REPORT

**기말 프로젝트**

**치킨 리뷰 커뮤니티**



|  |  |
| --- | --- |
| 과목명 | 웹 프로그래밍 |
| 담당교수 | 소선섭 교수님 |
| 전공 | 컴퓨터공학부 컴퓨터공학전공 |
| 학번 및 이름 | 201701718 길은영  201701794 최진영 |
|  |  |
| 제출일자 | 2020. 12. 15. |

목차

[1. 과제명(개발 프로그램명) 3](#_Toc58886408)

[2. 개발 필요성 및 목표 3](#_Toc58886409)

[3. 유사 프로그램 소개 3](#_Toc58886410)

[4. 프로그램 기능 3](#_Toc58886411)

[5. 데이터베이스 – CHICKENREVIEW 4](#_Toc58886412)

[A. REVIEW 4](#_Toc58886413)

[B. USER 4](#_Toc58886414)

[C. LIKEY 5](#_Toc58886415)

[6. 프로그램 코드 설명 5](#_Toc58886416)

[A. 회원 가입 5](#_Toc58886417)

[B. 로그인/로그아웃 6](#_Toc58886418)

[C. 리뷰 열람 및 검색 8](#_Toc58886419)

[D. 리뷰 작성 11](#_Toc58886420)

[E. 리뷰 삭제 16](#_Toc58886421)

[F. 회원 탈퇴 18](#_Toc58886422)

[G. 회원 관리 20](#_Toc58886423)

[7. 구현 화면 22](#_Toc58886424)

[A. 회원 가입 22](#_Toc58886425)

[B. 로그인 22](#_Toc58886426)

[C. 리뷰 커뮤니티 메인 페이지 – 사용자 23](#_Toc58886427)

[D. 리뷰 커뮤니티 메인 페이지 – 관리자 24](#_Toc58886428)

[E. 리뷰 상세 페이지 24](#_Toc58886429)

[F. 리뷰 작성 페이지 25](#_Toc58886430)

[G. 회원관리 창 25](#_Toc58886431)

[8. 느낀 점 26](#_Toc58886432)

# 과제명(개발 프로그램명)

치킨 리뷰 커뮤니티

# 개발 필요성 및 목표

코로나로 인해 사람들의 외식 횟수가 줄어들고 배달을 시켜 먹는 횟수가 늘어나고 있다. 배달 애플리케이션도 발전하면서 사람들이 손쉽게 배달을 시킬 수 있게 되어 그 수요는 더욱 증가하고 있다. 사람들은 처음 주문을 하는 가게라면 리뷰를 확인하고 메뉴와 가게를 선택하여 주문하게 된다. 하지만 배달 애플리케이션에서는 같은 메뉴라 하더라도 가게별 리뷰만 제공하며 같은 프랜차이즈더라도 지점이 다르다면 각 가게의 리뷰 창에서 리뷰를 확인할 수밖에 없다. 이러한 문제점을 해결하기 위해 어떠한 지역의 프랜차이즈 치킨을 주문한 사람들이 커뮤니티에 모여 의견을 공유할 수 있는 페이지를 만드는 것을 목적으로 한다.

# 유사 프로그램 소개

- 배달 애플리케이션 리뷰

# 프로그램 기능

* 관리자
* 회원이 작성한 리뷰를 사용자별로 열람한다.
* 회원이 작성한 리뷰 중 악의적인 리뷰를 삭제한다.
* 악의적인 리뷰를 작성한 회원을 탈퇴시킨다.
* 사용자
* 커뮤니티에 작성된 리뷰를 열람한다.
* 키워드를 검색하여 조건에 맞는 리뷰를 검색한다.
* 사용자 > 회원
* 로그인과 로그아웃을 할 수 있다.
* 로그인을 통해 커뮤니티에 리뷰를 작성할 수 있다.
* 다른 사람의 게시물이 도움이 되었다면 글을 추천할 수 있다.
* 회원 탈퇴를 할 수 있다.
* 사용자 > 비회원
* 회원가입을 할 수 있다.

