CSCE 5430 - Software Engineering

Code for Core Functionalities – Phase I

FREQ-1: Login (All Users):

/\*This a LoginController upon typing the website URL localhost:8080/dpa/login it will redirect users to the login page users when types in the username, password and role and hits login button he will be redirected to this controller post method handleLoginRequest() which calls a LoginService to validate user and redirects him to their home page based on the role\* /[1]

@Controller

**public** **class** LoginController {

@Autowired

LoginService loginService;

@Autowired

RetrieveUsersService retrieveUsersService;

//It redirects all the users to the login page

@RequestMapping(value = "/login", method = RequestMethod.***GET***)

**public** String login() {

**return** "login";

}

//This method is called upon hitting login button and validates user

@RequestMapping(value = "/login", method = RequestMethod.***POST***)

**public** String handleLoginRequest(@RequestParam String userName, @RequestParam String password, @RequestParam String role, ModelMap model, HttpServletRequest request, HttpServletResponse response) **throws** NoSuchAlgorithmException, UnsupportedEncodingException {

List<Login> result = **new** ArrayList<Login>();

Hashing hashing = **new** Hashing();

String pwd = hashing.hashString(password, "MD5");

result = loginService.checkCredentials(userName, pwd, role);

**if** (result.isEmpty() == **true**) {

model.put("errorMsg", "Invalid Credentials");

**return** "login";

} **else** {

HttpSession session = request.getSession();

session.setAttribute("userName",userName);

**if** (role.equals("admin")) {

**return** "adminhome";

} **else** **if**(role.equals("student")) {

List<Request> myAdvisors = retrieveUsersService.getMyAdvisors(userName);

model.addAttribute("myAdvisors", myAdvisors);

**return** "studenthome";

} **else** **if**(role.equals("professor") || role.equals("chair") || role.equals("associateChair")) {

model.addAttribute("myStudents", retrieveUsersService.getMyStudents(userName));

**return** "professorhome";

}**else** {

**return** "temp";

}

}

}

}

/\* This hashing class has methods for hashing password which are called by Login and Registration services to hash the password while insertion and before login \*/

**public** **class** Hashing {

**public** String hashString(String message, String algorithm)

**throws** NoSuchAlgorithmException, UnsupportedEncodingException {

MessageDigest digest = MessageDigest.*getInstance*(algorithm);

**byte**[] hashedBytes = digest.digest(message.getBytes("UTF-8"));

**return** *convertByteArrayToHexString*(hashedBytes);

}

**public** **static** String convertByteArrayToHexString(**byte**[] arrayBytes) {

StringBuffer stringBuffer = **new** StringBuffer();

**for** (**int** i = 0; i < arrayBytes.length; i++) {

stringBuffer.append(Integer.*toString*(

(arrayBytes[i] & 0xff) + 0x100,16).substring(1));

}

**return** stringBuffer.toString();

}

}

/\*The LoginServiceImpl Class calls a Login Dao class to check if the user details exist in the database and returns the result to the LoginController\*/

@Service

**public** **class** LoginServiceImpl **implements** LoginService{

@Autowired

LoginDao loginDao;

**public** List<Login> checkCredentials(String userName, String password, String role) {

List<Login> result = **new** ArrayList<Login>();

result = loginDao.validateUser(userName, password, role);

**return** result;

}

}

/\*This LoginServiceDaoImpl class checks if the credentials submitted by the user exist in the database and returns the acquired rows to the LoginService for it to accept/reject user login based on if the list is empty or not \*/[2]

//Checks if the combination of username, password and role exist in the database

@Service

**public** **class** LoginDaoImpl **implements** LoginDao {

@Autowired

DataSource dataSource;

**public** List<Login> validateUser(String userName, String password, String role) {

// **TODO** Auto-generated method stub

List<Login> loginList = **new** ArrayList<Login>();

String sql = "select \* from login where userName='" + userName + "' and password= '" + password + "' and role= '" + role +"'";

