PAMS: A WEB-BASED PROGRAMMING ASSIGNMENT MANAGEMENT SYSTEM

Peng Zhang
B.S., Nanjing Normal University, China, 2007

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

SPRING
2011

PAMS: A WEB-BASED PROGRAMMING ASSIGNMENT MANAGEMENT SYSTEM

A Project

by

Peng Zhang

Approved by:

___________________________________, Committee Chair
Du Zhang, Ph.D

___________________________________, Second Reader
Chung-E Wang, Ph.D

____
Date

Student: Peng Zhang

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

___________________________, Graduate Coordinator ____________________
Nikrouz Faroughi, Ph.D Date

Department of Computer Science

Abstract

of

PAMS: A WEB-BASED PROGRAMMING ASSIGNMENT
MANAGEMENT SYSTEM

Managing and grading programming assignments always takes a lot of effort including accepting submissions, checking timestamps of files, grading and publishing grades. PAMS is a web-based system that is designed for the purpose of automating the entire procedure of managing and grading programming assignments. With the help provided by PAMS, grading programming assignment becomes incredibly easy for instructors. The only thing that instructors need to do is setting up an assignment by providing information such as assignment name, due date, cutoff date and testing cases, and letting PAMS take care of the rest. PAMS collects the submissions from students before cutoff date, compiles and executes the programs, and then verifies for the correctness of the results. Students could check their grades after PAMS finishes grading. PAMS interacts with users by a web interface. Using PAMS requires just a browser with internet access.

My project is a perfect tool for instructors who need students to submit programming assignments. Since there is no similar tool could provide similar functions which are managing student information and grading C programming assignments, PAMS actually fills in this gap.

Designing PAMS follows the latest Java EE 6 specification which includes JavaServer Faces 2.0, Java Servlet 3.0, Enterprise JavaBeans 3.1 and Java Persistence 2.0. The JavaServer Faces provides rich web interface by Ajax components. Enterprise JavaBeans contains business logic which are database transaction control and grading logic. The Java Persistence maps the database relations to Java objects that can be used by Enterprise JavaBeans. Virtual machine technology is utilized for the grading component in PAMS. MySQL provides the backend database support for PAMS. For current version of PAMS, only C programming assignments can be graded automatically; however, PAMS does provide API for calling Java compiler and testing Java Class. Grading programming assignments written in other languages is the future work for PAMS.

______________________, Committee Chair
Du Zhang, Ph.D

____
Date

ACKNOWLEDGMENTS

I am heartily thankful to my advisor, Dr. Du Zhang, whose encouragement and guidance from the beginning to the final level enabled me to design and implement PAMS.

I also thank Dr. Chung-E Wang for his precious time, and all the people who supported me in any respect during the completion of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Many Computer Science courses require student to do programming assignments; however, grading these programming assignments takes a lot of time, furthermore, instructors have to grade these assignments repeatedly. These grading processes are monotonous and could be done by a program. Conventional way of grading a programming assignment includes several steps which are collecting assignments, compiling source code, testing result for each compiled file, checking roster and submission timestamps, giving grades and publishing grades. The entire process might take several hours for grading each assignment. I designed a management system called PAMS, Programming Assignment Management System, which not only automates grading programming assignment; but also manages student information and collects submission for instructors. PAMS provides web-based user interface, customizable grading logic and reliable information storage.

1.1. Related Works

Base on the searching result of Google, I can barely find some systems or tools which can provide same functions as PAMS does.

Currently, SacCT[1] provides the functions of managing student information, collecting submissions and publishing grades; however, it lacks grading function. Instructors have to manually download all the submissions, organize these source files before starting compiling them, and then type in the grades on SacCT after testing finished.

There is another assignment grading system called WASG [2] done by Palnitkar Rahul in 2003, however, there are still some deficiencies in WASG, it does not provide function of managing grades, it still needs instructor or grader to manually assign grades because it cannot check results of compiled file, also it requires a file server and a grading server which could cause some availability issues.

1.2. Scope of PAMS

PAMS provides simple, intuitive and user-friendly interface for students and instructor, current version of PAMS could accept all kinds of programming assignments, but it only can grade C programs right now.

PAMS allows students to

- Submit programming assignments written in C.
- View submission history.
- Check grades.

PAMS allows instructor to

- Manage student account.
- Choose manually or automatically grade assignment.
- Set up assignment information including assignment name, due date, weight and cutoff date.
- View and edit all grades.

In order to help instructors in grading C programming assignments, PAMS provides functions of managing relevant information, collecting submission, grading C programming assignments and publishing grades.

Therefore, PAMS could

- Authenticate users.

- Update information showed on web pages according to the requirement of instructor.

- Grade C programming assignments.
- Record timestamps and submission history for each user.
- Provide user-friendly interface.
- Maintain all the relevant data.
- Manage submitted files.
- Archive important data and files at the end of each semester.

Chapter 2

BACKGROUND

In general, programming assignment management system should have ability to maintain any relevant data. To grade an assignment, system must check the correctness of the source code by compiling it and checking program output, additionally, the submission date should be checked as well. This requires system keeps the timestamp of each submission and the due date of each assignment. For compilation part, usually people compile C program in Unix/Linux environment using GCC, however, there is another way of compiling C program in Windows environment using a tool called Cygwin [3] which is a Unix-like environment for Windows.

Web-based interface is very popular nowadays; it has many advantages, such as it does not require users to install any application on their machines, it is difficult to find computers that do not come with browsers. Web-based application is easier to develop for developers, so that developers only need to develop an application located on server instead of developing a server-side application and a client-side application. Besides these, web-based application is easy to update because it centralize data and code on the server.

PAMS runs on Java EE [4] platform which provides multi-tier and modularized software in Java programming language. The system contains several tiers which are web tier, business logic tier, persistence tier and data tier. Application server is GlassFish [5] and Database is MySQL.

- Web tier uses JavaServer Faces [6] technology. JavaServer Faces is a Java-based web application framework using Model-View-Controller [7] architecture.
- Business logic tier uses Enterprise JavaBean technology [8] which is a server-side model that encapsulates the business logic of an application.

- Persistence tier uses Java Persistence API [9] which is is a Java programming language framework managing relational data in applications using Java Platform.
- Data tier uses MySQL [10] database which is a relational database management system RDBMS that runs as a server providing multi-user access to a number of databases.
- Application server uses GlassFish which is an open source application server project led by Sun Microsystems for the Java EE platform.