# 데이터베이스 – CHICKENREVIEW

## REVIEW

리뷰에 관한 내용을 저장하는 테이블이다.

|  |  |  |
| --- | --- | --- |
| reviewID | 리뷰 아이디 | int |
| userID | 회원 아이디 | String |
| menuName | 메뉴 이름 | String |
| orderData | 주문 날짜 | String |
| whereOrder | 주문 지점 | String |
| todayDate | 오늘 날짜 | String |
| reviewTitle | 리뷰 제목 | String |
| reviewContent | 리뷰 내용 | String |
| totalScore | 총점 | String |
| delScore | 맛 점수 | String |
| deliveryScore | 배달 점수 | String |
| ServiceScore | 서비스 점수 | String |
| packingScore | 포장 점수 | String |
| chickenImage | 치킨 사진 | String |
| billImage | 영수증 사진 | String |
| likeCount | 추천 개수 | Int |

## USER

회원의 정보를 저장하는 테이블이다.

|  |  |  |
| --- | --- | --- |
| userID | 회원 아이디 | String |
| userPassword | 회원 비밀번호 | String |
| userEmail | 회원 이메일 주소 | String |
| userAddress | 회원 거주지역(동 단위) | String |

## LIKEY

각 회원이 추천한 리뷰의 아이디와 사용자의 IP를 저장하는 테이블이다.

|  |  |  |
| --- | --- | --- |
| userID | 회원 아이디 | String |
| reviewID | 리뷰 아이디 | int |
| userIP | 회원 아이피 | String |

# 프로그램 코드 설명

프로젝트의 코드 중 직접적인 동작을 하는 코드만 작성하였습니다.

## 회원 가입

|  |
| --- |
| **Logininput.jsp** |
| **if**(userID == **null** || userPassword == **null** || userEmail == **null**|| userAddress == **null**  ||userID == "" || userPassword == "" || userEmail == ""|| userAddress == ""){  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('전부 입력해주십시오')");  script.println("history.back()");  script.println("</script>");  script.close();  **return**;  } |
| * 회원가입 페이지에서 작성해야할 내용이 비어있다면 대화상자를 띄웁니다. |
|  |
| **int** result =user.join(**new** userEntity(userID, userPassword,userEmail,userAddress));  **if**(result == -1){  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('이미 존재하는 아이디입니다')");  script.println("history.back()");  script.println("</script>");  script.close();  **return**;  } **else**{  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('회원가입이 되었습니다.')");  script.println("location.href = 'login.jsp'");  script.println("</script>");  script.close();  **return**;  } |
| * 아이디의 중복 여부를 확인하여 중복된 아이디가 있다면 대화상자를 띄워 다시 입력하도록 하며, 중복된 아이디가 없다면 회원 가입을 허용합니다. |
|  |
| **userDB.java** |
| **public** **int** join(userEntity user) {  String SQL = "INSERT INTO user VALUES (?,?,?,?)";  Connection conn = **null**;  PreparedStatement pstmt= **null**;  ResultSet rs = **null**;  **try** {  conn = Databaseuser.*getConnection*();  pstmt = conn.prepareStatement(SQL);  pstmt.setString(1, user.getUserID());  pstmt.setString(2, user.getUserPassword());  pstmt.setString(3, user.getUserEmail());  pstmt.setString(4, user.getUserAddress());  **return** pstmt.executeUpdate();    }**catch**(Exception e) {  e.printStackTrace();  }**finally** {  **try**{**if**(conn!=**null**)conn.close();}**catch**(Exception e) {e.printStackTrace();}  **try**{**if**(pstmt!=**null**)pstmt.close();}**catch**(Exception e) {e.printStackTrace();}  **try**{**if**(rs!=**null**)rs.close();}**catch**(Exception e) {e.printStackTrace();}  }  **return** -1; //회원가입 실패  } |
| * 데이터베이스에 접근하여 insert문을 통해 회원가입을 시도합니다. |

