**인터넷 프로그래밍 프로젝트**

컴퓨터 공학과 3학년

12163786 장현수

12131606 조성철

1. **프로젝트 주제**

인하대학교 학생들을 위한 질문 답변 사이트이다.

1. **Client**

회원들은 각 회원이 수강하고 있는 과목마다 질의 응답을 할 수 있다. 질문에 답변을 할 때마다 포인트가 주어지며 main 페이지에서 모든 회원들의 point 순위가 주어진다. Login, 회원가입, My Page(회원 정보 수정), 탈퇴 기능이 가능하다.

1. **Server, DB**

DB에서 테이블을 크게 5개를 사용했다.

각각 Login\_Data, course\_data, user\_course, user\_profile, del\_user 테이블인데, 다음의 의미를 가진다.

1. Login\_Data : 사용자의 id와 password를 관리하는 데이터

* 사용자의 id를 키값으로 하여 각각 user\_id, user\_pw 필드에 저장한다.

1. course\_data : 과목 별로 등록된 질문과 답변, 과목 이름, 수업 날짜, 강의하시는 교수님의 정보를 저장한다.

* l0, l1, l2의 필드는 각각 “인터넷 프로그래밍”, “유닉스 시스템 프로그래밍”, “임베디드 시스템 프로그래밍” 과목을 의미한다.
* Qus은 과목에 등록된 질문들의 테이블, Ans는 과목에 등록된 답변들의 테이블이다.

1. user\_course : 사용자 마다 수강하는 과목을 관리하는 테이블이다.

* 사용자의 id를 키 값으로 하여 l0, l1, l2 필드에 수강하는 과목을 채웠고 해당하는 과목을 수강하지 않는다면 “None”의 값을 가지게 했다.

1. user\_profile : 사용자의 개인정보를 관리하는 테이블이다.

* 사용자의 id를 키값으로 하여 email, grade(학년), id(사용자의 id), major(전공), name, point(질문에 대한 답변을 하면 그에 따른 포인트를 획득한다), student\_id(학번)의 필드를 가지는 테이블이다.

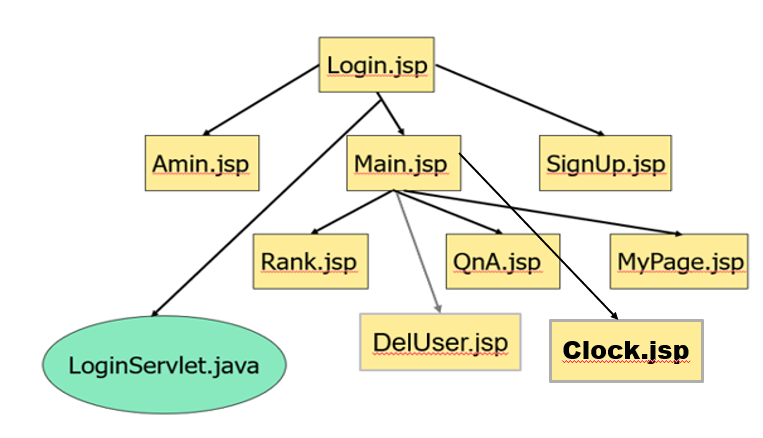
1. del\_user : 탈퇴한 회원들의 정보와 탈퇴 사유를 관리하는 테이블이다.

* 탈퇴한 사용자의 id를 key value로 사용하며, 사용자가 탈퇴한 날짜와 탈퇴한 사유를 저장하여 관리자에게 유용한 정보를 제공한다.

1. 관리자

관리자 페이지로 가는 비밀번호 설정을 위해 javascript에 내장되어 있는 Date 객체를 이용한다. 관리자 페이지로 가는 비밀번호는 접속 날짜의 년도 + 달 + 일을 더한 값이다. 관리자 페이지에서는 회원 정보를 출력하고 회원 아이디를 기반으로 회원을 탈퇴 시킬 수 있도록 한다.

