팀11 프로젝트 최종 보고서

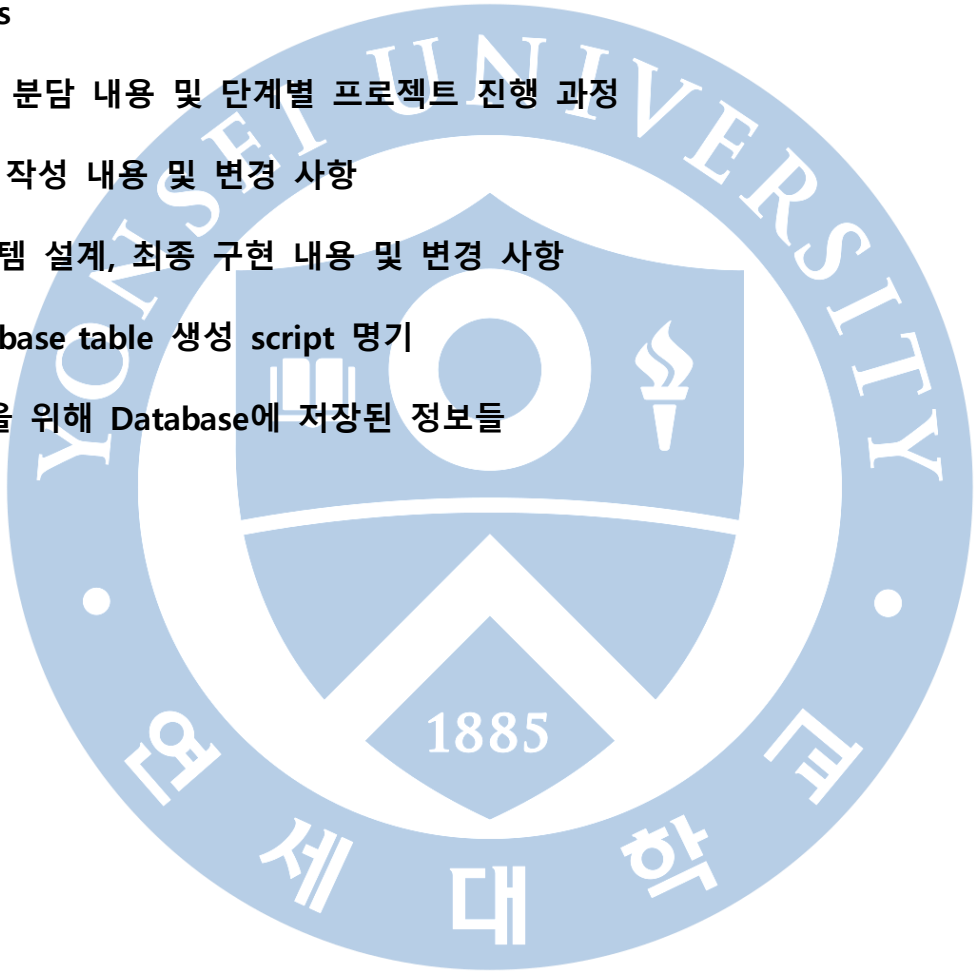
2020학년도 2학기

데이터베이스

최종 보고서

이동환, 이성원, 조윤수, 김연웅, 송건혁, 신동윤

Contents



Ⅰ. 역할 분담 내용 및 단계별 프로젝트 진행 과정

1. 역할 분담 내용

기초 작업

1. ER- Diagram & Mapping ERD to Relational Schema: 이성원, 조윤수
2. 시스템 설계 문서 및 중간 구현내용: 송건혁, 신동윤
3. Database table 생성 script 명기: 이동환, 김연웅

소주제 1

1. 태스크 참여 신청, 파싱 데이터 시퀀스 파일 평가 : 조윤수
2. 원본 데이터 시퀀스 파일 제출, 평가내역 모니터링: 이성원
3. 제출 현황 모니터링 및 평가 점수 확인 : 이동환

소주제 2

1. 회원가입 및 인증 기능, 태스크 생성 기능 : 김연웅
2. 태스크 통계 기능, 최종 보고서 작성 : 송건혁
3. 회원 관리 및 통계 기능, 태스크 관리 기능 : 신동윤

2. 단계별 프로젝트 진행 과정

8주 ~ 10주

- ERD, 시스템 설계 및 초기 Database Table 생성

11주

- ERD, 시스템 설계 및 초기 Database Table 생성
- 초기 설계 피드백 및 수정
- 시스템 설계 문서 및 UI 예시 작성
- 중간 보고서 작성

12주

- 소주제 1 : 제출 기능 및 평가 기능 개발
- 소주제 2 : 회원가입 및 인증 기능, 회원관리 및 통계 기능 개발

13주

- 소주제 1 : 모니터링 기능 개발
- 소주제 2 : 태스크 생성, 관리, 통계 기능 개발

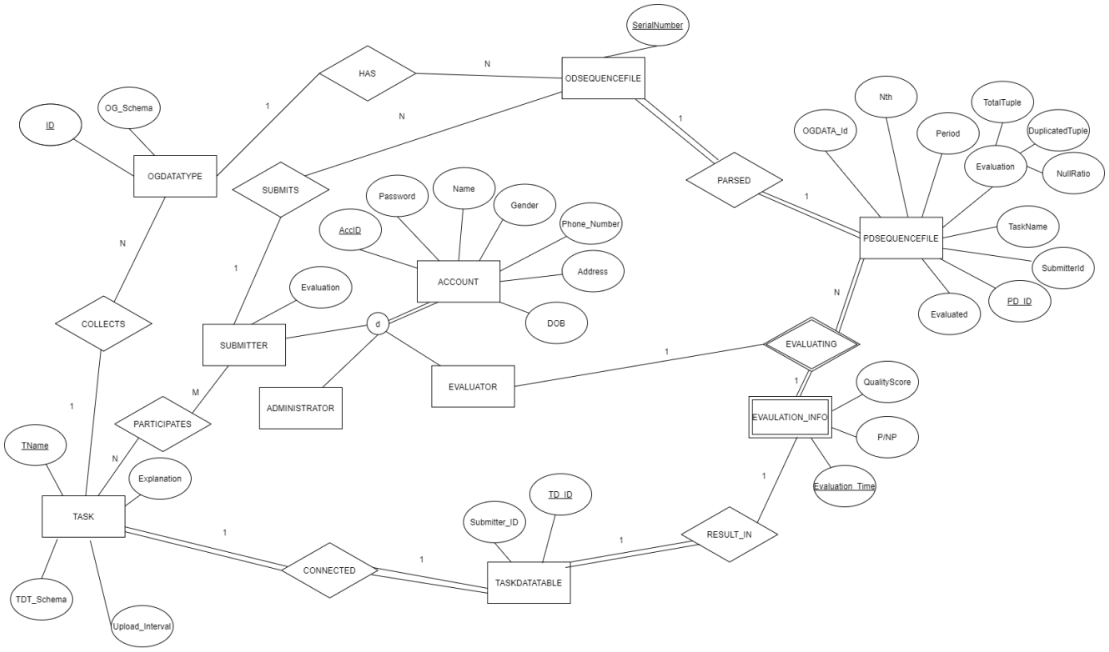
14주

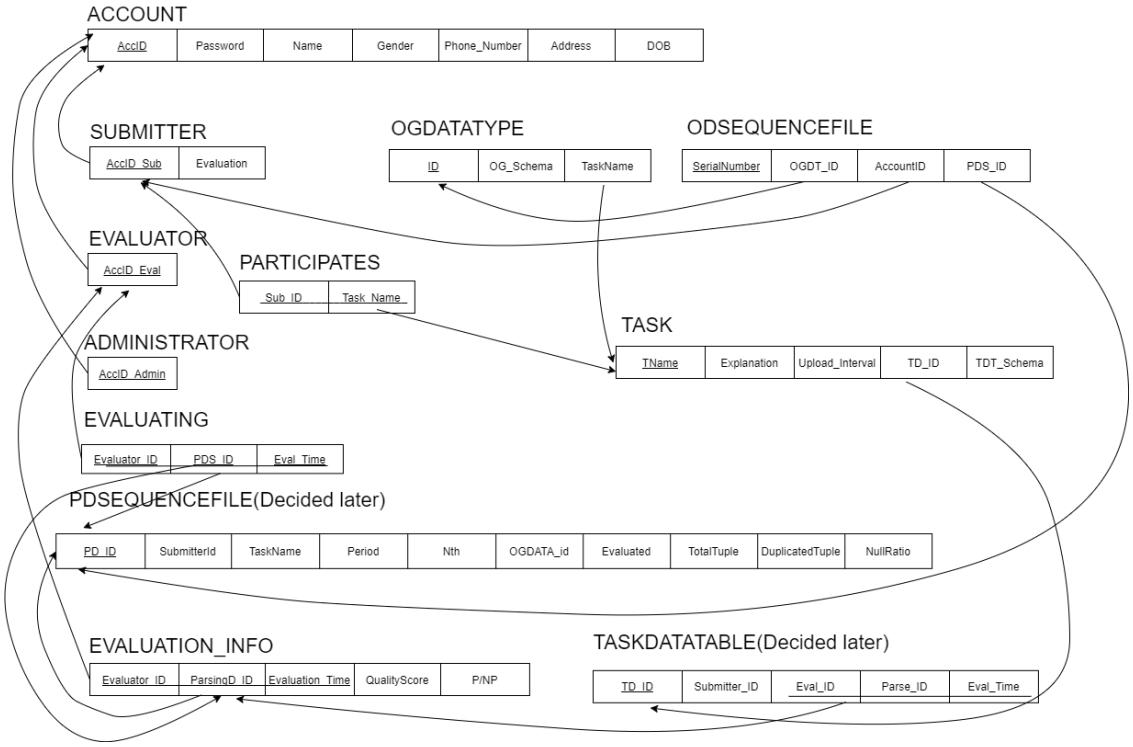
- 전체 프로그램 완성, UI 개선 및 버그 개선
- 최종 보고서 작성

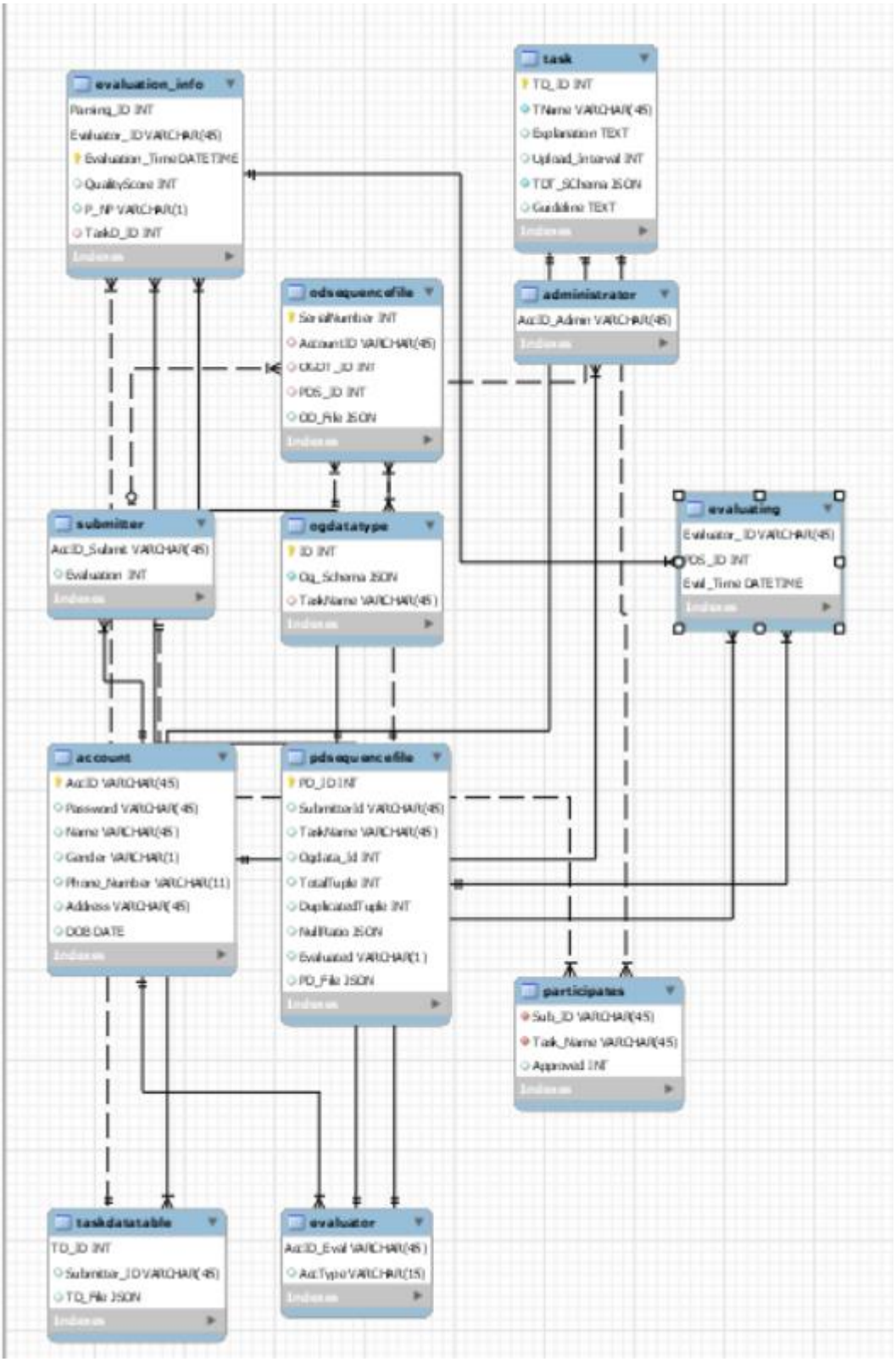
Ⅱ. ERD 작성 내용 및 변경 사항

1. ERD 작성 내용

ERD는 프로젝트 개발 명세서의 '시스템 요구 사항', '용어 정의', '시스템 프로세스 예시' 부분을 참고하여 작성하였다.






evaluation_info
Planing_ID INT
Evaluator_ID VARCHAR(45)
Evaluation_Time DATETIME
QualityScore INT
P_NP VARCHAR(1)
TaskD_ID INT
task
TD_ID INT
TName VARCHAR(45)
Explanation TEXT
Upload_Interval INT
TDT_SChema JSON
Guideline TEXT
odsequencefile
SerialNumber INT
AccountID VARCHAR(45)
OGDT_ID INT
PDS_ID INT
OD_File JSON
administrator
Acc!D_Admin VARCHAR(45)
submitter
Acc!D_Submit VARCHAR(45)
Evaluation INT
ogdatatype
ID INT
Og_Schema JSON
TaskName VARCHAR(45)
evaluating
Evaluator_ID VARCHAR(45)
PDS_ID INT
Eval_Time DATETIME
account
Acc!D VARCHAR(45)
Password VARCHAR(45)
Name VARCHAR(45)
Gender VARCHAR(1)
Phone_Number VARCHAR(11)
Address VARCHAR(45)
DOB DATE
pdsequencefile
PD_ID INT
SubmitterId VARCHAR(45)
TaskName VARCHAR(45)
OgData_Id INT
TotalTuple INT
DuplicatedTuple INT
NullRatio JSON
Evaluated VARCHAR(1)
PD_File JSON
participates
Sub_ID VARCHAR(45)
Task_Name VARCHAR(45)
Approval INT
taskdatatable
TD_ID INT
Submitter_ID VARCHAR(45)
TD_File JSON
evaluator
Acc!D_Eval VARCHAR(45)
Acc!Type VARCHAR(15)

TASK

태스크는 특정 도메인의 데이터 수집을 수행하는 과업의 단위로써, 관리자가 생성하여 수집하고 싶은 태스크와 그 속성들을 정한다. 태스크는 각자마다 고유하고 따라서 Primary key로서 TD_ID가 존재한다. 이에 따라 태스크의 이름인 TName도 유일해야 할 것이다. 그 외에, 태스크에 대한 간략한 설명, 최소 업로드 주기, 태스크 데이터 테이블의 스키마, 평가 기준의 정보가 태스크의 속성으로 추가된다. 태스크는 SUBMITTER subclass와 관계를 맺고 있는데, 여러 명의 제출자가 여러 개의 Task에 대해서 참여할 수 있으므로 N:M 관계로 정의하였고 이를 PARTICIPATE relation을 통해 확인할 수 있다. 또, 하나의 태스크는 한 개 이상의 원본 데이터 타입을 수집하기 때문에, 원본데이터 타입 entity OGDATATYPE과 1:N 관계를 맺고 있다.