## 로그인/로그아웃

|  |
| --- |
| **login.jsp** |
| <section style="max-width: *560px*; margin: *0 auto*;">  <form method=*"post"* action=*"logincheck.jsp"*>  <h2 style="text-align: *center*;">🍗천안 BHC 치킨 리뷰 커뮤니티🍗</h2>  <hr>  <h3 style="text-align: *center*;">로그인</h3>  <div class=*"form-group"*>  <label>아이디</label><br>  <input type=*"text"* class=*"from-control"* placeholder=*"아이디"* name=*"userID"*  maxlength=*"20"*  >  </div>  <div class=*"form-group"*>  <label>비밀번호</label><br>  <input type=*"password"* class=*"from-control"* placeholder=*"비밀번호"*  name=*"userPassword"* maxlength=*"20"*  >  </div>  <input type=*"submit"* class=*"btn btn-primary form-control"* value=*"로그인"*>  <input type=*"button"* onclick="location='new.jsp'"  class=*"btn btn-primary form-control"* value=*"회원가입"*  >  <input type=*"button"* class=*"btn btn-primary form-control"*  value=*"메인화면으로 돌아가기"* onclick="location='ReviewPage2.jsp'"  >  </form>  </section> |
| * 로그인 창을 띄워 아이디와 비밀번호를 입력할 수 있도록 합니다. * 로그인을 하고 싶지 않을 경우 회원가입 창을 보여주거나 메인으로 돌아갈 수 있도록 버튼을 추가합니다. |
|  |
| **login.jsp** |
| **public** **int** login(String userID, String userPassword) {  String SQL = "SELECT userPassword FROM user WHERE userID = ?";  Connection conn = **null**;  PreparedStatement pstmt= **null**;  ResultSet rs = **null**;  **try** {  conn = Databaseuser.*getConnection*();  pstmt = conn.prepareStatement(SQL);  pstmt.setString(1, userID);  rs = pstmt.executeQuery();  **if**(rs.next()) {  **if**(rs.getString(1).equals(userPassword)){  **return** 1; //로그인 성공  }  **else** {  **return** 0; //비밀번호 틀림  }  }  **return** -1; //아이디 없음    }**catch**(Exception e) {  e.printStackTrace();  }**finally** {  **try**{**if**(conn!=**null**)conn.close();}**catch**(Exception e) {e.printStackTrace();}  }  **return** -2; //데이터베이스 오류  } |
| * 데이터베이스에 접근하여 입력한 아이디와 비밀번호가 맞는지 확인한다. |

