

WEB TOOLS AND DEVELOPMENT

FINAL PROJECT TOPIC: NORTHEASTERN UNIVERSITY BLACKBOARD

NAME: KRISH PRAVIN JAIN

NU ID: 001881885

EMAIL: JAIN.KR@HUSKY.NEU.EDU

INDEX:

- 1. SUMMARY**
- 2. FUNCTIONALITIES PERFORMED**
- 3. TECHNOLOGIES USED**
- 4. ROLES AND THEIR FUNCTIONALITIES**
- 5. SCREENSHOTS OF PROJECT VIEW**
- 6. APPENDIX**
 - CONTROLLER CODE**
 - POJO CLASS CODE**
 - CONTACT APPLICATION USING AJAX AND NODE JS CODE**
- 7. READ ME FILE**

SUMMARY:

The project build here is NORTHEASTERN UNIVERSITY EDUCATIONAL PORTAL called BLACKBOARD. It is a responsive website. Northeastern's Blackboard is the learning management system. In my project I have tried to implement the following functionalities to enhance teaching technology like adding courses for the semesters, posting jobs and internships for students, creating assignments and sending notification to the students via blackboard email services. It allows the student to select their courses and drop them from variety of courses available which suits their interest and track. Students can research about various jobs opening via the portal. Students have the facilities to download the course content and submit their assignment via the northeastern blackboard. It helps students stay informed and involved, which results in a better educational experience. The Blackboard has the additional functionality of managing the Phonebook which has important contacts of all professors, students, NUPD etc.

FUNCTIONALITIES PERFORMED:

- **REGISTER THE USER USING CAPTCHA AND EMAIL VERIFICATION METHOD:** A user will have to fill the Registration form with a valid email id in order to use the blackboard. The blackboard will send a verification link to the user. On successful verification the user will get login access to the portal.
- **LOGIN:** On entering successful email id and password the user will get access to the blackboard or else he can click the **FORGET PASSWORD functionality** so the blackboard will send the registered user his password.
- **UPDATING THE COURSES AND SYLLABUS:** The teaching faculty can update the courses and syllabus for the students on the blackboard.
- **UPDATING THE JOB/INTERNSHIP POSTINGS:** The coop faculty can update the number of various jobs/internships for the students.
- **UPDATING THE ASSIGNMENTS, COURSE MATERIALS AND IMPORTANT DOCUMENTS FOR THE STUDENTS:** The faculty members will be able to upload the documents which will be available for students to view and download for better enhancement in the education process.
- **SENDING NOTIFICATIONS TO ALL THE STUDENTS VIA BLACKBOARD NOTIFY FEATURE USING EMAIL:** The faculty members can notify the students about various activities via the email feature of the blackboard.
- **SEARCH THE COURSES TO ADD AND DROP YOUR DESIRED COURSES:** The students have the functionality to search for various courses depending upon the name of the course, the college under which the course comes and via the professor's name too. The student can add and drop the courses.

- **SEARCH FOR JOBS/INTERNSHIPS:** The students have the functionality to search for various jobs and internship depending upon the name of the company, the payment and the employer.
- **UPLOADING THE ASSIGNMENTS:** The students can upload their assignments via the blackboard itself. This feature enables smooth functioning as the student is not needed to submit it by coming on campus.
- **SENDING EMAIL:** The users can send email via the blackboard.
- **THE CONTACT FEATURE:** The contact feature is an interesting feature giving the power to each user to add, update and view contact. EMERGENCY CONTACT like NUPD , HOSPITAL, PROFESSORS, STUDENTS are maintained on the global Phonebook application of NEU.

TECHNOLOGIES USED:

The entire application is build using Spring MVC and Hibernate on STS IDE. The project is hosted on localhost 8000. I have used Node Js along with AJAX to host the contact application on localhost 3000 as well. The contacts are updated, deleted maintained in JSON file.

DEPENDENCY USED:

- Hibernate-core(5.2.2.Final)
- Mysql-connector-java(5.1.14)
- Commons-email(1.5)
- Botdetect-jsp20(4.0beta3.2)
- Commons-io(1.3)
- Commons-fileupload(1.2.2)
- Csvjdbc(1.0.28)
- Poi(3.6)

ROLES AND THEIR FUNCTIONALITY:

1. THE BLACKBOARD SERVER:

- REGISTER THE USER USING CAPTCHA AND EMAIL VERIFICATION METHOD
- LOGIN

2. THE TEACHING FACULTY:

- UPDATING THE COURSES AND SYLLABUS
- UPDATING THE JOB/INTERNSHIP POSTINGS
- UPDATING THE ASSIGNMENTS, COURSE MATERIALS AND IMPORTANT DOCUMENTS FOR THE STUDENTS
- SENDING NOTIFICATIONS TO ALL THE STUDENTS VIA BLACKBOARD NOTIFY FEATURE USING EMAIL

3. THE STUDENTS:

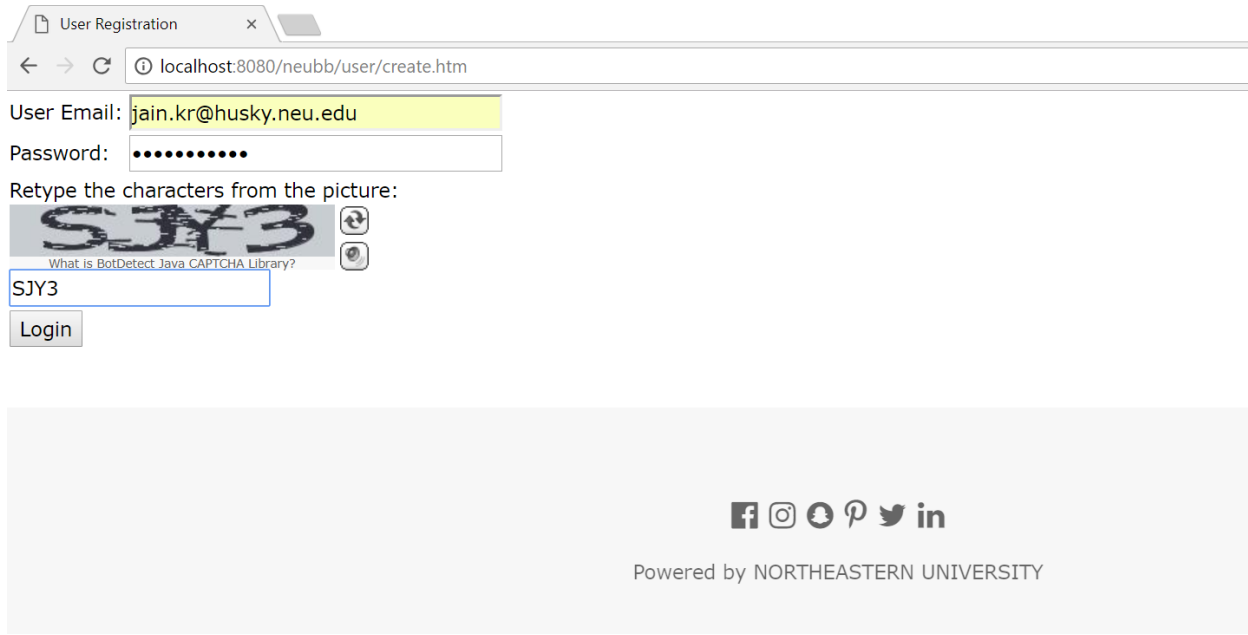
- SEARCH THE COURSES TO ADD AND DROP YOUR DESIRED COURSES
- SEARCH FOR JOBS/INTERNSHIPS
- UPLOADING THE ASSIGNMENTS

ADDITIONAL FEATURE:

- SENDING EMAIL
- THE CONTACT PHONEBOOK APPLICATION

SCREENSHOTS:

- **REGISTER THE USER VIA CAPTCHA CODE AUTHENTICATION**

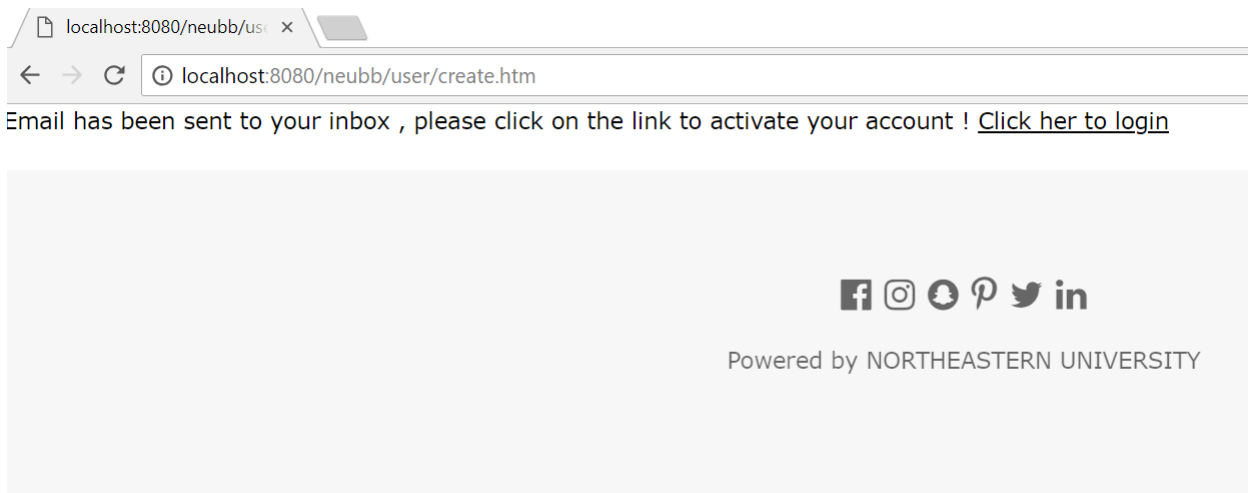


The screenshot shows a web browser window with the title "User Registration" and the URL "localhost:8080/neubb/user/create.htm". The form contains the following fields and elements:

- User Email:** A text input field containing "jain.kr@husky.neu.edu".
- Password:** A password input field with masked characters ".....".
- Retype the characters from the picture:** A label above a captcha image showing the characters "SJY3".
- Captcha Input:** A text input field containing "SJY3".
- Login:** A button located below the captcha input field.

Below the form, there is a footer section with social media icons for Facebook, Instagram, Snapchat, Pinterest, Twitter, and LinkedIn, followed by the text "Powered by NORTHEASTERN UNIVERSITY".

- **VERIFICATION LINK IS SENT FOR ACTIVATION OF THE USER ACCOUNT**



The screenshot shows the same web browser window as the previous one, but the form fields are no longer visible. Instead, a message is displayed:

Email has been sent to your inbox , please click on the link to activate your account ! [Click her to login](#)

Below the message, there is a footer section with social media icons for Facebook, Instagram, Snapchat, Pinterest, Twitter, and LinkedIn, followed by the text "Powered by NORTHEASTERN UNIVERSITY".

