

**2020-2 데이터베이스**

**프로젝트**

**최종 보고서**

**18 청춘(일팔 청춘)**

김채현, 박시원, 이혜지,

차승연, 최민수, 한승희

# <목차>

\* 중간 보고서 이후에 변경사항이 많아 중간 보고서 내용도 수정하여 같이 작성하였습니다.\*

## I. ER diagram & Mapping to Relational Schema

1. Mini-World
2. ER diagram
3. Mapping to relational schema

## II. 시스템 설계 문서

## III. 프로젝트 구현 진행 과정 및 역할 분담

## IV. 프로젝트 설명

1. 기능별 구현 내용
  - 1) 소주제 1 : 제출자, 평가자 기능
  - 2) 소주제 2 : 관리자 기능

# I. ER diagram & Mapping to Relational Schema

## 1. Mini-World

<p><b>사용자</b></p> <p>종류: 관리자, 제출자, 평가자</p> <p>회원 가입 시 제출하는 정보</p> <p>공통: 아이디, 비밀번호</p> <p>제출자: 이름, 생년월일, 주소, 성별, 전화번호</p> <p>평가자: 이름, 생년월일, 주소, 성별, 전화번호</p>	<p><b>관리자</b></p> <p>역할: 1) 관리자 UI로 필요한 태스크를 생성 2) 제출자가 제출한 원본 데이터 파일의 타입 설정 3) 태스크에 참여중인 제출자와 평가자의 평가 점수 등을 확인하여 식당 방문자의 코로나 확진 여부를 관리</p>
<p><b>평가자</b></p> <p>역할: 자신에게 주어진 파싱 데이터 시퀀스 파일을 정성적으로 평가한다.</p> <p>평가 결과는 파싱 데이터 시퀀스 파일에 기록되어 관리자가 볼 수 있다.</p>	<p><b>태스크</b></p> <p>태스크 제공자: 제출자</p> <p>태스크 종류: 식당 방문기록 / 보건소 코로나 검사기록</p> <p>태스크는 태스크의 이름에 따라 분류되며, 태스크에 대한 설명과 최소 업로드 주기를 가진다.</p> <p>태스크는 원본 데이터 시퀀스 파일로 업로드되어, 파싱 데이터 시퀀스 파일로 시스템에 의해서 자동으로 변환되어 테이블 스키마 형태를 가지게 된다. 이 파일들은 고유 파일 ID를 가지게 된다.</p>

## 식당 방문기록 태스크

제출자: 식당 주인 또는 식당 관계자

추가적으로 갖는 정보: 식당 전화번호, 식당 주소, 식당 이름

필수적으로 기입해야하는 정보: 방문자 전화번호, 주소, 방문 날짜, 방문 시각

제출 주기: 매일

원본 데이터 타입 종류: 1 가지 이상 존재할 수 있으며, 필수적으로 기입해야하는 정보는 모든 원본 데이터 타입에 포함되어 있다.

가정: 방문자는 모두 1 개의 유일한 전화번호를 가지고, 같은 식당을 같은 날짜에 2 번이상 방문하는 경우는 없다.

## 보건소 코로나 검사기록 태스크

제출자: 보건소

필수적으로 기입해야하는 정보: 검사자 전화번호, 주소, 검사 날짜, 코로나 음성 여부

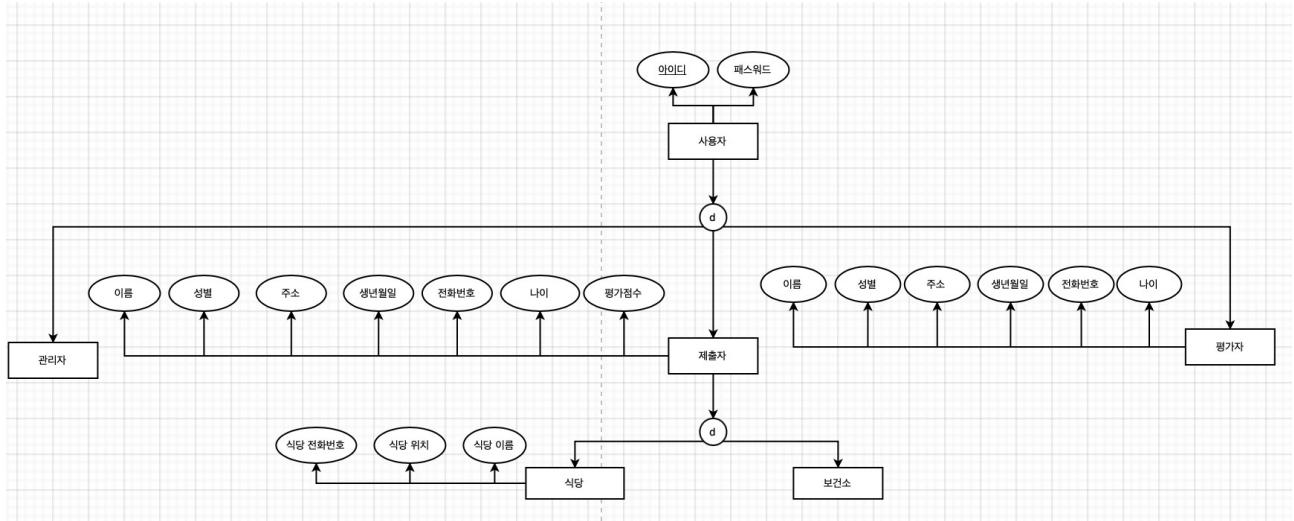
제출 주기: 매일

원본 데이터 타입 종류: 검사자 전화번호, 주소, 검사날짜, 코로나 음성 여부 이렇게 1 가지의 원본 데이터 타입만이 존재한다.

가정: 이 프로젝트에선 오로지 한 개의 보건소가 존재한다.

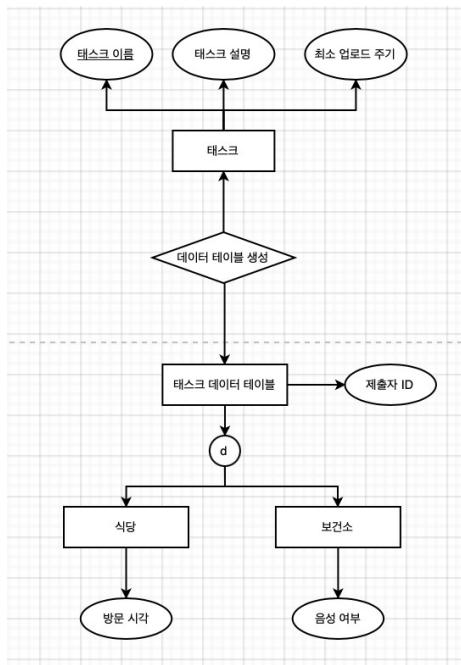
## 2. ER diagram

### 1) 사용자 entity



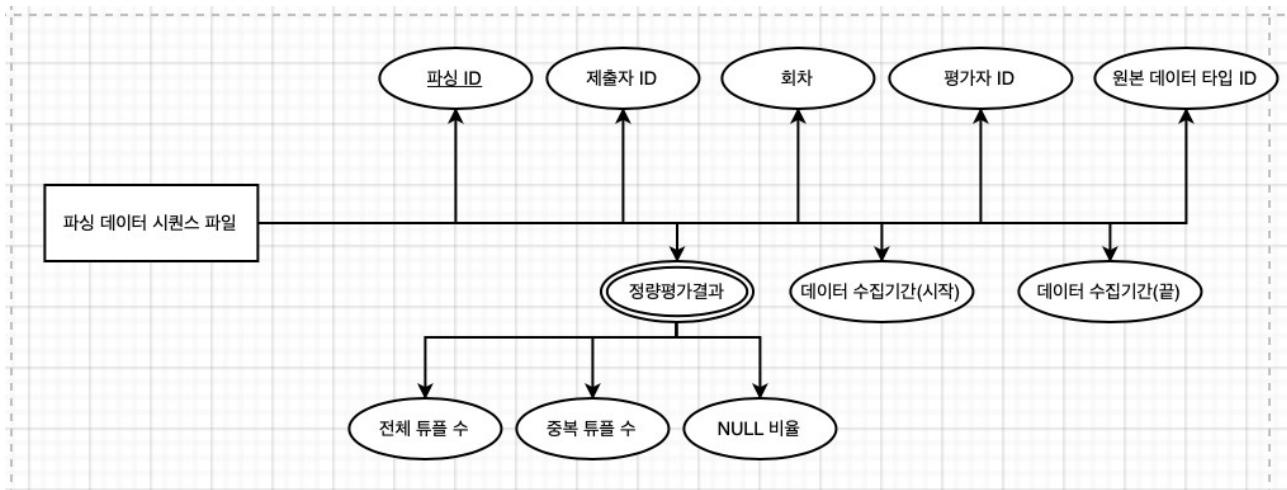
전체 사용자는 공통 속성으로 고유 ID, 비 번호를 가진다. 이들은 관리자, 제출자, 평가자 세 개의 disjoint subclass로 역할이 나뉘어진다. 제출자는 이름, 성별, 주소, 생년월일, 전화번호, 나이, 평가 점수라는 부가적인 속성을 가지며, 제출자는 disjoint subclass로 식당과 보건소로 나뉜다. 식당은 식당 전화번호, 식당 위치, 식당 이름이라는 부가적인 속성을 가진다. 평가자는 이름, 성별, 주소, 생년월일, 전화번호, 나이라는 부가적인 속성을 가진다.

## 2) 태스크 entity



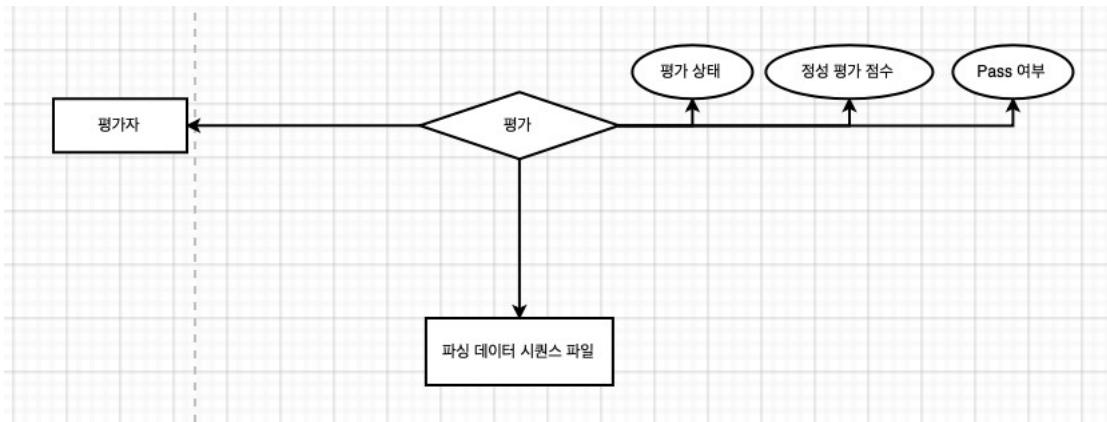
태스크는 key 속성인 태스크 이름과 최소 업로드 주기, 태스크 설명을 가진다. 태스크는 제출자인 식당이나 보건 소가 태스크를 제출하면 제출자의 아이디를 가진 태스크 데이터 테이블이 형성된다. 식당 태스크 데이터 테이블 일 경우에는 방문 시각을 부가적인 속성으로 가지게 되고, 보건소 태스크 데이터 테이블일 경우에는 음성 여부를 부가적인 속성으로 가지게 된다.

## 3) 파싱 데이터 시퀀스 파일



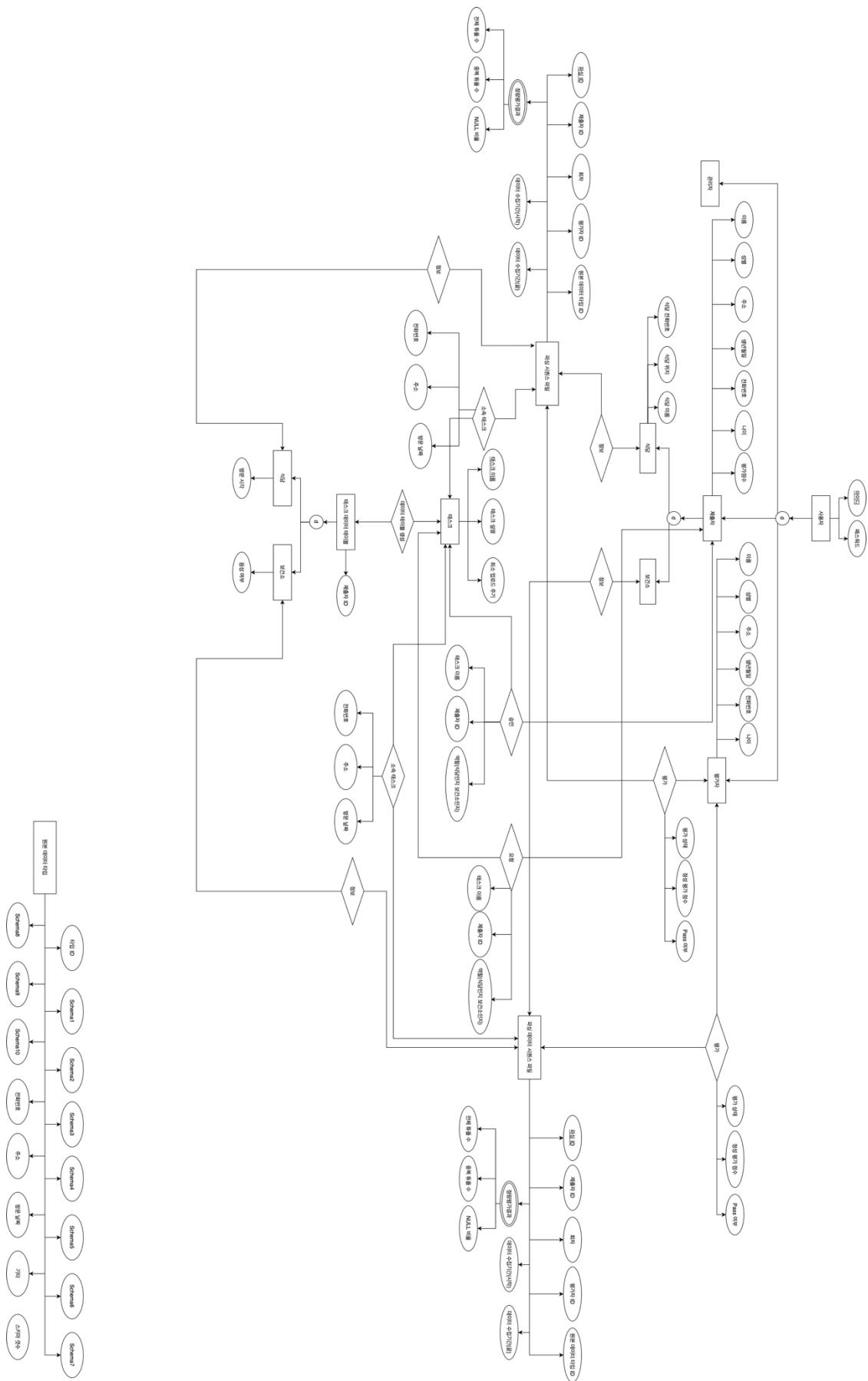
파싱 데이터 시퀀스 파일은 원본 데이터 시퀀스 파일이 제출되면, 시스템에서 자동으로 원본 데이터 시퀀스 파일이 파싱 데이터 시퀀스 파일로 변환된다. 파싱 데이터 시퀀스 파일은 보건소와 식당 2 종류로 되어 있으며, 이들이 가지는 속성은 위와 같다.

#### 4) 평가자의 평가



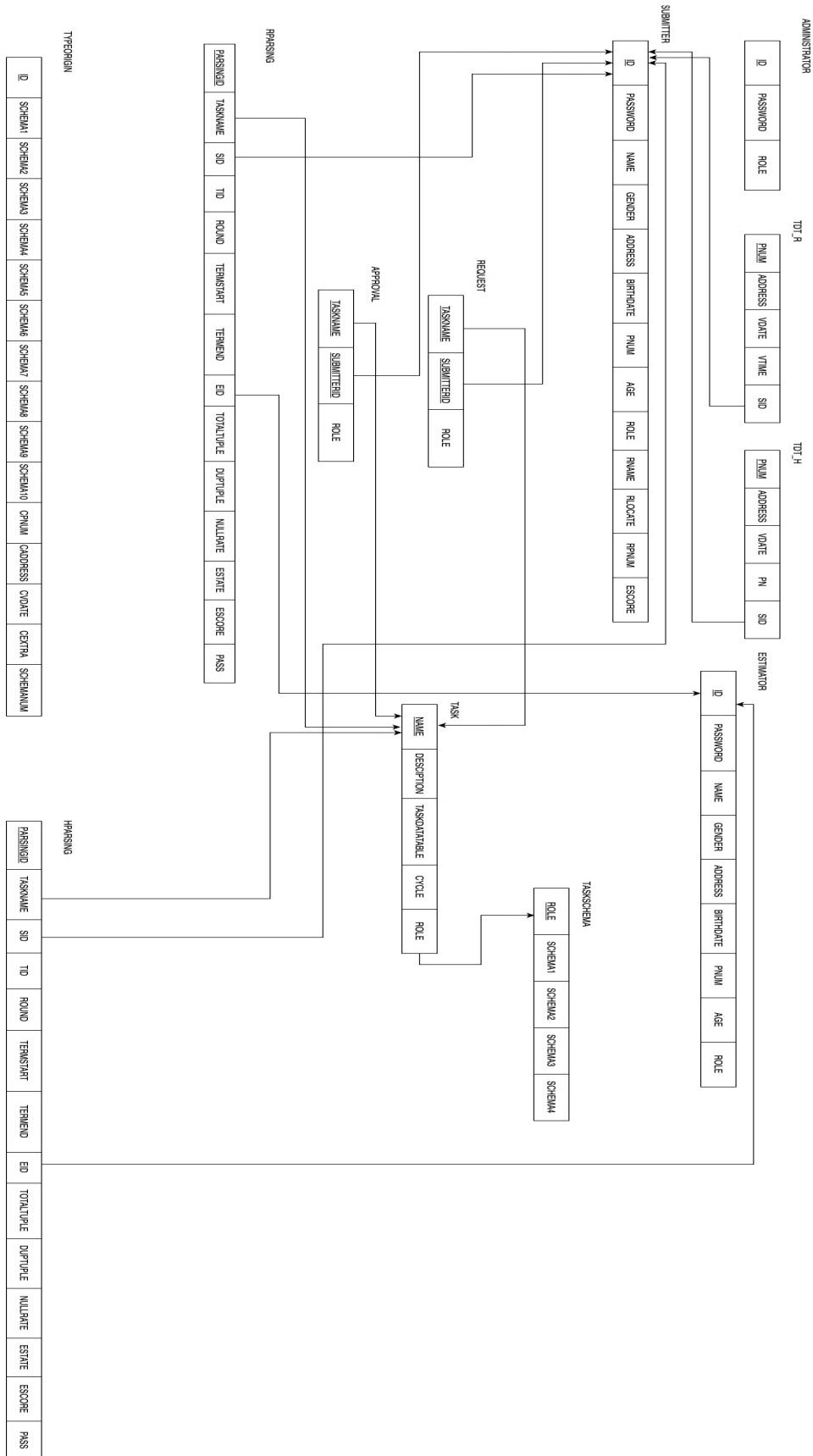
두 태스크 모두에서 평가자가 자신에게 주어진 파싱된 데이터 시퀀스 파일을 평가하게 되고, 그 결과를 파싱 데이터 시퀀스 파일로 보낸다. 평가는 평가 여부, 정성평가 점수, P/NP 를 속성으로 가진다.