OGDATATYPE

OGDATATYPE은 각 태스크에서 수집하고자 하는 원본데이터 타입이다. 같은 태스크 중에서도 여러 개의 다른 원본데이터 타입이 수집될 수 있으므로 TASK와 1:N 관계를 맺고 있다. 원본 데이터 타입은 key 속성인 고유의 ID를 갖고 있고, 해당 타입의 스키마 OG_Schema를 갖고 있다. OG_Schema는 TASK의 속성으로 포함된 TDT_Schema와의 매핑을 이루기 위해 있는 것인데, 실제로 입력되는 스키마와 데이터가 저장될 때 사용되는 스키마가 다르므로 매핑이 필요하다. 두 스키마는 속성으로 포함돼 있으므로 매핑은 Collects 관계를 통해 이룰 수 있다고 가정했다.

ODSEQUENCEFILE

ODSEQUENCEFILE은 명세서의 원본 데이터 시퀀스 파일의 entity이다. 이 파일은 제출자가 실제로 제출한 데이터 파일들을 의미한다. 이 파일들은 일정 주기를 갖고 제출이 되어야 하기 때문에 일련번호가 필요한데, 이 일련번호를 SerialNumber라는 속성으로 포함하였고 key 속성으로 설정했다. 각 원본 데이터 타입에 대해 여러 원본 데이터 시퀀스 파일이 제출될 수 있으므로, HAS라는 관계를 통해 원본데이터 타입 entity와 1:N 관계를 갖는다. 추가로 원본 데이터 시퀀스 파일은 제출자와 1:N 관계를 맺고 있는데 한 명의 제출자가 여러 원본 데이터 시퀀스 파일을 제출할 수 있기 때문이다.

ACCOUNT

ACCOUNT는 해당 서비스를 사용하는 데 필요한 계정이다. 계정은 고유한 ID를 갖고 이 ID가 key 속성이다. 계정에는 아이디, 패스워드, 이름, 성별, 주소, 생년월일, 휴대전화 번호 등의 정보들이 저장되고, 해당 정보들을 ACCOUNT의 속성으로 추가하였다. 또, 각 계정은 관리자, 제출자, 평가자 중 하나의 역할을 갖게 된다. 이 내용을 subclass로 구현하는 것이 쉬울 것으로 생각하여 SUBMITTER, ADMIN, EVALUATOR subclass를 만들었다. 각 계정은 단 하나의 역할을 갖고 있으므로, disjoint subclass로 나뉜다. 제출자는 본인이 제출한 내용에 대한 평가 점수를 부여받기 때문에 평가 점수 속성인 Evaluation을 가진다. 추가로 주목할 점은 EVALUATOR subclass인데, PDSEQUENCEFILE과 함께 EVALUATION_INFO의 identifying relationship 역할을 한다.

PDSEQUENCEFILE

PDSEQUENCEFILE은 파싱 데이터 시퀀스 파일의 entity로, 제출자가 제출한 원본 데이터 시퀀스 파일이 태스크 데이터 테이블의 스키마에 맞게 변환된 것들이다. 원본 데이터 타입의 아이디(OGDATA_ID), 태스크의 이름(TaskName), 제출자의 아이디(SubmitterId), 평가가 이루어졌는지의 상태(Evaluated), 파싱 데이터 시퀀스의 아이디(PD_ID)를 속성으로 갖는다. 정량 평가의 정보로는 총 튜플의 수(TotalTuple), 중복 튜플의 수(DuplicatedTuple), Null의 비율(NullRatio)이 존재한다. 그리고 속성들 중 파싱 데이터 시퀀스의 아이디(PD_ID)가 primary key가 된다. 이 외에도 PDSEQUENCEFILE는 태스크 데이터 테이블의 스키마가 가지게 될 속성을 똑같이 갖는다. 다만 태스크 데이터 테이블의 속성 즉 태스크 데이터 테이블의 구체적인 정보는 관리자가 태스크를 생성하면서 정해지므로 ERD에는 JSON 타입으로 존재한다. 추가적으로 PDSEQUENCEFILE은 ACCOUNT의 subclass인 EVALUATOR와 함께 EVALUATION_INFO의 identifying relationship 역할을 한다.

TASKDATATABLE

TASKDATATABLE은 평가자에 의해 저장할 만한 데이터로 분류되어 실제 파싱 데이터 시퀀스 파일이 저장되는 것이다. 이는 제출자의 아이디(Submitter_ID)와 태스크 데이터 테이블의 아이디(TD_ID)를 속성으로 갖는다. SerialNum은 primary key로, 태스크 데이터 테이블에 primary key를 부여하기 위해 임의로 추가한 속성이다. 이 때 태스크 내에서 태스크 데이터 테이블에 접근할 수 있어야 하기에 태스크 데이터 테이블에 TD_ID 속성을 추가하여 태스크와 연결 지을 수 있도록 하였다. 다만 태스크 데이터 테이블의 구체적인 정보는 관리자가 태스크를 생성하면서 정해지므로 ERD에는 JSON 타입으로 존재한다.

EVALUATION_INFO

EVALUATION_INFO는 EVALUATOR가 PDSEQUENCEFILE을 평가한 내용을 담고 있다. 이 개체는 평가 시간(Evaluation_Time), 저장 pass 여부(P/NP), 정성 평가 점수(QualitySore)를 속성으로 갖는다. 이 중 평가 시간이 EVALUATION_INFO의 partial key이다. 한 명의 평가자는 서로 다른 파싱 데이터 시퀀스 파일을 동시에 평가하고 제출할 수 없으므로 평가 시간이라는 속성을 추가하여 partial key로 삼았다. 또, identifying relationship인 EVALUATING에 참여하는 두 entity는 각각 1과 N으로 참여한다. 한 명의 평가자는 여러 파싱 데이터 시퀀스 파일을 평가할 수 있지만, 파싱 데이터 시퀀스 파일은 한 평가자에게 배당되므로 EVALUATOR는 1, PDSEQUENCEFILE은 N으로 참여하는 것이다. 이에 더불어 평가에서 통과한 파싱 데이터 시퀀스 파일 즉 P/NP 속성의 값이 P(pass)인 파싱 데이터 시퀀스만이 RESULT_IN에 참여한다. 따라서 EVALUATION_INFO는 RESULT_IN에 부분 참여한다. 그런데 이 RESULT_IN에 참여한다는 것은 태스크 데이터 테이블에 반드시 저장된다는 것을 의미하므로, TASKDATATABLE은 RESULT_IN에 전체 참여한다. 또한 완전히 동일한 파일이 두 번 저장될 수 없으므로 1:1 관계를 갖는다.

2. 변경 사항

- Participates에 approved column 추가

이를 통하여 참여 신청을 한 제출자가 관리자에 의해 승인되었는지 여부를 저장할 수 있음

- Task에 Guideline column 추가

이를 통하여 관리자가 평가 기준을 저장하고 평가자가 이를 조회할 수 있음

- Task의 Primary Key를 TD_ID로 부여

관리의 용이성을 위해 String인 Name 대신 ID를 사용하였고 구현 상 중복 체크를 통해 원래 Primary key였던 이름도 유일한 값을 가짐

- Pdsequencefile에 PD_File column 추가

관리자가 태스크를 생성하면서 정해지는 컬럼 정보 등을 JSON 파일의 형식으로 저장

- Taskdatatable에 SerialNum column을 Primary key로 추가

Taskdatatable과 task를 연결할 수 있어야 하기에 새로운 Primary Key를 생성하고 기존 Primary Key인 TD_ID을 통해 task와 연결

- Taskdatatable에 TD_File column 추가

관리자가 태스크를 생성하면서 정해지는 컬럼 정보 등을 JSON 파일의 형식으로 저장

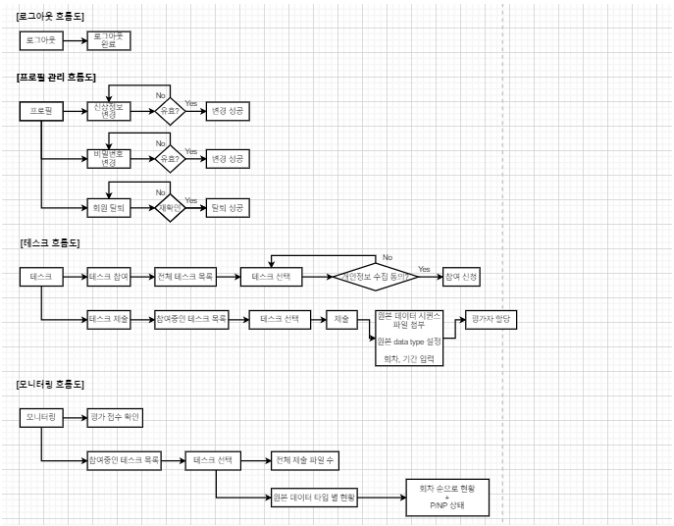
Ⅲ. 시스템 설계, 최종 구현 내용 및 변경 사항

1. 시스템 설계, 최종 구현 내용

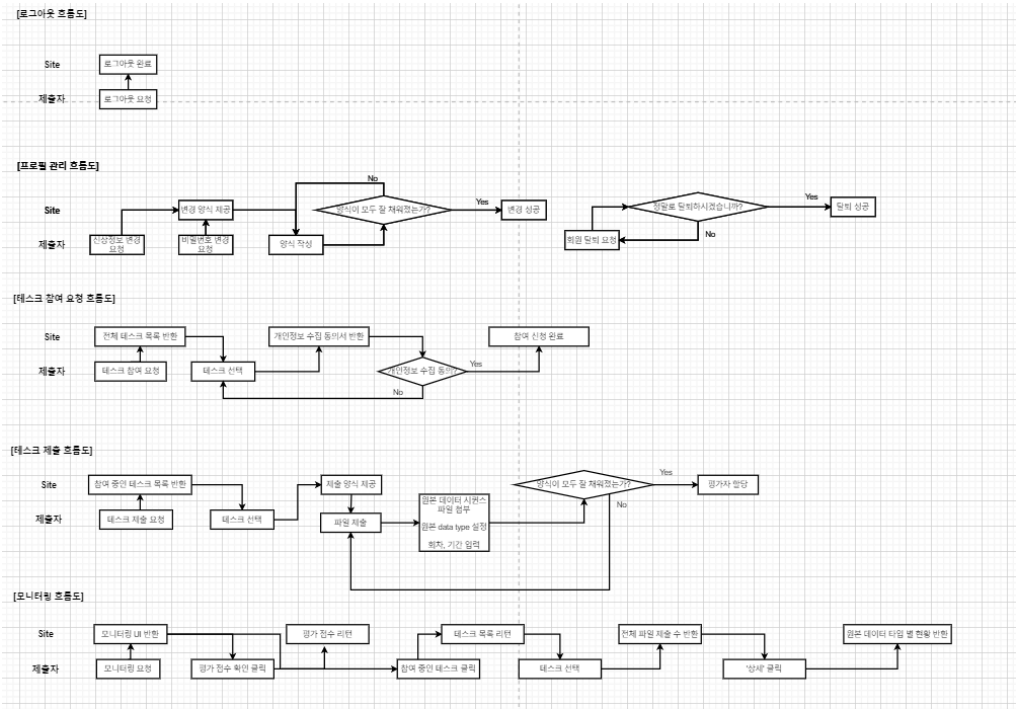
1.1 소주제 1

제출자 Flowchart

[Action 중심 flow chart]



[Actor 중심 flow chart]



1.1.1 제출자 기능

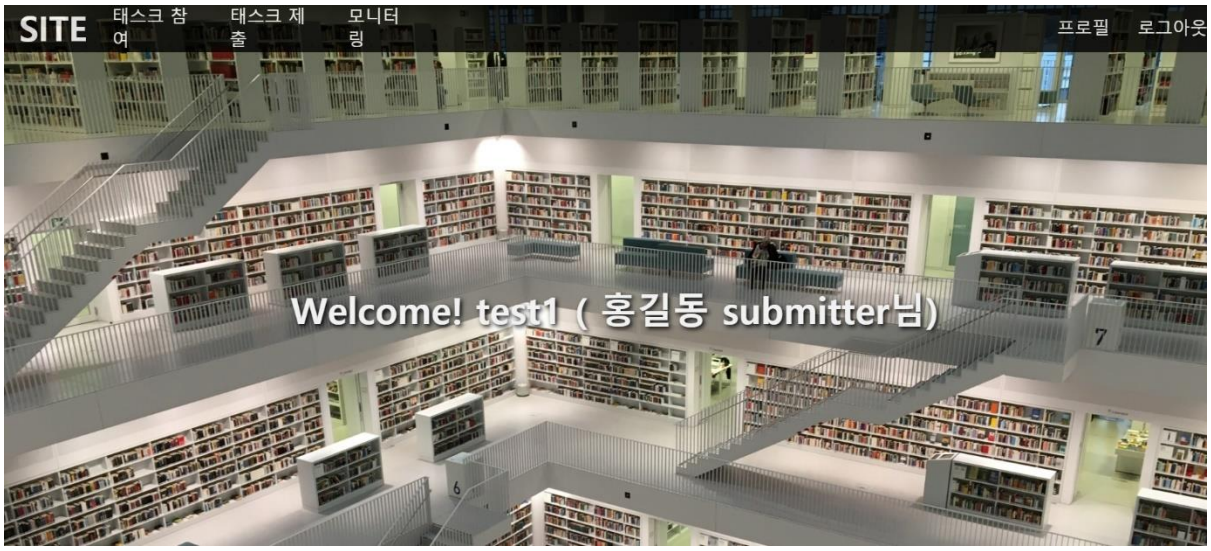


그림 1 제출자 계정 메인 화면

그림 1은 제출자 계정 메인 화면으로, 제출자인 사용자가 로그인을 하면, 메인페이지가 렌더링되면서 상단바의 '태스크 참여', '태스크 제출', '모니터링' 링크를 클릭할 수 있고, 여러 제출자 기능들을 사용할 수 있다.

