프로젝트 선비
Design Specification

Group 1
2011314730 김용성
2013313184 서원빈
2013314600 이영우
2014311876 조승진
2013314750 허만갑

2019.05.19 ver 1

1. Preface

1.1 Objectives

Preface에서는 본 문서의 대상 독자들과, 문서의 전반적인 구조, 각 부분의 역할에 대하여 제시한다. 그리고 버전(Version) 관리 정책, 버전 업데이트 기록, 그리고 문서 변경 사항들을 서술한다

1.2 Readership

본 문서의 대상 독자는 다음과 같다. 시스템을 직접 개발하는 소프트웨어 엔지니어, 시스템을 설계하는 아키텍처, 그 외 개발에 참여하는 모든 구성원을 독자로 정의한다. 만약, 시스템의 개발에 있어서 외주 업체를 이용한다면, 해당 업체 내에서 개발에 관련되는 모든 구성원 역시 독자에 포함된다. 다시 말해, 본 문서의 독자는 본 문서에서 소개하는 시스템의 개발 및 유지 보수에 관련된 모든 구성원이다

1.3 Document Structure

A. Preface

Preface에서는 본 문서의 독자를 정의하고, 문서의 전반적인 구조를 소개하며 각 목차의 목적을 서술한다. 또, 문서의 버전에 대해 버전 포맷, 버전 관 리 정책, 버전 업데이트 시 변경 사항 기록등을 서술한다.

B. Introduction

Introduction에서는 본 문서에서 시스템 설계에 사용하는 모든 종류의 다이어그램 및 툴에 관해 소개하고 서술한다.

C. System Architecture

System Architecture에서는 팀에서 개발하고자 하는 목표 시스템에 대해 고수준의 개요를 제시한다. 시스템의 구조와 시스템 기능의 전체적인 분포를 보여준다. 시스템의 구조를 Block diagram으로 나타내어, 각각의 관계와 실제 어떻게 사용되는지를 Package diagram과 Deployment diagram으로 표현한다. 각 서브 시스템에 대한 modular decomposition은 4~5장에 걸쳐 설명한다.

D. User management System

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

E. Filtering System

프로젝트 선비의 사용자들이 선정적인 요소를 제거 하고 싶은 동영상을 제공하면 그 동영상에서 선정적인 부분을 감지하고 블러링하고 필터링하는 시스템이다. 주어진 데이터 셋으로 학습한 인공지능을 통해 선정적인 부분을 감지한다. 사용자는 블러링, 필터링하고 싶은 부분을 시스템이 제공하는 리스트를 통해 확인하고 원하는 부분만 선택할 수 있다.
Filtering System 은 사용자가 업로드한 동영상에서 선정적인 부분을 탐색하는 Detection subsystem, 선정적인 부분은 흐릿하게 만드는 Blur subsystem 그리고 사용자의 입력을 받아 Filtering 결과물을 만드는 Filtering subsystem 으로 구성되어 있다. 데이터 베이스에 접근하여 해당 정보를 출력하는 시스템의 설계 구조를 Class diagram과 Sequence diagram, State diagram을 통해 표현하고 설명한다.

F. Protocol Design

Protocol Design에서는 각 각의 하위 시스템들이 상호작용 하는 프로토콜에 대하여 기술한다. Protocol의 기본적인 형식은 JSON으로 정하며, 통신 과정에서의 메시지(Message) 형식과 그 용도, 의미 등을 설명한다.

G. Database Design

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

H. Testing Plan

Testing Plan에서는 Test Policy과 Test Case에 대해 설명한다. 본 장의 목적은 전체 시스템이 의도한대로 실행되는지를 검증하기 위한 과정을 사전에 계획하는 데에 있다. Testing

Policy에서는 testing에 대한 단계적 접근을 설명 한다. Test Case에서는 각 Sub-System에서의 예시 케이스를 토대로 사용자를 판별하고 그에 따라 시스템에 기대하는 입, 출력 동작을 서술한다.

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

2. Introduction

2.1 Objectives

Introduction에서는 본 문서에서 시스템을 설계할 때 사용하는 모든 종류의 다이어그 램 및 툴에 관해 소개하고 설명한다.

2.2 Applied Diagram

A. **UML**
통합 모델링 언어 (Unified Modeling Language, 이하 UML)는 객체 지향 소프트웨어 설계를 위해 사용되던 여러 종류의 다이어그램들을 합하여 만든 모델링 표기법으로, 시스템 개발 과정에서 개발자 간의 의사소통이 원활하게 이루어지게 하기 위한 객체 지향적 분석과 설계 방법론의 표준 지정을 목표로 하고 있다. 1994년 소프트웨어 방법론의 선구자인 Grady Booch, James Rumbaugh, Ivar Jacobson에 의해서 연구되었고, 1997년 객체관리그 룹(OMG, Object Management Group)에서 객체 모델링 기술 (OMT;object modeling technique), OOSE 방법론 등을 통합하여 UML을 발표하 였다.
UML은 요구 분석, 시스템 설계, 시스템 구현 등의 과정에서 생길 수 있는 개발자 간의 의사 소통 불일치를 해소할 수 있다는 장점 덕분에, 현재 객체지향 시스템 개발 분야에서 가장 우수한 모델링 언어로 인식되고 있다. UML은 개발자가 구축하고자 하는 소프트웨어 시스템을 구현하기에 앞서서 표준화되고 이해가 쉬운 방식으로 설계되어, 소프트웨어에 관련된 모든 사람들과의 상호 소통을 가능하게 하는 효율적인 매개체 역할을 한다. 이 때문에 생략되거나 불일치한 모델링 구조에 대한 지적도 용이하고, 그 스스로 모델링에 대한 표현력이

강하고 비교적 모순이 적은 논리적인 표기법(notation)을 가진 언어라는 장점이 있다.
UML 이전의 시스템 개발에서 구현에 가장 큰 영향을 미치는 요인은, 시스템 의뢰인과 개발자 사이의 의사소통이 아닌 개발자들의 자의적인 이해와 논리였다. 그 결과로 개발된 시스템이 의뢰인의 입장에서 만족스럽지 않거나 의뢰인의 의도에 부합하지 않는 시스템이 개발되는 것은 필연적인 일이었다.
이러한 문제를 보완하기 위해서는, 시스템 구축 이전에 의뢰인과 개발자 사이의 충분한 의사소통과, 이를통해 개발하려는 시스템의 기능과 동작을 확실하고 철저하게 계획하는 과정이 필수적이다. 현재의 시스템 개발에는 수행하는 일이 다른 많은 팀이 필요하고, 하나의 팀은 다시 여러 개인으로 구성된다. 팀의 각 멤버는 자신의 역할을 충분히 파악해야 하고, 그 역할을 막론한 모든 사람들이 이해하고 동의할 수 있는 방법으로 설계 과정을 조직화해야 한다. 수행 업무가 다른 여러 팀들 사이에서의 충분한 소통 및 시스템 설계 과정의 표준화 또한 완전한 시스템 개발을 위해 필요로 되는 사항이다. 이에 더해, 의뢰인은 개발팀이 어떤 일을 해야 하는지 이해하고, 개발팀이 의뢰인의 요구를 제대로 이해하지 못했을 경우에는 변경사항을 지적해 줄 수 있어야한다. 시스템 개발에서 분석가, 의뢰인, 개발자가 서로 오해 없이 이해할 수 있고, 시스템 설계를 일반적으로 표현할 수 있는 수단에 대한 필요성이 중요해진 것이다. UML은 바로 이러한 표준화 수단을 제공하기 위해 마련한 표기법이다.
UML은 유스 케이스(use case) 다이어그램, 클래스 다이어그램 등 8개의 다이어그램을 기반으로 객체지향 소프트웨어를 개발하기 위한 풍부한 분석 및 설계 장치를 제공하고 있기 때문에 향후 상당 기간 동안 산업계의 표준으로 활용될 것이라 예상된다. 또한, 구축할 시스템의 유형에 관계 없이 모든 시스템에 대해 적용될 수 있고, 시스템 개발 방법론, 개발에 사용할 프로그래밍 언어, CASE 도구에 관계 없이 적용될 수 있는 일반적인 표현 방법이기 때문에 UML을 이용하여 시스템 설계를 하였다.

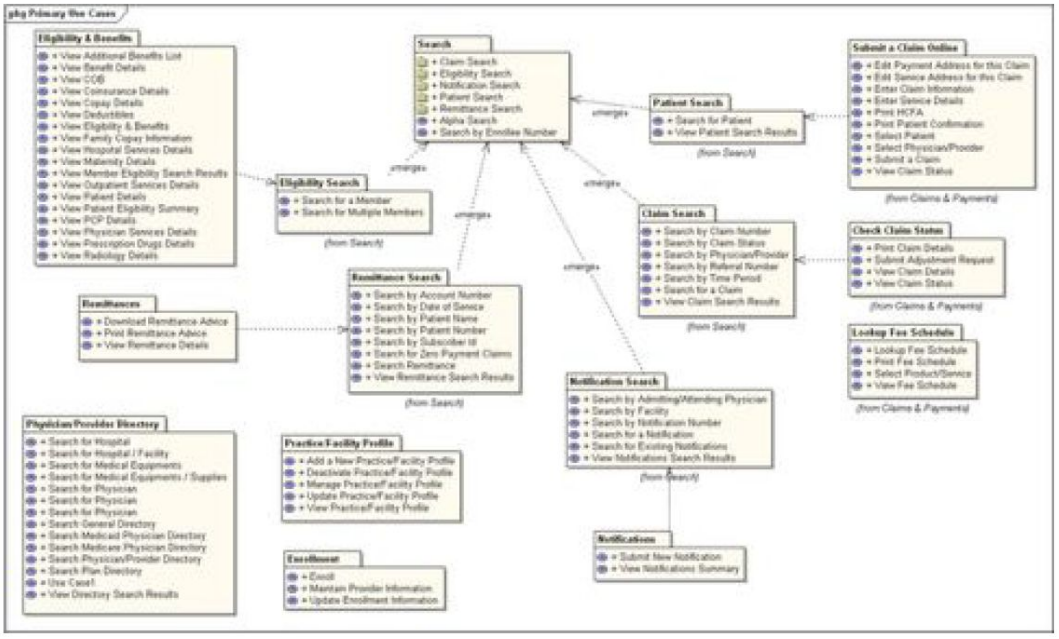
UML (United Modeling Language) Logo

B. Package Diagram

문제 도메인에 대해 더 깊이 이해할수록 개념적 모델이 더 복잡해지고 관리할 수 없을 정도로 크기가 커질 수 있다. 이렇게 복잡해진 시스템을 이해하기 위해서는 추상적인 개념들을 하나의 그룹으로 묶어서 용이하게 할 수 있다. 이렇게 만든 그룹들은 각각 쿨래스와 같은 개념으로 많은 인스턴스들을 가지고, 오로지 시스템을이해하기 위한 목적으로만 존재하게 된다. 이러한 그룹을 패키지(Package)라고 하는데, 패키지를 사용하면 전체 모델을 크기가 보다 작고 관리하기 쉬운 하위 집합으로 나눌 수 있다.

패키지는 UML 시스템 모델의 기본 구성 요소로, 요소들을 그룹으로 조직하기 위한 범용 메커니즘으로써 시스템 모델의 요소들을 조직하고 이해할 수 있도록 해준다. 패키지에 안에 담기는 것은 클래스에만 국한되지 않고, 하위 패키지들과 Use Case, Activity Diagram 등도 담을 수 있으며, 패키지의 표시 여부는 물론 패키지가 포함하는 요소의 표시 여부를 설정할 수 있다. 패키지를 구성할 때에는 여러 사람이 동의할 수 있는 형태로 구성되어야 하며, 패키지의 구성과 이름 체계는 개발자들이 쉽게 이해하고 사용할 수 있어야 한다.