## 리뷰 열람 및 검색

|  |
| --- |
| **ReviewPage2.jsp** |
| <section id=*"sectionTitle"*>  <form name=*"form1"* action=*"./ReviewPage2.jsp"* method=*"get"*>  <select id =*"arr"* name=*"arrangeType"*  onchange=*"javascript:document.form1.submit();"*  >  <option value=*"최신등록순"*  <%**if** (arrangeType.equals("최신등록순"))  out.println("selected");%>  >최신등록순</option>  <option value=*"추천수순"*  <%**if** (arrangeType.equals("추천수순"))  out.println("selected");%>  >추천수순</option>  <option value=*"랭킹순"*  <%**if** (arrangeType.equals("랭킹순"))  out.println("selected");%>  >랭킹순</option>  </select>  <input id=*"inputsearch"* type=*"search"* name=*"search"* placeholder=*"내용을 입력하세요."*>  <button id=*"searchbtn"* type=*"submit"*>검색</button>  </form>  </section> |
| * 표시할 게시물이 정렬 순서를 선택하고 검색할 내용이 있다면 검색어를 입력한다. |
|  |
| <%  ArrayList<ReviewEntity> reviewList = **new** ArrayList<ReviewEntity>();  reviewList = **new** ReviewAction().getList(arrangeType, search, pageNumber);  **if** (reviewList != **null**)  **for** (**int** i = 0; i < reviewList.size(); i++) {  **if**(i == 5)**break**;  ReviewEntity review = reviewList.get(i);  %>  <section style="cursor:*pointer*" class=*"reviewSection"* id=*"reviewSection*<%=review.getReviewID()%>*"*>  <span style="font-weight: *bold*; font-size:*20px*;"><%=review.getReviewTitle()%></span>  <%=review.getWhereOrder()%>  <%=review.getMenuName()%>  <span style="display: *inline-block*; float: *right*;">추천 수: <%=review.getLikeCount()%></span>  <hr>  맛:  <%=review.getDelScore()%>  배달:  <%=review.getDeliveryScore()%>  서비스:  <%=review.getServiceScore()%>  포장:  <%=review.getPackingScore()%>  <span style="display: *inline-block*; float: *right*;">총점: <%=review.getTotalScore()%></span>  <br>  </section> |
| * 데이터베이스에 접근하여 메인 화면에 리뷰 제목, 주문 지점, 메뉴 이름, 추천 수, 맛, 배달, 서비스, 포장 점수, 총점을 표시한다. |
|  |
| <section class=*"reviewModal"* id=*"reviewModal*<%=review.getReviewID()%>*"*>  <a class=*"modal\_close\_btn"*>닫기</a>  <span style="font-weight: *bold*; font-size:*30px*;"><%=review.getReviewTitle()%></span>  <%=review.getWhereOrder()%>  <%=review.getMenuName()%>  <span style="display: *inline-block*; float: *right*;">추천 수: <%=review.getLikeCount()%>  <a onclick="return confirm('추천하시겠습니까?')"  href=*"./recommand.jsp?reviewID=*<%=review.getReviewID()%>*"*  >추천</a> <a onclick="return confirm('삭제하시겠습니까?')"  href=*"./deleteAction.jsp?reviewID=*<%=review.getReviewID()%>*"*  >삭제</a></span>  <hr>  <img  src=*"http://119.70.168.9:9928/team/uploadImage/*<%=review.getChickenImage()%>*"*  width=*200* height=*200*  />  <img  src=*'http://119.70.168.9:9928/team/uploadImage/*<%=review.getBillImage()%>*'*  width=*200* height=*200*  />  <br>  <%=review.getReviewContent()%>  <br>    <hr>  맛:  <%=review.getDelScore()%>  배달:  <%=review.getDeliveryScore()%>  서비스:  <%=review.getServiceScore()%>  포장:  <%=review.getPackingScore()%>  <span style="display: *inline-block*; float: *right*;">총점: <%=review.getTotalScore()%></span>  </section>    <br>  <script>  document.getElementById("reviewSection<%=review.getReviewID()%>").addEventListener('click', function() {  // 모달창 띄우기  modal("reviewModal<%=review.getReviewID()%>");  });    </script> |
| * 섹션을 클릭하면 리뷰의 상세정보를 표시하는 모달 창을 띄워준다. 해당 모달 창에는 리뷰의 제목, 주문 지점, 메뉴 이름, 추천 수, 추천과 삭제 버튼, 치킨 사진, 영수증 사진, 리뷰 내용, 맛, 배달, 서비스, 포장 점수, 총점, 닫기 버튼을 표시한다. |
|  |
| **ReviewAction.jsp** |
| **public** ArrayList<ReviewEntity> getList(String arrangeType, String search, **int** pageNumber) {  ArrayList<ReviewEntity> reviewList = **null**;  String SQL = "";  Connection conn = **null**;  PreparedStatement pstmt = **null**;  ResultSet rs = **null**;  **try** {  **if** (arrangeType.equals("최신등록순")) {  SQL = "SELECT \* FROM review WHERE CONCAT(menuName, whereOrder, reviewTitle, reviewContent) LIKE ? "  + "ORDER BY reviewID DESC LIMIT " + pageNumber\*5+", "+ pageNumber \* 5 + 6;  } **else** **if** (arrangeType.equals("추천수순")) {  SQL = "SELECT \* FROM review WHERE CONCAT(menuName, whereOrder, reviewTitle, reviewContent) LIKE ? "  + "ORDER BY likeCount DESC LIMIT " + pageNumber\*5+", "+ pageNumber \* 5 + 6;  } **else** **if** (arrangeType.equals("랭킹순")) {  SQL = "SELECT \* FROM review WHERE CONCAT(menuName, whereOrder, reviewTitle, reviewContent) LIKE ? "  + "ORDER BY totalScore DESC LIMIT " + pageNumber\*5+", "+ pageNumber \* 5 + 6;  }  conn = Databaseuser.*getConnection*();  pstmt = conn.prepareStatement(SQL);  pstmt.setString(1, "%" + search + "%");  rs = pstmt.executeQuery();  reviewList = **new** ArrayList<ReviewEntity>();  **while** (rs.next()) {  ReviewEntity review = **new** ReviewEntity(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4),  rs.getString(5), rs.getString(6), rs.getString(7), rs.getString(8), rs.getString(9),  rs.getString(10), rs.getString(11), rs.getString(12), rs.getString(13), rs.getString(14),  rs.getString(15), rs.getInt(16));  reviewList.add(review);  }  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (conn != **null**)  conn.close();  } **catch** (Exception e) {  e.printStackTrace();  }  }  **return** reviewList; // 데이터베이스 오류  } |
| * 데이터베이스에 접근하여 설정된 정렬순서에 따라 리뷰를 select하고 한 페이지에 리뷰가 5개씩 나오도록 설정합니다. * 입력한 검색어가 있다면 검색어를 포함하여 리뷰를 select합니다. |