- **USER LOGIN**

Login page

localhost:8080/neubb/user/validateemail.htm?email=jain.kr@husky.neu.edu&key1=495900&key2=2347544

Email: jain.kr@husky.neu.edu

Password:

☒ Remember me

Login

[Forgot password?](#) [Register User](#)

Facebook Instagram YouTube Pinterest Twitter LinkedIn

Powered by NORTHEASTERN UNIVERSITY

Type here to search

7:44 PM 4/26/2018

- **IF USER FORGET PASSWORD THAN THE BLACKBOARD SERVER SENDS USER THE PASSWORD**

Login page

localhost:8080/neubb/user/forgotpassword.htm

Enter your email: jain.kr@husky.neu.edu

Retype the characters from the picture:

3EUX

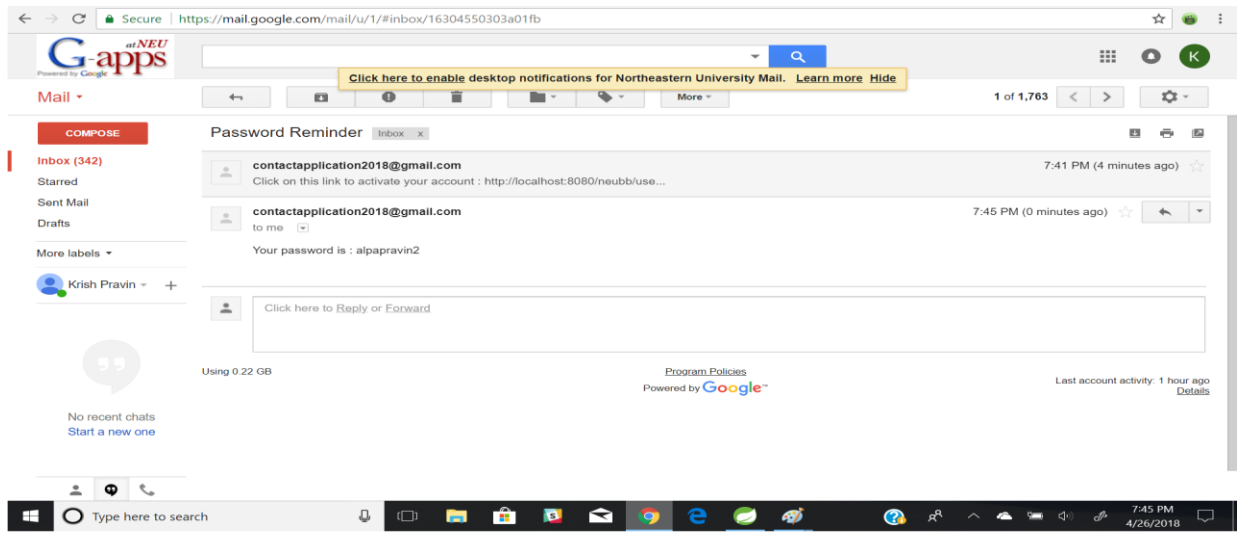
Submit

Facebook Instagram YouTube Pinterest Twitter LinkedIn

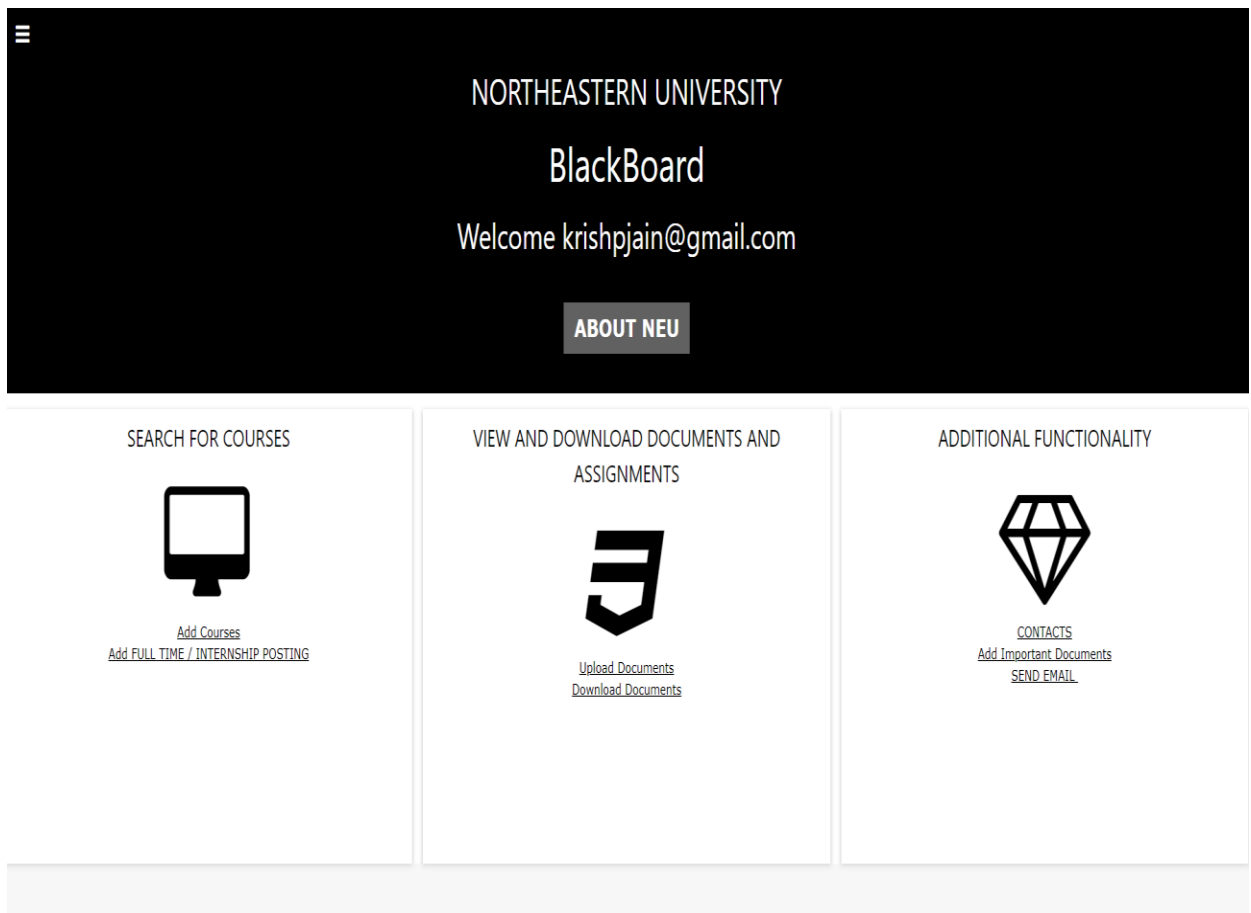
Powered by NORTHEASTERN UNIVERSITY

Type here to search

7:44 PM 4/26/2018



- **THE TEACHING FACULTY DASHBOARD (Here Prof email id is krishpjain@gmail.com)**



- **THE TEACHING FACULTY ADD COURSES SYLLABUS FUNCTIONALITY**

Add new Course

Title:

Professor:

College:

Credits:

Type:

[Go to Dashboard](#)



Powered by NORTHEASTERN UNIVERSITY

- **THE COOP FACULTY CAN ADD JOBS/INTERNSHIPS ON BLACKBOARD**

Add new JOB WORK

COMPANY NAME:

EMPLOYER NAME:

DURATION:

SALARY:

JOB TYPE:

[Go to Dashboard](#)



Powered by NORTHEASTERN UNIVERSITY

- **THE TEACHING FACULTY CAN DOWNLOAD THE ASSIGNMENTS UPLOADED BY STUDENTS**

[Download WEBTOOLASSIGNMENT](#)

[Download ASSIGNMENT1.pdf](#)

[Go to Dashboard](#)



Powered by NORTHEASTERN UNIVERSITY

localhost:8080/neubb/user/download/downloads/pdf/web_flow_tutorial.pdf

web_flow_tutorial (....pdf) ^

Show all



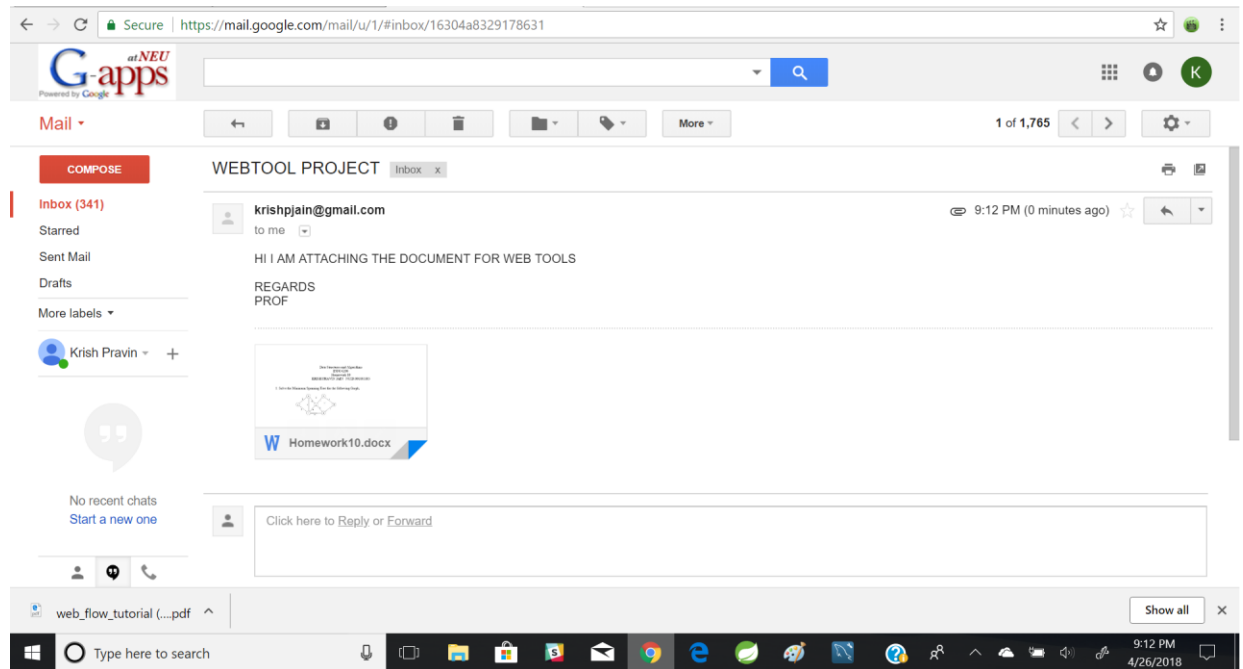
- **THE TEACHING FACULTY WILL SEND NOTIFICATIONS AND ASSIGNMENTS TO STUDENTS VIA BLACKBOARD EMAIL FEATURE**