## 5) 전체 ERD



### 3. Mapping to Relational Schema

- Relation Schema



- 각 relation에 대한 domain constraints
- 맥줄 친 속성 = primary key
- Bold 된 속성 = foreign key
- 관리자(ADMINISTRATOR)

ID : VARCHAR

PASSWORD : VARCHAR

ROLE : ENUM(A)

- 제출자(SUBMITTER)

ID : VARCHAR

PASSWORD : VARCHAR

NAME : VARCHAR (한국어)

GENDER : CHAR (F/M)

ADDRESS : ENUM('강남', '서대문', '서초', '마포', '용산')

BIRTHDATE : DATE

PNUM : CHAR(11)

AGE : INT

ROLE : ENUM(R/H) \*R = 식당, H = 보건소

RNAME : VARCHAR (한/영)

RLOCATE : ENUM('용산', '서대문', '강남')

RPNUM : CHAR(11)

ESCORE: INT

- 평가자(ESTIMATOR)

ID : VARCHAR

PASSWORD : VARCHAR

NAME : VARCHAR (한국어)

GENDER : CHAR (F/M)

ADDRESS : ENUM('강남', '서대문', '서초', '마포', '용산')

BIRTHDATE : DATE

PNUM : CHAR(11)

AGE : INT

ROLE : ENUM(E)

- 태스크 속성(TASKSCHEMA)

ROLE: ENUM(R/H) \*R = 식당, H = 보건소

SCHEMA1: VARCHAR

SCHEMA2: VARCHAR

SCHEMA3: VARCHAR

SCHEMA4: VARCHAR

- 요청(REQUEST)

TASKNAME : VARCHAR (TASK'S NAME)

SUBMITTERID: VARCHAR (SUBMITTER'S ID)

ROLE: ENUM(R/H) \* R = 식당, H = 보건소

- 태스크(TASK)

NAME : VARCHAR (한/영)

DESCRIPTION: VARCHAR

TASKDATATABLE: VARCHAR

CYCLE : TIME

**ROLE: ENUM(R/H) \*R = 식당, H = 보건소 (TASKSCHEMA'S ROLE)**

- 승인(APPROVAL)

**TASKNAME : VARCHAR (TASK'S NAME)**

**SUBMITTERID: VARCHAR (SUBMITTER'S ID)**

ROLE: ENUM(R/H) \* R = 식당, H = 보건소

- 원본 데이터 타입(TYPEORIGIN)

ID: INT

SCHEMA1: VARCHAR

SCHEMA2: VARCHAR

SCHEMA3: VARCHAR

SCHEMA4: VARCHAR

SCHEMA5: VARCHAR

SCHEMA6: VARCHAR

SCHEMA7: VARCHAR

SCHEMA8: VARCHAR

SCHEMA9: VARCHAR

SCHEMA10: VARCHAR

CNUM: INT

CADDRESS: INT

CVDATE: DATE

CEXTRA: INT

SCHEMANUM: INT

- 식당 파싱 데이터 시퀀스 파일(RPARSING)

PARSINGID : INT

**TASKNAME: VARCHAR(TASK'S NAME)**

**SID : VARCHAR(SUBMITTER'S ID)**

TID: INT

ROUND : INT

TERMSTART : DATE

TERMEND: DATE

**EID : VARCHAR (ESTIMATOR's ID)**

TOTALTUPLE : INT

DUPTUPLE : INT

NULLRATE : DOUBLE

ESTATE : ENUM(Y/N)

ESCORE : INT

PASS : ENUM (P/N)

- 식당 태스크 데이터 테이블(TDT\_R) \*R 에는 식당 태스크를 제출한 제출자의 ID 가 들어갑니다

PNUM: VARCHAR

ADDRESS: ENUM('강남', '서대문', '서초', '마포', '용산')

VDATE: DATE

VTIME: TIME

**SID: VARCHAR(SUBMITTER'S ID)**

- 보건소 파싱 데이터 시퀀스 파일(HPARSING)

PARSINGID : INT

**TASKNAME: VARCHAR(TASK'S NAME)**

**SID : VARCHAR(SUBMITTER'S ID)**

TID: INT

ROUND : INT

TERMSTART : DATE

TERMEND: DATE

**EID : VARCHAR (ESTIMATOR's ID)**

TOTALTUPLE : INT

DUPTUPLE : INT

NULLRATE : DOUBLE

ESTATE : ENUM(Y/N)

ESCORE : INT

PASS : ENUM (P/N)

- 보건소 태스크 데이터 테이블(TDT\_H) \*H 에는 식당 태스크를 제출한

제출자의 ID 가 들어갑니다

PNUM: VARCHAR

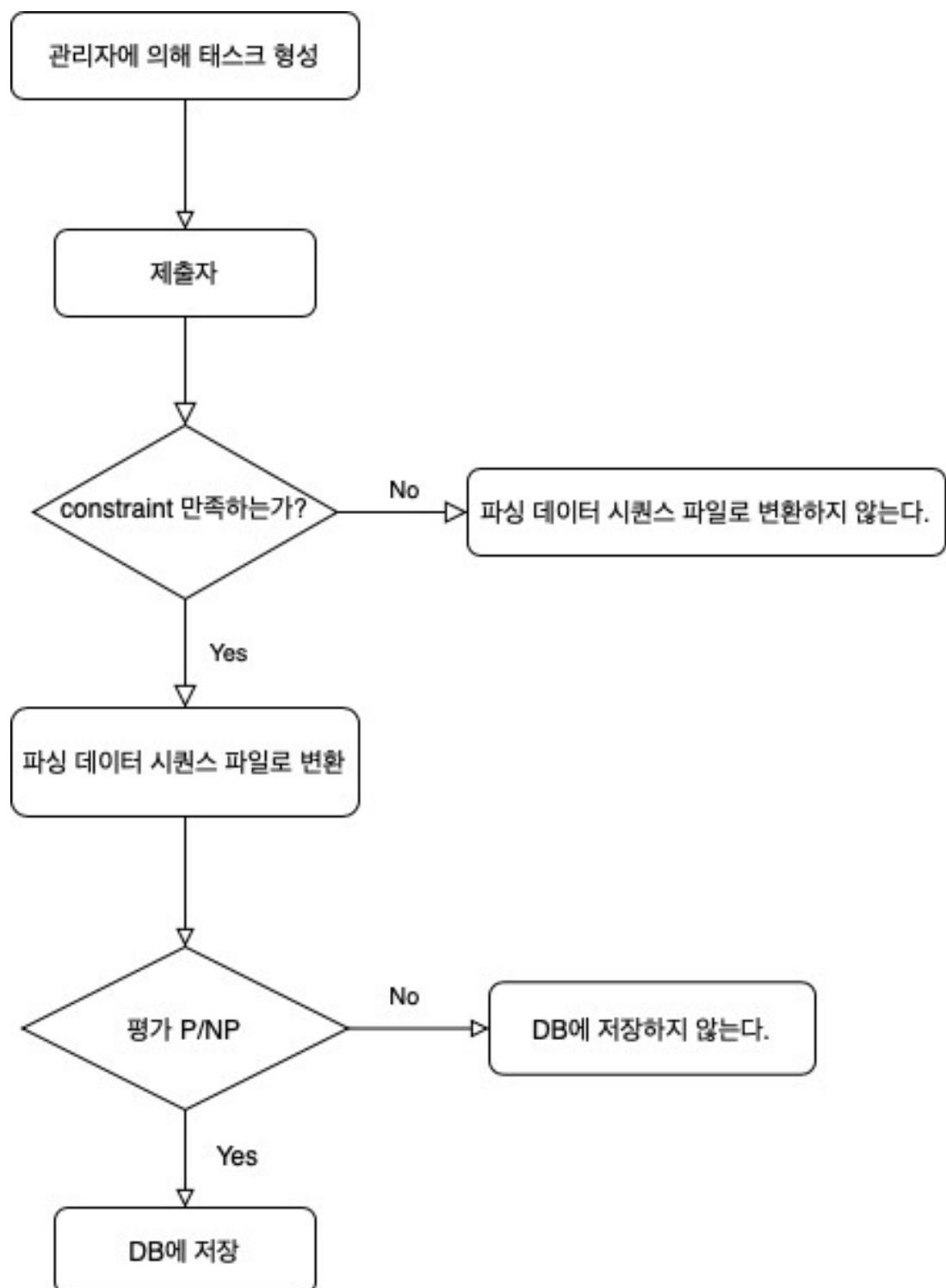
ADDRESS: ENUM('강남', '서대문', '서초', '마포', '용산')

VDATE: DATE

PN: ENUM(P/N)

**SID: VARCHAR(SUBMITTER'S ID)**

## II. 시스템 설계 문서 및 구현 내용



### III. 프로젝트 구현 진행 과정 및 역할 분담

월	화	수	목	금	토	일
16	17	18	19	20	21	22
Node.js 와 Mysql 공부					기능 구현	
23	24	25	26	27	28	29
기능 구현					기능 합치기 + 완성본 테스트	
30	1	2	3	4	5	
기능 합치기 + 완성본 테스트			최종 보고서 작성			

11 월 16 일 - 11 월 20 일

- Node.js 와 mysql 공부
  - 본격적으로 프로젝트를 시작하기 전, 주로 사용할 node.js 와 mysql 을 각자 공부하여, 프로젝트 기능 구현시 사용할 수 있도록 공부하는 기간을 가졌다.

11 월 21 일 - 11 월 26 일

- 기능 구현
  - 소주제별로 구현해야 할 기능을 나누고, 이후에 합치기로 11 월 20 일 회의에서 결정
- 역할 분담
  - 1) 소주제 1
    - 제출자 기능
    - 태스크 참여신청, 원본데이터 시퀀스 파일 제출 : 박시원
    - 제출 현황 모니터링 및 평가 점수 확인 : 김채현
    - 평가자 기능 : 한승희
  - 2) 소주제 2
    - 관리자 기능
    - 태스크 생성, 태스크 관리 : 이해지
    - 태스크 통계, 회원가입 및 통계 : 최민수
    - 회원가입 및 인증 기능 : 차승연

11 월 27 일 - 12 월 2 일

- 기능 합치기
  - 각자 구현한 기능을 하나의 파일로 합치는 작업
- 완성본 테스트
  - 합쳐진 하나의 파일로 테스트 해본 후, 프로젝트 개발 명세에 기재된 서버환경인 CentOS 7에서 테스트 진행

12 월 3 일 - 12 월 5 일

- 중간 보고서에서 변경된 사항들을 반영하여, 최종 보고서 작성

## IV. 프로젝트 설명

### 1. 기능별 구현 내용

#### 1) 소주제 1 – 제출자 기능

##### (1) 태스크 참여 신청

###### (i) 참여 가능한 태스크 목록 제공

- 제출자 UI 화면에서 ‘태스크 참여 신청’을 누르면, 시스템은 해당 제출자가 참여할 수 있는 태스크의 목록을 보여준다. 이때 ‘참여할 수 있는 태스크’란, 이미 태스크 참여신청을 하여 관리자의 승인을 기다리는 태스크를 제외하고, 자신의 role(식당 or 보건소)과 같은 role 을 가진 태스크이면서 현재 참가하고 있지 않은 태스크를 의미한다. 따라서 참여신청정보를 저장하는 DB 테이블인 REQUEST 에 중복된 tuple 이 들어가거나, 태스크와 제출자의 role 이 서로 다른 참여신청이 만들어지는 일은 사전에 차단된다.

- ‘참여할 수 있는 태스크’의 목록은 다음과 같은 질의를 통해 그 정보를 가져온다.

```
SELECT TASK.NAME FROM TASK WHERE  
TASK.NAME NOT IN (SELECT REQUEST.TASKNAME  
FROM REQUEST  
WHERE REQUEST.SUBMITTERID = '${submit_id}')  
AND TASK.NAME NOT IN (SELECT TASK.NAME  
FROM SUBMITTER, TASK  
WHERE SUBMITTER.ID= '${submit_id}' AND SUBMITTER.ROLE != TASK.ROLE)  
AND TASK.NAME NOT IN (SELECT APPROVAL.TASKNAME  
FROM APPROVAL  
WHERE APPROVAL.SUBMITTERID = '${submit_id}');
```

\* \${submit\_id}는 현재 로그인한 제출자의 id이다.)

- 참여 가능한 태스크의 목록은 참여가능한 태스크들의 이름을 보여주며, 각 이름 옆에는 ‘참여하기’ 버튼이 있다. 이 버튼을 누르면 해당 태스크에 참여할 수 있는 URL로 넘어가게 된다.

예시)

- 제출자 seawoon7은 식당용 태스크에 참여하는 제출자이다. 그리고 시스템에는 현재 4 개의 태스크가 생성되어 있다. 각각 태스크의 이름은 '맨도롱식당 태스크', '삼미식당 태스크', '밤치킨식당 태스크', '보건소 태스크'이다. 이 중 '보건소 태스크'만 보건소용 태스크로, seawoon7은 이 태스크에 참여할 수 없다.

corona14.task: 4 행 (총) (대략적)					
NAME	DESCRIPTION	TASKDATATABLE	CYCLE	ROLE	
맨도롱식당 태스크	맨도롱식당의 태스크입니다.	TDT_jdpark	24:00:00	R	
밤치킨식당 태스크	밤치킨식당의 태스크입니다.	TDT_nightchicken	24:00:00	R	
보건소 태스크	보건소의 태스크입니다.	TDT_hospital1004	24:00:00	H	
삼미식당 태스크	삼미식당의 태스크입니다.	TDT_sk0101	24:00:00	R	

(그림 1: 현재 생성된 태스크이다. TASK 테이블에 저장된다.)

- seawoon7은 현재 존재하는 태스크 중 '맨도롱식당 태스크'에 참여 중이며, '삼미식당 태스크'에 참여 신청을 보낸 상태이다. 그리고 그 참여 신청은 관리자가 아직 승인 또는 거절하지 않은 상태이다.

corona14.request: 5 행 (총) (대략적)				corona14.approval: 5 행 (총) (대략적)			
TASKNAME	SUBMITTERID	ROLE		TASKNAME	SUBMITTERID	ROLE	
밤치킨식당 태스크	waitchang01	R		맨도롱식당 태스크	jdpark	R	
밤치킨식당 태스크	waitshee02	R		맨도롱식당 태스크	seawoon7	R	
삼미식당 태스크	jdpark	R		밤치킨식당 태스크	nightchicken	R	
삼미식당 태스크	seawoon7	R		보건소 태스크	hospital1004	H	
삼미식당 태스크	waitchang01	R		삼미식당 태스크	sk0101	R	

(그림 2,3 : 태스크 참여 신청 목록(왼쪽), 태스크 참여 승인 목록(오른쪽)이다.

왼쪽에 '삼미식당 태스크'에 seawoon7이 참여신청한 기록이, 오른쪽에 '맨도롱식당 태스크'에 seawoon이 참여 승인된 기록이 보인다.)

- 따라서, 시스템에 의해 제출자 seawoon7이 참여할 수 있는 태스크는 '밤치킨식당 태스크' 뿐이므로 참여 가능한 신청 목록은 다음과 같이 뜬다.

The screenshot shows a web browser window with the URL `localhost:8080/submitter/notparticipate`. The page title is "참여 가능한 태스크 리스트". Below the title is a table with one row, showing a task numbered 1 with the name "밤치킨식당 태스크" and a button labeled "참여하기". At the bottom left of the page is a link "홈으로 돌아가기". The browser's address bar also lists other sites like YouTube, Google, NAVER, and Baekjoon Online J....

번호	태스크 이름	태스크 참여
1	밤치킨식당 태스크	<a href="#">참여하기</a>

[홈으로 돌아가기](#)

(그림 4 : 제출자 seawoon7 의 참여 가능한 태스크 리스트이다.)

- 그리고 그림에 보이는 '참여하기' 버튼을 누르면 '밤치킨식당 태스크'에 참여하기 위해 개인정보 이용동의서에 동의할 수 있는 페이지로 넘어간다.

