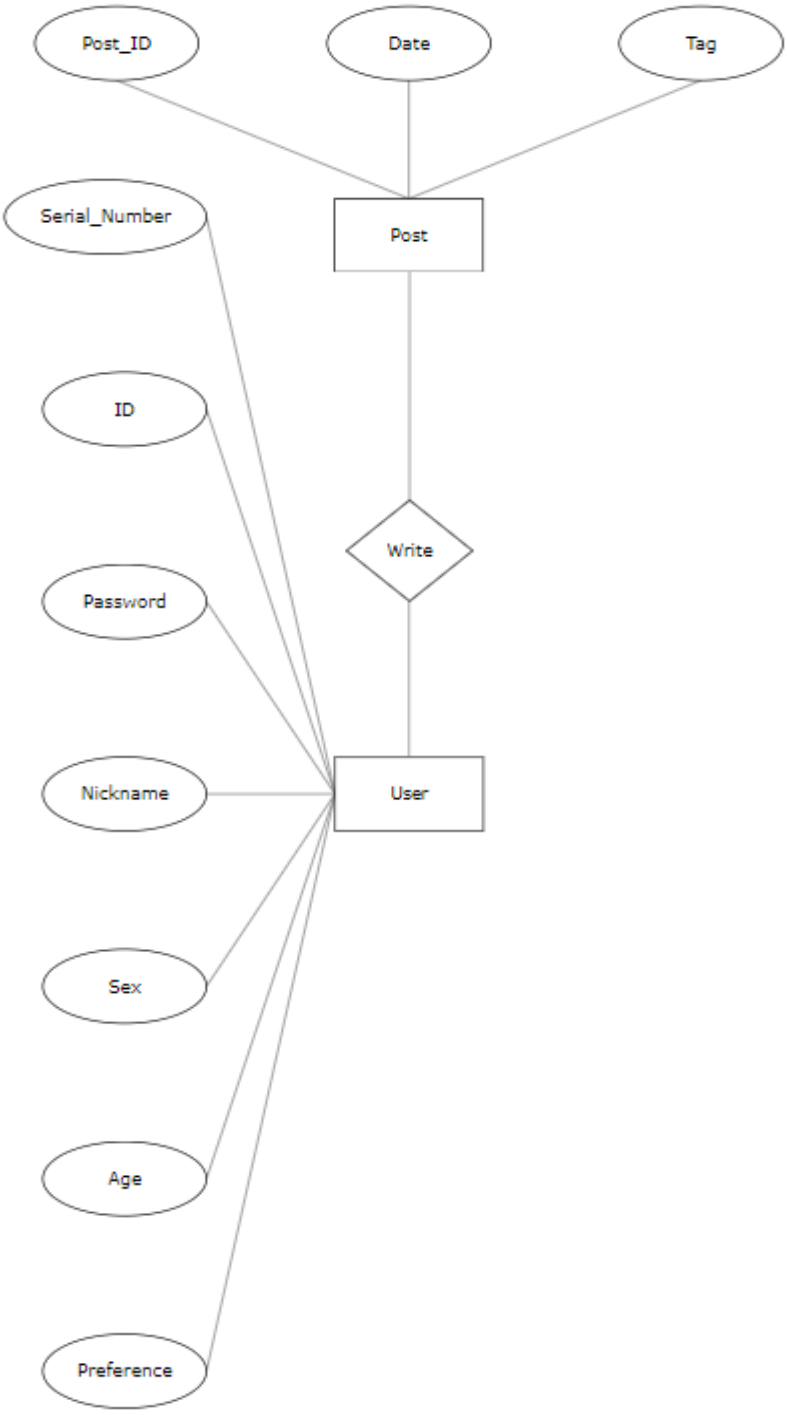


Diagram 57 relation- write post

Write post는 사용자(User)가 게시물을 작성하는 관계로, 1명의 사용자가 여러 게시물을 작성할 수 있으며, 모든 작성된 게시물은 작성자가 1명이다.