BLACKBOARD NOTIFY VIA EMAIL

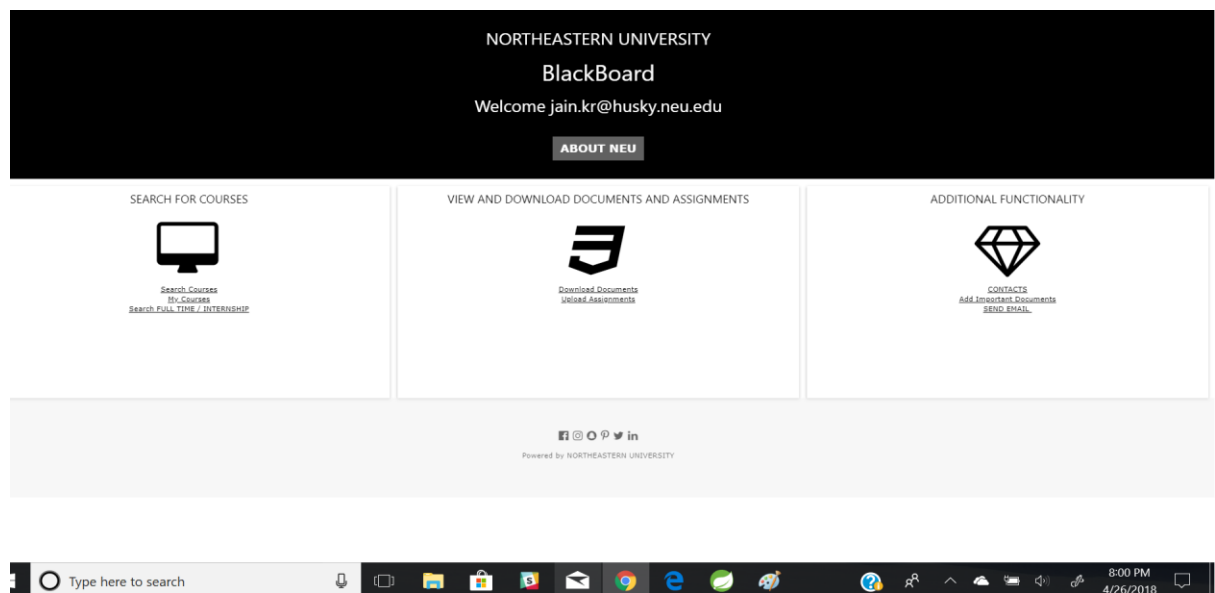
| | |
|--------------|--|
| Email To: | <input type="text" value="jain.kr@husky.neu.edu"/> |
| Subject: | <input type="text" value="WEBTOOL PROJECT"/> |
| Message: | <div>HI I AM ATTACHING THE DOCUMENT FOR WEB TOOLS REGARDS PROF</div> |
| Attach file: | <div>Choose File Homework10.docx</div> |

Send E-mail

(THE TEACHER KRISHPJAIN@GMAIL.COM SENDS NOTIFICATION ASSIGNMENT TO STUDENT JAIN.KR@HUSKY.NEU.EDU)



• **THE STUDENT DASHBOARD**



- **THE STUDENT CAN SEARCH, SELECT AND DELETE THE COURSES FROM VARIOUS COURSES**

Login page x Inbox (15,220) - alpaprav x Password Reminder - jain x Insert title here x





localhost:8080/neubb/user/search/

Search By Title: Search By PROFESSOR: Search By COLLEGE:

Keyword: COE

search

[Go to Dashboard](#)

Powered by NORTHEASTERN UNIVERSITY

Type here to search





localhost:8080/neubb/user/search/searchcourses

| ID | TITLE | PROFESSOR | COLLEGE | CREDITS | TYPE |
|--|-------------------------------------|-----------|---------|---------|---------|
| <input checked="" type="checkbox"/> 5 | Web Tools | Yusuf | COE | 4 | Lecture |
| <input checked="" type="checkbox"/> 6 | Web Design | Amulthan | COE | 4 | Lecture |
| <input checked="" type="checkbox"/> 7 | Application Engineering Development | Bugrara | COE | 4 | Lecture |
| <input type="checkbox"/> 12 | ADS | NIKBROWN | COE | 4 | LECTURE |
| <input type="checkbox"/> 14 | PSA | MOHSEN | COE | 4 | LECTURE |
| <input type="checkbox"/> 19 | Y | y | COE | y | y |
| <input checked="" type="checkbox"/> 23 | DBMS | Yusuf | COE | 4 | LECTURE |

[Add selected courses](#)

[Go to Main Page](#)

[Go to Dashboard](#)

Powered by NORTHEASTERN UNIVERSITY

←

→

↻

localhost:8080/neubb/user/course/save

☐ 6 Web Design Amulthan COE 4 Lecture

☐ 23 DBMS Yusuf COE 4 LECTURE

☐ 5 Web Tools Yusuf COE 4 Lecture

☐ 7 Application Engineering Development Bugarra COE 4 Lecture

Delete selected courses

[Go to Dashboard](#)

f

@

o

p

t

in

Powered by NORTHEASTERN UNIVERSITY

- **THE STUDENT CAN SEARCH JOBS/INTERNSHIPS POSTED ON BLACKBOARD**

Search By Company: ☒ Search By Payment: ☐ Search By Type: ☐

Keyword:

search

[Go to Dashboard](#)

f

@

o

p

t

in

Powered by NORTHEASTERN UNIVERSITY

←

→

↻

localhost:8080/neubb/user/worksearch/searchworks

| ID | COMPANY | EMPLOYER | DURATION | PAYMENT | TYPE1 |
|----|---------|-----------|----------|---------|------------|
| 11 | GOOGLE | MR GOOGLE | 4 MONTHS | 50000\$ | INTERNSHIP |

[Go to Dashboard](#)

f

@

o

p

t

in

Powered by NORTHEASTERN UNIVERSITY

- **THE STUDENT UPLOADS THE ASSIGNMENT VIA BLACKBAORD**

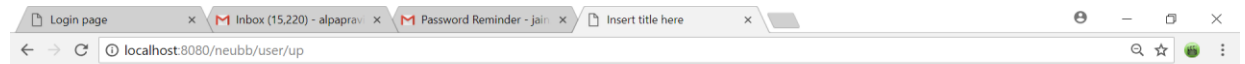
NAME:

TYPE: web_flow_tutorial.pdf

[Go to Dashboard](#)



Powered by NORTHEASTERN UNIVERSITY



SUCCESS

web_flow_tutorial.pdf web_flow_tutorial.pdf web_flow_tutorial.pdf web_flow_tutorial.pdf

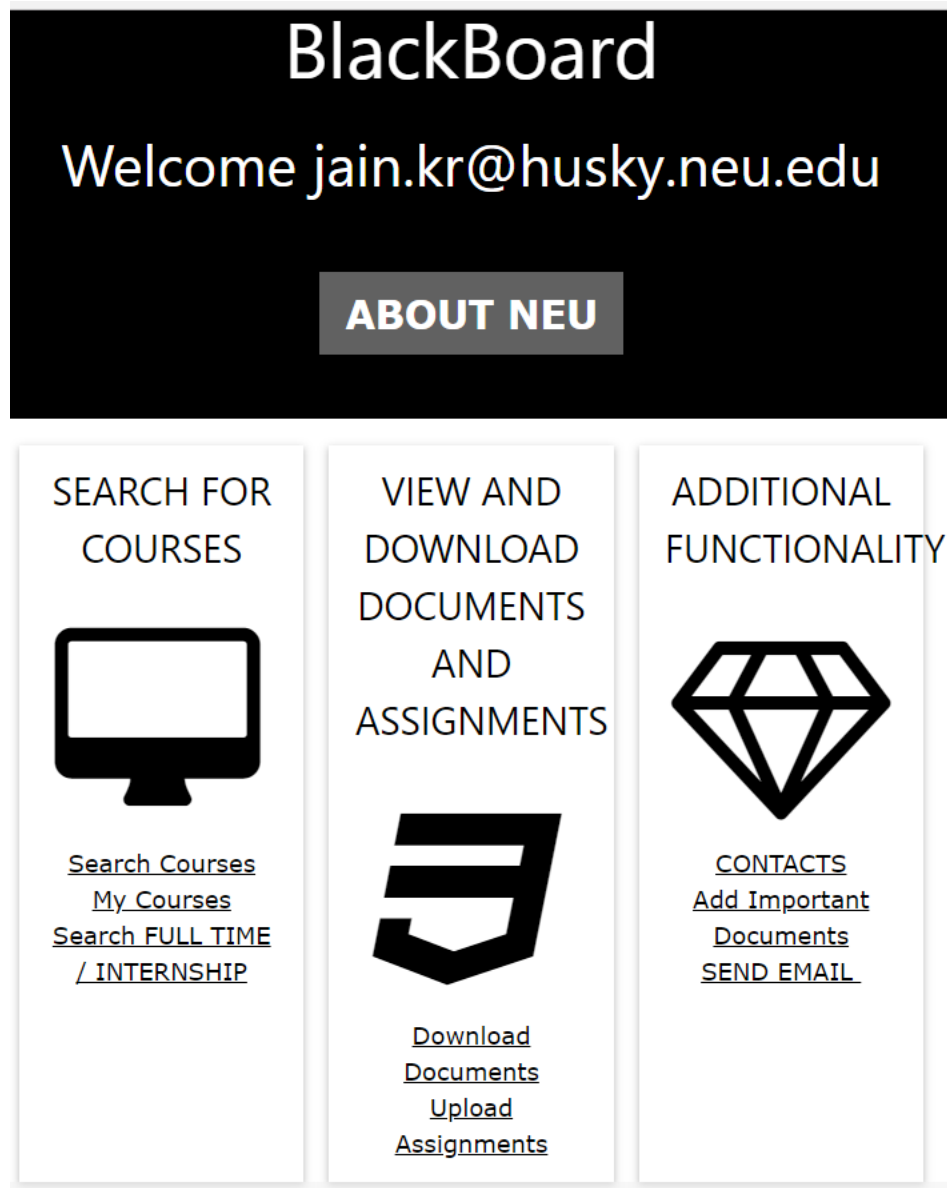
[Go to Dashboard](#)



Powered by NORTHEASTERN UNIVERSITY



- THE BLACKBOARD IS RESPONSIVE THE LAYOUTS ARE SHOWN BELOW: TABLET VIEW FOLLOWED BY MOBILE VIEW



Welcome
jain.kr@husky.neu.edu





ABOUT NEU

SEARCH FOR COURSES



[Search Courses](#)
[My Courses](#)
[Search FULL TIME / INTERNSHIP](#)

- THE ADDITIONAL PHONEBOOK APP APPLICATION FOR CONTACTS OF NUPD HOSPITAL PROFESSOR STUDENTS:
- PROFESSOR CONTACT_

| MY PHONEBOOK APP! | | Add New Contact |
|--|--------------------|---|
|  First Name: YUSUF Last Name: PROF Contact No.: 5123647982 Email: krishpjain@gmail.com Address: NEU BOSTON MA 02215 Date Of Birth: 08/09/1970 | Contacts List | |
| | STUDENT KRISH JAIN |  |
| | PROF YUSUF |  |
| | NUPD NORTHEASTERN |  |
| | | |
| | | |
| | | |



STUDENT CONTACT

MY PHONEBOOK APP!

Add New Contact




| | | |
|---|--------------------|---|
| First Name: KRISH JAIN | STUDENT KRISH JAIN |  |
| Last Name: STUDENT | PROF YUSUF |  |
| Contact No.: 5123647777 | NUPD NORTHEASTERN |  |
| Email: alpapravun24@gmail.com | | |
| Address: 75 saint alphonusus street BOSTON MA 02120 | | |
| Date Of Birth: 08/04/1994 | | |

web_flow_tutorial (...pdf)

Show all

Type here to search



9:22 PM
4/26/2018

NUPD CONTACT

MY PHONEBOOK APP!