JdbcTemplate jdbctem = **new** JdbcTemplate(dataSource);

loginList = jdbctem.query(sql, **new** LoginRowMapper());

**return** loginList;

}

}

/\* This is the code for login.jsp which is the view (HTML). It has two form fields and it is mapped to the login page by the form action=/dpa/login. As soon as the user fills in the username, password and role and clicks login the form action request sends it to the LoginController\*/[3]

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ include file="../common/header.jspf"%>

<!-- login page -->

<div class="container">

<form name="loginForm" action="/dpa/login" class="form-horizontal" method="POST" >

<div align = "center">

<font color="green">${succ}</font>

</div>

<div class="form-group">

<h3 style="text-align:center;">

<strong>Login</strong>

</h3>

</div>

<div class="form-group">

<label class="control-label col-sm-4" for="userName">UserName

</label>

<div class="col-sm-4">

<input type="text" class="form-control field" id="userName"

name="userName" placeholder="Enter UserName" required onblur="validateUserName()">

<span id="errorUserNameContent" style="color:red"></span>

</div>

</div>

<div class="form-group">

<label class="control-label col-sm-4" for="password">Password</label>

<div class="col-sm-4">

<input type="Password" class="form-control field" id="password"

name="password" placeholder="Enter password" onblur="validatePassword()" required>

<span id="errorPasswordContent" style="color:red"></span>

</div>

</div>

<div class="form-group">

<label class="control-label col-sm-4" for="role">role</label>

<div class="col-sm-4">

<select class="form-control field" id ="role" name="role">

<c:forEach items="${roles}" var="role">

<option value="${role}">${role}</option>

</c:forEach>

</select>

</div>

</div>

<div class="form-group">

<div class="col-sm-offset-4 col-sm-4">

<input type="submit" value="login" class="btn btn-default" />

</div>

</div>

<div align = "center">

<a href="register">NewUser? Register</a>

</div>

<div align = "center">

<a href="#">Forgot Password</a>

</div>

<div align = "center">

<font color="red">${errorMsg}</font>

</div>

</form>

</div>

<!—- Javascript to validate username and Password Fields -->

<script>

var var1=false;

var var2=false;

function validate(){

return (var1 && var2 )

}

function validateUserName(){

var usename =document.forms["loginForm"]["userName"].value;

if (!(/[A-Za-z0-9\_]+/).test(JSON.stringify(usename))) {

document.getElementById("errorUserNameContent").innerHTML = "Username should not contain special characters";

document.forms["loginForm"]["userName"].value="";

var1= false;

}

else{ document.getElementById("errorUserNameContent").innerHTML = "";

var1= true;

}

}

function validatePassword() {

var pwd = document.forms["loginForm"]["password"].value;

var patternpass=/[A-Za-z0-9]+/;

if((! patternpass.test(JSON.stringify(pwd)))){

document.getElementById("errorPasswordContent").innerHTML = "Password should not contain special characters";

document.forms["loginForm"]["Password"].value="";

var2= false;

}else{

document.getElementById("errorPasswordContent").innerHTML = "";

var2= true;

}

}

</script>

<%@ include file="../common/footer.jspf"%>

FREQ -2: Registration (All Users)

/\*When a user clicks newuser?register link he will be mapped to this controller with url pattern “ /register” the first register() method directs users to the registration form and the second method is called upon user clicking register button and calls the registerService to insert the user to the database. If the user is added successfully it redirects to the login page else he will be notifies of the error \*/[4]

//Controls all the user registration related URL's

@Controller

**public** **class** RegisterController {

@Autowired

RegisterService registerService;

//directs user to the registration form

@RequestMapping(value = "/register", method = RequestMethod.***GET***)

**public** String register() {

**return** "register";

}

//handles the register request

@RequestMapping(value = "/register", method = RequestMethod.***POST***)