B.2. Write Comment

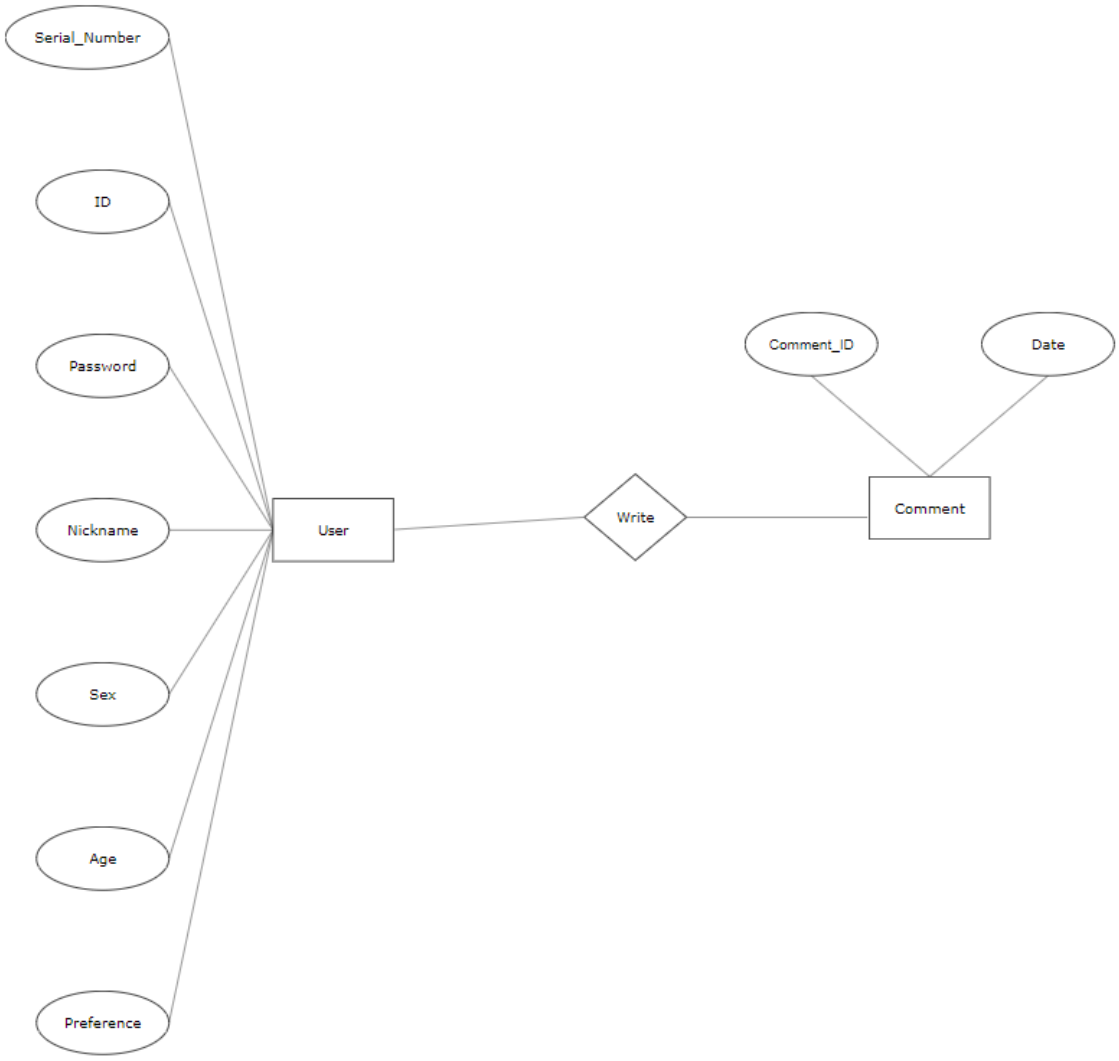


Diagram 58 relation – write comment

Write comment는 사용자(User)가 게시물에 댓글을 작성하는 관계로, 1명의 사용자가 여러 댓글을 작성할 수 있으며, 모든 작성된 댓글은 작성자가 1명이다.

B.3. Clothes in Style

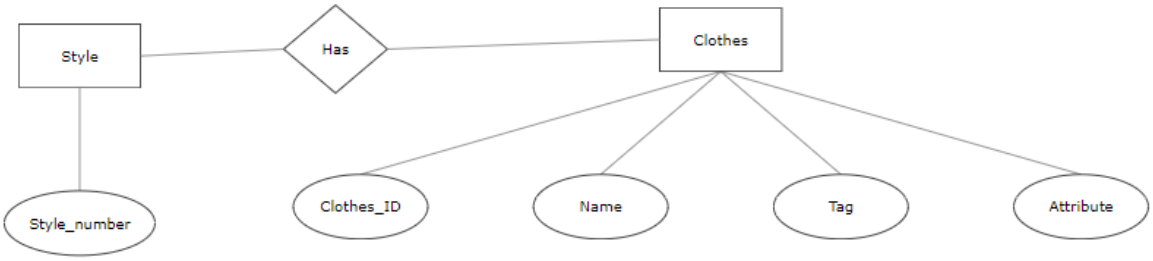


Diagram 59. Relation – clothes in style

Clothes in Style는 옷이 특정 스타일에 포함되는 관계로, 여러 옷이 여러 스타일에 포함될 수 있다.

