

SPRING HIBERNATE PROJECT ON BLACKBOARD

The project is a small replication of Northeastern Blackboard where there are different users: Student, Faculty and Admin, who can create their accounts and login with their accounts to use their functionalities. Each role is limited to their set of pages and their functionalities.

Summary of Functionality Performed:

- 1) Implemented different roles for different functionalities.
- 2) Unique Username and Email availability checking using AJAX.
- 3) Captcha implemented on account creation page to prevent bots from creating accounts.
- 4) Email verification.
- 5) File upload.
- 6) File Download.
- 7) Dynamic Excel Sheet generation.
- 8) Pagination.
- 9) Multiple Records Removal.
- 10) SQL Injection Prevention.
- 11) Validation Checks on input fields.
- 12) Filtering of Inputs.
- 13) Use of OneToOne, OneToMany, ManyToOne and ManyToMany Mappings.
- 14) Login and Logout (using Sessions)
- 15) Dynamic Content Creation and Display (Announcement, Course Material and Assignments) using CRUD operations.
- 16) Forgot Password Functionality
- 17) Extra layer of security for admin pages. (Using Declarative security)
- 18) Error Page for any invalid operation.
- 19) Storing of File separately in folder rather than storing it in database for efficiency.
- 20) Displaying error messages if user tries to access unauthorized pages from their account/profile.
- 21) Displaying error message if user tries to access pages without logging in.
- 22) Displaying error message if admins provide incorrect password or declarative web security login page.
- 23) Displaying error message if username/password didn't match in the database.

Summary of Technologies Used:

- 1) Spring MVC Design Pattern used for all the pages.
- 2) JSP for generating the html pages.
- 3) Controllers for navigation.
- 4) DAO used for interacting with the database.
- 5) Use of Inheritance mapping.
- 6) OneToOne, OneToMany, ManyToOne and ManyToMany Mappings between various POJOs.
- 7) Annotated POJOs.
- 8) Annotated Controllers.
- 9) Use of Cookies to store User's logged in status and invalidating it after logging out.
- 10) Use of Page, Session and Request Scopes.
- 11) Hibernate for interacting with Database.
- 12) Use of Criteria to limit results for Pagination.
- 13) Use of HQL queries.
- 14) Exception handling using try catch wherever required.
- 15) Use of DAO superclass for efficient transaction.
- 16) Rollback implemented in case of any Exception to protect the state of the the database.
- 17) Dynamic Excel Sheet generation from content on JSP Page.
- 18) Appropriate use of Set and ArrayList as required.
- 19) Automatic Timestamp and ID generation during database insert operation.
- 20) Implementation of Declarative Web Application Security.
- 21) Use of JSTL core library for dynamic content display.
- 22) Use of AJAX for getting username and email availability.
- 23) Use of Style sheets for Formatting and Displaying the front pages.
- 24) Proper Implementation of input Validation checks
- 25) Prevention of injected code using String filtering.

BRIEF LIST AND DESCRIPTION OF ROLE AND TASK EACH ROLE COULD PERFORM:

FACULTY :

- 1) Post Announcements for different courses.
- 2) View all the announcements.
- 3) Remove (Multiple or Single) Announcements.
- 4) Add Course Materials for different course.
- 5) View all the posted Course materials.
- 6) Download Already posted course materials.
- 7) Remove course material.
- 8) Add New Assignments along with any additional course material.
- 9) View all assignments.

STUDENT :

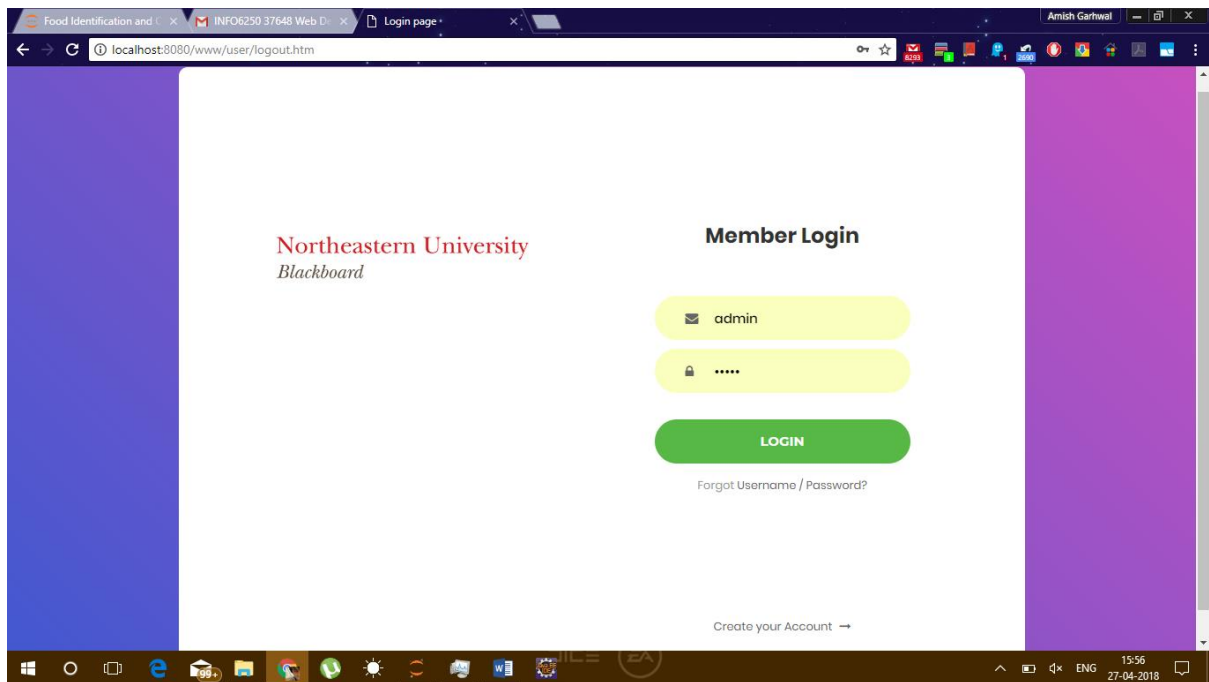
- 1) View and download course materials.
- 2) View the time at which the material was posted.
- 3) View announcements and Download them in an Excel Sheets.
- 4) View the time at which announcement was posted.
- 5) View Assignments along with the time at which it was posted.

ADMIN :

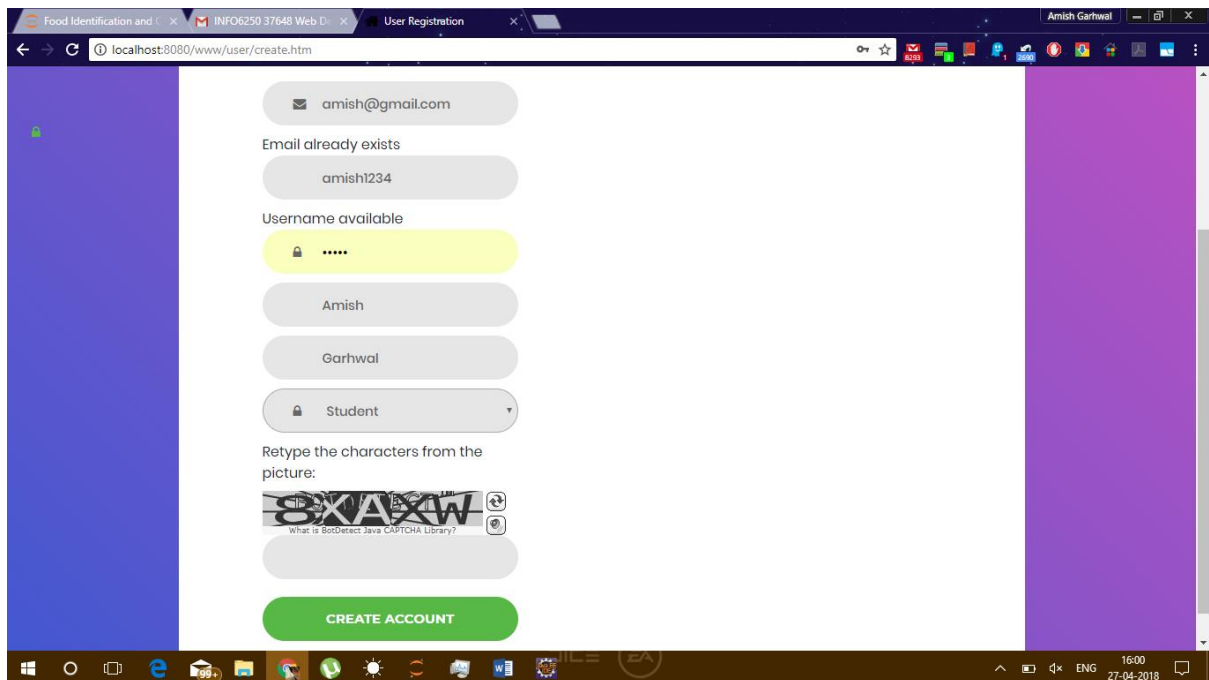
- 1) Add new courses.
- 2) Remove existing courses.
- 3) View all courses.
- 4) Admin goes through additional security check where it must provide the username/password as specified in the Tomcat properties.