1. **프로젝트 flow chart**



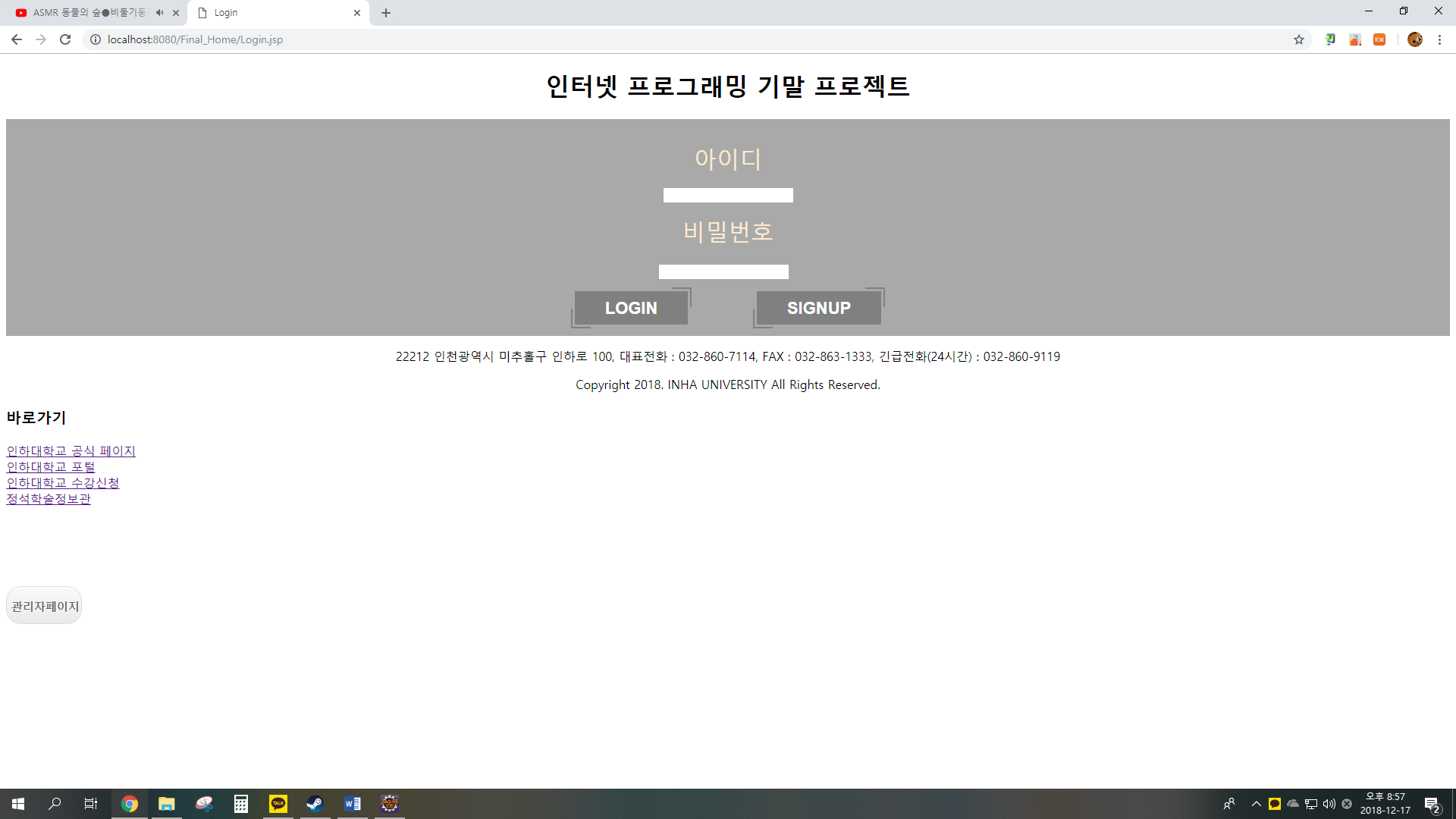
1. **프로젝트 page별 기능 및 코드 설명**
2. **Login.jsp**

**● 개요**

사이트를 처음 접속했을 때 사용자가 접하는 페이지이다. 사용자의 ID와 PW를 입력 받아 로그인을 시도할 수 있다. 또한 회원 정보가 없는 회원은 SIGNUP 버튼을 통해 회원가입을 할 수 있다. 관리자는 관리자 페이지로 이동 할 수 있다. 인하대학교 학생들을 위한 페이지인 만큼 인하대의 주요 사이트로 연결되는 바로가기가 존재한다.

* **기능**
* **로그인 정보를 확인**해 Main 페이지로 넘겨준다. 사용자에게 입력 받은 id와 Login\_Data table을 통해 얻어온 pw 값을 비교해 로그인 성공 여부를 판단한다. 만약 로그인이 성공하였다면 Main.jsp로 이동하고 실패하였다면 alert를 띄워준다.
* **회원 가입이 가능하다**. 사용자는 아이디, 비밀번호, 학년, 수강과목 등의 정보를 입력해 사이트의 회원이 될 수 있다.
* **관리자 페이지로의 진입이 가능하다.** 관리자 페이지로 진입하기 위해선 비밀번호를 입력해야 한다. 이 비밀번호는 매일 변경이 되는데, 그 날의 비밀번호는 현재 년도 + 월 + 일이다. 이를 위해 javascript 내장 객체인 Date객체를 이용한다
* **인하대학교 주요 사이트로의 바로가기**가 제공된다. 인하대학교 공식 홈페이지, 인하대학교 포털, 인하대학교 수강신청 사이트, 정석학술 정보관 사이트로의 이동이 가능하다