B.4. Style in User

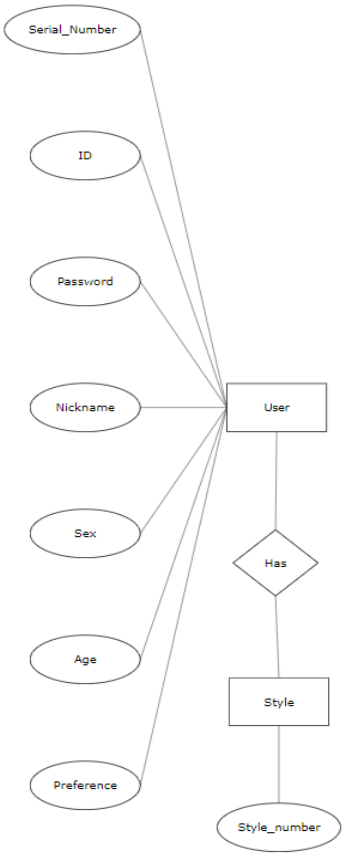


Diagram 60 relation – style in user

Style in Use는 스타일이 사용자에 포함되는 관계로, 1명의 사용자가 여러 스타일을 가질 수 있으며, 모든 스타일은 1명의 사용자에게 포함된다.

B.5. Be Friend

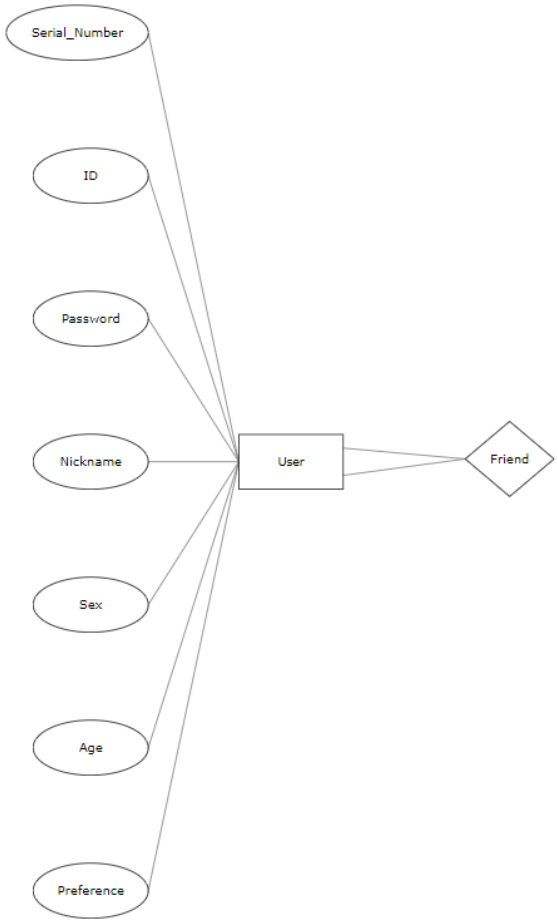


Diagram 61. Relation – Be a friend

Be Friend는 사용자(User)간의 친구 관계로, 여러 명의 사용자가 여러 명의 사용자와 친구가 될 수 있다.

B.6. Comment in Post

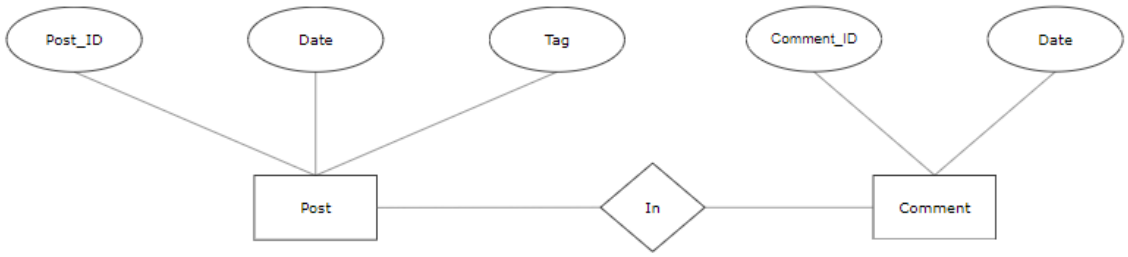


Diagram 62. Relation - comment in post

Comment in Post는 댓글이 게시물에 포함되는 관계로, 1개의 게시 글에 여러 댓글이 포함 될 수 있으며, 모든 작성된 댓글은 1개의 게시 글에 포함된다.

12.3. Relational Schema

A. Member

Member_SN	ID	PW	Nickname	Sex	Member_Tag	Age

Primary key (PK): Member_SN

Foreign key (FK): 없음

FUNCT DEP (FD):

Member_SN → {Member_SN, ID, PW, Nickname, Sex, Member_Tag, Age}

Description: Entity Member에 관한 테이블이다. Tag 속성을 제외한 모든 속성에 NULL을 허용하지 않는다.

