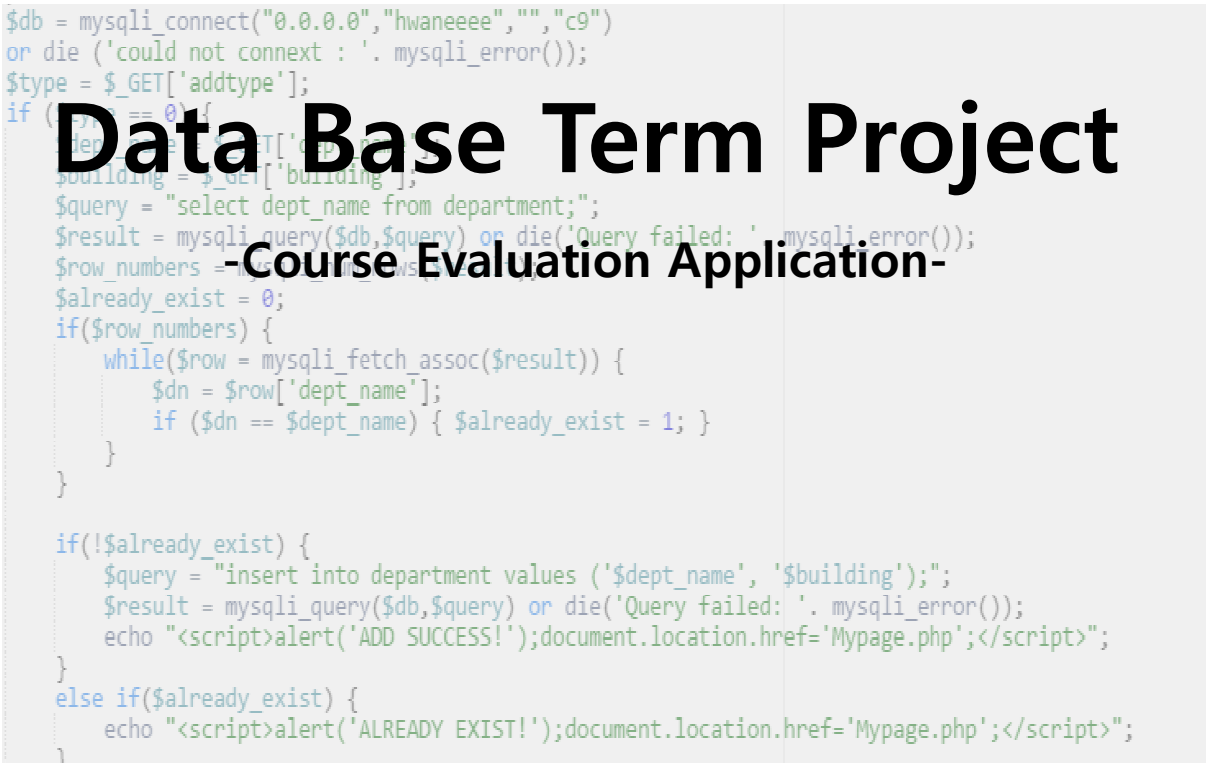


Data Base Term Project

-Course Evaluation Application-

컴퓨터학과

2013210064

최정환

1. Introducing

PHP를 이용하여 웹 기반에서 MYSQL DBMS를 이용하여 "강의 평가 Application"을 제작하였다. 처음에는 인터넷 서점을 구현하려고 하였으나, 너무나 많은 Relation과 구현하기 힘든 Function들로 인하여, 좀 더 간단한 Application을 구현하였다. "강의 평가 Application"은 총 8개의 Entity와 11개의 Relation으로 구성하였으며, User의 Type을 학생, 교수, 그리고 관리자로 나누었다. User의 Type에 따라서 강의를 평가하거나 평가를 확인할 수 있고, 관리자는 새로운 과목과 학생 그리고 강의를 추가할 수 있다. 결론적으로 총 12개의 PHP파일을 생성하였고 이를 통하여 다음과 같은 Function을 구현하였다.

I. Administer를 통한 회원가입, 과목추가, 강의 평가 삭제
II. Login / Logout
III. User Type이 학생일 경우, 강의 평가(삽입), 및 수정
IV. User Type이 교수일 경우, 자신의 강의 평가 확인(검색)
V. 강의 평가 통계 확인(검색)

Application 개발은 Cloud 9을 통해서 이루어 졌다. Cloud 9을 통하여 MYSQL 서버와 연동하여 Function을 구현하기 위하여 MYSQL DBMS를 이용하였다. Application의 전체적인 틀은 PHP를 이용하여 구현하였다.

2. Designing

(1) Conceptual Design

Conceptual Design을 구상 후 이를 바탕으로 E-R Diagram Logical Design을 수행하였다. 각각의 Entity Relation의 정의는 다음과 같다.