**● 실행화면**

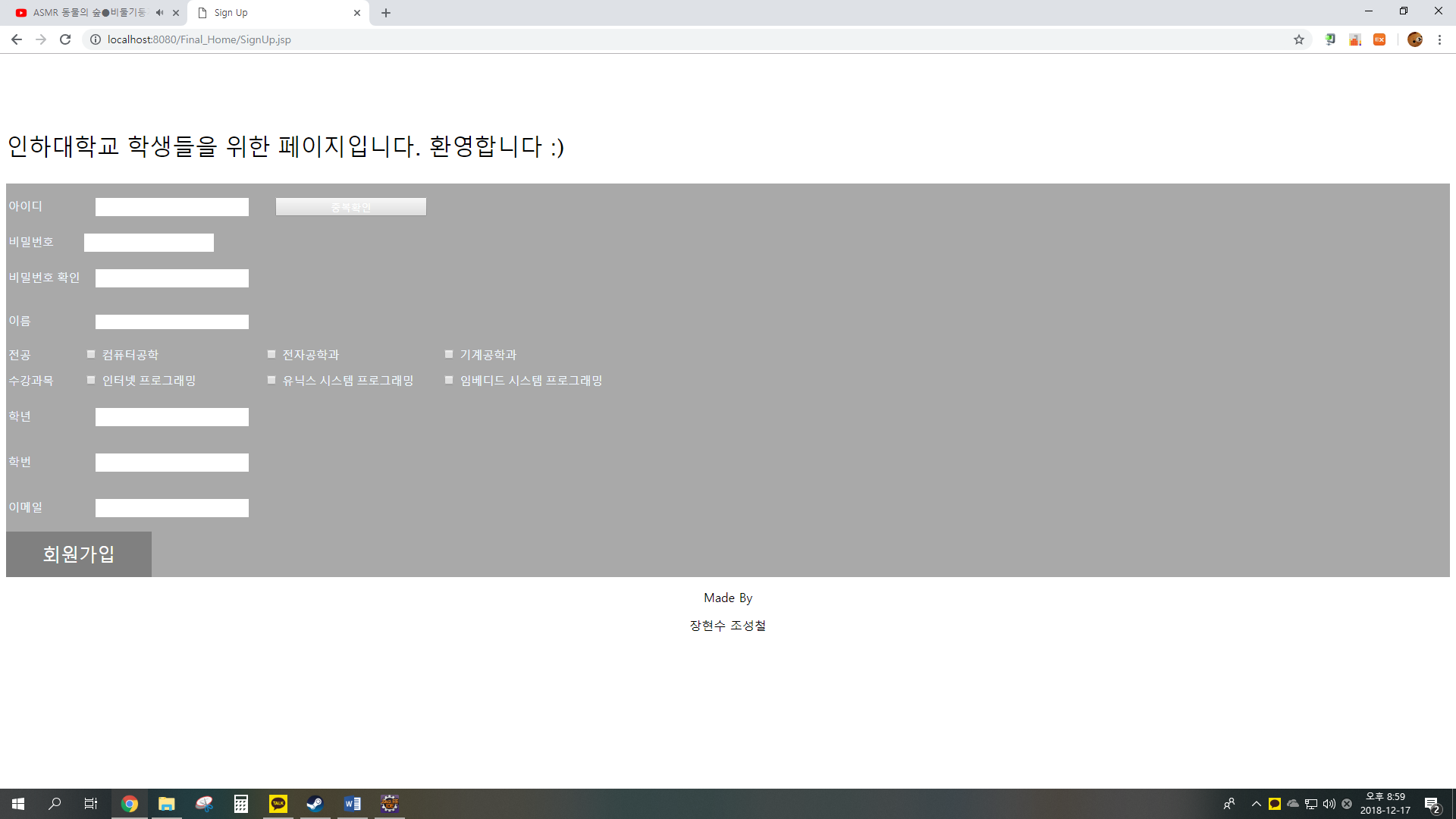


1. **SignUp.jsp**

* **개요**

현재 사이트의 회원이 아닌 사람이 사이트의 회원이 될 수 있도록 한다. 입력 받은 정보를 database에 저장해 회원으로써 관리될 수 있도록 한다.

* **기능**
* **회원 가입 기능.** 사용자는 아이디, 비밀번호, 학년, 수강과목 등의 정보를 입력해 사이트의 회원이 될 수 있다. 입력 받은 정보 중 아이디는 중복확인을 진행한다. 중복 확인은 Login\_Data의 user\_id중 입력받은 id와 같은 값이 없는지 확인 하여 구현한다. 비밀 번호는 중요 정보이기 때문에 사용자에게 입력을 2번 받고 서로 일치하는지 확인한다. 모든 정보가 입력되면 입력 받은 값을 읽어와 user\_profile table에 id를 키 값으로 저장한다.
* **실행 화면**



1. **Admin.jsp**

* **개요**

사이트를 관리하는 관리자를 위한 페이지이다. 현재 사이트를 이용중인 회원의 목록을 볼 수 있다. 또한 사이트를 탈퇴한 회원의 목록, 탈퇴한 시간(년월일)과 탈퇴 사유를 볼 수 있다. 회원 탈퇴 기능을 제공해 관리자가 특정 회원을 탈퇴 시킬 수 있도록 한다.

* **기능**
* **사이트를 이용 중인 모든 회원의 정보를 출력한다.** (allUserData() 함수 이용) 함수의 구현 방법은 다음과 같다.

우선 database의 Login\_Data table 참조해 그 table안의 모든 id값을 가져온다. 가져온 id 값 각각을 이용해 해당 id를 가진 회원의 정보를 출력하는 allUserProfile 함수를 호출한다. allUserPorfile함수는 동적으로 table(id = infotable)에 정보를 출력하는 함수이다. 이 table에 field를 추가하기 위해 tr 객체를 새로 만들어 준다.(createElement 함수 이용) Id를 이용해 user\_profile table에서 아이디, 이름, 학년, 전공, 학번, 포인트 정보를 읽어온다. 읽어온 레코드의 필드 각각을 new Element(td)로 만들어 준다. 만들어준 모든 td를 미리 만들어 놓은 tr에 추가한 이후 최종 tr을 table에 추가한다. 이로써 동적으로 회원정보가 table에 추가되게 된다.

* **이제까지 탈퇴한 모든 회원의 정보를 출력해 준다**. (함수 allDelData()이용)

우선 database의 Del\_User table을 참조해 그 안의 모든 child reference를 이용해 탈퇴한 모든 회원의 id값을 가져온다. 가져온 각각의 id 값에 대해 출력함수 allDelProfile를 호출한다. allDelProfile 함수는 동적으로 table( id = deltable)에 정보를 출력한다. 이 table에 동적으로 값을 추가하기 위해 tr 객체를 하나 만들어 준다.(createElement 함수 이용) 그 후 id를 이용해 Del\_User Table에서 탈퇴 일자, 탈퇴 사유 정보를 읽어온다. 읽어온 정보를 이용해 td객체를 만들어 준다. 만든 td 객체 각각을 미리 만들어 놓은 tr 객체에 추가하고 최종 tr을 table에 추가한다. 이로써 동적으로 탈퇴한 회원의 정보가 table에 추가되게 된다.

