**Final Project Report: Job Search Portal**

**Summary:**Job Search Portal is a Spring MVC web application that allows the communication between the students and employer. Hibernate annotations-based mapping for enhanced database interaction with MySQL.

The framework is completely menu driven and simple to utilize. It gives speedy and exact data. The application empowered students to apply for the employments posted by Employer/Recuriter. The Employer can likewise enlist, post jobs, and get applications for the activity they posted. The application had the CRUD application.

**Functionality:**

Employer Role:

1. Add new jobs to the portal
2. Update the Jobs posted by him only
3. Delete the Jobs
4. Check the applicants applied for the job and can download applicants’ resumes/cover Letter

Student Role:

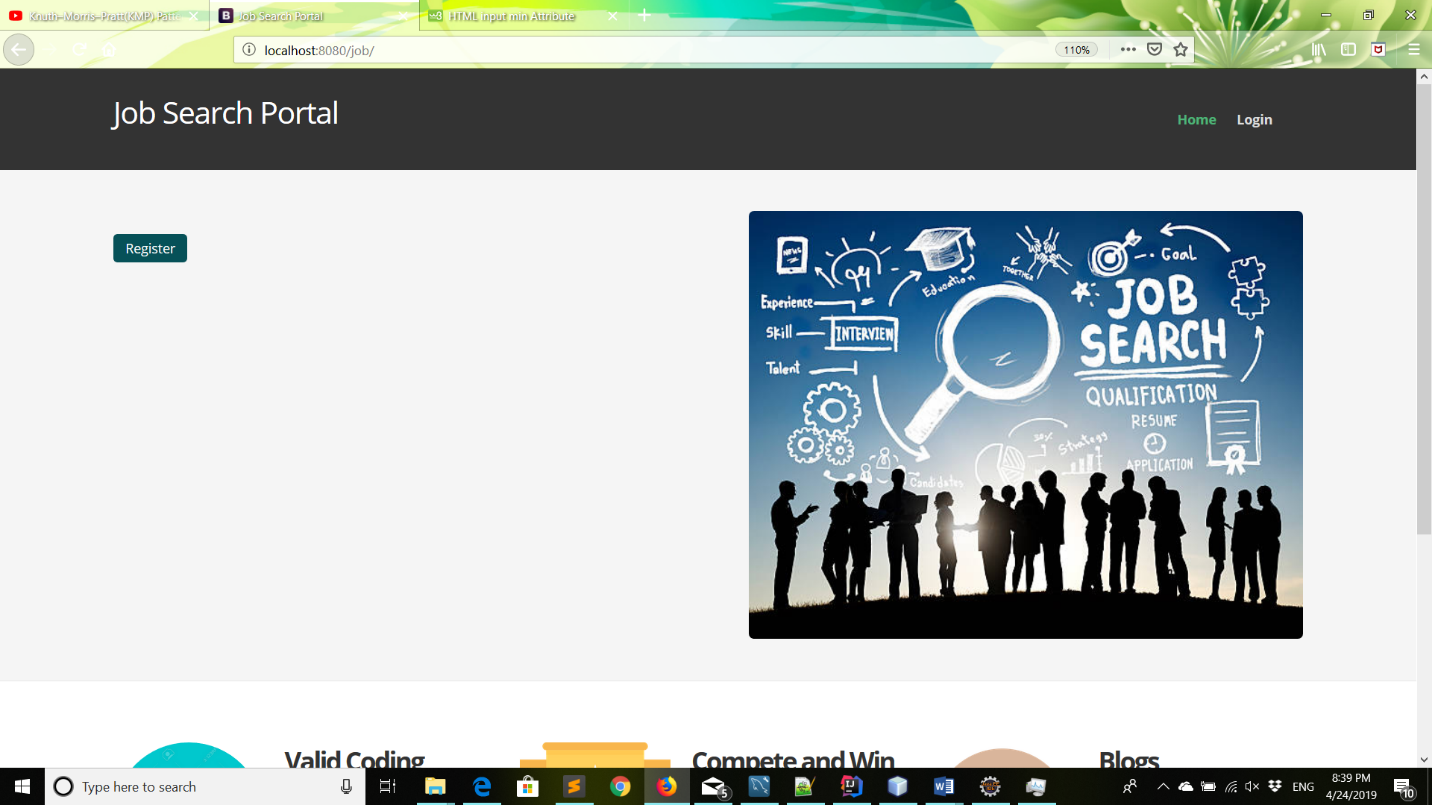
1. View all the jobs posted on the portal
2. Apply for the job
3. Withdraw the application
4. Download the PDF of all the job applied.

**Technologies**

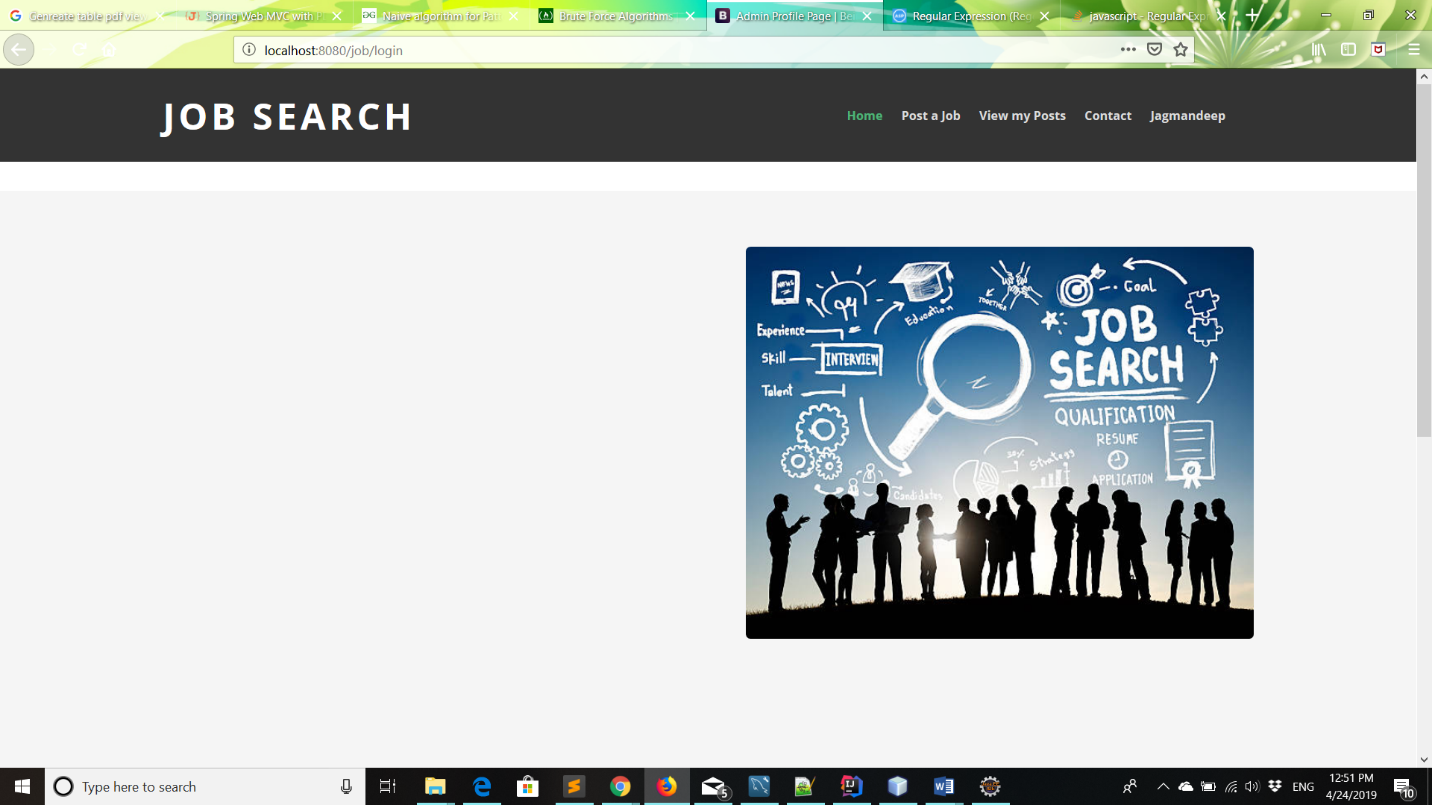
1. Spring MVC, Hibernate, Maven, HTML, CSS, Bootstrap, JQuery
2. Criteria for selection queries and HQL for deletion & Updation
3. Used Associations like oneToMany ,ManyToOne etc.
4. Applied Spring Validators for the validation of data
5. Added Interceptors, XSS Filter and session management techniques to secure web application.
6. Used the Java IO for downloading the resumes and cover letter on the system.
7. Used File Utility for creating dynamic directory structure and folder for each post and applicants.