엔터티 정의서

엔터티 Set 명	student			Super/Sub	
엔터티 설명	Information of Student			관련 엔터티	
엔터티 Set					
속성 명	속성정의	타입	길이	제약조건	Key	비고
Student_id	각 학생의 식별자	varchar	15	PK	PK	
Student_name	학생의 이름	Varchar	15	Not Null		
Dept_name	소속된 과의 이름	Varchar	15	Not Null	FK	

엔터티 정의서

엔터티 Set 명	Professor			Super/Sub	
엔터티 설명	교수에 대한 정보를 기록한다.			관련 엔터티	
엔터티 Set					
속성 명	속성정의	타입	길이	제약조건	key	비고
Professor_id	교수의 식별자	Varchar	15	PK	PK	
Professor_name	교수의 이름	Varchar	15	Not Null		
Dept_name	교수가 소속된 과의 이름	Varchar	15		FK	

엔터티 정의서

엔터티 Set 명	course			Super/Sub		
엔터티 설명	강의에 대한 Entity			관련 엔터티		
엔터티 Set						
속성 명	속성정의	타입	길이	제약조건	key	비고
Course_id	강의에 대한 식별자	Varchar	15	PK	PK	
Title	강의 이름	Varchar	15	Not Null		
Dept_name	강의 열리는 학과의 이름	Varchar	15	Not null	FK	
Credits	학점	Varchar	15			
Professor_id	교수에 대한 식별자	Varchar	15	Not null	FK	
Max_number	해당 과목의 최대 수용 인원	Int		Not Null		
Course_type	전공/교양임을 구분지음	Varchar	15	Not Null		

엔터티 정의서

엔터티 Set 명	Section			Super/Sub		
엔터티 설명	Section에 관한 entity			관련 엔터티		
엔터티 Set						
속성 명	속성정의	타입	길이	제약조건	key	비고
Course_id	어떤과목의 section인지를 나타내는 식별자	Varchar	15	Not Null	PK,FK	
Sec_id	Section의 고유번호	Varchar	15	Not Null, Unique	PK	
Semester	해당 section이 어떤학기에 열렸는지 명시해줌	Varchar	15	Not Null	PK	
Year	해당 Section이 어떤연도에 열렸는지 명시해줌	Varchar	15	Not Null	PK	

엔터티 정의서

엔터티 Set 명	Takes			Super/Sub		
엔터티 설명	학생의 강의 수강 관련 entity			관련 엔터티		
엔터티 Set						
속성 명	속성정의	타입	길이	제약조건	key	비고
Student_id	학생의 식별자	Varchar	15	Not Nul	PK,FK	
Course_id	강의 식별자	Varchar	15	Not Null	PK,FK	
Post_id	강의 평가 게시판 식별자	Varchar	15	Not Null	FK	
Sec_id	Section 식별자	Varchar	15	Not Null	PK,FK	
Semester	Take 한 학기	Varchar	15	Not Null	PK,FK	
Year	Take한 년도	Varchar	15	Not Null	PK,FK	
Grade	학점	Varchar	15	Not null		

엔터티 정의서

엔터티 Set 명	Department			Super/Sub		
엔터티 설명	학과에 대한 Entity			관련 엔터티		
엔터티 Set						
속성 명	속성정의	타입	길이	제약조건	key	비고
Dept_name	학과의 이름, 각 학과에 대한 식별자	Varchar	15	PK		
Building	학과 건물 이름	Varchar	15			

엔터티 정의서

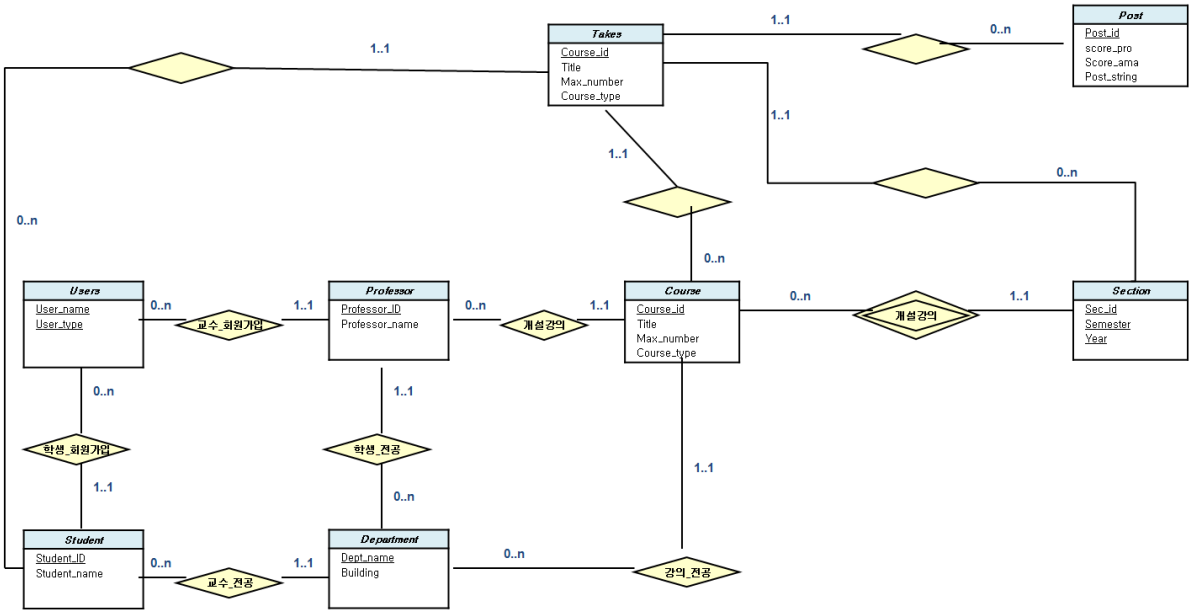
엔터티 Set 명	Post			Super/Sub		
엔터티 설명	게시판(강의 평가)에 대한 entity			관련 엔터티		
엔터티 Set						
속성 명	속성정의	타입	길이	제약조건	key	비고
Post_d	게시판에 대한 식별자	Varchar	15	PK	PK	
Score_pro	전공평점	Int				
Score_ama	교양평점	Int				
Post_string	평가 문구	Varchar	2000			