## ii ) 태스크 참여 신청

- ‘참여하기’ 버튼을 누르면 개인정보 이용동의서에 동의하는 절차가 진행된다. 제출자는 동의서를 읽고 ‘동의’ 또는 ‘비동의’ 중 하나를 선택한 후 버튼을 눌러 참여를 신청할 수 있다. ‘동의’를 선택한 뒤 버튼을 눌러 참여를 신청하면, 시스템은 해당 태스크와 제출자에 대한 참여 신청 정보를 DB 테이블인 REQUEST 에 새 튜플로 저장하고, 신청이 완료되었으며 관리자의 승인을 기다려야함을 알리는 URL 로 넘어간다. 만약 ‘비동의’를 누른 뒤 버튼을 눌러 참여를 신청하면, 시스템은 이를 감지하여 참여 신청을 반영하지 않고 해당 페이지에서 ‘동의를 해야만 참여를 신청할 수 있다’는 주의 문구를 띄운다.

예시)

- 제출자 seawoon7 이 ‘밤치킨식당 태스크’에 참여하기 위해 개인정보 이용동의서에 동의할 수 있는 페이지로 넘어왔을 때의 화면이다.

localhost:8080/submitter/privacy\_consent

YouTube Google NAVER Baekjoon Online J... YSCEC: 사이트에... 연세대학교 학술정... https://portal.yons... https://codeforces.... gen.lib.rus

## 밤치킨식당 태스크 개인정보동의서

코로나-19 확산과 관련한 연구자료를 수집하는 과정에서 다음과 같이 개인정보를 수집, 이용하는 것에 동의합니다.

- 목적 : 코로나-19 확산 동향 파악 및 확산 요인에 대한 연구를 위한 자료 수집
- - 항목 : 제출자ID, 전화번호, 주소, 방문날짜, 방문시각(식당에 한해), 코로나검사음성여부(보건소에 한해)
- - 폐기 : 회원의 탈퇴와 동시에 즉시 폐기

하나를 선택해주십시오.

개인정보동의서에 동의하시겠습니까? 동의 비동의

**태스크 참여**

- [홈으로 돌아가기](#)
- [태스크 참여 신청하기](#)

(그림 5 : 개인정보 이용동의서를 보여주는 페이지이다.)

- 이때 ‘비동의’를 누른 뒤 ‘태스크 참여’ 버튼을 누르면 아래와 같은 문구가 뜬다.



## 밤치킨식당 태스크 개인정보동의서

코로나-19 확산과 관련한 연구자료를 수집하는 과정에서 다음과 같이 개인정보를 수집, 이용하는 것에 동의합니다.

- 목적 : 코로나-19 확산 동향 파악 및 확산 요인에 대한 연구를 위한 자료 수집
- 항목 : 제출자ID, 전화번호, 주소, 방문날짜, 방문시각(식당에 한해), 코로나검사음성여부(보건소에 한해)
- 폐기 : 회원의 탈퇴와 동시에 즉시 폐기

개인정보 동의서에 동의하지 않으면 태스크를 참여하실 수 없습니다.

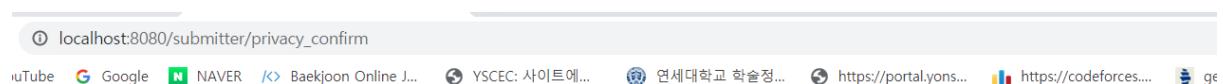
개인정보동의서에 동의하시겠습니까? 동의 비동의

[태스크 참여](#)

- [홈으로 돌아가기](#)
- [태스크 참여 신청하기](#)

(그림 6 : ‘개인정보 동의서에 동의하지 않으면 태스크에 참여하실 수 없습니다.’라는 문구가 보인다.)

- 그리고 ‘동의’를 누른 뒤 ‘태스크 참여’ 버튼을 누르면 다음과 같은 페이지로 넘어간다.



## 신청이 완료되었습니다. 관리자의 승인을 기다려주십시오.

- [홈으로 돌아가기](#)
- [태스크 참여 신청하기](#)

(그림 7 : 신청이 완료되었음을 알리는 페이지로 이동했다.)

## (2) 원본 데이터 시퀀스 파일 제출

### ( i ) 참여 중인 태스크 목록 제공

- 제출자 UI 홈에서 ‘태스크 참여 목록’을 누르면, 시스템은 현재 제출자가 관리자의 승인을 받아 데이터 제출이 가능한 태스크의 목록을 보여준다. 관리자의 승인을 받아 참여중인 태스크의 정보는 DB 테이블인 APPROVAL에 저장되므로, 해당 태스크들의 정보는 다음과 같은 질의를 통해 DB에서 가져온다.

```
SELECT TASK.NAME FROM TASK WHERE TASK.NAME
IN (SELECT APPROVAL.TASKNAME FROM APPROVAL WHERE APPROVAL.SUBMITTERID =
'${submit_id}');
```

\* \${submit\_id}는 현재 로그인한 제출자의 id이다.)

- 제출이 가능한 태스크의 목록은 제출가능한 태스크들의 이름을 보여주며, 각 이름 옆에는 '제출하기' 버튼이 있다. 이 버튼을 누르면 해당 태스크에 원본데이터시퀀스 파일을 제출할 수 있는 URL로 넘어가게 된다.

예시)

- 제출자 seawoon7이 '태스크 참여 목록'을 눌렀을 때 seawoon7의 태스크 참여 목록이다. 비록 '삼미식당 태스크'와 '밤치킨식당 태스크'에 참여 신청을 보냈지만, 아직 관리자가 승인하지 않았기 때문에 seawoon7의 태스크 참여 목록에는 '맨도롱식당 태스크'만 목록에 뜬다.

The screenshot shows a web browser window with the URL `localhost:8080/submitter/participate`. The page title is "태스크 참여 목록". A table lists one task:

번호	태스크 이름	태스크 제출하기
1	만도롱식당 태스크	<a href="#">제출하기</a>

Below the table is a blue link labeled "홈으로 돌아가기".

(그림 8 : 제출자 seawoon7의 태스크 참여 목록이다.)

이제 그림에 보이는 '제출하기' 버튼을 누르면 '만도롱식당 태스크'에 데이터를 제출하기 위한 페이지로 넘어가게 된다.

### ( ii ) 데이터 제출하기

- '제출하기' 버튼을 누르면 원본데이터시퀀스파일 및 관련한 정보를 제출할 수 있는 페이지로 넘어온다. 제출자가 원본데이터시퀀스파일을 제출하기 위해서는 CSV 파일, 원본 데이터 타입, 회차, 데이터 수집 기간을 빠짐없이 정확하게 입력해야 한다. 만약 이들 정보 중 하나라도 누락되거나 잘못된 정보를 입력하면 시스템은 이를 감지하고 제출을 거부하며 주의 문구를 띠운다. 이때 업로드한 파일도 자동 삭제한다.
- 다음은 시스템이 감지하고 제출을 거부하는 항목들이다. 이들 항목들 중 일부는 CSV 파일을 파싱하거나 파싱 정보를 저장하는 쿼리를 저장할 때 치명적인 오류를 일으킬 수 있으므로 시스템은 이를 사전에 감지한다. 그리고 그 외 제출자가 실수할 수 있는 부분도 시스템이 감지하고 파싱하기 전에 제출자에게 고지한다.

- CSV 파일, 원본데이터타입, 회차, 데이터 수집기간 중 하나라도 입력하지 않았다.
- 입력한 CSV 파일의 확장자가 '.csv'가 아니다.
- 입력한 원본데이터 타입 형식이 잘못되었다. (원본데이터 타입은 숫자이다.)
- 입력한 원본데이터 타입이 존재하지 않는다.
- 입력한 회차 형식이 잘못되었다. (회차는 숫자이다.)
- 입력한 데이터 수집기간의 시작 날짜가 끝난 날짜보다 늦다.
- 입력한 데이터 수집기간의 끝난 날짜가 오늘보다 늦다.

- 이 모든 사항을 지켰다면 시스템은 제출받은 원본데이터시퀀스파일에 대한 파싱을 시작한다.

예시)

- 제출자 seawoon7 은 '맨도롱식당 태스크'에 데이터를 제출하려 한다. 다음과 같은 화면에서 필요한 정보를 입력하고 '제출하기' 버튼을 누르면 원본데이터가 제출된다.

만도롱식당 태스크 파일 제출

**CSV 파일**

파일 선택 선택된 파일 없음

**원본 데이터 타입**

**회차**

ex) 3회차면 -> 3

**데이터 수집 기간** 연도-월-일 ~ 연도-월-일

**제출하기**

• 홈으로 돌아가기  
 • 태스크 참여 목록 보기

(그림 9 : 태스크 파일을 제출하는 페이지이다.)

- 하지만 seawoon7 의 실수로 데이터 수집이 끝난 기간을 입력하지 않은 채로 '제출하기' 버튼을 눌러 다음 화면이 나타났다.

(그림 10 : 수집 종료 기간을 입력하지 않아(왼쪽), 이를 알리는 문구가 나타났다(오른쪽).)

- 이번엔 seawoon7은 모든 정보를 잊지 않고 입력했다. 하지만 모든 정보를 입력하는데 집중한 나머지 seawoon7은 잘못된 확장자의 파일과 잘못된 형식의 회차를 입력한 채로 '제출하기' 버튼을 눌러 다음과 같은 화면이 나타났다.

(그림 11 : jpg 파일을 제출하고, 회차 형식이 틀려(왼쪽), 이를 알리는 문구가 나타났다(오른쪽).)

- 제출자 seawoon7은 2 번이나 제출 정보를 틀리게 입력한 것이 분하여, 이번에는 온 신경을 집중하여 모든 정보를 정확하게 입력하는데 성공한다. 이번에는 시스템의 요구사항을 충족하였기에 제출자 seawoon7이 '제출하기' 버튼을 누르자 시스템이 파싱을 시작한다.

### ( iii ) 원본데이터시퀀스파일 파싱 및 정성 평가

- 시스템은 원본데이터시퀀스파일을 파싱하기 위해 DB 테이블인 TYPEORIGIN에서 원본데이터타입에 해당하는 데이터타입 스키마와 매핑정보를 받아온다. TYPEORIGIN이 저장하고 있는 매핑정보는 파싱데이터시퀀스파일에 저장하는 정보, 즉 '전화번호', '주소', '방문날짜', '기타'(식당용 태스크는 '방문시각', 보건소용 태스크는 '코로나검사음성여부')가 해당 원본데이터 타입에서 '몇 번째 열'에 저장되어 있는가다.

- 시스템은 이를 이용해 각 행마다 파싱데이터시퀀스파일에 저장하는 정보들을 찾고, 그 정보들이 각각 누락되어 있거나 형식이 잘못되어 있는지 평가한다. 만약 그렇다면 시스템은 그 정보를 NULL로 취급한다. 그리고 그 행에서 NULL인 정보가 하나라도 나온다면, 파싱데이터시퀀스파일에 그 행에 나온 정보들을 저장하지 않는다. 만약 그러한 정보가 없다면, 시스템은 그 행에서 얻은 튜플을 파싱데이터시퀀스 파일의 스키마대로 정보를 저장하고, 이를 모든 행에 대해 반복한 후 파싱데이터시퀀스파일을 CSV 파일로 만들어 저장한다.

- 다음은 시스템이 요구하는 각 정보에 대한 올바른 형식이다. 이 '올바른 형식'은 mysql에서 요구하는 데이터형과 동일하고, 나중에 평가 후 튜플들을 태스크 데이터 테이블에 저장할 때 문제가 생기지 않도록 하기 위해 시스템이 요구하는 형식이다.

- 전화번호 : 모두 숫자로 이루어진 11자리 문자열이다
- 주소 : '강남', '마포', '서초', '서대문', '용산' 중 하나이다.
- 방문날짜 : YYYY-MM-DD 형식이면서 1000년 이후의 실존하는 날짜이다. (윤년이 아닌 해의 2월 29일은 검사하지 않는다.)
- 방문시간 : HH:MM:TT 형식이면서 00:00:00 ~ 23:59:59 사이의 값이다.
- 코로나검사음성여부 : 'P', 'N' 중 하나이다.

- 이후 정성평가가 이루어진다. 전체 튜플 수는 파싱데이터시퀀스파일에 최종적으로 저장된 튜플의 수이다. 원본데이터에서 NULL이 나온 행은 파싱데이터시퀀스 파일에 저장하지 않으므로, 전체 튜플 수는 원본데이터가 가진 튜플 수보다 작거나 같다.

- 중복 튜플 수는 원본데이터에서 중복된 행의 수이다. '중복된 행'은 나중에 태스크 데이터 테이블에 저장될 것을 고려하여, 태스크 데이터 테이블의 PK인 '전화번호'가 같은 행을 의미한다.

- 따라서 전화번호가 동일한 행이 앞서 있었다면, 그것은 중복된 행이고 시스템은 이를 파싱데이터시퀀스 파일에 저장하지 않는다.
  - NULL 비율은 (시스템이 NULL로 취급한 정보) / (전체 평가한 정보)의 백분율이다. 예를 들어, 원본데이터에서 '주소'에 해당하는 열만 전부 누락되었다면, NULL 비율은 25%인 것이다.
- 정성평가까지 끝난 파싱데이터시퀀스 파일은 그 태스크의 역할에 따라 다른 폴더에 저장한다. 그리고 파싱데이터시퀀스 파일에 대한 정보도 그 태스크 역할에 따라 다른 DB 테이블에 저장하는데, 식당용 태스크의 경우는 RPARSING, 보건소 태스크의 경우는 HPARSING 테이블에 저장한다. 그리고 이때 평가자는 앞으로 평가해야 할 시퀀스 파일의 수와 상관없이 완전 랜덤 배정한다. 그리고 이때 받은 파싱데이터시퀀스 파일의 PK(ID)가 파싱데이터시퀀스 파일의 파일이름이 된다. 그리고 해당 파일은 식당용 태스크인지, 보건소용 태스크인지에 따라 '../data/parsed\_csv\_restaurant' 또는 '../data/parsed\_csv\_hospital'에 저장된다.

예시)

- 제출자 seawoon7이 다음과 같은 정보와 원본데이터 시퀀스 파일(csv)을 제출했다. 이때 제출한 csv 파일은 첫 줄을 제외하고 15 개의 행으로 이루어져 있고, 잘못 입력하거나 중복되는 행이 없다.

```
 typeId_1_20200319_20200324 - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
방문날짜,방문시각,전화번호,이름,주소,나이,성별
2020-03-19,12:30:00,00011110001,임서연,강남,3,F
2020-03-19,12:31:00,00011110002,최하윤,강남,27,F
2020-03-19,13:05:00,00011110003,임도율,강남,4,M
2020-03-19,13:05:00,00011110004,임재윤,강남,25,M
2020-03-20,12:27:00,00055550001,양민규,용산,22,M
2020-03-20,12:29:00,00092100108,양민수,용산,58,M
2020-03-20,13:11:00,00055550003,이시은,용산,28,F
2020-03-21,13:12:00,00055550004,김강민,용산,1,M
2020-03-21,11:55:00,00055550005,김민수,용산,31,M
2020-03-21,11:59:00,00033330002,윤시현,서대문,38,F
2020-03-21,12:40:00,00033330003,박건우,서대문,2,M
2020-03-21,12:50:00,00033330004,박민우,서대문,6,M
2020-03-22,12:31:00,00033330005,박종수,서대문,37,M
2020-03-23,12:33:00,00033330006,홍예지,서대문,48,F
2020-03-23,12:33:00,00033330007,홍예은,서대문,48,F
```

**マンド롱식당 태스크 파일 제출**

CSV 파일  
 typeId\_1\_20200319\_20200324.csv

원본 데이터 타입

회차

데이터 수집 기간  ~

• [홈으로 돌아가기](#)  
• [태스크 참여 목록 보기](#)

(그림 12 : csv 파일(왼쪽)과 입력한 제출 정보(오른쪽)이다.)

- 파싱 결과 생성된 파싱데이터시퀀스파일은 다음과 같으며, 생성된 파싱데이터시퀀스 파일 튜플은 다음과 같다.

iAIN > data > parsed_csv_restaurant		▲	▼	⟳	🔍	☰					
이름		수정한 날짜									
13		2020-11-30 오후 2:25									
<b>13 - Windows 메모장</b>											
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)											
00011110001,강남,2020-03-19,12:30:00 00011110002,강남,2020-03-19,12:31:00 00011110003,강남,2020-03-19,13:05:00 00011110004,강남,2020-03-19,13:05:00 00055550001,용산,2020-03-20,12:27:00 00092100108,용산,2020-03-20,12:29:00 00055550003,용산,2020-03-20,13:11:00 00055550004,용산,2020-03-21,13:12:00 00055550005,용산,2020-03-21,11:55:00 00033330002,서대문,2020-03-21,11:59:00 00033330003,서대문,2020-03-21,12:40:00 00033330004,서대문,2020-03-21,12:50:00 00033330005,서대문,2020-03-22,12:31:00 00033330006,서대문,2020-03-23,12:33:00 00033330007,서대문,2020-03-23,12:33:00											

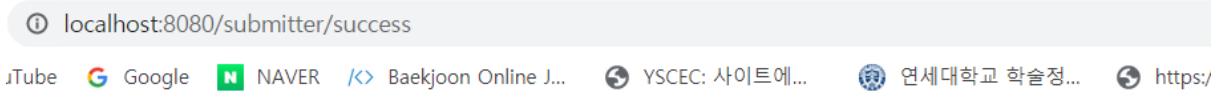
(그림 13 : 파싱된 결과가 저장된 파싱데이터시퀀스 파일 '13.csv'이다. 튜플이 15 개이다.)

corona14.rparsing: 13 행 (총) (대략적)												▶ 다음	
PARSINGID	TASKNAME	SID	TID	ROUND	TERMSTART	TERMEND	EID	TOTALTUPLE	DUPTUPLE	NULLRATE	ESTATE	ESCORE	PASS
1	멘도通报당 태스크	jdpark	1	1	2020-11-16	2020-11-16	evalchoi1	35	0	0 Y	10	P	
2	삼미식당 태스크	sk0101	1	1	2020-11-16	2020-11-16	evalhan42	40	0	5 Y	8	P	
3	방치킨식당 태스크	nightchicken	1	1	2020-11-16	2020-11-16	evalcha03	35	0	0 Y	10	P	
4	삼미식당 태스크	sk0101	2	2	2020-11-17	2020-11-17	evalhan42	30	0	0 Y	10	P	
5	멘도通报당 태스크	jdpark	1	2	2020-11-17	2020-11-18	evalchoi1	20	0	30 N	0	N	
6	삼미식당 태스크	sk0101	2	3	2020-11-18	2020-11-18	evalcha03	25	0	5 Y	10	P	
7	방치킨식당 태스크	nightchicken	2	2	2020-11-17	2020-11-18	evalhan42	25	0	30 Y	0	N	
8	멘도通报당 태스크	jdpark	1	3	2020-03-19	2020-03-25	evalchoi1	14	0	0 N	0	N	
9	멘도通报당 태스크	jdpark	1	4	2020-03-19	2020-03-24	evalcha03	14	0	0 N	0	N	
10	멘도通报당 태스크	jdpark	1	5	2020-03-19	2020-03-23	evalcha03	14	1	0 N	0	N	
11	멘도通报당 태스크	jdpark	1	6	2020-03-19	2020-03-25	evalchoi1	15	0	0 N	0	N	
12	멘도通报당 태스크	jdpark	1	7	2020-03-19	2020-03-25	evalpark07	14	0	3.3 N	0	N	
13	멘도通报당 태스크	seawoon7	1	1	2020-03-19	2020-03-24	evalpark07	15	0	0 N	0	N	

(그림 14 : 식당용 태스크에 제출된 원본데이터의 파싱데이터시퀀스 파일 정보를 저장하는 DB 테이블이다.