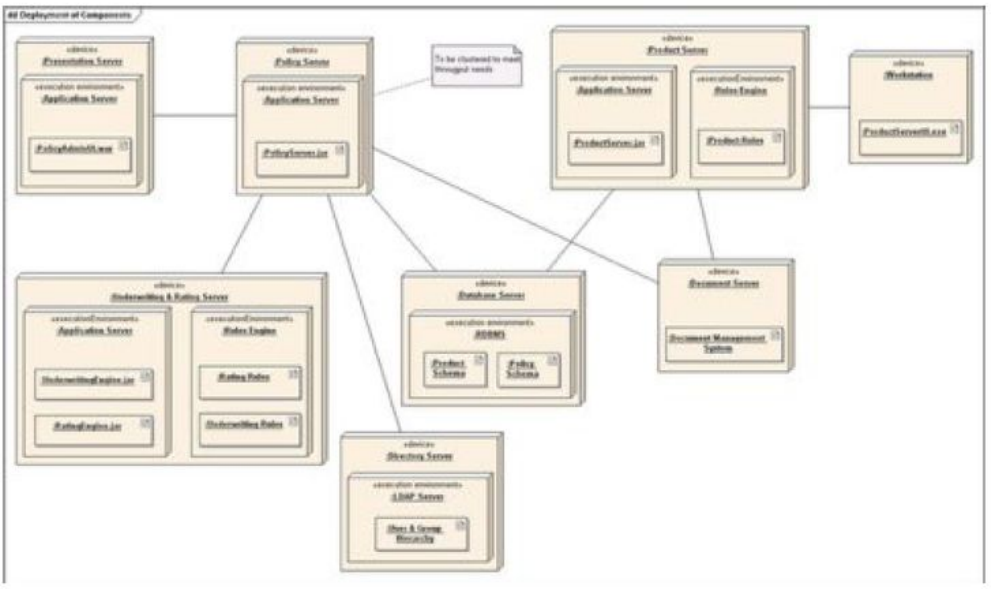
패키지 내부의 모든 클래스들은 다양한 기능적 측면에서 유사한 면을 가진다. 패키지 내부의 클래스들은 서로 밀접한 관련성을 가지며, 다른 패키지의 클래스들과는 의존관계가 약하다. 이와 같이 패키지 다이어그램은 패키지 삽입 및 패키지 확장을 포함하여 모델 요소들을 패키지로 그룹핑함으로써 조직 및 요소의 독립성을 묘사하고, 요소들간의 관계를 시각화하여 제공한다.



Package Diagram Example

C. Deploy Diagram

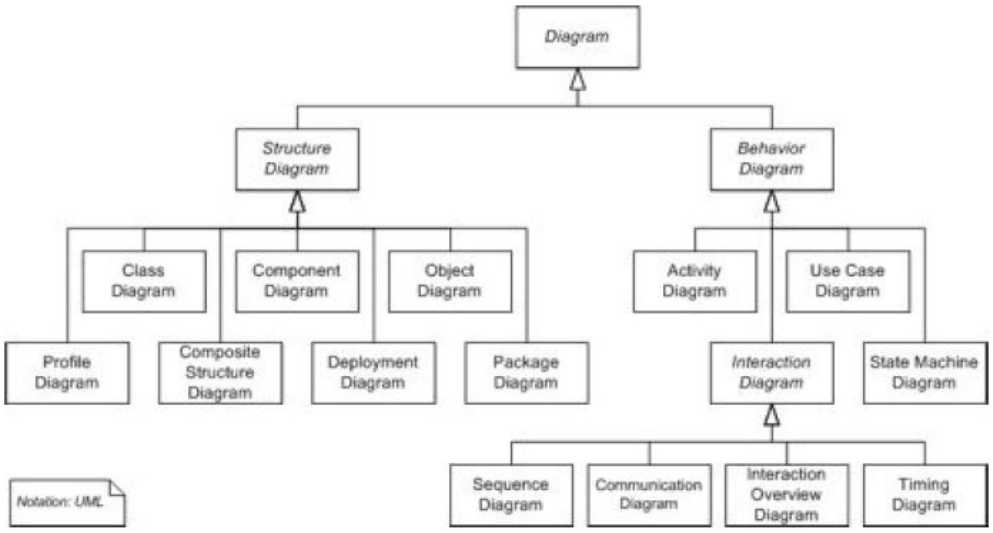
배치 다이어그램(Deployment Diagram)은 네트워크, 하드웨어 또는 소프트웨어들을 실행 파일 수준의 컴포넌트들과 함께 표현한 다이어그램이다. 이들은 시스템을 구성하는 하드웨어간의 연결 관계를 표현하고, 하드웨어 자원에 대한 소프트웨어 컴포넌트의 배치 상태를 표현한다. 즉, 시스템이 실행되는 환경인 노드와 그 노드에 배치된 컴포넌트의 구성, 각 노드들 사이의 네트워크 특성이나 프로토콜과 같은 관계를 나타내는 다이어그램이다. 노드는 처리 능력을 가진 장치를 의미하며, 각각의 노드에는 프로세서나 디바이스들에 대한 사항을 표현하고 Deployment Diagram에서 직육면체로 표기한다. Deployment Diagram은 시스템의 설계 단계의 마지막에 작성되는데, 이는 모든 설계가 마무리 되어야 소프트웨어의 컴포넌트가 정의되고, 시스템의 하드웨어 사양도 확정되기 때문이다. 배치 다이어그램은 소프트웨어 시스템이 배치, 실행될 하드웨어의 자원들을 정의하고, 소프트웨어의 컴포넌트가 어떤 하드웨어 자원에 탑재되어 실행될지를 정의하는 목적을 가지고 있다. Deployment Diagram은 운영 환경이 중요한 요소로 평가되는 개발 과정에서는 많은 Use Case 기술서나 클래스다이어그램들보다도 그 의미가 더 중요한 문서가 될 수도 있다.



Deployment Diagram Example

D. Class Diagram

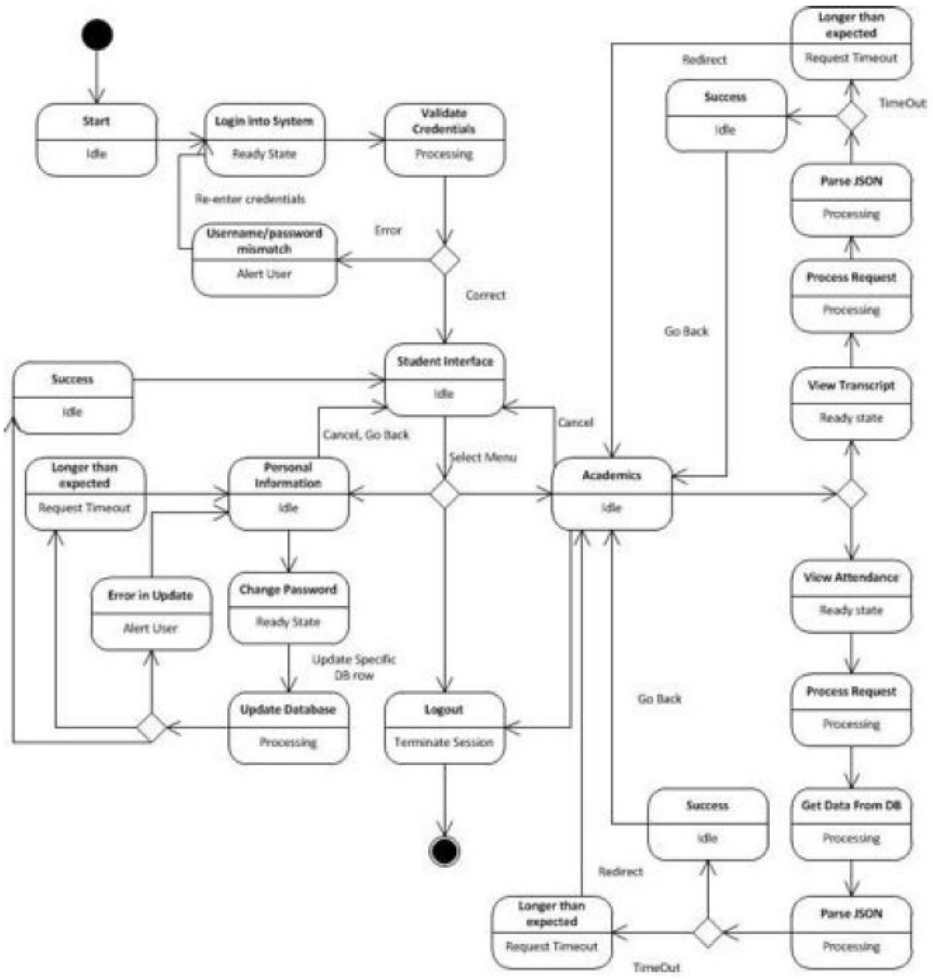
Class Diagram은 객체지향 모델링에서 가장 널리 사용되는 다이어그램으로, 시스템의 정적인 상태인 논리적인 구조를 표현하는 다이어그램이다. 클래스(Class)란 객체지향 프로그래밍에서 속성(Attribute)과 행위(메소드(Method), 또는 멤버 함수 (Member Function))를 갖는 하나의 객체 단위이다. 시스템의 클래스와 클래스 Attribute, 클래스들의 관계를 나타낼 수 있기 때문에 객체지향 시스템의 상속 등의 관계를 명확하게 표현할 수 있다. Class Diagram은 객체 지향 다이어그램에서 가장 중요한 요소이며, 코드 생성의 직접적인 원인이 되기 때문에 프로그래밍 개념과 같은 의미의 표현(Notation)을 통해 도식화한다.



Class Diagram Example

E. State Diagram

State Diagram은 객체지향 모델링에서 클래스의 인스턴스 건(Event)에 의거한 시스템의 전체적인 작동을 상세히 기술하는 다이어그램이다. State Diagram은 상태의 변화에 의한 동작 또는 하나의 상태에서 다른 상태로 변화되게 하는 사건의 주어진 시간 동안의 상태를 나타낸다. 다시 말해, State Diagram은 상태와 상태의 변화를 포함하는데, 이러한 표기는 실제 구현에서 UI를 정의하는 데 도움을 준다.
상태 다이어그램은 어떤 이벤트에 대한 반응적인(Reactive) 특성을 가지는 객체에 대해 기술하기 위해 이용되며, 객체가 갖는 여러 가지 상태를 표현한다. 객체는 기존 상태에 따라 동일한 메시지에 대해서 다른 행동을 보이기 때문에, 상태 다이어그램을 통해 시스템 내의 객체가 가질 수 있는 상태가 어떤 것이 있는지에 대해, 또 각 상태를 나타낼 때 특정 이벤트에 대해 어떤 반응을 보일지에 대해 기술한다.