1.1.1.1 태스크 참여 신청

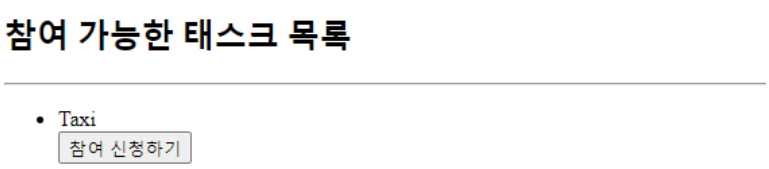


그림 2 태스크 잠여 신청 메인

그림 2는 태스크 참여 신청 메인 화면이다. 제출자는 관리자가 생성한 태스크들을 보고, 참여 신청을 할 수 있다. 관리자가 생성한 태스크가 없을 경우, 목록에 아무런 태스크가 뜨지 않는다.

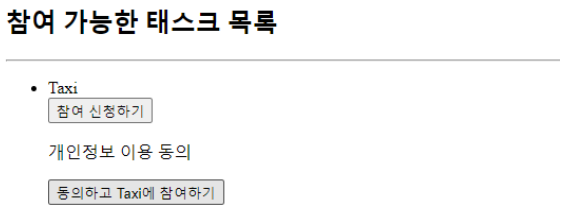


그림 3 참여 신청하기를 누르면 나타나는 화면

그림 3은 '참여 신청하기' 버튼을 클릭하면, 아래에 개인정보 이용 동의 버튼이 아래에 생겨나는 화면이다.

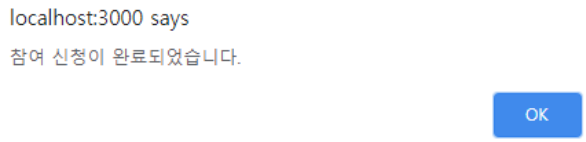


그림 4 참여 신청 완료

그림 4는 참여 신청 완료 메시지이다. 제출자가 개인정보 이용 동의 버튼을 클릭하면 참여 신청이 완료되었다는 alert가 뜨고, 제출자의 신청이 저장되며 관리자가 해당 제출자의 신청을 승인하면 제출에 참여 가능하다.

1.1.1.2 원본 데이터 시퀀스 파일 제출

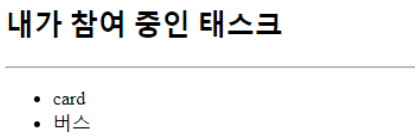


그림 5 태스크 제출 메인화면

그림 5는 태스크 제출 메인화면으로 태스크 신청에 대한 승인이 완료되면, 제출자는 메인페이지의 '태스크 제출' 버튼을 눌러 본인이 참여 중인 태스크에 대한 목록을 볼 수 있다.

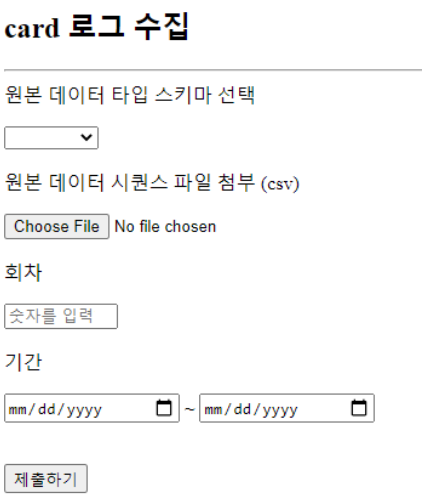


그림 6 개별 태스크 별 제출 화면

그림 6은 개별 태스크 별로 접근하였을 때 보여지는 제출 화면이다. 그림 5에서 제출자는 제출하고 싶은 태스크를 클릭하고 원본데이터시퀀스 파일을 제출하는 폼에서 제출을 할 수 있다. 원본데이터스키마는 drop-down 방식의 목록에서 선택할 수 있다. 원본데이터시퀀스 파일의 업로드는 CSV 파일에 한해 허용된다. 또, 아래에 제출의 회차와 기간정보를 선택할 수 있다. 모든 정보 입력칸에 대해 입력을 하지 않으면 입력을 하라는 prompt가 뜨고 제출을 할 수 없다.

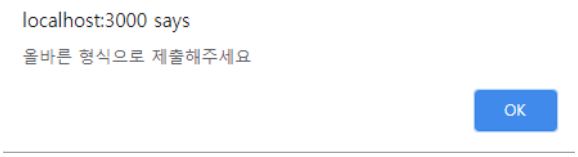


그림 7 제출 시 에러 메시지 1

그림 7은 제출 시 발생하는 에러 메시지 중 하나이다. 바르지 않은 형식의 데이터가 입력될 경우 올바른 형식으로 제출하라는 alert가 뜨며 제출할 수 없다.

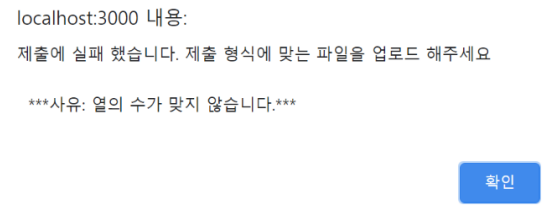


그림 8 제출 시 에러 메시지 2

그림 8은 제출 시 발생하는 또 다른 에러 메시지이다. CSV 파일은 제출과 동시에 데이터에 대한 constraint에 대한 검사가 진행되는데, 관리자가 설정한 constraint에 부합하지 않으면, 제출 형식의 맞는 파일을 제출하라는 alert가 뜨고, 다시 메인페이지로 redirect된다. Constraint에 맞지 않는 파일은 원본데이터시퀀스 파일로 저장되지 않는다.

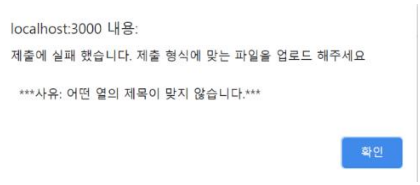


그림 9 제출 시 에러 메시지 3

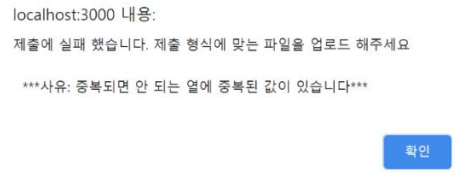


그림 10 제출 시 에러 메시지 4

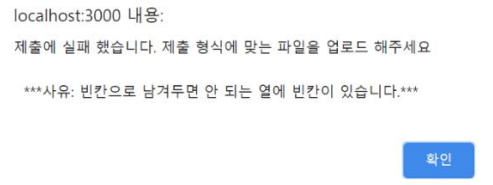


그림 11 제출 시 에러 메시지 5

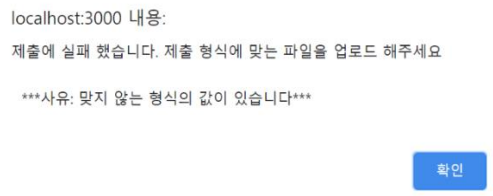


그림 12 제출 시 에러 메시지 6

그림 9~12는 또 다른 에러 메시지들이다. 제출에 실패한 사유가 alert에 써져 있기 때문에 제출자가 문제를 알기 쉽다. 위의 alert들은 발생 가능한 여러 제출 실패 사유들이다.

그림 13 성공적으로 제출할 경우

그림 13은 모든 constraint에 문제가 없을 경우 성공적으로 제출했다는 메시지가 뜨고 CSV 파일의 내용이 json 형태의 원본데이터시퀀스파일로 저장된다. 원본데이터시퀀스파일이 저장되면서, 원본데이터시퀀스파일의 Null Ratio, Total Tuples, Duplicated Tuples 등의 내용이 바로 파싱되면서 파싱데이터시퀀스파일에 저장된다. 동시에, 이 파싱데이터시퀀스파일은 랜덤의 평가자에게 배정된다.

1.1.1.3 제출 현황 모니터링 및 평가 점수 확인

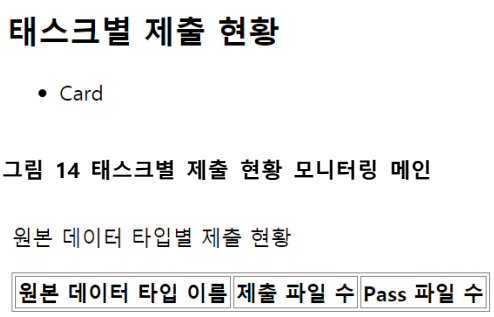


그림 14 태스크별 제출 현황 모니터링 메인

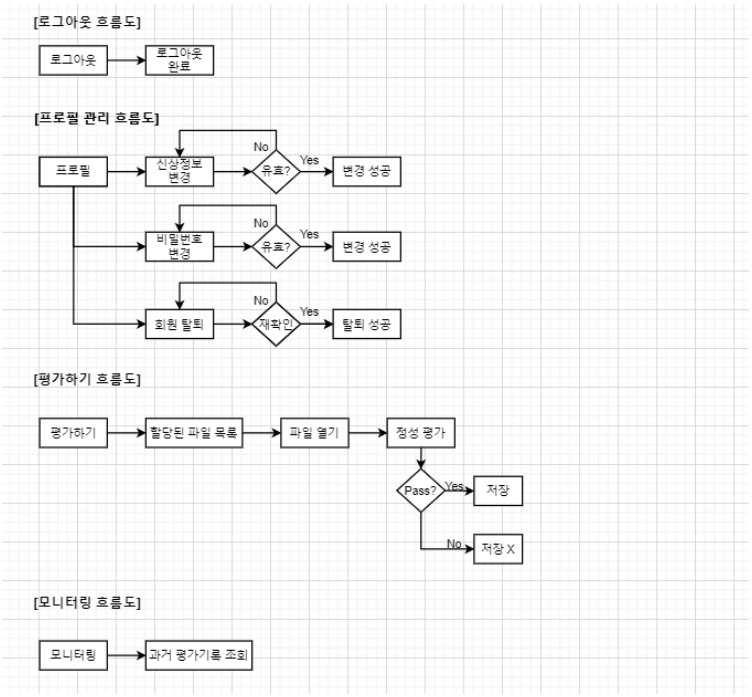
그림 15 개별 태스크별 제출 현황 모니터링 화면

제출자는 본인이 제출한 원본 데이터 타입별로 본인의 제출 현황을 확인할 수 있는데, 이 부분의 구현은 완성하지 못했다.

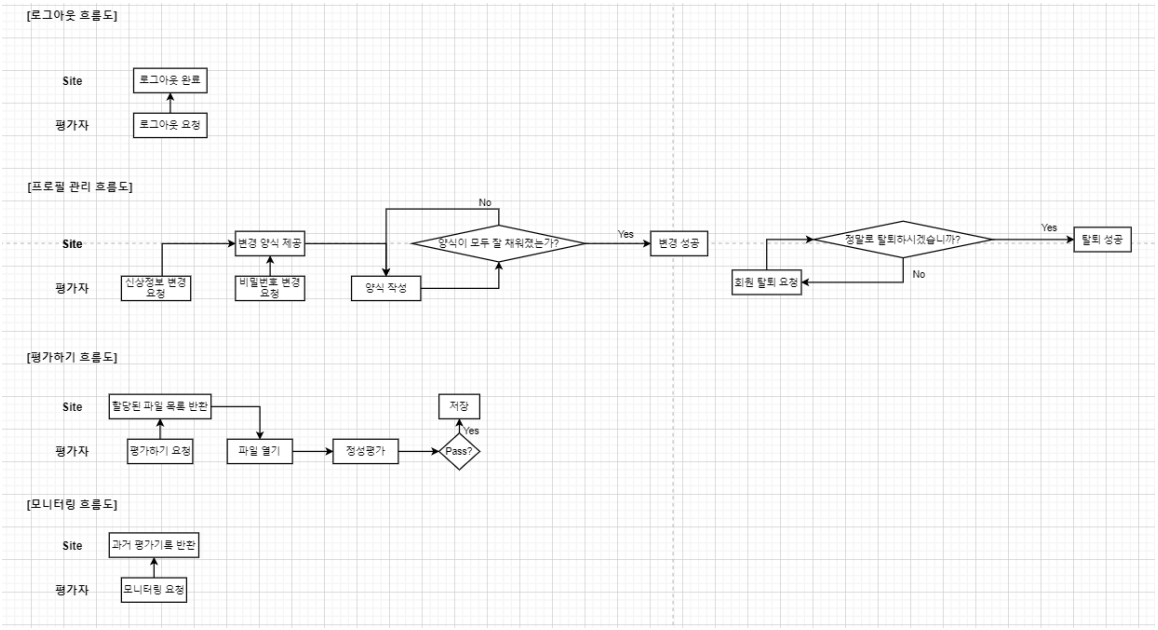
1.1.2 평가자 기능

평가자 flow chart

[Action 중심 flow chart]



[Actor 중심 flow chart]



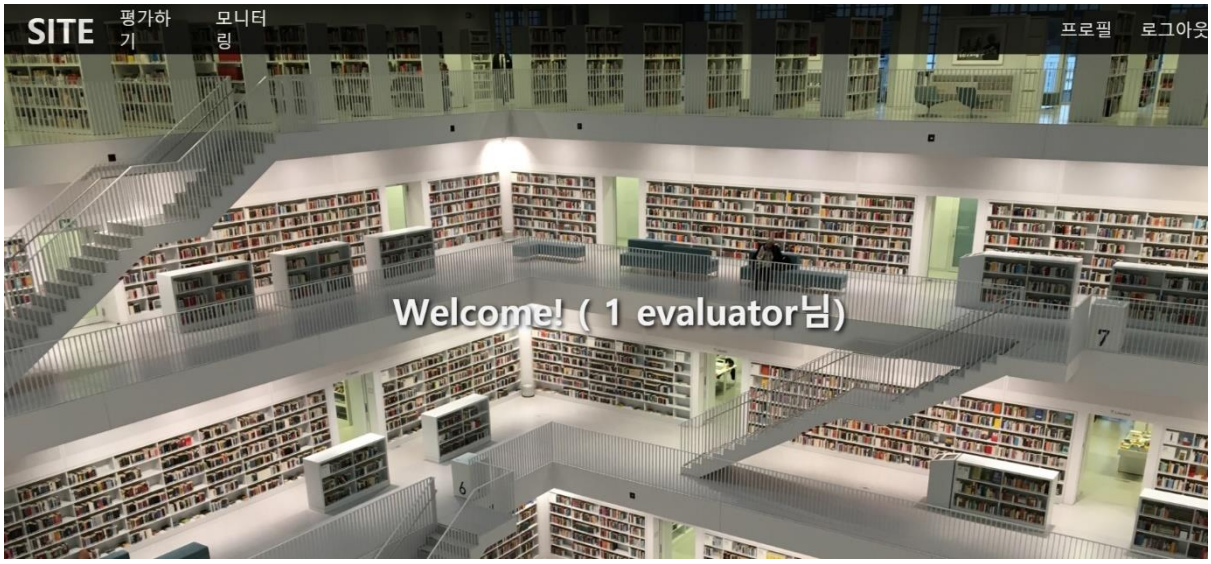


그림 16 평가자 메인 화면

그림 16은 평가자 메인 화면이다. 평가자는 로그인하면 메인페이지에서 '평가하기', '모니터링' 버튼을 통해 기능을 사용할 수 있다.