## 리뷰 작성

|  |
| --- |
| **review.jsp** |
| <h3 style="text-align: *center*;">리뷰작성하기</h3>  <form method=*"post"* action=*"reviewinput.jsp"* enctype=*"multipart/form-data"*>  <div class=*"form-row"*>  <label>주문한 메뉴 : </label>  <input type=*"text"* name=*"menuName"* class=*from-control* maxlength=*"20"*>  <label>아이디</label>  <%=userID%>  <br> <label>주문한 날짜 : </label>  <input type=*"text"* name=*"orderData"* class=*from-control* maxlength=*"20"*>  <label>주문한 지점 : </label>  <select name=*"whereOrder"*>  <option value=*"청수점"* selected>청수점</option>  <option value=*"불당점"* selected>불당점</option>  <option value=*"두정점"* selected>두정점</option>  <option value=*"안서점"* selected>안서점</option>  <option value=*"신방행복점"* selected>신방행복점</option>  <option value=*"청수점"* selected>아산점</option>  <option value=*"불당점"* selected>성환점</option>  </select>  <br> <label>오늘 날짜 </label>  <%=sf.format(today)%>  <br> <label>제목 : </label>  <input type=*"text"* name=*"reviewTitle"* class=*from-control* maxlength=*"50"*>  <br> <label>내용 : </label>  <textarea name=*"reviewContent"* class=*from-control* maxlength=*"2048"*  style="heigth: *300px*; weight: *180px*;"  ></textarea>  <br> <label>맛</label>  <select name=*"delScore"*>  <option value=*"5"* selected>5점</option>  <option value=*"4"* selected>4점</option>  <option value=*"3"* selected>3점</option>  <option value=*"2"* selected>2점</option>  <option value=*"1"* selected>1점</option>  </select>  <label>배달</label>  <select name=*"deliveryScore"*>  <option value=*"5"* selected>5점</option>  <option value=*"4"* selected>4점</option>  <option value=*"3"* selected>3점</option>  <option value=*"2"* selected>2점</option>  <option value=*"1"* selected>1점</option>  </select>  <label>서비스</label>  <select name=*"serviceScore"*>  <option value=*"5"* selected>5점</option>  <option value=*"4"* selected>4점</option>  <option value=*"3"* selected>3점</option>  <option value=*"2"* selected>2점</option>  <option value=*"1"* selected>1점</option>  </select>  <label>포장</label>  <select name=*"packingScore"*>  <option value=*"5"* selected>5점</option>  <option value=*"4"* selected>4점</option>  <option value=*"3"* selected>3점</option>  <option value=*"2"* selected>2점</option>  <option value=*"1"* selected>1점</option>  </select>  <label>포장</label> <label>음식 사진</label>  <input type=*"file"* name=*"chickenImage"*>  <br> <label>영수증 사진</label>  <input type=*"file"* name=*"billImage"*>  <br>  <button type=*"submit"*>등록하기</button>  </div>  </form> |
| * 리뷰에 필요한 내용을 작성할 수 있는 input 태그를 작성한다. |
|  |
| **reviewinput.jsp** |
| <%  Date today = **new** Date();  SimpleDateFormat sf = **new** SimpleDateFormat("yyyy년 MM월 dd일");  request.setCharacterEncoding("UTF-8");  String userID = **null**;  **if**(session.getAttribute("userID")!=**null**){  userID = (String)session.getAttribute("userID");  }  **if**(userID==**null**){  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('먼저 로그인을 해주세요!')");  script.println("location.href = 'login.jsp");  script.println("</script>");  script.close();  **return**;  }  String menuName = **null**;  String orderData = **null**;  String whereOrder = **null**;  String todayDate = **null**;  String reviewTitle = **null**;  String reviewContent = **null**;  String totalScore = **null**;  String delScore = **null**;  String deliveryScore = **null**;  