엔터티 정의서

엔터티 Set 명	Users			Super/Sub		
엔터티 설명	회원가입 entity			관련 엔터티		
엔터티 Set						
속성 명	속성정의	타입	길이	제약조건	key	비고
User_name	회원이름	Varchar	15	Not Nul	PK	
User_type	회원이 학생인지 교수인지 구분	Varchar	15	Not Null	PK	
Student_id	학생에 대한 식별자	Varchar	15		FK	
Professor_id	교수에 대한 식별자	Varchar	15		Fk	

관계 정의서

관계명	관계정의	관련엔터티	카디널리티	
			최소	최대
학생_회원가입	한 학생은 1명의 회원이 된다. 한 회원은 한 명의 학생일 될 수 있다.	Student	1	1
		Users	0	1
교수_회원가입	한 교수는 1명의 회원이 된다. 한 회원은 한 한 명의 교수가 될 수 있다.	Professor	1	1
		Users	0	1
개설 강의	한 강의는 하나의 교수에게만 열린다. 한 교수는 0개 이상의 강의를 열 수 있다.	Course	1	1
		Professor	0	N
학생_전공	한 학생은 한 개의 전공을 가진다. 한 전공은 0개 이상의 학생이 전공 중이다.	Student	1	1
		Department	0	N
교수_전공	한 교수는 한 개의 전공을 가진다. 한 전공은 0개 이상의 교수가 전공 중이다.	Professor	1	1
		Department	0	N
강의_전공	강의는 하나의 학과 과목이다. 학과는 0개 이상의 강의를 개설한다.	Course	1	1
		Department	0	N
Section_Course	하나의 강의는 0개 이상의 Section으로 열릴 수 있다. 하나의 Section은 하나의 강의로 열린다.	Course	0	N
		Section	1	1
Takes_*	한 학생/section/course/post의 0개 이상의 여러 수업을 듣는다. Take는 한 학생/section/course/post에게 대응된다. (같은 특징을 가지므로 생략한다.)	Student, Section, Course, Post	0	N
		Takes	1	1



STUDENT SCHEMA

ID : 학생의 고유 넘버, 수강평가 사이트의 아이디
name : 학생의 이름
dept_name : 소속학과

PROFESSOR SCHEMA

ID : 교수의 고유 넘버, 수강평가 사이트의 아이디
name : 교수의 이름
dept_name : 교수의 소속 학과

COURSE SCHEMA

course_id : 해당 과목의 고유한 아이디
title : 해당 과목의 이름 (예: 데이터베이스, 컴퓨터구조..)
dept_name : 해당 과목의 소속 학과
만약 해당 과목이 전공과목일 경우에는 dept_name 을 가지지만, 전공과목이 아니라 교양과목일 경우에는 dept_name 이 '교양'이라는 string 을 가지게 설정한다.
전공과목과 교양과목을 구분하기 위해서이다.
credits : 해당 과목의 학점
professor_id : 해당 과목을 강의하는 교수님의 고유번호
교수님으로 구분하여 검색하는 기능을 구현하기 위해서 교수님의 고유번호를 course schema 에 추가하였다. course schema 만 가지고도 교수님별로 구분할 수 있다.
max_number : 해당 과목의 최대 수용인원
최대 수용인원별로 검색하는 기능을 추가하기 위해서 구현하였다. 최소 수용인원도 작성하려 하였지만 제대로 된 정보를 구하기가 힘들어 최대 수용인원만 구현하려고 한다.
course_type : 해당 과목이 전공과목인지 교양과목인지 확인
위의 dept_name 에서 전공과목과 교양과목을 구분할 수 있지만, 만약의 경우를 대비하여 하나의 정보를 더 추가하였다. 전공과목과 교양과목을 구분하기 위해서만 쓰이는 attribute 이다.

SECTION SCHEMA

course_id : 어떤 과목의 section 인지를 나타내는 고유번호
sec_id : section schema 의 고유번호
semester : 해당 section 과 연결된 과목이 개설되었던 학기
year : 해당 section 과 연결된 과목이 개설되었던 연도
해당 과목이 개설되었던 연도와 학기를 알아와서 연도별로 검색하는 기능을 구현할 때 활용하고, 학기별로 검색하는 기능을 구현할 때 활용할 수 있다.

TAKES SCHEMA

ID : 해당 과목을 수강한 학생의 ID
course_id : 수강한 과목에 대한 고유번호
post_id : 강의평가 게시글에 대한 고유번호
sec_id : 어떤 section 에 수강했는지에 대한 고유번호
semester : 어떤 학기에 수강했는지에 대한 고유번호
year : 어떤 연도에 수강했는지에 대한 고유번호
grade : 받은 학점

DEPARTMENT SCHEMA

dept_name : 소속된 학과
building : 소속된 학과의 대표 건물

POST SCHEMA

post_id : 해당 schema 의 게시글 번호를 나타내는 고유번호
score_pro : 전공자의 평점
scroe_ama : 비전공자의 평점
교양과목 같은 경우에는 전부 비전공자 평점으로 들어가게끔 구현할 것이다.
구현하고자 하는 기능인 전공자의 평점과 비전공자의 평점을 다르게 하여 평점의 구분화를 주기 위해서이다.
post_string : 강의평가에 대한 내용

USER SCHEMA

회원가입을 하기 위해 새로이 만든 스키마로, 회원의 이름과 정보가 담겨 있다.
usertype 에는 교수인지 학생인지 구분하는 문자열이 들어가고 그 구분에 따라서 student_id (3 번째 파라미터)인지 professor_id (4 번째 파라미터) 인지 구분한다.