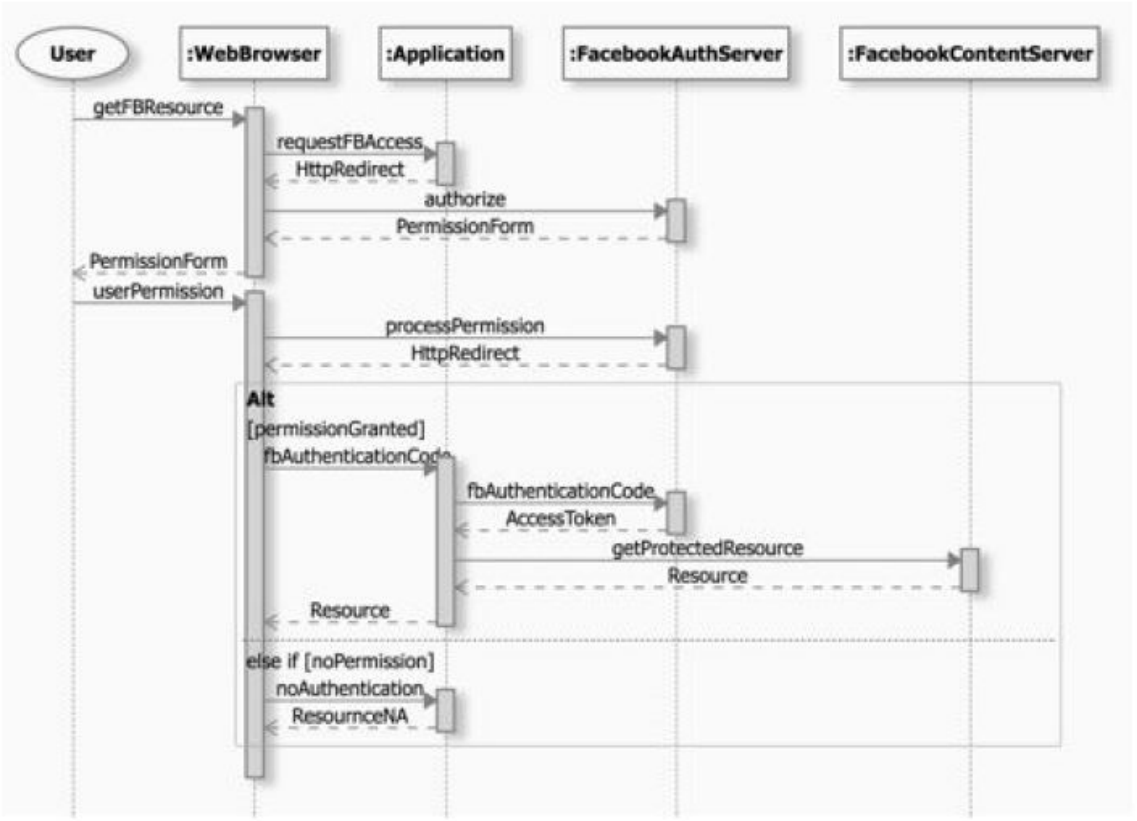
State Diagram Example

F. Sequence Diagram

Sequence Diagram은 시스템 내에서의 각 컴포넌트들이 주고 받는 메시지의 흐름을 시간 순차적으로 표현하는 상호작용 다이어그램이다. 상호작용 다이어그램은 다이어그램의 수직 방향이 시간을 나타내고, 메시지 교류의 주체인 객체와, 객체를 통해 실질적인 데이터를 주고받는 메시지(오퍼레이터)를 보여주는 역할을 하며, 그 구성 요소로는 Actor, 객체, 메시지, 회귀 메시지, 제어 블록 등이 있다.

Sequence Diagram은 각 컴포넌트들의 상호작용이 명확히 보이기 때문에 각 컴포넌트간의 관계와 각 컴포넌트들이 갖고 있는 속성, 행동들을 더욱 명확히 할 수 있다. Sequence Diagram은 시스템 내부의 동적인 움직임을 표현해주는 다이어그램이기 때문에, 속성 및 함수로 이루어진 Class Diagram을 동적 프로그래밍으로 설계하는 중요한 과정이다.

Sequence Diagram은 Use-Case Diagram과 자주 같이 사용되는데, 이는 각 Use-Case를 상세히 표현하는데 Sequence Diagram을 사용하는 것이 유리하기 때문이다. Sequence Diagram은 Use-Case를 실현한다. Use-Case Diagram에서는 시스템이 제공해야 하는 서비스를 정의하기 때문에, Use-Case는 프로그램으로 구현되기 전에 Sequence Diagram으로 설계되어야 한다. 따라서 Sequence Diagram은 각 Use-Case별로 작성되며 Use-Case에 필요한 객체가 주인공으로 등장하고, 객체 간 메시지를 통해서 Use-Case의 기능이 실현된다. Sequence Diagram에서는 객체의 오퍼레이션과 속성을 상세히 정의한다. 이는 객체간 상호작용을 정의하는 과정에서 객체들이 가져야 하는 오퍼레이션과 속성을 구체적으로 정의할 필요가 있기 때문이다. 객체는 다른 객체가 의뢰하는 일을 처리하는 객체의 책임으로서, 객체의 책임은 오퍼레이션으로 정의되어야 하며, 이 행위를 위해 필요한 객체의 속성도 정의되어야 한다.

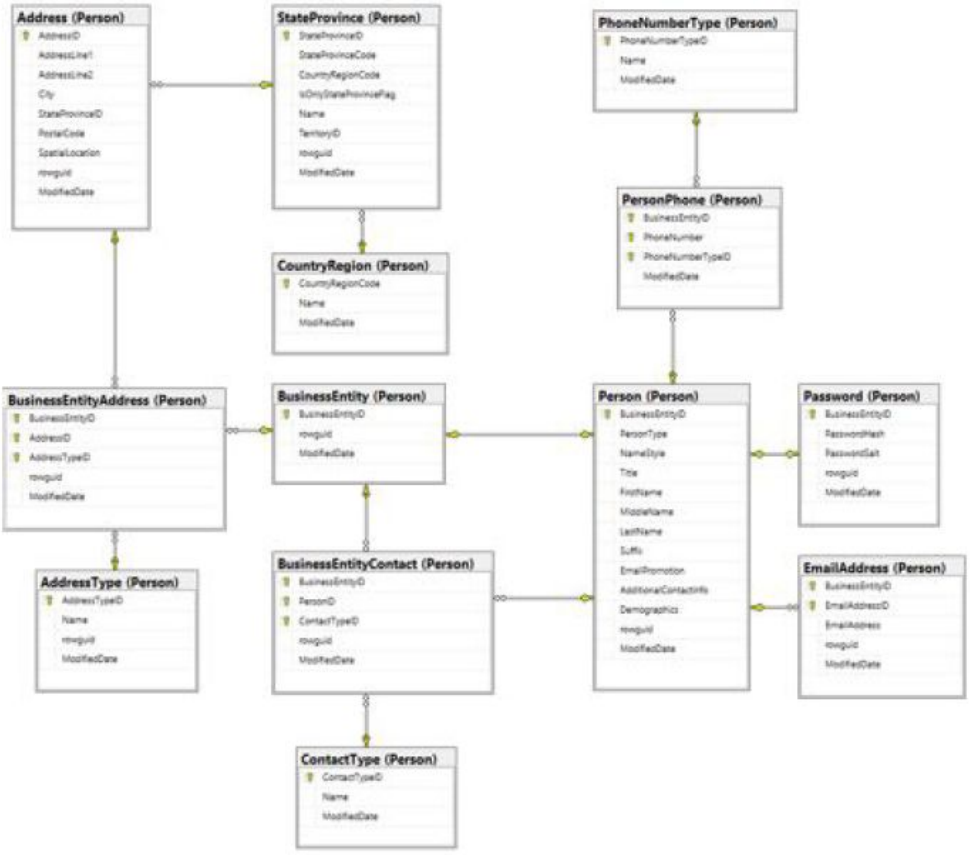


Sequence Diagram Example

G. ER Diagram

ER Diagram은 데이터베이스에서 구조화된 각 데이터 개체들의 관계를

표현하기 위해 사용하는 다이어그램으로, UML에 포함되지는 않는다. 데이터가 저장되는 공간인 데이터베이스의 구조 및 그에 수반한 제약 조건들은 다양한 기법에 의해 정의될 수 있는데, 그 기법 중 하나가 개체- 관계 모델링(Entity-Relationship Modelling)이다. 이 ER모델링 프로세스의 산출물이 ER Diagram(Entity-Relationship Diagram)인데, 여기에서 개체(Entity)란 현실 세계의 객체로서 유형 또는 무형의 정보를 대상으로 서로 구별할 수 있는 것이며, 관계(Relationship)란 두 개 이상의 개체 사이에 연관성을 기술한 것이다. 따라서, ER Diagram은 조직의 정보 자원(Resource)을 전반적으로 계획하는데 있어서 필수적이며 유용한 도구이다.

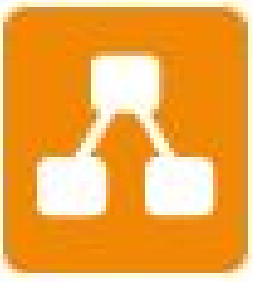


ER Diagram Example

2.3 Applied Tool

A. Draw.io

우리 팀에서는 시스템 설계도를 표현하는 데 Draw.io를 주로 사용하였다. Draw.io는 플로우 차트, 프로세스 다이어그램, 조직도, UML, ER 및 네트워크 다이어그램을 만들기 위한 무료 온라인 다이어그램 소프트웨어이다. 선 및 도형의 도식이 자유로워 설계도 구현에 있어 매우 용이하다.



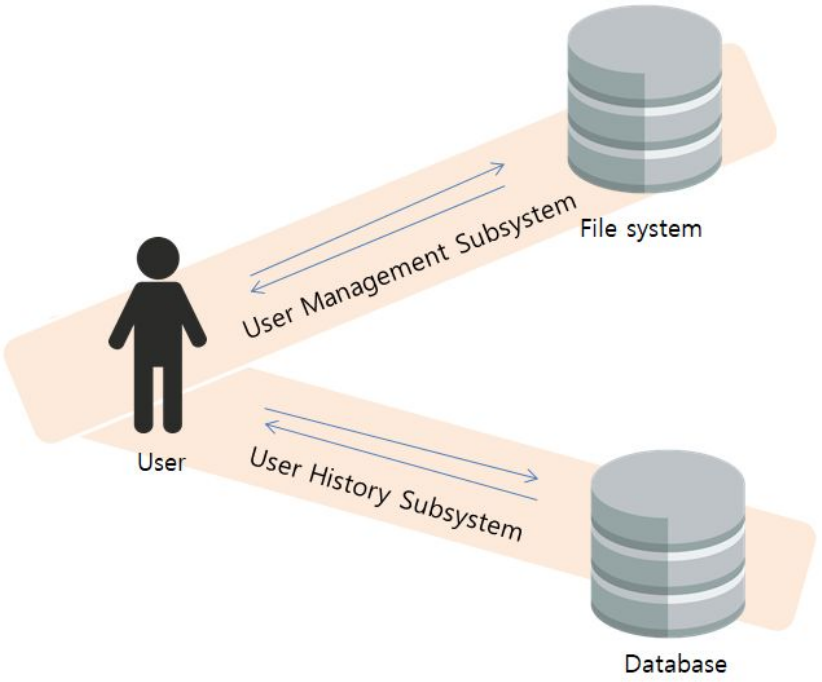
Draw.io Logo

2.4 Project Scope

프로젝트 선비 시스템은 인공지능을 활용해 동영상에서 선정적인 부분을 제거하는 시스템으로, 자동으로 선정적인 부분을 선정, 블러링, 필터링하는데 사용자가 직접적으로 선정적인 부분을 리스트를 통해 확인하고 선택할 수 있는 부가적인 기능을 제공해 사용자의 편리를 돕는 시스템이다. 프로젝트 선비는 크게 2개의 시스템으로 구성되어 있다.