**public** String handleLoginRequest(@ModelAttribute Register register, ModelMap model) **throws** NoSuchAlgorithmException, UnsupportedEncodingException {

String uPwd = register.getPassword();

Hashing hashing = **new** Hashing();

String pwd = hashing.hashString(uPwd, "MD5");

register.setPassword(pwd);

**int** result = registerService.insertUser(register);

**if** (result == 1) {

model.put("succ", "Registered Sucessfully, Please Login");

**return** "login";

} **else** {

model.put("error", "Username already existing");

**return** "register";

}

}

/\* This RegisterServiceImpl calls the registerDaoImpl method insertUserDetails(register) which inserts users to the database and returns the insert status \*/

//calls the registerDao to insert user into the database

@Service

**public** **class** RegisterServiceImpl **implements** RegisterService{

@Autowired

RegisterDao registerDao;

**public** **int** insertUser(Register register) {

// **TODO** Auto-generated method stub

**int** result = registerDao.insertUserDetails(register);

**return** result;

}

}

/\* This class inserts the new registered users to the database upon successful registration it will return 1 else 0 \*/

@Service

**public** **class** RegisterDaoImpl **implements** RegisterDao {

@Autowired

DataSource dataSource;

//insert the new users into the database

**public** **int** insertUserDetails(Register register) {

// **TODO** Auto-generated method stub

String role = register.getRole();

**if**(role == "chair" || role == "associatechair")

{

role = "professor";

}

**int** result;

**try** {

String sql = "INSERT INTO register " + "(name, email, role, userName, password) VALUES (?, ?, ?, ?, ?)";

JdbcTemplate jdbcTemplate = **new** JdbcTemplate(dataSource);

result = jdbcTemplate.update(sql, **new** Object[] {register.getName(), register.getEmail(), register.getRole(), register.getUserName(), register.getPassword()});

String sql1 = "INSERT INTO login " + "(userName, password, role) VALUES (?, ?, ?)";

JdbcTemplate jdbcTemp = **new** JdbcTemplate(dataSource);

**int** result1 = jdbcTemp.update(sql1, **new** Object[] {register.getUserName(),register.getPassword(),role});

}**catch**(Exception e) {

result = 0;

}

**return** result;

}

}

FREQ-3: Manage users (Administrator)

/\* Below is the admin controller which has methods to redirect administrator to his home page and also to manage users when the administrator selects which user to manage and clicks add he will be mapped to “/adduser” URL GET method which returns him a page where he can add chair, associate chair or admin specialist. The other “/adduser” POST method is called upon admin filling in the details and clicking add button, this controller will call “insertUser()” method of “registerService” class to insert the users managed by admin, then the “registerServiceImpl” uses “registerDao” to insert chair, associate chair, admin specialist to the database\*/[5]

@Controller

**public** **class** AdminController {

@Autowired

RegisterService registerService;

//directs user to adminhome

@RequestMapping(value = "/adminhome", method = RequestMethod.***GET***)

**public** String login() {

**return** "adminhome";

}

//returns add user page

@RequestMapping(value = "/adduser", method = RequestMethod.***GET***)

**public** String returnAddUserPage(HttpServletRequest request, HttpServletResponse response, @RequestParam String add, Model model) {

model.addAttribute("user", add);

HttpSession session = request.getSession(**false**);

**if** (session != **null**) {

String userName = (String)session.getAttribute("userName");

}

**return** "adduser";

}

//calls insertUser() of registerService class to add chair, associate chair and admin specialist to the database

@RequestMapping(value = "/adduser", method = RequestMethod.***POST***)

**public** String addUser(@ModelAttribute Register register, ModelMap model) {

**int** result = registerService.insertUser(register);

**if** (result == 1) {

model.put("success", "User registered successfully");

**return** "adduser";

} **else** {

model.put("error", "Username already existing");

**return** "adduser";

}

}

}

FREQ-5: Send Requests to professors and withdraw (Student)

FREQ-8: Accept/decline requests (Professor)