B. Friend

Follower	Following

Primary key (PK): 없음

Foreign key (FK): Following, Follower

FUNCT DEP (FD): 없음

Description: Entity Friend에 관한 테이블이다. 모든 속성에 NULL을 허용하지 않으며, Member의 PK를 FK로 가져온다.

C. Clothes

Clothes_SN	Name	Image	Clothes_Tag

Primary key (PK): SN

Foreign key (FK): 없음

FUNCT DEP (FD):

Clothes_SN → {Clothes_SN, Name, Image, Clothes_Tag}

Description: Entity Clothes에 관한 테이블이다. 모든 속성에 NULL을 허용하지 않는다.

D. Style

Style_Num	Style_Tag	Member_SN

Primary key (PK): Style_Num, Member_SN

Foreign key (FK): 없음

FUNCT DEP (FD):

{Style_Num, Member_SN} → {Style_Num, Style_Tag, Member_SN}

Description: Entity Style에 관한 테이블이다. 모든 속성에 NULL을 허용하지 않으며, Clothes의 PK를 PK로, Member의 PK를 FK로 가져온다.

E. Style list

Style_Num	Clothes_SN

Primary key (PK): 없음

Foreign key (FK): Style_Num, Clothes_SN

FUNCT DEP (FD): 없음

Description: Entity Style list에 관한 테이블이다. 모든 속성에 NULL을 허용하지 않으며, Clothes의 PK와 Style의 PK를 FK로 가져온다.

F. Post

Post_SN	Post_Tag	Post_date	Member_SN

Primary key (PK): Post_SN

Foreign key (FK): Member_SN

FUNCT DEP (FD):

Post_SN → {Post_SN, Post_Tag, Post_date, Member_SN}

Description: Entity Post에 관한 테이블이다. 모든 속성에 NULL을 허용하지 않으며, Member의 PK를 FK로 가져온다.

G. Comment

Comment_SN	Post_SN	Comment_date	Member_SN

Primary key (PK): Comment_SN, Post_SN

Foreign key (FK): Member_SN

FUNCT DEP (FD):

{Comment_SN, Post_SN} → {Comment_SN, Post_SN, Comment_date, Member_SN}

Description: Entity Comment에 관한 테이블이다. 모든 속성에 NULL을 허용하지 않으며, Post의 PK를 PK로, Member의 PK를 FK로 가져온다.

12.4. SQL DDL

A. Member

```
CREATE TABLE Member
(
    `Member_SN`    INT              NOT NULL     AUTO_INCREMENT COMMENT 'Serial Number',
    `ID`           VARCHAR(45)      NOT NULL     COMMENT 'ID',
    `PW`           VARCHAR(45)      NOT NULL     COMMENT 'Password',
    `Nickname`     VARCHAR(45)      NOT NULL     COMMENT 'Nickname',
    `Sex`          CHAR(1)          NOT NULL     COMMENT 'Sex',
    `Age`          INT              NOT NULL     COMMENT 'Age',
    `Member_Tag`   VARCHAR(45)      NULL         COMMENT 'Preference',
    PRIMARY KEY (Member_SN)
);
```

B. Style

```
CREATE TABLE Style

(

    `Member_SN`   INT                NOT NULL      COMMENT 'UserID',

    `Style_Num`   INT                NOT NULL      COMMENT 'StyleNum',

    `Style_Tag`   VARCHAR(45)      NOT NULL,

    PRIMARY KEY (Member_SN, Style_Num)

);


ALTER TABLE Style ADD CONSTRAINT FK_Style_Member_SN_Member_Member_SN FOREIGN KEY
(Member_SN)

 REFERENCES Member (Member_SN)   ON DELETE RESTRICT ON UPDATE RESTRICT;
```

C. Post

```
CREATE TABLE Post

(

    `Post_SN`      INT                NOT NULL      AUTO_INCREMENT COMMENT 'PostID',

    `Member_SN`   INT                 NOT NULL      COMMENT 'UserID',

    `Post_date`   DATE                NOT NULL      COMMENT 'Date',

    `Post_Tag`    VARCHAR(45)      NOT NULL      COMMENT 'Tag',

    PRIMARY KEY (Post_SN)

);


ALTER TABLE Post ADD CONSTRAINT FK_Post_Member_SN_Member_Member_SN FOREIGN KEY
(Member_SN)

 REFERENCES Member (Member_SN)   ON DELETE RESTRICT ON UPDATE RESTRICT;
```