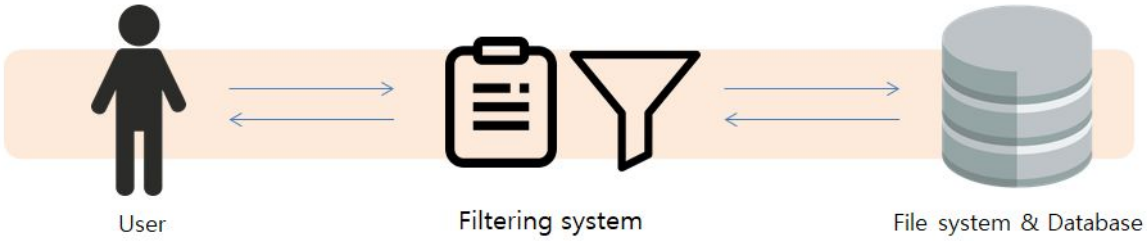
먼저 기존의 온라인 시스템에서 제공하는 기능을 기반으로 설계된 시스템으로 User Management System가 있다.

User Management System의 경우, 사용자의 등록, 이용 등과 관련된 시스템으로 하위 2개의 시스템으로 분류된다. 사용자의 기본 정보를 관리하는 User Management subsystem 과 사용자의 사용 기록을 관리하는 User History subsystem 으로 구성되어 있다. 해당 시스템의 구조는 다음과 같다.



User Management System

Filtering system의 경우, 사용자가 업로드한 동영상에서 선정적인 부분을 탐색하는 시스템으로 프로젝트 선비 서비스의 특화 시스템이다. Filtering system은 다시 하위 3개의 시스템으로 분류된다. 이는 Detection subsystem, Blur subsystem, Filtering subsystem 으로 구성되어 있는데 먼저 Detection system은 일정 범위의 데이터베이스를 기반으로 알고리즘을 통해 코디를 추천하는 시스템으로 주요 기능을 담당하게 되며 Blur system과 Filtering system은 사용자가 선택한 부분을 blur하고 filter하는 시스템이다. 하위 시스템의 구조는 다음과 같다.



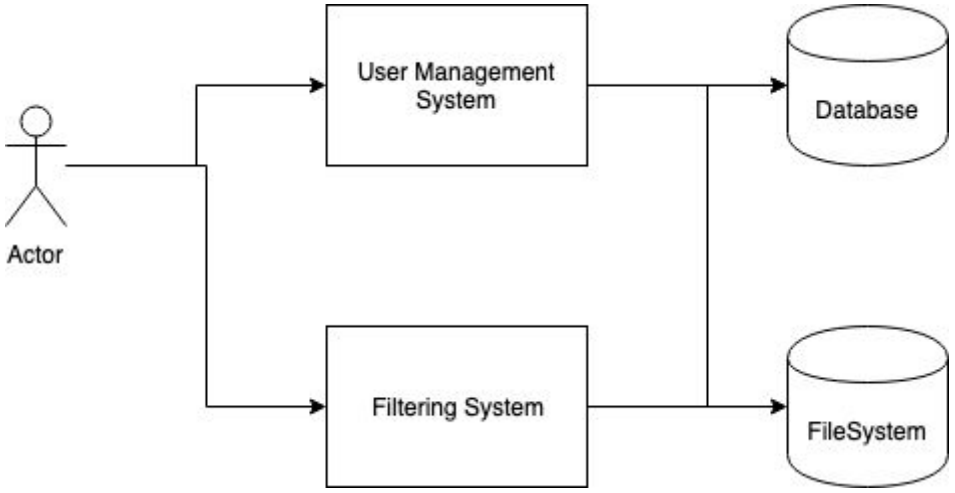
Filtering system

3. System Architecture

3.1 Objectives

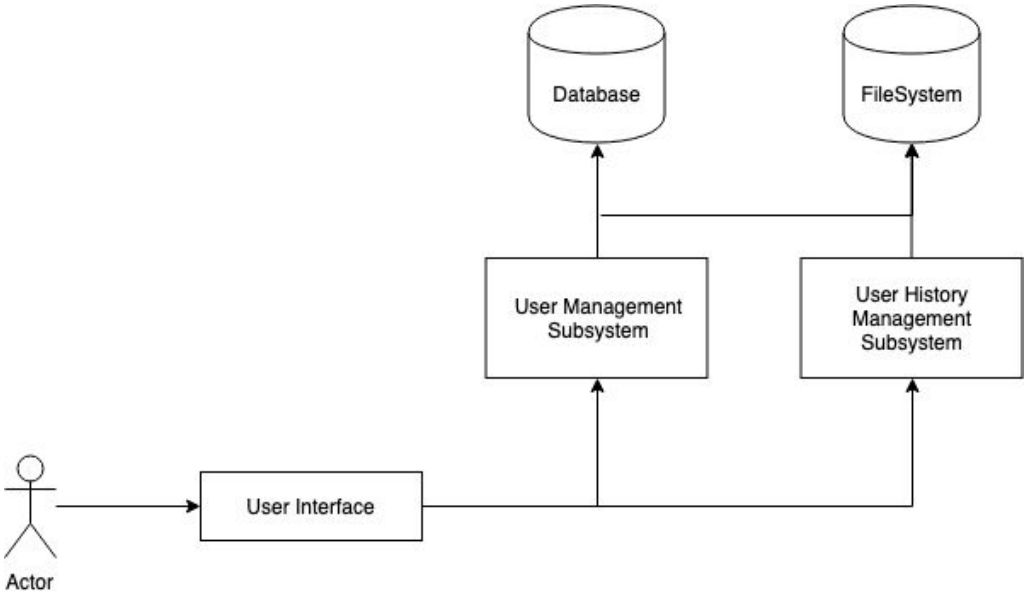
System Architecture 는 프로젝트의 전반적인 시스템 구조에 대해 서술한다. 또한 기능에 대해 다양한 diagram 을 통해 시스템이 어떻게 구성되는지 서술한다.

3.2 System Organization



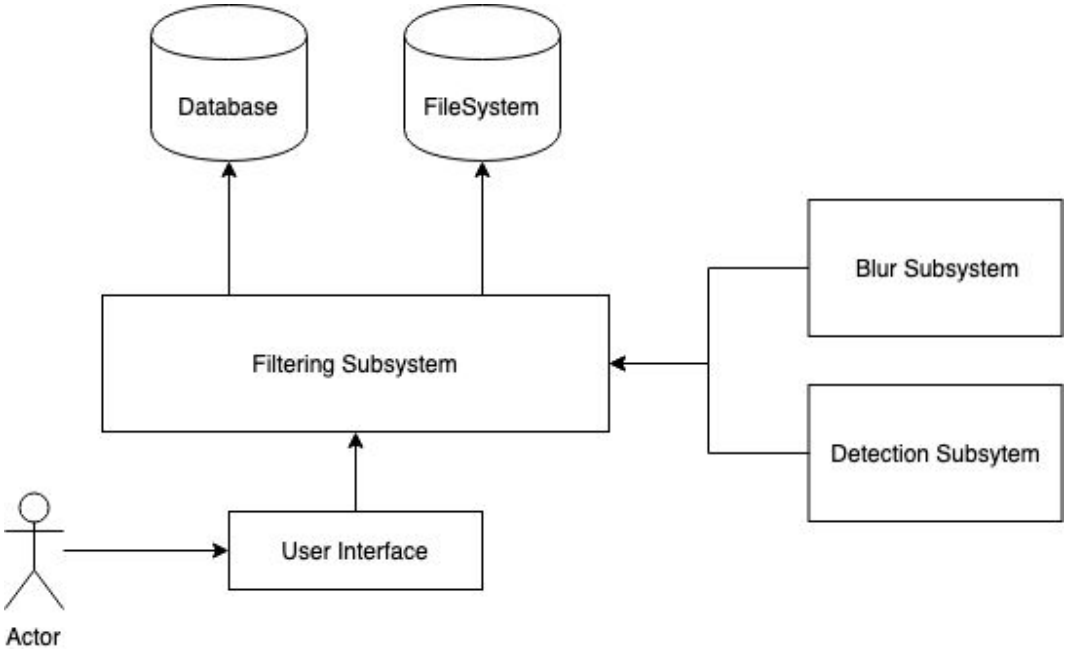
프로젝트 선비는 주로 영상 선정적인 부분을 자동으로 탐색하고 선정적인 부분을 흐릿하게 하는 데에 초점이 맞춰져 있다. 유저와 유저가 업로드한 동영상을 관리하는 User Management System 과 유저가 올린 동영상을 처리하는 Filtering System 으로 구성된다.

A. **User Management System**
   User Management System 은 사용자의 정보를 관리하는 시스템이다. 사용자 가입과 로그인, 지난 기록 보기 를 제공한다. 사용자의 기본 정보를 관리하는 User Management subsystem 과 사용자의 사용 기록을 관리하는 User History subsystem 으로 구성되어 있다.



B. Filtering System

Filtering System 은 동영상에서 선정적인 부분을 자동으로 탐색하고 블러링까지 넣는 시스템이다. 사용자가 업로드한 동영상에서 선정적인 부분을 탐색하는 Detection subsystem, 선정적인 부분은 흐릿하게 만드는 Blur subsystem 그리고 사용자의 입력을 받아 Filtering 결과물을 만드는 Filtering subsystem 으로 구성되어 있다.

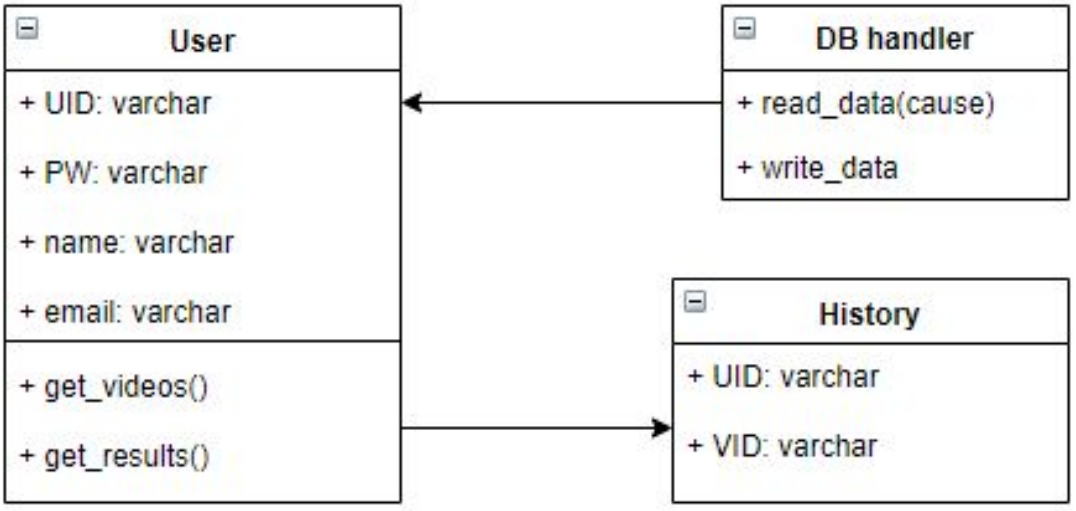


4. User Management System

4.1 Objectives

실제 사용자가 회원 가입과 로그인, 지난 기록 보기 기능 등의 시스템을 이용하는 중에 발생하는 데이터를 처리하는 과정을 도식화하여 사용자 관리 시스템의 설계를 설명한다. Class diagram, Sequence diagram, State diagram을 통해 User Management System의 구조를 표현하고 설명한다.