String serviceScore = **null**;  String packingScore = **null**;  String chickenImage = **null**;  String billImage = **null**;  **double** totalf=0;  String uploadDir ="C://Users/JINYOUNG/eclipse-workspace/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/wtpwebapps/team/uploadImage";  //uploadDir = uploadDir.substring(1,uploadDir.indexOf(".metadata"))+"team/WebContent/uploadImage";  System.out.println("절대경로 : " + uploadDir + "<br/>");  **int** maxSize = 1024 \* 1024 \* 100;  String encoding = "UTF-8";  MultipartRequest multipartRequest = **new** MultipartRequest(request, uploadDir, maxSize, encoding, **new** DefaultFileRenamePolicy());  chickenImage = multipartRequest.getFilesystemName("chickenImage");  billImage = multipartRequest.getFilesystemName("billImage");    **if**(menuName == **null** || orderData == **null** || whereOrder == **null**  || reviewTitle==**null** || reviewContent==**null**  || delScore == **null** || deliveryScore==**null** || serviceScore==**null**  || packingScore==**null** || reviewTitle.equals("")||reviewContent.equals("")){  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('전부 입력해주십시오')");  script.println("history.back()");  script.println("</script>");  script.close();  **return**;  }  ReviewDB reviewDB = **new** ReviewDB();  **int** result =reviewDB.write(**new** ReviewEntity(0, userID, menuName,orderData  ,whereOrder,todayDate,reviewTitle,reviewContent,  totalScore,delScore,deliveryScore,serviceScore,packingScore,chickenImage,billImage,0));    **if**(result == -1){  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('등록 실패')");  script.println("history.back()");  script.println("</script>");  script.close();  **return**;  } **else**{  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('등록 성공')");  script.println("location.href = 'ReviewPage2.jsp'");  script.println("</script>");  script.close();  **return**;  }  %> |
| * 작성한 내용 중 비어있는 내용이 있다면 대화상자를 띄워서 알림을 표시합니다. * 리뷰 등록에 실패하면 실패했다는 대화상자를, 성공하면 성공했다는 대화상자를 표시합니다. |
|  |
| **ReviewDB.java** |
| **public** **int** write(ReviewEntity reviewEntity) {  // 사용자가 리뷰 쓰는거  String SQL = "INSERT INTO REVIEW VALUES (NULL,?,?,?,?,?,?,?,?,?,?,?,?,?,?,0)";  Connection conn = **null**;  PreparedStatement pstmt = **null**;  **try** {  conn = Databaseuser.*getConnection*();  pstmt = conn.prepareStatement(SQL);  pstmt.setString(1, reviewEntity.getUserID());  pstmt.setString(2, reviewEntity.getMenuName());  pstmt.setString(3, reviewEntity.getOrderData());  pstmt.setString(4, reviewEntity.getWhereOrder());  pstmt.setString(5, reviewEntity.getTodayDate());  pstmt.setString(6, reviewEntity.getReviewTitle());  pstmt.setString(7, reviewEntity.getReviewContent());  pstmt.setString(8, reviewEntity.getTotalScore());  pstmt.setString(9, reviewEntity.getDelScore());  pstmt.setString(10, reviewEntity.getDeliveryScore());  pstmt.setString(11, reviewEntity.getServiceScore());  pstmt.setString(12, reviewEntity.getPackingScore());  pstmt.setString(13, reviewEntity.getChickenImage());  pstmt.setString(14, reviewEntity.getBillImage());  **return** pstmt.executeUpdate();  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (conn != **null**)  conn.close();  } **catch** (Exception e) {  e.printStackTrace();  }  }  **return** -1; // 데이터베이스 오류  } |
| * 데이터베이스에 접근하여 입력한 정보를 insert한다. |