13 번 tuple 이 새로 생겼음을 볼 수 있다.)

- 제출이 완료되어 다음과 같은 페이지로 이동하였다.



## 제출이 완료되었습니다. 감사합니다.

- [홈으로 돌아가기](#)
- [태스크 참여 목록 보기](#)

(그림 15 : 제출이 완료되었음을 알려주는 페이지다.)

- 이번엔 제출자 seawoon7 가 꼼꼼하지 않게 작성한 다음의 원본데이터 시퀀스 파일(csv)을 제출했다. 마찬가지로 이때 제출한 csv 파일은 첫 줄을 제외하고 15 개의 행으로 이루어져 있지만, 중복된 행이 1 개, '전화번호'가 잘못된 행이 1 개, '방문날짜'가 잘못된 행이 1 개, '주소'가 잘못된 행이 1 개, '방문시각'이 잘못된 행이 1 개 있다.

typeid\_1\_20200325\_20200328 - Windows 메모장  
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)  
방문날짜,방문시각,전화번호,이름,주소,나이,성별  
2020-03-25,12:30:00,01011110001,임서연,강남,3,F  
2020-03-25,12:31:00,01011110002,최하윤,강남,27,F  
2020-03-25,13:05:00,01011110003,임도율,강남,4,M  
2020-03-25,13:05:00,01011110004,임재윤,전주,25,M  
2020-03-26,12:27:00,01055550001,양민규,용산,22,M  
2020-03-26,12:29:00,01092100108,양민수,용산,58,M  
2020-03-26,13:11:00,01055550003,이시은,용산,28,F  
2020-03-27,13:12:00,01055550004,김강민,용산,1,M  
2020-03-27,11:55:00,01055550005,김민수,용산,31,M  
2020-03-27,11:59:00,01033330002,윤시현,서대문,38,F  
2020-03-27,12:00,01033330003,박건우,서대문,2,M  
2020-03-28,12:50:00,0103330004,박민우,서대문,6,M  
2020-03-28,12:31:00,01033330005,박종수,서대문,37,M  
2020-03-28,12:33:00,01033330006,홍예지,서대문,48,F  
2020-03-28,12:33:00,01033330006,홍예지,서대문,48,F

## 맨도롱식당 태스크 파일 제출

### CSV 파일

typeid\_1\_20200325\_20200328.csv

### 원본 데이터 타입

1

### 회차

2

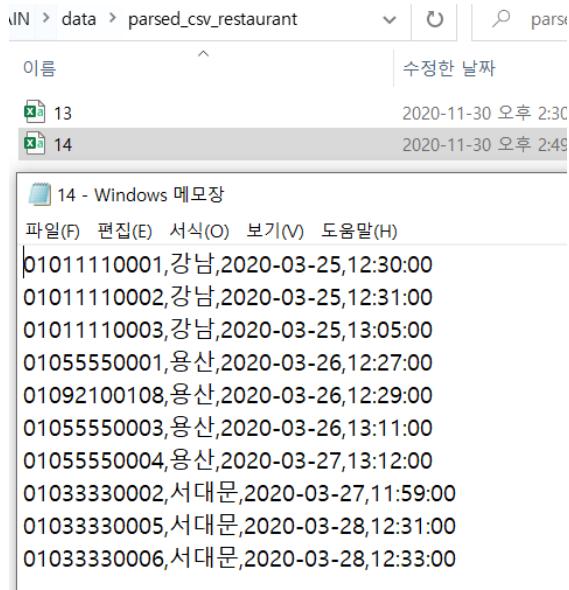
데이터 수집 기간  ~

- [홈으로 돌아가기](#)
- [태스크 참여 목록 보기](#)

(그림 16 : csv 파일(왼쪽)과 입력한 제출 정보(오른쪽)이다.

csv 파일에 잘못된 행이 총 5 개 있다.)

- 파싱 결과 생성된 파싱데이터시퀀스파일은 다음과 같으며, 생성된 파싱데이터시퀀스 파일 튜플은 다음과 같다.



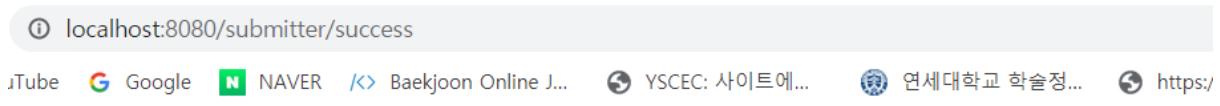
(그림 17 : 파싱된 결과가 저장된 파싱데이터시퀀스 파일 '14.csv'이다. 튜플이 10 개이다.)

PARSINGID	TASKNAME	SID	TID	ROUND	TERMSTART	TERMEND	EID	TOTALTUPLE	DUPLICATE	NULLRATE	ESTATE	ESCORE	PASS
1	멘도辱식당 테스크	jdpark	1	1	2020-11-16	2020-11-16	evalchoi1	35	0	0 Y	10 P		
2	삼미식당 테스크	sk0101	1	1	2020-11-16	2020-11-16	evalhan42	40	0	5 Y	8 P		
3	밥지킨식당 테스크	nightchicken	1	1	2020-11-16	2020-11-16	evalcha03	35	0	0 Y	10 P		
4	삼미식당 테스크	sk0101	2	2	2020-11-17	2020-11-17	evalhan42	30	0	0 Y	10 P		
5	멘도辱식당 테스크	jdpark	1	2	2020-11-17	2020-11-18	evalchoi1	20	0	30 N	0 N		
6	삼미식당 테스크	sk0101	2	3	2020-11-18	2020-11-18	evalcha03	25	0	5 Y	10 P		
7	밥지킨식당 테스크	nightchicken	2	2	2020-11-17	2020-11-18	evalhan42	25	0	30 Y	0 N		
8	멘도辱식당 테스크	jdpark	1	3	2020-03-19	2020-03-25	evalchoi1	14	0	0 N	0 N		
9	멘도辱식당 테스크	jdpark	1	4	2020-03-19	2020-03-24	evalcha03	14	0	0 N	0 N		
10	멘도辱식당 테스크	jdpark	1	5	2020-03-19	2020-03-23	evalcha03	14	1	0 N	0 N		
11	멘도辱식당 테스크	jdpark	1	6	2020-03-19	2020-03-25	evalchoi1	15	0	0 N	0 N		
12	멘도辱식당 테스크	jdpark	1	7	2020-03-19	2020-03-25	evalpark07	14	0	3.3 N	0 N		
13	멘도辱식당 테스크	seawoon7	1	1	2020-03-19	2020-03-24	evalpark07	15	0	0 N	0 N		
14	멘도辱식당 테스크	seawoon7	1	2	2020-03-25	2020-03-28	evalpark07	10	1	6.7 N	0 N		

(그림 18 : 식당용 테스크에 제출된 원본데이터의 파싱데이터시퀀스 파일 정보를 저장하는 DB 테이블이다.

14 번 tuple 이 새로 생겼고, 중복튜플이 1 개, null rate = 4/60 = 6.7% 으로 기록되었다.)

- 제출이 완료되어 다음과 같은 페이지로 이동하였다.



- [홈으로 돌아가기](#)
- [태스크 참여 목록 보기](#)

(그림 19 : 제출이 완료되었음을 알려주는 페이지다.)

### (3) 제출 현황 모니터링 및 평가 점수 확인

#### • 제출 현황 모니터링 방법



- 제출자의 내 정보로 들어가, 제출 현황 확인하기를 누른다.

<누른 후>

- 오른쪽의 화면에서 자신이 확인하고 싶은 정보를 클릭한다. 예시는 제출자가 식당의 태스크를 제출했을 경우로 하였다. 보건소의 태스크의 경우도 동일하게 동작한다.

[홈으로 돌아가기](#)

[나의 제출 현황 확인하기](#)

- [제출한 파일 수](#)
- [통과된 데이터 수](#)
- [제출 파일 현황](#)

## 1) 자신이 참여중인 태스크의 현황

제출한 파일들의 현황

태스크 이름	회차	제출자 ID	파일 ID	평가 여부	평가 점수	파일의 폐스 여부	원본 데이터 타입
밤치킨식당 태스크	1	dd	12	N	0	N	1
밤치킨식당 태스크	2	dd	13	N	0	N	1
삼미식당 태스크	13	dd	27	N	0	N	1
삼미식당 태스크	16	dd	30	Y	10	P	1
태스크 이름	회차	제출자 ID	파일 ID	평가 여부	평가 점수	파일의 폐스 여부	원본 데이터 타입
멘도롱식당 태스크	1	dd	11	N	0	N	2
삼미식당 태스크	1	dd	14	N	0	N	2
삼미식당 태스크	2	dd	16	N	0	N	2

[홈으로 돌아가기](#)  
[이전 페이지로](#)

(/views/file\_info.ejs , /src/web.js)

- 자신이 참여하고 있는 태스크에 대해 제출한 파일들의 균황을 볼 수 있다.

```
// information about files
router.get("/file_info", function (req, res) {
  const user = req.user;
  const role = user.ROLE;
  const parsingTable = role === "R" ? "rparsing" : "hparsing";
  const userId = [user.ID];
  const sql = `SELECT TASKNAME,PARSINGID,ROUND,ESTATE,ESCORE,PASS,SID,TID FROM ${parsingTable} WHERE SID = ? ORDER BY ROUND ASC`;
  connection.query(sql, userId, function (err, rows) {
    if (err) console.log("query is not executed. select fail...\n" + err);
    else res.render("file_info.ejs", { data: rows });
  });
});
```

- 위의 코드를 보면, 이 제출자의 역할이 식당인지, 보건소인지를 sql 문으로 확인하고, 확인한 결과를 이용하여 참조할 parsing table 을 설정한 다음, 제출자의 id 를 이용하여 제출자가 제출한 파일들의 현황을 불러온다.

## 2) 자신이 제출한 파일 수

제출한 파일 수 확인하기

태스크 이름	제출한 파일 수
멘도롱식당 태스크	1
밤치킨식당 태스크	2
삼미식당 태스크	4

(/views/num\_files.ejs , /src/web.js)

- 자신이 총 몇 개의 파일들을 제출했는지 태스크 별로 출력되도록 하였다.

```
// number of files user submitted
router.get("/num_files", function (req, res) {
  const user = req.user;
  const role = user.ROLE;
  const parsingTable = role === "R" ? "rparsing" : "hparsing";
  const userId = [user.ID];
  const sql = `SELECT count(*) AS count,taskname FROM ${parsingTable} WHERE sid = ? GROUP BY taskname`;
  connection.query(sql, userId, function (err, rows) {
    if (err) console.log("query is not executed. select fail...\n" + err);
    else res.render("num_files.ejs", { data: rows });
  });
});
```

- 위의 코드를 보면, 이 제출자의 역할이 식당인지, 보건소인지를 sql 문으로 확인하고, 확인한 결과를 이용하여 참조할 parsing table 을 설정한 다음, taskname 으로 group 을 지어서 count 로 제출한 총 파일 수를 확인한다.

### 3) Pass 되어 태스크 데이터 테이블에 저장된 tuple 수

#### 통과된 튜플 수 확인하기

태스크 이름	통과된 튜플 수
삼미식당 태스크	14

[홈으로 돌아가기](#)  
[이전 페이지로](#)

(/views/num\_tuples.ejs , /src/web.js)

- 자신이 제출한 파일에 대해서 평가자에 의해 통과된 전체 튜플 수를 태스크 별로 출력한다.

```
// number of tuples in files user submitted
router.get("/num_tuples", function (req, res) {
  const user = req.user;
  const role = user.ROLE;
  const parsingTable = role === "R" ? "rparsing" : "hparsing";
  const userId = [user.ID];
  const sql = `SELECT SUM(totaltuple) as sum,taskname FROM ${parsingTable} WHERE ESTATE = "Y" AND SID = ? GROUP BY TASKNAME`;
  connection.query(sql, userId, function (err, rows) {
    if (err) console.log("query is not executed. select fail...\n" + err);
    else res.render("num_tuples.ejs", { data: rows });
  });
});
```

- 위의 코드를 보면, 이 제출자의 역할이 식당인지, 보건소인지를 sql 문으로 확인하고, 확인한 결과를 이용하여 참조할 parsing table 을 설정한 다음, taskname 으로 group 을 지은 다음, totaltuple 을 sum 을 이용하여 태스크 별로 다 합친다.

## 소주제 1 - 평가자 기능

### 1) 평가 내역 모니터링

The screenshot shows a MySQL Workbench interface. On the left, a code editor window displays the SQL query: `SELECT* FROM rparsing WHERE eid='evaltest';`. To the right is a results grid titled "rparsing (4r x 14c)". The columns are: PARSINGID, TASKNAME, SID, TID, ROUND, TERMSTART, TERMEND, EID, TOTALTUPLE, DUPTUPLE, NULLRATE, and ESTATE. The data in the grid is as follows:

PARSINGID	TASKNAME	SID	TID	ROUND	TERMSTART	TERMEND	EID	TOTALTUPLE	DUPTUPLE	NULLRATE	ESTATE
1	면도를식당 태스크	jdpark	1	1	2020-11-16	2020-11-16	evaltest	35	0	0	N
2	상미식당 태스크	sk0101	1	1	2020-11-16	2020-11-16	evaltest	40	0	5	Y
6	삼미식당 태스크	sk0101	2	3	2020-11-18	2020-11-18	evaltest	25	0	5	Y
7	방치킨식당 태스크	nightchicken	2	2	2020-11-17	2020-11-18	evaltest	25	0	30	N

The sidebar on the right contains a "Filter ..." button and a dropdown menu with items: tdt\_nightchicken 열, SQL 함수, SQL 키워드, and 스니펫.

- 'evaltest'라는 ID를 가진 평가자에게 할당된 제출자의 식당 태스크 예시 목록이다.

PARSINGID 가 2, 7 인 태스크는 ESTATE 가 Y 이므로 이미 평가가 완료되었고, 1, 7 태스크는 아직 평가되지 않았다. 평가된 태스크와 평가되지 않은 태스크를 볼 수 있는 웹페이지를 분리하여 구현하였다. 아래와 같은 sql 쿼리를 통해 평가 완료된 parsing table 을 ejs 파일로 보냈다.

```
var sql = "SELECT * FROM rparsing WHERE estate='Y' AND eid=?";
connection.query(sql, [req.user.ID], function (err, rows, fields) {
  if (err) console.log("query is not executed. select fail...\n" + err);
  else res.render("estimatormain.ejs", { rparsed: rows });
});
```

#### 평가 완료 태스크 목록(식당): 수정할 수 없습니다.

파일 번호	제출자 아이디	전체 티플 수	중복 티플 수	null 컬럼 비율	당신의 평가 점수	승인상태(P/N)
2	sk0101	40	0	5	8	P
6	sk0101	25	0	5	10	P

- 평가 안된 레스토랑 데이터 목록
- 평가된 보건소 데이터 목록
- 홈으로 돌아가기

- /estimatormain 의 페이지 table view 는 위와 같다 고유 파일 번호, 그 파일을 제출한 제출자의 ID, 시스템이 파싱 과정에서 계산한 정량평가 결과, 그리고 평가자 본인의 평가 점수 및 승인 여부를 볼 수 있도록 하였다. Sql 테이블에서 ESTASTE='Y'였던 2, 6 번 파일 정보가 나타남을 확인할 수 있다. 테이블 아래 href 링크를 통해 평가되지 않은 데이터 목록과 평가된 보건소 데이터 목록을 볼 수 있는 페이지로 각각 이동할 수 있도록 하였다. 테이블은 과거 평가 기록이 존재하며, 이를 수정하거나 삭제할 수는 없다.

## 평가 안된 태스크 목록(레스토랑)

파일 번호	제출자 아이디	전체 튜플 수	중복 튜플 수	null 컬럼 비율
1	jdpark	35	0	0
7	nighthicken	25	0	30

- 레스토랑 평가하러 가기
- 평가된 레스토랑 목록
- 보건소 평가정보 보러가기
- 홈으로 돌아가기

- /estimatormain 페이지에서 '평가 안된 레스토랑 데이터 목록' 링크를 클릭했을 때 나타나는 /notestimated 페이지의 view 이다(예시를 위해 만든 데이터이므로 실제 값과 일치하지 않다) ESTATE='N'인 것을 제외하고는 평가된 태스크 목록과 쿼리문은 같다. 아직 평가되지 않았으므로 평가자 본인이 부여한 점수와 데이터 승인 여부는 볼 수 없다. 평가자는 '레스토랑 평가하러 가기'를 통해 테이블에 있는 태스크 위에서부터 하나씩 데이터를 평가할 수 있다.