**Screenshots:**

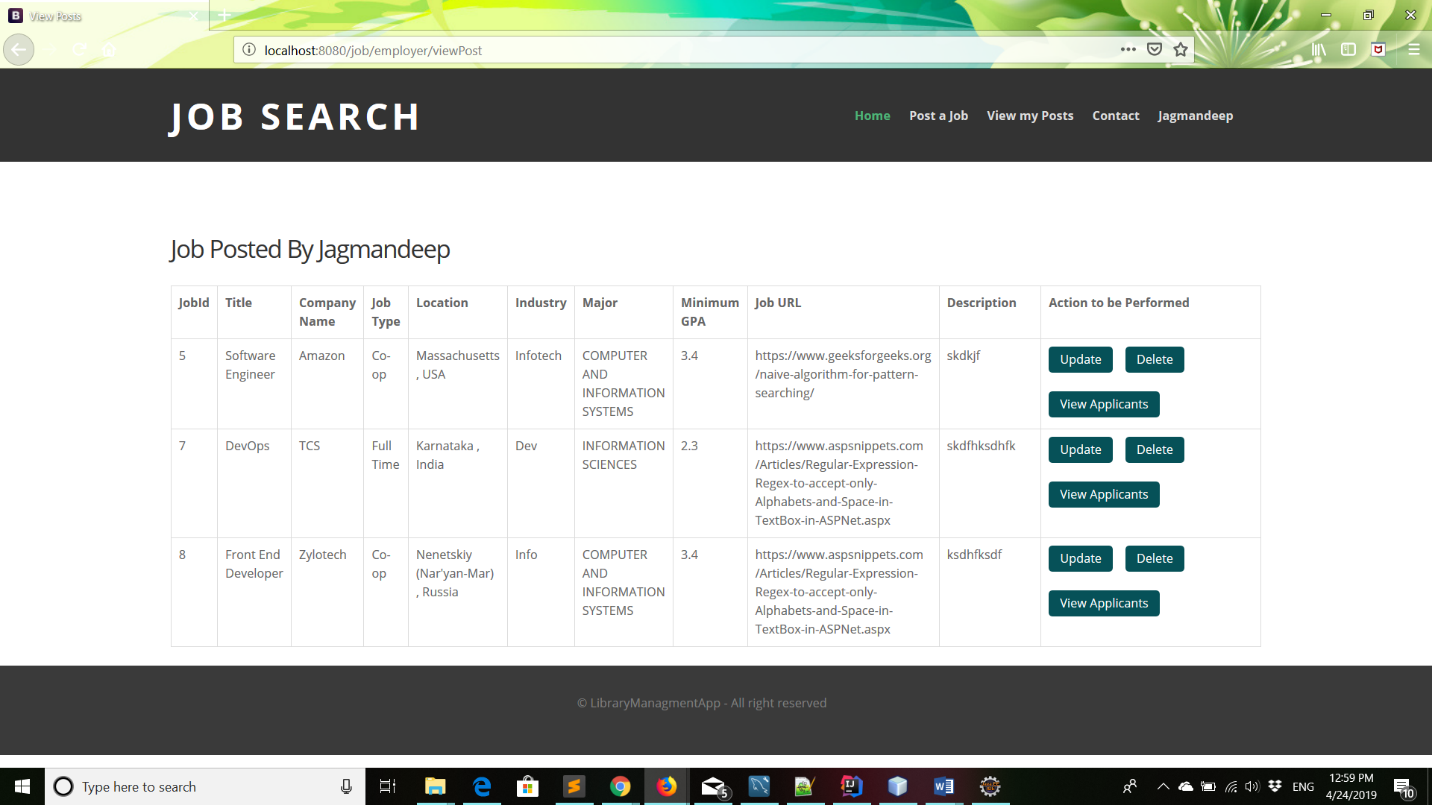
Main Screen:



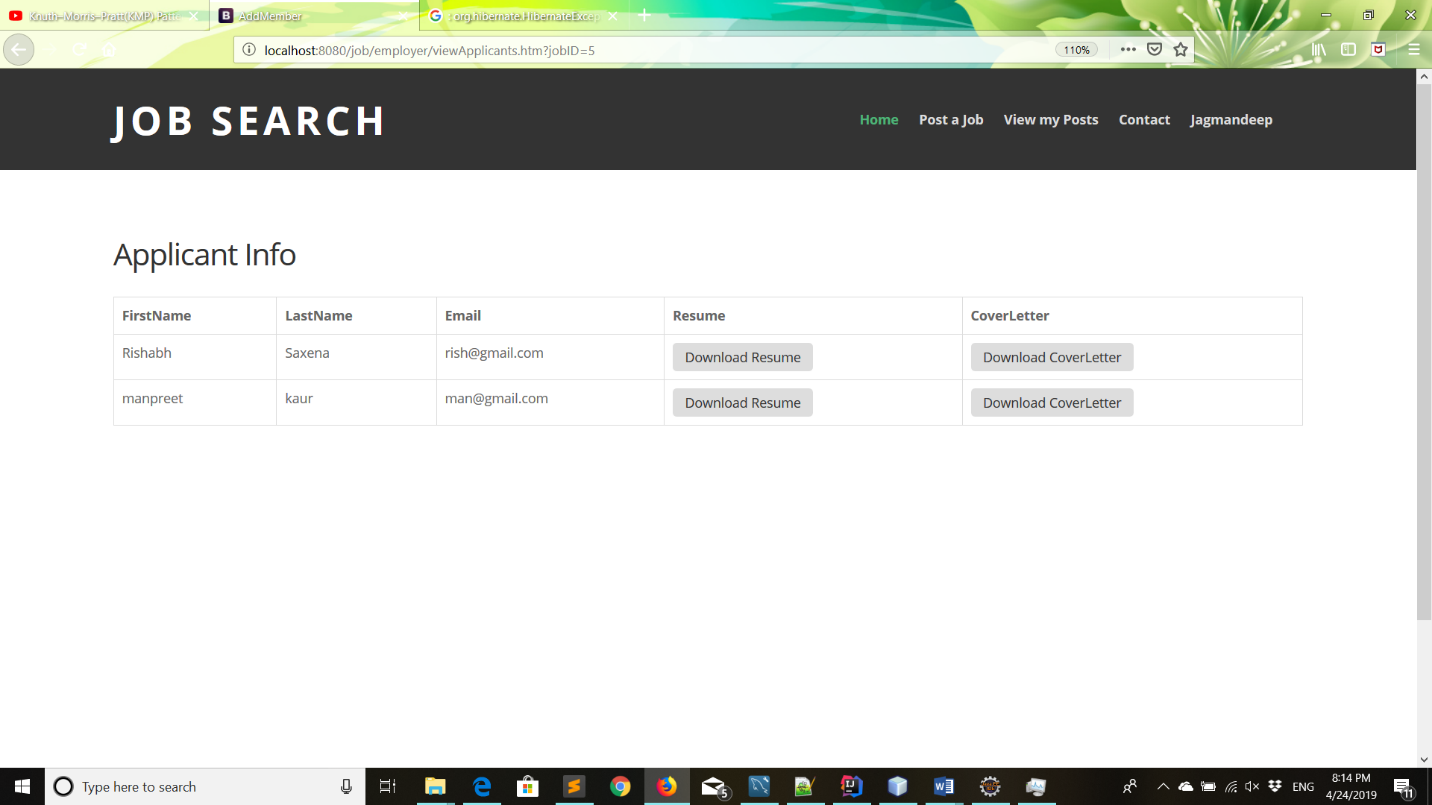
Employer Home:



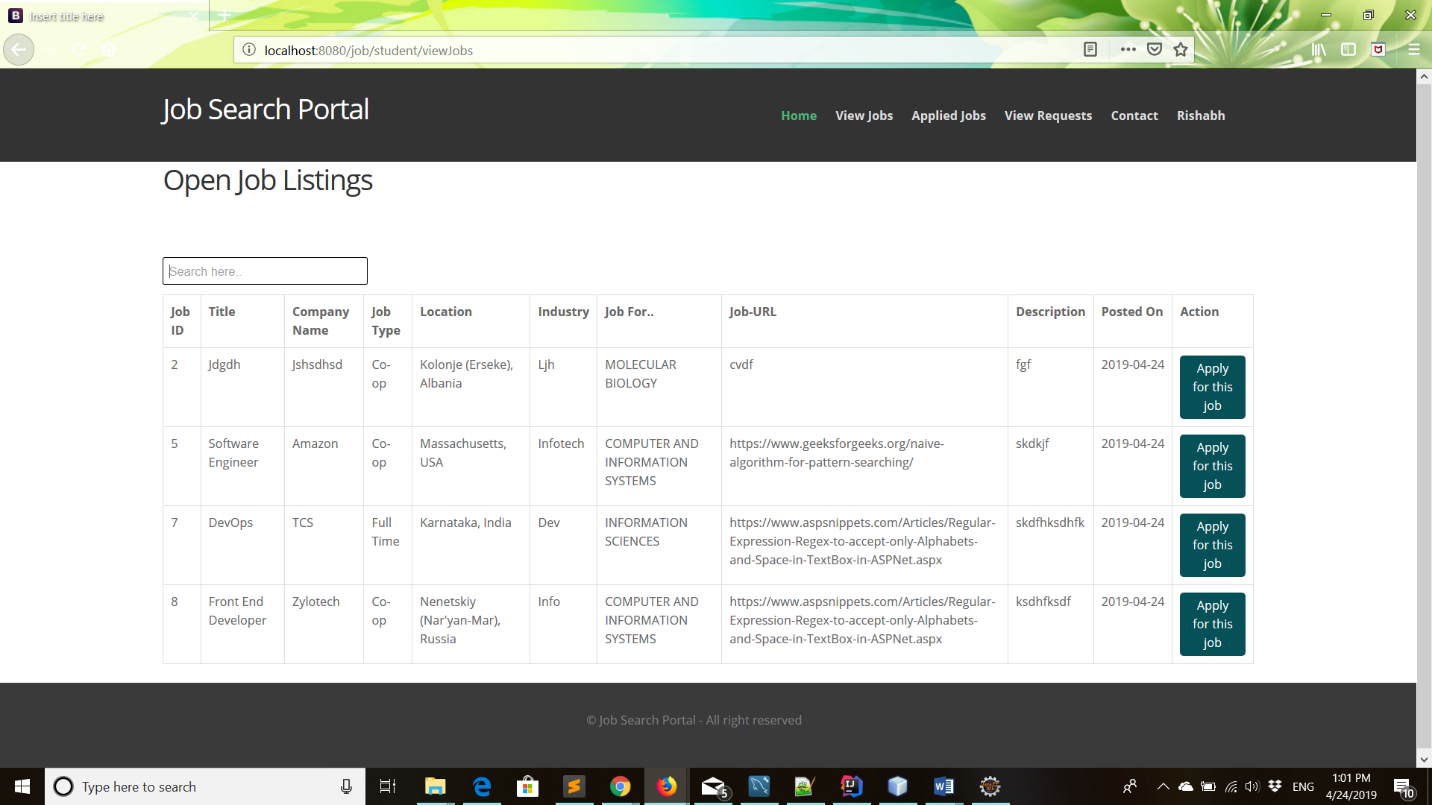
**Employer- Posted Job View:**

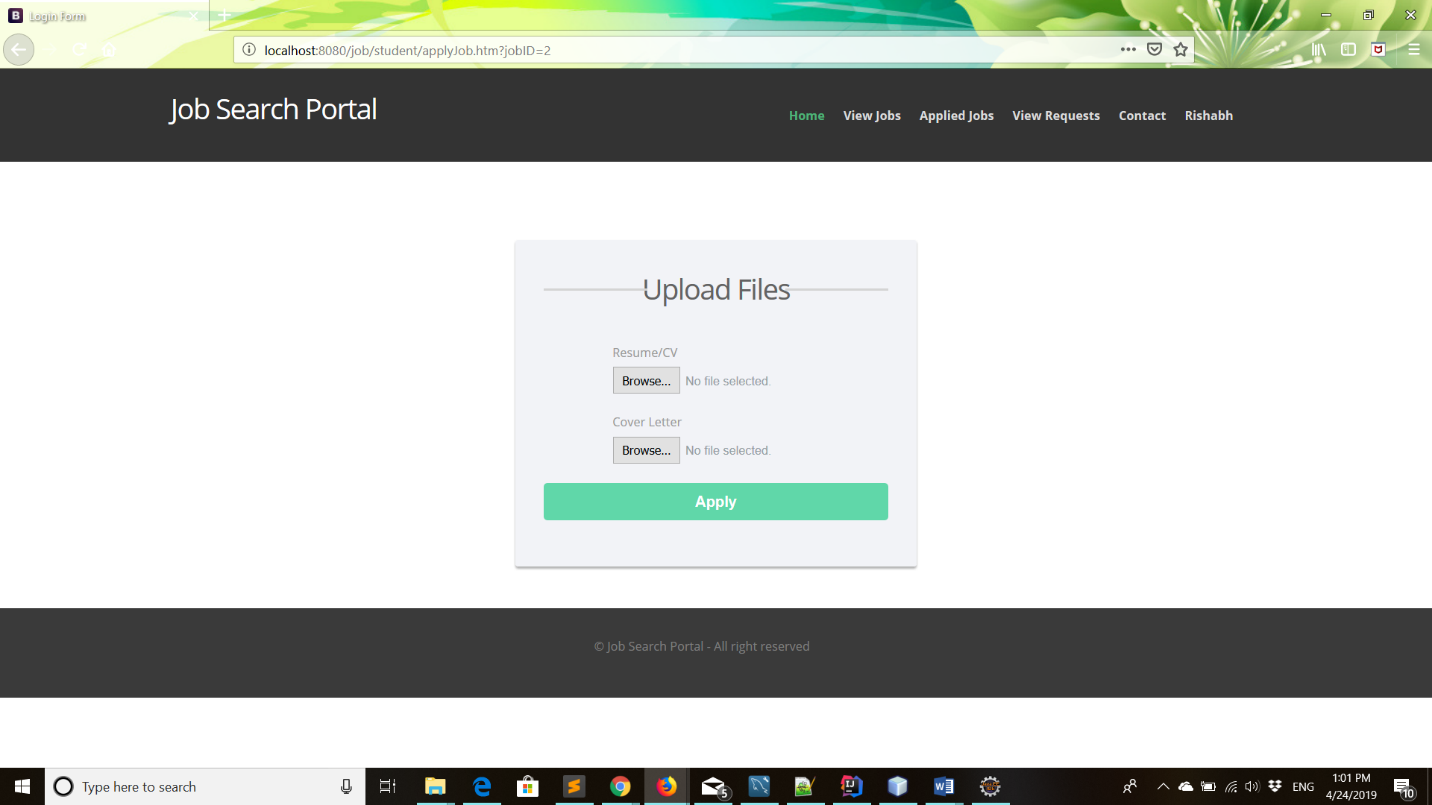


**Admin-View applicant Info:**

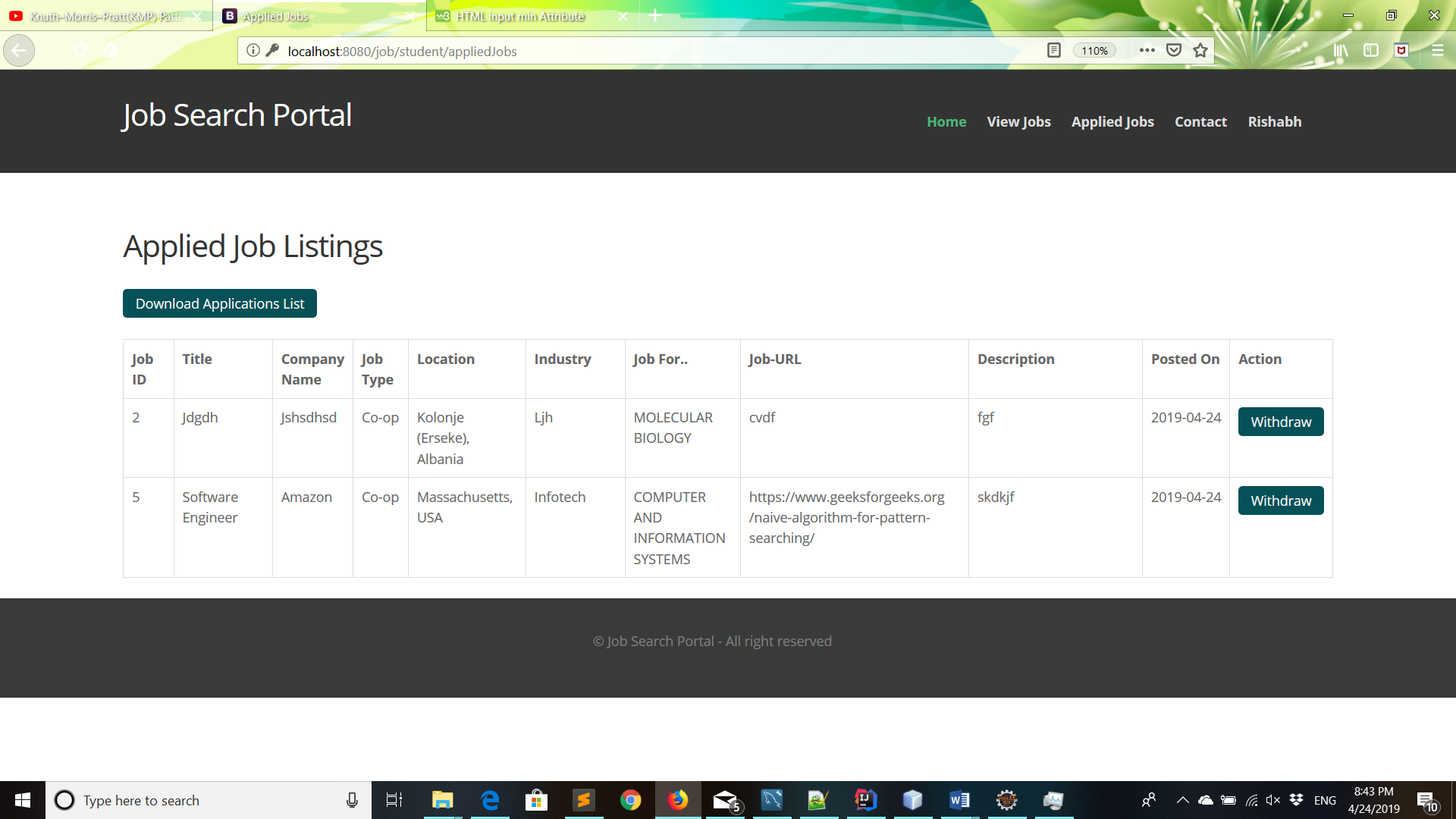


**Student : Apply for job**

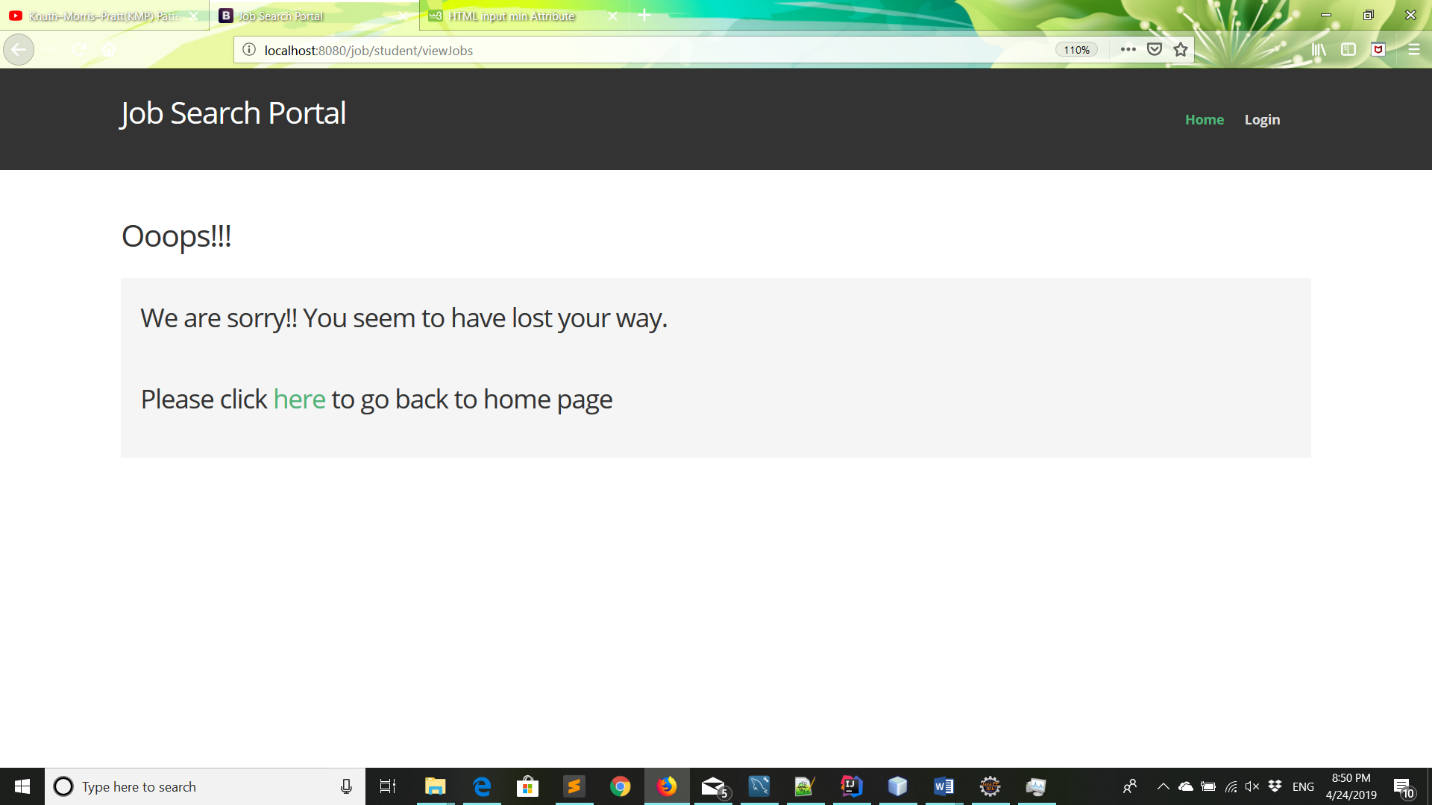




**Student-Withdraw application:**



**Unauthorized Access:**



**POJO**

**Users.java**

package my.web.job.Pojo;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import javax.persistence.Transient;

@Entity

@Table(name = "User")

public class Users {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(name = "userId", unique = true, nullable = false, length = 45)

private Long userId;