Add New Contact



| | | |
|-------------------------------|--------------------|---|
| First Name: NORTHEASTERN | STUDENT KRISH JAIN |  |
| Last Name: NUPD | PROF YUSUF |  |
| Contact No.: 6176663498 | NUPD NORTHEASTERN |  |
| Email: alpapravun24@gmail.com | | |
| Address: NEU BOSTON MA 02215 | | |
| Date Of Birth: 11/11/2000 | | |

web_flow_tutorial (...pdf)

Show all

Type here to search



9:22 PM
4/26/2018

APPENDIX:

CONTROLLER SOURCE CODE:

USER CONTROLLER:

```
package com.me.neubb.controller;
```

```
import java.io.File;
```

```
import java.io.IOException;
```

```
import java.util.ArrayList;
```

```
import java.util.HashSet;
```

```
import java.util.List;
```

```
import java.util.Random;
```

```
import java.util.Set;
```

```
import javax.security.auth.login.Configuration;
```

```
import javax.servlet.http.Cookie;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpSession;
```

```
import org.apache.commons.mail.DefaultAuthenticator;
```

```
import org.apache.commons.mail.Email;
```

```
import org.apache.commons.mail.EmailException;
```

```
import org.apache.commons.mail.SimpleEmail;
```

```
import org.hibernate.query.Query;
```

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
```

```
import org.springframework.stereotype.Controller;
```

```
import org.springframework.ui.ModelMap;
```

```
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.multipart.MultipartFile;
```

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

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

```
import com.captcha.botdetect.web.servlet.Captcha;
```

```
import com.me.neubb.dao.DAO;
```

```
import com.me.neubb.dao.UserDAO;
```

```
import com.me.neubb.pojo.Assignment;
import com.me.neubb.pojo.Course;
import com.me.neubb.pojo.UploadedFile;
import com.me.neubb.pojo.User;
```

```
@Controller
```

```
public class UserController extends DAO{
```

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

```
    @RequestMapping(value = "/user/login.htm", method = RequestMethod.GET)
```

```
    public String showLoginForm(HttpServletRequest request) {
```

```
        //<%
```

```
        Cookie[] ck = request.getCookies();
```

```
        ///String act =request.getParameter("action");
```

```
        if (ck != null) {
```

```
            for (int i = 0; i < ck.length; i++) {
```

```
                String name = ck[i].getName();
```

```
                String value = ck[i].getValue();
```

```
                if (name.equals("username")) {
```

```
                    request.getSession().setAttribute("username", ck[i].getValue());
```

```
                    return "user-dashboard";
```

```
                }
```

```
            }
```

```
        }
```

```
        else {
```

```
            return "user-login";
```

```
        }
```

```
        return "user-login";
```

```
    }
```

```
    @RequestMapping(value = "/user/login.htm", method = RequestMethod.POST)
```

```
    public String handleLoginForm(HttpServletRequest request, HttpServletResponse response,
```

```

        UserDao userDao, ModelMap map) {

    String username = request.getParameter("username");
    String password = request.getParameter("password");
    HttpSession session = request.getSession();
    session.setAttribute("username", username);

    boolean remember;
    if(request.getParameter("rememberMe")==null){
        remember=false;
    }
    else{
        remember=true;
    }
    try {
        User u = userDao.get(username, password);

        if (u != null && u.getStatus() == 1) {
            String userrole = u.getRole();
            session.setAttribute("userrole", userrole);
            if(remember) {
                Cookie ck=new Cookie("username",username);
                ck.setMaxAge(60*60);
                response.addCookie(ck);
            }
            return "user-dashboard";

        } else if (u != null && u.getStatus() == 0) {
            map.addAttribute("errorMessage", "Please activate your account to
login!");

            return "error";
        } else {
            map.addAttribute("errorMessage", "Invalid username/password!");
            return "error";
        }
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return null;
}

```

```

    }

    @RequestMapping(value = "/user/create.htm", method = RequestMethod.GET)
    public String showCreateForm() {

        return "user-create-form";
    }

    @RequestMapping(value = "/user/create.htm", method = RequestMethod.POST)
    public String handleCreateForm(HttpServletRequest request, UserDao userDao, ModelMap
map) {

        Captcha captcha = Captcha.load(request, "CaptchaObject");
        String captchaCode = request.getParameter("captchaCode");
        HttpSession session = request.getSession();
        if (captcha.validate(captchaCode)) {
            String useremail = request.getParameter("username");
            String password = request.getParameter("password");
            User user = new User();
            user.setUserEmail(useremail);
            user.setPassword(password);
            user.setStatus(0);
            user.setRole("student");
            try {
                User u = userDao.register(user);
                Random rand = new Random();
                int randomNum1 = rand.nextInt(5000000);
                int randomNum2 = rand.nextInt(5000000);
                try {
                    String str =
"http://localhost:8080/neubb/user/validateemail.htm?email=" + useremail + "&key1="
+ randomNum1 + "&key2=" + randomNum2;
                    session.setAttribute("key1", randomNum1);
                    session.setAttribute("key2", randomNum2);
                    System.out.println("session: " + session.getId());
                    System.out.println("session: " + session.getAttribute("key1"));
                    System.out.println("session: " + session.getAttribute("key2"));
                    sendEmail(useremail,
"Click on this link to activate your account : "+
str);

                } catch (Exception e) {
                    System.out.println("Email cannot be sent");
                }
            }
        }
    }

```

```

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    } else {
        map.addAttribute("errorMessage", "Invalid Captcha!");
        return "user-create-form";
    }

    return "user-created";
}

@RequestMapping(value = "/user/forgotpassword.htm", method = RequestMethod.GET)
public String getForgotPasswordForm(HttpServletRequest request) {

    return "forgot-password";
}

@RequestMapping(value = "/user/forgotpassword.htm", method = RequestMethod.POST)
public String handleForgotPasswordForm(HttpServletRequest request, UserDao userDao) {

    String useremail = request.getParameter("useremail");
    Captcha captcha = Captcha.load(request, "CaptchaObject");
    String captchaCode = request.getParameter("captchaCode");

    if (captcha.validate(captchaCode)) {
        User user = userDao.get(useremail);
        sendEmail(useremail, "Your password is : " + user.getPassword());
        return "forgot-password-success";
    } else {
        request.setAttribute("captchamsg", "Captcha not valid");
        return "forgot-password";
    }
}

@RequestMapping(value = "user/resendemail.htm", method = RequestMethod.POST)
public String resendEmail(HttpServletRequest request) {
    HttpSession session = request.getSession();
    String useremail = request.getParameter("username");
    Random rand = new Random();
    int randomNum1 = rand.nextInt(5000000);
    int randomNum2 = rand.nextInt(5000000);

```

```

        try {
            String str = "http://localhost:8080/neubb/user/validateemail.htm?email=" +
useremail + "&key1=" + randomNum1
                        + "&key2=" + randomNum2;
            session.setAttribute("key1", randomNum1);
            session.setAttribute("key2", randomNum2);
            sendEmail(useremail,
                    "Click on this link to activate your account : " + str);
        } catch (Exception e) {
            System.out.println("Email cannot be sent");
        }

        return "user-created";
    }

    public void sendEmail(String useremail, String message) {
        try {
            Email email = new SimpleEmail();
            email.setHostName("smtp.googlemail.com");
            email.setSmtpport(465);
            email.setAuthenticator(new
DefaultAuthenticator("contactapplication2018@gmail.com", "springmvc"));
            email.setSSLonConnect(true);
            email.setFrom("no-reply@msis.neu.edu"); // This user email does not

// exist

            email.setSubject("Password Reminder");
            email.setMsg(message); // Retrieve email from the DAO and send this
            email.addTo(useremail);
            email.send();
        } catch (EmailException e) {
            System.out.println("Email cannot be sent");
        }
    }

    @RequestMapping(value = "user/validateemail.htm", method = RequestMethod.GET)
    public String validateEmail(HttpServletRequest request, UserDao userDao, ModelMap map) {

        // The user will be sent the following link when the use registers
        // This is the format of the email
        //
        http://hostname:8080/lab10/user/validateemail.htm?email=useremail&key1=<random_number>&key2

```

=<body

```
// of the email that when user registers>
HttpSession session = request.getSession();
System.out.println("session(validateEmail): "+session.getId());
String email = request.getParameter("email");
int key1 = Integer.parseInt(request.getParameter("key1"));
int key2 = Integer.parseInt(request.getParameter("key2"));
System.out.println(session.getAttribute("key1"));
System.out.println(session.getAttribute("key2"));

if ((Integer)(session.getAttribute("key1")) == key1 &&
((Integer)session.getAttribute("key2"))== key2) {
    try {
        System.out.println("HI _____");
        boolean updateStatus = userDao.updateUser(email);
        if (updateStatus) {
            return "user-login";
        } else {

            return "error";

        }

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
} else {
    map.addAttribute("errorMessage", "Link expired , generate new link");
    map.addAttribute("resendLink", true);
    return "error";
}

return "user-login";

}

@RequestMapping(value = "/user/up", method = RequestMethod.GET)
public ModelAndView initializeForm(HttpServletRequest request) throws Exception {
    ModelAndView mv = new ModelAndView();
    mv.setViewName("user-form");
    mv.addObject("user", new User());
}
```

```

        return mv;
    }

    @RequestMapping(value="/user/up" , method = RequestMethod.POST)
    public String uploadFile1(HttpServletRequest request, @ModelAttribute("uploadedFile")
UploadedFile uploadedFile
        ,UserDAO userDao, ModelMap map) throws Exception {
        HttpSession session1 = request.getSession();
        Assignment a = new Assignment();
        // CommonsMultipartFile fileinMemory= (CommonsMultipartFile)
request.getAttribute("File");
        MultipartFile fileinMemory = uploadedFile.getFile();
        a.setFilename(fileinMemory.getOriginalFilename());
        File file= new
File("C:/Users/krish/Desktop/GOD/Northeasternuniversity/src/main/webapp/WEB-INF/downloads/" ,
fileinMemory.getOriginalFilename());
        fileinMemory.transferTo(file);
        String email = (String)session1.getAttribute("username");
        Query q = getSession().createQuery("from User where userEmail = :useremail");
        q.setString("useremail", email);
        User user = (User) q.uniqueResult();
        Set<Assignment> assList = new HashSet<Assignment>();
        Set<Assignment> existingAssList = user.getAssignment();
        assList.add(a);
        for(Assignment as:existingAssList) {
            assList.add(as);
        }
        user.setAssignment(assList);
        userDao.updateUser(email);
        session1.setAttribute("myassignments", assList);
        return "userform-success";
    }
}
}

```

COURSE CONTROLLER:

```
package com.me.neubb.controller;
```

```
import java.util.ArrayList;
```

```
import java.util.HashSet;
```



```
import java.util.List;
import java.util.Set;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
```

```
import org.hibernate.Session;
import org.hibernate.query.Query;
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.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 com.me.neubb.dao.CourseDAO;
import com.me.neubb.dao.DAO;
import com.me.neubb.dao.UserDAO;
import com.me.neubb.pojo.Course;
import com.me.neubb.pojo.User;
import com.me.neubb.validator.CourseValidator;
```

```
@Controller
@RequestMapping(value="/user/course/*")
public class CourseController {
```

```
    @Autowired
    @Qualifier("courseDao")
    CourseDAO courseDao;
```

```
    @Autowired
    @Qualifier("courseValidator")
    CourseValidator courseValidator;
```

```
    @InitBinder
```

```

private void initBinder(WebDataBinder binder) {

    binder.setValidator(courseValidator);
}

@RequestMapping(value = "/", method = RequestMethod.GET)
public ModelAndView initializeForm(HttpServletRequest request) throws Exception {
    ModelAndView mv = new ModelAndView();
    mv.setViewName("course-form");
    mv.addObject("course", new Course());
    return mv;
}