## 리뷰 삭제

|  |
| --- |
| **deleteAction.jsp** |
| <%  request.setCharacterEncoding("UTF-8");  String userID = **null**;  String reviewID=**null**;  **if**(session.getAttribute("userID")!=**null**){  userID = (String)session.getAttribute("userID");  }  **if**(userID==**null**){  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('리뷰를 삭제하려면 로그인이 필요합니다.')");  script.println("location.href = 'login.jsp'");  script.println("</script>");  script.close();    }  **if**(request.getParameter("reviewID")!=**null**){  reviewID = request.getParameter("reviewID");  }  ReviewAction review = **new** ReviewAction();  **if**(userID.equals(review.getUserID(reviewID)) || userID.equals("manager")){  **int** result = **new** ReviewAction().delete(reviewID);  **if**(result==1){  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('삭제가 완료되었습니다.')");  script.println("location.href = 'ReviewPage2.jsp'");  script.println("</script>");  script.close();  **return**;  } **else**{  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('데이터베이스 오류')");  script.println("location.href = 'ReviewPage2.jsp'");  script.println("</script>");  script.close();  **return**;  }  }  **else**{  System.out.println("오류");  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('글쓴 사람만 삭제 가능')");  script.println("location.href = 'ReviewPage2.jsp'");  script.println("</script>");  script.close();  }    %> |
| * 현재 로그인 된 아이디와 리뷰를 작성한 아이디가 같거나, 리뷰를 작성한 아이디가 관리자 아이디라면 삭제를 실행한다. * 삭제가 정상적으로 이루어졌다면 완료 대화상자를 띄워주고 실패했다면 데이터베이스 오류를 보여준다. |
|  |
| **ReviewAction.java** |
| **public** String getUserID(String reviewID) {  String SQL = "SELECT userID FROM REVIEW WHERE reviewID = ?";  Connection conn = **null**;  PreparedStatement pstmt = **null**;  ResultSet rs = **null**;  **try** {  conn = Databaseuser.*getConnection*();  pstmt = conn.prepareStatement(SQL);  pstmt.setInt(1, Integer.*parseInt*(reviewID));  rs = pstmt.executeQuery();  **if** (rs.next()) {  **return** rs.getString(1);  }  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (conn != **null**)  conn.close();  } **catch** (Exception e) {  e.printStackTrace();  }  **try** {  **if** (pstmt != **null**)  pstmt.close();  } **catch** (Exception e) {  e.printStackTrace();  }  **try** {  **if** (rs != **null**)  rs.close();  } **catch** (Exception e) {  e.printStackTrace();  }  }  **return** **null**; // 존재하지않음  } |
| * 리뷰의 아이디를 가져온다. |
|  |
| **public** **int** delete(String reviewID) {  String SQL = "DELETE FROM REVIEW WHERE reviewID = ?";  Connection conn = **null**;  PreparedStatement pstmt = **null**;  ResultSet rs = **null**;  **try** {  conn = Databaseuser.*getConnection*();  pstmt = conn.prepareStatement(SQL);  pstmt.setInt(1, Integer.*parseInt*(reviewID));  **return** pstmt.executeUpdate();  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (conn != **null**)  conn.close();  } **catch** (Exception e) {  e.printStackTrace();  }  **try** {  **if** (pstmt != **null**)  pstmt.close();  } **catch** (Exception e) {  e.printStackTrace();  }  **try** {  **if** (rs != **null**)  rs.close();  } **catch** (Exception e) {  e.printStackTrace();  }  }  **return** -1; // 데이터베이스 오류  } |
| * 데이터베이스에 접근에 성공하면 delete문을 사용하여 리뷰를 삭제한다. |

## 회원 탈퇴

|  |
| --- |
| **outMember.jsp** |
| **int** result = **new** userDB().delete(userID);  **if**(result==1){  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('탈퇴되었습니다.')");  script.println("history.back()");  script.println("</script>");  script.close();  **return**;  } **else**{  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('데이터베이스 오류')");  script.println("window.open('','\_self').close();");  script.println("</script>");  script.close();    } |
| * 탈퇴를 성공하면 탈퇴 성공 대화상자를, 실패하면 데이터베이스 오류 대화상자를 표시합니다. |
|  |
| **userDB.java** |
| **public** **int** delete(String userID) {  String SQL = "DELETE FROM USER WHERE userID = ?";  Connection conn = **null**;  PreparedStatement pstmt = **null**;  ResultSet rs = **null**;  **try** {  conn = Databaseuser.*getConnection*();  pstmt = conn.prepareStatement(SQL);  pstmt.setString(1, userID);  **return** pstmt.executeUpdate();  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (conn != **null**)  conn.close();  } **catch** (Exception e) {  e.printStackTrace();  }  **try** {  **if** (pstmt != **null**)  pstmt.close();  } **catch** (Exception e) {  e.printStackTrace();  }  **try** {  **if** (rs != **null**)  rs.close();  } **catch** (Exception e) {  e.printStackTrace();  }  }  **return** -1; // 데이터베이스 오류  } |
| * 데이터베이스에 접근하여 delete문을 사용하여 회원을 탈퇴합니다. |