1.1.2.1 파싱 데이터 시퀀스 파일 평가

평가하기

아래 파싱 데이터 시퀀스 파일을 평가해주세요.

파싱 데이터 시퀀스 파일 정보

제출자 아이디	태스크 이름	총 튜플 수	중복 튜플 수	name Colunm의 Null 비율	time Colunm의 Null 비율	entry Colunm의 Null 비율	amount Colunm의 Null 비율	address Colunm의 Null 비율
test1	Card	9	2	0	0	0	0	0

그림 17 평가하기 메인

파싱 데이터 시퀀스 파일 내용

name	time	entry	amount	address
홍길동	2020-12-06	1	2000	서울시 OO로 1462
홍길동	2020-12-07	1	20000	서울시 OO로 1462
홍길동	2020-12-08	1	3500	서울시 OO로 1464
홍길동	2020-12-09	1	52000	서울시 OO로 1462
홍길동	2020-12-10	1	1200	서울시 OO로 1462
홍길동	2020-12-11	1	30000	서울시 OO로 1462
홍길동	2020-12-11	1	7000	서울시 OO로 1462
홍길동	2020-12-11	1	7000	서울시 OO로 1462
홍길동	2020-12-11	1	7000	서울시 OO로 1462

그림 18 평가하기 파일 내용 화면

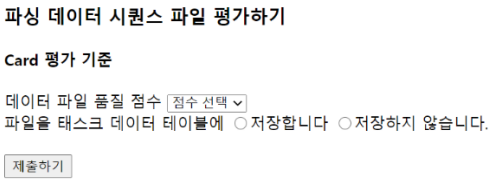


그림 19 평가하기 점수 보여

그림 17~19는 평가자가 평가하는 것을 나타낸 화면이다. 평가자가 메인페이지의 평가하기 버튼을 누르면, 평가자에게 배당된 파싱데이터파일의 정량 지표와 내용이 표시된다. 평가자는 이 정보들을 보고 데이터의 정성적인 품질 점수를 선택하고 해당 파일이 기준에 부합해 Pass를 부여할지 (태스크 데이터 테이블에 저장할지) 그렇지 않다면 Non-Pass를 부여할지 정한다.

1.1.2.2 평가 내역 모니터링

PD_ID	Task_Name	OGDATA_id	SubmitterId	Nth	Total tuple	Duplicated tuple	Null ratio	정성 평가	P/NP
15	Card	1	test1	1	9	2	{"name": 0.0, "time": 0.0, "entry": 0.0, "amount": 0.0, "address": 0.0}	5	P

그림 20 평가 내용 모니터링

평가자가 메인페이지에서 모니터링 버튼을 누르면 과거에 평가한 모든 내역을 보여주는 페이지를 렌더링한다. 이 정보는 열람만 가능하고 수정, 삭제가 불가능하다.

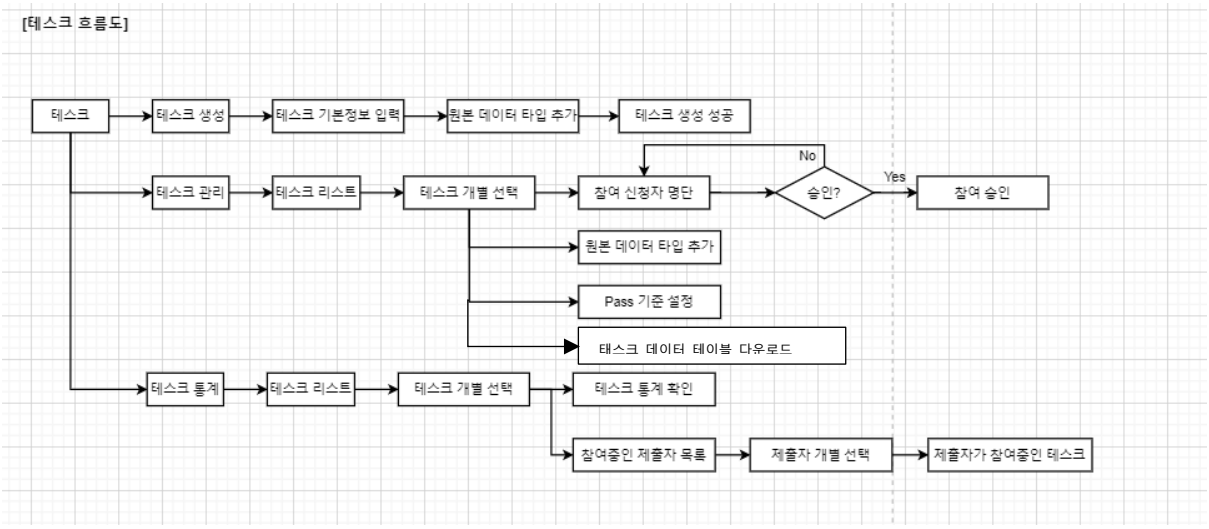
1.1.2.3 제출자의 평가 점수 관리 기능

제출자의 평가 점수 관리 기능은 앞서 설명한대로 정량적인 정보는 시스템이 원본데이터시퀀스파원본 받아들일 때 자동으로 파싱되고, 정성적인 평가 내용은 평가자에게 부여해 평가자가 관리자의 평가 기준을 참고해서 결정한다. 제출자의 최종적인 점수는 모든 정성 평가와 정량 평가 점수의 평균을 계산해 결정된다.

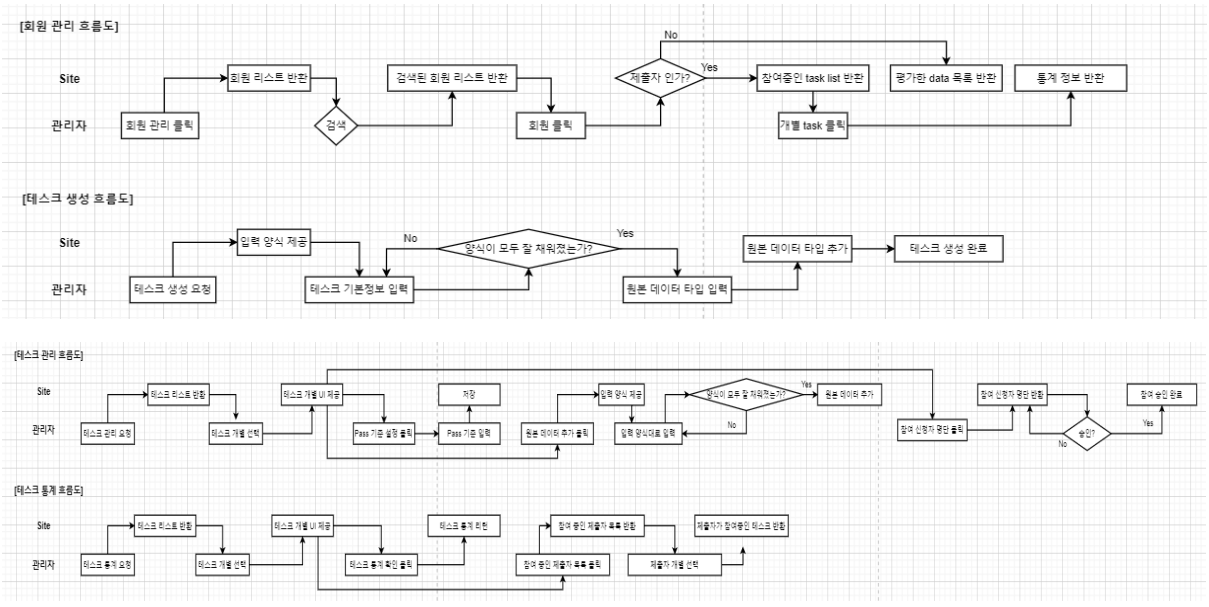
1.2 소주제 2

1.2.1 관리자 기능

[Action 중심 flow chart]



[Actor 중심 flow chart]



[실제 구현 UI]

그림 21 관리자 계정 접속 시 상단바

그림 21은 관리자 계정으로 접속 시 상단바의 구성이다. 관리자 계정은 [태스크 생성], [태스크 관리], [태스크 통계], [회원관리]가 좌측에 추가되며 우측의 [프로필], [로그아웃]은 회원 권한에 무관하게 로그인 이후 접근 가능한 카테고리이다.

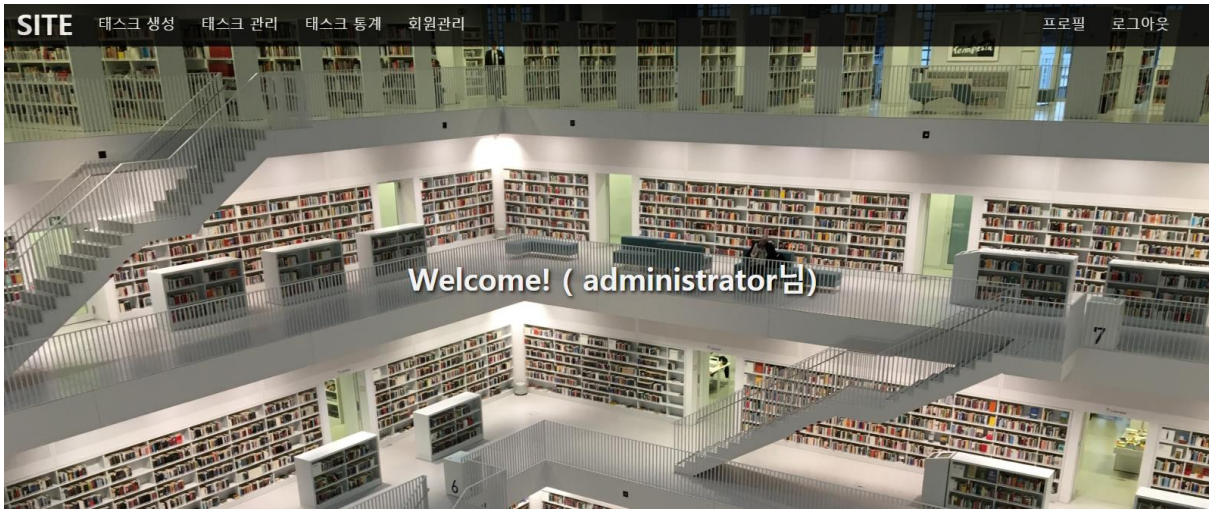


그림 22 관리자 계정 메인화면

그림 22은 관리자 계정으로 접속 시 메인화면의 구성이다. 관리자 계정으로 접속할 경우 화면 정중앙에 'Welcome! (administrator 님)'이라는 문구가 뜨게 되며, 상단바를 이용하여 각각의 기능으로 접근이 가능하다.

1.2.1.1. 태스크 생성

영어로만 입력해 주세요

[실제 구현 UI]

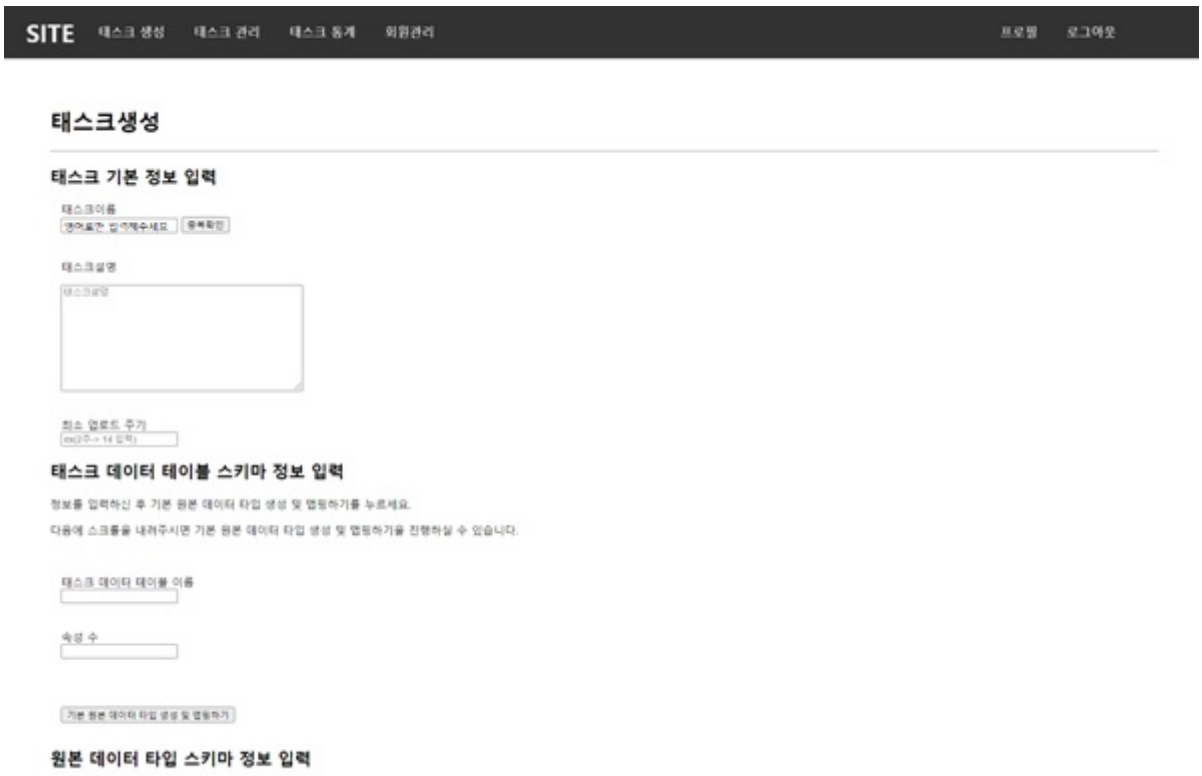


그림 23 태스크 생성 기능 메인

그림 23은 상단바의 태스크 생성 메뉴 클릭 시 접근 가능한 태스크 생성 기능 메인 화면이다. 태스크 기본 정보 입력 부분에서 태스크 이름을 입력할 수 있으며, 태스크 설명을 적고 최소 업로드 주기를 날짜 base로 입력 받는다. 다만, 여기에서 우리는 태스크의 이름을 영어로만 받는다. 이는 한글을 사용함으로써 혹시 일어날 수도 있는 에러를 막기 위함이다.

태스크 데이터 테이블 스키마 정보 입력에서, 태스크 데이터 테이블의 이름을 입력할 수 있으며 속성 수를 입력 시 속성마다 속성 이름, 데이터형, Constraint를 정할 수 있는 input 창과 Primary Key로 지정할 속성의 이름을 받는 input 창이 생겨난다.