Chapter 3

REQUIREMENT ANALYSIS

3.1. General Description

3.1.1. User Characteristics

There are two types of users in this system, including administrator and student. Administrator could set up the system, update or delete an assignment, view student's information and start grading process. Student could submit programming assignment, view submission history and check grades.

3.1.2. Functional Requirements

3.1.2.1. Administrator Functional Requirements

- Administrator has ability to add or delete a student.
- Administrator has ability to add or delete an assignment.
    - Set the due date and cutoff date for an assignment.
    - Name the assignment.
    - Set weight of assignment.
    - Define test cases and their corresponding result.
- Administrator has ability to check and edit the grading result of system.
- Administrator has ability to view all the submission details.
- Administrator has ability to start a grading process.

3.1.2.2. Student Functional Requirements

- Student can upload his/her source code to the system drop box.
  - View the list of assignment.
  - View the due date and cutoff date of the assignment.
  - Submit the source code.
- Student can check his/her submission history.
- Student can check the grade for each assignment.

3.2. Specific Requirements

3.2.1. Interface Requirement

3.2.1.1. Browser

PAMS requires JavaScript-enabled web browser including Mozilla Firefox, Google Chrome and Apple Safari.

3.2.1.2. Hardware Interface

PAMS requires a computer accessed to the ECS network domain.

3.2.1.3. Communication Interface

PAMS server communicate with the client through HTTP protocols.

3.2.2. Non-functional Requirements

- Usability – The user interface should be intuitive for both administrator and students.
- Availability – The system should be available all the time.
- Fault-tolerance – The system should tolerate some small runtime error without crushing.
- Security – The system should provide secure service for both administrator and students.
  - System archives all the submitted source code every two weeks.

- o System backup database by using database dump.
- o System prevents unauthorized access.
- o System ensures session expired after certain idle time.

3.3. System Constraints

- System only accepts submission before cutoff date for each assignment.
- For security purpose, system doesn't allow student who haven't updated his/her profile to submit assignment or check grades.
- System doesn't allow administrator to set up an assignment with an inappropriate weight that makes the sum of weight of all the assignments exceed one hundred percent.