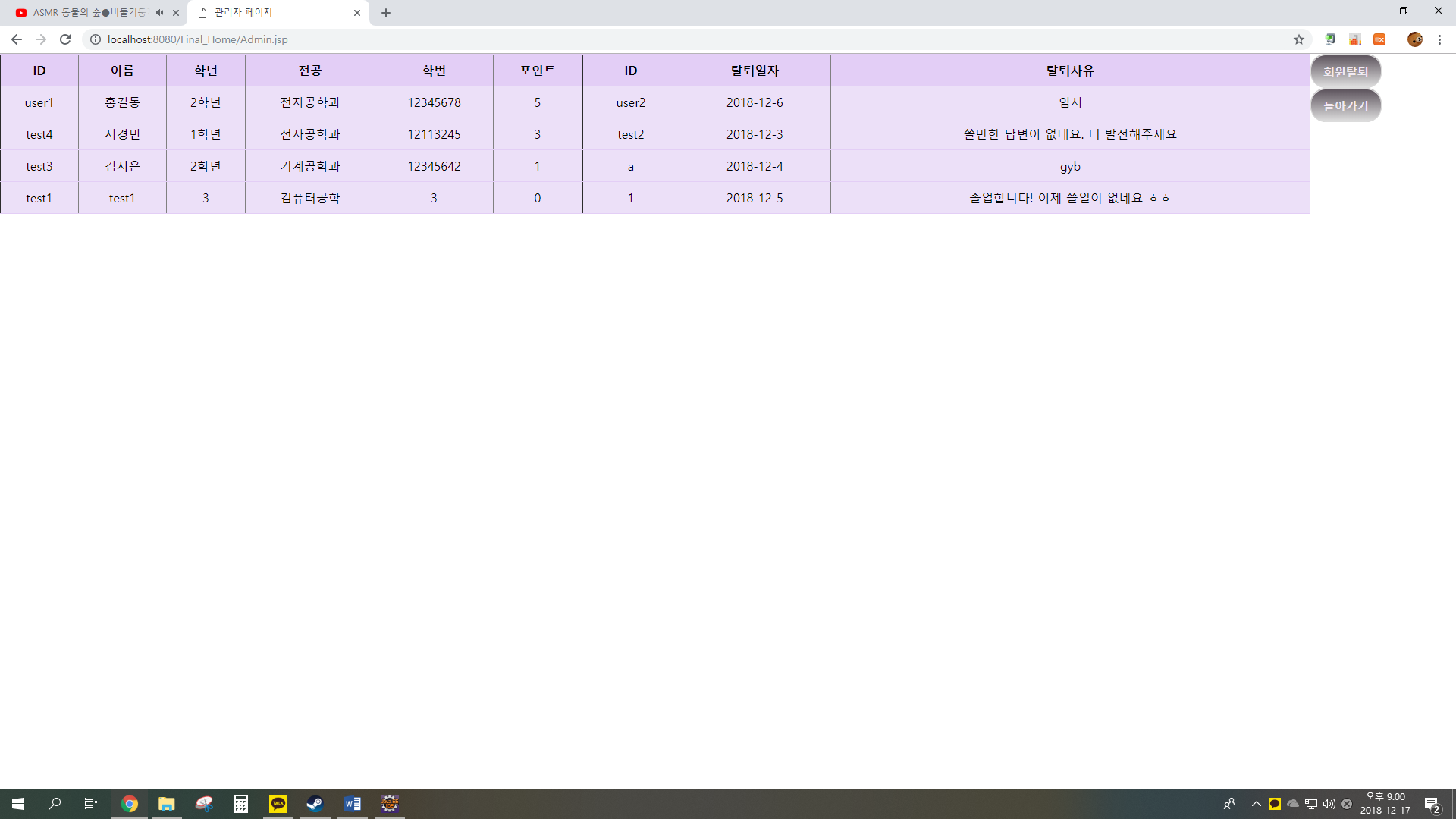
* **특정 User를 탈퇴 시킬 수 있다.**

회원 탈퇴 버튼을 누르게 되면 prompt로 탈퇴 시킬 회원의 아이디를 입력받는다. 이 입력 받은 아이디를 key값으로 이용해 user\_course, user\_profile, Login\_Data table의 레코드를 삭제한다.

(관리자가 삭제한 유저는 Del\_User 테이블에 추가 되지 않는다.)

* **관리자 페이지에서 나가 Login Page로 돌아갈 수 있다**. 이는 단순히 location.href 함수를 이용해 구현하도록 한다.

**● 실행 화면**



1. **Main.jsp**

* 개요

이 페이지의 상단부에는 현재 로그인 한 사용자의 정보를 볼 수 있는 My page, 회원에서 탈퇴하는 페이지로 연결하는 버튼과 로그아웃 하는 버튼이 있다.

페이지의 좌측에는 로그인한 사용자가 수강하고 있는 과목을 list로 보여준다. 각 과목명은 하이퍼링크로서 작동하며 해당 과목의 QnA 페이지로 넘어갈 수 있다.

페이지의 우측 중앙부에는 사용자들의 포인트로 순위를 표현한 Rank.jsp 파일을 보여준다.