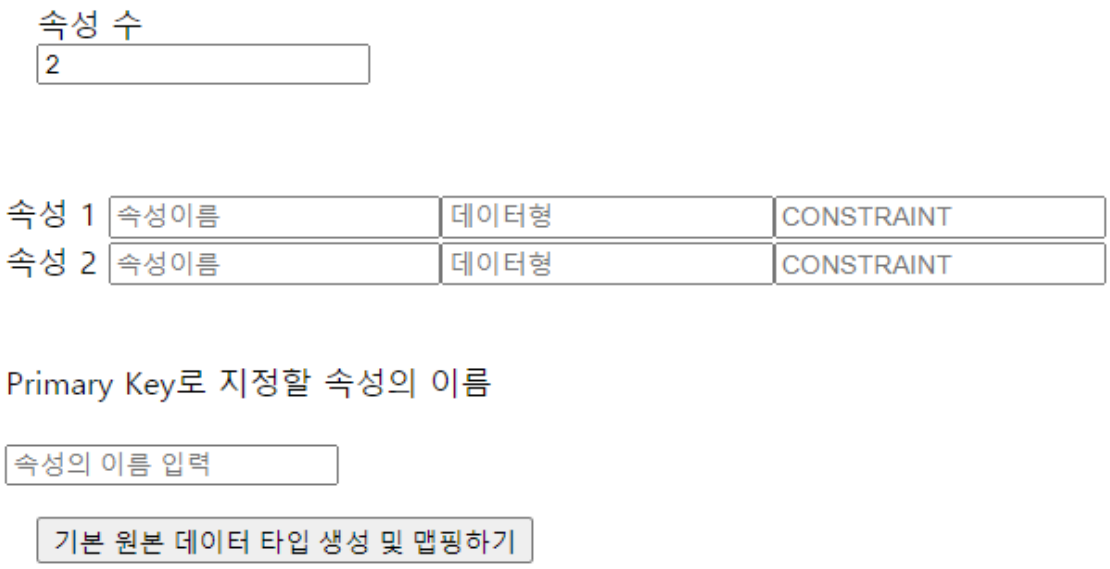


그림 24 속성 수 입력 시 나타나는 input 창

그림 24은 속성 수 입력 시 나타나는 input 창으로, 속성마다 속성이름, 데이터형, CONSTRAINT를 입력할 수 있다. 또한 Primary Key로 지정할 속성의 이름을 정해주어야 한다. 이 때, 지원되는 데이터형은 int와 string이고 지원되는 Constraint는 unique와 not null이다.

모든 input을 입력한 후 '기본 원본 데이터 타입 생성 및 맵핑하기'를 누르면 밑에 추가적인 input 창들이 나타난다. 이 때, 태스크 이름 중복확인을 하지 않은 경우, 태스크 이름을 입력하지 않은 경우, 최소 업로드 주기를 입력하지 않거나 숫자가 아닌 다른 input을 넣은 경우, 태스크 데이터 테이블의 이름을 입력하지 않은 경우, 각각 속성의 이름을 넣지 않은 경우, 각각 속성의 데이터형을 넣지 않은 경우, 중복된 속성 이름을 넣은 경우, Primary Key로 지정할 속성의 이름을 넣지 않거나 올바르게 입력하지 않은 경우, 진행이 되지 않으며, 추가적으로 입력이 필요한 부분을 알려주는 메시지가 뜬다.

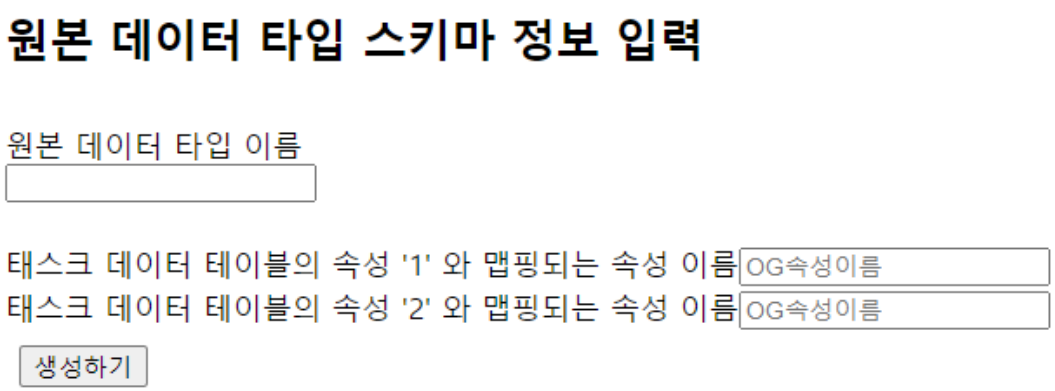


그림 25 '기본 원본 데이터 타입 생성 및 맵핑하기' 클릭시 나타나는 나타나는 input 창

그림 25은 '기본 원본 데이터 타입 생성 및 맵핑하기' 클릭시 나타나는 나타나는 input 창으로, 원본 데이터 타입 이름과 맵핑되는 속성 이름을 적을 수 있다. 모두 작성완료 후에 생성하기 버튼을 누르면 테스크 생성이 완료된다. 이 때, 원본 데이터 타입 이름을 입력하지 않은 경우, 맵핑되는 각각의 속성 이름을 입력하지 않은 경우, 각각의 속성 이름을 중복해서 입력한 경우 생성이 되지 않으며, 추가적으로 입력이 필요한 부분을 알려주는 메시지가 뜬다.

1.2.1.2. 태스크 관리

[실제 구현 UI]

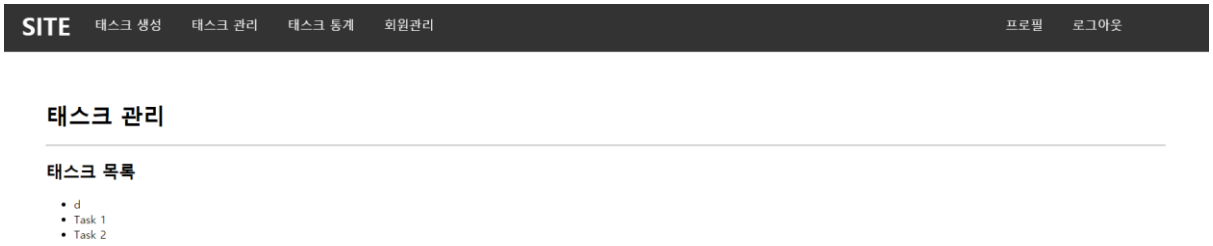


그림 26 '태스크 관리' 메인

그림 26은 '태스크 관리' 메인 화면으로, 생성된 태스크 목록들이 보여 진다. 이 때 각각의 태스크 이름을 클릭하면 태스크 별 관리 화면으로 이동할 수 있다.

그림 27 '태스크 관리' 태스크별 관리 페이지

그림 27은 '태스크 관리'에서 태스크별 관리 페이지로 접근하게 되면 보여지는 화면이다. 태스크 관리 바로 밑에 있는 '태스크 목록으로 돌아가기'를 통하여 '태스크 관리' 메인 화면으로 접근할 수 있고, 그 옆의 '메인페이지로 돌아가기'를 통하여 '관리자' 메인 화면으로 돌아갈 수 있다. 그 밑으로 접근한 태스크 이름이 뜨며 태스크 설명도 불러온다. 밑으로 '참여 신청자 명단', '원본 데이터 타입 추가', 'PASS 기준 설정', '태스크 데이터 테이블 다운로드' 버튼이 있다. 각각의 버튼을 누르면 해당 기능이 하단에 나타난다.

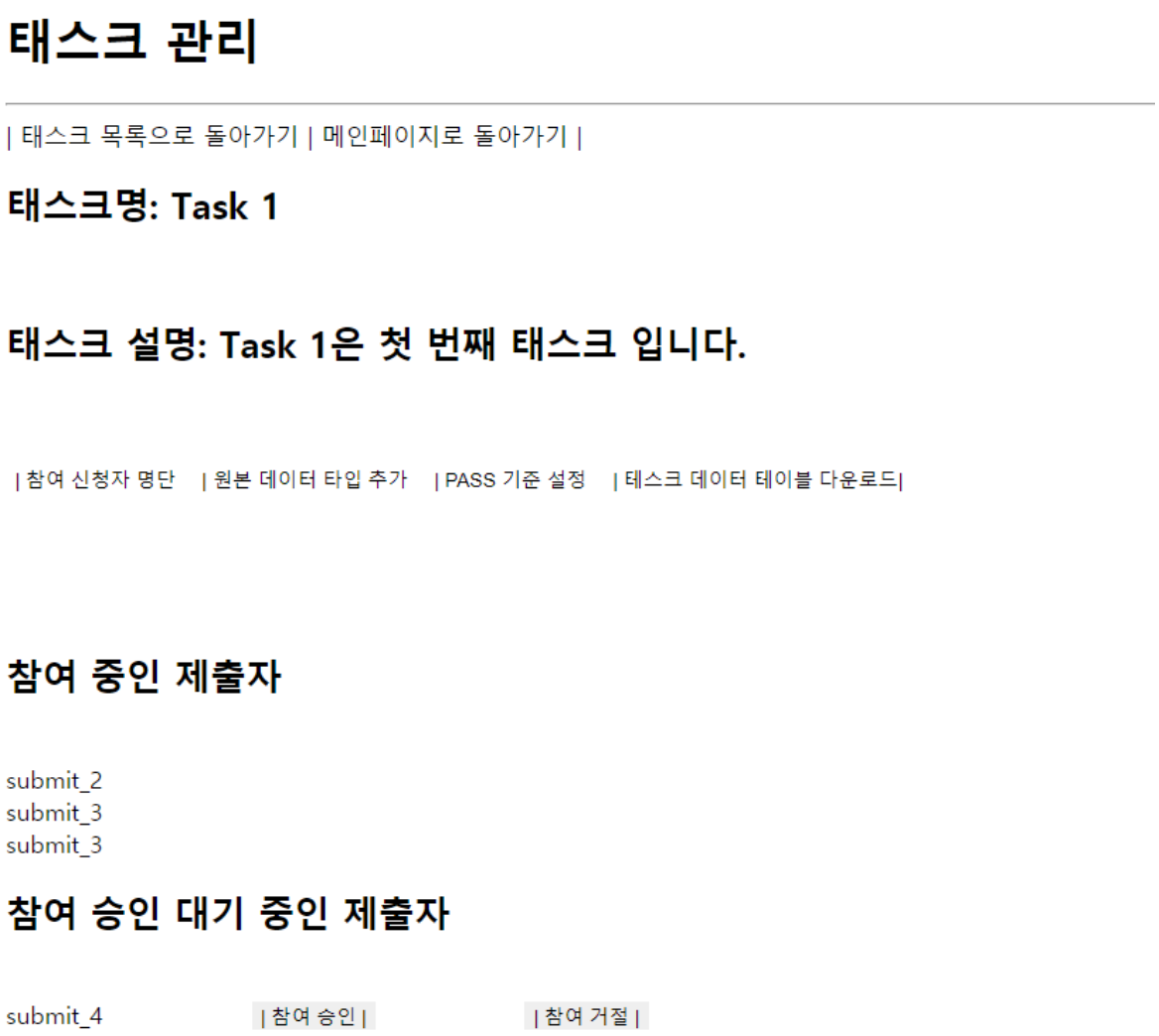


그림 28 '참여 신청자 명단' 기능

그림 28은 '태스크 관리'에서 태스크별 관리 페이지로 접근하여 '참여 신청자 명단'을 클릭하면 보여지는 화면이다. 참여 중인 제출자와 참여 승인 대기 중인 제출자가 보여지며, 승인 대기 중인 제출자에 대하여 '참여 승인'을 누르게 되면 승인을 할 수 있고, '참여 거절'을 누르게 되면 거절을 할 수 있다.

그림 29 '원본 데이터 타입 추가' 기능

그림 29은 '태스크 관리'에서 태스크별 관리 페이지로 접근하여 '원본 데이터 타입 추가'을 클릭하면 보여지는 화면이다. 맵핑되는 태스크 데이터 타입 테이블 이름이 공개되며, 이에 대해 맵핑을 진행할 원본 데이터 타입의 이름과 각각 속성에 맵핑되는 속성의 이름을 작성할 수 있다. 이 때, 입력되지 않은 값이 있거나 중복되어 속성 이름을 입력하게 되면 생성이 되지 않는다.

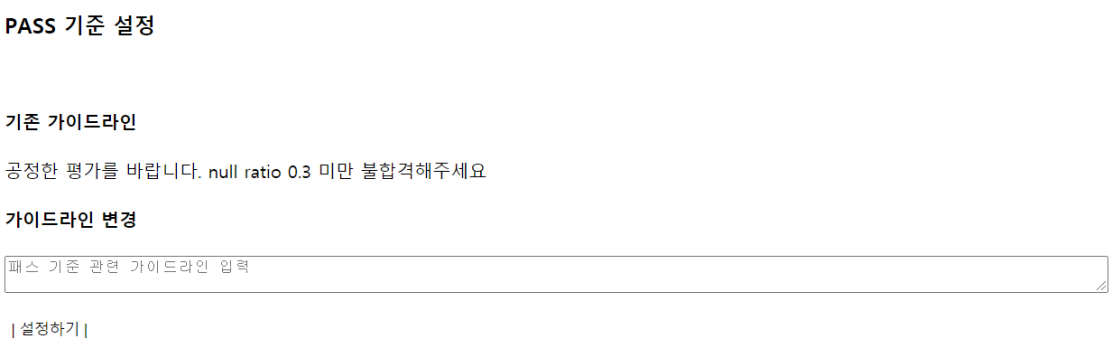


그림 30 'PASS 기준 설정' 기능

그림 30은 '태스크 관리'에서 태스크별 관리 페이지로 접근하여 'PASS 기준 설정'을 클릭하면 보여지는 화면이다. 기존 가이드라인과 가이드라인을 변경할 수 있는 input 창이 제공되며, 설정하기를 통해 패스 기준을 변경할 수 있다. 가이드라인 변경에 아무것도 입력하지 않으면 진행되지 않는다.

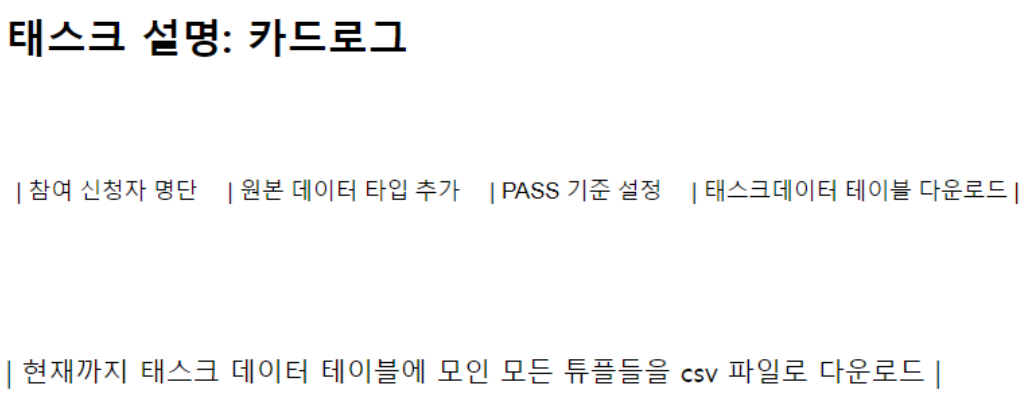


그림 31 '태스크 데이터 테이블 다운로드' 기능

그림 31은 '태스크 관리'에서 태스크별 관리 페이지로 접근하여 '테스크 데이터 테이블 다운로드'을 클릭하면 보여지는 화면이다. '현재까지 태스크 데이터 테이블에 모인 모든 튜플들을 csv 파일로 다운로드'를 누르면 csv 파일이 다운 받아진다. 만일, 튜플이 없을 경우 태스크 목록 화면으로 이동된다.