3.4. Data Model Diagram

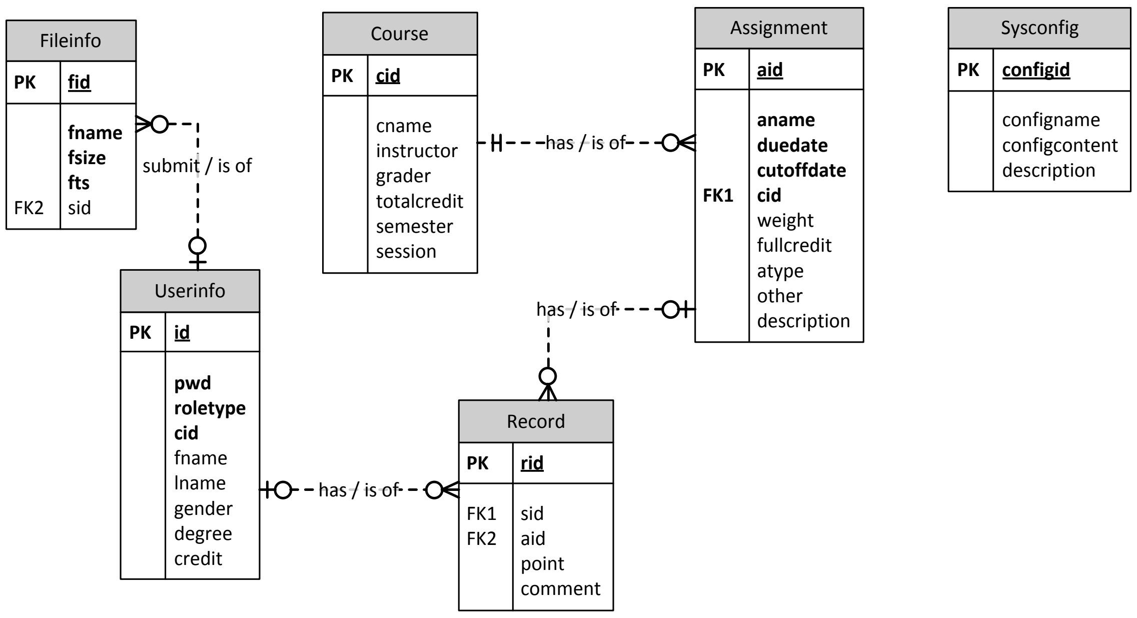


Figure 1: Entity-relationship Diagram

3.5. Use Case Model

3.5.1. Use Case Diagram

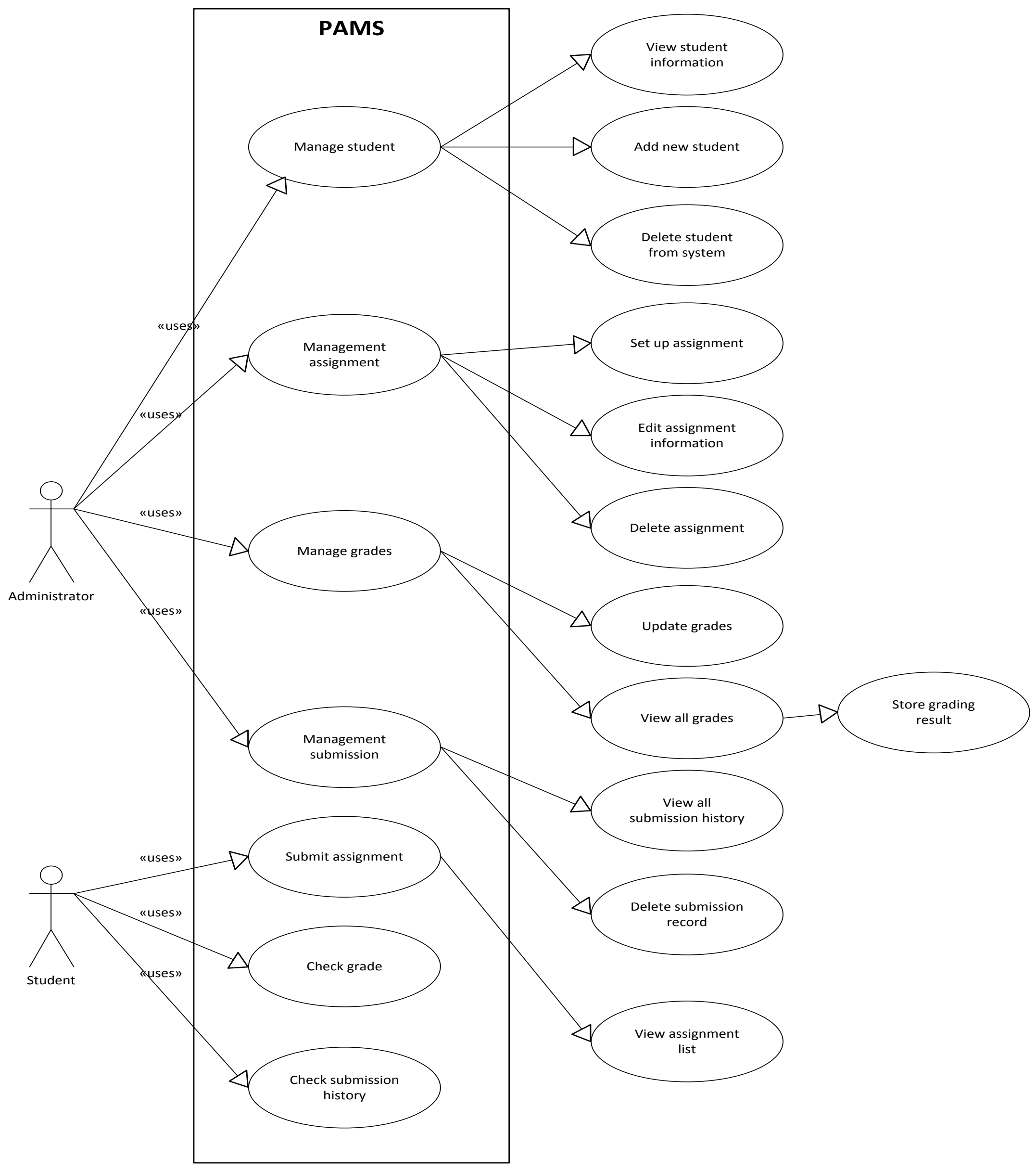


Figure 2: Use Case Diagram

3.5.2. Typical Flow Scenario of the System

3.5.2.1. Use Case: Students Submit Assignments

- Student log into the system by providing his/her user name and password.
- Student clicks on the submit button on navigation bar in panel page.
- Student is redirected to an assignment list page which lists all the available assignments.
- Student picks an appropriate assignment by clicking the assignment name on the list.
- Student is redirected to upload page in which he/she could choose a file on his/her local machine.
- Student is able to see the details of his/her submitted file after he/she clicks submit button in upload page.
- Student could choose to go back to panel page, assignment list page, check submission history page or upload another file after he/she successfully submit a file.

3.5.2.2. Use Case: Students Check Grades

- Student log into the system by providing his/her user name and password.
- Student clicks on the check grades button on the navigation bar in panel page.
- Student could see a grades table with table headers assignment name, grade and comment.

3.5.2.3. Use Case: Administrator Manages the Student Information

- Administrator log into the system by providing his/her user name and password.
- Administrator is redirected to dashboard page in which there are five tabs, submission, grades, students, assignment and course.
- Administrator clicks on the student tab.
- Administrator is able to see all the information of registered students including their user name, class name, first name, last name and their password.

- Administrator clicks add a new student button on the control panel which is located within the dashboard page if he/she wants to add a new student into the system.
- Administrator types in the user name for the new student and choose a class in the drop-down list in the pop-up panel.
- Administrator clicks add button to confirm adding a new student or clicks cancel button to cancel adding this student.
- Administrator picks a student in the student table and clicks on delete this student button on the control panel if he/she wants to delete a student from the system.
- Administrator confirms deleting selected student by clicking confirm button in the pop-up panel, or he/she cancels deleting selected student by clicking cancel button in the pop-up panel.

3.5.2.4. Use Case: Administrator Sets up System

- Administrator clicks on set up button on navigation bar in dashboard page.
- Administrator types in information of the course, and then hits the button next.
- Administrator adds assignments of this course, and then hits the button done to finish.

3.5.2.5. Use Case: System Grades Assignments

- Administrator clicks on grade assignment button on navigation bar in dashboard page.
- Administrator is redirected to grading page.
- Administrator types in test cases and their corresponding correct result.
- Administrator clicks the start grading button on the page.
- After the grading process is finished, administrator can see grades under grades tab on the dashboard page.

Chapter 4

DESIGN AND IMPLEMENTATION

4.1. System Overview

Figure 3 shows the structure of PAMS.

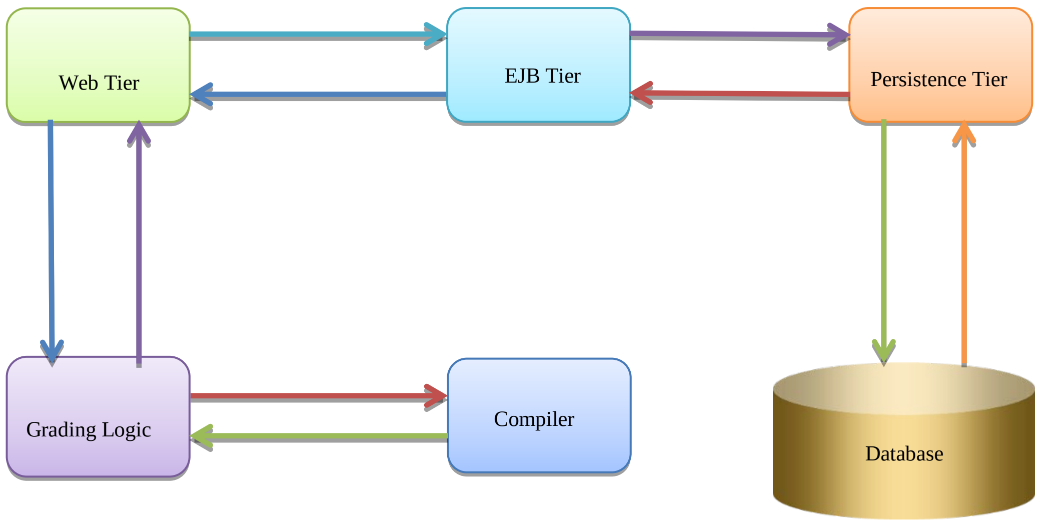


Figure 3: Structure of PAMS

PAMS has four tiers and one grading logic module:

- Data tier – Database which keeps all the required data.
- Persistence tier – Persistence entities which are mapped from database relations.
- EJB tier – EJBs which takes care of database transaction processing.
- Web tier – Web pages and managed beans which provides rich Internet applications.
- Grading logic – Classes contains the methods of checking timestamps, calling compiler and managing files.