/\* This is request controller which has mapping URL’s for sendrequest, sentrequests, receivedrequests, acceptrequest and deleterequest which upon request by either student or professor calls the respective services in the RequestServiceImpl for student to send request, delete sent request, view sent requests and for professor to accept request, reject request or view received requests \*/[6]

@Controller

**public** **class** RequestController {

@Autowired

RequestService requestService;

//returns advisor request form

@RequestMapping(value = "/sendrequest", method = RequestMethod.***GET***)

**public** String login(HttpServletRequest request, HttpServletResponse response, Model model) {

HttpSession session = request.getSession(**false**);

**if** (session != **null**) {

String userName = (String) session.getAttribute("userName");

}

**return** "advisorrequest";

}

//calls send request of requestService and redirects student to studenthome

@RequestMapping(value = "/sendrequest", method = RequestMethod.***POST***)

**public** String sendRequest(HttpServletRequest request, HttpServletResponse response, @ModelAttribute Request sendRequest, ModelMap model) {

HttpSession session = request.getSession(**false**);

**if** (session != **null**) {

String userName = (String) session.getAttribute("userName");

}

**int** result = requestService.sendRequest(sendRequest);

**return** "studenthome";

}

//calls getSentRequests method of requestService and redirects student to sentrequests

@RequestMapping(value = "/sentrequests", method = RequestMethod.***GET***)

**public** String getSentRequests(HttpServletRequest request, HttpServletResponse response, ModelMap model) {

HttpSession session = request.getSession(**false**);

List<Request> sentRequests = **new** ArrayList<Request>();

**if** (session != **null**) {

String userName = (String) session.getAttribute("userName");

model.addAttribute("sentRequests", requestService.getSentRequests(userName));

**return** "sentrequests";

}**else** {

**return** "login";

}

}

//calls getReceivedRequests method of requestService and redirects professor to receivedrequests

@RequestMapping(value = "/receivedrequests", method = RequestMethod.***GET***)

**public** String getReceivedRequests(HttpServletRequest request, HttpServletResponse response, ModelMap model) {

HttpSession session = request.getSession(**false**);

List<Request> receivedRequests = **new** ArrayList<Request>();

**if** (session != **null**) {

String userName = (String) session.getAttribute("userName");

model.addAttribute("receivedRequests", requestService.getReceivedRequests(userName));

**return** "receivedrequests";

}**else** {

**return** "login";

}

}

//calls acceptRequest method of requestService and redirects professor to professorhome

@RequestMapping(value = "/acceptrequest", method = RequestMethod.***POST***, consumes="application/json")

**public** String acceptRequest(@RequestBody String details, HttpServletRequest request, HttpServletResponse response, ModelMap model){

String[] requestDetails = requestService.splitString(details);

HttpSession session = request.getSession(**false**);

**if** (session != **null**) {

String userName = (String) session.getAttribute("userName");

**int** Result = requestService.acceptRequest(requestDetails[0], requestDetails[1], requestDetails[2], requestDetails[3]);

**return** "professorhome";

}**else** {

**return** "login";

}

}

//calls deleteRequest method of requestService and redirects user to their home

@RequestMapping(value = "/deleterequest", method = RequestMethod.***POST***, consumes="application/json")

**public** String deleteRequest(@RequestBody String details, HttpServletRequest request, HttpServletResponse response, ModelMap model){

String[] receivedDetails = requestService.split(details);

HttpSession session = request.getSession(**false**);

**if** (session != **null**) {

String userName = (String) session.getAttribute("userName");

**int** Result = requestService.deleteRequest(receivedDetails[0], receivedDetails[1]);

**return** "studenthome";

}**else** {

**return** "login";

}

}

}

/\* This RegisterDaoImpl’s inserts the request details into the database by calling “sendAdvisorRequest”, deletes the requests details from database upon reject by professor or delete by student by calling “deleteRequest()” method. It gets the list of received requests “getReceivedRequests()” from the database upon request by professors. It gets a list of sent requests by using “getSentRequests()” upon a request by student. It also changes the request status by using method “acceptrequest()”.\*/