### 2) 파싱 데이터 시퀀스 파일 평가

- 제출자가 제출한 파일은 올바르게 파싱되어

/project/data/parsed\_csv\_restaurant/<parsingid>.csv 형태로 시스템의 데이터베이스에 저장된다. 예시를 설명하기 쉽도록 현재 'evaltest' 평가자에게 할당된 parsingid=1,2,6,7 인

#### parsed\_csv\_restaurant

이름

1

2

6

7

파일만이 파싱되었다고 가정하자.

그렇다면 parsed\_csv\_restaurant 에 저장된 파싱 데이터 파일은 다음과 같다.

구분하기 쉽도록 제출자가 파일을 제출할 때마다 부여되는 parsingid 를

csv 파일의 이름으로 설정하였다. 평가자의 파일 승인 여부(P, NP)에 따라 이

csv 파일들은 DB 테이블에 저장될 수도, 그러지 않을 수도 있다.

'레스토랑 평가하러 가기' 링크를 누르면 보이는 화면은 다음과 같다.

식당 평가하기 : 한 개 씩 평가할 수 있습니다.

파일 번호	제출자 아이디	전체 튜플 수	중복 튜플 수	null 컬럼 비율
1	jdpark	35	0	0

0점 ▾

non-pass ▾

- csv 파일 열기

평가 제출

- 평가자는 평가 안된 태스크 목록의 위에서부터 한 개 씩 파일을 평가할 수 있다. 이 때 시스템이 자동적으로 계산한 정량 평가 점수를 보여주며, 평가자는 csv 파일을 직접 열어 데이터의 품질을 검사하고 정량평가 점수를 참고하여 0 점부터 10 점까지 자연수로 된 점수를 부여한다. 평가 점수는 전적으로 평가자의 자유에 맡겼다. 그리고 non-pass 와 pass 역시 평가자가 직접 결정하게 된다. 평가자가 non-pass 를 선택하고 평가 제출 버튼을 누르면

csv 파일은 테이블에 저장되지 않고, pass 를 선택하고 평가 제출 버튼을 누르면 csv 파일은 제출자가 지정한 태스크 데이터 테이블에 올바르게 저장된다.

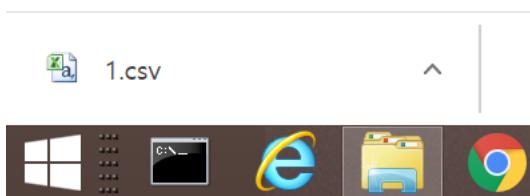
csv 파일을 열기 위한 javascript 코드는 다음과 같다.

```
var file = __dirname + "/../../data/parsed_csv_restaurant/" + pid + ".csv";
try {
  if (fs.existsSync(file)) {
    // 파일이 존재하는지 체크
    var filename = path.basename(file); // 파일 경로에서 파일명(확장자포함)만 추출
    var mimetype = mime.getType(file); // 파일의 타입(형식)을 가져옴

    res.setHeader(
      "Content-disposition",
      "attachment; filename=" + filename
    ); // 다운받아질 파일 명 설정
    res.setHeader("Content-type", mimetype); // 파일 형식 지정

    var filestream = fs.createReadStream(file);
    filestream.pipe(res);
  } else {
    res.send("해당 파일이 없습니다.");
    return;
  }
} catch (e) {
  // 에러 발생시
  console.log(e);
  res.send("파일을 다운로드하는 중에 에러가 발생하였습니다.");
  return;
}
```

- Pid 는 제출 파일의 parsingid 이다. var file 에 다운받고자 하는 파일의 경로와 이름을 올바르게 설정했다. Mime, path, file system 모듈을 사용하여 파일을 다운받도록 하는 파이프를 구축했다. 위의 예시에서 'csv 파일 열기' 링크를 누르면,



위와 같이 다운로드가 잘 되었고, 파일을 다시 열어보았을 때 우리가 원래 원하던 파싱된 csv 파일을 얻을 수 있음을 확인할 수 있다.

	A	B	C	D	E	F	G
1	11110001	강남	#####	12:30:00			
2	11110002	강남	#####	12:31:00			
3	11110003	강남	#####	13:05:00			
4	11110004	강남	#####	13:05:00			
5	55550001	용산	#####	12:27:00			
6	92100108	용산	#####	12:29:00			
7	55550003	용산	#####	13:11:00			
8	55550004	용산	#####	13:12:00			
9	55550005	용산	#####	11:55:00			
10	33330002	서대문	#####	11:59:00			
11	33330003	서대문	#####	12:40:00			
12	33330004	서대문	#####	12:50:00			
13	33330005	서대문	#####	12:31:00			
14	33330006	서대문	#####	12:33:00			
15	11110001	강남	#####	12:30:00			
16	11110002	강남	#####	12:31:00			
17	11110003	강남	#####	13:05:00			
18	11110004	강남	#####	13:05:00			
19	55550001	용산	#####	12:27:00			
20	92100108	용산	#####	12:29:00			

- 이젠 점수를 부여해보자. 웬지 데이터가 그닥 끌리지 않아 1 점에 non-pass 를 부여하고 싶다. 'evaltest' 평가자는 1 번 파일에 1 점과 non-pass 를 부여하였다. 데이터가 pass 되지 않았으므로 csv 파일은 DB table 에 저장되지 않는다. 대신 evaltest 평가자가 부여한 평가 점수와

P/NP 여부는 평가 기록으로 남아 최초의 '평가 안 된 태스크 목록(레스토랑)'에 저장되게 된다.  
이전 페이지로 돌아가면

### 평가 완료 태스크 목록(식당)

파일 번호	제출자 아이디	전체 티틀 수	중복 티틀 수	null 컬럼 비율	당신의 평가 점수	승인상태(P/N)
1	jdpark	35	0	0	1	N
2	sk0101	40	0	5	8	P
6	sk0101	25	0	5	10	P

- 평가 완료 태스크 목록에 우리가 평가한 평가 점수와 승인 상태 여부(1 점, Non-Pass)가 추가된 것을 확인할 수 있다. 평가 완료와 동시에 테이블을 업데이트 했기 때문이다. 이를 위한 코드는 다음과 같다.

```
var sql =  
  "UPDATE rparsing SET estate='Y', escore=?, pass=? WHERE eid=? AND parsingid=?";
```

```
connection.query(  
  sql,  
  [body.score, body.pass, req.user.ID, pid],  
  function (err, rows, fields) {  
    if (err)  
      console.log(  
        | "query is not executed. select fail...\n" + err  
      );  
    else {  
      res.redirect("/notestimated");  
    }  
  }  
);
```

- 평가 완료와 동시에 estate 를 yes 로 바꾸고, 평가자가 부여한 score, pass 여부, 등을 모두 업데이트 했기 때문이다. 평가 완료 태스크 목록은 estate='Y'인 파싱 데이터 테이블에서 데이터를 가져오므로 테이블은 올바르게 업데이트된다.

- 이번엔 평가자가 pass 를 선택하여 csv 파일의 내용이 실제 태스크 데이터 테이블에 저장되는 과정을 확인해보겠다.

### 식당 평가하기

: 한 개씩 평가할 수 있습니다.

파일 번호	제출자 아이디
7	nightchicken

10점 ▾

pass ▾  
• csv 파일 열기

평가 제출

7 번 파일의 데이터가 맘에 들어 10 점과 pass 점수를 부여했다. 평가자가 평가 데이터를 승인했으므로 7.csv 파일에 있는 데이터는 실제로 DB 의 태스크 데이터 테이블에 저장될 것이다. 제출한 파일이 어떤 태스크에 속하느냐에 따라서 7 번 파일의 데이터가 저장될 태스크 데이터 테이블 이름이 변화한다.

7 밤치킨식당 태스크 nightchicken

- 7 번 파일의 경우, nightchicken 제출자는 밤치킨식당 태스크에 참가신청을 하여 파일을 제출했다. '밤치킨식당 태스크'에서 생성된 태스크 데이터 테이블을 찾기 위해 태스크 이름과 태스크 데이터 테이블 이름의 매칭 정보가 있는 task 테이블을 확인한다.

```

1  SELECT* FROM task
2  |

```

task (4r x 5c)				
NAME	DESCRIPTION	TASKDATATABLE	CYCLE	ROLE
맨도롱식당 태스크	맨도롱식당의 태스크입니다.	TDT_jdpark	24:00:00	R
밤치킨식당 태스크	밤치킨식당의 태스크입니다.	TDT_nightchicken	24:00:00	R
보건소 태스크	보건소의 태스크입니다.	TDT_hospital1004	24:00:00	H
삼미식당 태스크	삼미식당의 태스크입니다.	TDT_sk0101	24:00:00	R

- 밤치킨 식당 태스크의 태스크 데이터 테이블 이름은 'TDT\_nightchicken'이다. 이 테이블에 7.csv 파일의 내용을 그대로 집어넣으면 된다. 이를 추론하기 위해서 실행한 쿼리문은 다음과 같다.

```

connection.query(
  "SELECT * FROM task WHERE NAME=?",
  [task], // task = 제출자가 참여한 태스크 이름
  function (err, rows) {
    if (err)
      console.log("query is not executed. select fail...\\n" + err);
    else {
      nameOfTask = rows[0].TASKDATATABLE; // nameOfTask = 태스크 데이터 테이블 이름
    }
  }
)

```

- 이렇게 특정 row 의 nameOfTask 를 불러오고, 만약 평가자가 내린 body.pass 여부가 "P"일 경우,

```

var data = fs.readFileSync(filePath, { encoding: "utf8" });
var rowData = data.split("\n");
var result = [];
for (var i = 0; i < rowData.length; i++) {
  var row = rowData[i].split(",");
  var datas = [];
  for (var j = 0; j < row.length; j++) {
    datas[j] = row[j];
  }
  result.push(datas);
}
for (var rowIndex in rowData) {
  var sql3 ="INSERT INTO " + String(nameOfTask) + " VALUES(?,?,?,?,?)";
  connection.query(sql3,
    [result[rowIndex][0], result[rowIndex][1], result[rowIndex][2], result[rowIndex][3],
     sid,],function (err, rows) {
      if (err) console.log("query is not executed. insert fail...\\n" + err);
    }
  );
}

```

위와 같은 쿼리문이 실행된다.

- data 는 file system 의 readFileSync 모듈을 통해 ./data/parsed\_csv\_restaurant/<parsingid>.csv 내용 전체를 읽어들인 것이고, rowData 는 그것을 행(row)별로, row 는 각각의 rowData 를 열(column) 별로 나눈 array 이다. 각 행과 열의 수에 맞춰 이중 for 문을 돌려 최종 result 에 2

dimension array 형태로 데이터들을 저장한다. 그리고 우리가 원하는 테이블의 특성 개수에 맞춰서 sql3 문을 이용해서 nameOfTask 데이터 테이블에 각 데이터를 저장하게 된다.

	A	B	C	D
1	11110001	강남	#####	12:30:00
2	11110002	강남	#####	12:31:00
3	11110003	강남	#####	13:05:00
4	11110004	강남	#####	13:05:00
5	55550001	용산	#####	12:27:00
6	92100108	용산	#####	12:29:00
7	55550003	용산	#####	13:11:00
8	55550004	용산	#####	13:12:00
9	55550005	용산	#####	11:55:00
10	33330002	서대문	#####	11:59:00
11	33330003	서대문	#####	12:40:00
12	33330004	서대문	#####	12:50:00
13	33330005	서대문	#####	12:31:00
14	33330006	서대문	#####	12:33:00

- Pass로 평가하기 전, 텅 비어있는 '밤치킨식당 태스크'의 태스크 데이터 테이블과 parsingid가 7번인 csv 파일의 데이터 내용이다. Pass로 평가한 뒤 웹에서 확인해 볼 수 있는 평가 안된 태스크 목록은 다음과 같이 비어있다. 이 때 평가하려 가기 링크를 누르면, 평가할 데이터가 없다는 안내문구가 뜨게 된다.

### 평가 안된 태스크 목록(레스토랑)

파일 번호	제출자 아이디	전체 티틀 수	중복 티틀 수	null 컬럼 비율
-------	---------	---------	---------	------------

- 그리고 평가 완료된 태스크 목록은 다음과 같이 바뀐 것으로 확인할 수 있다. 7번 파일의 점수가 10점으로, 승인상태(P/N)이 P로 바뀐 것에 주목한다.

### 평가 완료 태스크 목록(식당): 수정할 수 없습니다.

파일 번호	제출자 아이디	전체 티틀 수	중복 티틀 수	null 컬럼 비율	당신의 평가 점수	승인상태(P/N)
1	jdpark	35	0	0	1	N
2	sk0101	40	0	5	8	P
6	sk0101	25	0	5	10	P
7	nightchicken	25	0	30	10	P

- 그리고 마지막으로, 7번 파일이 참여한 태스크 데이터 테이블인 'tdt\_nightchicken'의 테이블을 확인해보겠다.

1	SELECT* FROM tdt_nightchicken			
<b>tdt_nightchicken (13r x 5c)</b>				
PNUM	ADDRESS	VDATE	VTIME	SID
00011110002	강남	2020-03-19	12:31:00	nightchicken
00011110003	강남	2020-03-19	13:05:00	nightchicken
00011110004	강남	2020-03-19	13:05:00	nightchicken
00033330002	서대문	2020-03-21	11:59:00	nightchicken
00033330003	서대문	2020-03-21	12:40:00	nightchicken
00033330004	서대문	2020-03-21	12:50:00	nightchicken
00033330005	서대문	2020-03-22	12:31:00	nightchicken
00033330006	서대문	2020-03-23	12:33:00	nightchicken
00055550001	용산	2020-03-20	12:27:00	nightchicken
00055550003	용산	2020-03-20	13:11:00	nightchicken
00055550004	용산	2020-03-21	13:12:00	nightchicken
00055550005	용산	2020-03-21	11:55:00	nightchicken
00092100108	용산	2020-03-20	12:29:00	nightchicken

- 7.csv에 쓰여있던 데이터가 잘 들어간 것을 확인할 수 있다. 제출자 ID를 확인할 수 있는 SID(Submitter ID) 필드도 추가했다.

## 2) 소주제 2 관리자 기능

### (1) 태스크 생성



- localhost:8080에서 아이디 admin, 비밀번호 admin으로 로그인하면 이러한 창이 뜨고 '태스크 처리하기'- '태스크 만들기'로 들어간다.

mysql> SELECT* FROM TASK;				
NAME	DESCRIPTION	TASKDATATABLE	CYCLE	ROLE
매도총식당태스크	매도총식당의 태스크입니다.	TDT_jdpark	24:00:00	R
방치킨식당태스크	방치킨식당 태스크입니다.	waitchang01	24:00:00	R
보건소태스크	보건소의 태스크입니다.	TDT_hospital1004	24:00:00	H
삼미식당태스크	삼미식당의 태스크입니다.	TDT_sk0101	24:00:00	R

4 rows in set (0.00 sec)

```

mysql> SELECT * FROM TYPEORIGIN;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | SCHEMA1 | SCHEMA2 | SCHEMA3 | SCHEMA4 | SCHEMA5 | SCHEMA6 | SCHEMA7 | SCHEMA8 | SCHEMA9 | SCHEMA10 | CNUM | CADDRESS | CDATE | CEXTRA | SCHEMANUM |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | VDATE | VTIME | PNUM | NAME | ADDRESS | AGE | GENDER | NLL | NLL | NLL | 2 | 4 | 0 | 1 | 7 | |
| 2 | AGE | GENDER | NAME | PNUM | ADDRESS | DEPENDENT | VDATE | VTIME | NLL | NLL | NLL | 3 | 4 | 6 | 7 | 8 |
| 3 | NAME | AGE | PNUM | ADDRESS | VDATE | VTIME | PN | NLL | NLL | NLL | NLL | 2 | 3 | 4 | 6 | 7 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

- 처음 corona 데이터베이스의 TASK 테이블과 TYPEORIGIN 테이블에는 이러한 정보가 들어있다.
- generatetask.ejs 파일에는 태스크 생성을 위한 UI 가 설정되어있다. 태스크 이름, 테스크데이터테이블 이름, 제출 주기, 제출자 유형, 입력할 스키마 수, 추가할 스키마를 입력하도록 되어있다.

<generatetask.ejs>

```

13      <form method="POST" action="/generatetaskAf">
14          <div class="form-group">
15              <label for="name">태스크 이름</label>
16              <input class="form-control" type="text" name="name" />
17          </div>
18          <div class="form-group">
19              <label for="description">설명</label>
20              <input
21                  class="form-control"
22                  type="text"
23                  name="description"
24                  required
25              />
26          </div>
27          <div class="form-group">
28              <label for="taskdatatable">태스크 데이터 테이블 이름</label>
29              <input
30                  class="form-control"
31                  type="text"
32                  name="taskdatatable"
33                  required
34              />
35          </div>
36          <div class="form-group">
37              <label for="cycle">주기</label>
38              <input
39                  class="form-control"
40                  type="text"
41                  name="cycle"
42                  placeholder="ex) 235959 / 23:59:59"
43                  required
44              />
45          </div>
46          <div class="form-group">
47              <label for="role">제출자 유형</label>
48              <div>
49                  <label for="restaurant" class="radio">
50                      <input type="radio" name="role" value="R" id="restaurant" />
51                      식당
52                  </label>
53                  <label for="hospital" class="radio">
54                      <input type="radio" name="role" value="H" id="hospital" />
55                      보건소
56                  </label>
57              </div>
58          </div>

```

```

68 <div class="form-group">
69   <label for="schema5">스키마5(필수아님, NULL이면 NULL입력)</label>
70   <input class="form-control" type="text" name="schema5" required />
71 </div>
72 <div class="form-group">
73   <label for="schema6">스키마6(필수아님, NULL이면 NULL입력)</label>
74   <input class="form-control" type="text" name="schema6" required />
75 </div>
76 <div class="form-group">
77   <label for="schema7">스키마7(필수아님, NULL이면 NULL입력)</label>
78   <input class="form-control" type="text" name="schema7" required />
79 </div>
80 <div class="form-group">
81   <label for="schema8">스키마8(필수아님, NULL이면 NULL입력)</label>
82   <input class="form-control" type="text" name="schema8" required />
83 </div>
84 <div class="form-group">
85   <label for="schema9">스키마9(필수아님, NULL이면 NULL입력)</label>
86   <input class="form-control" type="text" name="schema9" required />
87 </div>
88 <div class="form-group">
89   <label for="schema10">
90     >스키마10(필수아님, NULL이면 NULL입력)</label>
91   </div>
92   <input
93     class="form-control"
94     type="text"
95     name="schema10"
96     required
97   />
98 </div>
99 <button class="btn btn-primary" type="submit" id="task">
100   태스크 생성
101 </button>
102 </form>
103 <hr class="mb-3" />
104 <div><a href="/">홈으로 돌아가기</a></div>
105 </div>
106 </div>
107 </div>
108 </body>
109 </html>
110

```

- 그래서 태스크만들기로 들어가면 아래와 같은 창이 뜬다. 스키마 1~ 스키마 4는 고유 스키마이므로 최소 스키마 개수는 4이고 스키마 5부터 입력하면 된다.

### 태스크 생성

태스크 이름	<input type="text"/>
설명	<input type="text"/>
태스크 데이터 테이블 이름	<input type="text"/>
주기	<input type="text"/> ex) 235959 / 23:59:59
제출자 유형	<input checked="" type="radio"/> 식당 <input type="radio"/> 보건소
입력할 스키마 수(4이상 10이하 정수)	<input type="text"/>
스키마5(필수아님, NULL이면 NULL입력)	<input type="text"/>
스키마6(필수아님, NULL이면 NULL입력)	<input type="text"/>
스키마7(필수아님, NULL이면 NULL입력)	<input type="text"/>
스키마8(필수아님, NULL이면 NULL입력)	<input type="text"/>
스키마9(필수아님, NULL이면 NULL입력)	<input type="text"/>
스키마10(필수아님, NULL이면 NULL입력)	<input type="text"/>
<b>태스크 생성</b>	
<a href="#">홈으로 돌아가기</a>	

- 각 테이블의 고유 스키마는 제출자 유형이 식당인 경우와 보건소인 경우 각각 4 개로 아래와 같다.