Dividing PAMS into several components is good for:

- Reusability – All components including four tiers and a grading logic module can be reuse by other system. EJB tier, grading logic module and persistence tier have their own APIs to interact with other components.
- Maintainability – It is easy to find and solve the problem in the system if it is using multi-tier structure. We can simply remove one component and modify it, then put it back, because each component is independent, and its interface is same to its neighbor components.

Because loosely coupled PAMS is written in Java, it can be deployed on any Windows, Unix/Linux platform, the only component that need to be modified is grading logic which only provides the function calls for Cygwin environment in Windows.

4.2. Database Design and Implementation

Base on the ERD diagram in software requirement analysis chapter, this project needs six tables/entities.

- Userinfo, the relation is used to store the user information. It has nine attributes in which userId, password, roletype and classId are required, other attributes are optional.

Table 1 Userinfo table

Attribute	Type	Not Null	Key	Foreign Key
id	varchar	Yes	Yes	
pwd	varchar	Yes		
roletype	varchar	Yes		
cid	varchar	Yes		
fname	varchar			
lname	varchar			
gender	varchar			
degree	varchar			
credit	varchar			

- Fileinfo, it is used to store submission information.

Table 2 Fileinfo table

Attribute	Type	Not Null	Key	Foreign Key
fid	integer	Yes	Yes	
fname	varchar	Yes		
fsize	varchar	Yes		
fts	date	Yes		
sid	varchar	Yes		Userinfo.id

- Course, it is used to store course information.

Table 3 Course table

Attribute	Type	Not Null	Key	Foreign Key
cid	varchar	Yes	Yes	
cname	varchar	Yes		
totalcredit	varchar	Yes		
semester	varchar	Yes		
section	varchar	Yes		
grader	varchar			
instructor	varchar			

- Assignment, it is used to store assignment information.

Table 4 Assignment table

Attribute	Type	Not Null	Key	Foreign Key
aid	varchar	Yes	Yes	
aname	varchar	Yes		
duedate	date	Yes		
cutoffdate	date	Yes		
cid	varchar	Yes		Course.cid
weight	integer			
fullcredit	integer			
atype	varchar			
other	varchar			

description	varchar			

- Record, it is used to store student grades.

Table 5 Record table

Attribute	Type	Not Null	Key	Foreign Key
rid	integer	Yes	Yes	
sid	varchar	Yes		Userinfo.id
aid	varchar	yes		Assignment.aid
point	integer	Yes		
comment	varchar			

- Sysconfig, it is used to store system variables.

Table 6 Sysconfig table

Attribute	Type	Not Null	Key	Foreign Key
configid	integer	Yes	yes	
configname	varchar			
configvalue	varchar			
description	varchar			

4.3. Persistence Tier Design and Implementation

4.3.1. Description of Persistence Tier

Persistence Architecture API is a Java specification for accessing, persisting and managing data between Java objects and a relational database. The main objective of designing persistence tier is making it easier for a program to persist its state, in other word; Persistence Tier bridges the gap between object models and the relational models.

4.3.2. Mapping Relational Tables to Persistence Entities

There are two ways to specify entity metadata which is not persisted in the database: annotations or XML. They are equally valid, using annotations is more convenient for all developers, and using XML is fairly straightforward for developers who still prefer to use traditional XML. In this project, I use annotations. javax.persistence package contains all the useful API's for writing persistence entities.

For writing a persistence class, annotations @Entity and @Table are required, they designate this class as an entity class and its corresponding table name. Annotation @NamedQueries is used to preset some queries for this entity, such as select statement. Annotations @OneToMany or @ManyToOne shows the cardinality between entities.

- Persistence class for table assignment

```
@Entity
@Table(name = "assignment")
@NamedQueries({
   @NamedQuery(name = "Assignment.findAll", query = "SELECT a FROM
Assignment a"),
 })
public class Assignment implements Serializable {
   private static final long serialVersionUID = 1L;
   @Id
   @Basic(optional = false)
   @Column(name = "aid")                  // Column aid
   private String aid;                    // Mapped variable
```

```
@Column(name = "aname")          // Column aname
private String aname;
@Column(name = "duedate")        // Column duedate
@Temporal(TemporalType.DATE)
private Date duedate;
@Column(name = "cutoffdate")     // Column cutoffdate
@Temporal(TemporalType.DATE)
private Date curoffdate;
@Column(name = "weight")         // Column weight
private String weight;
@Column(name = "fullcredit")     // Column fullcredit
private String fullcredit;
@Column(name = "atype")          // Column atype
private String atype;
@Column(name = "other")          // Column other
private String other;
@Column(name = "description")    // Column description
private String description;
@OneToMany(mappedBy = "assignment")   // Cardinality
private Collection<Record> recordCollection;
@JoinColumn(name = "cid", referencedColumnName = "cid")
@ManyToOne
private Course course;
```

```
    public Assignment() {
    }
    Getters and Setters                    // For variables(Columns)
    …
}
```

- Persistence class for table course

```
@Entity
@Table(name = "course")
public class Course implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "cid")
    private String cid;                    // Map column cid
    @Column(name = "cname")
    private String cname;                  // Map column cname
    @Column(name = "instructor")
    private String instructor;             // Map column instructor
    @Column(name = "grader")
    private String grader;                 // Map column grader
    @Column(name = "totalcredit")
    private String totalcredit;            // Map column totalcredit
    @Column(name = "semester")
    private String semester;                              // Map column semester
    @Column(name = "section")
    private String section;                               // Map column section
    @OneToMany(mappedBy = "course")
    private Collection<Assignment> assignmentCollection;

    public Course() {
    }
    Getters and Setters
    …
}
```

- Persistence class for table fileinfo

```
@Entity
@Table(name = "fileinfo")
public class Fileinfo implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "fid")
    private Integer fid;                                  // Map column fid
    @Basic(optional = false)
    @Column(name = "fname")
    private String fname;                          // Map column fname
    @Basic(optional = false)
    @Column(name = "fsize")
    private String fsize;                          // Map column fsize
    @Basic(optional = false)
    @Column(name = "fts")
    @Temporal(TemporalType.DATE)
    private Date fts;                              // Map column fts
    @JoinColumn(name = "sid", referencedColumnName = "id")
    @ManyToOne
    private Userinfo userinfo;

    public Fileinfo() {
    }
    Getters and Setters
   …
}
```