//handles all the database operations of requests and responses

@Service

**public** **class** RequestDaoImpl **implements** RequestDao{

@Autowired

DataSource dataSource;

//inserts request into the database

**public** **int** sendAdvisorRequest(Request sendRequest) {

// **TODO** Auto-generated method stub

**int** result;

String requestStatus = "Request Sent";

String sql = "INSERT INTO requests " + "(userName, professorName,professorEmail, department, admissionSemester, admissionYear, message, requestStatus) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

JdbcTemplate jdbcTemplate = **new** JdbcTemplate(dataSource);

result = jdbcTemplate.update(sql, **new** Object[] {sendRequest.getUserName(), sendRequest.getProfessorName(), sendRequest.getProfessorEmail(), sendRequest.getDepartment(), sendRequest.getAdmissionSemester(), sendRequest.getAdmissionYear(),sendRequest.getMessage(), requestStatus});

**return** result;

}

//get the sent requests

**public** List<Request> getSentRequests(String userName) {

// **TODO** Auto-generated method stub

List<Request> requests = **new** ArrayList<Request>();

String requestStatus = "Request Sent";

String sql = "select \* from requests where userName='" + userName + "' and requestStatus='" + requestStatus + "'";

JdbcTemplate jdbctem = **new** JdbcTemplate(dataSource);

requests = jdbctem.query(sql, **new** ReceivedRequestRowMapper());

**return** requests;

}

//get the accepted requests

**public** List<Request> getacceptedRequests(String userName) {

// **TODO** Auto-generated method stub

List<Request> requests = **new** ArrayList<Request>();

String requestStatus = "Request Accepted";

String sql = "select professorName, requestStatus from requests where userName='" + userName + "' and requestStatus='" + requestStatus + "'";

JdbcTemplate jdbctem = **new** JdbcTemplate(dataSource);

requests = jdbctem.query(sql, **new** RequestRowMapper());

**return** requests;

}

//get received requests

**public** List<Request> getReceivedRequests(String userName) {

// **TODO** Auto-generated method stub

List<Request> receivedRequests = **new** ArrayList<Request>();

String requestStatus = "Request Sent";

String sql = "select email from register where userName='" + userName + "'" ;

JdbcTemplate jdbctem = **new** JdbcTemplate(dataSource);

String email = jdbctem.queryForObject(sql, String.**class**);

String sql1 = "select \* from requests where professorEmail='" + email + "'and requestStatus='" + requestStatus + "'";

JdbcTemplate jdbctem1 = **new** JdbcTemplate(dataSource);

receivedRequests = jdbctem1.query(sql1, **new** ReceivedRequestRowMapper());

**return** receivedRequests;

}

//accept the request

**public** **int** acceptRequest(String sName, String sMajor, String pName,String pEmail) {

// **TODO** Auto-generated method stub

String sql = "INSERT INTO majorprofessor " + "(studentName, StudentMajor, professorName, professorEmail) VALUES (?, ?, ?, ?)";

JdbcTemplate jdbcTemplate = **new** JdbcTemplate(dataSource);

**int** result = jdbcTemplate.update(sql, **new** Object[] {sName, sMajor, pName, pEmail});

String requestStatus = "Request Accepted";

String sql1 = "update requests set requestStatus = ? where userName = ? and professorEmail = ?";

JdbcTemplate jdbcTemp = **new** JdbcTemplate(dataSource);

**int** result1 = jdbcTemp.update(sql1, **new** Object[] {requestStatus, sName, pEmail});

**return** result;

}

//delete the sent request

**public** **int** deleteRequest(String pEmail, String sName) {

String sql = "delete from requests where professorEmail='" + pEmail + "' and userName='" + sName + "'";