```
CREATE TABLE TASKSCHEMA(ROLE ENUM('R', 'H') NOT NULL, SCHEMA1 VARCHAR(20), SCHEMA2 VARCHAR(20), SCHEMA3 VARCHAR(20), SCHEMA4 VARCHAR(20), PRIMARY KEY(ROLE));  
  
INSERT INTO TASKSCHEMA VALUE('R', 'PNUM', 'ADDRESS', 'VDATE', 'VTIME');  
INSERT INTO TASKSCHEMA VALUE('H', 'PNUM', 'ADDRESS', 'VDATE', 'PN');
```

- 태스크를 생성하면 TASK 테이블에 생성한 태스크 정보를 추가하고 태스크데이터테이블을 생성한다. TYPEORIGIN 테이블에 생성한 태스크의 스키마 종류와 고유스키마의 위치, 총 스키마 수를 추가한다. 제출자 유형이 식당일 때와 보건소일 때 태스크데이터테이블의 스키마가 다르므로 TYPEORIGIN 테이블에 스키마를 추가할 때와 태스크데이터 테이블을 생성할 때 나누어서 sql 문을 작성한다.

```
88 router.post("/generatetaskAf", function (req, res) {
89     var body = req.body;
90     var sql = "INSERT INTO task VALUES(?, ?, ?, ?, ?, ?)";
91     var params = [
92         body.name,
93         body.description,
94         body.taskdatatable,
95         body.cycle,
96         body.role,
97     ];
98     connection.query(sql, params, function (err) {
99
100    if (body.role === "R") {
101        var sql2 =
102            "CREATE TABLE " +
103            body.taskdatatable +
104            "(PNUM VARCHAR(11) NOT NULL, ADDRESS ENUM('강남', '서초', '용산', '마포', '서대문') NOT NULL, VDATE DATE NOT NULL, VTIME TIME NOT NULL, SID
105    } else {
106        var sql2 =
107            "CREATE TABLE " +
108            body.taskdatatable +
109            "(PNUM VARCHAR(11) NOT NULL, ADDRESS ENUM('강남', '서초', '용산', '마포', '서대문') NOT NULL, VDATE DATE NOT NULL, PN VARCHAR(1) NOT NULL,
110    }
111
112    connection.query(sql2, params, function (err) {
```

```
123 if(body.role === "R"){
124     var sql3 = "INSERT INTO TYPEORIGIN(SCHEMA1, SCHEMA2, SCHEMA3, SCHEMA4, SCHEM5, SCHEM6, SCHEM7, SCHEM8, SCHEM9, SCHEM10, CPNUM, CADDRESS, CVDATE, CEXTRA, SCHEMANUM) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
125     var params = [
126         'PNUM',
127         'ADDRESS',
128         'VDATE',
129         'VTIME',
130         body.schema5,
131         body.schema6,
132         body.schema7,
133         body.schema8,
134         body.schema9,
135         body.schema10,
136         0,
137         1,
138         2,
139         3,
140         body.numschema,
141     ];
142     connection.query(sql3, params, function (err) {
143         if (err) {
144             console.log("Error: " + err);
145         }
146     });
147 }
148 else{
149     var sql3 = "INSERT INTO TYPEORIGIN(SCHEMA1, SCHEMA2, SCHEMA3, SCHEMA4, SCHEM5, SCHEM6, SCHEM7, SCHEM8, SCHEM9, SCHEM10, CPNUM, CADDRESS, CVDATE, CEXTRA, SCHEMANUM) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
150     var params = [
151         'PNUM',
152         'ADDRESS',
153         'VDATE',
154         'PN',
155         body.schema5,
156         body.schema6,
157         body.schema7,
158         body.schema8,
159         body.schema9,
160         body.schema10,
161         0,
162         1,
163         2,
164         3,
165         body.numschema,
166     ];
167     connection.query(sql3, params, function (err) {
168         if (err) {
169             console.log("Error: " + err);
170         }
171     });
172 }
```

- 아래와 같이 입력하고 하단의 태스크 생성 버튼을 누르면

**태스크 생성**

태스크 이름  
**우유태스크**

설명  
**우유태스크입니다**

태스크 데이터 테이블 이름  
yyy

주기  
10:00:00

제출자 유형  
 식당 ○ 보건소

입력할 스키마 수(4이상 10이하 정수)  
7

스키마5(필수아님, NULL이면 NULL입력)  
TEMP

스키마6(필수아님, NULL이면 NULL입력)  
PN

스키마7(필수아님, NULL이면 NULL입력)  
BDATE

스키마8(필수아님, NULL이면 NULL입력)  
NULL

스키마9(필수아님, NULL이면 NULL입력)  
NULL

스키마10(필수아님, NULL이면 NULL입력)  
NULL

**태스크 생성**

- 아래와 같이 corona 데이터베이스에 저장되고 태스크데이터테이블이 생성된 것을 확인할 수 있다.

```
mysql> SELECT* FROM TASK;
+-----+-----+-----+-----+
| NAME | DESCRIPTION | TASKDATATABLE | CYCLE | ROLE |
+-----+-----+-----+-----+
| 맨도통식당태스크 | 맨도통식당의 태스크입니다. | TDT_jdpark | 24:00:00 | R |
| 밤치킨식당태스크 | 밤치킨식당 태스크입니다. | waitchang01 | 24:00:00 | R |
| 보건소태스크 | 보건소의 태스크입니다. | TDT_hospital1004 | 24:00:00 | H |
| 삼미식당태스크 | 삼미식당의 태스크입니다. | TDT_sk0101 | 24:00:00 | R |
| 우유태스크 | 우유태스크입니다. | yyy | 10:00:00 | R |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> SELECT* FROM TYPEORIGIN;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | SCHEMA1 | SCHEMA2 | SCHEMA3 | SCHEMA4 | SCHEMA5 | SCHEMA6 | SCHEMA7 | SCHEMA8 | SCHEMA9 | SCHEMA10 | CNUM | ADDRESS | CYDATE | CEXTRA | SCHEMANUM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | VDATE | VTIME | PNUM | NAME | ADDRESS | AGE | GENDER | NULL | NULL | NULL | 2 | 4 | 0 | 1 | 7 |
| 2 | AGE | GENDER | NAME | PNUM | ADDRESS | DEPENDENT | VDATE | VTIME | NULL | NULL | 3 | 4 | 6 | 7 | 8 |
| 3 | NAME | AGE | PNUM | ADDRESS | VDATE | VTIME | PN | NULL | NULL | 2 | 3 | 4 | 6 | 7 |
| 4 | PNUM | ADDRESS | VDATE | VTIME | TEMP | PN | BDATE | NULL | NULL | 0 | 1 | 2 | 3 | 7 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> SELECT* FROM yyy;
Empty set (0.01 sec)
```

- 제출자 유형이 보건소일 때도 확인해보자. 아래와 같이 입력하고 하단의 태스크 생성 버튼을 누르면

**태스크 생성**

태스크 이름	신발 태스크
설명	신발 태스크입니다.
태스크 데이터 테이블 이름	asd
주기	11:11:11
제출자 유형	<input type="radio"/> 식당 <input checked="" type="radio"/> 보건소
입력할 스키마 수(4이상 10이하 정수)	4
스키마5(필수아님, NULL이면 NULL입력)	NULL
스키마6(필수아님, NULL이면 NULL입력)	NULL
스키마7(필수아님, NULL이면 NULL입력)	NULL
스키마8(필수아님, NULL이면 NULL입력)	NULL
스키마9(필수아님, NULL이면 NULL입력)	NULL
스키마10(필수아님, NULL이면 NULL입력)	NULL
<b>태스크 생성</b>	
<a href="#">홈으로 돌아가기</a>	

- 아래와 같이 데이터베이스에 저장되고 태스크 데이터 테이블이 생성된다.

```
mysql> SELECT* FROM TASK;
+-----+-----+-----+-----+-----+
| NAME | DESCRIPTION | TASKDATATABLE | CYCLE | ROLE |
+-----+-----+-----+-----+-----+
| 맨도총식당태스크 | 맨도총식당의 태스크입니다. | TDT_jdpark | 24:00:00 | R |
| 방치킨식당태스크 | 방치킨식당 태스크입니다. | waitchang01 | 24:00:00 | R |
| 보건소태스크 | 보건소의 태스크입니다. | TDT_hospital004 | 24:00:00 | H |
| 삼미식당태스크 | 삼미식당의 태스크입니다. | TDT_sk0101 | 24:00:00 | R |
| 신발 태스크 | 신발 태스크입니다. | asd | 11:11:11 | H |
| 우유태스크 | 우유태스크입니다. | vvv | 10:00:00 | R |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> SELECT* FROM TYPEORIGIN;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | SCHEMA1 | SCHEMA2 | SCHEMA3 | SCHEMA4 | SCHEMA5 | SCHEMA6 | SCHEMA7 | SCHEMA8 | SCHEMA9 | SCHEM10 | CPNUM | CADDRESS | CVDATE | CEXTRA | SCHEMANUM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | VDATE | VTIME | PNUM | NAME | ADDRESS | AGE | GENDER | NULL | NULL | NULL | 2 | 4 | 0 | 1 | 7 |
| 2 | AGE | GENDER | NAME | PNUM | ADDRESS | DEPENDENT | VDATE | VTIME | NULL | NULL | 3 | 4 | 6 | 7 | 8 |
| 3 | NAME | AGE | PNUM | ADDRESS | VDATE | VTIME | PN | NULL | NULL | NULL | 2 | 3 | 4 | 6 | 7 |
| 4 | PNUM | ADDRESS | VDATE | VTIME | TEMP | PN | BDATE | NULL | NULL | NULL | 0 | 1 | 2 | 3 | 7 |
| 5 | PNUM | ADDRESS | VDATE | PN | NULL | NULL | NULL | NULL | NULL | NULL | 0 | 1 | 2 | 3 | 4 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> SELECT* FROM asd;
Empty set (0.00 sec)
```

- 이미 TASK 테이블에 존재하는 태스크 이름을 입력한 경우, NAME 이 TASK 테이블의 primary key 이므로 아래와 같이 데이터베이스에 추가되지 않는다.

### 태스크 생성

태스크 이름	<input type="text" value="맨도총식당태스크"/>
설명	<input type="text" value="맨도총식당태스크입니다."/>
태스크 데이터 테이블 이름	<input type="text" value="vvv"/>
주기	<input type="text" value="14:14:14"/>
제출자 유형	<input checked="" type="radio"/> 식당 <input type="radio"/> 보건소
입력할 스키마 수(4이상 10이하 정수)	<input type="text" value="4"/>
스키마5(필수아님, NULL이면 NULL입력)	<input type="text" value="NULL"/>
스키마6(필수아님, NULL이면 NULL입력)	<input type="text" value="NULL"/>
스키마7(필수아님, NULL이면 NULL입력)	<input type="text" value="NULL"/>
스키마8(필수아님, NULL이면 NULL입력)	<input type="text" value="NULL"/>
스키마9(필수아님, NULL이면 NULL입력)	<input type="text" value="NULL"/>
스키마10(필수아님, NULL이면 NULL입력)	<input type="text" value="NULL"/>
<input type="button" value="태스크 생성"/>	
<a href="#">홈으로 돌아가기</a>	

```
mysql> SELECT* FROM TASK;
+-----+-----+-----+-----+-----+
| NAME | DESCRIPTION | TASKDATATABLE | CYCLE | ROLE |
+-----+-----+-----+-----+-----+
| 맨도총식당태스크 | 맨도총식당의 태스크입니다. | TDT_jdpark | 24:00:00 | R |
| 밤치킨식당태스크 | 밤치킨식당 태스크입니다. | waitchang01 | 24:00:00 | R |
| 보건소태스크 | 보건소의 태스크입니다. | TDT_hospital1004 | 24:00:00 | H |
| 삼미식당태스크 | 삼미식당의 태스크입니다. | TDT_sk0101 | 24:00:00 | R |
| 신발 태스크 | 신발 태스크입니다. | asd | 11:11:11 | H |
| 후유태스크 | 후유태스크입니다. | yyy | 10:00:00 | R |
+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)
```

```
mysql> SELECT* FROM TYPEORIGIN;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | SCHEMA1 | SCHEMA2 | SCHEMA3 | SCHEMA4 | SCHEMA5 | SCHEMA6 | SCHEMA7 | SCHEMA8 | SCHEMA9 | SCHEMA10 | CPNUM | CADDRESS | CYDATE | CEXTRA | SCHEMANUM |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | VDATE | VTIME | PNUM | NAME | ADDRESS | AGE | GENDER | NULL | NULL | NULL | 2 | 4 | 0 | 1 | 7 |
| 2 | AGE | GENDER | NAME | PNUM | ADDRESS | DEPENDENT | VDATE | VTIME | PN | NULL | 3 | 4 | 6 | 7 | 8 |
| 3 | NAME | AGE | PNUM | ADDRESS | VDATE | VTIME | PN | NULL | NULL | 2 | 3 | 4 | 6 | 7 |
| 4 | PNUM | ADDRESS | VDATE | VTIME | TEMP | PN | BDATE | NULL | NULL | 0 | 1 | 2 | 3 | 7 |
| 5 | PNUM | ADDRESS | VDATE | PN | NULL | NULL | NULL | NULL | NULL | 0 | 1 | 2 | 3 | 4 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

## (2) 태스크 관리

- 훔으로 돌아와서 '태스크 관리'-'태스크 요청 확인하기'를 들어가면 REQUEST 테이블에 있는 제출자가 요청한 태스크 요청 리스트를 볼 수 있다. '태스크 관리'-'참여중인 태스크 확인하기'를 들어가면 APPROVAL 테이블에 있는 관리자가 참여 요청을 승인한 태스크 참여 명단을 볼 수 있다.

```
router.get("/request", function (req, res) {
  var sql = "SELECT * FROM REQUEST";
  connection.query(sql, function (err, rows, fields) {
    if (err) console.log("query is not executed. select fail...\n" + err);
    else res.render("request.ejs", { REQUEST: rows });
  });
});
```

```

15           <th>태스크 이름</th>
16           <th>제출자 ID</th>
17           <th>역할</th>
18       </tr>
19   </thead>
20   <tbody>
21       <% REQUEST.forEach(function(item, index) { %>
22       <tr>
23           <td><%=item.TASKNAME %></td>
24           <td><%=item.SUBMITTERID%></td>
25           <% if (item.ROLE === "R") {%
26           <td>식당</td>
27           <%} %> <% if (item.ROLE === "H") {%
28           <td>보건소</td>
29           <%} %>
30       </tr>
31   <% }()%>
32   </tbody>
33 </table>

```

```

mysql> SELECT* FROM REQUEST;
+-----+-----+-----+
| TASKNAME | SUBMITTERID | ROLE |
+-----+-----+-----+
| 밤지킨식당태스크 | waitchang01 | R |
| 밤지킨식당태스크 | waitsheo02 | R |
| 삼미식당태스크 | waitchang01 | R |
+-----+-----+-----+
3 rows in set (0.01 sec)

```

### 태스크 요청 명단

태스크 이름	제출자 ID	역할
밤지킨식당태스크	waitchang01	식당
밤지킨식당태스크	waitsheo02	식당
삼미식당태스크	waitchang01	식당

첫 요청부터 처리 가능

[승인](#) [거절](#)

[홈으로 돌아가기](#)

```

249 router.get("/approval", function (req, res) {
250     var sql = "SELECT * FROM APPROVAL";
251     connection.query(sql, function (err, rows, fields) {
252         if (err) console.log("query is not executed. select fail...\n" + err);
253         else res.render("approval.ejs", { APPROVAL: rows });
254     });
255 });