- Persistence class for table record

```
@Entity
@Table(name = "record")
public class Record implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "rid")
    private Integer rid;                        // Map column rid
    @Column(name = "point")
    private String point;                       // Map column point
    @Column(name = "comment")
    private String comment;                     // Map column comment
    @JoinColumn(name = "aid", referencedColumnName = "aid")
    @ManyToOne
    private Assignment assignment;              // Object assignment mapped by aid
    @JoinColumn(name = "sid", referencedColumnName = "id")
    @ManyToOne
    private Userinfo userinfo;
    public Record() {
    }
    Getters and Setters …
}
```

- Persistence class for table sysconfig

```
@Entity
@Table(name = "sysconfig")
public class Sysconfig implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "configid")
    private Integer configid;                    // Map column configid
    @Basic(optional = false)
    @Column(name = "configname")
    private String configname;                   // Map column configname
    @Basic(optional = false)
    @Column(name = "configcontent")
    private String configcontent;                // Map column configcontent
    @Column(name = "description")
    private String description;                  // Map column description
    public Sysconfig() {
   }
   Getters and Setters
   …
}
```

- Persistence class for table userinfo

```
@Entity
@Table(name = "userinfo")
public class Userinfo implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
```

```
@Column(name = "id")
private String id;                         // Map column id
@Basic(optional = false)
@Column(name = "password")
private String password;                   // Map column password
@Basic(optional = false)
@Column(name = "roletype")
private String roletype;                   // Map column roletype
@Basic(optional = false)
@Column(name = "cid")
private String cid;                        // Map column cid
@Column(name = "fname")
private String fname;                      // Map column fname
@Column(name = "lname")
private String lname;                      // Map column lname
@Column(name = "gender")
private String gender;                     // Map column gender
@Column(name = "degree")
private String degree;
@Column(name = "credit")
private Integer credit;                    // Map column credit
@OneToMany(mappedBy = "userinfo")
private Collection<Record> recordCollection;// One student has multiple records
@OneToMany(mappedBy = "userinfo")
```

```
        private Collection<Fileinfo> fileinfoCollection;// One student submits multiple files
        public Userinfo() {
        }
        Getters and Setters…
    }
```

Here, we already finished designing persistence tier, and those persistence entities are ready for being called by upper tier as long as the appropriate JDBC driver was set up successfully.

4.4. Enterprise JavaBean Tier Design and Implementation

4.4.1. Objectives of Using EJB

The EJB specification intends to provide a standard way to implement the back-end business. In this project, EJB provides transaction processing for upper tier which is web tier.

4.4.2. Design EJB Classes

EJB container has three types of session bean; they are stateful session bean [11], stateless session bean [12] and singleton session bean [13]. In this project, only stateless session beans are used due to the fact that there is no need to keep a "state" during session.

Because there are six stateless session beans, and each of them controls transaction processing of corresponding entity, furthermore, they share some functions, such as create(), edit(), remove() and find(), so we could take advantage of inheritance in Java, writing a parent class which contains all the common methods.

4.4.3. Implement EJB

- Parent Class

```
public abstract class AbstractFacade<T> {
    private Class<T> entityClass;
    public AbstractFacade(Class<T> entityClass) {
        this.entityClass = entityClass;
    }
    protected abstract EntityManager getEntityManager();
    public void create(T entity) {
        getEntityManager().persist(entity);
    }
    public void edit(T entity) {
        getEntityManager().merge(entity);
    }
    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }
    public T find(Object id) {
        return getEntityManager().find(entityClass, id);
    }
}
```

- Assignment stateless session bean

```
@Stateless
```

```
public class AssignmentFacade extends AbstractFacade<Assignment> {
    @PersistenceContext(unitName = "PrayingMantisPU")
    private EntityManager em;
    protected EntityManager getEntityManager() {
        return em;
    }
    public AssignmentFacade() {
        super(Assignment.class);
    }
}
```

- Course stateless session bean

```
@Stateless
public class CourseFacade extends AbstractFacade<Course> {
    @PersistenceContext(unitName = "PrayingMantisPU")
    private EntityManager em;
    protected EntityManager getEntityManager() {
        return em;
    }
    public CourseFacade() {
        super(Course.class);
    }
}
```

- Fileinfo stateless session bean

```
@Stateless
public class FileinfoFacade extends AbstractFacade<Fileinfo> {
    @PersistenceContext(unitName = "PrayingMantisPU")
    private EntityManager em;
    protected EntityManager getEntityManager() {
        return em;
    }
    public FileinfoFacade() {
        super(Fileinfo.class);
    }
}
```

- Record stateless session bean

```
@Stateless
public class RecordFacade extends AbstractFacade<Record> {
    @PersistenceContext(unitName = "PrayingMantisPU")
    private EntityManager em;
    protected EntityManager getEntityManager() {
        return em;
    }
    public RecordFacade() {
        super(Record.class);
    }
}
```

- Sysconfig stateless session bean

```
@Stateless
public class SysconfigFacade extends AbstractFacade<Sysconfig> {
    @PersistenceContext(unitName = "PrayingMantisPU")
    private EntityManager em;
    protected EntityManager getEntityManager() {
        return em;
    }
    public SysconfigFacade() {
        super(Sysconfig.class);
    }
}
```

- Userinfo stateless session bean

```
@Stateless
public class UserinfoFacade extends AbstractFacade<Userinfo> {
    @PersistenceContext(unitName = "PrayingMantisPU")
    private EntityManager em;
    protected EntityManager getEntityManager() {
        return em;
    }
    public UserinfoFacade() {
        super(Userinfo.class);
      }
    }
```

4.5. Web Tier Design and Implementation

4.5.1. Description of Web Tier

Web tier provides graphical user interface, this project uses ICEfaces Framework as web tier. ICEfaces framework is an extension to the standard JavaServer Faces framework, and it provides rich Internet application. The standard JavaServer Faces framework has MVC architecture, the JSF web pages are view which presents a UI to the user, the JSF managed beans are model which contains business logic or non-UI code and FacesServlet plays role of controller which handles the user's request and dispatches the appropriate view. A JSF framework processes user's request in six phases [14]: restore view, apply request values, process validations, update model values, invoke application and render response.

- Restore view: Creates server-side component trees to represent UI information from a client.
- Apply request values: Updates the server-side components with the data retrieved from client.
- Process validations: Validates the data, such as Email format, length of input data.
- Update model values: Update server-side models.
- Invoke application: Runs logic for the request.
- Render response: Renders a response to client.

Therefore, the way of designing an ICEfaces application is:

- Put UI Components on the page. For example, <ice:inputText values="#{model.data}"/>
- Write corresponding managed bean.