JdbcTemplate jdbcTemplate = **new** JdbcTemplate(dataSource);

jdbcTemplate.update(sql);

**return** 0;

}

}

FREQ -6: Fill in the Degree Plan (Student)

FREQ -7: Submit The Degree Plan to Professor for Signature (student)

/\* this controller handles the action requests for the degree plan form and degree plan save and submissions. It calls the various methods of “degreePlanService” class get the core courses of each of four CSCE groups and also submitDegreePlan() method to submit or save degree plan. The same submitDegreePlan() method saves the degree plan if the requestparam act is save and it submits or sends it to the professor if the requestparam act is submit\*/

//handles the view, submit, save and all other degree plan action requests

@Controller

**public** **class** DegreePlanController {

@Autowired

DegreePlanService degreePlanService;

//gets mandatory and optional courses form the database and makes them available to the view degreepla.jsp

@RequestMapping(value = "/degreeplan", method = RequestMethod.***POST***)

**public** String openDegreePlan(@RequestParam String selectPlan, HttpServletRequest request, HttpServletResponse response, ModelMap model){

model.put("major", selectPlan);

HttpSession session = request.getSession(**false**);

**if** (session != **null**) {

String userName = (String) session.getAttribute("userName");

List<Integer> optionalCourses = Arrays.*asList*(1, 2, 3, 4, 5, 6, 7, 8);

model.addAttribute("groupACourses", degreePlanService.getGroupACourses(selectPlan));

model.addAttribute("groupBCourses", degreePlanService.getGroupBCourses(selectPlan));

model.addAttribute("groupCCourses", degreePlanService.getGroupCCourses(selectPlan));

model.addAttribute("groupDCourses", degreePlanService.getGroupDCourses(selectPlan));

model.addAttribute("optionalCourses", optionalCourses);

model.addAttribute("optionalCourseLists", degreePlanService.getOptionalCourses());

**return** "degreeplan";

} **else** {

**return** "login";

}

}

//redirects user to the degree plan page with the success message upon success else to the degree plan with error message

@RequestMapping(value = "/degreeplanform", method = RequestMethod.***POST***)

**public** String submitDegreePlan(@ModelAttribute DegreePlan degreePlan, @RequestParam act, HttpServletRequest request, HttpServletResponse response, ModelMap model){

System.***out***.println(degreePlan);

HttpSession session = request.getSession(**false**);

**if** (session != **null**) {

String userName = (String) session.getAttribute("userName");

**int** result = degreePlanService.submitDegreePlan(degreePlan, userName);

model.put("success", "Degree Plan Submitted Successfully");

**return** "degreeplan";

} **else** {

model.put("error", "Submission Failed");

**return** "degreeplan";

}

}

}

/\* The below DegreePlanDaoImpl saves the degree plan list that is all the core courses and optional courses into the database and it also has method getCoreCourses() which gets all the core courses from the database and also getOptionalCourses() to get all the CSCE department courses which are displayed in the dropdowns of degreeplan.jsp file\*/

@Service

**public** **class** DegreePlanDaoImpl **implements** DegreePlanDao{

@Autowired

DataSource dataSource;

//Get the course list from the database

@Override

**public** List<Course> getCoreCourses() {

List<Course> courseList = **new** ArrayList<Course>();

String sql = "select \* from mandatorycourses";

JdbcTemplate jdbctem = **new** JdbcTemplate(dataSource);

courseList = jdbctem.query(sql, **new** CourseRowMapper());

**return** courseList;

}

//return all CSCE Department courses

@Override

**public** List<String> getOptionalCourses() {

List<String> optionalCourses = **new** ArrayList<String>();

String sql = "select \* from courses";

JdbcTemplate jdbctem = **new** JdbcTemplate(dataSource);

optionalCourses = jdbctem.query(sql, **new** OptionalCourseRowMapper());