```

```

7   <body>
8     <div class="container">
9       <div class="row mt-4">
10      <h2>태스크 참여 명단</h2>
11      <hr class="mb-3" />
12      <div class="table-responsive-sm"></div>
13      <table class="table table-hover table-striped table-sm table-bordered">
14        <thead class="thead-dark">
15          <tr>
16            <th>태스크 이름</th>
17            <th>제출자 ID</th>
18            <th>역할</th>
19          </tr>
20        </thead>
21        <tbody>
22          <% for(i = 0; i < APPROVAL.length; i++) { %>
23          <tr>
24            <td><%=APPROVAL[i].TASKNAME %></td>
25            <td><%=APPROVAL[i].SUBMITTERID %></td>
26            <% if (APPROVAL[i].ROLE ==="R") { %> <td>식당</td> <%}>
27            <% if (APPROVAL[i].ROLE === "H") { %> <td>보건소</td> <%}>
28          </tr>
29          <% } %>
30        </tbody>
31      </table>
32    </div>

```

```

mysql> SELECT* FROM APPROVAL;
+-----+-----+-----+
| TASKNAME | SUBMITTERID | ROLE |
+-----+-----+-----+
| 맨도借此당태스크 | jdpark | R |
| 밤치킨식당태스크 | nightchicken | R |
| 보건소태스크 | hospital1004 | H |
| 삼미식당태스크 | sk0101 | R |
+-----+-----+-----+
4 rows in set (0.01 sec)

```

## 태스크 참여 명단

태스크 이름	제출자 ID	역할
맨도借此당태스크	jdpark	식당
밤치킨식당태스크	nightchicken	식당
보건소태스크	hospital1004	보건소
삼미식당태스크	sk0101	식당

[홈으로 돌아가기](#)

- 태스크 요청 명단에서 승인을 누르면 REQUEST 테이블에서는 없어지고 APPROVAL 테이블이 추가되어야 한다. 거절을 누르면 REQUEST 테이블에서 없어지기만 한다.

```

213 router.get("/apply_request", function (req, res) {
214   connection.query("SELECT * FROM REQUEST", function (err, row, fields) {
215     if (err) console.log("query is not executed. select fail...\n" + err);
216     else {
217       connection.query(
218         "insert into APPROVAL VALUE (?, ?, ?)",
219         [row[0].TASKNAME, row[0].SUBMITTERID, row[0].ROLE],
220         function (err, row, fields) {
221           if (err)
222             console.log("query is not executed. select fail...\n" + err);
223           else {
224             connection.query(
225               "delete from request LIMIT 1",
226               function (err, row, fields) {
227                 if (err)
228                   console.log(
229                     "query is not executed. select fail...\n" + err
230                   );
231                 else {
232                   res.redirect("/request");
233                 }
234               }
235             );
236           }
237         );
238       });
239     });
240   });
241 });

```

```

201 router.get("/deny_request", function (req, res) {
202   connection.query(
203     "delete from REQUEST LIMIT 1",
204     function (err, row, fields) {
205       if (err) console.log("query is not executed. select fail...\n" + err);
206       else {
207         res.redirect("/request");
208       }
209     );
210   });
211 });

```

- 홈에서 '모든 회원 리스트'에 들어가서 ID 기준 검색에서 제출자 ID 를 검색하면 제출자의 점수를 확인할 수 있다. 태스크 요청 명단은 첫 요청부터 처리가 가능하므로 일단 첫 요청의 제출자 ID 인 waitchang01 을 모든 회원 리스트에서 검색하면 이렇게 나온다.

### 기능별 검색

ID	이름	성별	주소	생일	전화번호	나이	유형(역할)	제출자 점수
waitchang01	강대기	F	서초	1983-08-01	01010001004	38	R	0

- 승인 요청을 받는 기준은 점수 5 점 이상이다. waitchang01 은 0 점 이므로 태스크 요청 명단에서 거절을 누른다. 거절을 누르면 아래와 같이 맨 위의 태스크가 사라진다. REQUEST 테이블에도 이 태스크가 사라진다. 당연히 태스크 참여 명단과 APPROVAL 테이블은 위와 그대로이다.

## 태스크 요청 명단

태스크 이름	제출자 ID	역할
밤치킨식당태스크	waitsheo02	식당
삼미식당태스크	waitchang01	식당

```
mysql> SELECT* FROM REQUEST;
+-----+-----+-----+
| TASKNAME | SUBMITTERID | ROLE |
+-----+-----+-----+
| 밤치킨식당태스크 | waitsheo02 | R |
| 삼미식당태스크 | waitchang01 | R |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

- 검색하면 waitsheo02 점수도 0 점으로 기준 미달이지만 일단 시현을 위해 승인을 누른다.
- 승인을 누르면 아래와 같이 요청 명단에서 사라지고 REQUEST 테이블에서 사라진다. 태스크 참여 명단과 APPROVAL에 사라진 태스크가 추가된다.

## 태스크 요청 명단

태스크 이름	제출자 ID	역할
삼미식당태스크	waitchang01	식당

```
mysql> SELECT* FROM REQUEST;
+-----+-----+-----+
| TASKNAME | SUBMITTERID | ROLE |
+-----+-----+-----+
| 삼미식당태스크 | waitchang01 | R |
+-----+-----+-----+
1 row in set (0.00 sec)
```

## 태스크 참여 명단

태스크 이름	제출자 ID	역할
멘도借此당태스크	jdpark	식당
밤치킨식당태스크	nightchicken	식당
밤치킨식당태스크	waitsheo02	식당
보건소태스크	hospital1004	보건소
삼미식당태스크	sk0101	식당

```
mysql> SELECT* FROM APPROVAL ;
+-----+-----+-----+
| TASKNAME | SUBMITTERID | ROLE |
+-----+-----+-----+
| 맨도借此당태스크 | jdpark | R |
| 밤치킨식당태스크 | nightchicken | R |
| 밤치킨식당태스크 | waitsheo02 | R |
| 보건소태스크 | hospital1004 | H |
| 삼미식당태스크 | sk0101 | R |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

### (3) 태스크 통계

#### 4.3.4.3. 태스크 통계

- 관리자는 각 태스크 별로 전체 제출된 파일 수와 pass되어 태스크 데이터 테이블에 저장된 tuple 수를 확인할 수 있다. 또한 이를 원본 데이터 탑재 수준에서도 볼 수 있다.
- 또한 각 태스크에 참여중인 제출자들의 목록은 살펴 볼 수 있다. 이때 제출자를 개별 선택하면 각 제출자가 참여중인 태스크 등을 확인할 수 있다.

- 관리자(ID=admin/PW=admin)으로 로그인하여 들어가면 위 상단 메뉴 바 중 “태스크 통계” 기능에 대한 설명이다.

#### 태스크별 제출된 전체파일 수

태스크 이름	전체 파일 수
멘도롱식당 태스크	3
밤치킨식당 태스크	2
삼미식당 태스크	4
보건소 태스크	2

#### PASS되어 태스크 데이터 테이블에 저장된 TUPLE 수

태스크 이름	PASS후 저장된 튜플 수
멘도롱식당 태스크	2
밤치킨식당 태스크	1
삼미식당 태스크	3
보건소 태스크	1

- 먼저 관리자는 각 태스크 별로 제출된 전체파일 수를 확인할 수 있다.  
메뉴상단 메뉴바에서 <태스크 통계>를 클릭하면 web.js 파일에서 "/taskStats"가 실행되었음을 알고 적힌 여러 sql 문을 수행하고, 오류 없이 수행되면 마지막으로 taskStats.ejs 파일을 실행시킨다. taskStats.ejs 파일에는 sql에서 수행한 결과들을 끌어온다.

- “태스크별 제출된 전체파일 수”에 표시될 테이블에는 식당파싱데이터 테이블(RPARSING)에서 가져온 태스크이름과 몇 개의 파일 제출되었는지를 count 해주는 열이 필요하다. 보건소 파싱데이터 테이블(HPARSING)에서도 마찬가지의 방법으로 수행해주면 된다. (sql1 & sql3)

```
app.get("/taskStats", function (req, res) {
  var sql =
    "SELECT TASKNAME, COUNT(TASKNAME) AS NUM FROM RPARSING GROUP BY TASKNAME";
  var sql2 =
    "SELECT E.taskname, count(*) as cnt from (select taskname, pass from RPARSING) as E where E.pass = 'P' group by E.taskname";
  var sql3 =
    "SELECT TASKNAME, COUNT(TASKNAME) AS NUM FROM HPARSING GROUP BY TASKNAME";
  var sql4 =
    "SELECT E.taskname, count(*) as cnt from (select taskname, pass from HPARSING) as E where E.pass = 'P' group by E.taskname";
```

- “PASS 되어 태스크 데이터 테이블에 저장된 tuple 수” 테이블은 마찬가지로 식당파싱데이터 테이블(RPARSING)에서 태스크이름과 개수를 가져오되, pass 속성값이 ‘P’인 값을 뽑는 sql 질의를 추가해준다. 보건소 파싱데이터 테이블(HPARSING)에서도 마찬가지의 방법이다. (sql2 & sql4)

- “원본 데이터 탑입 수준에서도” 전체 제출된 파일과 PASS 되어 태스크 데이터 테이블에 저장된 tuple 수를 볼 수 있게 하였다.

#### [원본 데이터 탑입 수준](#)에서 전체 제출된 목록보기

태스크 이름	원본 데이터 탑입
맨도롱식당 태스크	1
삼미식당 태스크	1
밤치킨식당 태스크	1
삼미식당 태스크	2
밤치킨식당 태스크	2
보건소 태스크	3

#### [원본 데이터 탑입 수준](#)에서 PASS되어 태스크 데이터 테이블에 저장된 TUPLE 수 보기

태스크 이름	원본 데이터 탑입
맨도롱식당 태스크	1
삼미식당 태스크	1
밤치킨식당 태스크	1
삼미식당 태스크	2
보건소 태스크	3

```
var sql5 = "SELECT taskname,tid FROM RPARSING GROUP BY taskname,tid";
var sql6 =
| "SELECT taskname,tid FROM RPARSING where pass= 'P' GROUP BY taskname,tid";
var sql7 = "SELECT taskname,tid FROM HPARSING GROUP BY taskname,tid";
var sql8 =
| "SELECT taskname,tid FROM HPARSING where pass= 'P' GROUP BY taskname,tid";
```

- 위와 동일한 방법이지만 tid 를 추가로 선택하여 어떠한 원본데이터 탑입에 해당하는 태스크인지 확인할 수 있게 “원본 데이터 탑입 수준에서도” 볼 수 있다. (sql5 & sql7, sql6 & sql8)
- “각 태스크에 참여중인 제출자들의 목록”은 화면 아래에 링크를 클릭하면 확인할 수 있다. 클릭하여 링크로 들어가면 각 태스크별로 현재 참여중인 제출자들의 id 를 확인할 수 있다.

#### [각 태스크에 참여중인 제출자들의 목록](#)

[홈으로 돌아가기](#)

#### 각 태스크에 참여중인 제출자 목록

태스크 이름	제출자 목록
맨도롱식당 태스크	jdpark,sk0101
밤치킨식당 태스크	nightchicken
삼미식당 태스크	nightchicken,sk0101
보건소 태스크	hospital1004

- “이때 제출자를 개별 선택하면 각 제출자가 참여중인 태스크를 보여주는” 것은 <제출자별 참여태스크 확인> 테이블에서 가능하다.

## 제출자별 참여태스크 확인

제출자 이름
jdpark
nightchicken
sk0101
hospital1004

- 이곳에 현재까지 모든 제출자들의 명단이 있으며, 클릭하여 참여중인 태스크를 확인하고 싶은 제출자가 있는 행을 클릭하여 참여중인 태스크를 확인할 수 있다. 예시로 제출자 이름이 'sk0101'이 참여중인 태스크를 확인하고 싶다면 세번째 행을 클릭하자.

localhost:8080/move?id=sk0101

## 제출자별 참여중인 태스크 보기

태스크 이름
삼미식당 태스크
맨도롱식당 태스크

[이전으로 돌아가기](#)

- 클릭하면 클릭한 특정 제출자가 참여중인 태스크를 볼 수 있고, 이것은 submitterPerTask.ejs 파일에 <script> jquery 를 추가해 구현했다. 클릭한 row 의 내용을 받아서 move.ejs 파일에 값을 넘겨주면, move.ejs 파일에서 테이블을 만들어 결과를 display 해주는 방식이다.

```
<script>
    $("#submitter_task tr").click(function(){
        var str = ""

        // clicked Row(<tr>)
        var tr = $(this);
        var td = tr.children();
        var submitterid = td.eq(0).text();

        window.location.href = "/move?id=" + submitterid;
    });
</script>
```

10

관리자는 각 태스크 별로 현재까지 태스크 데이터 테이블에 모인 모든 튜플들을 csv 파일로 다운로드 할 수 있어야 한다.

- 관리자의 기능 중 각 태스크 별로 현재까지 태스크 데이터 테이블에 모인 모든 튜플들을 csv 파일로 다운로드 할 수 있다는 기능에 대한 설명이다.

- 관리자(admin/admin)으로 로그인하여 들어가면 메뉴 상단에 "다운로드"를 클릭하여 들어가면 다음과 같은 화면이 나타난다.

#### 태스크 별 태스크 데이터 테이블에 모인 csv 파일 다운로드

(태스크를 클릭하면 해당 태스크에 해당하는 태스크 데이터 테이블에 모인 튜플들을 csv 파일로 다운로드 할 수 있습니다)

식당 태스크 이름	원본 데이터 타입
밥치킨식당 태스크	1
밥치킨식당 태스크	2

보건소 태스크 이름	원본데이터 타입
보건소 태스크	3

[홈으로 돌아가기](#)

- 제출자가 태스크에 참여 신청을 하고 관리자가 이를 승인한 뒤 원본 데이터파일을 제출하면 그 파일은 파싱되어 파싱데이터 파일이 된다. 평가자는 이 파일을 평가하게 되고 평가여부와 패스여부가 결정되는데, 태스크 데이터 테이블의 튜플로 들어가게 되는 것은 이 중에서도 패스를 받은 것들이다. 따라서 화면에 보이는 것과 같이 해당 태스크의 이름과 원본 데이터파일의 타입에 따라 태스크 데이터 테이블에 저장된 csv 파일을 다운로드 받을 수 있는 기능이다.

- 식당 태스크와 보건소 태스크 테이블에서 본인이 다운로드 받고자 하는 csv 파일의 행을 클릭하면 다운로드가 진행된다.

- 이를 download.ejs 파일에 구현하였고 각 행의 클릭이 csv 파일을 다운로드 받을 수 있는 기능은 <script>태그를 이용한 jquery로 구현하였다

```
<script>
  $("#restaurant_task tr").click(function(){
    var str = ""

    // clicked Row(<tr>
    var tr = $(this);
    var td = tr.children();
    var name = td.eq(0).text();
    var tid = td.eq(1).text();

    str += " * Task name : <font color='red'>" + name + "</font>" + tid;
    $("#restaurant_Result").html(str);
    window.location.href = "/download_r?id=" + name + "?tid:" + tid;
  });

  $("#hospital_task tr").click(function(){
    var str = ""

    // clicked Row(<tr>
    var tr = $(this);
    var td = tr.children();
    var name = td.eq(0).text();
    var tid = td.eq(1).text();

    str += " * Task name : <font color='red'>" + name + "</font>" + tid;
  });
</script>
```

- 각 행을 클릭하면 해당 태스크 이름과 원본 데이터 탑입의 숫자를 넘겨주고 web.js에서 구현한 app.get으로 지정한 링크를 따라 sql 문이 실행되고 csv 파일 다운로드가 진행되게 된다.

- csv 파일의 다운로드의 방법은 평가자가 해당 태스크를 평가할 때 csv 파일을 다운로드하는 방법과 비슷하게 구현하였다. 아래 코드는 web.js에서 app.get으로 식당 태스크 csv 파일을 다운로드 하는 코드의 일부이다.

```
app.get("/download_r", function (req, res) {
  var index = req.query.id.indexOf("?");
  var index2 = req.query.id.indexOf(":");
  var name = req.query.id.substr(0, index);
  var type = req.query.id.substr(index2 + 1);

  var sql1 = "SELECT * FROM rparsing WHERE estate='Y' and pass = 'P' and taskname= ? and tid = ?";
  connection.query(sql1, [name, type], function (err, result1, fields) {
    if (err) console.log("query is not executed. select fail...\n" + err);
    else {
      var pid = result1[0].PARSINGID;
      var file =
        __dirname + "/../../data/parsed_csv_restaurant/" + pid + ".csv";
      try {
        if (fs.existsSync(file)) {
          // 파일이 존재하는지 체크
          var filename = path.basename(file); // 파일 경로에서 파일명(확장자포함)만 추출
          var mimetype = mime.getType(file); // 파일의 타입(형식)을 가져옴

          res.setHeader(
            "Content-disposition",
            "attachment; filename=" + filename
          ); // 다운받아질 파일명 설정
          res.setHeader("Content-type", mimetype); // 파일 형식 지정

          var filestream = fs.createReadStream(file);
        }
      } catch (e) {
        console.log(e);
      }
    }
  });
});
```

## (4) 회원 관리 및 통계

### 4.3.4.4. 회원 관리 및 통계

- 관리자는 현재 가입중인 모든 회원들의 리스트를 볼 수 있다.
- 역할, 나이대, 성별, 참여중인 태스크, ID를 기준으로 검색 가능하다.
- 검색 결과로 나온 각 회원을 클릭할 시 상세한 정보를 확인할 수 있다.
- 회원이 제출자인 경우 해당 회원이 참여중인 태스크와 그 태스크에 참여한 통계정보를 볼 수 있다.
- 회원이 평가자인 경우 해당 회원이 평가한 파싱 데이터 시퀀스 파일의 목록을 모두 볼 수 있다.

- 관리자(ID=admin/PW=admin)으로 로그인하여 들어가면 위 상단 메뉴 바 중 “모든 회원리스트”에 대한 설명이다.