## 회원 관리

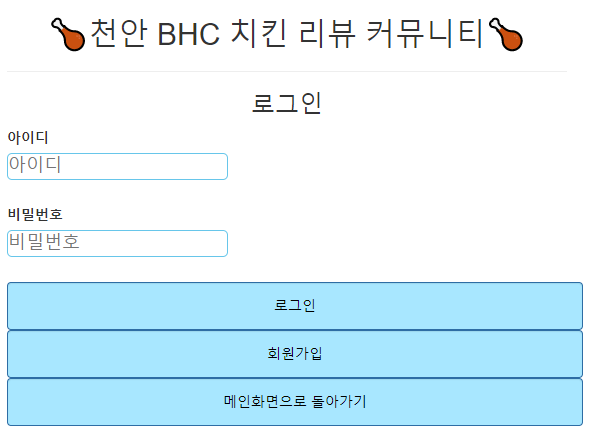
|  |
| --- |
| **Usercheck.jsp** |
| <h2>회원관리</h2>  <table>  <tr>  <th>아이디</th>  <th>비밀번호</th>  <th>주소</th>  <th>이메일</th>  <th>회원삭제</th>  </tr>  <%  **for** (**int** i = 0; i < users.size(); i++) {  userEntity review = users.get(i);  %>  <tr>  <td><%=review.getUserID()%></td>  <td><%=review.getUserPassword()%></td>  <td><%=review.getUserAddress()%></td>  <td><%=review.getUserEmail()%></td>  <td>  <a onclick="return confirm('삭제하시겠습니까?')"  href=*"./deleteid.jsp?userID=*<%=review.getUserID()%>*"*  >삭제</a>  </td>  </tr>  <%  }  %>  </table> |
| * 회원의 아이디, 비밀번호, 주소, 이메일을 표시합니다. * 삭제 버튼을 클릭하면 삭제합니다. |
|  |
| **delete.jsp** |
| userDB user = **new** userDB();    **int** result = **new** userDB().delete(userID);  **if**(result==1){  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('삭제가 완료되었습니다.')");  script.println("window.open('','\_self').close();");  script.println("</script>");  script.close();  **return**;  } **else**{  PrintWriter script = response.getWriter();  script.println("<script>");  script.println("alert('데이터베이스 오류')");  script.println("window.open('','\_self').close();");  script.println("</script>");  script.close();    } |
| * 회원이 삭제되면 삭제 성공 대화상자를, 실패하면 데이터베이스 오류 대화상자를 표시합니다. |
|  |
| **userDB.java** |
| **public** **int** delete(String userID) {  String SQL = "DELETE FROM USER WHERE userID = ?";  Connection conn = **null**;  PreparedStatement pstmt = **null**;  ResultSet rs = **null**;  **try** {  conn = Databaseuser.*getConnection*();  pstmt = conn.prepareStatement(SQL);  pstmt.setString(1, userID);  **return** pstmt.executeUpdate();  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (conn != **null**)  conn.close();  } **catch** (Exception e) {  e.printStackTrace();  }  **try** {  **if** (pstmt != **null**)  pstmt.close();  } **catch** (Exception e) {  e.printStackTrace();  }  **try** {  **if** (rs != **null**)  rs.close();  } **catch** (Exception e) {  e.printStackTrace();  }  }  **return** -1; // 데이터베이스 오류  } |
| 데이터베이스에 접근하여 delete문을 사용하여 회원을 탈퇴합니다. |

# 구현 화면

## 회원 가입



## 로그인



## 리뷰 커뮤니티 메인 페이지 – 사용자



## 리뷰 커뮤니티 메인 페이지 – 관리자



## 리뷰 상세 페이지



## 리뷰 작성 페이지

테이블이(가) 표시된 사진

자동 생성된 설명

## 회원관리 창