(2) Physical Design

앞에서 행했던 Conceptual Design과 Logical Design을 바탕으로 Physical Design을 수행하였다. MYSQL DBMS 상에 구축하려는 Table은 다음과 같다.

Department Table

```
create table department(
        dept_name varchar(15) not null,
        building varchar(15),
        primary key(dept_name));
```

Student Table

```
create table student(
        student_id varchar(15) not null unique,
        student_name varchar(15) not null,
        dept_name varchar(15) not null,
        primary key(student_id),
        foreign key(dept_name) references department(dept_name));
```

Professor Table

```
create table professor(
        professor_id varchar(15) not null unique,
        professor_name varchar(15) not null,
        dept_name varchar(15) not null,
        primary key(professor_id),
        foreign key(dept_name) references department(dept_name));
```

Course Table

```
create table course(
        course_id varchar(15) not null unique,
        title varchar(15) not null,
        dept_name varchar(15) not null,
        credits varchar(15),
        professor_id varchar(15) not null,
        max_number int not null,
        course_type varchar(15) not null,
        primary key(course_id),
        foreign key(dept_name) references department(dept_name),
        foreign key(professor_id) references professor(professor_id));
```

Section Table

```
create table section(
        course_id varchar(15) not null,
        sec_id varchar(15) not null unique,
        semester varchar(15) not null,
        year varchar(15) not null,
        primary key(course_id, sec_id, semester, year),
        foreign key(course_id) references course(course_id));
```

Post Table

```
create table post(
        post_id varchar(15) not null unique,
        score_pro int,
        score_ama int,
        post_string varchar(2000),
        primary key(post_id));
```

Takes Table

```
create table takes(
        student_id varchar(15) not null,
        course_id varchar(15) not null,
        post_id varchar(15) not null,
        sec_id varchar(15) not null,
        semester varchar(15) not null,
        year varchar(15) not null,
        grade varchar(15) not null,
        primary key(student_id, course_id, sec_id, semester, year),
        foreign key(post_id) references post(post_id),
        foreign key(student_id) references student(student_id),
       foreign key(course_id, sec_id, semester, year) references section(course_id, sec_id, semester,
year));
```

Users Table

```
create table users(
        user_name varchar(15) not null,
        user_type varchar(15) not null,
        student_id varchar(15),
        professor_id varchar(15),
        primary key (user_name, user_type),
        foreign key(student_id) references student(student_id),
        foreign key(professor_id) references professor(professor_id));
```

후에 이용될 결과페이지에서 JOIN 쿼리는 다음과 같다.

select distinct professor_name, title, credits, max_number, (pro_sum) / (pro_count), (ama_sum) / (ama_count), year, semester from (select section.sec_id, sum(score_pro) as pro_sum, count(score_pro) as pro_count,sum(score_ama) as ama_sum, count(score_ama) as ama_count from takes join post using(post_id) join section using(sec_id) where section.sec_id = takes.sec_id group

by section.sec_id) as avg_score join (((course) join (professor) using (professor_id)) join takes using(course_id)) using(sec_id) where (((pro_sum / pro_count) >= 3 or (ama_sum / ama_count) >= 3) or pro_sum is null or ama_sum is null) and 40 <= max_number;

```
mysql> select distinct professor_name, title, credits, max_number, (pro_sum) / (pro_count), (ama_sum) / (ama_count), year, semester from (select section.sec_id, sum(score_pr
o) as pro_sum, count(score_pro) as pro_count,sum(score_ama) as ama_sum, count(score_ama) as ama_count from takes join post using(post_id) join section using(sec_id) where se
ction.sec_id = takes.sec_id group by section.sec_id) as avg_score join (((course) join (professor) using (professor_id)) join takes using(course_id)) using(sec_id) where (((
pro_sum / pro_count) >= 3 or (ama_sum / ama_count) >= 3) or pro_sum is null or ama_sum is null) and 40 <= max_number;
+--------------+------------+--------+------------+------------------------+------------------------+------+----------+
| professor_name | title      | credits | max_number | (pro_sum) / (pro_count) | (ama_sum) / (ama_count) | year | semester |
+--------------+------------+--------+------------+------------------------+------------------------+------+----------+
| Jeong        | DB         | 3      |         50 |                   NULL |                   NULL | 2017 | 1        |
| H.C.Lim      | AI         | 3      |         70 |                   NULL |                   NULL | 2016 | 1        |
| K.H.Lee      | C.A        | 3      |         60 |                 3.0000 |                   NULL | 2017 | 1        |
| Jeong        | Data System | 3      |         60 |                 4.0000 |                   NULL | 2017 | 1        |
| S.W.Jeong    | C.A        | 3      |         80 |                   NULL |                   NULL | 2016 | 2        |
| H.C.Lim      | AI         | 3      |         70 |                   NULL |                   NULL | 2017 | 1        |
| H.C.Lim      | M.L        | 3      |         70 |                   NULL |                   NULL | 2016 | 2        |
| H.C.You      | OS         | 3      |         80 |                   NULL |                   NULL | 2017 | 1        |
| Kang         | D.S        | 3      |        100 |                   NULL |                   NULL | 2017 | 1        |
| Kang         | Algorithm  | 3      |         80 |                   NULL |                   NULL | 2016 | 2        |
+--------------+------------+--------+------------+------------------------+------------------------+------+----------+
10 rows in set (0.00 sec)
```