- 관리자는 “현재 가입중인 모든 회원들의 리스트”를 볼 수 있어야 한다. “모든 회원리스트”를 클릭하면 web.js 에서 “/allUsers”가 실행되었음을 알고 적힌 sql 을 수행하고 오류없이 수행했다면 그 결과를 넘겨주고 allUsers.ejs 파일을 불러온다.

```
// All the list of users for administrator
router.get("/allUsers", function (req, res) {
  var sql = "SELECT * FROM submitter";
  var sql2 = "SELECT * FROM ESTIMATOR";
```

- 식당 제출자, 보건소 제출자, 평가자까지 모든 회원을 볼 수 있다. 평가는 제출자 점수에 해당하는 값이 없으므로 NULL 값을 넣어주었다.

### 모든 회원 명단

ID	이름	성별	주소	생일	전화번호	나이	유형(역할)	제출자 점수
hospital1004	신보건	F	강남	-02-22	01010001003	34	H	10
jdpark	박회원	M	서대문	-04-11	01010001000	31	R	10
newbiekim01	김새로이	F	마포	-01-26	01010001006	39	R	0
newbieyu02	유새로이	M	서대문	-11-26	01010001007	38	R	0
nightchicken	이회원	F	강남	-11-16	01010001002	36	R	5
sample	Submitter_Sample	M	강남	-01-01	01011114444	22	R	0
sk0101	김회원	M	용산	-08-29	01010001001	41	R	10
waitchang01	강대기	F	서초	-08-01	01010001004	38	R	0
waitsheo02	서대기	M	마포	-05-09	01010001005	40	R	0
estimator	Estimator_Sample	F	서초	-01-02	01033334444	22	E	NULL
evalcha03	차평가	M	마포	-10-30	01030001002	23	E	NULL
evalchoi01	최평가	F	서대문	-05-10	01030001000	22	E	NULL
evalhan42	한평가	M	서초	-01-04	01030001001	24	E	NULL
evalpark07	박평가	F	강남	-09-11	01030001003	25	E	NULL

- “역할, 나이대, 성별, 참여중인 태스크, id 를 기준으로 검색 가능”하도록 하면 아래에 다음과 같이 구현하였다.

## 검색 기능

역할 기준으로 검색 (제출자(식당) -> R, 제출자(보건소) -> H, 평가자 -> E)

**검색**

나이대 기준으로 검색 (ex) 22살 -> 22, 30살 -> 30)

**검색**

성별 기준으로 검색 (남자 -> M, 여자 -> F)

**검색**

태스크 기준으로 검색 (참여중인 태스크 이름 입력)

**검색**

ID 기준으로 검색 (ID 입력)

- 검색기능을 순서대로 추가했고, 검색하고자 하는 기준에 해당하는 폼에 입력값을 넣고 검색을 누르면 된다. 입력할 값의 형식은 괄호로 나타냈다. 만약 해당하는 값이 없다면 빈 테이블이 출력되며 성별 남자를 기준으로 검색하고 싶다면 위 그림과 같이 입력하고 이어 나오는 검색을 클릭한다.

## 기능별 검색

ID	이름	성별	주소	생일	전화번호	나이	유형(역할)	제출자 점수
jdpark	박회원	M	서대문	1990-04-11	01010001000	31	R	10
newbieyu02	유새로이	M	서대문	1983-11-26	01010001007	38	R	0
sample	Submitter_Sample	M	강남	1999-01-01	01011114444	22	R	0
sk0101	김회원	M	용산	1980-08-29	01010001001	41	R	10
waitsheo02	서대기	M	마포	1981-05-09	01010001005	40	R	0

ID	이름	성별	주소	생일	전화번호	나이	유형(역할)
evalcha03	차평가	M	마포	1998-10-30	01030001002	23	E
evalhan42	한평가	M	서초	1997-01-04	01030001001	24	E

[이전 페이지로 돌아가기](#)

- 그림과 같이 제출자와 평가자 테이블은 나누어서 출력되며 결과값을 확인하면 검색된 모든 값은 성별이 'M'임을 확인할 수 있다.
- 검색하고자 하는 기준에 해당하는 폼에 입력값을 넣은 뒤 검색을 클릭하면 입력값이 get 형식으로 전달되어 web.js에서 순서대로 "/searchByRole", "/searchByAge", "/searchByGender", "/searchByTask", "/searchByID"로 이동되고, 각각 해당하는 sql 문의 결과값을 수행하여 search.ejs 파일로 넘겨주게 된다. 마지막으로 search.ejs 파일에서 넘겨받은 sql 결과값에서 id, 이름 등 필요한

속성을 뽑아 테이블로 만들어 display 하는 방식이다. 다음 그림은 성별기준으로 검색할 때 수행한 과정이다.

```
router.get("/searchByGender", (req, res) => {
  var role = req.query.Gender;
  var sql = "SELECT * FROM submitter WHERE Gender= ?";
  var sql2 = "SELECT * FROM ESTIMATOR WHERE Gender= ?";

  connection.query(sql, [role[0]], function (err, rows, fields) {
    if (err) console.log("query is not executed. select fail...\n" + err);
    else {
      connection.query(sql2, [role[0]], function (err, results, fields) {
        if (err) console.log("query is not executed. select fail...\n" + err);
        else {
          res.render("search.ejs", { submitter: rows, estimator: results });
        }
      });
    }
  });
});
```

- 이어서, “검색결과로 나온 회원을 클릭하면 상세한 정보를 확인”할 수 있다. 정보를 확인하고 싶은제출자를 테이블에서 클릭하면 클릭한 특정 제출자가 참여중인 태스크와 참여 통계정보를 볼 수 있고, 이것은 search.ejs 파일에 <script> jquery 를 추가해 구현했다. 클릭한 row 의 내용을 받아서 searchSubmitter.ejs 파일에 값을 넘겨주면, searchSubmitter.ejs 파일에서 테이블을 만들어 결과를 display 해주는 방식이다.

```
<script>
$("#search_submit tr").click(function(){
  var str = ""

  // clicked Row(<tr>)
  var tr = $(this);
  var td = tr.children();
  var ID = td.eq(0).text();

  str += " * ID : <font color='red'>" + ID + "</font>";
  $("#searchsubmit_Result").html(str);
  window.location.href = "/searchSubmitter?id=" + ID;
});
```

- 정보를 확인하고 싶은 평가자를 테이블에서 클릭하면 마찬가지 방법으로 searchEst.ejs 파일을 읽어들여 특정 평가자가 평가한 파싱 데이터 시퀀스 파일의 목록을 모두 볼 수 있다.

- 예시로 성별 기준 'M'을 검색하여 제출자 ID 가 'jdpark'인 사람을 클릭했다면 다음 그림과 같이 표시된다. 태스크이름과 그 태스크의 파싱 ID 그리고 전체 제출한 파일 수와 PASS 되어 태스크 데이터 테이블에 저장된 tuple 수의 통계정보까지 보여준다.

### 선택한 제출자가 참여중인 태스크와 통계정보

해당 태스크의 ParsingID	태스크 이름	전체 제출한 파일 수
1	맨도롱식당 태스크	1
5	맨도롱식당 태스크	1

### 그 외의 통계정보

해당 태스크의 ParsingID	태스크 이름	PASS되어 태스크 데이터 테이블에 저장된 tuple수
1	맨도롱식당 태스크	1

- 클릭한 회원이 평가자라면 평가한 파싱 데이터 시퀀스 파일의 목록을 보여준다.

- 예시로 성별 기준 'M'을 검색하여 평가자 ID 가 'evalcha03'인 사람을 클릭했다면 다음 그림과 같이 표시된다. 평가자가 평가한 태스크의 이름과 그 태스크의 파싱 ID 를 나타낸다.

### 선택한 평가자가 평가한 파싱 데이터 시퀀스 파일 목록

데이터 시퀀스 파일 ParsingID	태스크 이름
3	밤치킨식당 태스크
6	삼미식당 태스크

## (5) 회원가입 및 인증 기능

### 1) 회원가입

#### 회원 가입

아이디

비밀번호 (비밀번호는 3자리 이상이어야 합니다.)

비밀번호 확인

이름

성별

남자  여자

주소

강남  서초  용산  마포  서대문

생년월일

ex) 19991021

전화번호

ex) 01012345678

역할

보건소

이미 회원이신가요? [로그인하기](#)

(/controllers/registerController.js, /service/registerService.js,

/validation/authValidaiton.js, /views/register.ejs)

- 회원가입 대상은 식당과 보건소 두 종류의 제출자와 평가자로 이루어져 있으며, 관리자는 사전 생성된 하나의 계정만을 가지므로 회원가입 영역에서 제외되었다. 모든 회원은 고유의 아이디, 비밀번호, 이름, 성별, 주소, 생년월일, 전화번호, 역할을 가지며, 이 역할들 중 식당을 선택하는 경우 상호명, 가게 주소, 가게 전화번호를 추가적으로 가진다.

- 아이디는 기준 회원과 중복되지 않아야 하며, 비밀번호는 3 자리 이상이어야 하며 확인자 입력하는 값과 일치해야 한다. 나머지 값들 또한 필수적으로 입력해야 하고, 전화번호, 생년월일의 경우 placeholder로 보여지는 정해진 형식을 따르도록 제약조건을 설정하였다.

실제 회원가입란을 채우고 기본적인 형식이 조건에 맞게 제출되면, 아이디의 존재 여부를 제출자, 평가자, 관리자 table에서 먼저 확인하고, 비밀번호 제출이 올바르게 되었는지를 authValidation.js에서 확인한다.

```
const today = new Date();
const salt = bcryptjs.genSaltSync(10);
const data = {
  id: user.id,
  password: bcryptjs.hashSync(user.password, salt),
  name: user.name,
  gender: user.gender,
  address: user.address,
  birthdate: user.birthdate,
  pnum: user.pnum,
  age: today.getFullYear() - user.birthdate.substring(0, 4) + 1,
  role: user.role,
};
```

```
if (data.role === "R") {
  data.rname = user.rname;
  data.rlocate = user.rlocate;
  data.rpnum = user.rpnum;

  connection.query([
    "INSERT INTO submitter SET ?",
    data,
    function (error, rows) {
      if (error) {
        reject(error);
      }
      resolve("Create a new user successfully");
    }
]);
```

- 가입한 역할에 맞게 정보를 넣는 과정에서, bcryptjs를 통해 비밀번호를 암호화하여 저장하도록 한다. 회원의 나이는 현재 연도와 생년월일을 기준으로 계산한 나이가 들어가도록 하였다. 식당은 R, 보건소는 H, 평가자는 E로 각각의 role을 구별하여 그에 맞는 table에 회원이 새로 들어가도록 INSERT INTO 쿼리문을 사용하였다.
- 위의 코드는 모든 회원들이 공통적으로 가지는 정보들을 받아와 data 변수에 저장하고, 이 중 역할이 식당일 경우 추가적인 정보들을 저장하여 submitter table에 삽입하는 코드이다.

## 2) 로그인

### 로그인

아이디

비밀번호

로그인하기

아직 회원이 아니신가요? [회원가입하기](#)

(/controllers/loginController.js, /controllers/passportController.js,

/service/loginService.js, /views/login.ejs)

- 로그인은 제출자, 평가자, 관리자 모두를 대상으로 하며 회원 DB에 있는 회원인지 확인한다. 아이디가 존재하지 않거나 비밀번호가 일치하지 않을 때 오류 메시지를 띄운다. 로그인에 성공하면 바로 홈페이지로 넘어가게 되고, 로그아웃하기 전까지 로그인 상태를 유지하도록 세션에 회원 정보를 저장해둔다.
- 사용자가 입력한 아이디와 비밀번호로 로그인을 하게 되면, 사용자가 작성한 값을 받아와 제출자, 평가자, 관리자에 해당하는 모든 table에서 존재하는 회원인지 조회를 한다. 아이디가 존재하는지 먼저 체크하고, 존재한다면 비밀번호가 일치하는지 bcryptjs에 있는 compare로 확인한다. 사용자가 DB에 없으면 에러메세지를 띄우고 로그인창을 redirect하고, 있다면 세션에 회원 정보를 저장하여 홈페이지로 redirect 한다.

```
const findUserById = (id) => {
  return new Promise((resolve, reject) => {
    try {
      connection.query(
        "SELECT * FROM submitter WHERE id = ?",
        id,
        function (error, rows) {
          if (error) {
            reject(error);
          }
          // the user is a submitter
          if (rows.length > 0) {
            resolve(rows[0]);
          }
        }
      );
    } catch (err) {
      reject(err);
    }
  });
}
```

- findUserById는 제출자, 평가자, 관리자 모든 table에 사용자가 로그인 폼에 입력한 아이디가 존재하는지 SELECT 문을 통해 확인한다. 존재한다면 rows.length의 길이가 0보다 클 것이고 아니라면 에러를 띄운다.

```

const comparePasswordUser = (user, password, id) => {
  return new Promise(async (resolve, reject) => {
    try {
      if (id === "admin") {
        if (user.PASSWORD === "admin" && password === "admin") resolve(true);
      }
      await bcrypt
        .compare(password.toString(), user.PASSWORD)
        .then((isMatch) => {
          if (isMatch) resolve(true);
          else resolve("잘못된 비밀번호입니다.");
        });
    } catch (e) {
      reject(e);
    }
  });
};

```

- comparePasswordUser 는 아이디가 존재한다면, 그 비밀번호가 일치하는지 확인하는 함수이다. 처음 사전 생성된 관리자 계정의 비밀번호가 시스템 상에서 admin 으로 자동으로 올라간 경우를 제외한 나머지 회원 가입/정보 수정을 통해 들어온 비밀번호들은

해시값으로 저장되어있으므로 compare 함수로 일치 여부를 확인한다.

### 3) 정보 수정

#### 내 정보 수정

아이디 (변경 불가능)

qwetrrt

비밀번호

(비밀번호를 변경하지 않으려면  
기존 비밀번호를 입력하세요)

비밀번호 확인

이름

김해나

성별

남자  여자

주소

강남  서초  용산  마포  서대문

생년월일

19950413

전화번호

01065458844

역할 (변경 불가능)

식당

상호명

우리카페

회원이 정보를 수정하며 입력한 품의 값으로 다시 회원 tuple 을 update 한다.

(/controllers/editinfoController.js,

/service/editinfoService.js,

/views/userinfo.ejs)

- 회원이 자신의 정보를 확인하고, 수정하는 부분을 같은 페이지에 두어 수정하지 않을 경우 홈으로 돌아오면 되는 형식을 취했다. 모든 회원 공통적으로 아이디와 역할은 변경이 불가능하며, 각 회원 종류마다 가지고 있는 정보에 따라 수정 가능한 항목들이 다르다. 예를 들어 회원가입을 하지 않아 가진 정보가 아이디와 비밀번호뿐인 관리자는 비밀번호만 수정이 가능하다.  
- 회원이 정보 수정창에서 입력한 비밀번호가 저장되며, 변경하지 않을 경우에는 기존 번호를, 변경할 경우에는 새로운 번호를 쓰도록 안내되어 있다. 회원의 정해진 아이디와 역할로 바꿔야 할 table 과 회원 tuple 을 찾는다.

```

} else if (role === "보건소") {
  const name = user.name;
  const gender = user.gender;
  const address = user.address;
  const birthdate = user.birthdate;
  const pnum = user.pnum;
  const age = today.getFullYear() - user.birthdate.substring(0, 4) + 1;
  connection.query(
    "UPDATE submitter SET password = ?, name = ?, gender = ?, address = ?, birthdate = ?, pnum = ?, age = ? WHERE id = ?",
    [password, name, gender, address, birthdate, pnum, age, id],
    function (error, rows) {
      if (error) {
        reject(error);
      } else {
        resolve("Update a user successfully");
      }
    });
}

```

- 회원가입하는 과정과 유사하지만, 이미 기존에 존재하는 아이디로 회원을 찾고 그 정보들을 바꿔주는 기능이다. 옆의 코드에서는 보건소 역할을 가진 회원의 정보를 UPDATE 문을 통해 새롭게 저장한다.

#### 4) 회원 탈퇴

### 회원 탈퇴하기

아이디

qwetrert

비밀번호

탈퇴 확인을 위한 비밀번호 입력

회원 탈퇴를 하면 모든 기록들이 삭제됩니다. 동의합니까?

동의  비동의

탈퇴가 완료되면 로그인 창으로 넘어갑니다.

탈퇴하기

(/controllers/disconnectController.js, /service/disconnectService.js, /views/disconnect.ejs)

- 관리자를 제외한 일반 회원들은 자신의 비밀번호를 정확히 입력하고, 탈퇴 동의를 하면 탈퇴가 완료된다. 잘못된 비밀번호를 쓰거나 비동의를 선택한다면 탈퇴가 되지 않고 오류메세지가 뜨며, 정상적으로 탈퇴가 되면 로그인창으로 넘어가게 된다.

- 로그인을 하는 상황과 유사하게 세션에 저장된 회원 아이디를 통해 실제 DB에 저장된 비밀번호와 탈퇴 확인용 비밀번호를 비교하고, 서로 다를 경우에 잘못된 비밀번호임을 알리며 탈퇴가 되지 않는다. 만약 비밀번호가 일치할 경우, 회원 탈퇴에 동의하면 역할에 맞는 table에서 DELETE 하고, 동의하지 않으면 탈퇴를 진행하지 않는다.

```
const foundUser = await findUserById(id, role);
await bcrypt
.compare(password.toString(), foundUser.PASSWORD)
.then((isMatch) => {
  if (!isMatch) {
    reject(`잘못된 비밀번호입니다.`);
  } else if (isMatch && disconnect === "A") {
    if (role === "R" || role === "H") {
      connection.query(
        "DELETE FROM submitter WHERE id = ?",
        id,
        function (error, rows) {
          if (error) {
            reject(error);
          }
          resolve("Delete a user successfully");
        }
      );
    }
  }
});
```

- 옆의 코드에서는 현재 로그인한 회원이 식당 혹은 보건소일 경우 회원 탈퇴를 하면 제출자 table에서 삭제되는 쿼리문을 돌리게 된다.

\*\* DB 를 dump 한 파일은 따로 첨부하였고, 관리자와 제출자, 평가자의 비밀번호는 모두 1234 로 설정하였으나 보안 때문에 암호화되어있습니다., 프로그램을 동작할 때에는 아무 문제 없습니다.