  For example,

  ```
  @ManagedBean   //Annotation for managed bean
  @RequestScoped //Define scope of this bean, could be session scope or others
  public class model {
          private String data; //The field holds data from client's request
          Getter and Setter for field data
          public void logic() {
                  //put logic here
          }
  }
  ```
- Add validator for components. For example, <f:validator validatorId="dataValidator"/>
- The JSF controller which is FacesServlet will take care of the rest.

So when students fill in the input box which is rendered by inputText UI Component, the input data will be kept in String variable data in managed bean after he/she submits the form. Then the logic that defined in the method logic() in managed bean will be executed after the JSF validates the data. Ultimately, students will see the changes or redirection on the page base on the logic method and data he/she inputs.

4.5.2. Model and View Design and Implementation

The JavaServer Faces framework already provides a powerful controller, FacesServlet, for us, so we do not need to do anything about controller design and implementation. However, the models and views still need to be done by ourselves.

Login page is the first thing to take into consideration, whoever uses PAMS should be authenticated and redirected to appropriate pages. We also want to maintain a session map in memory to keep track of user's identity and activities.

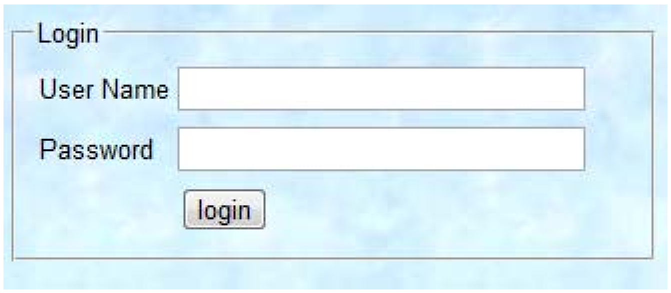


Figure 4: Login Box

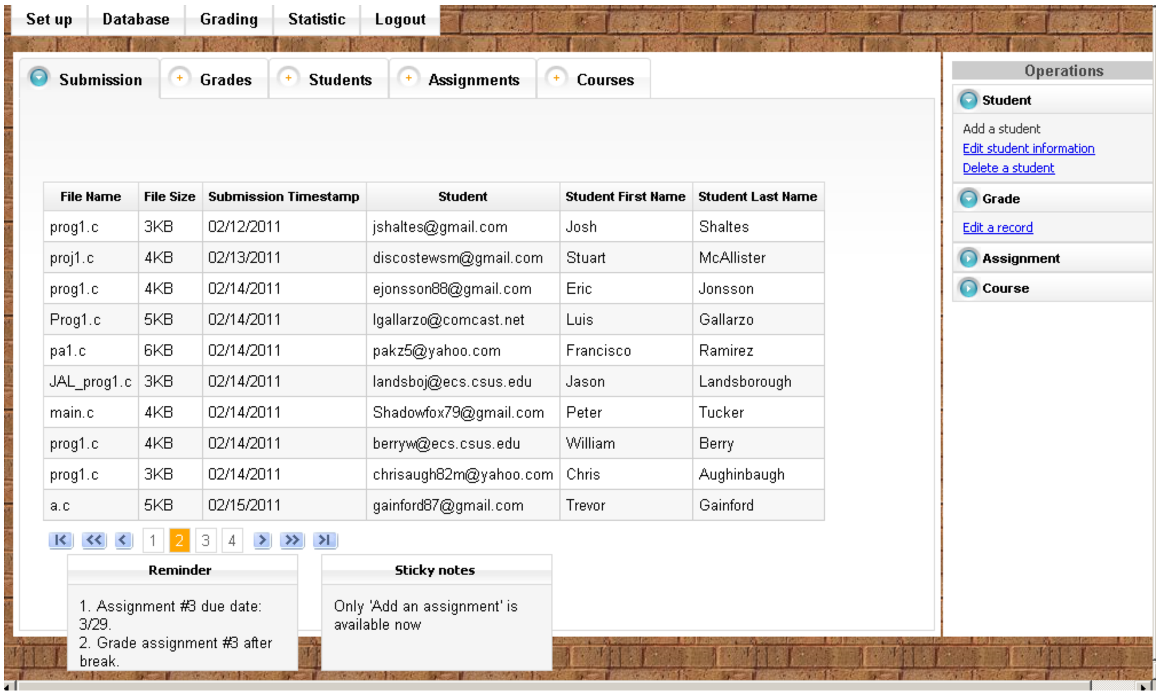
This is the login box for PAMS, user only need providing his/her user name and password to log into the system. The Userinfo table has an attribute called roletype which ensures the role of user, so he/she could be redirected to appropriate pages. For creating a session map, we could write a method to return a session map object.

```
public Map<String, Object> getSessionMap() {

    FacesContext context = FacesContext.getCurrentInstance();

    sessionMap = context.getExternalContext().getSessionMap();
```

```
        return sessionMap;

}
```

When user hits the login button, managed bean puts his/her user name and a mapped object of userinfo entity to the session map.

Administrator and student should have different main pages since they have different requirements of using PAMS.



File Name	File Size	Submission Timestamp	Student	Student First Name	Student Last Name
prog1.c	3KB	02/12/2011	jshaltes@gmail.com	Josh	Shaltes
proj1.c	4KB	02/13/2011	discostewsm@gmail.com	Stuart	McAllister
prog1.c	4KB	02/14/2011	ejonsson88@gmail.com	Eric	Jonsson
Prog1.c	5KB	02/14/2011	lgallarzo@comcast.net	Luis	Gallarzo
pa1.c	6KB	02/14/2011	pakz5@yahoo.com	Francisco	Ramirez
JAL_prog1.c	3KB	02/14/2011	landsboj@ecs.csus.edu	Jason	Landsborough
main.c	4KB	02/14/2011	Shadowfox79@gmail.com	Peter	Tucker
prog1.c	4KB	02/14/2011	berryw@ecs.csus.edu	William	Berry
prog1.c	3KB	02/14/2011	chrisaugh82m@yahoo.com	Chris	Aughinbaugh
a.c	5KB	02/15/2011	gainford87@gmail.com	Trevor	Gainford