@RequestMapping(value = "/user/course/add", method = RequestMethod.POST)
public ModelAndView addMovie(HttpServletRequest request, @ModelAttribute("course") Course
course, BindingResult result) throws Exception {

    courseValidator.validate(course, result);

    if (result.hasErrors()) {
        return new ModelAndView("course-form", "course", course);
    }

    try {
        System.out.println("calling create");
        course = courseDao.create(course);
    } catch (Exception e) {
        System.out.println(e.getMessage());
        return new ModelAndView("course-error", "errorMessage", "error while creating
course");
    }
    return new ModelAndView("course-success", "course", course);
}

@RequestMapping(value="/user/course/save", method= RequestMethod.POST)
public String saveCourse(HttpServletRequest request
)throws Exception{
    HttpSession session1 = request.getSession();

```

```

        System.out.println("save function");
        UserDAO userDao = new UserDAO();
        String[] courseToAdd = request.getParameterValues("addselected");
        Set<Course> courses = new HashSet<Course>();
        for (String str : courseToAdd) {
            int id = Integer.parseInt(str);
            System.out.print(id);

            Session session = DAO.getSession();
            Query query = session.getNamedQuery("findCourseById").setInteger("id", id);
            query.setMaxResults(1);
            Course course = (Course) query.uniqueResult();
            courses.add(course);
        }

        System.out.print(courses.size());
        String email = (String)session1.getAttribute("username");
        System.out.print(email);
        Set<Course> mycourse = new HashSet<Course>();
        mycourse = userDao.updateUser(email, courses);
        session1.setAttribute("mycourses", mycourse);
        return "mycourse-view";
    }

    @RequestMapping(value = "/user/course/save", method = RequestMethod.GET)
    public String send(HttpServletRequest request) {
        return "mycourse-view";
    }

    @RequestMapping(value = "/user/course/delete", method = RequestMethod.POST)
    public String deleteCourse(HttpServletRequest request
        )throws Exception{
        HttpSession session1 = request.getSession();

        UserDAO userDao = new UserDAO();
        String[] courseToDelete = request.getParameterValues("delete");
        Set<Course> courses = new HashSet<Course>();
        Set<Course> mycourse = new HashSet<Course>();
        for (String str : courseToDelete) {
            int id = Integer.parseInt(str);

```

```

        System.out.print(id);

        Session session = DAO.getSession();
        Query query = session.getNamedQuery("findCourseById").setInteger("id",id);
        query.setMaxResults(1);
        Course course = (Course) query.uniqueResult();
        courses.add(course);
    }

    System.out.print(courses.size());
    String email = (String)session1.getAttribute("username");
    System.out.print(email);
    mycourse = userDao.updateUser1(email, courses);
    session1.setAttribute("mycourses", mycourse);
    return "mycourse-view";
}

}

```

SEARCH CONTROLLER:

```
package com.me.neubb.controller;
```

```
import java.util.List;
```

```
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.stereotype.Controller;
```

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

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

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

```
import com.me.neubb.dao.CourseDAO;
```

```
import com.me.neubb.pojo.*;
```

```
@Controller
```

```
@RequestMapping("/user/search/*")
```

```

public class SearchController {

    @Autowired
    @Qualifier("courseDao")
    CourseDAO courseDao;

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public ModelAndView initializeForm(HttpServletRequest request) throws Exception {
        ModelAndView mv = new ModelAndView();
        mv.setViewName("search-form");
        return mv;
    }

    @RequestMapping(value = "/user/search/searchcourses", method = RequestMethod.POST)
    public ModelAndView searchCourse(HttpServletRequest request) throws Exception {

        String searchType=request.getParameter("selection");
        String query=request.getParameter("query");
        HttpSession session = request.getSession();
        List<Course> courses;
        System.out.println(searchType);
        System.out.println(query);

        try {
            courses=(List<Course>) courseDao.getCoursesByQuery(searchType,query);
            if(!courses.isEmpty())
            {
                session.setAttribute("courses",courses);
                System.out.println(courses);
            }

        } catch (Exception e) {
            System.out.println(e.getMessage());
            return new ModelAndView("course-error", "errorMessage", "error while searching
course");
        }

        return new ModelAndView("success-search", "courses", courses);
    }
}

```

WORK CONTROLLER:

```
package com.me.neubb.controller;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
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.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 com.me.neubb.dao.WorkDAO;  
import com.me.neubb.pojo.Course;  
import com.me.neubb.pojo.Work;  
import com.me.neubb.validator.WorkValidator;
```

```
@Controller
```

```
@RequestMapping(value="/user/work/*")
```

```
public class WorkController {
```

```
    @Autowired
```

```
    @Qualifier("workDao")
```

```
    WorkDAO workDao;
```

```
    @Autowired
```

```
    @Qualifier("workValidator")
```

```
    WorkValidator workValidator;
```

```
    @InitBinder
```

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

```
        binder.setValidator(workValidator);
```

```
    }
```

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

```
    public ModelAndView initializeForm(HttpServletRequest request) throws Exception {
```

```
        ModelAndView mv = new ModelAndView();
```

```

        mv.setViewName("work-form");
        mv.addObject("work", new Work());
        return mv;
    }

    @RequestMapping(value = "/user/work/add", method = RequestMethod.POST)
    public ModelAndView addMovie(HttpServletRequest request, @ModelAttribute("work") Work
work, BindingResult result) throws Exception {

        workValidator.validate(work, result);

        if (result.hasErrors()) {
            return new ModelAndView("work-form", "work", work);
        }

        try {
            System.out.println("calling create");
            work = workDao.create(work);
        } catch (Exception e) {
            System.out.println(e.getMessage());
            return new ModelAndView("error", "errorMessage", "error while creating work");
        }
        return new ModelAndView("work-success", "work", work);
    }
}

```

WORKSEARCH CONTROLLER

```
package com.me.neubb.controller;
```

```
import java.util.List;
```

```
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.stereotype.Controller;
```

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

```
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
```

```
import com.me.neubb.dao.WorkDAO;
import com.me.neubb.pojo.Course;
import com.me.neubb.pojo.Work;
```

```
@Controller
```

```
@RequestMapping("/user/worksearch/*")
```

```
public class WorkSearchController {
```

```
    @Autowired
```

```
    @Qualifier("workDao")
```

```
    WorkDAO workDao;
```

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

```
    public ModelAndView initializeForm(HttpServletRequest request) throws Exception {
        ModelAndView mv = new ModelAndView();
        mv.setViewName("worksearch-form");
        return mv;
    }
```

```
    @RequestMapping(value = "/user/worksearch/searchworks", method = RequestMethod.POST)
```

```
    public ModelAndView searchCourse(HttpServletRequest request) throws Exception {
```

```
        String searchType=request.getParameter("selection");
```

```
        String query=request.getParameter("query");
```

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

```
        List<Work> works;
```

```
        System.out.println(searchType);
```

```
        System.out.println(query);
```

```
        try {
```

```
            works=(List<Work>) workDao.getWorkByQuery(searchType,query);
```

```
            if(!works.isEmpty())
```

```
            {
```

```
                session.setAttribute("works",works);
```

```
                System.out.println(works);
```

```
            }
```

```
        } catch (Exception e) {
```



```

        System.out.println(e.getMessage());
        return new ModelAndView("error", "errorMessage", "error while searching work");
    }

    return new ModelAndView("success-worksearch", "works", works);
}

}

```

FILE UPLOAD CONTROLLER:

```
package com.me.neubb.controller;
```

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
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.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;
```

```
import com.me.neubb.pojo.UploadedFile;
import com.me.neubb.validator.FileValidator;
```

```
@Controller
```

```
@RequestMapping(value="/user/upload/*")
```

```
public class UploadController {
```

```
    @Autowired
```

```
    FileValidator fileValidator;
```

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

```
        public ModelAndView initializeForm(HttpServletRequest request) throws Exception {
```

```

        ModelAndView mv = new ModelAndView();
        mv.setViewName("uploadForm");
        return mv;
    }

    @RequestMapping("/user/upload/fileUploadForm")
    public ModelAndView getUploadForm(
        @ModelAttribute("uploadedFile") UploadedFile uploadedFile,
        BindingResult result) {
        return new ModelAndView("uploadForm");
    }

    @RequestMapping("/user/upload/fileUpload")
    public ModelAndView fileUploaded(
        @ModelAttribute("uploadedFile") UploadedFile uploadedFile,
        BindingResult result) {
        InputStream inputStream = null;
        OutputStream outputStream = null;

        MultipartFile file = uploadedFile.getFile();

        fileValidator.validate(uploadedFile, result);

        String fileName = file.getOriginalFilename();

        if (result.hasErrors()) {
            return new ModelAndView("uploadForm");
        }

        try {
            inputStream = file.getInputStream();

            File newFile = new
File("C:/Users/krish/Desktop/GOD/Northeasternuniversity/src/main/webapp/WEB-INF/downloads/" +
fileName);
            if (!newFile.exists()) {
                newFile.createNewFile();
            }
            outputStream = new FileOutputStream(newFile);
            int read = 0;

```

```

        byte[] bytes = new byte[1024];

        while ((read = inputStream.read(bytes)) != -1) {
            outputStream.write(bytes, 0, read);
        }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    return new ModelAndView("showFile", "message", fileName);
}
}

```

FILE DOWNLOAD CONTROLLER:

```

package com.me.neubb.controller;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

@Controller
@RequestMapping("/user/download/*")
public class FileDownloadController {
    @Autowired
    ServletContext context;

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

```

```

        public ModelAndView initializeForm(HttpServletRequest request) throws Exception {
            ModelAndView mv = new ModelAndView();
            mv.setViewName("down-form");
            return mv;
        }

@RequestMapping("/downloads/pdf/{fileName:.+}")
public void downloader(HttpServletRequest request, HttpServletResponse
response,@PathVariable("fileName") String fileName) {

    System.out.println("Downloading file :- " + fileName);

    String downloadFolder = context.getRealPath("/WEB-INF/downloads/");
    Path file = Paths.get(downloadFolder, fileName);
    // Check if file exists
    if (Files.exists(file)) {
        // set content type
        response.setContentType("application/pdf");
        // add response header
        response.addHeader("Content-Disposition", "attachment; filename=" + fileName);
        try {
            //copies all bytes from a file to an output stream
            Files.copy(file, response.getOutputStream());
            //flushes output stream
            response.getOutputStream().flush();
        } catch (IOException e) {
            System.out.println("Error :- " + e.getMessage());
        }
    } else {
        System.out.println("Sorry File not found!!!!");
    }
}
}

```

CSV CONTROLLER:

```
package com.me.neubb.controller;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
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.stereotype.Controller;
import org.springframework.validation.BindingResult;
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 com.me.neubb.dao.JobDAO;
import com.me.neubb.pojo.Jobs;
```

```
@Controller
```

```
@RequestMapping(value="/user/job/*")
```

```
public class CSVController {
```

```
    @Autowired
```

```
    @Qualifier("jobDao")
```

```
    JobDAO jobDao;
```

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

```
        public ModelAndView initializeForm(HttpServletRequest request) throws Exception {
```

```
            ModelAndView mv = new ModelAndView();
```

```
            mv.setViewName("jobs-view");
```

```
            return mv;
```

```
        }
```

```
    @RequestMapping(value = "/user/job/upload" , method = RequestMethod.POST)
```

```
    public ModelAndView UploadData(HttpServletRequest request, @ModelAttribute("filename")
String filename, BindingResult result) throws Exception {
```