D. Clothes

```
CREATE TABLE Clothes

(

    `Clothes_SN`    INT                NOT NULL      AUTO_INCREMENT COMMENT 'ClothesID',

    `Name`            VARCHAR(45)      NOT NULL      COMMENT 'Name',

    `Clothes_Tag`   VARCHAR(45)      NOT NULL      COMMENT 'Tag',

    `Image`           TEXT               NOT NULL,

    PRIMARY KEY (Clothes_SN)

);
```

E. Comment

```
CREATE TABLE Comment

(

    `Comment_SN`      INT       NOT NULL      AUTO_INCREMENT COMMENT 'CommentID',

    `Post_SN`         INT       NOT NULL      COMMENT 'PostID',

    `Member_SN`       INT       NOT NULL      COMMENT 'UserID',

    `Comment_date`    DATE      NOT NULL      COMMENT 'Date',

    PRIMARY KEY (Comment_SN)

);


ALTER TABLE Comment ADD CONSTRAINT FK_Comment_Post_SN_Post_Post_SN FOREIGN KEY (Post_SN)

 REFERENCES Post (Post_SN)   ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE Comment ADD CONSTRAINT FK_Comment_Member_SN_Member_Member_SN FOREIGN KEY (Member_SN)

 REFERENCES Member (Member_SN)   ON DELETE RESTRICT ON UPDATE RESTRICT;
```

F. Friend

```
CREATE TABLE Friend

(

    `Follower`    INT      NOT NULL      COMMENT 'From',

    `Following`   INT      NOT NULL      COMMENT 'To',

    PRIMARY KEY (Follower, Following)

);


ALTER TABLE Friend ADD CONSTRAINT FK_Friend_Follower_Member_Member_SN FOREIGN KEY
(Follower)

 REFERENCES Member (Member_SN)   ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE Friend ADD CONSTRAINT FK_Friend_Following_Member_Member_SN FOREIGN KEY
(Following)

 REFERENCES Member (Member_SN)   ON DELETE RESTRICT ON UPDATE RESTRICT;
```

G. Style_list

```
CREATE TABLE Style_list

(

    `Clothes_SN`   INT      NOT NULL,

    `Member_SN`    INT      NOT NULL,

    `Style_Num`    INT      NOT NULL,

    PRIMARY KEY (Clothes_SN, Member_SN, Style_Num)

);


ALTER TABLE Style_list ADD CONSTRAINT FK_Style_list_Clothes_SN_Clothes_Clothes_SN FOREIGN KEY (Clothes_SN)

 REFERENCES Clothes (Clothes_SN)   ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE Style_list ADD CONSTRAINT FK_Style_list_Member_SN_Style_Member_SN FOREIGN KEY (Member_SN)

 REFERENCES Style (Member_SN)   ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE Style_list ADD CONSTRAINT FK_Style_list_Style_Num_Style_Style_Num FOREIGN KEY (Style_Num)

 REFERENCES Style (Style_Num)   ON DELETE RESTRICT ON UPDATE RESTRICT;
```

13. Testing Plan

13.1. Objectives

시스템이 의도한 방향으로 실행되고 시스템 내부의 결함을 찾기 위해 testing 을 한다. 이를 위해 설계단계에 미리 계획한다. 이 때 Testing Plan 에서는 Testing Policy 와 여러 Test Case 에 대해 기술한다.

12.2. Testing Policy

중년들의 패션친구 시스템의 개발에서는 크게 3 단계로 나누어 testing 을 진행한다. DevelopingTesting, Release Testing, User Testing 으로 나눠지며, Developing Testing 은 다시 Component Testing, Integrating Testing, System Testing, Acceptance Testing 의 4 단계로 나눠진다.

1) Developing Testing

개발 과정에서 수행되는 testing 으로 Component Testing 은 각 component(unit) 단위로 개발이 진행되면, 각 요소들이 개발한 후에 제대로 작동하는지 확인하는 testing 이다.

2) Integrating Testing

Component Testing 후 각 요소들을 하나씩 점진적으로 합치면서 하는 testing 이다. System Testing 은 모든 하위 시스템을 하나로 합친 후, 그 시스템이 잘 동작하는지 testing 하는 것이다. Acceptance Testing 은 사용자의 정보를 이용하여 시스템에 대한 사용자의 요구사항을 testing 하는 것이다.

3) Release Testing

사용자에게 출시하기 전에 최종 시스템을 testing 하는 것이다. 요구사항 명세서에서 작성되었던 요구사항이 제대로 반영되었는지를 확인한다.

3) User Testing

사용자가 사용자의 환경에서 시스템을 testing 하는 것이다.

12.3. Test Case

A. User Management System

A.1. Sign Up With ‘Kakao Talk’

1) User: ‘Kakao Talk’을 이용하여 회원가입을 시도한다.