@Column(name = "username", unique = true, nullable = false, length = 45)

private String username;

@Column(name = "password", nullable = false, length = 60)

private String password;

@OneToOne(fetch = FetchType.LAZY, mappedBy = "user")

private UserRole userRole;

@Column(name = "firstName")

private String firstName;

@Column(name = "lastName")

private String lastName;

@Transient

private String confirmPassword;

@Column(name = "enabled", nullable = false)

private boolean enabled;

@Column

private String isAllowedAccess;

public Users() {

super();

}

public Users(String username, String password, boolean enabled, String firstName, String lastName) {

super();

this.username = username;

this.password = password;

this.enabled = enabled;

this.firstName = firstName;

this.lastName = lastName;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public boolean isEnabled() {

return enabled;

}

public void setEnabled(boolean enabled) {

this.enabled = enabled;

}

public UserRole getUserRole() {

return userRole;

}

public void setUserRole(UserRole userRole) {

this.userRole = userRole;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getConfirmPassword() {

return confirmPassword;

}

public void setConfirmPassword(String confirmPassword) {

this.confirmPassword = confirmPassword;

}

public String getIsAllowedAccess() {

return isAllowedAccess;

}

public void setIsAllowedAccess(String isAllowedAccess) {

this.isAllowedAccess = isAllowedAccess;

}

public Long getUserId() {

return userId;

}

public void setUserId(Long userId) {

this.userId = userId;

}

}

package my.web.job.Pojo;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.OneToOne;

import javax.persistence.Table;

@Entity

@Table(name = "user\_roles")

public class UserRole {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "user\_role\_id", unique = true)

private Integer userRoleId;

@OneToOne(fetch = FetchType.LAZY, targetEntity = Users.class)

@JoinColumn(name = "userId")

private Users user;

public UserRole() {

}

public UserRole(String role) {

super();

this.role = role;

}

@Column(name = "role", length = 45)

private String role;

public Integer getUserRoleId() {

return userRoleId;

}

public void setUserRoleId(Integer userRoleId) {

this.userRoleId = userRoleId;

}

public Users getUser() {

return user;

}

public void setUser(Users user) {

this.user = user;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

}

package my.web.job.Pojo;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

@Entity

@Table

public class JobDetails {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(name = "jobId", unique=true, nullable = false)

private long jobId;

@ManyToOne(cascade= CascadeType.ALL)

@JoinColumn(name = "userid")

private Users user;

@Column

private String title;

@Column

private String companyName;

@Column

private String typeOfJob;

@Column

private String country;

@Column

private String state;

@Column

private String area;

@Column

private String major;

@Column

private float minGpa ;

@Column

private String jobUrl;

@Column

private String description;

@Temporal(TemporalType.DATE)

private Date postedOn;

@OneToMany(fetch = FetchType.LAZY)

private List<Applications> app = new ArrayList<Applications>();

public JobDetails() {

super();

postedOn = new Date();

}

public long getJobId() {

return jobId;

}

public void setJobId(long jobId) {

this.jobId = jobId;

}

public Users getUser() {

return user;

}

public void setUser(Users user) {

this.user = user;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getCompanyName() {

return companyName;

}

public void setCompanyName(String companyName) {

this.companyName = companyName;

}

public String getTypeOfJob() {

return typeOfJob;

}

public void setTypeOfJob(String typeOfJob) {

this.typeOfJob = typeOfJob;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

public String getArea() {

return area;

}

public void setArea(String area) {

this.area = area;

}

public String getMajor() {

return major;

}

public void setMajor(String major) {

this.major = major;

}

public float getMinGpa() {

return minGpa;

}

public void setMinGpa(float minGpa) {

this.minGpa = minGpa;

}

public String getJobUrl() {

return jobUrl;

}

public void setJobUrl(String jobUrl) {

this.jobUrl = jobUrl;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public Date getPostedOn() {

return postedOn;

}

public void setPostedOn(Date postedOn) {

this.postedOn = postedOn;

}

public List<Applications> getApp() {

return app;

}

public void setApp(List<Applications> app) {

this.app = app;

}

}

package my.web.job.Pojo;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import javax.persistence.Transient;

import org.springframework.web.multipart.MultipartFile;

@Entity

@Table

public class Applications {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "appId", unique=true, nullable = false)

private long appId;

@Transient

private MultipartFile coverLetter;

@Transient

private MultipartFile resume;

@Column

private String resumePath;

@Column

private String coverLetterPath;

@ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)

@JoinColumn(name = "userid")

private Users user;

@ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)

@JoinColumn(name = "jobId")

private JobDetails jobdetails;

public long getAppId() {

return appId;

}

public void setAppId(long appId) {

this.appId = appId;

}

public MultipartFile getResume() {

return resume;

}

public void setResume(MultipartFile resume) {

this.resume = resume;

}

public String getResumePath() {

return resumePath;

}

public void setResumePath(String resumePath) {

this.resumePath = resumePath;

}

public MultipartFile getCoverLetter() {

return coverLetter;

}

public void setCoverLetter(MultipartFile coverLetter) {

this.coverLetter = coverLetter;

}

public String getCoverLetterPath() {

return coverLetterPath;

}

public void setCoverLetterPath(String coverLetterPath) {

this.coverLetterPath = coverLetterPath;

}

public Users getUser() {

return user;

}

public void setUser(Users user) {

this.user = user;

}

public JobDetails getJobdetails() {

return jobdetails;

} public void setJobdetails(JobDetails jobdetails) {

this.jobdetails = jobdetails;

}

}

**DAO**

package my.web.job.Dao;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import org.hibernate.criterion.Restrictions;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import my.web.job.Controller.HomeController;

import my.web.job.Exception.JobDetailsException;

import my.web.job.Exception.JobApplicationException;

import my.web.job.Pojo.Applications;

import my.web.job.Pojo.JobDetails;

import my.web.job.Pojo.Users;

public class EmployeeDao extends DAO {

private static final Logger logger = LoggerFactory.getLogger(EmployeeDao.class);

public JobDetails jobPosting(JobDetails job) throws JobDetailsException {

try {

begin();

getSession().save(job);

commit();

close();

return job;

} catch (HibernateException e) {

rollback();

throw new JobDetailsException("Could not post job " + e);

}

}

public List<JobDetails> viewJobPosting(Users u) throws JobDetailsException {

try {

begin();

Criteria c = getSession().createCriteria(JobDetails.class);

c.add(Restrictions.eq("user", u));

List<JobDetails> l = c.list();

commit();

close();

return l;

} catch (HibernateException e) {

rollback();

throw new JobDetailsException("Error while getting list of jobs for User : " + e);

}

}

public JobDetails getJobDetails(Long id) throws JobDetailsException {

try {

begin();

Criteria c = getSession().createCriteria(JobDetails.class);

c.add(Restrictions.eq("jobId", id));

c.setMaxResults(1);

JobDetails job = (JobDetails) c.uniqueResult();

commit();

close();

return job;

} catch (HibernateException e) {

rollback();

throw new JobDetailsException("Error while getting job Details of id = " + id + " : " + e);

}

}

public boolean deleteJob(Long id) throws JobDetailsException {

try {

begin();

Query q = getSession().createQuery("delete from JobDetails where jobId = :id");

q.setLong("id", id);

q.executeUpdate();

commit();

close();

return true;

} catch (HibernateException e) {

rollback();

throw new JobDetailsException("Error while deleting job Details of id = " + id + " : " + e);

}

}

/\*\*

\*

\* @param job

\* @return

\* @throws JobDetailsException

\*/

public JobDetails updateJob(JobDetails job) throws JobDetailsException {

try {

logger.info("Going to update job with id : " + job.getJobId());

begin();

Query q = getSession().createQuery(

"update JobDetails set title = :t , companyName = :c ,typeOfJob = :type , country = :country , state = :state , industry = :industry , major = :major , minGpa = :minGpa , jobUrl = :jobUrl , description = :description where jobId = :id");

q.setString("t", job.getTitle());

q.setString("c", job.getCompanyName());

q.setString("type", job.getTypeOfJob());

q.setString("country", job.getCountry());

q.setString("state", job.getState());

q.setString("industry", job.getIndustry());

q.setString("major", job.getMajor());

q.setFloat("minGpa", job.getMinGpa());

q.setString("jobUrl", job.getJobUrl());

q.setString("description", job.getDescription());

q.setLong("id", job.getJobId());

q.executeUpdate();

commit();

return job;

} catch (HibernateException e) {

rollback();

throw new JobDetailsException("Error while updating job Details of id = " + job.getJobId() + " : " + e);

}

}

public boolean checkIfUserPostedJob(Users u, long id) throws JobDetailsException {

try {

begin();

Criteria c = getSession().createCriteria(JobDetails.class);

c.add(Restrictions.eq("user", u));

c.add(Restrictions.eq("jobId", id));

c.setMaxResults(1);

JobDetails job = (JobDetails) c.uniqueResult();

if (job != null) {

return true;

} else {

return false;

}

} catch (HibernateException e) {

rollback();

throw new JobDetailsException("Error while updating job Details of id = " + id + " : " + e);

}

}

package my.web.job.Dao;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import org.hibernate.criterion.Restrictions;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import my.web.job.Controller.HomeController;

import my.web.job.Exception.JobApplicationException;

import my.web.job.Pojo.Applications;

import my.web.job.Pojo.JobDetails;

import my.web.job.Pojo.Users;

public class StudentDao extends DAO {

private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

public List<JobDetails> listAllJobs() throws JobApplicationException {

try {

begin();

Query q = getSession().createQuery("from JobDetails");

List<JobDetails> list = q.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new JobApplicationException("Error while getting jobs" + e.getMessage());

}

}

public boolean checkIfAlreadyApplied(Users user, JobDetails jobdetails) throws JobApplicationException {

try {

boolean result = false;

logger.info(

"Going to chekck if user has already applied" + jobdetails.getJobId() + "-----" + user.getUserId());

begin();

Criteria criteria = getSession().createCriteria(Applications.class);

criteria.add(Restrictions.eq("jobdetails", jobdetails));

criteria.add(Restrictions.eq("user", user));

List<Applications> results = criteria.list();

logger.info("list of results" + results);

commit();

if (results.isEmpty() || results == null) {

result = false;

} else {

result = true;

}

logger.info("Has User applied for this job ? " + result);

return result;

} catch (HibernateException e) {

rollback();

throw new JobApplicationException("Error while fetching job for Userid - " + user.getUserId() + " : " + e);

}

}

public void saveFiles(Applications application) throws JobApplicationException {

try {

begin();

getSession().save(application);

commit();