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

```
        if(!filename.equals("")){
```

```
            int page = Integer.parseInt(request.getParameter("page"));
```

```

int totalcount = jobDao.getData(filename);
session.setAttribute("totalcount", totalcount);

List<Jobs> orders = new ArrayList<Jobs>();
orders = jobDao.readData(filename, page);
System.out.println(orders.size());
session.setAttribute("orders", orders);
ModelAndView mv = new ModelAndView();
mv.getModelMap().addAttribute("orders", orders);
mv.setViewName("jobs-view");

return mv;

}

else{
    session.setAttribute("message", "Please enter a valid filename");
    return new ModelAndView("jobs-view", "message", "Please enter a valid filename");
}
}

@RequestMapping(value = "/user/job/page", method = RequestMethod.GET)
public ModelAndView UploadPaginatedData(HttpServletRequest request) throws Exception {
    HttpSession session = request.getSession();
    int page = Integer.parseInt(request.getParameter("page"));

    List<Jobs> orders = new ArrayList<Jobs>();
    orders = jobDao.readData("JobSheet", page);

    session.setAttribute("orders", orders);

    ModelAndView mv = new ModelAndView("jobs-view", "orders", orders);

    return mv;

}

@RequestMapping(value = "/user/job/add", method = RequestMethod.POST)
public ModelAndView AddData(HttpServletRequest request) throws Exception {

```

```

        HttpSession session = request.getSession();
        List<Jobs> orders = (List<Jobs>) session.getAttribute("orders");

        jobDao.addData(orders);
        session.setAttribute("rowsAffected", orders.size());
        ModelAndView mv = new ModelAndView("jobs-view");
        return mv;

    }

    @RequestMapping(value = "/user/job/save" , method = RequestMethod.POST)

    public ModelAndView getExcel(HttpServletRequest request) {
        HttpSession session = request.getSession();
        ArrayList<Jobs> orders = new ArrayList<Jobs>();
        orders = (ArrayList<Jobs>) session.getAttribute("data");
        ModelAndView mv = new ModelAndView();
        mv.getModelMap().addAttribute("orderData",orders);
        mv.setView(new XlsView());
        return mv;
    }
}

```

XLS VIEW:

```

package com.me.neubb.controller;

import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.hssf.util.HSSFColor;
import org.apache.poi.ss.usermodel.CellStyle;
import org.apache.poi.ss.usermodel.Font;
import org.springframework.web.servlet.view.document.AbstractExcelView;

```

```

import com.me.neubb.pojo.Jobs;

public class XlsView extends AbstractExcelView {

    @Override
    protected void buildExcelDocument(Map<String, Object> model,
                                      HSSFWorkbook workbook,
                                      HttpServletRequest request,
                                      HttpServletResponse response)
        throws Exception {
        // TODO Auto-generated method stub
        HttpSession session = request.getSession();
        List<Jobs> orders = (List<Jobs>) session.getAttribute("orders");
        CellStyle style = workbook.createCellStyle();
        Font font = workbook.createFont();
        font.setFontName("Arial");
        font.setColor(HSSFColor.WHITE.index);
        style.setFont(font);

        // create a new Excel sheet
        HSSFSheet sheet = workbook.createSheet("JobSheet");
        sheet.setDefaultColumnWidth(30);
        // create header row
        HSSFRow header = sheet.createRow(0);

        header.createCell(0).setCellValue("JobID");
        header.getCell(0).setCellStyle(style);

        header.createCell(1).setCellValue("JobTitle");
        header.getCell(1).setCellStyle(style);

        header.createCell(2).setCellValue("JobRole");
        header.getCell(2).setCellStyle(style);

        header.createCell(3).setCellValue("JobLocation");
        header.getCell(3).setCellStyle(style);

        header.createCell(4).setCellValue("JobVacancy");
        header.getCell(4).setCellStyle(style);
    }
}

```



```

        header.createCell(5).setCellValue("JobWebsite");
        header.getCell(5).setCellStyle(style);

        int rowNum = 1;
        for(Jobs salesorder:orders){
            HSSFRow aRow = sheet.createRow(rowNum++);
            aRow.createCell(0).setCellValue(salesorder.getJobID());
            aRow.createCell(1).setCellValue(salesorder.getJobTitle());
            aRow.createCell(2).setCellValue(salesorder.getJobRole());
            aRow.createCell(3).setCellValue(salesorder.getJobLocation());
            aRow.createCell(4).setCellValue(salesorder.getJobVacancy());
            aRow.createCell(5).setCellValue(salesorder.getJobWebsite());
        }
    }
}

```

SEND EMAIL ATTACHED CONTROLLER:

```

package com.me.neubb.controller;

import java.io.IOException;
import java.io.InputStream;

import javax.mail.internet.MimeMessage;
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamSource;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.mail.javamail.MimeMessagePreparator;
import org.springframework.stereotype.Controller;
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;

@Controller
@RequestMapping("/user/sendEmail")
public class SendEmailAttachController {
    @Autowired
    private JavaMailSender mailSender;
}

```

```

@RequestMapping(value = "/user/sendEmail", method = RequestMethod.GET)
    public String send(HttpServletRequest request) {
        return "EmailForm";
    }

```

```

@RequestMapping(value = "/user/sendEmail", method = RequestMethod.POST)
public String sendEmail(HttpServletRequest request,
    final @RequestParam CommonsMultipartFile attachFile) {

```

```

    // reads form input
    final String emailTo = request.getParameter("mailTo");
    final String subject = request.getParameter("subject");
    final String message = request.getParameter("message");

```

```

    // for logging
    System.out.println("emailTo: " + emailTo);
    System.out.println("subject: " + subject);
    System.out.println("message: " + message);
    System.out.println("attachFile: " + attachFile.getOriginalFilename());

```

```

    mailSender.send(new MimeMessagePreparator() {

```

```

        @Override
        public void prepare(MimeMessage mimeMessage) throws Exception {
            MimeMessageHelper messageHelper = new MimeMessageHelper(
                mimeMessage, true, "UTF-8");
            messageHelper.setTo(emailTo);
            messageHelper.setSubject(subject);
            messageHelper.setText(message);

```

```

            // determines if there is an upload file, attach it to the e-mail
            String attachName = attachFile.getOriginalFilename();
            if (!attachFile.equals("")) {

```

```

                messageHelper.addAttachment(attachName, new InputStreamSource() {

```

```

                    @Override
                    public InputStream getInputStream() throws IOException {
                        return attachFile.getInputStream();
                    }

```

```
        });  
    }  
  
    }  
  
    });  
  
    return "Result";  
    }  
}
```

POJO CLASSES:

USER JAVA

```
package com.me.neubb.pojo;  
  
import java.util.Set;  
  
import javax.persistence.CascadeType;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.JoinTable;  
import javax.persistence.ManyToMany;  
import javax.persistence.Table;  
import javax.persistence.JoinColumn;  
  
@Entity  
@Table(name = "user_table")  
public class User {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Column(name = "id", unique = true, nullable = false)  
    private long id;  
  
    @Column(name = "userEmail")  
    private String userEmail;
```

```

@Column(name = "password")
private String password;

@Column(name = "status")
private int status;

@Column(name = "role")
private String role;

@ManyToMany(targetEntity=Course.class,cascade=CascadeType.ALL)
@JoinTable(name="user_table_course",joinColumns=@JoinColumn(name="user_table_id_fk",r
eferencedColumnName="id"),inverseJoinColumns=@JoinColumn(name="course_id_fk",referencedColu
mnName="ID"))
private Set course;

@ManyToMany(targetEntity=Assignment.class,cascade=CascadeType.ALL)
@JoinTable(name="user_table_assignment",joinColumns=@JoinColumn(name="user_table_id_
fk1",referencedColumnName="id"),inverseJoinColumns=@JoinColumn(name="assignment_id_fk",refer
encedColumnName="Id"))
private Set assignment ;

public User() {

}

public String getRole() {
    return role;
}

public void setRole(String role) {
    this.role = role;
}

public Set getAssignment() {
    return assignment;
}

public void setAssignment(Set assignment) {
    this.assignment = assignment;
}

```

```
public long getId() {  
    return id;  
}
```

```
public void setId(long id) {  
    this.id = id;  
}
```

```
public Set getCourse() {  
    return course;  
}
```

```
public void setCourse(Set course) {  
    this.course = course;  
}
```

```
public String getUserEmail() {  
    return userEmail;  
}
```

```
public void setUserEmail(String userEmail) {  
    this.userEmail = userEmail;  
}
```

```
public String getPassword() {  
    return password;  
}
```

```
public void setPassword(String password) {  
    this.password = password;  
}
```

```

        public int getStatus() {
            return status;
        }

        public void setStatus(int status) {
            this.status = status;
        }
    }
}

```

COURSE JAVA

```

package com.me.neubb.pojo;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "Course")
@NamedQueries ({
    @NamedQuery(name = "findCourseById", query = "FROM Course where id = :id")
})
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name="ID", unique = true, nullable = false)
    private int id;

    @Column(name="TITLE")
    private String title;

    @Column(name="PROFESSOR")

```

```
private String professor;
    @Column(name="COLLEGE")
private String college;
    @Column(name="CREDITS")
private String credits;
    @Column(name="TYPE")
private String type;

    @ManyToMany(targetEntity=User.class,mappedBy="course")
    private Set user_table;

public Course(){

}

public Set getUser_table() {
    return user_table;
}

public void setUser_table(Set user_table) {
    this.user_table = user_table;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getTitle() {
    return title;
}
```

```
public void setTitle(String title) {  
    this.title = title;  
}
```

```
public String getProfessor() {  
    return professor;  
}
```

```
public void setProfessor(String professor) {  
    this.professor = professor;  
}
```

```
public String getCollege() {  
    return college;  
}
```

```
public void setCollege(String college) {  
    this.college = college;  
}
```

```
public String getCredits() {  
    return credits;  
}
```

```
public void setCredits(String credits) {  
    this.credits = credits;  
}
```

```
public String getType() {  
    return type;  
}
```

```
public void setType(String type) {  
    this.type = type;  
}
```



```
    }  
}
```

ASSIGNMENT JAVA

```
package com.me.neubb.pojo;
```

```
import java.util.Set;
```

```
import javax.persistence.Column;
```

```
import javax.persistence.Entity;
```

```
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
```

```
import javax.persistence.ManyToMany;
```

```
import javax.persistence.NamedQueries;
```

```
import javax.persistence.NamedQuery;
```

```
import javax.persistence.Table;
```

```
import javax.persistence.Transient;
```

```
import org.springframework.web.multipart.MultipartFile;
```

```
@Entity
```

```
@Table(name = "Assignment")
```

```
@NamedQueries ({
```

```
    @NamedQuery(name = "findAssignmentById", query = "FROM Assignment where id = :id")
```

```
})
```

```
public class Assignment {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

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

```
    private int id;
```

```
    @Transient
```

```
    private MultipartFile File;
```

```
    @Column(name="NAME")
```

```
    private String filename;
```

```
@ManyToMany(targetEntity=User.class,mappedBy="assignment")
private Set user_table;
```

```
public String getFilename() {
    return filename;
}
```

```
public void setFilename(String filename) {
    this.filename = filename;
}
```