SCREENSHOTS OF KEY SCREENS:

Login page:



Create Account page:



Faculty Dashboard:



Faculty Dashboard











[Go to Announcements](#)
[Go to Course Materials](#)
[Go to Assignment](#)
[Logout](#)



Faculty Announcement:



[Add Announcement](#)

	ID	Announcement	Posted By	Posted on	Course	
	21	Announcement 1	amish garhwal	2018-04-24T23:27:16	ADS	Remove
	21	a1	amish garhwal	2018-04-25T16:00:44	ADS	Remove
	37	asdf	amish garhwal	2018-04-26T11:47:31	ADS	Remove
	2	Announcement 2	amish garhwal	2018-04-24T23:27:25	Web Tools	Remove
	22	a2	amish garhwal	2018-04-25T16:00:56	Web Tools	Remove
	24	a2	amish garhwal	2018-04-25T16:01:01	Web Tools	Remove
	25	a2	amish garhwal	2018-04-25T16:01:01	Web Tools	Remove
	27	a2	amish garhwal	2018-04-25T16:01:02	Web Tools	Remove
	28	a2	amish garhwal	2018-04-25T16:01:03	Web Tools	Remove
	29	a2	amish garhwal	2018-04-25T16:01:04	Web Tools	Remove

remove announcement(s)

1 2 3 4 5 >>

[Go Back](#)



Faculty Course Material:

Food Identification and ... x INFO6250 37648 Web D... x Faculty Course Material x Amish Garhwal

localhost:8080/www/faculty/removeCourseMaterial.htm?field=12

[Add Course material](#)

ID	Name	File Description	Posted On	Course	Posted By	Download	
10	Beginning JSON.pdf	Beginning json	2018-04-26T11:49:03	AED	amish garhwal	Download	Remove File
9	Beginning JSON.pdf	new file	2018-04-26T11:48:40	ADS	amish garhwal	Download	Remove File
4	angular01.html-abcd	afdsf	2018-04-26T04:12:48	ADS	amish garhwal	Download	Remove File
11	WebFlowSlides1.pdf-abcd	new assignment	2018-04-26T11:49:36	ADS	amish garhwal	Download	Remove File
6	angular04.html-asdsf	asd	2018-04-26T04:16:04	ADS	amish garhwal	Download	Remove File
7	Closing JDBC Objects.pdf-fdsgf	sdfdgf	2018-04-26T04:22:53	AED	amish garhwal	Download	Remove File
8	Beginning JSON.pdf-Assignment 1	Hibernate	2018-04-26T11:25:10	Web Tools	amish garhwal	Download	Remove File
13	IMP SLIDES.txt	asdf	2018-04-26T12:44:14	Web Tools	amish garhwal	Download	Remove File
14	Chapter07 - Custom Tags Basics.pdf-assignment3	abcd	2018-04-26T12:45:23	Web Tools	amish garhwal	Download	Remove File

[Go Back](#)

Faculty Add Assignment:

Food Identification and ... x INFO6250 37648 Web D... x Faculty Add-Assignment x Amish Garhwal

localhost:8080/www/faculty/addAssignment.htm

Enter Title :

Enter Description :

Select a File: No file chosen

Enter Deadline: 30-04-2018

Select Course: April, 2018

[Go Back](#)

Mon	Tue	Wed	Thu	Fri	Sat	Sun
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Faculty add Course Material:

Food Identification and ... x INFO6250 37648 Web D... x Add Course-Material x Amish Garhwal

localhost:8080/www/faculty/addCourseMaterial.htm

Enter Description :

Select a File: No file chosen

Enter Filename:

Select Course: ADS

Add Course Material

[Go Back](#)

ADS

Web Tools

AED

Windows taskbar: 16:04 27-04-2018

Student Course material:

Food Identification and ... x INFO6250 37648 Web D... x Student course material x Amish Garhwal

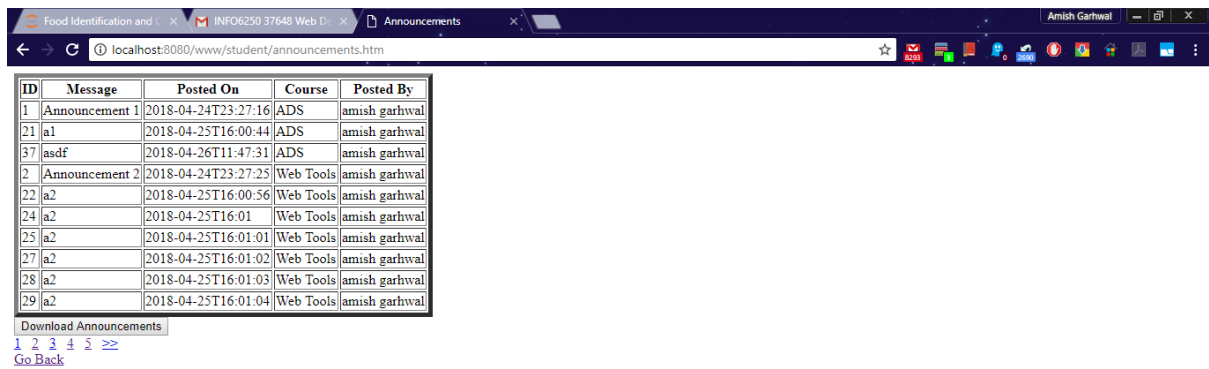
localhost:8080/www/student/course-material.htm

ID	Name	File Description	Posted On	Posted By	Download
10	Beginning JSON.pdf	Beginning json	2018-04-26T11:49:03	amish garhwal	Download
9	Beginning JSON.pdf	new file	2018-04-26T11:48:40	amish garhwal	Download
4	angular01.html-abcd	afdsf	2018-04-26T04:12:48	amish garhwal	Download
11	WebFlowSlides1.pdf-abcd	new assignment	2018-04-26T11:49:36	amish garhwal	Download
6	angular04.html-asdsf	asd	2018-04-26T04:16:04	amish garhwal	Download
7	Closing JDBC_Objects.pdf-fdsf	sdfdsf	2018-04-26T04:22:53	amish garhwal	Download
8	Beginning JSON.pdf-Assignment 1	Hibernate	2018-04-26T11:25:10	amish garhwal	Download
13	IMP SLIDES.txt	asdf	2018-04-26T12:44:14	amish garhwal	Download
14	Chapter07 - Custom Tags Basics.pdf-assignment3	abcd	2018-04-26T12:45:23	amish garhwal	Download

[Go Back](#)

Windows taskbar: 16:05 27-04-2018

Student Announcement Page:

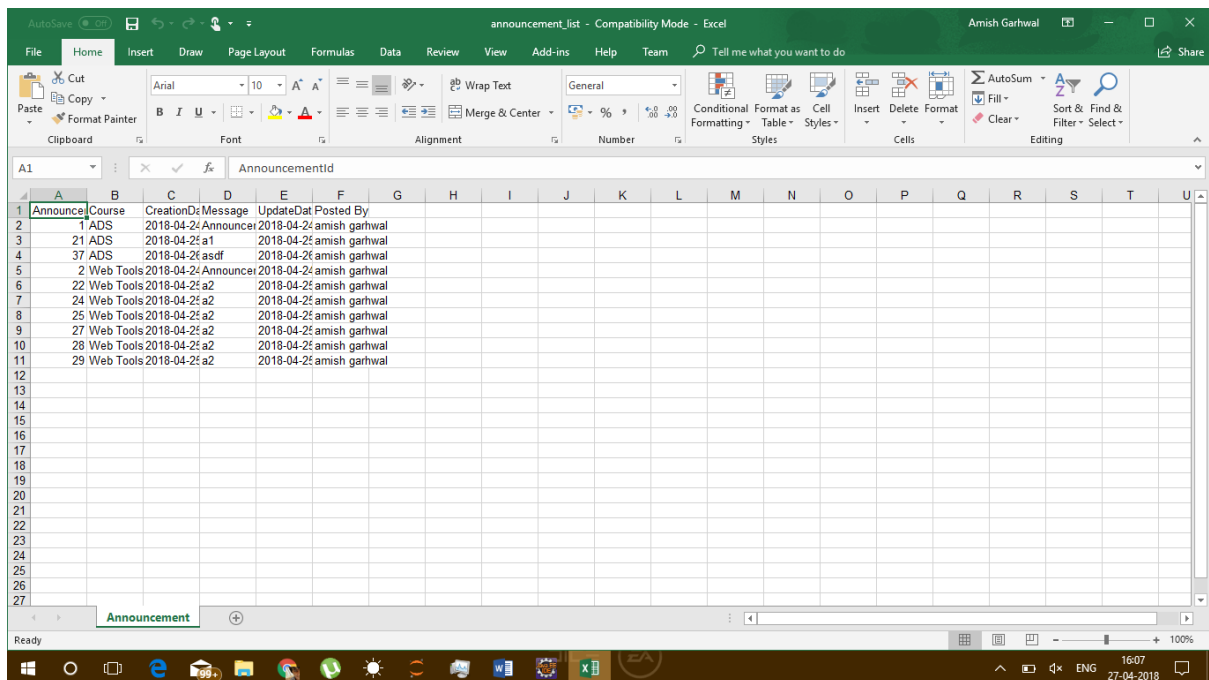


ID	Message	Posted On	Course	Posted By
1	Announcement 1	2018-04-24T23:27:16	ADS	amish garhwal
21	a1	2018-04-25T16:00:44	ADS	amish garhwal
37	asdf	2018-04-26T11:47:31	ADS	amish garhwal
2	Announcement 2	2018-04-24T23:27:25	Web Tools	amish garhwal
22	a2	2018-04-25T16:00:56	Web Tools	amish garhwal
24	a2	2018-04-25T16:01	Web Tools	amish garhwal
25	a2	2018-04-25T16:01:01	Web Tools	amish garhwal
27	a2	2018-04-25T16:01:02	Web Tools	amish garhwal
28	a2	2018-04-25T16:01:03	Web Tools	amish garhwal
29	a2	2018-04-25T16:01:04	Web Tools	amish garhwal

Download Announcements

1 2 3 4 5 >>
Go Back

Student downloaded announcement list:



AnnouncementID	Course	CreationDate	Message	UpdateDate	Posted By
1	ADS	2018-04-24	Announcement 1	2018-04-24	amish garhwal
21	ADS	2018-04-25	a1	2018-04-25	amish garhwal
37	ADS	2018-04-26	asdf	2018-04-26	amish garhwal
2	Web Tools	2018-04-24	Announcement 2	2018-04-24	amish garhwal
22	Web Tools	2018-04-25	a2	2018-04-25	amish garhwal
24	Web Tools	2018-04-25	a2	2018-04-25	amish garhwal
25	Web Tools	2018-04-25	a2	2018-04-25	amish garhwal
27	Web Tools	2018-04-25	a2	2018-04-25	amish garhwal
28	Web Tools	2018-04-25	a2	2018-04-25	amish garhwal
29	Web Tools	2018-04-25	a2	2018-04-25	amish garhwal

Admin View Courses:

The screenshot shows a web browser window with the address bar displaying `localhost:8080/www/admin/viewCourse.htm`. The browser tabs include "Food Identification and...", "INFO6250 37648 Web D...", and "Add Courses". The page content features a link "Add New Course" and a table with the following data:

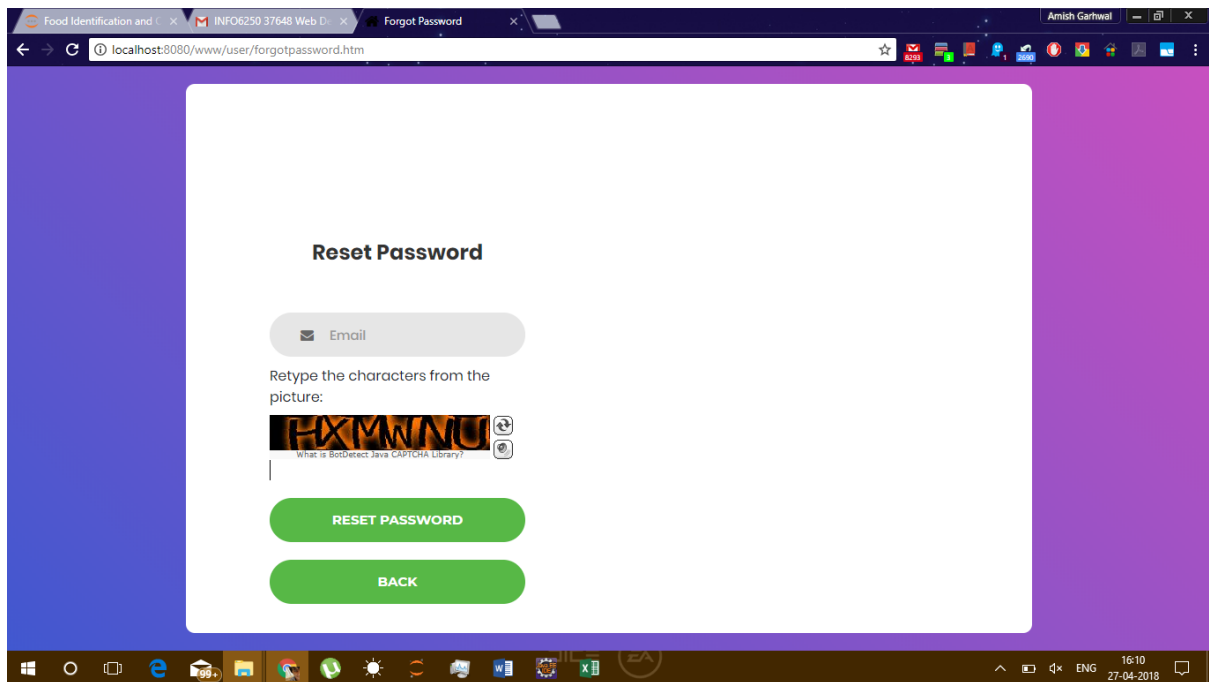
Course ID	Course Name	
1	ADS	Remove
2	Web Tools	Remove
3	AED	Remove

Below the table is a link "Go Back". The Windows taskbar at the bottom shows the time as 16:09 on 27-04-2018.

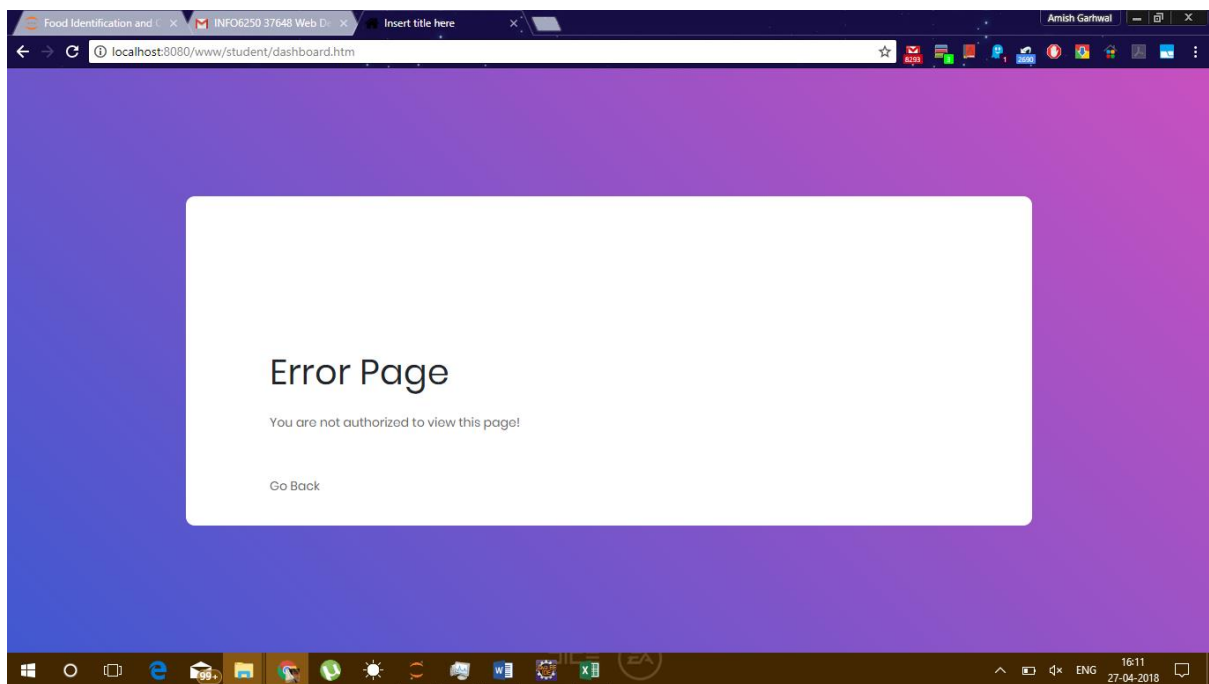
Admin Add Course:

The screenshot shows a web browser window with the address bar displaying `localhost:8080/www/admin/addCourse.htm`. The browser tabs include "Food Identification and...", "INFO6250 37648 Web D...", and "Admin Add Course". The page content features a form with the label "Enter Course Name : " followed by a text input field. Below the input field is a button labeled "Add Course" and a link "Go Back". The Windows taskbar at the bottom shows the time as 16:09 on 27-04-2018.

Forgot Password Page:



Error Page(When logged in as faculty and tried to access student dashboard):



APPENDIX:

File	PageNo.
1) Faculty Controller	11
2) Student Controller	23
3) Admin Controller	29
4) Login Controller	34
5) Ajax Controller	40
6) User POJO	41
7) Person POJO	44
8) Email POJO	46
9) Announcement POJO	48
10) Assignment POJO	50
11) Course POJO	52
12) CourseMaterial POJO	55

Faculty Controller:

```
package com.blackboard.www.controller;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
```

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

```
import org.hibernate.annotations.SourceType;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
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.bind.annotation.RequestParam;
import org.springframework.web.multipart.commons.CommonsMultipartFile;
```

```
import com.blackboard.www.dao.AnnouncementDAO;
import com.blackboard.www.dao.AssignmentDAO;
import com.blackboard.www.dao.CourseDAO;
import com.blackboard.www.dao.CourseMaterialDAO;
import com.blackboard.www.pojo.Announcement;
import com.blackboard.www.pojo.Assignment;
import com.blackboard.www.pojo.Course;
import com.blackboard.www.pojo.CourseMaterial;
import com.blackboard.www.pojo.User;
```

```
@Controller
```

```
// @RequestMapping(value = "/faculty")
```

```
public class FacultyController {
```

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

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

```
    public String showAnnouncements(HttpServletRequest request, AnnouncementDAO
annDao, ModelMap map) {
```

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

```
        User u = (User) session.getAttribute("logged-user");
```

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

```
            // session.setAttribute("user-type", "Faculty");
```

```
            String userType = (String) session.getAttribute("user-type");
```

```
            if (userType.equals("Faculty")) {
```

```
                try {
```

```
                    ArrayList<Announcement> list;
```

```
                    session.setAttribute("ROWS_TO_DISPLAY", 5);
```

```

        if (request.getParameter("page") == null) {
            list = annDao.getSome(1);
        } else {
            list =
annDao.getSome(Integer.parseInt(request.getParameter("page")));
        }
        int rowCount = annDao.getAll().size();
        session.setAttribute("rowCount",
String.valueOf(Math.abs(rowCount / 5)));
        map.addAttribute("success", list);
        return "faculty-announcement";
    } catch (Exception e) {
        System.out.println(e.getMessage());
        map.addAttribute("errorMessage", "Some error occurred!");
        map.addAttribute("backLink", "/faculty/dashboard.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "You are not authorized to view
this page!");

    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}
}

```

```

@RequestMapping(value = { "/faculty/dashboard.htm", "/faculty/" }, method =
RequestMethod.GET)
public String showDashboard(HttpServletRequest request, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Faculty")) {
            return "faculty-dashboard";
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");

            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {
        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}
}
}

```



```

        // a.setPostedBy(u.getEmail().getUser().toString());
        a.setMessage(message);
        a.setCourse(courseDao.get(course));
        a.setPostedBy(u);
        annDao.createAnnouncement(a);

        ArrayList<Announcement> list;

        session.setAttribute("ROWS_TO_DISPLAY", 5);
        if (request.getParameter("page") == null) {
            list = annDao.getSome(1);
        } else {
            list =
annDao.getSome(Integer.parseInt(request.getParameter("page")));
        }
        int rowCount = annDao.getAll().size();
        session.setAttribute("rowCount",
String.valueOf(Math.abs(rowCount / 5)));

        map.addAttribute("success", list);

        return "redirect:announcements.htm";

    } catch (Exception e) {
        System.out.println(e.getMessage());
        map.addAttribute("errorMessage", "Some error occurred!");
        map.addAttribute("backLink",
"/faculty/createAnnouncement.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "You are not authorized to view
this page!");

    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}
}

```

```

@RequestMapping(value = "/faculty/course-material.htm", method = RequestMethod.GET)
public String accessCourseMaterials(HttpServletRequest request, CourseMaterialDAO
courseMaterialDao, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");

```

```

        if (userType.equals("Faculty")) {

            try {
                ArrayList<CourseMaterial> list = courseMaterialDao.getAll();
                map.addAttribute("success", list);
                return "faculty-course-material";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink", "/faculty/dashboard.htm");
                return "error";
            }
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");

            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {
        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}

@RequestMapping(value = "/faculty/addCourseMaterial.htm", method =
RequestMethod.GET)
public String addCourseMaterials(HttpServletRequest request, CourseDAO courseDao,
ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Faculty")) {
            try {
                map.addAttribute("courseList", courseDao.getAll());
                return "faculty-add-material";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink", "/faculty/course-
material.htm");

                return "error";
            }
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");

            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {

```



```

        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}

@RequestMapping(value = "/faculty/addCourseMaterial.htm", method =
RequestMethod.POST)
public String addCourseMaterial(HttpServletRequest request, AssignmentDAO
assignmentDao,
                                CourseMaterialDAO courseMaterialDao, CourseDAO courseDao, ModelMap
map,
                                @RequestParam("file") CommonsMultipartFile file) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Faculty")) {

            System.out.println("inside controller method");
            try {
                System.out.println("inside controller method");
                String description = request.getParameter("desc");
                description = description.replaceAll("[^a-zA-Z0-9]", "");
                String fileName = file.getOriginalFilename();
                if (fileName == null || fileName.equals(""))
                    fileName = file.getOriginalFilename();
                String course = request.getParameter("course");
                User uploadedBy = u;
                String filePath = "D:\\Project_Files\\";
                try {
                    File uploadedFile = new File(filePath, fileName);
                    uploadedFile.createNewFile();
                    FileInputStream fileInputStream = new
FileInputStream(uploadedFile);

                    // convert file into array of bytes
                    fileInputStream.read(file.getBytes());
                    fileInputStream.close();
                } catch (Exception e) {
                    map.addAttribute("errorMessage", "Error uploading
file!");

                    map.addAttribute("backLink", "/faculty/course-
material.htm");

                    return "error";
                }

                CourseMaterial cm = new CourseMaterial();
                cm.setCourse(courseDao.get(course));
                cm.setFileName(fileName);
                cm.setFilePath(filePath);
                cm.setUploadedBy(uploadedBy);
            }
        }
    }
}