1. Assignment #3 due date: 3/29.
2. Grade assignment #3 after break.

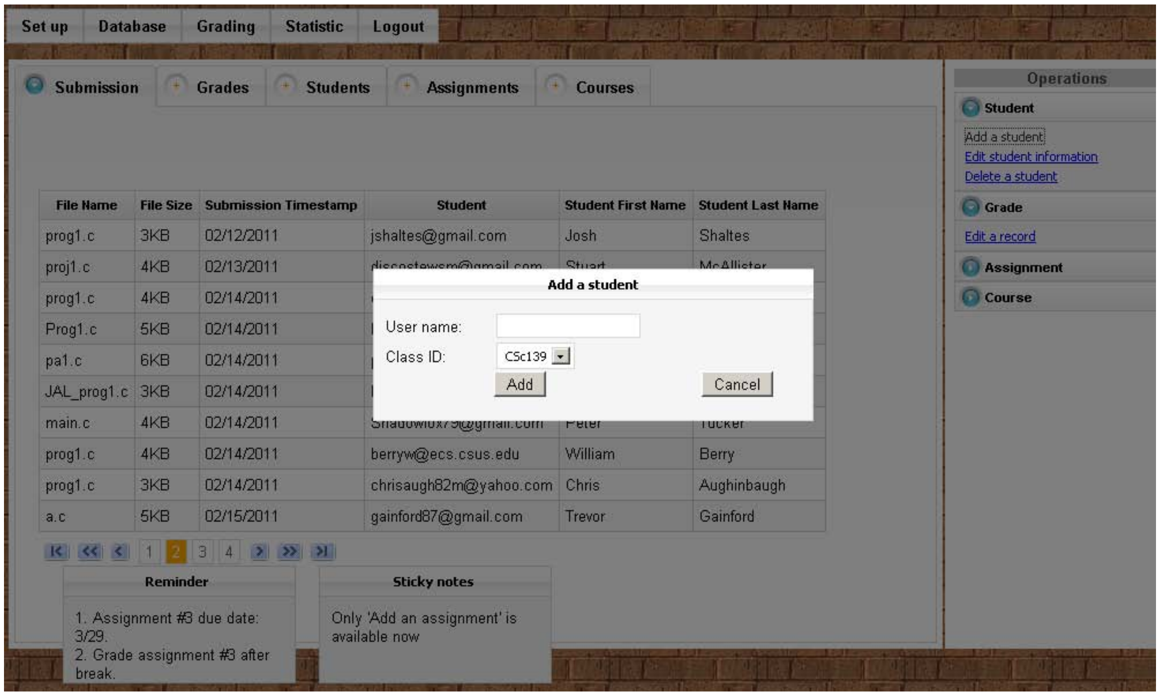
Only 'Add an assignment' is available now

Figure 5: Administrator Dashboard

Base on the requirement analysis of administrator, the dashboard for administrator provides these functions:

- View submission, grades, student information, assignments and courses.

- Edit grades, assignment information and course information.
- Add a new user, or delete a user.
- Start grading process, and set up test cases.



File Name	File Size	Submission Timestamp	Student	Student First Name	Student Last Name
prog1.c	3KB	02/12/2011	jshaltes@gmail.com	Josh	Shaltes
proj1.c	4KB	02/13/2011	discostewsm@gmail.com	Stuart	McAllister
prog1.c	4KB	02/14/2011			
Prog1.c	5KB	02/14/2011			
pa1.c	6KB	02/14/2011			
JAL_prog1.c	3KB	02/14/2011			
main.c	4KB	02/14/2011	Shadowfox79@gmail.com	Peter	Tucker
prog1.c	4KB	02/14/2011	berryw@ecs.csus.edu	William	Berry
prog1.c	3KB	02/14/2011	chrisaugh82m@yahoo.com	Chris	Aughinbaugh
a.c	5KB	02/15/2011	gainford87@gmail.com	Trevor	Gainford

Add a student

User name:

Class ID: CSc139

Add Cancel

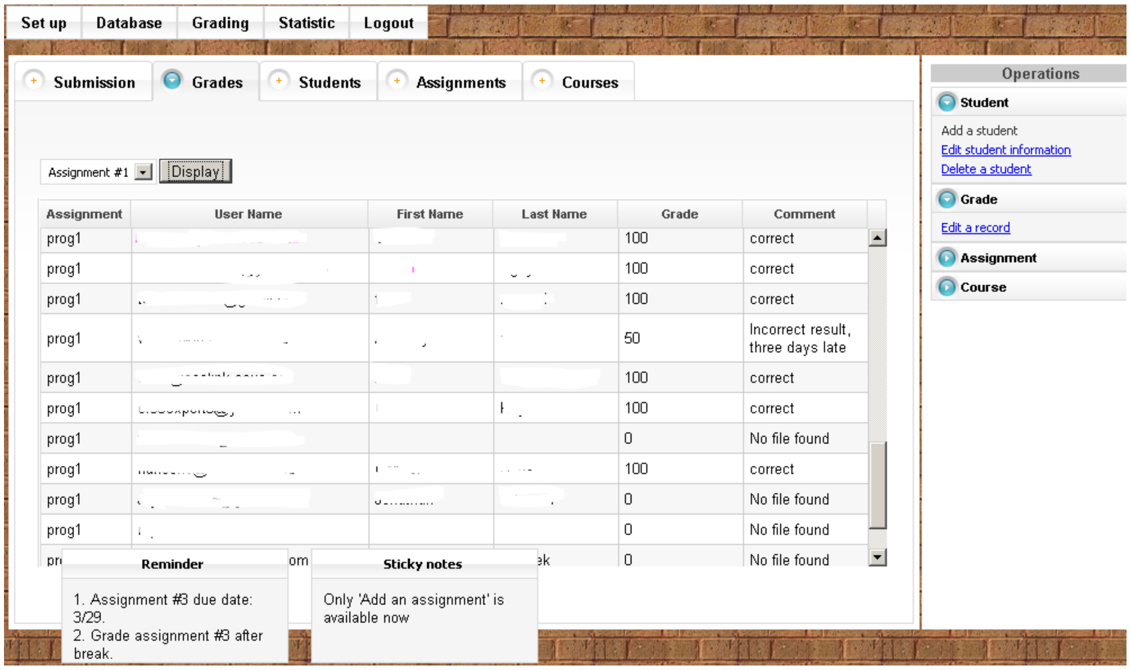
Reminder

1. Assignment #3 due date: 3/29.
2. Grade assignment #3 after break.

Sticky notes

Only 'Add an assignment' is available now

Figure 6: Add Student Function



Assignment	User Name	First Name	Last Name	Grade	Comment
prog1	[redacted]	[redacted]	[redacted]	100	correct
prog1	[redacted]	[redacted]	[redacted]	100	correct
prog1	[redacted]	[redacted]	[redacted]	100	correct
prog1	[redacted]	[redacted]	[redacted]	50	Incorrect result, three days late
prog1	[redacted]	[redacted]	[redacted]	100	correct
prog1	[redacted]	[redacted]	[redacted]	100	correct
prog1	[redacted]			0	No file found
prog1	[redacted]	[redacted]	[redacted]	100	correct
prog1	[redacted]	[redacted]	[redacted]	0	No file found
prog1	[redacted]			0	No file found
pr	om		ek	0	No file found