```
public int getId() {
    return id;
}
```

```
public void setId(int id) {
    this.id = id;
}
```

```
public MultipartFile getFile() {
    return File;
}
```

```
public void setFile(MultipartFile file) {
    File = file;
}
```

```
public Set getUser_table() {
    return user_table;
}
```

```
public void setUser_table(Set user_table) {
    this.user_table = user_table;
}
```

```
}
```

JOBS JAVA

```
package com.me.neubb.pojo;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Jobs {

    @Id
    @Column(name="JobID", unique=true, nullable=false )
    private String jobID;

    @Column(name="JobTitle")
    private String jobTitle;

    @Column(name="JobRole")
    private String jobRole;

    @Column(name="JobLocation")
    private String jobLocation;

    @Column(name="JobVacancy")
    private String jobVacancy;

    @Column(name="JobWebsite")
    private String jobWebsite;

    public String getJobID() {
        return jobID;
    }

    public void setJobID(String jobID) {
        this.jobID = jobID;
    }

    public String getJobTitle() {
        return jobTitle;
    }
}
```

```
    public void setJobTitle(String jobTitle) {
        this.jobTitle = jobTitle;
    }

    public String getJobRole() {
        return jobRole;
    }

    public void setJobRole(String jobRole) {
        this.jobRole = jobRole;
    }

    public String getJobLocation() {
        return jobLocation;
    }

    public void setJobLocation(String jobLocation) {
        this.jobLocation = jobLocation;
    }

    public String getJobVacancy() {
        return jobVacancy;
    }

    public void setJobVacancy(String jobVacancy) {
        this.jobVacancy = jobVacancy;
    }

    public String getJobWebsite() {
        return jobWebsite;
    }

    public void setJobWebsite(String jobWebsite) {
        this.jobWebsite = jobWebsite;
    }

}
```

WORK JAVA:

```
package com.me.neubb.pojo;
```

```

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "Work")
public class Work {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name="ID1", unique = true, nullable = false)
    private int id1;

    @Column(name="COMPANY")
    private String company;
    @Column(name="EMPLOYER")
    private String employer;
    @Column(name="DURATION")
    private String duration;
    @Column(name="PAYMENT")
    private String payment;
    @Column(name="TYPE1")
    private String type1;

    public Work(){

    }

    public int getId1() {
        return id1;
    }

    public void setId1(int id1) {
        this.id1 = id1;
    }
}

```

```
public String getCompany() {  
    return company;  
}  
  
public void setCompany(String company) {  
    this.company = company;  
}  
  
public String getEmployer() {  
    return employer;  
}  
  
public void setEmployer(String employer) {  
    this.employer = employer;  
}  
  
public String getDuration() {  
    return duration;  
}  
  
public void setDuration(String duration) {  
    this.duration = duration;  
}  
  
public String getPayment() {  
    return payment;  
}  
  
public void setPayment(String payment) {  
    this.payment = payment;  
}  
  
public String getType1() {  
    return type1;  
}  
  
public void setType1(String type1) {  
    this.type1 = type1;  
}}  

```

UPLOADEDFILE JAVA:

```
package com.me.neubb.pojo;
import org.springframework.web.multipart.MultipartFile;

public class UploadedFile {

    private MultipartFile file;

    public MultipartFile getFile() {
        return file;
    }

    public void setFile(MultipartFile file) {
        this.file = file;
    }
}
```

CODE FOR CONTACT APPLICATION USING AJAX AND NODE JS:

INDEX.JS FILE

```
var http = require("http");
var fs = require("fs");
var gravatar = require('gravatar');
var cors = require('cors');
```

```
// We are considering local JSON file, for backup we can also store it on the remote server
const fileName = "contacts.json"
```

```
//Check if file exists and if not create a new file and initialize the default json structure
```

```
function checkForFile(fileName, callback) {
    fs.exists(fileName, function (exists) {
        if(exists)
        {
            callback();
        }else
        {
            fs.writeFile(fileName, '{"contacts":[]}',{flag: 'wx'}, function (err, data)
            {
                callback();
            })
        }
    });
}
```

```
// Add the new contact to the JSON file and sorts it by lastname while adding it
// We can optimize the sorting by using different data structures like tree where insertion in balanced
BST would be log n
```

```
function addContact(newContact,callback) {
    fs.readFile(fileName, 'utf8', function (err, data) {
        let listOfContacts = JSON.parse(data);
        listOfContacts.contacts.push(JSON.parse(newContact));
        listOfContacts.contacts.sort(function(a, b) {
            if(a.lastname > b.lastname) return -1;
            else if (a.lastname < b.lastname) return 1;
            return 0;
        });

        fs.writeFile(fileName, JSON.stringify(listOfContacts),'utf8');
        callback();
    });
}
```

```
// Delete the asked contact from the JSON file.
```

```
function deleteContact(id,callback) {
    fs.readFile(fileName, 'utf8', function (err, data) {
        let listOfContacts = JSON.parse(data);
        listOfContacts.contacts.splice(id,1);

        fs.writeFile(fileName, JSON.stringify(listOfContacts),'utf8');
        callback();
    });
}
```

```
// Helper function to get string format of a JSON structure
```

```
function getStringJson(text) {
    var json = {}, text = text.split("&");
    for (let i in text){
        let box = text[i].split("=");
        box[1] = box[1].replace(/%2C/g, ',').replace(/%2F/g, '/').replace(/%40/g,'@').replace(/%2B/g, '+');
        json[box[0]] = box[1];
    }
    return JSON.stringify(json);
}
```

```
//We will send them a 404 response if page doesn't exist
```



```

function send404Response(response){
    response.writeHead(404, {"Content-Type": "text/plain"});
    response.write("Error 404 - Page not found");
    response.end();
}

//Handles different kinds of request
function onRequest(request, response) {

    // Request method to load the initial homepage
    if( request.method == 'GET' && request.url == '/' ){
        response.writeHead(200, {"Content-Type": "text/html"});
        //Open file as readable stream, pipe stream to response object
        fs.createReadStream("./homepage.html").pipe(response);
    }
    // Request method to load the list of contacts on the homepage
    // The contacts are loaded from JSON file asynchronously
    else if( request.method == 'GET' && request.url == '/loadData' ){
        checkForFile(fileName, function() {
            fs.readFile(fileName, 'utf8', function (err, data) {
                response.writeHead(200, {"Content-Type": "text/html"});
                response.write(data);
                response.end();
            });
        });
    }
    // Request to add new contact to our JSON file
    else if( request.method == 'POST' && request.url == '/addContact' ){
        var jsonString = "";

        request.on('data', function (data) {
            jsonString += data;
        });

        request.on('end', function () {
            newContact = getStringJson(jsonString);
            addContact(newContact, function() {
                fs.readFile(fileName, 'utf8', function (err, data) {
                    response.writeHead(200, {"Content-Type": "text/html"});
                    response.write(data);
                    response.end();
                });
            });
        });
    }
}

```

```

        });

    });
}
// Request to extract the contact details of a specific person
else if( request.method == 'POST' && request.url == '/viewContact' ){
    var jsonString = "";
    request.on('data', function (data) {
        jsonString += data;
    });

    request.on('end', function () {
        let id = parseInt(jsonString.split("=")[1]);
        checkForFile(fileName, function() {
            fs.readFile(fileName, 'utf8', function (err, data) {
                response.writeHead(200, {"Content-Type": "text/html"});
                data = JSON.parse(data);
                data = data.contacts[id]
                var profile = gravatar.profile_url(data.email, {protocol: 'https'});
                data["profile_pic"] = profile;
                response.write(JSON.stringify(data));
                response.end();
            });
        });
    });
}
// Request to extract the contact details of a specific person
else if( request.method == 'POST' && request.url == '/deleteContact' ){
    var jsonString = "";
    request.on('data', function (data) {
        jsonString += data;
    });

    request.on('end', function () {
        let id = parseInt(jsonString.split("=")[1]);
        deleteContact(id,function(){
            response.writeHead(200, {"Content-Type": "text/html"});
            //Open file as readable stream, pipe stream to response object
            fs.createReadStream("./homepage.html").pipe(response);
        });
    });
}

```

```
// Request if page not found
else{
    send404Response(response);
}
}

// Creates server
http.createServer(onRequest).listen(3000);
console.log("Server is now running...");
```

Homepage.html file

```
<!DOCTYPE html>
<html>
    <head lang="en">
        <meta charset="UTF-8">
        <title>Phonebook WebApp</title>
        <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

        <style type="text/css">

            /* CSS for body*/
            body {
                background: #ffffff;
                margin: 0;
                padding: 0;
                border: 0;
                font-size: 100%;
                font: inherit;
                vertical-align: baseline;
            }

            .container {
                position: relative;
                margin: 0 auto;
                width: 94%;
                max-width: 100%;
            }
```

```
.content {  
    position: relative;  
    padding-top: 60px;  
}  
  
.content p {  
    margin-bottom: 10px;  
}  
  
#header {  
    z-index: 2;  
    position: fixed;  
    width: 100%;  
    height: 60px;  
    line-height: 60px;  
    background: #222;  
    color: white;  
}  
  
#header h1 {  
    top: 0px;  
    margin: 0px;  
    text-transform: uppercase;  
    font-size: 1.2em;  
}  
  
#nav {  
    position: absolute;  
    right: 0;  
    top: -15px;  
}  
  
#nav ul li {  
    float: left;  
    list-style: none;  
}  
  
#nav ul li a {  
    display: block;  
    color: white;  
    text-decoration: none;
```

```

        padding: 0 10px;
    }

    .list-group {
        display: flex;
        flex-direction: column;
        padding-left: 0;
        margin-bottom: 0;
        margin-top: auto;
    }

    .list-group-item {
        position: relative;
        display: block;
        padding: 0.75rem 1.25rem;
        margin-bottom: -1px;
        background-color: #fff;
        border: 1px solid #DFD7CA;
    }

    #contact-header {
        padding: 0.75rem 1.25rem;
        margin-bottom: 0;
        margin-top: auto;
        background-color: #F8F5F0;
        border-bottom: 1px solid #DFD7CA;
        text-align: center;
    }

    .view-contact {
        text-decoration: none;
        padding: 0px 10px;
        font-size: 1rem;
        background: transparent;
        border: 1px solid transparent;
        border-radius: 0.25rem;
        align-content: right;
    }

    /* Modal Styling For Adding Contact*/
    /* The Modal (background) */
    .modal {

```

```
        display: none;
        position: fixed;
        z-index: 1;
        left: 0;
        top: 0;
        width: 100%;
        height: 100%;
        overflow: auto;
        background-color: rgb(0,0,0);
        background-color: rgba(0,0,0,0.4);
    }
```

```
/* Modal Content/Box */
.modal-content {
    background-color: #fefefe;
    margin: 15% auto;
    padding: 20px;
    border: 1px solid #888;
    width: 80%;
}
```

```
/* The Close Button */
.close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}
```

```
.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
}
```

```
/* Form styling*/
.form-group {
    margin-bottom: 1rem;
}
```

```

label {
    display: inline-block;
    margin-bottom: .5rem;
}

.form-control {
    display: block;
    width: 95%;
    padding: 0.375rem 0.75rem;
    font-size: 0.875rem;
    line-height: 1.5;
    color: #495057;
    background-color: #fff;
    background-image: none;
    -webkit-background-clip: padding-box;
    background-clip: padding-box;
    border: 1px solid #ced4da;
    border-radius: 0.25rem;
    -webkit-transition: border-color ease-in-out 0.15s,
-webkit-box-shadow ease-in-out 0.15s;
    transition: border-color ease-in-out 0.15s, -webkit-box-shadow
ease-in-out 0.15s;
    -o-transition: border-color ease-in-out 0.15s, box-shadow ease-in-out
0.15s;
    transition: border-color ease-in-out 0.15s, box-shadow ease-in-out
0.15s;
    transition: border-color ease-in-out 0.15s, box-shadow ease-in-out
0.15s, -webkit-box-shadow ease-in-out 0.15s;
}