```

```

        cm.setFileDesc(description);
        cm.setType(file.getContentType());
        courseMaterialDao.uploadFile(cm);

        ArrayList<CourseMaterial> list = courseMaterialDao.getAll();
        map.addAttribute("success", list);

        return "faculty-course-material";
    } catch (Exception e) {
        System.out.println(e.getMessage());
        map.addAttribute("errorMessage", "Some error occurred!");
        map.addAttribute("backLink", "/faculty/course-
material.htm");

        return "error";
    }
} else {
    map.addAttribute("errorMessage", "You are not authorized to view
this page!");

    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}

@RequestMapping(value = "/faculty/addAssignment.htm", method = RequestMethod.POST)
public String addAssignment(HttpServletRequest request, AssignmentDAO assignmentDao,
CourseMaterialDAO courseMaterialDao, CourseDAO courseDao, ModelMap
map,

    @RequestParam("file") CommonsMultipartFile file) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Faculty")) {

            System.out.println("inside controller method");
            try {
                System.out.println("inside controller method");
                String title = request.getParameter("title");
                title = title.replaceAll("[^a-zA-Z0-9]", "");
                String description = request.getParameter("desc");
                description = description.replaceAll("[^a-zA-Z0-9]", "");
                String startDateStr = request.getParameter("deadline");

                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-
dd");

```

```

throws checked

// surround below line with try catch block as below code

// exception
Date startDate = sdf.parse(startDateStr);
String fileName = file.getOriginalFilename();
String course = request.getParameter("course");
User uploadedBy = u;
String filePath = "D:\\Project_Files\\";
try {
    File uploadedFile = new File(filePath, fileName);
    uploadedFile.createNewFile();
    FileInputStream fileInputStream = new
FileInputStream(uploadedFile);

    // convert file into array of bytes
    fileInputStream.read(file.getBytes());
    fileInputStream.close();
} catch (Exception e) {
    map.addAttribute("errorMessage", "Error uploading
file!");

    map.addAttribute("backLink", "/faculty/course-
material.htm");

    return "error";
}

CourseMaterial cm = new CourseMaterial();
cm.setCourse(courseDao.get(course));
cm.setFileName(fileName + "-" + title);
cm.setFilePath(filePath);
cm.setUploadedBy(uploadedBy);
cm.setFileDesc(description);
cm.setType(file.getContentType());
courseMaterialDao.uploadFile(cm);

Assignment a = new Assignment();
a.setCourse(courseDao.get(course));
a.setDeadline(startDate);
a.setDesc(description);
a.setTitle(title);
assignmentDao.createAssignment(a);
ArrayList<Assignment> list = assignmentDao.getAll();
map.addAttribute("success", list);

return "faculty-assignment-page";
} catch (Exception e) {
    System.out.println(e.getMessage());
    map.addAttribute("errorMessage", "Some error occurred!");
    map.addAttribute("backLink", "/faculty/course-
material.htm");

    return "error";
}
} else {

```

```

        map.addAttribute("errorMessage", "You are not authorized to view
this page!");

        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}

@RequestMapping(value = "/faculty/removeCourseMaterial.htm", method =
RequestMethod.GET)
public String removeCourseMaterials(HttpServletRequest request, CourseMaterialDAO
courseDao, ModelMap map) {
    System.out.println("inside controller method");
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Faculty")) {

            try {

                System.out.println("inside controller method");
                String fileId = request.getParameter("fileId");

                courseDao.delete(fileId);
                ArrayList<CourseMaterial> list = courseDao.getAll();
                map.addAttribute("success", list);

                return "faculty-course-material";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink", "/faculty/course-
material.htm");

                return "error";
            }
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");

            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {
        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}

```

```

    }

}

@RequestMapping(value = "/faculty/removeAnnouncement.htm", method =
RequestMethod.GET)
public String removeAnnouncement(HttpServletRequest request, AnnouncementDAO
annDao, ModelMap map) {
    System.out.println("inside controller method");
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Faculty")) {

            try {

                System.out.println("inside controller method");
                String Id = request.getParameter("Id");

                annDao.delete(Id);

                ArrayList<Announcement> list;

                session.setAttribute("ROWS_TO_DISPLAY", 5);
                if (request.getParameter("page") == null) {
                    list = annDao.getSome(1);
                } else {
                    list =
annDao.getSome(Integer.parseInt(request.getParameter("page")));
                }
                int rowCount = annDao.getAll().size();
                session.setAttribute("rowCount",
String.valueOf(Math.abs(rowCount / 5)));

                map.addAttribute("success", list);

                return "faculty-announcement";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink",
"/faculty/announcements.htm");
                return "error";
            }
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");

            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    }
}

```

```

        } else {
            map.addAttribute("errorMessage", "Please login first!");
            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    }

    @RequestMapping(value = "/faculty/removeAnnouncement.htm", method =
RequestMethod.POST)
    public String removeSomeAnnouncement(HttpServletRequest request, AnnouncementDAO
annDao, ModelMap map) {
        System.out.println("inside controller method");
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("logged-user");
        if (u != null) {
            String userType = (String) session.getAttribute("user-type");
            if (userType.equals("Faculty")) {

                try {
                    System.out.println("inside controller method");
                    String Id = request.getParameter("Id");

                    String select[] = request.getParameterValues("ids");

                    if (select != null) {

                        for (String s : select) {
                            annDao.delete(s);
                        }
                    }

                    ArrayList<Announcement> list;

                    session.setAttribute("ROWS_TO_DISPLAY", 5);
                    if (request.getParameter("page") == null) {
                        list = annDao.getSome(1);
                    } else {
                        list =
annDao.getSome(Integer.parseInt(request.getParameter("page")));
                    }
                    int rowCount = annDao.getAll().size();
                    session.setAttribute("rowCount",
String.valueOf(Math.abs(rowCount / 5)));

                    map.addAttribute("success", list);

                    return "faculty-announcement";
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                    map.addAttribute("errorMessage", "Some error occurred!");
                }
            }
        }
    }

```

```

        map.addAttribute("backLink",
"/faculty/announcements.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "You are not authorized to view
this page!");

    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}

@RequestMapping(value = "/faculty/assignments.htm", method = RequestMethod.GET)
public String showAssignments(HttpServletRequest request, AssignmentDAO aDao,
ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        // session.setAttribute("user-type", "Faculty");
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Faculty")) {
            try {
                ArrayList<Assignment> list;
                list = aDao.getAll();
                map.addAttribute("success", list);
                return "faculty-assignment-page";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink", "/faculty/dashboard.htm");
                return "error";
            }
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");

            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {
        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}
}

```

```

@RequestMapping(value = "/faculty/addAssignment.htm", method = RequestMethod.GET)
public String addAssignment(HttpServletRequest request, CourseDAO cDao, ModelMap
map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Faculty")) {
            try {
                map.addAttribute("courseList", cDao.getAll());
                return "faculty-add-assignment";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink", "/faculty/assignments.htm");
                return "error";
            }
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");
            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {
        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}
}

```


Student Controller:

```
package com.blackboard.www.controller;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

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

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.util.FileCopyUtils;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

import com.blackboard.www.dao.AnnouncementDAO;
import com.blackboard.www.dao.AssignmentDAO;
import com.blackboard.www.dao.CourseMaterialDAO;
import com.blackboard.www.dao.UserDAO;
import com.blackboard.www.excelview.ExcelAnnouncementView;
import com.blackboard.www.pojo.Announcement;
import com.blackboard.www.pojo.Assignment;
import com.blackboard.www.pojo.CourseMaterial;
import com.blackboard.www.pojo.User;

@Controller
public class StudentController {

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

    @RequestMapping(value = { "/student/dashboard.htm", "/" }, method =
RequestMethod.GET)
    public String showDashboard(HttpServletRequest request, ModelMap map) {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("logged-user");
        if (u != null) {
            // session.setAttribute("user-type", "Faculty");
            String userType = (String) session.getAttribute("user-type");
            if (userType.equals("Student")) {
```

```

        return "student-dashboard";
    } else {
        map.addAttribute("errorMessage", "You are not authorized to view
this page!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}

```

```

@RequestMapping(value = "/student/assignments.htm", method = RequestMethod.GET)
public String showAssignments(HttpServletRequest request, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {

        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Student")) {
            return "student-assignments";
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");
            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {
        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}
}

```

```

@RequestMapping(value = "/student/announcements.htm", method =
RequestMethod.GET)
public String showAnnouncements(HttpServletRequest request, AnnouncementDAO
annDao, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {

        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Student")) {

            try {

                ArrayList<Announcement> list;

```

```

        session.setAttribute("ROWS_TO_DISPLAY", 5);
        if (request.getParameter("page") == null) {
            list = annDao.getSome(1);
        } else {
            list =
annDao.getSome(Integer.parseInt(request.getParameter("page")));
        }
        int rowCount = annDao.getAll().size();
        session.setAttribute("rowCount",
String.valueOf(Math.abs(rowCount / 5)));

        map.addAttribute("success", list);

        return "student-announcements";
    } catch (Exception e) {
        System.out.println(e.getMessage());
        map.addAttribute("errorMessage", "Some error occurred!");
        map.addAttribute("backLink", "/faculty/dashboard.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "You are not authorized to view
this page!");

    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}
}