4.2 Class diagram



A. Account

a. Attributes

+UID : 사용자 ID값
+PW : 비밀번호
+name : 사용자 이름
+email : 이메일 주소

b. Methods

+get_videos() : DB에서 data 수신
+get_results() : DB에서 data 수신

B. DB Handler

a. Attributes

해당사항 없음.

b. Methods

+read_data(cause) : Censor DB에서 cause data를 읽어온다.
+write_data : 해당되는 DB에 data를 저장한다.

C. History

a. Attributes

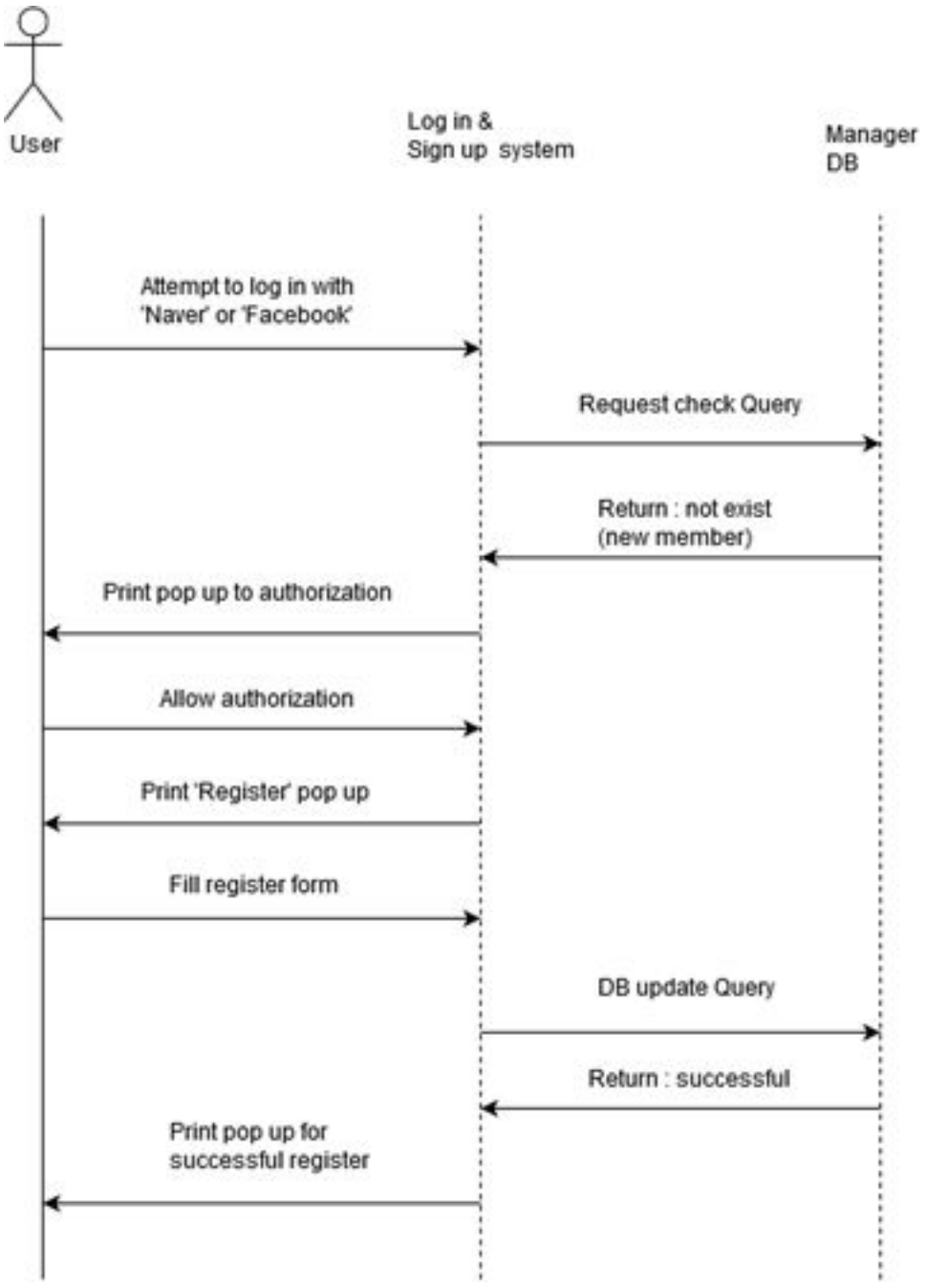
+UID : 사용자 ID값
+VID : 영상 ID

b. Methods

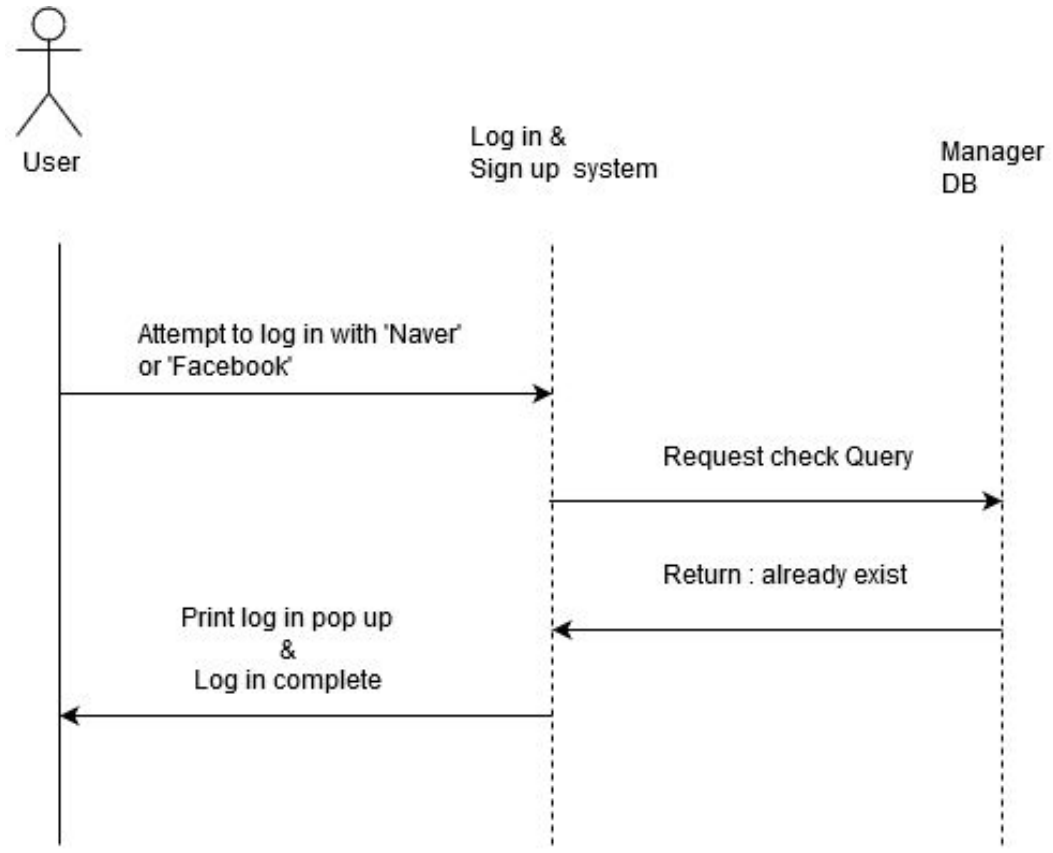
해당사항 없음.

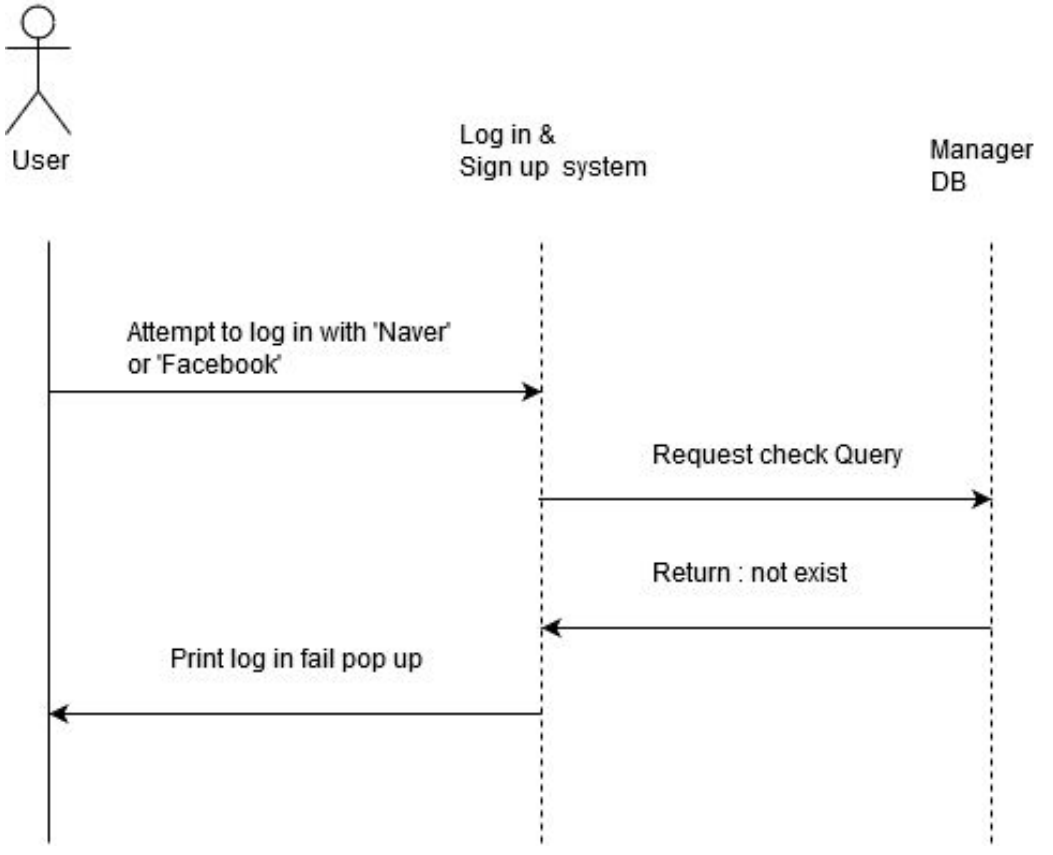
4.3 Sequence diagram

A. Sign Up using SNS(Naver/Facebook)



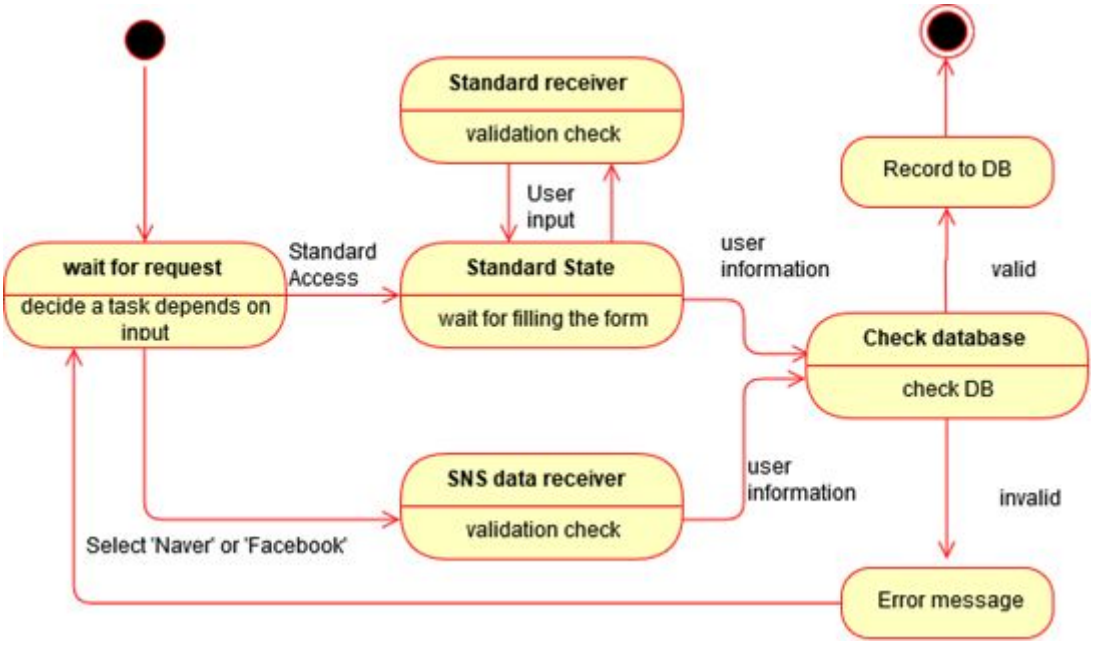
B. Sign Up using SNS(Naver/Facebook)