} catch (HibernateException e) {

rollback();

throw new JobApplicationException("Error while saving Job Application : " + e);

}

}

public List<Applications> listAllAppliedJobs(Users user) throws JobApplicationException {

try {

begin();

Criteria c = getSession().createCriteria(Applications.class);

c.add(Restrictions.eq("user", user));

List<Applications> list = c.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new JobApplicationException(

"Error while fetching Job Application details for User - " + user.getUserId() + " : " + e);

}

}

public void deleteApplication(JobDetails job, Users user) throws JobApplicationException {

try {

begin();

Query q = getSession().createQuery("delete from Applications where jobdetails = :job AND user = :user");

q.setParameter("job", job);

q.setParameter("user", user);

q.executeUpdate();

commit();

close();

} catch (HibernateException e) {

rollback();

throw new JobApplicationException("Error while deleting Job Application : " + e);

}

}

public List<Applications> listOfApplicants(long jobId) throws JobApplicationException {

try {

begin();

logger.info("Going to fetch the list of applicants for job id :" + jobId);

Query query = getSession().createQuery("from Applications where jobdetails= '" + jobId + "' ");

List<Applications> list = query.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new JobApplicationException(

"Could not fetch the list for applicants for job Id :" + jobId + " : " + e.getMessage());

}

}

}

package my.web.job.Dao;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import org.hibernate.criterion.Restrictions;

import org.springframework.stereotype.Component;

import my.web.job.Exception.UserException;

import my.web.job.Pojo.UserRole;

import my.web.job.Pojo.Users;

@Component

public class UserDao extends DAO {

public UserDao() {

super();

}

public Users getUser(String email ,String password) throws UserException {

try {

begin();

Query q = getSession().createQuery("from Users where username = :email and password = :password");

q.setString("email", email);

q.setString("password", password);

Users user = (Users) q.uniqueResult();

commit();

return user;

} catch (HibernateException e) {

rollback();

throw new UserException("Could not get user " + email, e);

}

}

public Users register(Users u,String r) throws UserException {

try {

begin();

UserRole role = new UserRole(r);

Users newUser = new Users(u.getUsername(), u.getPassword(), true, u.getFirstName(),u.getLastName());

newUser.setUserRole(role);

role.setUser(newUser);

u.setEnabled(true);

getSession().save(newUser);

getSession().save(role);

commit();

close();

return u;

}catch (HibernateException e) {

rollback();

throw new UserException("Exception while creating user: " + e.getMessage());

}

}

public void delete(Users user) throws UserException {

try {

begin();

getSession().delete(user);

commit();

} catch (HibernateException e) {

rollback();

throw new UserException("Could not delete user " + user.getFirstName() + " " + user.getLastName(), e);

}

}

public Users findByUserName(String email) throws UserException{

try {

begin();

Query q = getSession().createQuery("from Users where username = :e");

q.setString("e" , email);

q.setMaxResults(1);

Users u = (Users) q.uniqueResult();

commit();

return u ;

}catch (HibernateException e) {

rollback();

throw new UserException("Exception while getting user with email " +email + " : " + e.getMessage());

}

}

public List<Users> getPendingUsers() throws UserException{

try {

begin();

Criteria cr = getSession().createCriteria(Users.class);

cr.add(Restrictions.eq("isEnabled" , "N"));

cr.add(Restrictions.eq("role" ,"ROLE\_USER"));

List<Users> u = cr.list();

commit();

return u ;

}catch (HibernateException e) {

rollback();

throw new UserException("Exception while getting user with Status N" + e.getMessage());

}

}

public int updateStatus(String emailId) throws UserException{

try {

begin();

Query query = getSession().createQuery("update Users set isAllowedAccess = :e" +

" where username = :email");

query.setParameter("e", "Y");

query.setParameter("email", emailId);

int result = query.executeUpdate();

commit();

return result ;

}catch (HibernateException e) {

rollback();

throw new UserException("Exception while Updating user Status with email " +emailId + " : " +e.getMessage());

}

}

public Users getUser(long id) throws UserException{

try {

begin();

Criteria cr = getSession().createCriteria(Users.class);

cr.add(Restrictions.eq("userId" , id));

cr.setMaxResults(1);

Users u = (Users) cr.uniqueResult();

commit();

return u;

}catch (HibernateException e) {

rollback();

throw new UserException("Exception while fetching User info for user id -" +id+ " : " +e.getMessage());

}

}}

**Controllers**

**package** my.web.job.Controller;

**import** java.io.IOException;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpSession;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.beans.factory.annotation.Qualifier;

**import** org.springframework.mail.SimpleMailMessage;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.validation.BindingResult;

**import** org.springframework.validation.annotation.Validated;

**import** org.springframework.web.bind.WebDataBinder;

**import** org.springframework.web.bind.annotation.InitBinder;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.servlet.ModelAndView;

**import** my.web.job.Dao.UserDao;

**import** my.web.job.Exception.UserException;

**import** my.web.job.Pojo.Users;

**import** my.web.job.Service.EmailService;

**import** my.web.job.Validators.UsersValidator;

@Controller

**public** **class** AuthenicationController {

@Autowired

@Qualifier("usersValidator")

**private** UsersValidator validator;

@InitBinder("user")

**private** **void** initBinder(WebDataBinder binder) {

binder.setValidator(validator);

}

@Autowired

**private** EmailService emailService;

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

**public** String loginerror(ModelMap model) {

model.addAttribute("error", "true");

**return** "denied";

}

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

**public** ModelAndView displayRegistration(ModelAndView mv) {

mv.addObject("user", **new** Users());

mv.setViewName("register");

**return** mv;

}

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

**public** ModelAndView registerUser(ModelAndView mv, @ModelAttribute("user") @Validated Users users,BindingResult result, UserDao userDao, HttpServletRequest request) **throws** UserException, IOException {

**if** (result.hasErrors()) {

mv.addObject("user", users);

mv.setViewName("register");

**return** mv;

}

Users existingUser;

existingUser = userDao.findByUserName(users.getUsername());

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

mv.addObject("error", "This email is already registered!!");

mv.setViewName("register");

} **else** {

//users.setIsAllowedAccess("N");

String role = request.getParameter("role");

users = userDao.register(users, role);

SimpleMailMessage mailMessage = **new** SimpleMailMessage();

mailMessage.setTo(users.getUsername());

mailMessage.setSubject("Registration Complete !");

mailMessage.setFrom("jagmangrewal9@gmail.com");

mailMessage.setText("You have been registered to the Job Portal ");

emailService.sendEmail(mailMessage);

mv.addObject("emailId", users.getUsername());

mv.setViewName("home");

}

**return** mv;

}

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

**public** ModelAndView displayLogin(ModelAndView mv) {

mv.addObject("user", **new** Users());

mv.setViewName("login");

**return** mv;

}

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

**public** ModelAndView getUser(@ModelAttribute("user") Users user, ModelAndView mv, HttpServletRequest request,

UserDao userDao, BindingResult result) {

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

String email = request.getParameter("username");

String password = request.getParameter("password");

Users existingUser;

**try** {

existingUser = userDao.getUser(email, password);

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

{

**if** (existingUser.getUserRole().getRole().equalsIgnoreCase("ROLE\_STUDENT")) {

mv.addObject("user", existingUser);

mv.setViewName("student");

mv.addObject("role", "student");

session.setAttribute("userSession", existingUser);

} **else** **if** (existingUser.getUserRole().getRole().equalsIgnoreCase("ROLE\_EMPLOYEE")) {

mv.addObject("user", existingUser);

mv.setViewName("employer");

mv.addObject("role", "employer");

session.setAttribute("userSession", existingUser);

}

}

}**else** {

mv.addObject("error", "Incorrect credentials !! Kindly Login with correct credentials");

mv.setViewName("login");

}

} **catch** (UserException e) {

e.printStackTrace();

mv.addObject("message", "Sorry for the inconvience. There is some issue !!");

mv.setViewName("message");

}

**return** mv;

}

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

**public** ModelAndView logout(HttpServletRequest request, ModelAndView mv) {

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

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

Users u = (Users) session.getAttribute("userSession");

**if** (u == **null**) {

mv.addObject("message", "Kindly Login to post a Job");

mv.setViewName("/home");

} **else** {

request.getSession(**false**).invalidate();

mv.setViewName("/home");

}

}

**return** mv;

}

}

package my.web.job.Controller;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

import org.springframework.validation.annotation.Validated;

import org.springframework.web.bind.WebDataBinder;

import org.springframework.web.bind.annotation.InitBinder;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;

import my.web.job.Dao.EmployeeDao;

import my.web.job.Dao.StudentDao;

import my.web.job.Dao.UserDao;

import my.web.job.Exception.JobApplicationException;

import my.web.job.Exception.JobDetailsException;

import my.web.job.Exception.UserException;

import my.web.job.Pojo.Applications;

import my.web.job.Pojo.JobDetails;

import my.web.job.Pojo.Users;

import my.web.job.Util.DownloadFile;

import my.web.job.Validators.JobDetailsValidator;

@Controller

public class EmployerController {

private static final Logger logger = LoggerFactory.getLogger(EmployerController.class);

@Autowired

@Qualifier("jobDetailsValidator")

private JobDetailsValidator validator;

@InitBinder("job")

private void initBinder(WebDataBinder binder) {

binder.setValidator(validator);

}

@RequestMapping(value="/employer/postJob", method=RequestMethod.GET)

public ModelAndView displayJobPosting(ModelAndView mv ,JobDetails job)

{ mv.addObject("job", job);

mv.setViewName("postJob");

return mv;

}

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

public ModelAndView displayEmployerPage(ModelAndView mv ,JobDetails job)

{ mv.addObject("job", job);

mv.setViewName("employer");

return mv;

}

@RequestMapping(value="/employer/postJob", method=RequestMethod.POST)

public ModelAndView postJob(ModelAndView mv ,@ModelAttribute("job") @Validated JobDetails job , BindingResult result , HttpServletRequest request,EmployeeDao empDao) throws JobDetailsException

{

if (result.hasErrors()) {

mv.addObject("job", job);

mv.setViewName("postJob");

return mv;

}

HttpSession session = request.getSession(false);

if(session != null) {

Users u = (Users)session.getAttribute("userSession");

if(u==null) {

mv.addObject("message","Kindly Login to post a Job");

mv.setViewName("error");

}else {

job.setUser(u);

job = empDao.jobPosting(job);

if (job != null) {

mv.addObject("message", "Job has been posted successfully!");

mv.setViewName("employer");

} else {

mv.addObject("message", "Some issue occurred while posting the job.Kindly try again !");

mv.setViewName("postJob");