2) 시스템 동작: Member DB 에 저장되어 있는 데이터에 Kakao Talk 서버에서 전송받은 User 의 ID 가 존재하는지를 확인한다.

2-1) (중복 확인 성공) 시스템 알림: ”중복되는 ID 가 없습니다.” 시스템 동작: 다음 정보 입력 단계로 이동한다.

2-2) (중복 확인 실패) 시스템 알림: ”중복되는 ID 가 없습니다.” 시스템 동작: 초기 화면으로 돌아간다.

3) User: ID 이외의 회원정보를 입력한다.

4) 시스템 알림: ”회원가입에 성공했습니다.” 시스템 동작: 입력받은 정보를 Member DB 에 저장한다.

A.2. Sign Up Without ‘Kakao Talk’

1) User: ID 를 입력하여 회원가입을 시도한다.

2) 시스템 동작: Member DB 에 저장되어 있는 데이터에 입력받은 User 의 ID 가 존재하는지를 확인한다.

2-1) (중복 확인 성공) 시스템 알림: ”중복되는 ID 가 없습니다.” 시스템 동작: 다음 정보 입력 단계로 이동한다. (3 번으로 이동)

2-2) (중복 확인 실패) 시스템 알림: ”중복되는 ID 가 없습니다.” 시스템 동작: 초기 화면으로 돌아간다. (기능 종료)

3) User: ID 이외의 회원정보를 입력한다.

4) 시스템 알림: "회원가입에 성공했습니다." 시스템 동작: 입력받은 정보를 Member DB 에 저장한다.

A.3. Log In With 'Kakao Talk'

1) User: 'Kakao Talk'을 이용하여 로그인을 시도한다.

2) 시스템 동작: Member DB 에 저장되어 있는 데이터와 Kakao Talk 서버에서 받아온 User 의 정보와 일치 여부를 확인한다.

2-1) (로그인 성공) 시스템 알림: "Welcome to fashion friend." 시스템 동작: 해당 아이디로 로그인 된다.

2-2) (로그인 실패) 시스템 동작: Kakao Talk 을 통한 회원가입을 자동으로 실행한다. 중년들의 패션친구로의 정보 제공 동의를 요청한 후, 자동으로 Kakao Talk 으로 회원가입 양식을 불러온다.

A.4. Log In Without 'Kakao Talk'

1) User: ID 와 PW 를 입력하여 로그인을 시도한다.

2) 시스템 동작: Member DB 에 저장되어 있는 데이터와 입력한 ID 와 PW 와 일치 여부를 확인한다.

2-1) (로그인 성공) 시스템 알림: "Welcome to fashion friend." 시스템 동작: 해당 아이디로 로그인 된다.

2-2) (로그인 실패) 시스템 동작: 초기 화면으로 돌아간다.

B. Post Management System

B.1. Add or Revise Comment and Post

1) User: Comment 나 Post 를 새로 입력하거나 수정한다.

2) 시스템 동작 : 입력되거나 수정된 사항을 각각 Comment DB 와 Post DB 에 반영한다.

B.2. Display Comment and Post

1) User: Post 와 Comment 가 포함된 Page 에 들어간다. (ex) 마이페이지, 친구페이지 등)

2) 시스템 동작: Post DB 와 Comment DB 에서 각각 해당하는 User 의 Post 와 Comment 를 가져온다.

C. Display System

C.1. Display My Page

1) User: My page 탭을 누른다.

2) 시스템 동작: 자신의 SN 을 통해 Post DB 와 Style DB 에서 검색되어 출력된다.

C.2. Display Friend Page

1) User: Friend Page 에 접속한다.

2) 시스템 동작: 접속한 친구의 SN 을 통해 Post DB 와 Style DB 에서 검색되어 출력된다.

C.3. Display Style Maker Page

1) User: Style maker 탭을 누른다.

2) 시스템 동작: Clothes DB 에서 옷을 랜덤으로 선택하여 출력된다.

3) User: Style maker 탭에서 옷을 선택한 후 스타일 저장을 시도한다.

4) 시스템 알림: "Success"시스템 동작: 스타일이 저장되어 My page 에서 볼 수 있다.

D. Friend Management System

D.1. View Friend List

1) User: Friend 탭에 접속한다.

2) 시스템 동작: Friend DB 에서 User 의 ID 를 기반으로 친구 리스트를 추출해낸 뒤, 출력한다.

D.2. Add Friend

1) User: 친구 추가 버튼을 누른다.

2) 시스템 동작: Friend DB 에 자신의 ID 와 친구의 ID 를 기반으로 친구 데이터를 추가한다.

3) 시스템 알림: "친구 추가가 정상적으로 완료되었습니다." 시스템 동작: 친구 추가 버튼을 비활성화 한다.