```

```

@RequestMapping(value = "/student/course-material.htm", method = RequestMethod.GET)
public String accessCourseMaterials(HttpServletRequest request, CourseMaterialDAO
courseDao, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {

        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Student")) {
            try {
                ArrayList<CourseMaterial> list = courseDao.getAll();
                map.addAttribute("success", list);
                return "student-course-material";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink", "/student/dashboard.htm");
                return "error";
            }
        }
    }
}

```

```

    }
    } else {
        map.addAttribute("errorMessage", "You are not authorized to view
this page!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}

```

```

@RequestMapping(value = "/student/assignment.htm", method = RequestMethod.GET)
public String accessAssignments(HttpServletRequest request, AssignmentDAO
assignmentDao,
    CourseMaterialDAO courseDao, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Student")) {
            try {
                ArrayList<Assignment> list = assignmentDao.getAll();
                map.addAttribute("success", list);
                return "student-assignment";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink", "/student/dashboard.htm");
                return "error";
            }
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");
            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {
        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}

```

```

@RequestMapping(value = "/student/course-material-{id}.htm", method =
RequestMethod.GET)
public @ResponseBody void downloadCourseMaterials(@PathVariable int id,
CourseMaterialDAO courseDao, ModelMap map,

```

```

        HttpServletResponse response) {

    try {
        System.out.println(id);
        CourseMaterial cm = courseDao.get(id);
        String fullFilePath = cm.getFilePath() + "\\ " + cm.getFileName();
        File file = getFile(fullFilePath);
        InputStream in = new FileInputStream(file);

        response.setContentType(cm.getType());
        response.setHeader("Content-Disposition", "attachment; filename=\"" +
file.getName());

        response.setHeader("Content-Length", String.valueOf(file.length()));
        FileCopyUtils.copy(in, response.getOutputStream());
        // return cm;
    } catch (Exception e) {
        System.out.println(e.getMessage());
        map.addAttribute("errorMessage", "Some error occurred!");
        map.addAttribute("backLink", "/student/dashboard.htm");
        // return null;
    }

}

}

@RequestMapping(value = "/student/downloadAnnouncements.htm", method =
RequestMethod.POST)
public ModelAndView downloadAnnouncements(HttpServletRequest request,
ExcelAnnouncementView excel, ModelMap map) {
    HttpSession session = request.getSession();
    System.out.println("inside excel controller method");
    ArrayList<Announcement> list = (ArrayList<Announcement>)
session.getAttribute("announcements");
    return new ModelAndView(new ExcelAnnouncementView(), "list", list);

}

private File getFile(String FILE_PATH) throws FileNotFoundException {
    File file = new File(FILE_PATH);
    if (!file.exists()) {
        throw new FileNotFoundException("file with path: " + FILE_PATH + " was not
found.");
    }
    return file;
}

}

```

AdminController:

```
package com.blackboard.www.controller;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

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

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.util.FileCopyUtils;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

import com.blackboard.www.dao.AnnouncementDAO;
import com.blackboard.www.dao.AssignmentDAO;
import com.blackboard.www.dao.CourseMaterialDAO;
import com.blackboard.www.dao.UserDAO;
import com.blackboard.www.excelview.ExcelAnnouncementView;
import com.blackboard.www.pojo.Announcement;
import com.blackboard.www.pojo.Assignment;
import com.blackboard.www.pojo.CourseMaterial;
import com.blackboard.www.pojo.User;

@Controller
public class StudentController {

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

    @RequestMapping(value = { "/student/dashboard.htm", "/" }, method =
RequestMethod.GET)
    public String showDashboard(HttpServletRequest request, ModelMap map) {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("logged-user");
        if (u != null) {
            // session.setAttribute("user-type", "Faculty");
            String userType = (String) session.getAttribute("user-type");
            if (userType.equals("Student")) {
```

```

        return "student-dashboard";
    } else {
        map.addAttribute("errorMessage", "You are not authorized to view
this page!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}

```

```

@RequestMapping(value = "/student/assignments.htm", method = RequestMethod.GET)
public String showAssignments(HttpServletRequest request, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {

        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Student")) {
            return "student-assignments";
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");
            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {
        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}
}

```

```

@RequestMapping(value = "/student/announcements.htm", method =
RequestMethod.GET)
public String showAnnouncements(HttpServletRequest request, AnnouncementDAO
annDao, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {

        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Student")) {

            try {

                ArrayList<Announcement> list;

```

```

        session.setAttribute("ROWS_TO_DISPLAY", 5);
        if (request.getParameter("page") == null) {
            list = annDao.getSome(1);
        } else {
            list =
annDao.getSome(Integer.parseInt(request.getParameter("page")));
        }
        int rowCount = annDao.getAll().size();
        session.setAttribute("rowCount",
String.valueOf(Math.abs(rowCount / 5)));

        map.addAttribute("success", list);

        return "student-announcements";
    } catch (Exception e) {
        System.out.println(e.getMessage());
        map.addAttribute("errorMessage", "Some error occurred!");
        map.addAttribute("backLink", "/faculty/dashboard.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "You are not authorized to view
this page!");

    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}
}

```

```

@RequestMapping(value = "/student/course-material.htm", method = RequestMethod.GET)
public String accessCourseMaterials(HttpServletRequest request, CourseMaterialDAO
courseDao, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {

        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Student")) {
            try {
                ArrayList<CourseMaterial> list = courseDao.getAll();
                map.addAttribute("success", list);
                return "student-course-material";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink", "/student/dashboard.htm");
                return "error";
            }
        }
    }
}

```



```

    }
    } else {
        map.addAttribute("errorMessage", "You are not authorized to view
this page!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
} else {
    map.addAttribute("errorMessage", "Please login first!");
    map.addAttribute("backLink", "user/login.htm");
    return "error";
}
}

```

```

@RequestMapping(value = "/student/assignment.htm", method = RequestMethod.GET)
public String accessAssignments(HttpServletRequest request, AssignmentDAO
assignmentDao,
    CourseMaterialDAO courseDao, ModelMap map) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("logged-user");
    if (u != null) {
        String userType = (String) session.getAttribute("user-type");
        if (userType.equals("Student")) {
            try {
                ArrayList<Assignment> list = assignmentDao.getAll();
                map.addAttribute("success", list);
                return "student-assignment";
            } catch (Exception e) {
                System.out.println(e.getMessage());
                map.addAttribute("errorMessage", "Some error occurred!");
                map.addAttribute("backLink", "/student/dashboard.htm");
                return "error";
            }
        } else {
            map.addAttribute("errorMessage", "You are not authorized to view
this page!");
            map.addAttribute("backLink", "user/login.htm");
            return "error";
        }
    } else {
        map.addAttribute("errorMessage", "Please login first!");
        map.addAttribute("backLink", "user/login.htm");
        return "error";
    }
}

```

```

@RequestMapping(value = "/student/course-material-{id}.htm", method =
RequestMethod.GET)
public @ResponseBody void downloadCourseMaterials(@PathVariable int id,
CourseMaterialDAO courseDao, ModelMap map,

```

```

        HttpServletResponse response) {

    try {
        System.out.println(id);
        CourseMaterial cm = courseDao.get(id);
        String fullFilePath = cm.getFilePath() + "\\ " + cm.getFileName();
        File file = getFile(fullFilePath);
        InputStream in = new FileInputStream(file);

        response.setContentType(cm.getType());
        response.setHeader("Content-Disposition", "attachment; filename=\"" +
file.getName());

        response.setHeader("Content-Length", String.valueOf(file.length()));
        FileCopyUtils.copy(in, response.getOutputStream());
        // return cm;
    } catch (Exception e) {
        System.out.println(e.getMessage());
        map.addAttribute("errorMessage", "Some error occurred!");
        map.addAttribute("backLink", "/student/dashboard.htm");
        // return null;
    }

}

}

@RequestMapping(value = "/student/downloadAnnouncements.htm", method =
RequestMethod.POST)
public ModelAndView downloadAnnouncements(HttpServletRequest request,
ExcelAnnouncementView excel, ModelMap map) {
    HttpSession session = request.getSession();
    System.out.println("inside excel controller method");
    ArrayList<Announcement> list = (ArrayList<Announcement>)
session.getAttribute("announcements");
    return new ModelAndView(new ExcelAnnouncementView(), "list", list);

}

private File getFile(String FILE_PATH) throws FileNotFoundException {
    File file = new File(FILE_PATH);
    if (!file.exists()) {
        throw new FileNotFoundException("file with path: " + FILE_PATH + " was not
found.");
    }
    return file;
}

}