User
Log in &
Sign up system
Manager
DB
Attempt to log in with 'Naver'
or 'Facebook'
Request check Query
Return : already exist
Print log in pop up
&
Log in complete

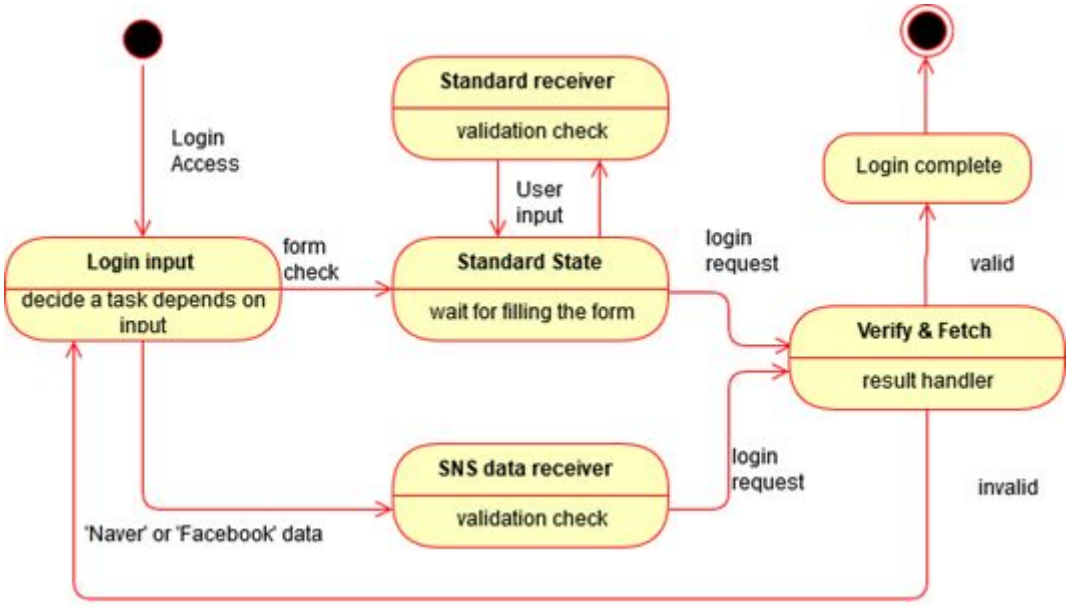

User
Log in &
Sign up system
Manager
DB
Attempt to log in with 'Naver'
or 'Facebook'
Request check Query
Return : not exist
Print log in fail pop up

4.4 State diagram

A. Sign Up using SNS



B. Sign Up without SNS

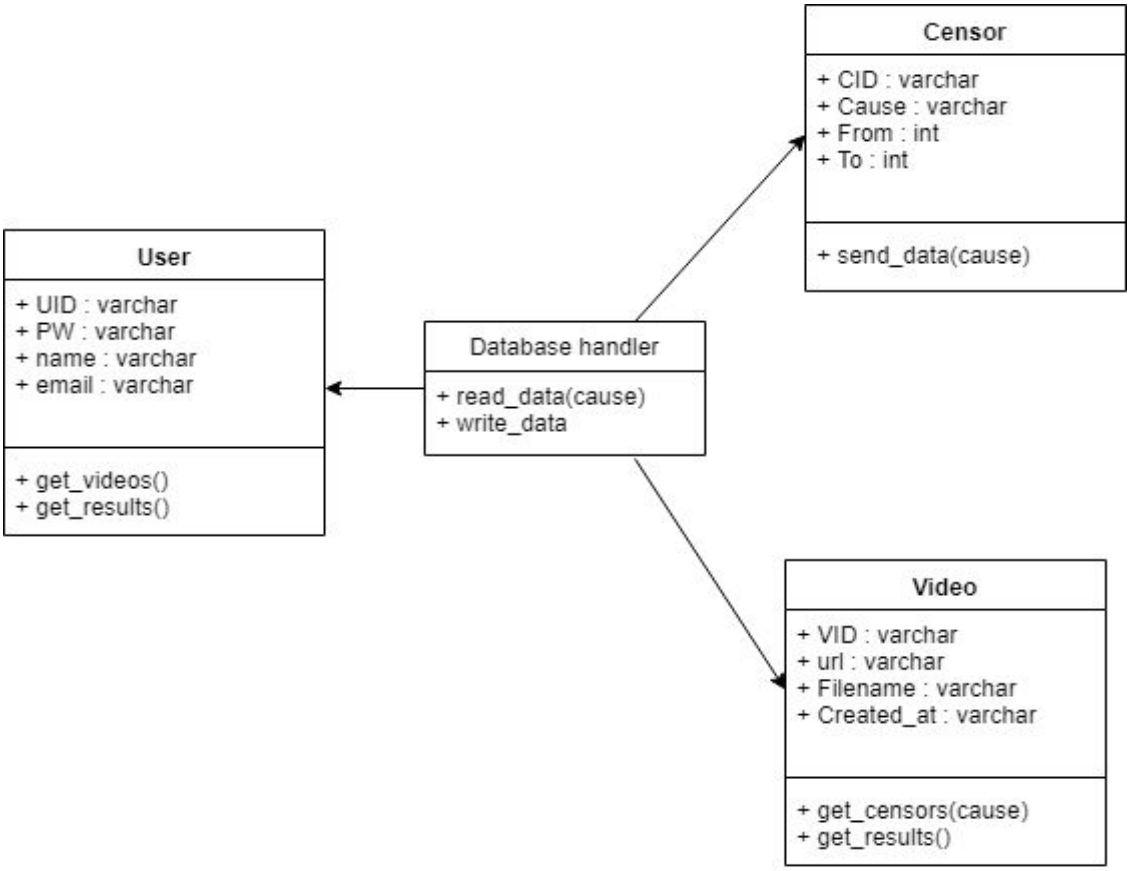


5. Filtering System

5.1 Objectives

본 장에서는 사용자가 올린 영상의 선정적인 부분을 처리해주는 필터링 시스템의 설계를 설명한다. 사용자가 영상을 업로드하면 서버에서 선정적인 부분을 감지해 블러링한다. 사용자는 블러링된 리스트를 확인하고 필터링 처리를 원하는 부분을 선택하는 시스템의 설계를 Class Diagram, Sequence Diagram, State Diagram을 통해 상호작용을 표현하고 설명한다.

5.2 Class Diagram



A. User

A.1 Attributes

+UID : 사용자 ID값
+PW : 비밀번호
+name : 사용자 이름
+email : 이메일 주소

A.2 Methods

+get_videos() : DB에서 data 수신
+get_results() : DB에서 data 수신

B. Censor

B.1 Attributes

+CID : 필터링 구간 ID
+Cause : 필터링 정보
+From : 구간 시작 시간
+To : 구간 종료 시간

B.2 Methods

+send_data(cause) : DB로 cause data 송신

C. Video

C.1 Attributes

+VID : 영상 ID
+url : 영상 주소
+Filename : 영상 이름
+Created_at : 생성 시간

C.2 Methods

+get_censors(cause) : DB에서 cause data 수신
+get_results() : DB에서 data 수신

D. DB Handler

D.1 Attributes

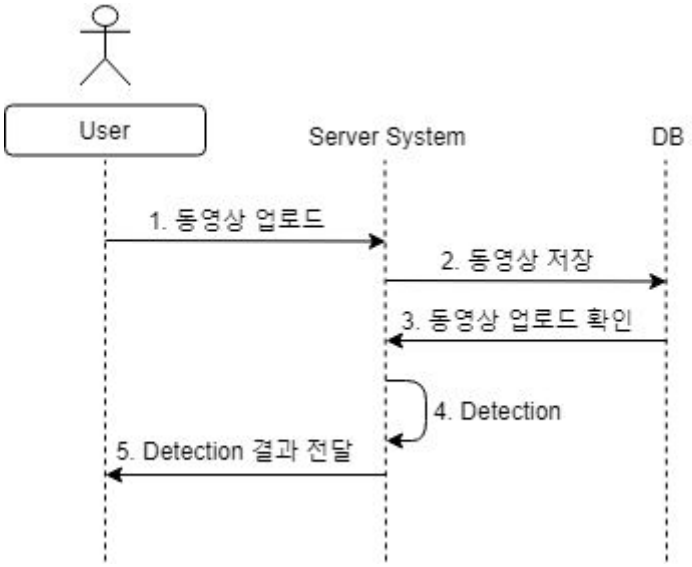
해당 사항 없음.

D.2 Methods

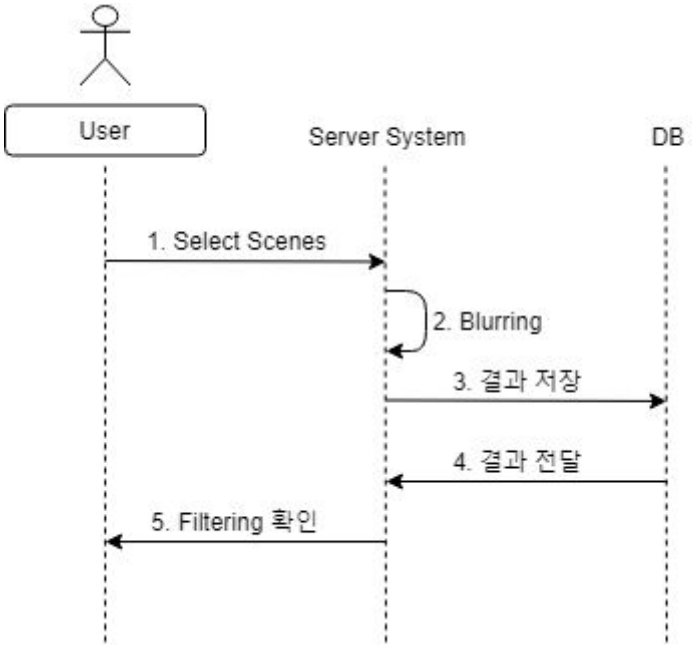
+read_data(cause) : Censor DB에서 cause data를 읽어온다.
+write_data : 해당되는 DB에 data를 저장한다.