1.2.1.3. 태스크 통계

[실제 구현 UI]

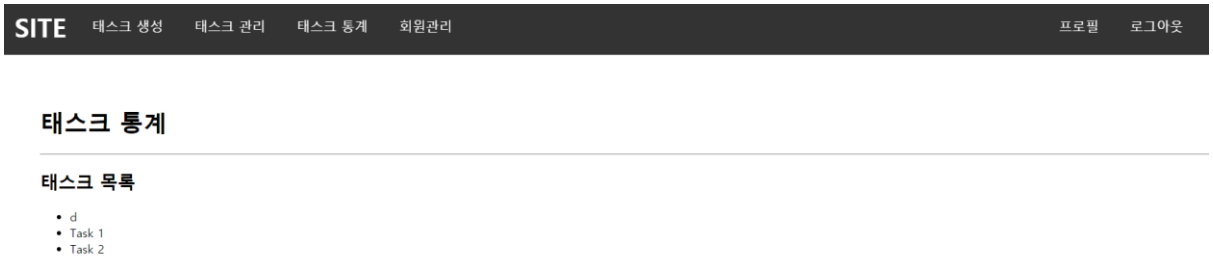


그림 32 '태스크 통계' 메인

그림 32은 '태스크 통계' 메인 화면으로, 생성된 태스크 목록들이 보여 진다. 이 때 각각의 태스크 이름을 클릭하면 태스크 별 통계 화면으로 이동할 수 있다.

그림 33 '태스크 통계' 테스크 별 통계 화면

그림 33은 '태스크 통계' 에서 접근 가능한 태스크 별 통계 화면으로, 전체 제출된 파일 수와 Task table의 tuple로 추가된 수, 원본 데이터 타입 별 제출 파일 수와 원본 데이터 타입 별 Pass 된 파일 수가 보여진다. 그 밑으로 현재 해당 태스크에 참여중인 제출자 리스트가 보여진다. 각각의 제출자를 클릭하게 되면 해당 제출자가 참여 중인 태스크를 불러온다. 맨 위의 ← 버튼을 클릭하면 '태스크 통계' 메인 화면으로 이동한다.

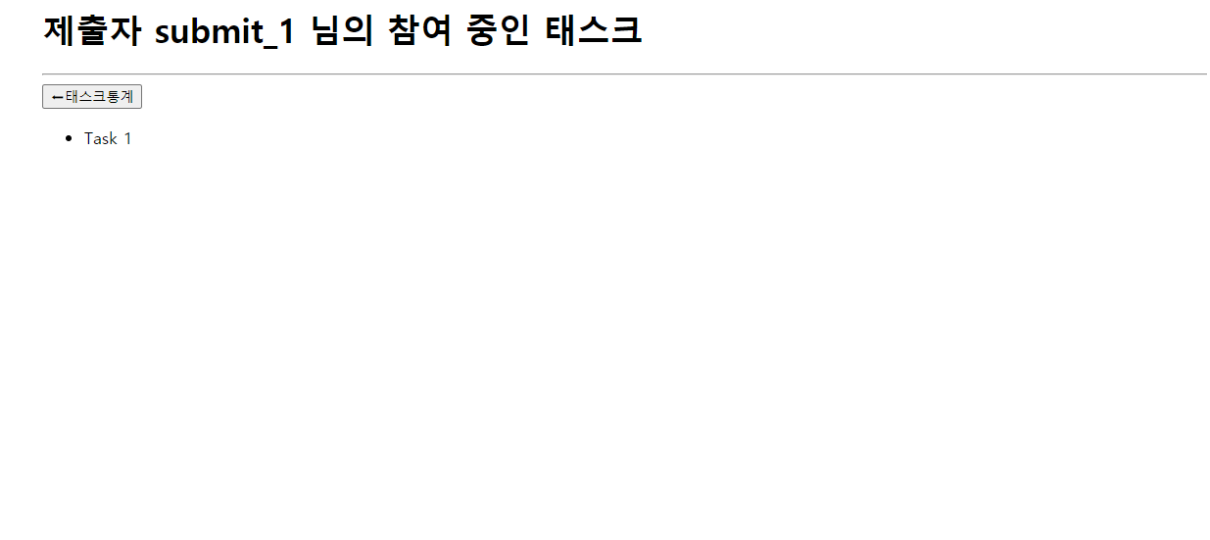


그림 34 '태스크 통계' 제출자별 참여중인 태스크 화면

그림 34은 '태스크 통계' 의 태스크 별 통계 화면에서 접근 가능한 제출자별 참여중인 태스크 목록으로, 해당 제출자가 참여하고 있는 태스크 목록이 보여진다. 이 때 각각의 태스크 이름을 클릭하면 태스크 별 통계 화면으로 돌아갈 수 있다..

1.2.1.4 회원 관리 및 통계

[실제 구현 UI]

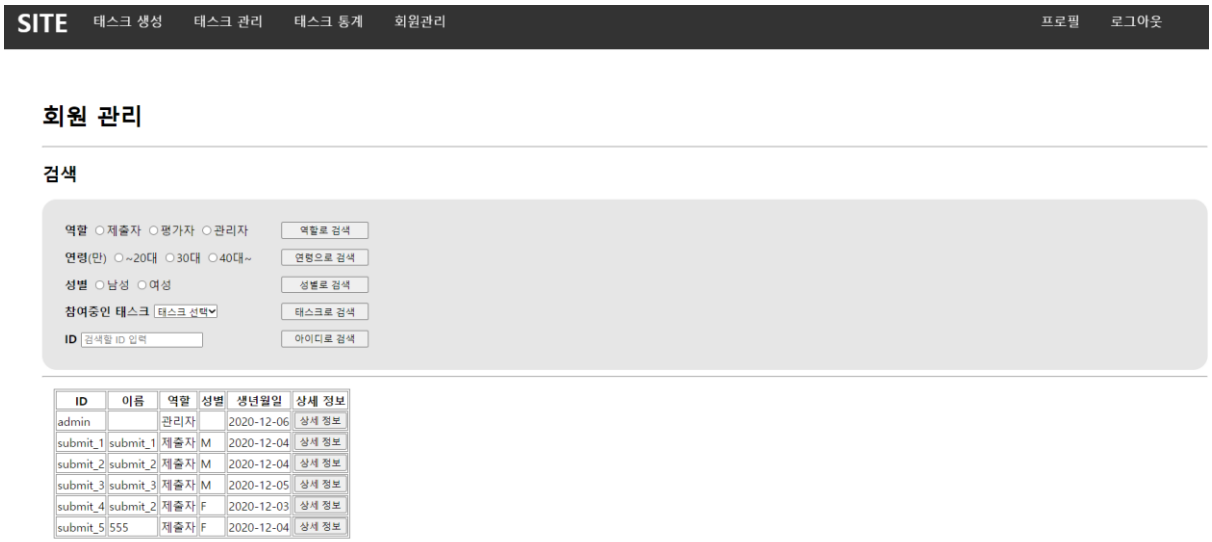


그림 35 관리자 [회원관리] UI

그림 35는 [회원관리]에서 볼 수 있는 UI이다. 관리자가 역할, 연령, 성별, 참여중인 태스크, ID를 기준으로 회원들을 검색할 수 있으며, [상세 정보] 클릭 시 회원에 대한 더 많은 정보를 볼 수 있다.

그림 36 관리자 [회원관리] 검색 결과

그림 36는 [회원관리]에서 검색을 하게 되면 나오는 결과 화면이다. [상세 정보] 클릭 시 회원에 대한 더 많은 정보를 볼 수 있다.

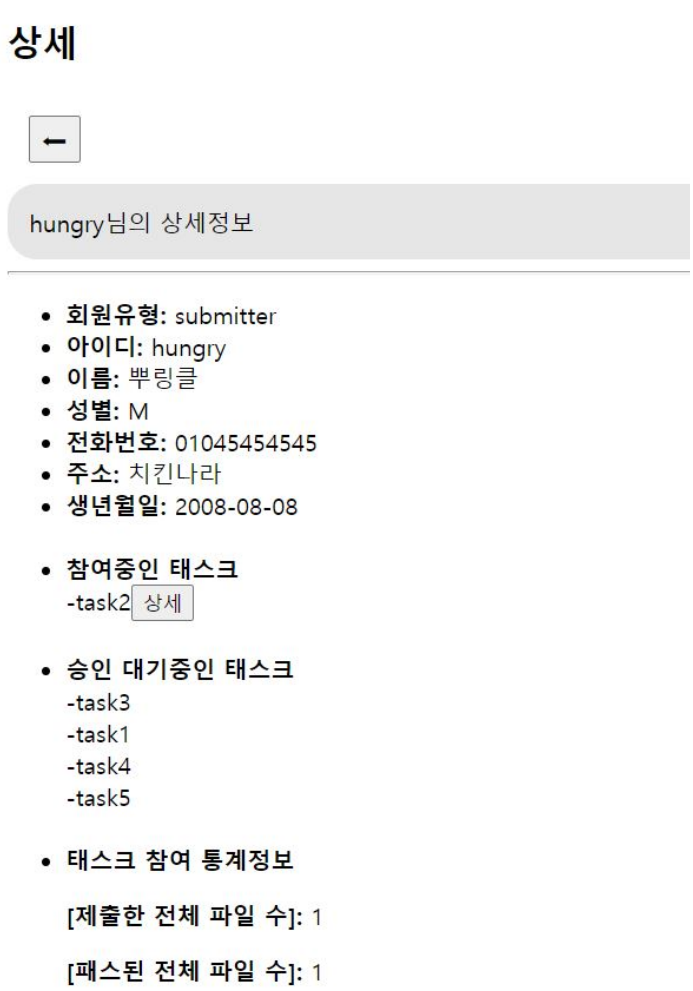


그림 37 관리자 [회원관리] 상세정보 화면 – 대상이 제출자인 경우

그림 37는 [회원관리]에서 대상이 제출자인 경우 회원별 상세정보를 클릭하면 볼 수 있는 UI이다. 기본 정보와 함께 태스크 참여와 관련된 통계 정보가 주어지게 된다. 참여 중인 태스크의 상세를 누르면 각각의 태스크 별로 제출한 파일 수와 패스된 파일 수가 보여지는 화면으로 연결된다.

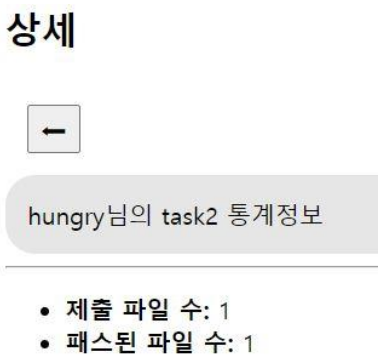


그림 38 관리자 [회원관리] 상세정보 화면 – 대상이 제출자일 경우 – 개별 태스크 통계 정보

그림 38는 그림 37에서 개별 태스크 옆에 나타나는 상세 버튼을 누를 경우 보여지는 페이지이다. 각각의 태스크 별로 제출 파일과 패스된 파일 수를 보여준다.

그림 39 관리자 [회원관리] 상세정보 화면 – 대상이 제출자일 경우

그림 39는 [회원관리]에서 대상이 평가자인 경우 회원별 상세정보를 클릭하면 볼 수 있는 UI이다. 기본 정보와 함께 평가한 파일의 목록 정보가 주어지게 된다. 평가한 파싱데이터시퀀스파일 목록 상세보기를 누르게 되면 평가에 대한 보다 자세한 정보가 보여진다.

회원 관리

blueberry님의 과거 평가 기록

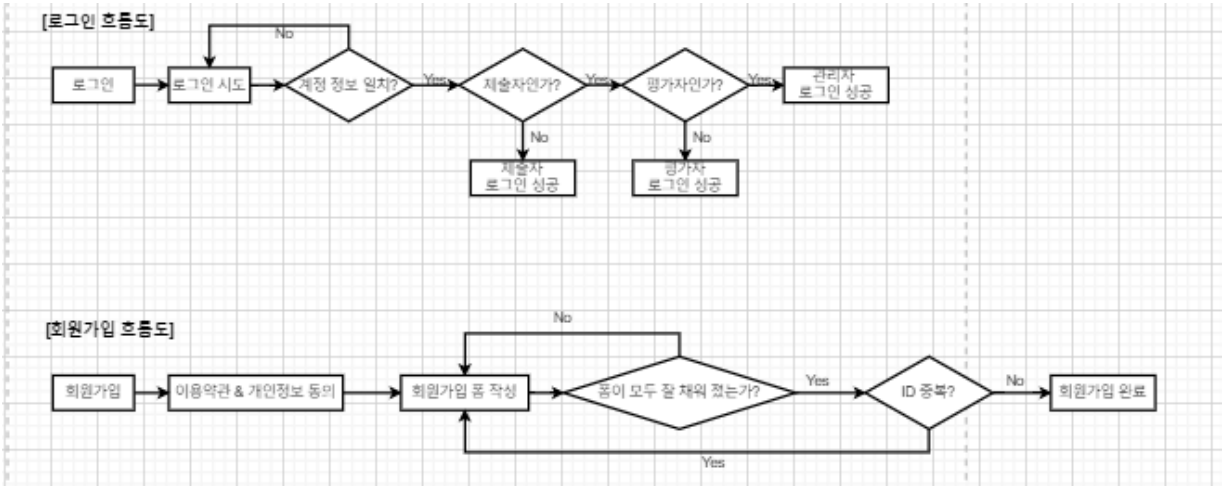
PD_ID	Task_Name	OGDATA_id	SubmitterId	Nth	Total tuple	Duplicated tuple	Null ratio	정성 평가	P/NP
1	task1	1	submitter	1	3	0	0	null	null
2	task1	2	submitter	2	3	0	0	null	null
3	task2	3	hungry	3	3	0	0	null	null

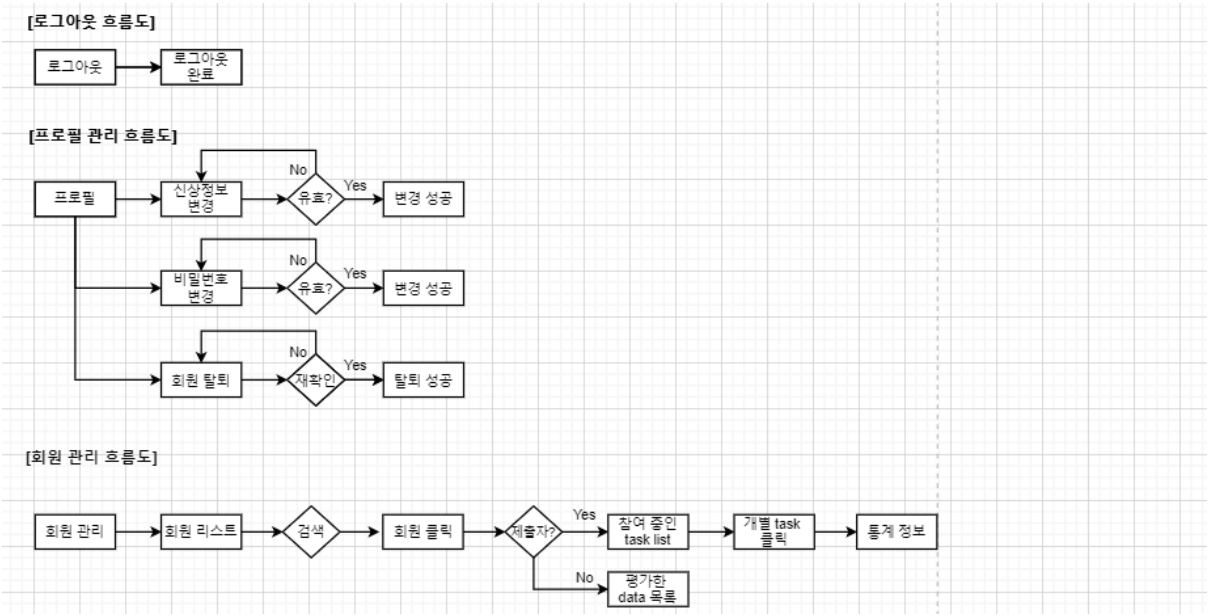
그림 40 관리자 [회원관리] 상세정보 화면 – 대상이 평가자일 경우 – 평가 기록 상세

그림 40는 그림 39에서 개별 태스크 옆에 나타나는 상세 버튼을 누를 경우 보여지는 페이지이다. 본인이 평가한 기록의 자세한 내용을 표로 제공한다.