}

}

}

return mv;

}

@RequestMapping(value="/employer/viewPost", method=RequestMethod.GET)

public ModelAndView viewJobPosting(ModelAndView mv ,HttpServletRequest request,EmployeeDao empDao) throws JobDetailsException

{ HttpSession session = request.getSession(false);

if(session != null) {

Users u = (Users)session.getAttribute("userSession");

if(u==null) {

mv.addObject("message","Kindly Login to view Job");

mv.setViewName("error");

}else {

List<JobDetails> list = empDao.viewJobPosting(u);

mv.addObject("list" ,list);

mv.setViewName("employer-viewPost");

}

}

return mv ;

}

@RequestMapping(value="/employer/deleteJob", method=RequestMethod.GET)

public ModelAndView JobPosting(ModelAndView mv ,HttpServletRequest request,EmployeeDao empDao,StudentDao studentDao,UserDao userDao) throws JobDetailsException, JobApplicationException, UserException

{ HttpSession session = request.getSession(false);

Users u ;

if(session != null) {

u = (Users)session.getAttribute("userSession");

if(u==null) {

mv.addObject("message","Kindly Login to view Job");

mv.setViewName("unauthorizedaccess");

}else {

Long id = Long.parseLong(request.getParameter("jobID"));

logger.info("Going to delete the job with Id :" +id);

boolean isPosted = empDao.checkIfUserPostedJob(u,id);

if(isPosted) {

empDao.deleteJob(id);

mv.addObject("message", "Post has been deleted");

mv.setViewName("employer-message");

}else {

List<JobDetails> list = empDao.viewJobPosting(u);

mv.addObject("list" ,list);

mv.addObject("message", "This job is not posted by you !! Kindly delete your post!!");

mv.setViewName("employer-viewPost");

}

}

}

return mv ;

}

@RequestMapping(value="/employer/updateJobDetails.htm", method=RequestMethod.GET)

public ModelAndView showUpdatePage(ModelAndView mv ,HttpServletRequest request,EmployeeDao empDao , JobDetails job) throws JobDetailsException

{ HttpSession session = request.getSession(false);

if(session != null) {

Users u = (Users)session.getAttribute("userSession");

if(u==null) {

mv.addObject("message","Kindly Login to view Job");

mv.setViewName("error");

}else {

Long id = Long.parseLong(request.getParameter("jobID"));

logger.info("Going to upate the job with Id :" +id);

boolean isPosted = empDao.checkIfUserPostedJob(u,id);

if(isPosted) {

job = empDao.getJobDetails(id);

mv.addObject("job", job);

mv.setViewName("updateJobPost");

}else {

List<JobDetails> list = empDao.viewJobPosting(u);

mv.addObject("list" ,list);

mv.addObject("message", "This job is not posted by you !! So you cant update it");

mv.setViewName("employer-viewPost");

}

}

}

return mv ;

}

@RequestMapping(value="/employer/updateJob", method=RequestMethod.POST)

public ModelAndView UpdateJobPosting(ModelAndView mv ,HttpServletRequest request,EmployeeDao empDao ,@ModelAttribute("job") @Validated JobDetails job ,BindingResult result) throws JobDetailsException

{

if (result.hasErrors()) {

mv.addObject("job", job);

mv.setViewName("updateJobPost");

return mv;

}

HttpSession session = request.getSession(false);

if(session != null) {

Users u = (Users)session.getAttribute("userSession");

if(u==null) {

mv.addObject("message","Kindly Login to view Job");

mv.setViewName("error");

}else {

Long id = Long.parseLong(request.getParameter("id"));

job.setJobId(id);

empDao.updateJob(job);

List<JobDetails> list = empDao.viewJobPosting(u);

mv.addObject("list" ,list);

mv.addObject("message", "The job has been updated !");

mv.setViewName("employer-viewPost");

}

}

return mv ;

}

@RequestMapping(value="/employer/viewApplicants.htm", method=RequestMethod.GET)

public ModelAndView viewApplicantsApplied(ModelAndView mv ,HttpServletRequest request, UserDao userDao ,EmployeeDao empDao , JobDetails job,StudentDao studentDao) throws JobApplicationException, UserException, JobDetailsException

{ HttpSession session = request.getSession(false);

Users u = null;

if(session != null) {

u = (Users)session.getAttribute("userSession");

if(u==null) {

mv.addObject("message","Kindly Login to view Job");

mv.setViewName("error");

}else {

Long id = Long.parseLong(request.getParameter("jobID"));

boolean ifPosted = empDao.checkIfUserPostedJob(u, id) ;

if(ifPosted) {

List<Applications> app = studentDao.listOfApplicants(id);

if(app==null || app.isEmpty()) {

List<JobDetails> list = empDao.viewJobPosting(u);

mv.addObject("list" ,list);

mv.addObject("message", "No applicants for this job");

mv.setViewName("employer-viewPost");

} else {

List<Users> users = new ArrayList<Users>();

for(Applications a : app) {

Long userid = a.getUser().getUserId();

users.add(userDao.getUser(userid));

mv.addObject("listapp",app);

mv.addObject("listuser",users);

mv.setViewName("applicantsInfo");

}

}

}else {

List<JobDetails> list = empDao.viewJobPosting(u);

mv.addObject("list" ,list);

mv.addObject("message", "This job is not posted by you !! So you cannot see the applicants address");

mv.setViewName("employer-viewPost");

}}}

return mv ;

}

@RequestMapping(value="/employer/downloadResume.htm", method=RequestMethod.GET)

public ModelAndView downloadResume(ModelAndView mv ,HttpServletRequest request, DownloadFile file ,UserDao userDao ,EmployeeDao empDao , JobDetails job,StudentDao studentDao) throws JobApplicationException, UserException

{ logger.info("In Download Resume Method");

HttpSession session = request.getSession(false);

if(session != null) {

Users u = (Users)session.getAttribute("userSession");

if(u==null) {

mv.addObject("message","Kindly Login to view Job");

mv.setViewName("error");

}else {

String url = (request.getParameter("path"));

String name = (request.getParameter("email"));

logger.info("Email address is :: " +name);

boolean isFileDownloaded = file.download(name, "resume", url);

if(isFileDownloaded)

{ mv.addObject("message", "FileDownloaded. Kindly check Downloads");

mv.setViewName("employer-message");

}else {

mv.addObject("message", "File could not be downloaded due to some internal issue");

mv.setViewName("employer-message");

}

}

}

return mv ;

}

@RequestMapping(value="/employer/downloadCoverLetter.htm", method=RequestMethod.GET)

public ModelAndView downloadCoverLetter(ModelAndView mv ,HttpServletRequest request, DownloadFile file ,UserDao userDao ,EmployeeDao empDao , JobDetails job,StudentDao studentDao) throws JobApplicationException, UserException

{ logger.info("In DownloadCoverLetterMethod");

HttpSession session = request.getSession(false);

if(session != null) {

Users u = (Users)session.getAttribute("userSession");

if(u==null) {

mv.addObject("message","Kindly Login to view Job");

mv.setViewName("error");

}else {

String url = (request.getParameter("path"));

logger.info(" Url for Coverletter" +url);

String name = (request.getParameter("email"));

boolean isFileDownloaded = file.download(name, "cover", url);

if(isFileDownloaded)

{ mv.addObject("message", "FileDownloaded. Kindly check Downloads");

mv.setViewName("employer-message");

}else {

mv.addObject("message", "File could not be downloaded due to some internal issue");

mv.setViewName("employer-message");

}

}

}

return mv ;

}

}

package my.web.job.Controller;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.multipart.commons.CommonsMultipartFile;

import org.springframework.web.servlet.ModelAndView;

import my.web.job.Dao.EmployeeDao;

import my.web.job.Dao.StudentDao;

import my.web.job.Exception.JobDetailsException;

import my.web.job.Exception.JobApplicationException;

import my.web.job.Pojo.Applications;

import my.web.job.Pojo.JobDetails;

import my.web.job.Pojo.Users;

import my.web.job.Util.DirectoryManager;

@Controller

public class JobApplicationController {

private static final Logger logger = LoggerFactory.getLogger(JobApplicationController.class);

@RequestMapping(value = "/student/viewJobs", method = RequestMethod.GET)

public ModelAndView viewAllJobs(HttpServletRequest request, StudentDao studentDao) throws Exception {

Users users = (Users) request.getSession().getAttribute("userSession");

try {

request.getSession().setAttribute("name", users);

List<JobDetails> jobDetails = studentDao.listAllJobs();

request.getSession().setAttribute("jobID", jobDetails);

return new ModelAndView("student-viewJob", "allJobs", jobDetails);

} catch (JobApplicationException e) {

System.out.println(e.getMessage());

return new ModelAndView("student-message", "message", "Error occured while retrieving jobs");

}

}

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

public ModelAndView displayEmployerPage(ModelAndView mv) {

mv.setViewName("student");

return mv;

}

@RequestMapping(value = "/student/applyJob.htm", method = RequestMethod.GET)

public ModelAndView showApplicationPage(ModelAndView mv, HttpServletRequest request, EmployeeDao empDao,

StudentDao studentDao) throws JobDetailsException, JobApplicationException {

logger.info("\*\*\*\*\*\*\*\*\*\*\*\*\*In apply JOb");

HttpSession session = request.getSession(false);

if (session != null) {

Users u = (Users) session.getAttribute("userSession");

if (u == null) {

mv.addObject("message", "Kindly Login to post a Job");

mv.setViewName("unauthorizedaccess");

} else {

boolean ifAlreadyApplied = true;

List<JobDetails> details = new ArrayList<JobDetails>();

Long id = Long.parseLong(request.getParameter("jobID"));

JobDetails jobdetails = empDao.getJobDetails(id);

ifAlreadyApplied = studentDao.checkIfAlreadyApplied(u, jobdetails);

if (ifAlreadyApplied == true) {

List<JobDetails> jobDetails = studentDao.listAllJobs();

request.getSession().setAttribute("jobID", jobDetails);

mv.addObject("error", "You have already applied for this job");

mv.setViewName("student-viewJob");

} else {

JobDetails job = null;

try {

job = empDao.getJobDetails(id);

} catch (JobDetailsException e) {

return new ModelAndView("student-message", "message",

"Error while getting jobDetails.Please try after some time !!");

}

mv.addObject("job", job);

mv.addObject("id", id);

mv.addObject("files", new Applications());

mv.setViewName("applyJob");

}

}

}

return mv;

}

@RequestMapping(value = "/student/applyJob", method = RequestMethod.POST)

public ModelAndView applyJob(@ModelAttribute("files") Applications app, ModelAndView mv, HttpServletRequest request,