5.3 Sequence Diagram

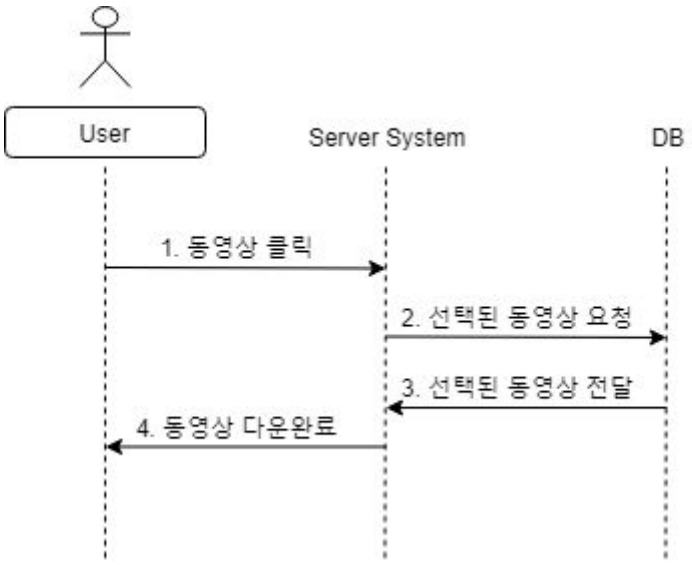
A. Upload Video



B. Select Scenes

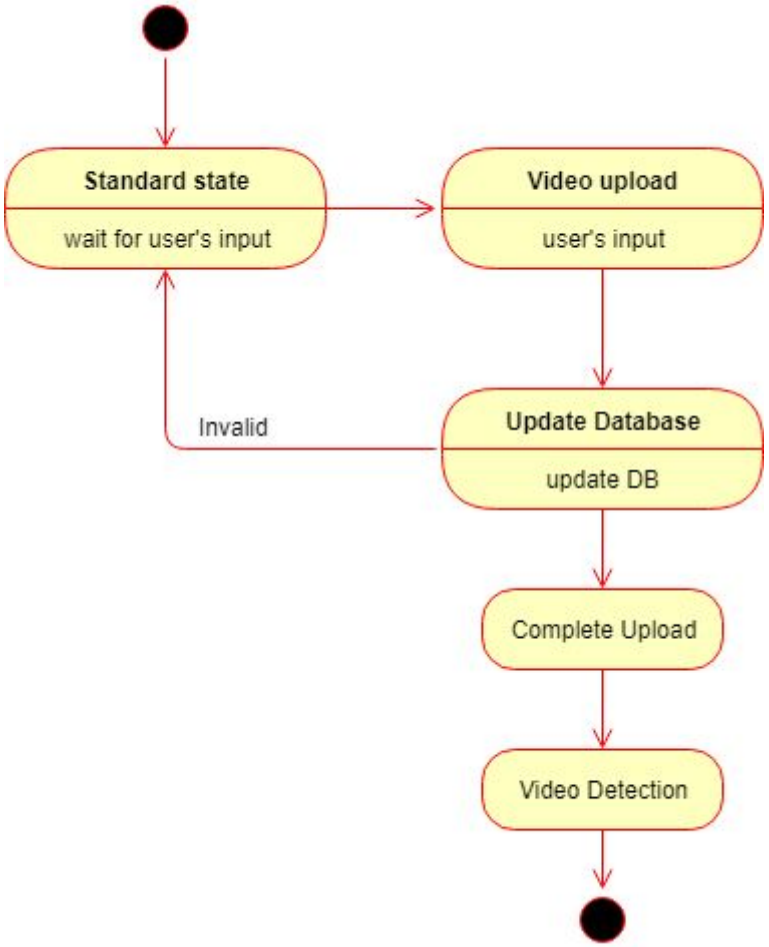


C. Download Video

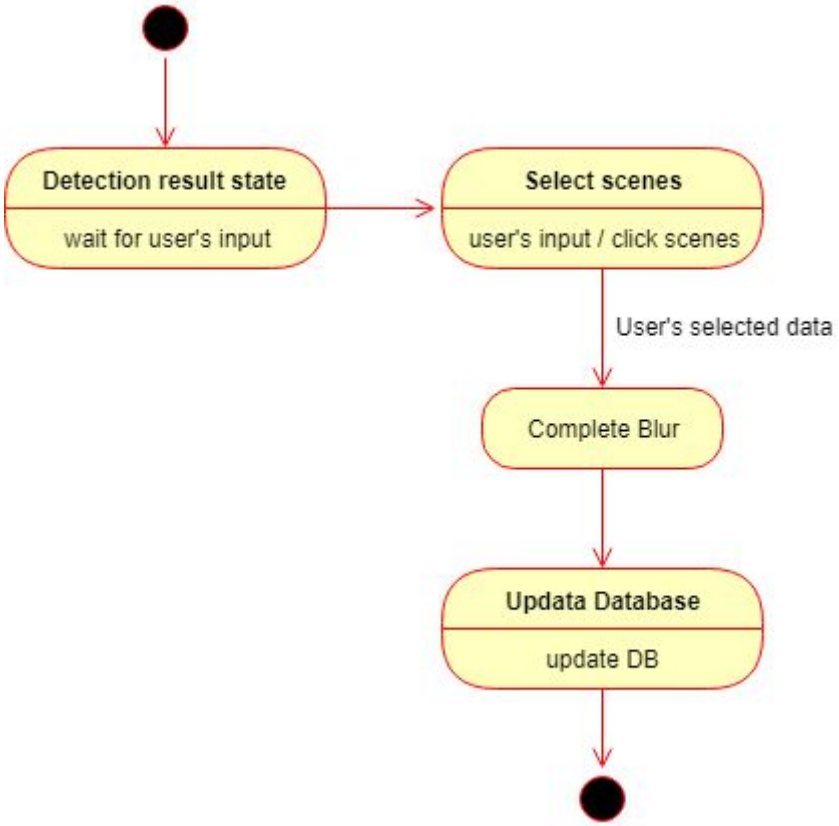


5.4 State Diagram

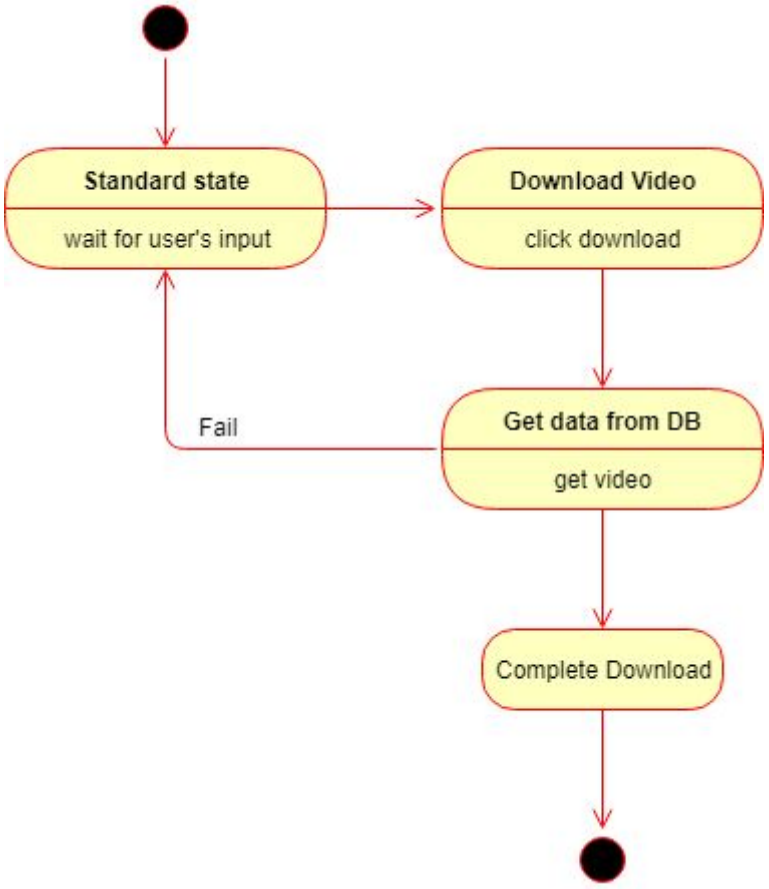
A. Upload Video



B. Select Scenes



C. Download Video



6. Protocol Design

6.1 Objectives

Protocol Design 에서는 subsystem 사이에 message 교환 프로토콜과 User 와 System 사이에 Request, Response 에 대한 프로토콜을 서술한다.

6.2 Protocol Description

subsystem 사이에 message 교환과 User 와 System 간 Request, Response 모두 JSON 포맷을 이용한다. JSON 은 JavaScript object notation 으로 key-value 쌍으로 이루어진 표준 포맷이다.

A. Sign Up

Request

- email: String
- password: String

Response
- cookie: Cookie(User 정보를 담고 있는 Cookie)

B. Sign In

Request
- email: String
- password: String

Response
- cookie: Cookie (User 정보를 담고 있는 Cookie)

C. Upload Video

Request
- video: File

Response
- process: String

D. Check Video Process

Request
- video_id: String

Response
- process: String (Processing 상태를 나타내는 정도로 DETECTING, DETECTION_COMPLETE 를 나타낸다.)

E. Select Scenes

Request
- video_id: String
- removed_censor_ids: List<String> (영상에서 제거할 구간)
- blurred_censor_ids: List<String> (blur 처리할 구간)

Response
- video: Video (Video 정보로 name, url, censors 를 포함한다.)

F. Detect

Request
- video: Video

Response
- censors: List<Censor> (탐색된 결과물로 영상 구간을 나타내는 from, to, 선택된 이유인 cause 를 포함한다.)

G. Get Videos (업로드 했던 영상들을 보기위한 History)

Request
- last_video_id: String (페이지에서 보이는 마지막 video id)
- limit: Int (response 로 받을 video 수의 max 값)

Response
- videos: List<Video>

H. Get Censors (이미 탐색된 선정적인 구간들)

Request

- video_id: String
- last_censor_id: String (페이지에서 보이는 마지막 censor_id, 없으면 맨 처음 censor list 를 보여준다.)
- limit: Int (response 로 받을 censor 수의 max 값)

Response

- censors: List<Censor>

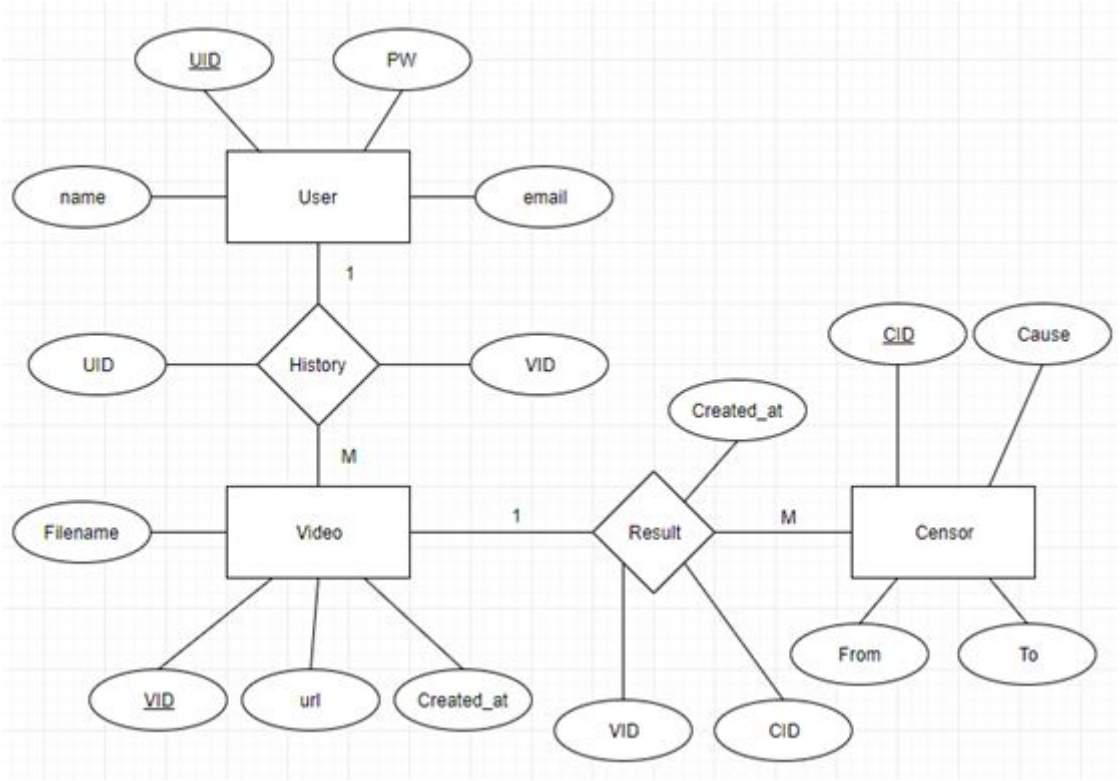
7. Database Design

7.1 Objectives

Database Design에서는 요구사항 명세서에서 기술하였던 요구사항을 기반으로 한 데이터베이스를 설계하고 이에 대하여 설명한다. 요구사항에 기반한 데이터베이스의 E-R Diagram을 제시하고, 이에 대한 Relation Schema를 서술한다. 추가적으로 SQL DDL을 작성한다.
해당 시스템에서는 회원의들의 목록과 정보 및 필터링한 영상 기록을 저장하고, 각 필터링 비디오의 필터링 구간 정보를 저장하기 위해 데이터베이스를 이용한다.