* Login.jsp로부터 url을 통해 user\_id를 파라미터로 받는다. 그 id로 firebase의 테이블들을 참조한다.
* 데이터 베이스 참조 변수

var user\_course = firebase.database().ref("user\_course/"+id);

var user\_profile = firebase.database().ref('user\_profile/' + id);

var user\_data = firebase.database().ref('Login\_Data/' + id);

* 기능
* **My Page 버튼**

onclick 이벤트 리스너를 이용하여 mypage() 함수를 호출하는데 이 함수는 id 가 form1인 form 객체를 불러와서 action 필드를 ./LoginServlet ?id=사용자의 아이디 형태로 설정한 후 form.submit()을 통해 MyPage.jsp로 페이지를 넘겼다. 즉 Servlet을 통해 파라미터를 전송하여 원하는 기능을 구현했다.

* **탈퇴 버튼**

DelUser.jsp로 넘어가 회원을 탈퇴시킨다.

* **로그 아웃 버튼**

onclick 이벤트 리스너로 logout() 함수를 실행했다. 이 함수는 단순히 location.href = “Login.jsp”를 실행하여 로그 아웃을 수행한다.

* **id가 course\_list인 details 태그**

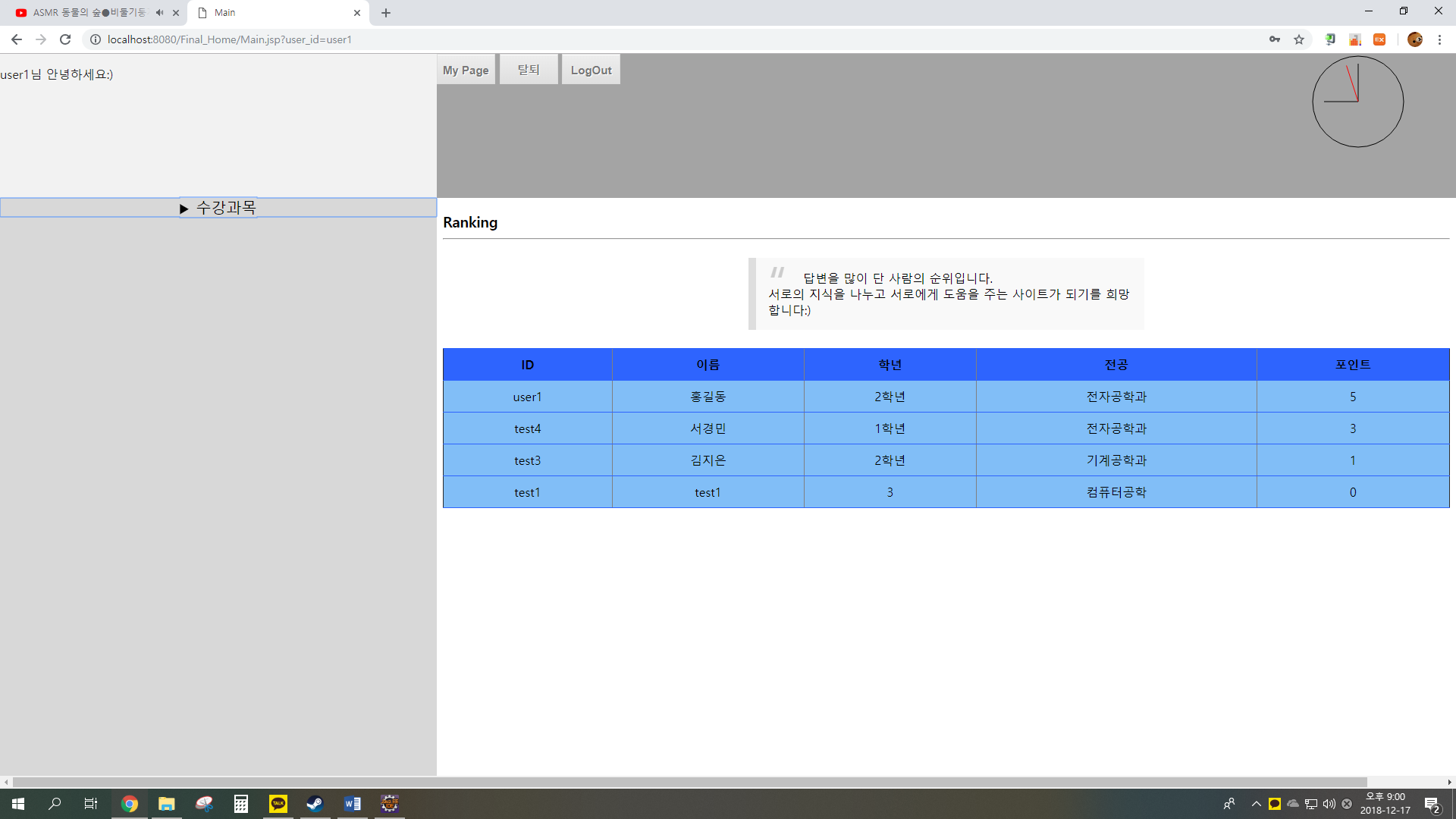
user\_course.once(‘value’ , function(snapshot) {} ); 은 data 라는 배열에 각 과목의 수강 여부를 저장한다.

onclick 이벤트 리스너로 mylist(this) 함수를 실행한다. 이 함수는 사용자가 수강한 과목을 <ul> 에 추가하고 없애는 기능을 수행한다.

id가 course\_list인 객체를 가져와서 myul이라 할 때, myul의 자식 노드들의 개수가 5개 초과 일 때는 이미 리스트에 <li>가 추가되어 있다는 의미이므로 그것들을 삭제해준다.

data 배열을 순회하면서 data[i]의 값이 None이 아니라면 <a href = “./QnA.jsp?course\_id=[과목의 key값] && user\_id = [사용자의 아이디]”>과목이름</a> 태그를 가지고 있는 새로운 <li> 객체를 생성하여 myul에 appendChild를 이용해 추가한다.