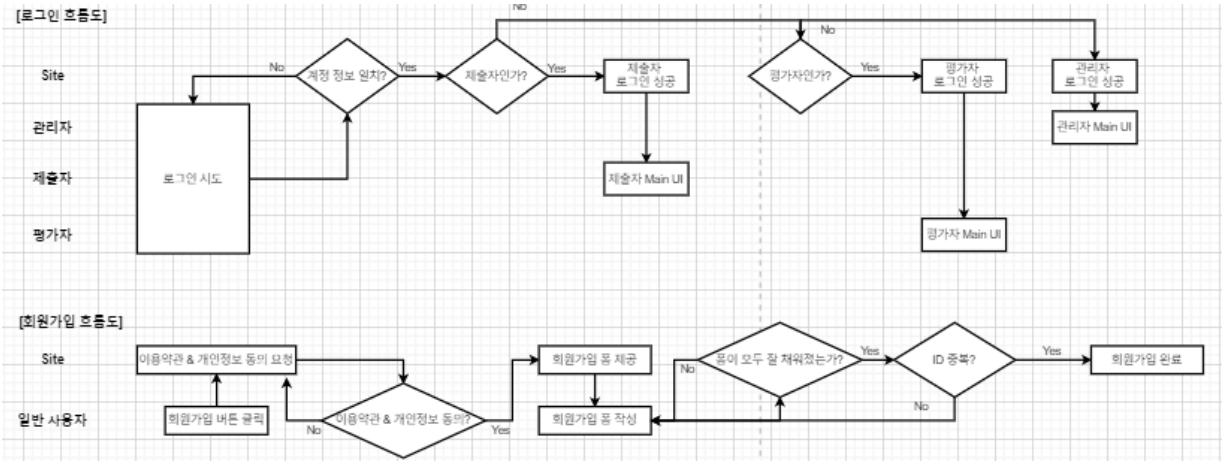
1.2.2 회원가입 및 인증 기능

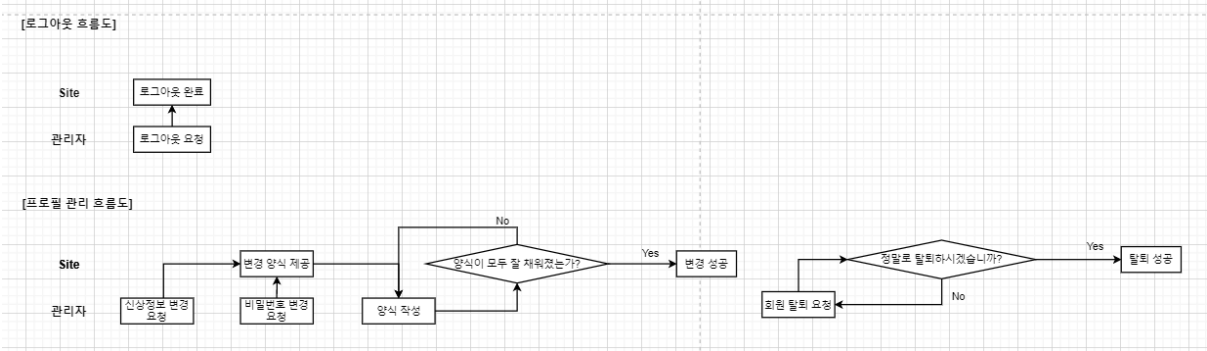
[Action 중심 flow chart]





[Actor 중심 flow chart]





[실제 구현 UI]

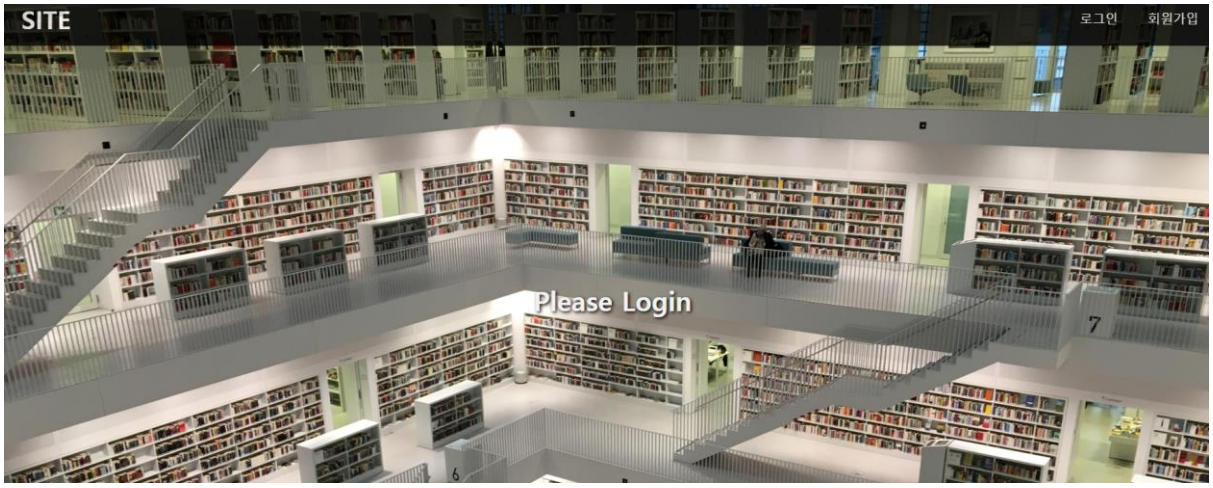


그림 41 로그인 전 화면

그림 41는 로그인 전에 뜨는 메인화면 UI이다. 상단 바를 통하여 로그인과 회원가입 화면으로 접근이 가능하다.

그림 42 이용약관 및 개인정보 동의

그림 42는 회원가입 시작 시 보여지는 이용약관 및 개인정보 동의 화면이다. 이용약관 및 개인정보 동의에 대한 동의서를 받고 있다. 동의함 버튼을 클릭하면 회원가입 화면으로 진행된다.

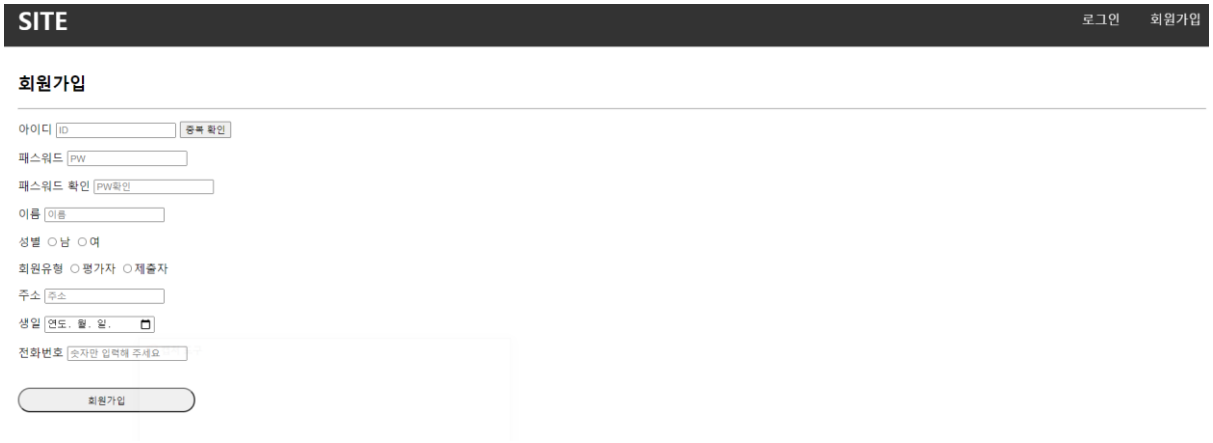


그림 43 회원가입 화면

그림 43는 회원가입이 진행되는 화면이다. 정해진 폼에 맞추어 제대로 작성하지 않으면 회원가입이 진행되지 않는다. 제대로 작성된 경우 회원가입이 완료된다.

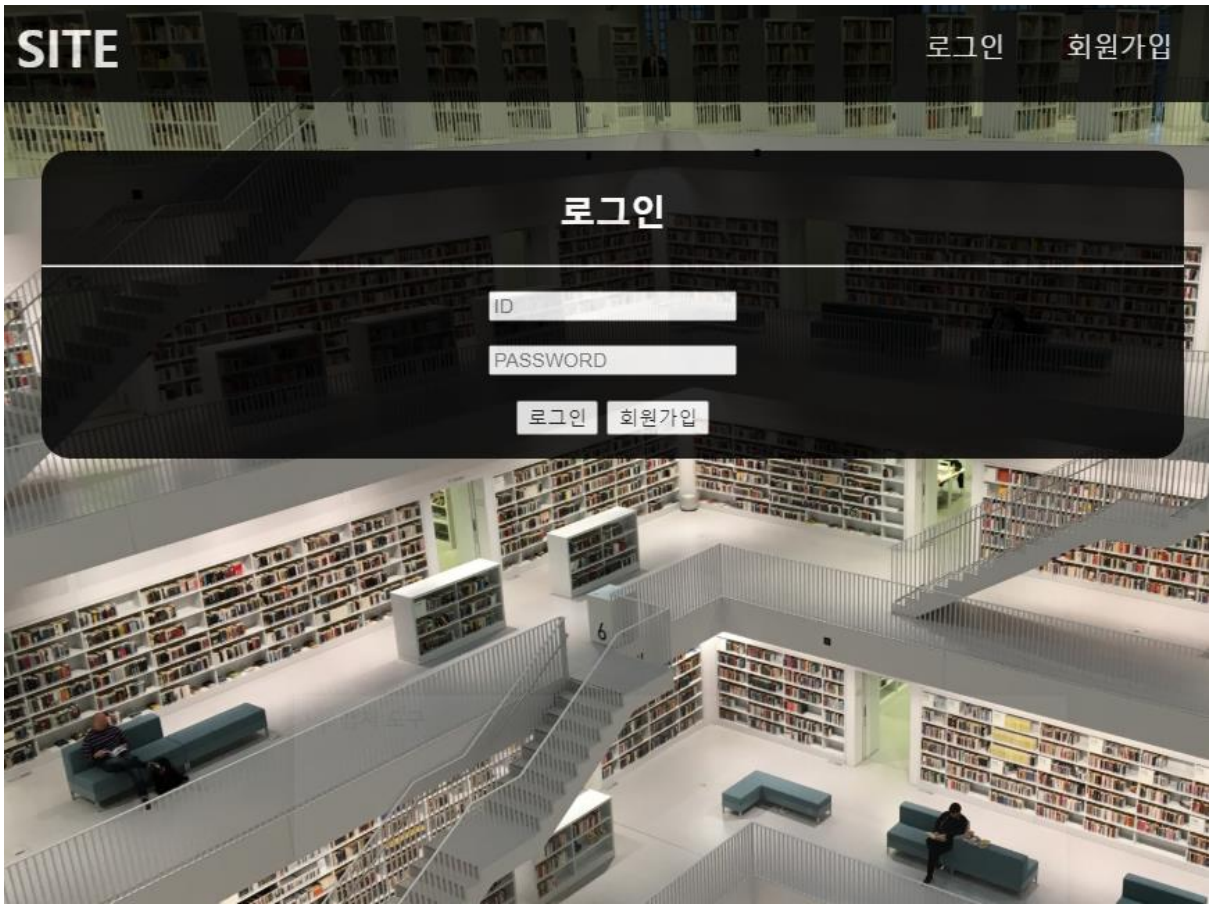


그림 44 로그인 화면

그림 44는 로그인이 진행되는 화면이다. 정확한 ID와 Password를 제출할 경우 로그인이 진행되고 그렇지 않은 경우 에러 메시지가 보여진다. 회원가입 버튼으로 회원가입 화면으로 접근이 가능하다.

그림 45 제출자 로그인 시 상단 바

그림 46 평가자 로그인 시 상단 바

그림 47 관리자 로그인 시 상단 바

그림 45~47은 회원 유형별로 달라지는 상단 바이다. 로그인한 회원에 유형 별로 UI가 각각 제공되고 있다.

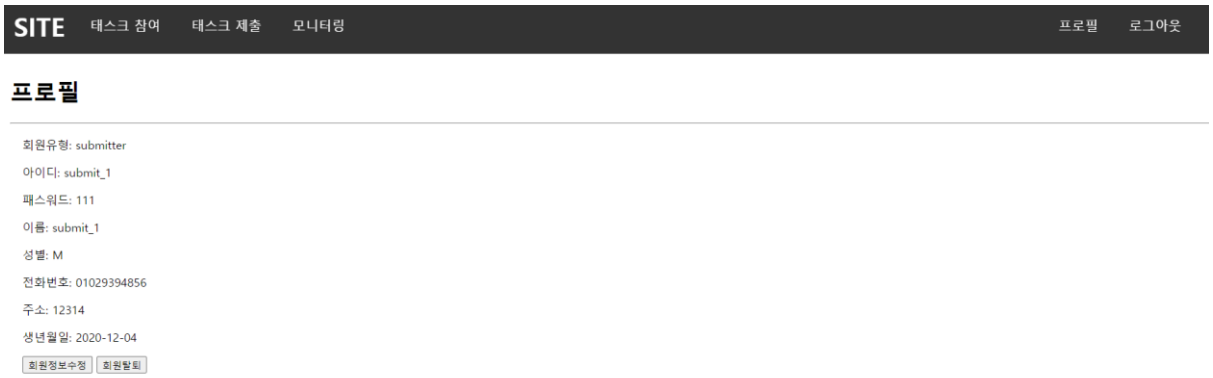


그림 48 공통 [프로필] UI

그림 48는 [프로필]로 이동 후 볼 수 있는 UI의 예시이며 하단의 버튼을 통해 회원정보수정, 회원탈퇴 기능에 접근할 수 있다.