7.2 E-R Diagram

Entity는 독립된 주체 하나를 나타낸다. 각 각체 사이의 Relation 은 다이아몬드로, Entity는 사각형으로 표현된다. Relation, Entity는 속성을 가질 수 있으며, 이 특성들은 실선으로 연결된 타원형으로 표현한다. 관계는 둘 이상의 객체 간 연관 관계를 표현하고, 모든 주체는 고유하게 식별되는 속성을 가지고 있는데 최소한의 고유 식별 속성을 그 주체의 기본 키(Primary Key)라고 한다.



Overall E-R Diagram

A. Entity

A.1. User

User Entity는 각 회원이 가입을 하면서 기입한 회원에 대한 정보를 나타낸다. UID, PW, name, email의 속성으로 구성되어 있고, UID는 기본키이면서 외래키이다.

A.2. Video

Video Entity는 필터링 이후 생성된 비디오에 대한 정보를 나타낸다. VID, Filename, url, Created_at의 속성으로 구성되어 있고, Filename의 경우 영상 파일의 파일명, url은 해당 영상의 url정보, Created_at은 해당 영상의 필터링이 언제 되었는지를 나타낸다. VID의 경우 기본키이면서 외래키이다.

A.3. Censor

Censor Entity는 각 비디오에 대하여 행해진 필터링에 대한 정보이다. CID, From, To, Cause의 속성으로 구성되어 있고, From, To의 경우 필터링의 구간정보, Cause는 해당 필터링의 사유에 대한 정보이다. CID의 경우 기본키이면서 외래키이다.

B. Relationship

B.1. History

History Relationship은 각 User가 지금까지 실행한 필터링 영상에 대한 정보를 나타내고, 해당 기록을 확인 및 다시 다운로드 할 수 있도록 하기 위해 유지한다. 1명의 User가 여러 개의 필터링 영상을 소유할 수 있는 1:M의 관계이다.

B.2. Result

Result Relationship은 각 Video에 대하여 어떠한 필터링 구간에 대해 필터링 하였으며,. 1개의 Video가 여러 개의 필터링 구간을 소유할 수 있는 1:M의 관계이다.

7.3 Relational Schema

A. User

UID	PW	name	email

Primary Key(PK) : UID
Foreign Key(FK) : UID -> History(UID)
Func Dep(FD) : UID -> {PW, name, email}

Description : User의 정보를 저장하는 테이블이다. Primary Key로 UID를 가지며, UID는 외래키로도 사용되며 History에 관계된다.

B. Video

VID	Filename	url	Created_at

Primary Key(PK) : VID
Foreign Key(FK) : VID -> History(VID), Result(VID), Created_at -> Result(Created_at)
Func Dep(FD) : VID -> {Filename, url, Created_at}
Description : 필터링 된 Video의 정보를 저장하는 테이블이다. Primary Key로 VID를 가지며, VID는 외래키로도 사용되며 History, Result에 관계된다. Created_at 역시 외래키로서, Result의 필터링 구간 정보에 관계된다.

C. Censor

CID	From	To	Cause

Primary Key(PK) : CID
Foreign Key(FK) : CID -> Result(CID)
Func Dep(FD) : CID -> {From, To, Cause}
Description : 필터링 된 Video에서의 필터링 된 구간에 대한 정보를 저장하는 테이블이다. Primary Key로 CID를 가지며, CID는 외래키로도 사용되며 Result에 관계된다.

D. History

UID	VID

Primary Key(PK) : None
Foreign Key(FK) : None
Description : User 테이블의 UID, Video 테이블의 VID를 참조하여 각 유저 별 필터링한 비디오의 정보를 저장하는 테이블

E. Result

VID	CID	Created_at

Primary Key(PK) : None
Foreign Key(FK) : None
Description : Video 테이블의 VID, Created_at, Censor 테이블의 CID를 참조하여 각 비디오별 필터링 정보를 저장하는 테이블

7.4 DDL

A. User

```
CREATE TABLE User(
    UID VARCHAR(20) NOT_NULL
    PW VARCHAR(20) NOT_NULL
    name VARCHAR(20) NOT_NULL
    email VARCHAR(100) NOT_NULL
    PRIMARY KEY(UID)
);
```

B. Video

```
CREATE TABLE Video(
    VID VARCHAR(20) NOT_NULL
    Filename VARCHAR(255) NOT_NULL
    url VARCHAR(255) NOT_NULL
    Created_at VARCHAR(100) NOT_NULL
    PRIMARY KEY(VID)
);
```

C. Censor

```
CREATE TABLE Censor(
    CID VARCHAR(20) NOT_NULL
    From INT(30) NOT_NULL
    To INT(30) NOT_NULL
    Cause VARCHAR(100) NOT_NULL
```

```
    PRIMARY KEY(CID)
);
```

D. History

```
CREATE TABLE History(
    UID VARCHAR(20) NOT_NULL
    VID VARCHAR(20) NOT_NULL
    FOREIGN KEY UID REFERENCES User(UID)
    FOREIGN KEY UID REFERENCES Video(VID)
);
```

E. Result

```
CREATE TABLE Result(
    CID VARCHAR(20) NOT_NULL
    VID VARCHAR(20) NOT_NULL
    Created_at VARCHAR(100) NOT_NULL
    Cause VARCHAR(100) NOT_NULL
    FOREIGN KEY CID REFERENCES Censor(CID)
    FOREIGN KEY VID REFERENCES Video(VID)
    FOREIGN KEY VID REFERENCES Video(Created_at)
);
```

8. Testing plan

8.1 Objectives

시스템이 설계 의도대로 실행되며, 시스템 내부의 결함을 찾기 위해서 Testing Plan을 설계단계에서 계획 후 구현 과정에서 해당 Testing을 진행한다.

8.2 Testing Policy

해당 시스템의 경우 크게 3단계로 나누어 Testing을 진행하며, 해당 단계는 Developing Testing, Release Testing, User Testing와 같이 구성되어 있고, Developing Testing은Component Testing, Integration Testing, System Testing, Acceptance Testing으로 나뉜다

A. Developing Testing
개발 과정에서 수행되는 Testing으로 Component Testing의 경우 component 단위로 개발한 후에 제대로 작동하는지 확인하는 작업이고, Integrating Testing은 각 요소 들을 점진적으로 합치면서 확인하는 Testing, System Testing은 모든 하위 시스템을 합친 후, 시스템이 잘 동작하는지 확인하며, Acceptance Testing은 사용자의 정보를 이 용하여 시스템에 대한 사용자의 요구 사항을 확인하는 작업이다.

B. Release Testing
사용자에게 서비스를 출시하기 전제 최종적으로 서비스를 Testing하는 작업이고, 요구사항이 모두 반영되었는지를 확인한다.

C. User Testing
사용자의 환경에서 시스템을 Testing하는 작업이다.

8.3 Test Case

A. User Management System

A.1 Sign up

1) User : 회원가입 버튼을 누른다
2) 시스템 동작 : 회원가입 양식 페이지로 이동한다.
3) User : 회원가입 양식을 채운 뒤 가입 버튼을 누른다.
4) 시스템 동작: 모든 양식이 채워졌는지 검사한다.
    4-1) 성공
      시스템 알림 : “회원가입에 성공했습니다.”
      시스템 동작 : DB에 회원정보를 저장한다.
    4-2) 실패
      시스템 알림 : “회원가입 양식을 채워주세요.”
      시스템 동작 : 해당 페이지에 머무른다.

A.2 Sign in

1) User : ID와 PW를 기입한 뒤 로그인 버튼을 누른다.
2) 시스템 동작 : DB에 저장된 데이터와 User가 기입한 정보가 일치하는지 확인한다.

2-1) 성공
시스템 알림 : “로그인에 성공했습니다.”
시스템 동작 : 해당 User가 로그인 된다.
2-2) 실패
시스템 알림 : “ID또는 PW를 확인해주세요.”
시스템 동작 : 로그인 페이지로 돌아간다.

A.3 History

1) User : History 메뉴를 통해 해당 페이지로 이동한다.
2) 시스템 동작 : DB에 해당 User의 ID를 이용하여, 필터링 피디오 목록을 가져온다.
2-1) 성공
시스템 동작 : 해당 User의 History 정보를 보여준다.
2-2) 실패
시스템 동작 : 해당 페이지에 오류 메시지가 출력된다.

B. Filtering System

B.1 Upload Video

1) User : 필터링을 원하는 비디오를 업로드한다.
2) 시스템 동작 : 해당 영상을 서버에 저장하고, 필터링 과정을 수행한다.
2-1) 성공
시스템 알림 : “원하시는 필터링 구간을 선택해주세요.”
시스템 동작 : 필터링 될 구간을 유저에게 보여준다 (썸네일, 구간).
2-2) 실패
시스템 동작 : 어떠한 결과도 출력되지 않는다.

B.2 Select scenes

1) User : 필터링을 원하는 구간을 선택하고 제거한다.
2) 시스템 동작 : 해당 구간에 대한 최종 블러링 / 삐처리 기능을 수행한다.
2-1) 성공

시스템 알림 : “필터링이 완료되었습니다.”
시스템 동작 : 필터링 된 결과를 유저에게 보여준다.
2-2) 실패
시스템 동작 : 어떠한 결과도 출력되지 않는다.

B.3 Download

1) User : 필터링을 완료 후, Download 버튼을 클릭한다.
2) 시스템 동작 : 동영상을 User의 Local 저장소에 저장한다.
2-1) 성공
시스템 동작 : User에게 Download 된다.
2-2) 실패
시스템 동작 : 어떠한 결과도 출력되지 않는다.

9. Development plan

9.1 Objectives

Development Plan에서는 개발 계획에 대해 서술한다. 이 때, Gantt chart를 이용하여 개발 계획과 실제 개발 흐름에 대해 서술하도록 한다.

9.2 Gantt Chart (Plan)

	수행 내역	개발 일정											
		4	5	6	7	8	9	10	11	12	13	14	15
1	Idea												
2	Requirement												
3	Design Study												
4	Design												
5	Develop												
6	Integration												
7	Documentation												