* JavaScript Canvas를 이용해 만든 시계 그림을 Main Page에서 포함한다.(Clock.jsp) 자세한 모양 및 디자인은 실행화면에서 확인할 수 있다.

**●실행 화면** 

1. **LoginServlet.java**

* 개요

Main.jsp에서 My Page 버튼을 누르면 Servlet 파일을 통해 MyPage.jsp로 파라미터를 전송한다.

* **doGet 함수**

String id = request.getParameter(“id”); 🡪 id 파라미터를 넘겨받는다.

request.getSession().setAttribute(“id”, id); 🡪 세션을 생성하여 세션의 id 속성을 변수 id 값으로 설정한다.

request.getRequestDispather(“MyPage.jsp”).forward(request, response);

* MyPage.jsp에 request와 response 객체를 넘긴다.

1. **MyPage.jsp**

<% String ID = (String)request.getSession().getAttribute(“id”); %>

🡪LoginServlet.java에서 세션의 id 속성 값으로 전달한 파라미터 값을 가져온다.

* 개요

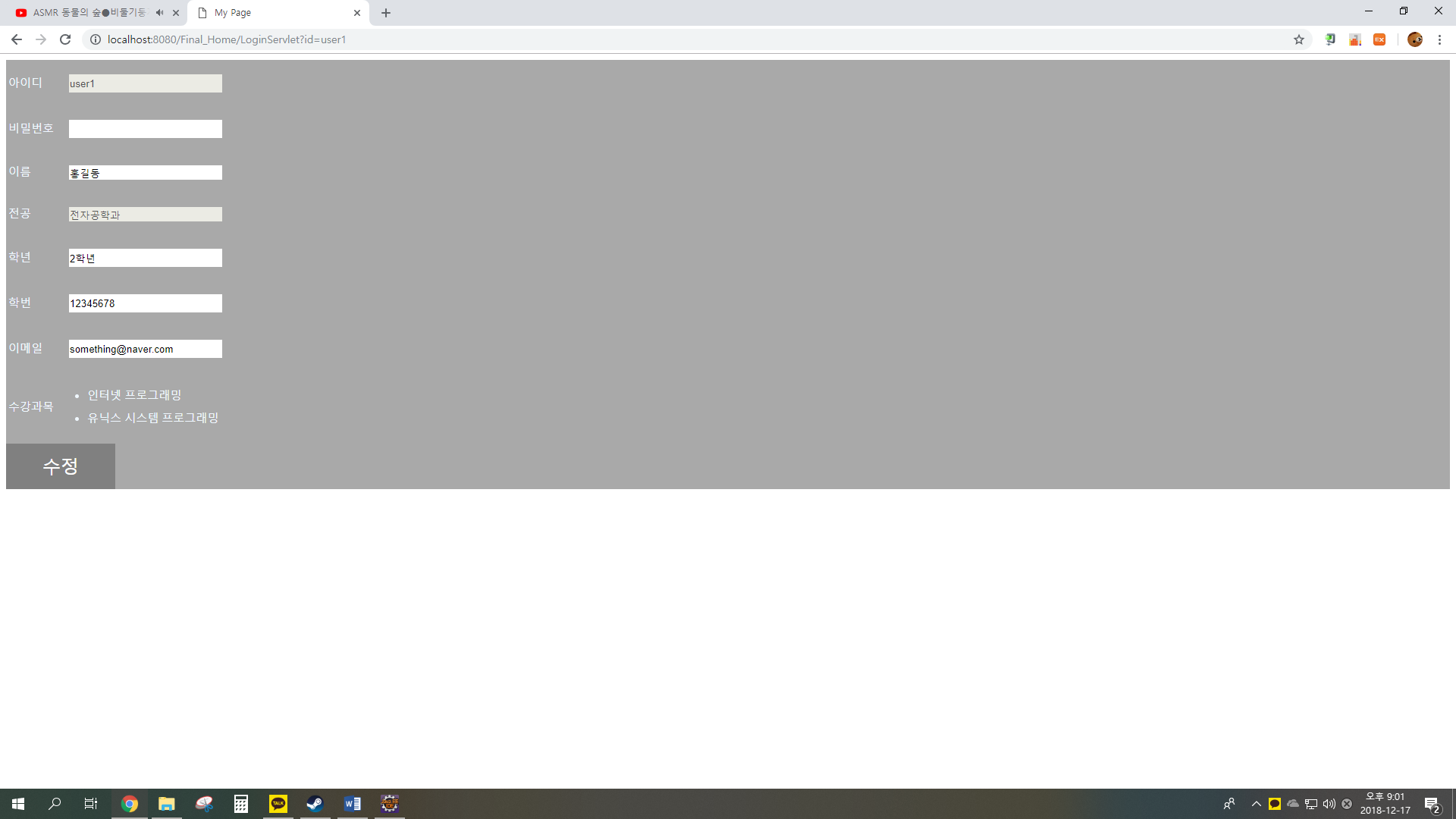
데이터 베이스의 user\_profile 테이블에서 사용자의 정보를 읽어와 화면에 출력한다.