그림 49 회원정보 수정 화면

그림 49는 [프로필]로 이동 후 하단의 버튼을 통해 접근할 수 있는 회원정보수정 화면으로 각각의 정보를 변경하기를 통해 변경할 수 있다. 하단의 메인페이지로 버튼을 통해 각각의 회원 유형별 메인페이지로 이동한다.

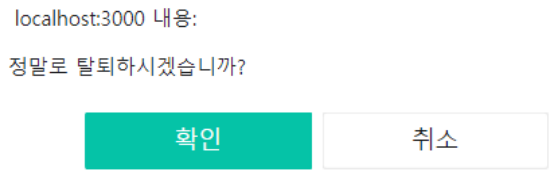


그림 50 회원탈퇴 메시지

그림 50는 [프로필]로 이동 후 하단의 버튼을 통해 접근할 수 있는 회원탈퇴 메시지로 확인을 클릭하게 되면 회원 탈퇴가 진행된다.

2. 변경 사항

2.1 소주제 1

- [제출자 원본데이터시퀀스파일 constraint 확인후 alert 메세지]

제출자가 constraint에 맞지 않는 원본데이터시퀀스파일을 제출할 경우, 시스템이 리디렉션 후 alert를 보여주고 사유를 명시해 주기 때문에 제출자가 본인의 데이터의 integrity를 확인하기 쉬워졌다.

- [평가하기 페이지 할당 파일 목록의 UI 변경]

평가하기 페이지에서 할당된 파일 목록을 확인할 수 있도록 초기 UI를 설정했으나, 가독성을 위해서 평가하기 페이지 내에는 하나의 할당된 파일 목록이 뜨게 구성하였다. 여러 개의 파일이 할당되었다면, 파일들을 순차적으로 평가하는 방법으로 모든 파일을 평가하는 것이다.

2.2 소주제 2

- [관리자 상단 바] 관리자 상단 바에 '태스크' 메뉴 세분화

이를 통하여 태스크로 들어가 세부 메뉴 (생성, 관리, 통계)로 들어가야 하는 불편함을 덜어 UI를 개선하였다. 이제 관리자 상단 바를 이용해 바로 태스크 생성, 태스크 관리, 태스크 통계로 접근 가능하다.

- [관리자 기능 – 태스크 생성] 원본 데이터 타입 정보 입력 시 UI 개선

기존에 속성이름과 데이터형, constraints, 태스크 데이터 테이블 매핑 정보까지 모두 입력해주어야 했던 UI를 맵핑되는 속성 이름만 입력하여도 그 속성이 갖고 있는 데이터형과 constraints를 따라가게 바꾸었다. 이를 통해 일일이 맵핑을 해야 하는 번거로움을 개선하였다.

- [관리자 기능 – 태스크 생성] 원본 데이터 타입 정보 입력 시 UI 개선

기존에 속성이름과 데이터형, constraints, 테스크 데이터 테이블 매핑 정보까지 모두 입력해주어야 했던 UI를 맵핑되는 속성 이름만 입력하여도 그 속성이 갖고 있는 데이터형과 constraints를 따라가게 바꾸었다. 이를 통해 관리자가 일일이 맵핑을 해야 하는 번거로움을 개선하였다.

- [관리자 기능 – 태스크 관리] 원본 데이터 타입 추가 정보 입력 시 UI 개선

기존에 속성이름과 데이터형, constraints, 테스크 데이터 테이블 매핑 정보까지 모두 입력해주어야 했던 UI를 맵핑되는 속성 이름만 입력하여도 그 속성이 갖고 있는 데이터형과 constraints를 따라가게 바꾸었다. 이를 통해 관리자가 일일이 맵핑을 해야 하는 번거로움을 개선하였다.

- [관리자 기능 – 태스크 관리] 기존 태스크 통계 부분에 있었던 태스크 데이터 테이블 다운로드 기능 태스크 관리로 이동

중간 보고서에 태스크 관리에서 구현되었던 다운로드 기능이 태스크 관리 기능으로 이동하였다. 이는 다운로드 기능이 통계보다는 관리에 적합하다고 생각했기 때문이다.

- [회원가입 및 인증기능 - 프로필] 신상정보 변경과 비밀번호 변경을 회원정보 수정으로 통합

중간 보고서에 따로 구현되었던 부분을 UI 상 통합되어 있는 것이 편리하다고 판단하여 통합하였다.

- [회원가입 및 인증기능 – 회원 탈퇴] 회원탈퇴 기능 간소화

중간 보고서에서는 따로 회원 탈퇴 화면을 제공하였으나, UI 상 빠르게 탈퇴가 가능한 것이 나을 것이라고 판단하여 따로 화면을 제공하지 않고 팝업창으로 탈퇴 시 보다 편리하게 기능을 간소화하였다.

IV. Database table 생성 script 명기

```
-- MySQL Workbench Forward Engineering


SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ER
ROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';


-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema crowdsourcing

-- -----------------------------------------------------


-- -----------------------------------------------------

-- Schema crowdsourcing

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `crowdsourcing` DEFAULT CHARACTER SET utf8 ;

USE `crowdsourcing` ;


-- -----------------------------------------------------

-- Table `crowdsourcing`.`account`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `crowdsourcing`.`account` (
```

```
  `AccID` VARCHAR(45) NOT NULL,
  `Password` VARCHAR(45) NULL DEFAULT NULL,
  `Name` VARCHAR(45) NULL DEFAULT NULL,
  `Gender` VARCHAR(1) NULL DEFAULT NULL,
  `Phone_Number` VARCHAR(11) NULL DEFAULT NULL,
  `Address` VARCHAR(45) NULL DEFAULT NULL,
  `DOB` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`AccID`),
  UNIQUE INDEX `AccType_UNIQUE` (`AccID` ASC),
  INDEX `Eval_Foreign` (`AccID` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `crowdsourcing`.`administrator`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`administrator` (
  `AccID_Admin` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`AccID_Admin`),
  INDEX `AccType_Admin_idx` (`AccID_Admin` ASC),
  CONSTRAINT `Admin_Foreign`
    FOREIGN KEY (`AccID_Admin`)
    REFERENCES `crowdsourcing`.`account` (`AccID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
```

```
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`evaluator`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`evaluator` (
  `AccID_Eval` VARCHAR(45) NOT NULL,
  `AccType` VARCHAR(15) NULL DEFAULT 'Evaluator',
  PRIMARY KEY (`AccID_Eval`),
  INDEX `AccType_Eval_idx` (`AccType` ASC, `AccID_Eval` ASC),
  CONSTRAINT `Eval_Foreign`
    FOREIGN KEY (`AccID_Eval`)
    REFERENCES `crowdsourcing`.`account` (`AccID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`pdsequencefile`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`pdsequencefile` (
  `PD_ID` INT NOT NULL,
```

```
  `SubmitterId` VARCHAR(45) NULL DEFAULT NULL,
  `TaskName` VARCHAR(45) NULL DEFAULT NULL,
  `Ogdata_Id` INT NULL DEFAULT NULL,
  `TotalTuple` INT NULL DEFAULT NULL,
  `DuplicatedTuple` INT NULL DEFAULT NULL,
  `NullRatio` JSON NULL DEFAULT NULL,
  `Evaluated` VARCHAR(1) NULL DEFAULT NULL,
  `PD_File` JSON NULL DEFAULT NULL,
  PRIMARY KEY (`PD_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`task`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`task` (
  `TD_ID` INT NOT NULL AUTO_INCREMENT,
  `TName` VARCHAR(45) NOT NULL,
  `Explanation` TEXT NULL DEFAULT NULL,
  `Upload_Interval` INT NULL DEFAULT NULL,
  `TDT_SChema` JSON NOT NULL,
  `Guideline` TEXT NULL DEFAULT NULL,
  PRIMARY KEY (`TD_ID`),
  INDEX `index3` (`TName` ASC))
ENGINE = InnoDB
```

```
AUTO_INCREMENT = 14
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`taskdatatable`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`taskdatatable` (
  `TD_ID` INT NOT NULL DEFAULT '1',
  `Submitter_ID` VARCHAR(45) NULL DEFAULT NULL,
  `TD_File` JSON NULL DEFAULT NULL,
  PRIMARY KEY (`TD_ID`),
  CONSTRAINT `TD_ID`
    FOREIGN KEY (`TD_ID`)
    REFERENCES `crowdsourcing`.`task` (`TD_ID`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`evaluation_info`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`evaluation_info` (
  `Parsing_ID` INT NOT NULL,
  `Evaluator_ID` VARCHAR(45) NOT NULL,
  `Evaluation_Time` DATETIME NOT NULL,
```

```
  `QualityScore` INT NULL DEFAULT NULL,
  `P_NP` VARCHAR(1) NULL DEFAULT NULL,
  `TaskD_ID` INT NULL DEFAULT NULL,
  PRIMARY KEY (`Parsing_ID`, `Evaluator_ID`, `Evaluation_Time`),
  INDEX `evaluation_idx` (`Evaluator_ID` ASC),
  INDEX `Result_in_idx` (`TaskD_ID` ASC),
  CONSTRAINT `Evaluator_ID`
    FOREIGN KEY (`Evaluator_ID`)
    REFERENCES `crowdsourcing`.`evaluator` (`AccID_Eval`)
    ON UPDATE CASCADE,
  CONSTRAINT `Parsing_ID`
    FOREIGN KEY (`Parsing_ID`)
    REFERENCES `crowdsourcing`.`pdsequencefile` (`PD_ID`),
  CONSTRAINT `TaskD_ID`
    FOREIGN KEY (`TaskD_ID`)
    REFERENCES `crowdsourcing`.`taskdatatable` (`TD_ID`)
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`evaluating`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`evaluating` (
  `Evaluator_ID` VARCHAR(45) NOT NULL,
```

```
  `PDS_ID` INT NOT NULL,
  `Eval_Time` DATETIME NOT NULL,
  PRIMARY KEY (`PDS_ID`, `Eval_Time`, `Evaluator_ID`),
  INDEX `Foreign_PD_idx` (`PDS_ID` ASC),
  INDEX `Foreign_Info1` (`Evaluator_ID` ASC, `PDS_ID` ASC, `Eval_Time` ASC),
  CONSTRAINT `Foreign_Eval1`
    FOREIGN KEY (`Evaluator_ID`)
    REFERENCES `crowdsourcing`.`evaluator` (`AccID_Eval`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Foreign_Info1`
    FOREIGN KEY (`Evaluator_ID` , `PDS_ID` , `Eval_Time`)
    REFERENCES `crowdsourcing`.`evaluation_info` (`Evaluator_ID` , `Parsing_ID` , `Evaluation_Time`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Foreign_PD1`
    FOREIGN KEY (`PDS_ID`)
    REFERENCES `crowdsourcing`.`pdsequencefile` (`PD_ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`submitter`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`submitter` (
  `AccID_Submit` VARCHAR(45) NOT NULL,
  `Evaluation` INT NULL DEFAULT NULL,
  PRIMARY KEY (`AccID_Submit`),
  INDEX `Submit_Foreign_idx` (`AccID_Submit` ASC),
  CONSTRAINT `Submit_Foreign`
    FOREIGN KEY (`AccID_Submit`)
    REFERENCES `crowdsourcing`.`account` (`AccID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`ogdatatype`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`ogdatatype` (
  `ID` INT NOT NULL DEFAULT '1',
  `Og_Schema` JSON NOT NULL,
  `TaskName` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC),
  INDEX `TaskName_idx` (`TaskName` ASC),
  CONSTRAINT `TaskName`
```

```
    FOREIGN KEY (`TaskName`)
    REFERENCES `crowdsourcing`.`task` (`TName`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`odsequencefile`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`odsequencefile` (
  `SerialNumber` INT NOT NULL AUTO_INCREMENT,
  `AccountID` VARCHAR(45) NULL DEFAULT NULL,
  `OGDT_ID` INT NULL DEFAULT NULL,
  `PDS_ID` INT NULL DEFAULT NULL,
  `OD_File` JSON NULL DEFAULT NULL,
  PRIMARY KEY (`SerialNumber`),
  INDEX `fk_ODSEQUENCEFILE_OGDATATYPE1_idx` (`OGDT_ID` ASC),
  INDEX `fk_ODSEQUENCEFILE_PDSEQUENCEFILE(DecidedLater)1_idx` (`PDS_ID` ASC),
  INDEX `AccountID_idx` (`AccountID` ASC),
  CONSTRAINT `AccountID`
    FOREIGN KEY (`AccountID`)
    REFERENCES `crowdsourcing`.`submitter` (`AccID_Submit`)
    ON DELETE SET NULL
    ON UPDATE CASCADE,
  CONSTRAINT `OGDT_ID`
    FOREIGN KEY (`OGDT_ID`)
```

```
    REFERENCES `crowdsourcing`.`ogdatatype` (`ID`)
    ON UPDATE CASCADE,
  CONSTRAINT `PDS_ID`
    FOREIGN KEY (`PDS_ID`)
    REFERENCES `crowdsourcing`.`pdsequencefile` (`PD_ID`))
ENGINE = InnoDB
AUTO_INCREMENT = 18
DEFAULT CHARACTER SET = utf8;



-- -----------------------------------------------------
-- Table `crowdsourcing`.`participates`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `crowdsourcing`.`participates` (
  `Sub_ID` VARCHAR(45) NOT NULL,
  `Task_Name` VARCHAR(45) NOT NULL,
  `Approved` INT NULL DEFAULT '0',
  INDEX `fk_Submitter_has_Task_Submitter1_idx` (`Sub_ID` ASC),
  INDEX `Task_Name_idx` (`Task_Name` ASC),
  CONSTRAINT `Part_AccID`
    FOREIGN KEY (`Sub_ID`)
    REFERENCES `crowdsourcing`.`submitter` (`AccID_Submit`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `Part_Task_Name`
    FOREIGN KEY (`Task_Name`)
```

```
    REFERENCES `crowdsourcing`.`task` (`TName`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```
변경된 script 명기

V. 시연을 위해 Database에 저장된 정보들

<계정>

- 관리자

아이디	비밀번호
admin	admin

- 제출자

아이디	비밀번호
submitter	1111

- 평가자

아이디	비밀번호
evaluator	1111

- 태스크

태스크 이름
CardLog

Admin 계정과 함께 전체 table에 예시 data들이 일부 들어가 있다.