**return** optionalCourses;

}

//inserts the degree plan into database

@Override

**public** **int** submitDegreePlan(DegreePlan degreePlan, String userName) {

String sql1 = "INSERT INTO gre " + "(userName, verbal, quantitative, analytical, dateTaken) VALUES (?, ?, ?, ?, ?)";

JdbcTemplate jdbcTemp = **new** JdbcTemplate(dataSource);

**int** result1 = jdbcTemp.update(sql1, **new** Object[] {userName,degreePlan.getGre().getVerbal(), degreePlan.getGre().getQuantitative(), degreePlan.getGre().getAnalytical(), degreePlan.getGre().getDateTaken()});

String sql = "INSERT INTO degreeplan " + "(userName, name, studentId, localAddress, email, degree, major, minor, interestArea, majorProfessor, coMajorProfessor, totalCreditHours, degreePlanStatus) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

JdbcTemplate jdbcTemplate = **new** JdbcTemplate(dataSource);

**int** result = jdbcTemplate.update(sql, **new** Object[] {userName, degreePlan.getName(), degreePlan.getStudentId(), degreePlan.getLocalAddress(), degreePlan.getEmail(), degreePlan.getDegree(), degreePlan.getMajor(), degreePlan.getMinor(), degreePlan.getInterestArea(), degreePlan.getMajorProfessor(), degreePlan.getCoMajorProfessor(), degreePlan.getTotalCreditHours(), "submitted"});

**int** coreReturn = insertCoreCourses(degreePlan, userName);

**int** courseReturn = insertOptionalCourses(degreePlan, userName);

String sql2 = "update majorprofessor set degreePlanStatus = ? where studentName = ?";

JdbcTemplate jdbcTemp1 = **new** JdbcTemplate(dataSource);

jdbcTemp.update(sql2, **new** Object[] {"yes", userName});

**return** result;

}

//inserts optional courses

**private** **int** insertOptionalCourses(DegreePlan degreePlan, String userName) {

String sql = "INSERT INTO courses\_dp " + "(userName, courseCatergory, courseName, semesterTaken, cHrs, grade) VALUES (?, ?, ?, ?, ?, ?)";

JdbcTemplate jdbcTemp = **new** JdbcTemplate(dataSource);

String courseCategory = "optional";

for(i=1;i<9;i++){

jdbcTemp.update(sql, **new** Object[] {userName, courseCategory, degreePlan.getOpCoursei().getCourse(), degreePlan.getOpCoursei().getSemesterTaken(), degreePlan.getOpCoursei().getcHrs(), degreePlan.getOpCoursei().getGrade()});

}

**return** result;

}

//inserts core courses

**private** **int** insertCoreCourses(DegreePlan degreePlan, String userName) {

//inserts core course A

String sql = "INSERT INTO courses\_dp " + "(userName, courseCatergory, courseName, semesterTaken, cHrs, grade) VALUES (?, ?, ?, ?, ?, ?)";

JdbcTemplate jdbcTemp = **new** JdbcTemplate(dataSource);

String courseCategory = "courseA";

jdbcTemp.update(sql, **new** Object[] {userName, courseCategory, degreePlan.getCourseA().getCourse(), degreePlan.getCourseA().getSemesterTaken(), degreePlan.getCourseA().getcHrs(), degreePlan.getCourseA().getGrade()});

//inserts core course B

courseCategory = "courseB";

jdbcTemp.update(sql, **new** Object[] {userName, courseCategory, degreePlan.getCourseB().getCourse(), degreePlan.getCourseB().getSemesterTaken(), degreePlan.getCourseB().getcHrs(), degreePlan.getCourseB().getGrade()});

//inserts core course C

courseCategory = "courseC";

jdbcTemp.update(sql, **new** Object[] {userName, courseCategory, degreePlan.getCourseC().getCourse(), degreePlan.getCourseC().getSemesterTaken(), degreePlan.getCourseC().getcHrs(), degreePlan.getCourseC().getGrade()});

//inserts core course D

courseCategory = "courseD";