```

LoginController:

```
package com.blackboard.www.controller;

import java.util.Random;

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.SimpleEmail;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.blackboard.www.dao.UserDAO;
import com.blackboard.www.pojo.User;
import com.captcha.botdetect.web.servlet.Captcha;

/**
 * Handles requests for the application home page.
 */
@Controller
public class LoginController {

    /**
     * @Autowired
     *
     * @Qualifier("userDao") UserDAO userDao;
     */
    private static final Logger logger = LoggerFactory.getLogger(LoginController.class);

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

        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");
// String role = request.getParameter("role");
try {
    User u = userDao.get(username, password);

    if (u != null && u.getStatus() == 1) {
        HttpSession session = request.getSession();
        session.setAttribute("logged-user", u);
        String[] isSelected = request.getParameterValues("rememberMe");
        if (isSelected != null) {
            Cookie c = new Cookie("username", username);
            c.setMaxAge(24 * 60 * 60);
            response.addCookie(c);
        }
        if (u.getRole().equals("Faculty")) {
            session.setAttribute("user-type", "Faculty");
            return "faculty-dashboard";
        } else if (u.getRole().equals("Student")) {
            session.setAttribute("user-type", "Student");
            return "student-dashboard";
        } else if (u.getRole().equals("Admin")) {
            session.setAttribute("user-type", "Admin");
            return "admin-panel";
        }
    } 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/logout.htm", method = RequestMethod.GET)
public String logoutuser(HttpServletRequest request, HttpServletResponse response) {
    HttpSession session = request.getSession();
    Cookie cookie = new Cookie("username", "");
    cookie.setMaxAge(0);
    response.addCookie(cookie);
    session.invalidate();
    return "user-login";
}

```

```

@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)) {
        try {
            String emailid = request.getParameter("email");
            String password = request.getParameter("password");
            String username = request.getParameter("username");
            String fname = request.getParameter("fname");
            String lname = request.getParameter("lname");
            String role = request.getParameter("role");

            User user = new User();
            user.setUsername(username);
            com.blackboard.www.pojo.Email email = new
com.blackboard.www.pojo.Email(emailid);
            user.setEmail(email);
            user.setPassword(password);
            user.setLastName(lname);
            user.setFirstName(fname);
            user.setRole(role);

            // Change later
            user.setStatus(1);
            email.setUser(user);
            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/www/user/validateemail.htm?email=" + emailid + "&key1="
+ randomNum1 + "&key2=" +
randomNum2;

                session.setAttribute("key1", randomNum1);
                session.setAttribute("key2", randomNum2);
                sendEmail(emailid, "Click on this link to activate your
account : " + str);
            } catch (Exception e) {
                System.out.println(e.getMessage());
                e.printStackTrace();
                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,
ModelMap map) {

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

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

    if (captcha.validate(captchaCode)) {
        String emailaddress = userDao.getEmail(email).getEmailAddress();
        if (emailaddress != null) {
            sendEmail(emailaddress, "Your password is : " +
userDao.getEmail(email).getUser().getPassword());
            System.out.println("Your password is : " +
userDao.getEmail(email).getUser().getPassword());
            return "forgot-password-success";
        } else {
            map.addAttribute("errorMessage", "No user found with that email
address!");
            return "error";
        }
    } 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/www/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(e.getMessage());
            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.gmail.com");
            email.setSmtpPort(587);
            email.setAuthenticator(new
DefaultAuthenticator("blackboard.test132@gmail.com", "blackboard132"));
            email.setSSLOnConnect(true);
            email.setFrom("blackboard.test132@gmail.com"); // This user email does
not
            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(e.getMessage());
            * System.out.println("Email cannot be sent"); }
            */
        } catch (Exception e) {
            System.out.println(e);
        }
    }

    @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>&k
ey2=<body
        // of the email that when user registers>

```

```

        HttpSession session = request.getSession();
        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";

    }

}

```


AjaxController:

```
package com.blackboard.www.controller;

import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest;

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.ResponseBody;

import com.blackboard.www.dao.UserDAO;
import com.blackboard.www.pojo.Email;
import com.blackboard.www.pojo.User;

@Controller
public class AjaxController {
    ArrayList<String> courseList;

    public AjaxController() {

    }

    @RequestMapping(value = "/usernameavailability.htm", method = RequestMethod.POST)
    @ResponseBody
    public String ajaxCheckusername(HttpServletRequest request, UserDAO userDao) {
        String queryString = request.getParameter("username");
        User u = userDao.getUsername(queryString);

        if (u != null) {
            return "Username already exists";
        } else
            return "Username available";
    }

    @RequestMapping(value = "/emailavailability.htm", method = RequestMethod.POST)
    @ResponseBody
    public String ajaxCheckemail(HttpServletRequest request, UserDAO userDao) {
        String queryString = request.getParameter("email");
        Email e = userDao.getEmail(queryString);

        if (e != null) {
            return "Email already exists";
        } else
            return "Email available";
    }
}
```

User POJO:

```
package com.blackboard.www.pojo;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
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.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.OneToOne;
import javax.persistence.OneToOne;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

@Entity
@Table(name = "user_table")
@PrimaryKeyJoinColumn(name = "personID")
public class User extends Person {

    @OneToOne(mappedBy = "user", cascade = CascadeType.ALL)
    private Email email;

    @Column(name = "username", unique = true, nullable = false)
    private String username;

    @Column(name = "password", nullable = false)
    private String password;

    @Column(name = "status", nullable = false)
    private int status;

    @Column(name = "role", nullable = false)
    private String role;

    @OneToMany(mappedBy = "uploadedBy", cascade = { CascadeType.PERSIST,
        CascadeType.MERGE, CascadeType.DETACH,
        CascadeType.REFRESH })
    private Set<CourseMaterial> courseMaterial = new HashSet<CourseMaterial>();

    @OneToMany(mappedBy = "postedBy", cascade = { CascadeType.PERSIST,
        CascadeType.MERGE, CascadeType.DETACH,
        CascadeType.REFRESH })
```

```

private Set<Announcement> announcements = new HashSet<Announcement>();

@OneToMany(mappedBy = "givenBy", cascade = { CascadeType.PERSIST,
CascadeType.MERGE, CascadeType.DETACH,
        CascadeType.REFRESH })
private Set<Assignment> assignment = new HashSet<Assignment>();

@ManyToMany(cascade = { CascadeType.PERSIST, CascadeType.MERGE,
CascadeType.DETACH, CascadeType.REFRESH })
@JoinTable(name = "STUDENT_COURSES", joinColumns = { @JoinColumn(name =
"personID" ) }, inverseJoinColumns = {
        @JoinColumn(name = "courseID" ) })
Set<Course> courses = new HashSet<Course>();

public User() {

}

public User(String username, String password) {
    this.username = username;
    this.password = password;
}

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 int getStatus() {
    return status;
}

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

public String getRole() {
    return role;
}

```

```

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

    public Email getEmail() {
        return email;
    }

    public void setEmail(Email email) {
        this.email = email;
    }

    public Set<CourseMaterial> getCourseMaterial() {
        return courseMaterial;
    }

    public void setCourseMaterial(Set<CourseMaterial> courseMaterial) {
        this.courseMaterial = courseMaterial;
    }

    public Set<Announcement> getAnnouncements() {
        return announcements;
    }

    public void setAnnouncements(Set<Announcement> announcements) {
        this.announcements = announcements;
    }

    public Set<Assignment> getAssignment() {
        return assignment;
    }

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

    public Set<Course> getCourses() {
        return courses;
    }

    public void setCourses(Set<Course> courses) {
        this.courses = courses;
    }

    public String toString() {
        return super.getFirstName() + " " + super.getLastName();
    }
}

```

Person POJO:

```
package com.blackboard.www.pojo;
```

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
import javax.persistence.Table;
```

```
@Entity
```

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

```
@Inheritance(strategy = InheritanceType.JOINED) // table per subclass
```

```
public class Person {
```

```
    @Id
```

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

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