EmployeeDao empDao, StudentDao studentDao)

throws JobApplicationException, JobDetailsException, IllegalStateException, IOException {

logger.info("\*\*\*\*\*\*\*\*\*\*\*\*\*In apply JOb");

HttpSession session = request.getSession(false);

Users u;

if (session != null) {

u = (Users) session.getAttribute("userSession");

if (u == null) {

mv.addObject("message", "Kindly Login to post a Job");

mv.setViewName("student-message");

} else {

Long id = Long.parseLong(request.getParameter("id"));

JobDetails jobdetails = empDao.getJobDetails(id);

DirectoryManager m = new DirectoryManager();

String resumePath = m.createResumeJobFolder(request.getParameter("id"), u);

String coverPath = m.createCoverJobFolder(request.getParameter("id"), u);

CommonsMultipartFile resumeUploaded = (CommonsMultipartFile) app.getResume();

String fileName = resumeUploaded.getOriginalFilename().replace(" ", "\_");

File localFile = new File(resumePath + File.separator + fileName);

resumeUploaded.transferTo(localFile);

app.setResumePath(localFile.getPath());

System.out.println(app.getResumePath());

request.getSession().setAttribute("JobID", jobdetails);

// Going to set path for Cover Letter

CommonsMultipartFile coverLetterUploaded = (CommonsMultipartFile) app.getCoverLetter();

String coverName = coverLetterUploaded.getOriginalFilename().replace(" ", "\_");

try {

File file1 = new File(coverPath, coverName);

FileOutputStream fos = new FileOutputStream(file1);

fos.write(app.getCoverLetter().getBytes());

fos.close();

fos.flush();

} catch (Exception e) {

e.printStackTrace();

}

app.setCoverLetterPath(coverPath + File.separator + coverName);

System.out.println(app.getResumePath());

app.setUser(u);

app.setJobdetails(jobdetails);

studentDao.saveFiles(app);

}

List<JobDetails> list = studentDao.listAllJobs();

mv.addObject("allJobs", list);

}

mv.setViewName("student-viewJob");

return mv;

}

@RequestMapping(value = "/student/appliedJobs", method = RequestMethod.GET)

public ModelAndView getAllAppliedJobs(ModelAndView mv, HttpServletRequest request, Applications application,

EmployeeDao empDao, StudentDao studentDao) throws JobApplicationException, JobDetailsException {

HttpSession session = request.getSession(false);

Users u;

if (session != null) {

u = (Users) session.getAttribute("userSession");

if (u == null) {

mv.addObject("message", "Kindly Login to post a Job");

mv.setViewName("unauthorizedaccess");

} else {

List<Applications> app = studentDao.listAllAppliedJobs(u);

List<JobDetails> jobs = new ArrayList<JobDetails>();

for (Applications a : app) {

long jobId = a.getJobdetails().getJobId();

jobs.add(empDao.getJobDetails(jobId));

}

mv.addObject("jobs", jobs);

mv.setViewName("student-appliedJobs");

}

}

return mv;

}

@RequestMapping(value = "/student/deleteApplication", method = RequestMethod.GET)

public ModelAndView deleteApplication(ModelAndView mv, HttpServletRequest request, Applications application,

EmployeeDao empDao, StudentDao studentDao) throws JobApplicationException, JobDetailsException {

logger.info("In apply Job");

HttpSession session = request.getSession(false);

Users u;

if (session != null) {

u = (Users) session.getAttribute("userSession");

if (u == null) {

mv.addObject("message", "Kindly Login to post a Job");

mv.setViewName("unauthorizedaccess");

} else {

Long id = Long.parseLong(request.getParameter("jobId"));

JobDetails job = empDao.getJobDetails(id);

boolean isApplied = studentDao.checkIfAlreadyApplied(u, job);

if (isApplied) {

studentDao.deleteApplication(job, u);

mv.addObject("message", "Your Application has been withdrawn");

mv.setViewName("student-message");

} else {

mv.addObject("message", "You are trying to delete a applicantion which doesnt belong to you");

mv.setViewName("student-message");

}

}

}

return mv;

}

@RequestMapping(value = "/student/downloadPdf", method = RequestMethod.GET)

public ModelAndView downloadPdf(ModelAndView mv, HttpServletRequest request, Applications application,

EmployeeDao empDao, StudentDao studentDao) throws JobApplicationException, JobDetailsException {

logger.info("In PDF downloader");

List<JobDetails> jobs = new ArrayList<JobDetails>();

HttpSession session = request.getSession(false);

Users u;

if (session != null) {

u = (Users) session.getAttribute("userSession");

if (u == null) {

mv.addObject("message", "Kindly Login to post a Job");

mv.setViewName("unauthorizedaccess");

} else {

List<Applications> app = studentDao.listAllAppliedJobs(u);

for (Applications a : app) {

long jobId = a.getJobdetails().getJobId();

jobs.add(empDao.getJobDetails(jobId));

}

}

}

return new ModelAndView("pdfView", "jobs", jobs);

}

}

**VIEWS**

applicationInfo.jsp

<body>

<c:set var=*"path"* value=*"* ${pageContext.request.contextPath}*"*/>

<section id=*"maincontent"*>

<div class=*"container"* style="margin-top: *10%*">

<h2>Applicant Info</h2>

<div style="color: *red*">${message}</div>

<table class=*"table table-bordered"*>

<tr>

<td><b>FirstName</b></td>

<td><b>LastName</b></td>

<td><b>Email</b>

<td><b>Resume</b>

<td><b>CoverLetter</b>

</tr>

<c:forEach var=*"list"* items=*"*${listapp}*"*>

<tr>

<td>${ *list*.user.firstName}</td>

<td>${ *list*.user.lastName}</td>

<td>${ *list*.user.username}</td>

<td><a href=*"*${path}*/employer/downloadResume.htm?path=*${ *list*.resumePath}*&email=*${ *list*.user.username}*"* class=*"btn btn-default"* role=*"button"* aria-pressed=*"true"* >Download Resume</a></td>

<td><a href=*"*${path}*/employer/downloadCoverLetter.htm?path=*${ *list*.coverLetterPath}*&email=*${ *list*.user.username}*"* class=*"btn btn-default"* role=*"button"* aria-pressed=*"true"* >Download CoverLetter</a></td>

</tr>

</c:forEach>

</table>

</div>

</section>

</body>

Applyjob.jsp

<body>

<c:set var=*"contextPath"* value=*"*${pageContext.request.contextPath}*"* />

<div class=*"signup-form"* style="margin-top: *10%*">

<form:form method=*"post"* name=*"contact-form"* action=*"*${contextPath}*/student/applyJob"* modelAttribute=*"files"* enctype=*"multipart/form-data"*>

<h2>Upload Files</h2>

<div style="color: *red*">${error}</div>

<input type=*"hidden"* name=*"id"* value=*"*${id}*"*>

<br>

<div class=*"form-group"*>

<label>Resume/CV</label>

<form:input type=*"file"* class=*"form-control"* path=*"resume"* placeholder=*"Upload CV"* required=*"required"* style=*"height : 30px"*/>

</div>

<div class=*"form-group"*>

<label>Cover Letter</label>

<form:input type=*"file"* class=*"form-control"* path=*"coverLetter"* placeholder=*"Upload CoverLetter"* required=*"required"* style=*"height : 30px"* />

</div>

<div class=*"form-group"* style="margin-left:*0%*">

<button type=*"submit"* class=*"btn btn-success btn-lg btn-block"* style="height : *40px*">Apply</button>

</div>

</form:form>

</div>

</body>

Message.jsp

<body>

<div class=*"container"* style="margin-top: *10%*">

<h3>Message</h3>

<div class=*"well"*>

<h4>${message}</h4><br>

</div>

</div>

Employer-viewPost

<div class=*"container"* style="margin-top: *10%*">

<h3>Job Posted By ${sessionScope.userSession.firstName} </h3>

<div style="color: *red*">${message}</div>

<table class=*"table table-bordered"*>

<tr>

<td><b>JobId</b></td>

<td><b>Title</b></td>

<td><b>Company Name</b></td>

<td><b>Job Type</b></td>

<td><b>Location</b>

<td><b>Industry</b>

<td><b>Major</b>

<td><b>Minimum GPA</b></td>

<td style="width: *5%*;"><b>Job URL</b></td>

<td style="width: *10%*;"><b>Description</b></td>

<td style="width: *25%*;"><b>Action to be Performed</b></td>

</tr>

<c:forEach var=*"list"* items=*"*${list}*"*>

<tr>

<td>${list.jobId}</td>

<td>${list.title}</td>

<td>${list.companyName}</td>

<td>${list.typeOfJob}</td>

<td>${list.state} , ${list.country}</td>

<td>${list.industry}</td>

<td>${list.major}</td>

<td>${list.minGpa}</td>

<td>${list.jobUrl}</td>

<td>${list.description}</td>

<td><a href=*"*${path}*/employer/updateJobDetails.htm?jobID=*${list.jobId}*"* class=*"btn btn-primary btn-sm active"* role=*"button"* aria-pressed=*"true"* >Update</a>

&nbsp;&nbsp;

<a href=*"*${path}*/employer/deleteJob.htm?jobID=*${list.jobId}*"* class=*"btn btn-primary btn-sm active"* role=*"button"* >Delete</a>

&nbsp;&nbsp;<br></br>

<a href=*"*${path}*/employer/viewApplicants.htm?jobID=*${list.jobId}*"* class=*"btn btn-primary btn-sm active"* role=*"button"* aria-pressed=*"true"* >View Applicants</a>

</tr>

</c:forEach>

</table>

</div>

</section>

Login.jsp

<div class=*"signup-form"* style="margin-top: *10%*">

<form:form method=*"post"* name=*"contact-form"* action=*"*${contextPath}*/login"* modelAttribute=*"user"*>

<h2>Login</h2>

<div style="color: *red*">${error}</div>

<br>

<div class=*"form-group"*>

<form:input type=*"email"* class=*"form-control"* path=*"username"* placeholder=*"Email"* required=*"required"* style=*"height : 30px"*/>

</div>

<div class=*"form-group"*>

<form:input type=*"password"* class=*"form-control"* path=*"password"* placeholder=*"Password"* required=*"required"* style=*"height : 30px"* />

</div>

<div class=*"form-group"* style="margin-left:*0%*">

<button type=*"submit"* class=*"btn btn-success btn-lg btn-block"* style="height : *40px*">Sign in</button>

</div>

</form:form>

<div class=*"text-center"*>Don't have an account? <a href=*"*${contextPath}*/register"*>Register Here!!</a></div>