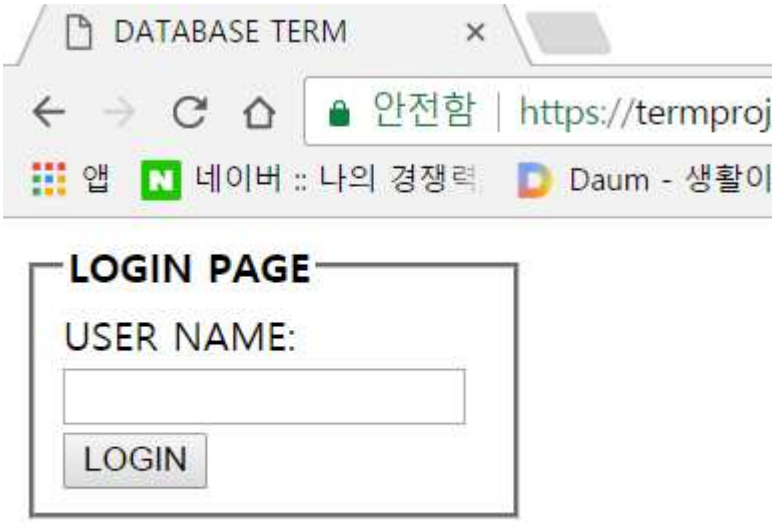
위의 그림과 같이 JOIN 쿼리가 무리 없이 나타나는 것을 확인할 수 있다.

3. Application 설명

"강의 평가 Application"은 6개의 보이는 페이지와 3개의 Hidden Page(경로를 연결하기 위하여 설정한 보이지 않는 페이지)로 이루어져 있다. 각각에 대해서는 아래에서 설명한다.

(1) 로그인 페이지

로그인을 하기 위한 페이지로 굳이 비밀번호까지 설정하지 않고 회원 이름만 가지고 로그인 할 수 있게 하였다. 세션을 이용하기 위하여 login_ok.php 에서 로그인이 되었는지의 여부를 확인한 후 이후에 로그인되어서 사용할 수 있는 모든 페이지에 접근할 때에 session 의 여부를 체크한다. 만약 제대로 설정되어 있지 않다면 즉시 로그인 페이지로 다시 돌아온다.
회원의 정보 (회원의 이름, 회원의 타입 : 교수 혹은 학생)는 항상 SESSION 변수로 저장되어 모든 페이지에서 쓰일 수 있게 하였다

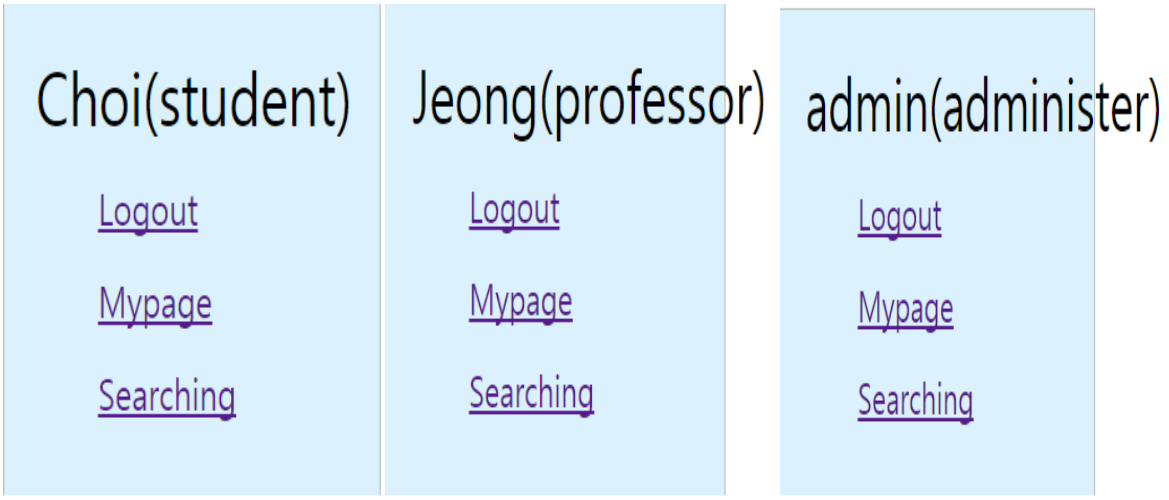


(2) Login_Ok(Hidden Page)

SESSION 을 시작하게 해주는 숨겨진 페이지로, 입력한 회원의 이름이 미리 등록되어있는 유저의 정보와 일치한지를 확인한 후에 일치한다면 session_start()를 해주고 메인 페이지로 넘어가는 역할을 한다. 만약 일치하지 않을 경우에는 alert 로 팝업창을 띄운 후 login 페이지로 되돌아가는데 그냥 되돌아갈 경우에는 미리 입력한 text 가 존재하므로 새롭게 처음으로 경로를 설정하는 방식으로 해결하였다.