‘수정’ 버튼을 클릭하면 현재 <input type=”text”> 태그에 써 넣은 정보로 user\_profile 데이터를 업데이트 한다.

* 기능
* **수정 버튼**

onclick 이벤트 리스너로 Setting() 함수를 호출한다. 이 함수는 각각의 사용자 정보를 받아서 user\_profile 테이블과 Login\_Data 테이블을 갱신한다.

* id와 전공학과에 대해선 수정할 수 없도록 disabled=true로 설정했다.
* 실행 화면



1. **Rank.jsp**

* **개요**

사이트를 이용중인 모든 회원의 포인트 순위를 출력한다.

* **기능**

User-Profile 테이블을 이용해 모든 사용자의 정보와 포인트를 읽어온다. 이 때 얻어온 모든 포인트의 값을 배열에 저장한 후, 내림 차순으로 정렬한다.

showRanking함수는 위에서 정렬한 point 배열을 받아 그 포인트에 해당하는 사용자를 테이블에 출력해준다. 이 때 동일 point에 대한 사용자는 한번 씩만 출력되어야 하므로 전의 값과 비교해 같지 않은 경우에만 출력해 준다.

해당 포인트에 대한 유저 정보를 출력해주는 함수는 show(value) 함수이다. 이 함수는 매개변수로 전달 받은 point 값에 대하여 User\_profile을 탐색하며 point가 일치하는 유저의 정보를 테이블에 출력해준다. Admin 페이지에서 사용한 방법과 마찬가지로 동적으로 tr, td 등의 element를 만들며 각각의 element에 속성과 값을 배정한다. 그 후 table에 append 해준다.

* 실행 화면
* **Main Page에서 포함되어 있다.**
* 

1. **DelUser.jsp**

* **개요**

회원이 사이트를 탈퇴하려고 할 때 방문하게 되는 페이지이다. 사용자에게 비밀번호를 입력 받아 일치 여부를 판단해 잘못된 페이지 탈퇴를 막고, 탈퇴사유와 탈퇴 시간을 입력 받아 관리자에게 유용한 정보를 제공한다.