```
    private long personID;
```

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

```
    private String firstName;
```

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

```
    private String lastName;
```

```
    public Person() {
```

```
    }
```

```
    public long getPersonID() {
```

```
        return personID;
```

```
    }
```

```
    public void setPersonID(long personID) {
```

```
        this.personID = personID;
```

```
    }
```

```
    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;  
    }  
  
    @Override  
    public String toString() {  
        return firstName + " " + lastName;  
    }  
  
}
```

Email POJO:

```
package com.blackboard.www.pojo;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;
import org.hibernate.annotations.Parameter;
import org.hibernate.annotations.GenericGenerator;

@Entity
@Table(name = "email_table")
public class Email {

    @Id
    @GeneratedValue(generator = "generator")
    @GenericGenerator(name = "generator", strategy = "foreign", parameters =
@Parameter(name = "property", value = "user"))
    @Column(name = "emailID", unique = true, nullable = false)
    private long id;

    @Column(name = "email_address")
    private String emailAddress;

    @OneToOne
    @PrimaryKeyJoinColumn
    private User user;

    public Email() {
    }

    public Email(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public long getId() {
        return id;
    }

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

    public String getEmailAddress() {
        return emailAddress;
    }

    public void setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
```

```
        this.user = user;
    }
}
```


Announcement POJO:

```
package com.blackboard.www.pojo;

import java.time.LocalDateTime;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
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.print.attribute.standard.DateTimeAtCompleted;

import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

@Entity
@Table(name = "Announcement_table")
public class Announcement {

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

    @ManyToOne
    @JoinColumn(name = "posted_by")
    private User postedBy;

    @Column
    @CreationTimestamp
    private LocalDateTime createDateTime;

    @Column
    @UpdateTimestamp
    private LocalDateTime updateDateTime;

    @ManyToOne
    @JoinColumn(name = "course")
    private Course course;

    @Column(name = "message", nullable = false)
    private String message;

    public User getPostedBy() {
        return postedBy;
    }
}
```

```
    public void setPostedBy(User postedBy) {
        this.postedBy = postedBy;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public Integer getId() {
        return id;
    }

    public LocalDateTime getCreateDateTime() {
        return createDateTime;
    }

    public LocalDateTime getUpdateDateTime() {
        return updateDateTime;
    }

    public Course getCourse() {
        return course;
    }

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

Assignment POJO:

```
package com.blackboard.www.pojo;
```

```
import java.time.LocalDateTime;  
import java.util.Date;  
import java.util.HashSet;  
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.JoinColumn;  
import javax.persistence.ManyToOne;  
import javax.persistence.OneToMany;  
import javax.persistence.Table;
```

```
import org.hibernate.annotations.CreationTimestamp;
```

```
@Entity
```

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

```
public class Assignment {
```

```
    @Id
```

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

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

```
    private Integer Id;
```

```
    @Column(name="title", nullable=false)
```

```
    private String title;
```

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

```
    private String desc;
```

```
    @Column
```

```
    @CreationTimestamp
```

```
    private LocalDateTime postedOn;
```

```
    @Column
```

```
    private Date Deadline;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "course")
```

```
    private Course course;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "givenBy")
```

```
    private User givenBy;
```

```

    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getDesc() {
        return desc;
    }
    public void setDesc(String desc) {
        this.desc = desc;
    }
    public Date getDeadline() {
        return Deadline;
    }
    public void setDeadline(Date deadline) {
        Deadline = deadline;
    }
    public Course getCourse() {
        return course;
    }
    public void setCourse(Course course) {
        this.course = course;
    }

    public Integer getId() {
        return Id;
    }
    public LocalDateTime getPostedOn() {
        return postedOn;
    }
    public User getGivenBy() {
        return givenBy;
    }
    public void setGivenBy(User givenBy) {
        this.givenBy = givenBy;
    }
}

```

Course POJO:

```
package com.blackboard.www.pojo;
```

```
import java.util.ArrayList;
import java.util.List;
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.OneToOne;
import javax.persistence.Table;
```

```
@Entity
```

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

```
public class Course {
```

```
    @Id
```

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

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

```
    private int courseID;
```

```
    @Column(name = "course_name", nullable = false)
```

```
    private String courseName;
```

```
    public Course() {
```

```
    }
```

```
    public Course(String courseName) {
```

```
        this.courseName = courseName;
```

```
    }
```

```
    public int getCourseID() {
```

```
        return courseID;
```

```
    }
```

```
    public void setCourseID(int courseID) {
```

```
        this.courseID = courseID;
```

```
    }
```

```
    public String getCourseName() {
```

```
        return courseName;
```

```
    }
```

```
    public void setCourseName(String courseName) {
```

```
        this.courseName = courseName;
```

```
    }
```

```

/*public Person getInstructor() {
    return instructor;
}

public void setInstructor(Person instructor) {
    this.instructor = instructor;
}

public Set<Announcement> getAnnouncements() {
    return announcements;
}

public void setAnnouncements(Set<Announcement> announcements) {
    this.announcements = announcements;
}

public Set<Person> getStudents() {
    return students;
}

public void setStudents(Set<Person> students) {
    this.students = students;
}

public String getDepartment() {
    return department;
}

public void setDepartment(String department) {
    this.department = department;
}

public List<CourseMaterial> getCourseMaterials() {
    return courseMaterials;
}

public void setCourseMaterials(List<CourseMaterial> courseMaterials) {
    this.courseMaterials = courseMaterials;
}

// Convenience method for bi-directional relationship
public void add(CourseMaterial tempCM) {
    if (courseMaterials == null)
        courseMaterials = new ArrayList();

    courseMaterials.add(tempCM);

    tempCM.setCourse(this);
}*/

```

```
@Override
public String toString() {
    return courseName;
}

}
```

CourseMaterial POJO:

```
package com.blackboard.www.pojo;
```

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

```
import java.time.LocalDateTime;
```

```
import javax.persistence.Basic;
```

```
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.Lob;
```

```
import javax.persistence.ManyToOne;
```

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

```
import org.hibernate.annotations.CreationTimestamp;
```

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

```
@Entity
```

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

```
public class CourseMaterial {
```

```
    @Id
```

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

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

```
    private Integer id;
```

```
    @Column(name = "fileName", nullable = false)
```

```
    private String fileName;
```

```
    @Column(name = "filePath", nullable = false)
```

```
    private String filePath;
```

```
    @Column(name = "type", nullable = false)
```

```
    private String type;
```

```
    @Column
```

```
    @CreationTimestamp
```

```
    private LocalDateTime uploadedOn;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "uploaded_by")
```

```
    private User uploadedBy;
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "course")
```

```
    private Course course;
```



```

// @ManyToOne(cascade = {CascadeType.PERSIST, CascadeType.MERGE,
// CascadeType.DETACH, CascadeType.REFRESH})
// @JoinColumn(name="courseID")
// private Course course;

@Column(name = "fileDesc", nullable = true)
private String fileDesc;

public String getFileName() {
    return fileName;
}

public void setFileName(String fileName) {
    this.fileName = fileName;
}

public User getUploadedBy() {
    return uploadedBy;
}

public void setUploadedBy(User uploadedBy) {
    this.uploadedBy = uploadedBy;
}

public Integer getId() {
    return id;
}

public LocalDateTime getUploadedOn() {
    return uploadedOn;
}

public String getFileDesc() {
    return fileDesc;
}

public void setFileDesc(String fileDesc) {
    this.fileDesc = fileDesc;
}

public Course getCourse() {
    return course;
}

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

```

```

public String getFilePath() {
    return filePath;
}

public void setFilePath(String filePath) {
    this.filePath = filePath;
}

public String getType() {
    return type;
}

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

@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((id == null) ? 0 : id.hashCode());
    result = prime * result + ((fileName == null) ? 0 : fileName.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (!(obj instanceof CourseMaterial))
        return false;
    CourseMaterial other = (CourseMaterial) obj;
    if (id == null) {
        if (other.id != null)
            return false;
    } else if (!id.equals(other.id))
        return false;
    if (fileName == null) {
        if (other.fileName != null)
            return false;
    } else if (!fileName.equals(other.fileName))
        return false;
    return true;
}

@Override
public String toString() {
    return "UserDocument [id=" + id + ", name=" + fileName + ", description=" + fileDesc
+ ", type=" + type + "]\n";
}

```

} }