(3) 헤더 페이지

로그인이 된 이후에는 항상 같은 정보를 표현하기 위해서 헤더를 include 하여 항상 왼쪽에 sidebar 로 존재한다. 이 사이드바는 모든 페이지에서 공유하는 세션의 변수인 유저의 이름과 유저의 타입이 나타나며 로그아웃, 마이페이지, 검색 버튼을 기본적으로 제공한다.
(세션은 모든 페이지에서 공유하게 만들었으므로, 중간에 어디로 넘어가든 상관이 없다.)

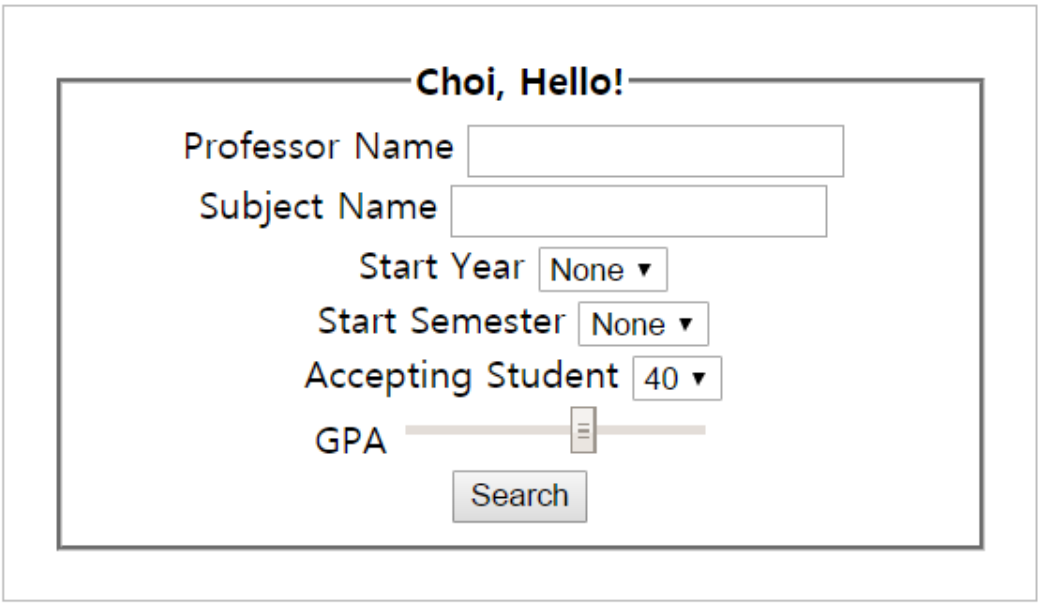


위의 그림과 같이 Login 한 User 의 Type 에 따라서 헤더페이지의 창이 다른 모습으로 나타난다.

(4) 검색 페이지

처음에 로그인을 하게 되면 헤더와 함께 가장 먼저 뜨는 검색 페이지이다. 검색 페이지에서는 교수명, 과목명, 개시연도, 개시학기, 수용인원, 평균평점으로 과목을 검색(Select Query)할 수 있다. 교수명이나 과목명을 입력하지 않는다면 해당 검색을 사용하지 않는다고 간주하고 검색하며, 개시연도와 개시학기는 select option 으로 구현하였는데, 기본적으로 미입력상태에 있게 하였다. 미입력상태에 있는 경우에는 개시연도와 개시학기를 무시하고 나머지 정보를 가지고 검색한다. 수용인원은 기본적으로 40 의 값을 가지며 여기서 의미하는 수용인원은 최소수용인원을 의미한다. 평균평점은 최소 평균 평점을 의미하고, range type 으로 구현하였으며 기본적으로 3 의 값을 두게 하였다.

다른 페이지에서도 마찬가지겠지만 항상 헤더파일이 왼쪽에 존재하므로, 어디든지 이동할 수 있고 언제든지 로그아웃 할 수 있다.



(5) 결과 페이지

Next.php 로 저장되어 있으며, 검색페이지의 검색 결과를 보여주는 페이지이다. 교수명, 과목명, 학점, 최대수용인원, 전공평점, 비전공평점, 개시연도, 개시학기를 표시하고 있으며 테이블 마지막의 '평가' 버튼을 누르면 해당 과목의 평가 페이지로 넘어간다. 이는 form 의 hidden type 을 이용하여서 변수를 다음 페이지로 전달하였다.

추가로 전공 평점과 비전공 평점을 구분해서 구현하는 바람에 최소평점으로 검색하는 기능에 애매함이 생기게 되었는데 이를 전공 평점이나 비전공 평점이 하나라도 존재하지 않는 경우에는 무조건 검색이 가능하게 하였다. 만약 두 평점 모두 존재할 경우에는 하나만 최소평점을 넘어도 검색되게 구현하였다.

Select Query 를 이용하면서 Join 을 주로 이용하였다.