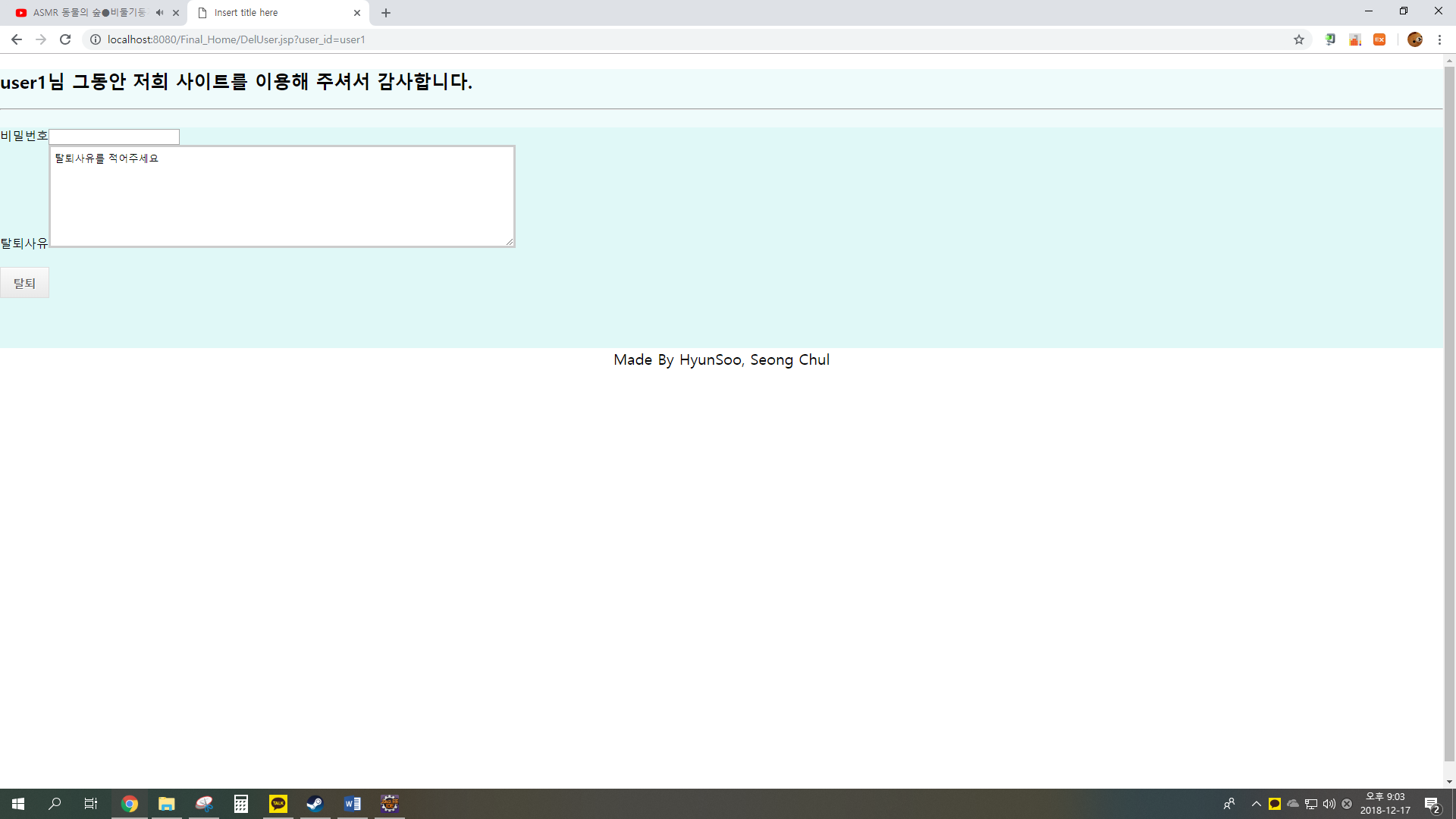
* **기능**

사용자가 ‘회원 탈퇴’ 버튼을 누르면 우선 올바른 비밀번호를 입력 했는지 판단한다. Main Page에서 jsp to jsp로 넘겨받은 id값을 키로 이용해 Login\_Data에서 비밀번호 값을 찾는다. 만약 올바는 비밀번호라면 회원탈퇴를 진행하고 틀렸다면 에러메시지를 띄운다.

올바는 비밀번호일 경우, 회원 탈퇴를 진행한다. 이 때 우리는 textarea에 입력된 회원 탈퇴 사유를 Del\_User 테이블의 reason 필드에 입력한다. 이 때 입력하는 데이터의 키 값은 id를 이용한다. 그 후 내장되어 있는 Date 객체를 이용해 탈퇴 날짜를 yyyy-mm-dd 형식의 String으로 만든 후 이를 Del\_User 테이블의 time 필드에 입력한다.

회원을 실제로 탈퇴시키는 것은 doremove함수에서 이루어진다. Id를 매개변수로 입력 받아 그 값을 키로 이용해 user\_coures, user\_profile, Login\_Data에서 레코드를 삭제한다. 모든 과정이 완료되면 페이지는 Login.jsp로 이동한다..

* 실행 화면



1. **QnA.jsp**

* Database 참조

var course\_data = firebase.database().ref("course\_data/"+course\_id);

var course\_ques = firebase.database().ref("course\_data/"+course\_id+"/Qus");

var course\_ans = firebase.database().ref("course\_data/"+course\_id+"/Ans");

* 개요

Main.jsp에서 수강과목 <details>에서 각 목록의 과목 이름 하이퍼 링크를 클릭할 때 과목의 키값(course\_id)과 사용자의 id (user\_id)를 url의 파라미터로 넘겨 받아서 넘겨 받은 과목의 키 값에 따라 과목의 정보를 출력하고 그 과목에 관련된 질문과 답변을 화면에 출력한다.

* 기능
* course\_data.once(‘value’, function(snapshot) {}) 을 통해 **과목의 정보를 화면에 출력한다.**
* **course\_ques.once(‘value’, function(snapshot){} ) 구문**
* qlist에 “Ques”를 id로 가지는 <ul> 객체를 가져온다.
* snapshot.forEach ( function ( childSnapshot ) {} )는 현재 과목의 Qus 테이블의 field를 전부 읽으며 count 변수로 개수를 세고 답변이 되지 않은 질문이라면 ‘답변하기’ 버튼을 추가 한 후, qlist에 <li> 객체를 추가한다. 답변이 된 질문이라면 버튼을 추가하지 않고 <li> 객체를 qlist에 추가한다.
* **‘답변하기’ 버튼**

“q+질문 번호”를 id로 가진다.

onclick 이벤트 리스너로 answer(“답변하는 질문”, 질문 번호, 답변을 갱신할 firebase의 경로)를 설정한다.

* **course\_ans.once(‘value’, function(snapshot) {} ) 구문**
* alist에 “Ans”를 id로 가지는 <ul> 객체를 가져온다.
* snapshot.forEach( function(childSnapshot) {} ) 는 현재 과목의 Ans 테이블의 field를 전부 읽으며 alist에 <li> 객체를 추가한다.
* **‘질문하기’ 버튼**

onclick 이벤트 리스너로 makeQuestion() 함수를 호출한다.

이 함수는 사용자에게 prompt를 실행하여 질문을 입력 받고 현재 과목의 Qus 테이블에 등록한다. 이 때 Qus 테이블은 질문 번호(count)를 key값으로 갖고 isAns와 question을 필드로 갖는다.

isAns는 이 질문이 답변이 되었는지에 대한 true/false 값을 가지고 question은 질문 문자열을 가지고 있다.

Qus 테이블을 갱신하면서 현재 출력되고 있는 <ul>로 같이 갱신한다.

* **answerQ(question, num, a\_path) 함수**

‘답변하기’ 버튼을 눌렀을 때 실행되는 함수이다. 우선 답변을 받았으므로 답변하기 버튼을 삭제하고 prompt를 통해 답변을 사용자로부터 입력 받는다. 답변이 null값이 아니라면 user\_profile 테이블의 point 필드의 값을 1 증가시키고 Ans테이블의 answer 필드를 입력 받은 답변으로 갱신한다.

* 실행 화면



1. **사용된 CSS오픈 소스**

* <https://nanati.me/css-button-design/> (Singup.jsp, Login.jsp, DelUser.jsp)
* <http://www.cssbuttongenerator.com/> (대부분의 button에서 사용)
* <https://cssnewbie.com/8-awesome-css-textarea-snippets/#.XBeOmVwzaUk> (DelUser.jsp 에서 회원 탈퇴 사유를 입력 받는 textArea에 사용)
* <https://1stwebdesigner.com/css3-text-effects-typography/> (Rank.jsp 위의 문구 출력 <p>태그에 사용)