D.3. Search Friend

1) User: 친구의 Nickname 을 입력하여 검색을 시도한다.

2) 시스템 동작: 입력받은 Nickname 을 기반으로 Member DB 에서 친구의 정보가 존재하는지를 확인한다.

3) (탐색 성공) 시스템 동작: Member DB 에서 친구의 ID 와 Nickname 을 가져온다.

4) (탐색 실패) 시스템 알림: "일치하는 결과가 없습니다." 시스템 동작: 알림을 띄운 뒤, 초기화면으로 돌아간다.

E. Preferred Posts

E.1. Extract Preferred Posts

1) User: 관심 게시글 탭을 누른다.

2-1) (관심 tag 가 있을 때) 시스템 동작: Friend DB 에 접속하여 친구의 게시물을 추출하고, 그 중 관심 tag 가 포함된 게시물을 찾아 출력한다.

2-2) (관심 tag 가 없을 때) 시스템 알림: "Tag select please." 시스템 동작: Tag 선택 창으로 이동한다.

F. Preference Management System

F.1. Preference Analysis

1) User: 취향분석을 시작한다.

2) 시스템 동작: Clothes DB 에서 임의의 사진을 가져온다.

3) User: 출력된 옷 중에서 마음에 드는 옷을 선택한다.

4) 시스템 동작: 선택한 옷을 Style DB 에 저장하고, 최종적으로 완성된 Style 을 기반으로 Preference Vector 를 생성해 Member DB 에 저장한다.

5) 시스템 동작: 완성된 Preference Vector 를 기반으로 Member DB 에 있는 다른 User 의 Vector 와 비교하여 유사한 User 의 정보를 추천친구로 표시해준다.

F.2. Preference Update

1) User: Style Maker Page 에서 옷을 선택한다.

2) 선택한 옷을 Style DB 에 추가한다. 그 후, 수정된 Style 을 기반으로 Member DB 에 있는 Preference Vector 에 새로 옷의 Vector 를 추가한다.

F.3. Recommend Friend

1) User: 추천친구를 요청한다.

2) 시스템 동작: User 의 ID 를 기반으로 Member DB 에서 User 의 Preference Vector 를 가져온다. 그 후, Member DB 에 있는 다른 User 의 Vector 와 비교하여 유사도를 계산하는 것으로 유사한 User 의 정보를 추천친구로 표시해준다.

14. Development Environment

14.1 Objectives

Developemnt Environment 에서는 개발자의 환경에 대해 설명한다. 사용한 프로그래밍 언어와 IDE에 대해 서술한다.

14.2. Programming Language & IDE

A. Programming Language

서버를 개발하기 위해서 PHP 를 사용하였다. PHP는 php hypertext Preprocessor의 약자로, 서버 측에서 실행되는 프로그래밍 언어이다. 대부분의 운영체제에서 사용할 수 있고, 생산성이 빠르기 때문에 사용하게 되었다.

또한 발생하는 데이터들을 처리하고, 저장하기 위해서 My SQL을 사용하였다. 오픈쏘스이고 공짜이기 때문에 사용하게되었다.

Python을 통해서 제공되는 tensorflow나 keras, openCV와 같은 라이브러리들을 통해서 이미지 인식을 해야하기 때문에 사용하게 되었다.

중년들의 패션친구에서는 중, 장년 층을 대상으로 하기 때문에 그 접근성의 향상을 위해서 웹보다는 모바일 어플리케이션으로 개발하기로 하였다. 그래서 안드로이드 스튜디오를 통해서 안드로이드 어플을 사용하게 되었다.

14.3 Coding rule

1. 각종 클래스의 import를 미리 하지 않는다.

안드로이드 스튜디오에서는 해당하는 코드에서 class가 존재하지 않는다면 자동으로 그 해당 class가 import 된다. 그렇기 때문에 자기 코드에서 필요한 class 만이 import될 수 있다. 그렇기 때문에 미리 class를 import를 하게 된다면 쓰지도 않을 class를 import하게될 가능성이 높아짐으로써 불필요하게 프로그램의 크기가 커질 수도 있다.

2. 임의로 클래스를 import 하지 않는다.

즉 자동생성을 통한 class의 import만을 허용한다. 왜냐하면 개발자가 임의로 class를 찾아보고 import를 하게 된다면 다른 개발자가 개발을 진행할 때, 그 class 때문에 error가 발생할 수 있다. 예를 들어 handler와 같은 경우에 java기반의 handler가 있고, android 기반의 handler가 있다. 이때 똑 같은 handler 이더라도, class의 출신이 조금 다르기 때문에 method나 attribute가 조금씩 다를 수 있다. 그래서 error가 발생하기도 한다. 그렇기 때문에 임의로 찾아보고 class를 import하지 않도록 한다.