Professor	Subject	Credits	Available	Major GPA	Non-major GPA	Start Year	Start Semester	
Jeong	DB	3	50	5.0000		2017	1	Evaluation
K.H.Lee	C.A	3	60	3.0000		2017	1	Evaluation
Jeong	Data System	3	60	4.0000		2017	1	Evaluation
S.W.Jeong	C.A	3	80			2016	2	Evaluation
H.C.Lim	AI	3	70			2017	1	Evaluation
H.C.Lim	M.L	3	70			2016	2	Evaluation
H.C.You	OS	3	80			2017	1	Evaluation
Kang	D.S	3	100			2017	1	Evaluation
Kang	Algorithm	3	80			2016	2	Evaluation
H.C.Lim	AI	3	70			2016	1	Evaluation

(6) 평가 페이지

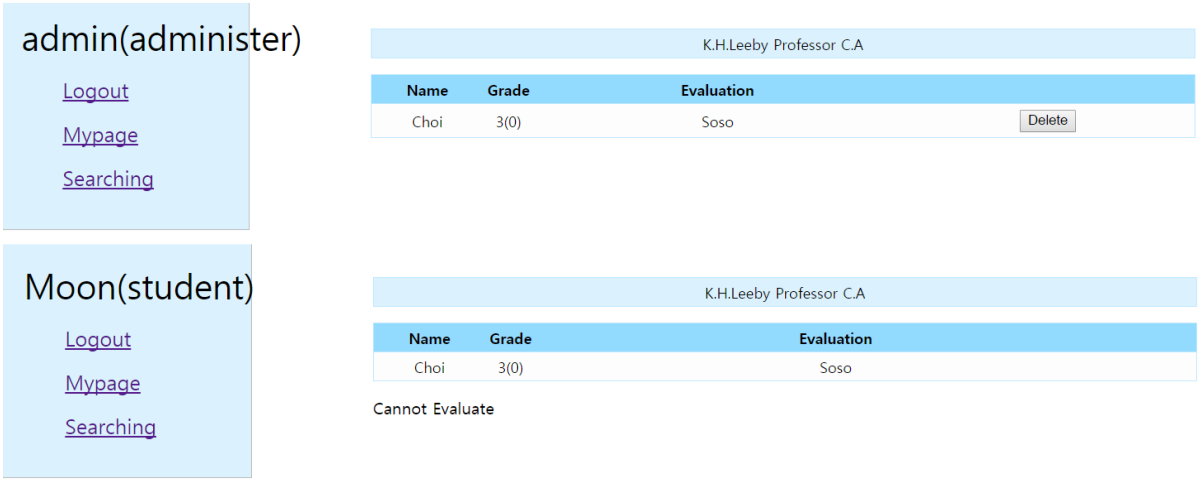
결과 페이지나 마이페이지에서 이동되는 페이지로 해당 과목의 평가내역을 보여준다. 위의 테이블은 해당 과목에 대해서 다른 사람이 어떻게 평가했는지 먼저 보여주고, 자신의 평가를 제일 밑에 위치시켰다. 만약 재평가를 하고 싶을 경우에는 새롭게 평점과 평가내용을 입력하고 평가 버튼을 누르면 자동으로 재평가가 이루어진다. 만약 이 과목을 수강하지 않았을 경우에는 'Cannot Evaluate' 메시지를 뜨게 하고 다른 사람들의 평가내역만 확인할 수 있다.

추가적으로 운영자가 평가 페이지를 들어갔을 때에는 모든 평가들에 대해 삭제할 수 있다.

K.H.Leeby Professor C.A

Name	Grade	Evaluation
Choi	3(0)	Soso

3	Soso	
Grade 5 ▾		Evaluation

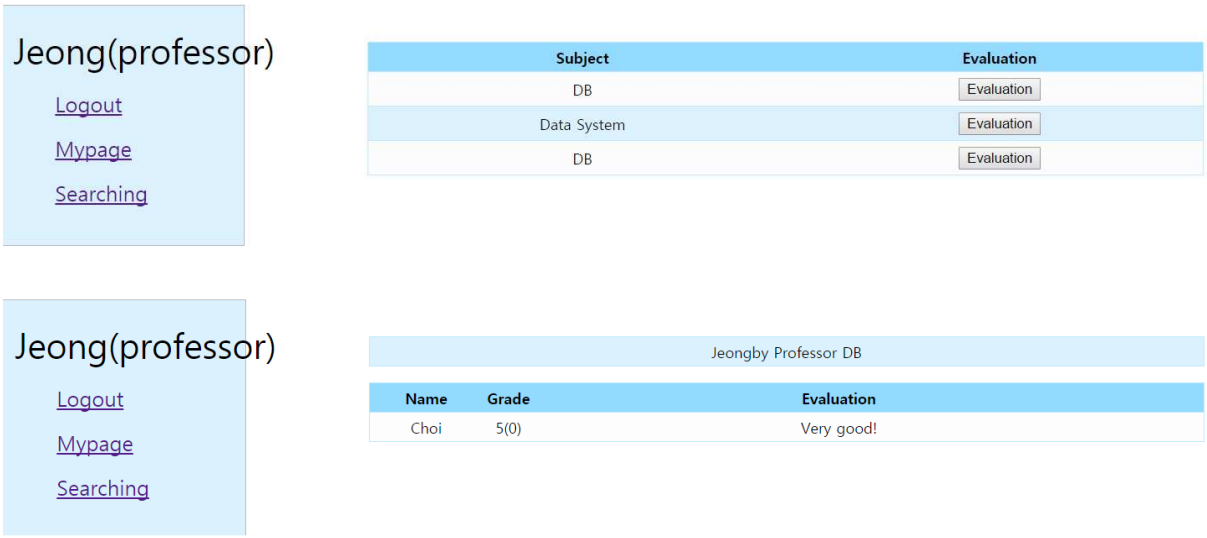


(7) 마이페이지

마이페이지의 경우에는 교수와 학생을 다르게 구현하였는데, 먼저 학생의 경우에는 자신이 수강한 과목들의 목록을 전부 불러왔다. 그에 대해서 평가버튼을 만들어 바로 평가할 수 있게 만들었으며, 자신이 평가한 내용도 한번에 볼 수 있게 구현하였다.