.center {
    text-align: center;
    padding-top: 10px;
}

img {
    display: block;
    margin: auto;
}

.left-card {
    margin-top: auto;

```

```

        width: 30%;
        height:100%;
        float:left;
    }

    .right-card {
        width: 70%;
        float:left;
    }

    .icon-bar {
        float: right;
        width: 7%;
    }

    .icon-bar button {
        float: left;
        width: 100%;
        text-align: center;
        transition: all 0.3s ease;
        color: #222;
        font-size: 18px;
    }

    .icon-bar button:hover {
        background-color: #fff;
    }

    .contact-name-p {
        float: left;
        margin: auto;
    }

```

</style>

<script src="https://code.jquery.com/jquery-1.10.2.js"></script>

</head>

<body>

<!--Header bar-->


```

<header id="header">
<div class="container">
  <h1>
    My PhoneBook App!
  </h1>

  <nav id="nav">
    <ul>
      <li>
        <a id="newContact" onclick="loadAddContactForm()">Add
New Contact</a>
      </li>
    </ul>
  </nav>
</div>
</header>

<!--Rest of the page-->
<div class="content">

  <!--Contact card-->
  <div class="left-card">
    <div width="70%">
      <img id="image" class="center" src="" alt="No profile pic! It seems you do
not have a Gravatar.">
    </div>
    <div class="center" id="contact-details">

  </div>
</div>

  <!--Contacts list ordered by descending order of last name-->
  <div class="right-card">
    <h3 id="contact-header">Contacts List</h3>
    <ul class="list-group" id="contacts-list">
      </ul>
    </div>
  </div>

  <!-- The Modal for add contact -->
  <div id="myModal" class="modal">

```

```

<!-- Modal content -->
<div class="modal-content">
  <span class="close">&times;</span>
  <form>
    <fieldset>
      <legend>Legend</legend>
      <div class="form-group">
        <label>First Name</label>
        <input type="text" class="form-control" id="fname"
placeholder="Enter first name">
      </div>
      <div class="form-group">
        <label>Last Name</label>
        <input type="text" class="form-control" id="lname"
placeholder="Enter last name">
      </div>
      <div class="form-group">
        <label>Email address</label>
        <input type="email" class="form-control" id="email"
aria-describedby="emailHelp" placeholder="Enter email">
      </div>
      <div class="form-group">
        <label>Street</label>
        <input type="text" class="form-control" id="street"
placeholder="Enter street">
      </div>
      <div class="form-group">
        <label>City</label>
        <input type="text" class="form-control" id="city" placeholder="Enter
city">
      </div>
      <div class="form-group">
        <label>State</label>
        <input type="text" class="form-control" id="state" placeholder="Enter
state">
      </div>
      <div class="form-group">
        <label>Zip Code</label>
        <input type="text" class="form-control" id="zip" placeholder="Enter
zip code">
      </div>
    </fieldset>
  </form>
</div>

```

```

        <div class="form-group">
            <label>Contact No.</label>
            <input type="text" class="form-control" id="phone" placeholder="+1
(XXX) XXX-XXXX">

        </div>
        <div class="form-group">
            <label>Date of Birth</label>
            <input type="text" class="form-control" id="dob"
placeholder="MM/DD/YYYY">
        </div>

        <button type="button" class="btn btn-primary"
onclick="addContact()">Submit</button>
    </fieldset>
</form>
</div>

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

// Load the Modal form on click of add new contact
function loadAddContactForm() {
    // Get the modal
    var modal = document.getElementById('myModal');

    // Get the button that opens the modal
    var btn = document.getElementById("newContact");

    // Get the <span> element that closes the modal
    var span = document.getElementsByClassName("close")[0];

    // When the user clicks on the button, open the modal
    btn.onclick = function() {
        modal.style.display = "block";
    }

    // When the user clicks on <span> (x), close the modal
    span.onclick = function() {
        modal.style.display = "none";
    }

    // When the user clicks anywhere outside of the modal, close it

```

```

        window.onclick = function(event) {
            if (event.target == modal) {
                modal.style.display = "none";
            }
        }
    }
}

```

// Load the contact card on click on a specific name

```
function loadContactCard(id, src, contactDetails) {
```

```
    // Set the image
```

```
    let image = document.getElementById('image');
    image.setAttribute("src", src);
```

```
    contactsList = document.getElementById('contact-details');
```

```
    // Clear the details of previously loaded contact card
```

```
    while (contactsList.firstChild) {
        contactsList.removeChild(contactsList.firstChild);
    }

```

```
    // Load the details of newly clicked contact
```

```
    contactList = document.createElement('li');
    contactList.className = "list-group-item";
    contactList.innerHTML = 'First Name: ' + contactDetails.firstname;
    contactsList.appendChild(contactList);
```

```
    contactList = document.createElement('li');
```

```
    contactList.className = "list-group-item";
    contactList.innerHTML = 'Last Name: ' + contactDetails.lastname;
    contactsList.appendChild(contactList);
```

```
    contactList = document.createElement('li');
```

```
    contactList.className = "list-group-item";
    contactList.innerHTML = 'Contact No.: ' + contactDetails.phone;
    contactsList.appendChild(contactList);
```

```
    contactList = document.createElement('li');
```

```
    contactList.className = "list-group-item";
    contactList.innerHTML = 'Email: ' + contactDetails.email;
    contactsList.appendChild(contactList);

```

```

        contactList = document.createElement('li');
        contactList.className = "list-group-item";
        contactList.innerHTML = 'Address: ' + contactDetails.street + " " +
contactDetails.city + " " + contactDetails.state + " " + contactDetails.zipcode;
        contactsList.appendChild(contactList);

        contactList = document.createElement('li');
        contactList.className = "list-group-item";
        contactList.innerHTML = 'Date Of Birth: ' + contactDetails.dob;
        contactsList.appendChild(contactList);

    }
    // Load the contact details of recently clicked contact from the JSON file
    function viewContact(id) {

        // AJAX API call to the backend to get the contact details
        $.ajax({
            type: 'POST',
            url: "http://localhost:3000/viewContact",
            data: {"id": id},
            datatype: 'json',
            success: function (data) {

                data = JSON.parse(data);
                let contactDetails = data;

                // On getting the gravtar link from contact details, load the
image from gravatar

                // Free Cors: https://crossorigin.me/https://cors.io/?
https://cors-anywhere.herokuapp.com/
                var getPic =
$.getJSON('https://cors-anywhere.herokuapp.com/' + data.profile_pic, function(data) {

                    contactsList =
document.getElementById('contact-details');

                    while (contactsList.firstChild) {

contactsList.removeChild(contactsList.firstChild);

```

gravtar email id

```
    }

    let imgSrc = data.entry[0].photos[0].value;
    loadContactCard(id,imgSrc,contactDetails);

    })
    // Clear the image if the image is not found in associated

    .fail(function( jqxhr, textStatus, error ) {
        loadContactCard(id,"",contactDetails);
        console.log(error);
    });

    },
    error: function (xhr, status, error) {
        console.log(error);
    }
    });
}

// Take the details of new contact from the form and pass it to the backend
function addContact() {

    var newContact = {
        "firstname": document.getElementById('fname').value,
        "lastname": document.getElementById('lname').value,
        "email": document.getElementById('email').value,
        "street": document.getElementById('street').value,
        "city": document.getElementById('city').value,
        "state": document.getElementById('state').value,
        "zipcode": document.getElementById('zip').value,
        "phone": document.getElementById('phone').value,
        "dob": document.getElementById('dob').value
    }

    // AJAX API call to add the new contact to the JSON
    $.ajax({
        type: 'POST',
        url: "http://localhost:3000/addContact",
        data: newContact,
        datatype: 'json',
        success: function (data) {
```

```

        console.log(data);
    },
    error: function (xhr, status, error) {
        console.log(error);
    }
});

location.reload();

}

// AJAX API call to get all the contact list when the home page is loaded
$.ajax({
    type: 'GET',
    url: "http://localhost:3000/loadData",
    datatype: 'json',
    success: function (data) {

        data = JSON.parse(data).contacts;

        var contactsList, contactList, noContactList;

        // If no contacts found
        if(data.length == 0) {
            contactsList = document.getElementById('contacts-list');
            noContactList = document.createElement('li');
            noContactList.className = "list-group-item";
            noContactList.innerHTML = "Oops! No contacts in your
phonebook";

            contactsList.appendChild(noContactList);

        }

        // Display all the contacts
        for(var i in data) {
            contactsList = document.getElementById('contacts-list');
            contactList = document.createElement('li');
            contactList.className = "list-group-item";
            contactList.setAttribute("id",i);
            //contactList.setAttribute("onclick","viewContact(" + i + ")")
            contactList.innerHTML = '<p class="contact-name-p"

```

```

onclick="viewContact(' + i + ')">' + data[i].lastname + ' ' + data[i].firstname + '</p><div
class="icon-bar"><button onclick="deleteContact(' + i + ')"><i class="fa fa-trash"></i></button><div>';
                                contactsList.appendChild(contactList);
                                }
                                },
                                error: function (xhr, status, error) {
                                    console.log(error);
                                }
                                });

// Load the contact details of recently clicked contact from the JSON file
function deleteContact(id) {

    // AJAX API call to the backend to get the contact details
    $.ajax({
        type: 'POST',
        url: "http://localhost:3000/deleteContact",
        data: {"id": id},
        datatype: 'json',
        success: function (data) {
            location.reload();
        },
        error: function (xhr, status, error) {
            console.log(error);
        }
    });
}

</script>

</body>
</html>

```

CONTACTS JSON FILE:

```

{"contacts":[{"firstname":"KRISH
JAIN","lastname":"STUDENT","email":"alpapravin24@gmail.com","street":"75 saint alphonsus
street","city":"BOSTON","state":"MA","zipcode":"02120","phone":"5123647777","dob":"08/04/1994
"},{"firstname":"YUSUF","lastname":"PROF","email":"krishpjain@gmail.com","street":"NEU","city":"
BOSTON","state":"MA","zipcode":"02215","phone":"5123647982","dob":"08/09/1970"},{"firstname"
:"NORTHEASTERN","lastname":"NUPD","email":"alpapravinjain@gmail.com","street":"NEU","city":"

```


BOSTON","state":"MA","zipcode":"02215","phone":"6176663498","dob":"11/11/2000"}}}

READ ME FILE:

Steps to run CONTACT APPLICATION :

- 1. Download Node and NPM using the following link: <https://nodejs.org/en/download/>**
- 2. Make sure if the NPM is running.**
- 3. Open the commandline in the same directory and write 'npm install'.**
- 4. Once it's done, type in command 'node index.js' and you can run the app on localhost at port 8080. Type 'localhost:3000' in the browser and you are good to go.**

Steps to run SPRING MVC HIBERNATE BLACKBOARD : Host the application in STS on localhost 8080

Regards

KRISH PRAVIN JAIN

NUID: 001881885