2. 함수 단위로 주석을 단다.

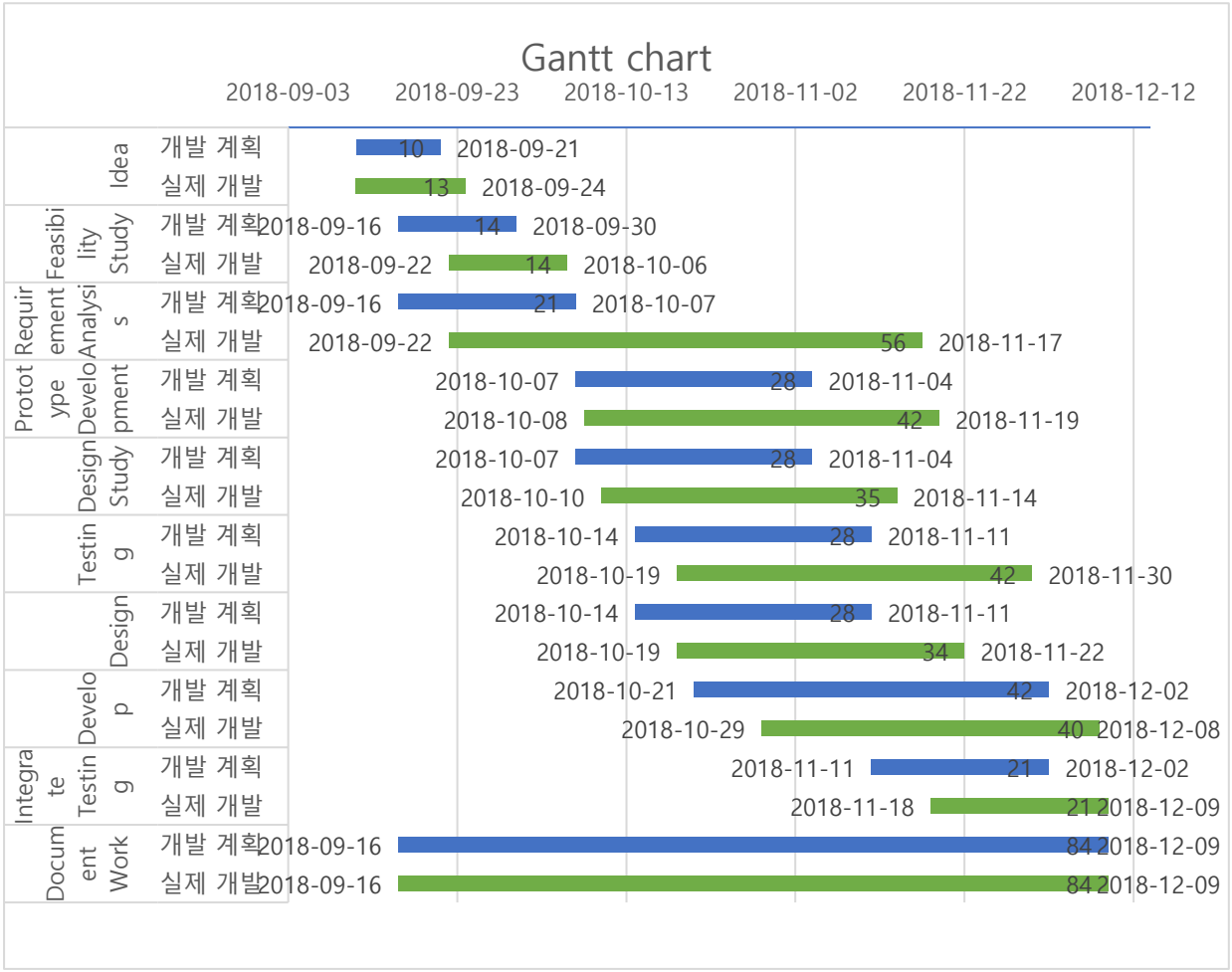
모두가 다 같이 개발을 진행하기 때문에 그 코드가 무슨 코드인지 파악할 수 있어야 한다. 그렇기 때문에 주석을 달아야 하는데 line by line으로 달기에는 시간이 너무 오래걸리기 때문에, 함수 단위로 이 함수의 input output, 이 함수의 목적 정도를 주석을 다는 것을 규칙으로 한다.

15. Development Plan

15.1 Objective

Develop plan에서는 개발 계획에 대해 서술한다. 이 때, Gantt chart를 이용하여 개발 계획과 실제 계발 흐름에 대해 서술하고자 한다.

15.2 Gantt Chart



16. Index