교수의 경우에는 자신이 강의했던 과목들을 표시하는데, 평가보기 버튼을 클릭하면 수강생들에 대한 전체적인 평가를 볼 수 있다. (이는 평가페이지와 같은 페이지이나 평가하는 부분을 뜨지 않게 처리하고 구현하였다.)

추가적으로 운영자의 경우에는 마이 페이지에서 과목, 학과, 교수, 학생 등을 추가할 수 있는 기능을 구현하였다.



Choi(student)

Logout

Mypage

Searching

Subject	Year-Semester	Grade	Evaluation	
DB	2017-1	A+	Very good!	Evaluation
C.A	2017-1	A	Soso	Evaluation
Data System	2017-1	B+	Thanks	Evaluation
C.A	2016-2	A+		Evaluation
AI	2017-1	A		Evaluation
M.L	2016-2	B+		Evaluation
OS	2017-1	A+		Evaluation
D.S	2017-1	C+		Evaluation
Algorithm	2016-2	C		Evaluation

Choi(student)

Logout

Mypage

Searching

H.C.Limby Professor AI

Name	Grade	Evaluation

Grade 5

Evaluation

admin(administer)

Logout

Mypage

Searching

Add Department

Department Name :

Department Building :

Addition

Add Students

Name :

Department : Business

Add

Add Professor

Name :

Department : Business

Add

Add Subject

Department : Business

Professor Name : H.C.Lim

Subject :

Grade : 1

Available :

Type : major

Add

(8) 기타 페이지

Login_Ok.php 와 같이 세션을 종료하는 Logout.php 와 평가를 알맞게 했는지 확인하는 Score_Ok.php 도 숨겨진 페이지로 구현되어 있다.
이는 확인만 하고 다른 경로로 바로 연결된다.
Score_Delete.php 와 Administer.php 에서 관리자의 추가 기능과 평가 삭제 기능을 확인한다.

(9) 기타 특이사항

수강을 한 강의에 대해서만 평가할 수 있게 구현하고자 TAKES 라는 스키마를 사용하여 자신이 수강을 했는지의 여부를 확인하였다. '회원가입'이나 별도의 페이지를 통해 자신이 수강하였는지 받아오는 방법도 있지만 그것 역시 선택적이라서 마음만 먹으면 자신이 수강하지 않은 과목이어도 수강했다고 하고 평가할 수 있다. 이를 막기 위해서 수강신청 사이트에서 직접 파싱해오는 방법이 있는데, 이 부분까지는 지금까지의 지식으로는 해결이 불가능하다고 판단하였다. 따라 dump 파일을 이용하여 직접적으로 입력해주는 방법으로 전체적인 강의 평가 사이트를 만들었다. 회원가입 기능은 dump 파일에서 직접 입력하거나 Aminister 가 Mypage 에서 회원가입을 하는 방식으로 구현하였다. 이러한 방법이면 새롭게 유저가 추가된다면, 아무 과목도 수강하지 않은 상태로 추가하는 방법밖에 없는데 (혹은 위와 같은 방법으로 입력받거나) 이는 전체적으로 문제가 된다. 이를 해결하는 데에는 전체적인 스키마를 수정해야 해서 일단은 계획한 스키마로 페이지가 제대로 구성되게 구현하였다. 누가 무슨 과목을 수강했는지의 정보가 없는 상태에서는 입력받은 정보로만 검색, 평가를 할 수 있으므로 내가 무슨 정보를 제대로 입력했는지에 따라서 수강신청 사이트가 작동한다.

4. Conclusion

PHP와 MYSQL을 이용하여 웹 기반의 "강의평가 Application"을 구현하였다. 이 Application은 회원가입 기능, Login / Logout 기능, 강의 평가 기능, 강의 검색 기능, 강의 평가 보기 기능 등이 구현되어있다. 또한 User Type이 관리자일 경우에는 학생들이 작성한 평가를 삭제하는 것도 가능하다. 웹 상에서 DBMS와 연동되어 있기 때문에 관리자는 학생과 교수, 학과와 강의를 추가하는 것도 가능하다.

웹에 따로 서버를 구축함에 따라서, Data의 수정이 필요하고 전체적인 Entity의 수정이 필요할 때 수정해야 되는 것은 PHP 문구가 아닌 MYSQL의 DBMS였다. 이 점을 통해 DBMS 사용의 편리성을 크게 느낄 수 있었다.

이번 Project를 통해서는 Github과 C9의 장점에 대해서도 잘 익힐 수 있었는 데, 협업에 정말 유용한 Utility라고 생각했다.

이번 구현에 있어서의 아쉬움은 좀 더 Design부분에 있어서 더 신경을 써서, 회원가입과 같은 조금은 어색하게 구현된 기능들을 더 발전시키지 못한 점이다. 이번 Application 구현에 있어서, 주제가 '강의평가'이다 보니 회원가입이 전체적인 Application에 미치는 영향이 적었다. 회원가입을 하더라도 수강한 강의가 없어서 강의평가가 불가능 하기 때문이다. 만약, Entity와 Relation을 Design하는 부분에 있어서 이 점을 더 고려하였다면 훌륭한 Application이 되었을 것이라고 믿어 의심치 않는다.