</div>

postJob.jsp

<div class=*"signup-form"* style="margin-top: *10%*">

<form:form name=*"contact-form"* method=*"post"* action=*"*${contextPath}*/employer/postJob"* modelAttribute=*"job"*>

<h2>Post Job</h2>

<div class=*"form-group"*>

<form:input type=*"text"* class=*"form-control"* path=*"title"* placeholder=*"Title"* required=*"required"* style=*"height : 30px"*/>

<form:errors path=*"title"* cssClass=*"error"*/>

</div>

<div class=*"form-group"*>

<form:input type=*"text"* class=*"form-control"* path=*"companyName"* placeholder=*"Company Name"* required=*"required"* style=*"height : 30px"*/>

<form:errors path=*"companyName"* cssClass=*"error"*/>

</div>

<div class=*" form-group row-of-radio"*>

<p>Job Type <span class=*"asterisk"*>&#42;</span><p>

<div>

<form:radiobutton required=*"required"* value=*"Full Time"* path=*"typeOfJob"* id=*"job\_type\_full"*/>

<label for=*"job\_type\_full"*>Full Time</label>

</div>

<div>

<form:radiobutton required=*"required"* value=*"Co-op"* path=*"typeOfJob"* id=*"job\_type\_coop"*/>

<label for=*"job\_type\_coop"*>Co-op</label>

</div>

<div>

<form:radiobutton required=*"required"* value=*"Contract"* path=*"typeOfJob"* id=*"job\_type\_contract"*/>

<label for=*"job\_type\_contract"*>Contract</label>

</div>

</div>

<div class=*"form-group"*>

<div class=*"locationDropdown"*>

<script type= *"text/javascript"* src = *"*${contextPath}*/resources/assets/js/countries.js"*></script>

<select required=*"required"* id=*"job\_type\_country"* name =*"country"*></select> <br>

<label for=*"job\_type\_state"*>State</label><br>

<select required=*"required"* name =*"state"* id =*"job\_type\_state"*></select>

</div>

</div>

<script type=*"text/javascript"*>

populateCountries("job\_type\_country", "job\_type\_state"); // first parameter is id of country drop-down and second parameter is id of state drop-down

</script>

<div class=*"form-group"*>

<form:input type=*"text"* class=*"form-control"* id=*"industry"* path=*"industry"* placeholder=*"industry"* required=*"required"* style=*"height : 30px"* onkeyup=*"checkPass()"* />

<form:errors path=*"industry"* cssClass=*"error"*/>

</div>

<div class=*"form-group"*>

<script src = *"*${contextPath}*/resources/assets/js/majorsList.js"*></script>

<select required=*"required"* id=*"majorCat"* name =*"majCategory"*></select> <br><br>

<label for=*"majorList"*>Select the major applicable for this job <span class=*"asterisk"*>&#42;</span></label><br>

<select required=*"required"* name =*"major"* id =*"majorList"*></select>

</div>

<script type=*"text/javascript"*>

populateCategories("majorCat", "majorList"); // first parameter is id of country drop-down and second parameter is id of state drop-down

</script>

<div class=*"form-group"*>

<form:input type=*"number"* class=*"form-control"* path=*"minGpa"* step =*"0.1"* min=*"0"* max=*"4"* placeholder=*"minGpa"* required=*"required"* style=*"height : 30px"* onkeyup=*"checkPass()"* />

</div>

<div class=*"form-group"*>

<form:input type=*"text"* class=*"form-control"* path=*"jobUrl"* placeholder=*"job\_url"* required=*"required"* style=*"height : 30px"* onkeyup=*"checkPass()"* />

</div>

<div class=*"form-group"*>

<form:input type=*"text"* class=*"form-control"* path=*"description"* placeholder=*"description"* required=*"required"* style=*"height : 30px"* onkeyup=*"checkPass()"* />

</div>

<div class=*"form-group"* style="margin-left:*0%*">

<button type=*"submit"* class=*"btn btn-success btn-lg btn-block"* style="height : *40px*">Post a Job</button>

</div>

</form:form>

Register.jsp

<div class=*"signup-form"* style="margin-top: *10%*">

<form:form name=*"contact-form"* method=*"post"* action=*"*${contextPath}*/register"* modelAttribute=*"user"*>

<h2>Register</h2>

<div style="color: *red*">${error}</div>

<div class=*"form-group"*>

<form:input type=*"text"* class=*"form-control"* path=*"firstName"* placeholder=*"First Name"* required=*"required"* style=*"height : 30px"*/>

<form:errors path=*"firstName"* cssClass=*"error"*/>

</div>

<div class=*"form-group"*>

<form:input type=*"text"* class=*"form-control"* path=*"lastName"* placeholder=*"Last Name"* required=*"required"* style=*"height : 30px"*/>

<form:errors path=*"lastName"* cssClass=*"error"*/>

</div>

<div class=*"form-group"*>

<form:input type=*"email"* class=*"form-control"* path=*"username"* placeholder=*"Email"* required=*"required"* style=*"height : 30px"*/>

<form:errors path=*"username"* cssClass=*"error"*/>

</div>

<div class=*"form-group"*>

<form:input type=*"password"* class=*"form-control"* id=*"password"* path=*"password"* placeholder=*"Password"* required=*"required"* style=*"height : 30px"* />

<form:errors path=*"password"* cssClass=*"error"*/>

</div>

<div class=*"form-group"*>

<form:input type=*"password"* class=*"form-control"* id=*"c\_password"* path=*"confirmPassword"* placeholder=*"Confirm Password"* required=*"required"* style=*"height : 30px"* onkeyup=*"checkPass()"* />

<p id=*"confirmMessage"* ></p>

</div>

<div class=*"form-group"*>

<label>Select what best suits you:</label><br>

<input type=*"radio"* name=*"role"* value=*"ROLE\_STUDENT"*/> I am a student &nbsp&nbsp

<input type=*"radio"* name=*"role"* value=*"ROLE\_EMPLOYEE"*/> I am an employer <br> <br>

</div>

<div class=*"form-group"* style="margin-left:*0%*">

<label class=*"checkbox-inline"*><input type=*"checkbox"* required=*"required"*> I accept the <a href=*"#"*>Terms of Use</a> &amp; <a href=*"#"*>Privacy Policy</a></label>

</div>

<div class=*"form-group"* style="margin-left:*0%*">

<button type=*"submit"* class=*"btn btn-success btn-lg btn-block"* style="height : *40px*" >Register Now</button>

</div>

</form:form>

<div class=*"text-center"*>Already have an account? <a href=*"*${path}*/login"*>Sign in</a></div>

</div>

</body>

Student-appliedJob.jsp

<div class=*"container"* style="margin-top: *9%*">

<h2>Applied Job Listings</h2>

<a href=*"*${path}*/student/downloadPdf"* class=*"btn btn-primary btn-sm active"* role=*"button"* aria-pressed=*"true"*>Download Applications List</a>

<br><br>

<table id = *"jobTable"* class = *"table table-bordered"*>

<%-- <input type="hidden" name = "jobID" value = "${j.id}" /> --%>

<tr class=*"header"*>

<th>Job ID</th>

<th >Title</th>

<th >Company Name</th>

<th >Job Type</th>

<th>Location</th>

<th>Industry</th>

<th>Job For..</th>

<th>Job-URL</th>

<th>Description</th>

<th>Posted On</th>

<tjh>Action</th>

</tr>

<c:forEach var=*"list"* items=*"*${jobs}*"*>

<tr>

<td>${list.jobId}</td>

<td>${list.title}</td>

<td>${list.companyName}</td>

<td>${list.typeOfJob}</td>

<td>${list.state}, ${list.country}</td>

<td>${list.industry}</td>

<td>${list.major}</td>

<c:if test= *"*${**empty** list.jobUrl}*"*>

<td>No link available. Please check the company's website</td></c:if>

<c:if test= *"*${**not empty** list.jobUrl}*"*>

<td>${list.jobUrl}</td></c:if>

<c:if test= *"*${**empty** list.description}*"*>

<td>No description provided</td></c:if>

<c:if test= *"*${**not empty** list.description}*"*>

<td>${list.description}</td></c:if>

<td>${list.postedOn}</td>

<td><a href=*"*${path}*/student/deleteApplication?jobId=*${list.jobId}*"* class=*"btn btn-primary btn-sm active"* role=*"button"* aria-pressed=*"true"*>Withdraw</a></td>

</tr>

</c:forEach>

</table>

Student-viewJobs

<div class=*"container"* style="margin-top: *8%*">

<h2>Open Job Listings</h2>

<br><br>

<div style="color: *red*">${error}</div>

<input type=*"text"* id=*"search"* placeholder=*"Search here.."* style="border: *1px solid black*;">

<table id = *"jobTable"* class = *"table table-bordered"*>

<%-- <input type="hidden" name = "jobID" value = "${j.id}" /> --%>

<tr class=*"header"*>

<th >Job ID</th>

<th >Title</th>

<th>Company Name</th>

<th>Job Type</th>

<th>Location</th>

<th>Industry</th>

<th>Job For..</th>

<th>Job-URL</th>

<th>Description</th>

<th>Posted On</th>

<th>Action</th>

</tr>

<c:forEach var=*"list"* items=*"*${allJobs}*"*>

<tr>

<td>${list.jobId}</td>

<td>${list.title}</td>

<td>${list.companyName}</td>

<td>${list.typeOfJob}</td>

<td>${list.state}, ${list.country}</td>

<td>${list.industry}</td>

<td>${list.major}</td>

<c:if test= *"*${**empty** list.jobUrl}*"*>

<td>No link available. Please check the company's website</td></c:if>

<c:if test= *"*${**not empty** list.jobUrl}*"*>

<td>${list.jobUrl}</td></c:if>

<c:if test= *"*${**empty** list.description}*"*>

<td>No description provided</td></c:if>

<c:if test= *"*${**not empty** list.description}*"*>

<td>${list.description}</td></c:if>

<td>${list.postedOn}</td>

<td><a href=*"*${path}*/student/applyJob.htm?jobID=*${list.jobId}*"* class=*"btn btn-primary btn-sm active"* role=*"button"* aria-pressed=*"true"*>Apply for this job</a></td>

</tr>

</c:forEach>

</table>

</div>

Unauthorized access:

<div class=*"container"* style="margin-top: *10%*">

<h2>Ooops!!!</h2>

<div class=*"well"*>

<h3>We are sorry!! You seem to have lost your way.</h3><br>

<h3>Please click <a href=*"/job"*>here</a> to go back to home page</h3>

</div>

</div>