**int** result = jdbcTemp.update(sql, **new** Object[] {userName, courseCategory, degreePlan.getCourseD().getCourse(), degreePlan.getCourseD().getSemesterTaken(), degreePlan.getCourseD().getcHrs(), degreePlan.getCourseD().getGrade()});

**return** result;

}

/\* this class is called when the student request to view the CSCE Department professors to send requests\*/

//This service class retrieves professor list to display when requested by student

@Service

**public** **class** RetrieveStaffDaoImpl **implements** RetrieveStaffDao{

@Autowired

DataSource dataSource;

**public** List<Register> getStaffDirectory() {

// **TODO** Auto-generated method stub

List<Register> professorList = **new** ArrayList<Register>();

String professor = "professor";

String chair = "chair";

String associatechair = "associatechair";

String sql = "select name, email from register where role='" + professor + "' or role= '" + chair + "' or role= '" + associatechair +"'";

JdbcTemplate jdbctem = **new** JdbcTemplate(dataSource);

professorList = jdbctem.query(sql, **new** ProfessorRowMapper());

**return** professorList;

}

}

/\*This service class has methods to get the professor’s approved students and student’s major professor from the database. These lists are displayed on their corresponding home pages. This class also has method getStudentList() which gets the students list from the database and sends it to the controller for it to be displayed on the view when requested\*/

//This class is gets the advisor’s students and student’s major professor

@Service

**public** **class** RetrieveUsersDaoImpl **implements** RetrieveUsersDao{

@Autowired

DataSource dataSource;

**public** List<Request> getMyStudents(String uName) {

List<Request> myStudents = **new** ArrayList<Request>();

String sql = "select email from register where userName='" + uName + "'" ;

JdbcTemplate jdbctem = **new** JdbcTemplate(dataSource);

String email = jdbctem.queryForObject(sql, String.**class**);

String sql1 = "select \* from majorprofessor where professorEmail='" + email + "'";

JdbcTemplate jdbctem1 = **new** JdbcTemplate(dataSource);

myStudents = jdbctem1.query(sql1, **new** MyStudentRowMapper());

**return** myStudents;

}

**public** List<Request> getMyAdvisors(String userName) {

List<Request> myAdvisors = **new** ArrayList<Request>();

String sql1 = "select professorName, professorEmail from majorprofessor where studentName='" + userName + "'";

JdbcTemplate jdbctem1 = **new** JdbcTemplate(dataSource);

myAdvisors = jdbctem1.query(sql1, **new** MyAdvisorRowMapper());

**return** myAdvisors;

}

**public** List<Register> getStudentList() {

// **TODO** Auto-generated method stub

List<Register> studentList = **new** ArrayList<Register>();

String role = "student";

String sql = "select name, email from register where role='" + role + "'";

JdbcTemplate jdbctem = **new** JdbcTemplate(dataSource);

studentList = jdbctem.query(sql, **new** ProfessorRowMapper());

**return** studentList;

}

/\* This is a logout controller which is called upon any user clicking logout. The logout() method invalidates the user session and log him out \*/[7]

//This controller method will clear the user session and logs out the user

@Controller

**public** **class** LogoutController {

//this method logs out the user

@RequestMapping(value = "/logout", method = RequestMethod.***GET***)

**public** String logout(HttpServletRequest request, HttpServletResponse response) {

HttpSession session = request.getSession(**false**);

**if** (session != **null**) {

session.invalidate();

}

**return** "login";

}

}

References:

[1]. https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html

[2]. https://www.w3schools.com/sql/default.asp

[3]. <https://www.youtube.com/watch?v=Ke7Tr4RgRTs>

[4]. <http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm>

[5]. <https://www.baeldung.com/spring-mvc-tutorial>

[6]. <https://www.journaldev.com/14476/spring-mvc-example>

[7]. https://dzone.com/tutorials/java/spring/spring-mvc-tutorial-1.html