Reminder

1. Assignment #3 due date: 3/29.
2. Grade assignment #3 after break.

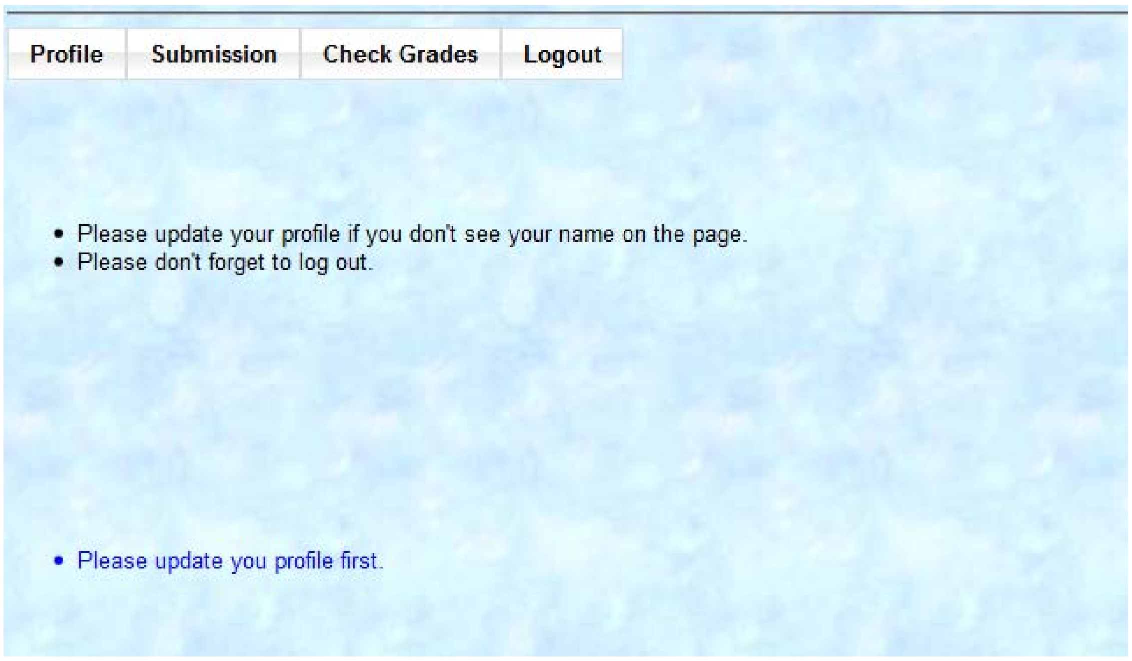
Sticky notes

Only 'Add an assignment' is available now

Figure 7: View Grades

User Interface for students should provide the functions of submitting source code and checking grades. According to the system constraints, student is not allowed to do anything unless he/she updates his/her profile; we add profile checking function to student interface. Student has to update his/her default password which was set by system, and provides his/her actual name before he/she uses any function of PAMS.

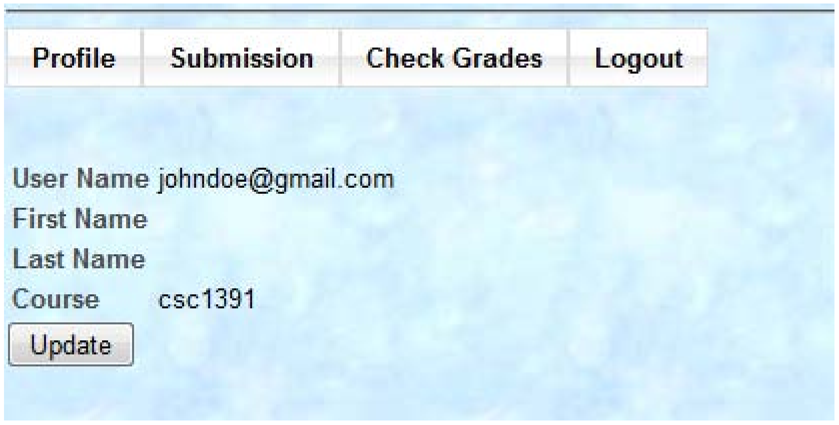
Here we make use of session map to check student's identity and attach student information to the submitted file.



- Please update your profile if you don't see your name on the page.
- Please don't forget to log out.

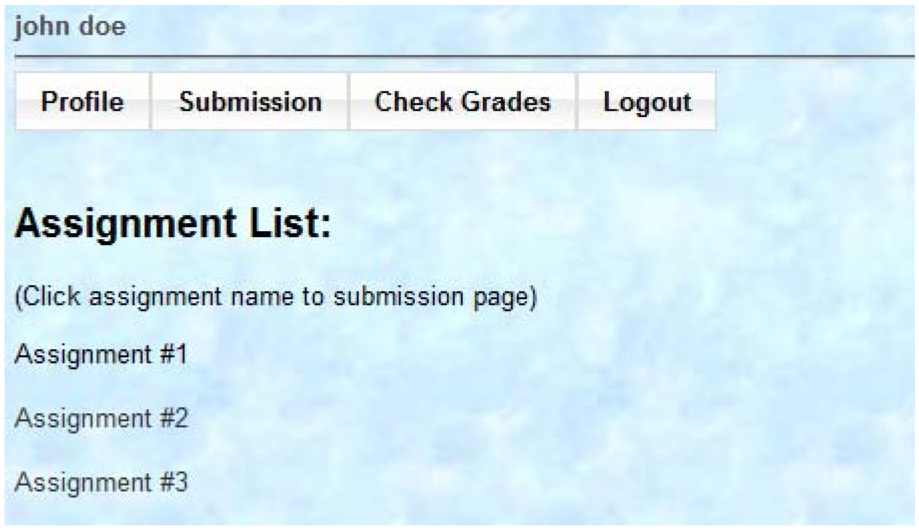
- Please update you profile first.

Figure 8: Student Dashboard



User Name johndoe@gmail.com
First Name
Last Name
Course csc1391

Figure 9: Student Profile Page



Assignment List:

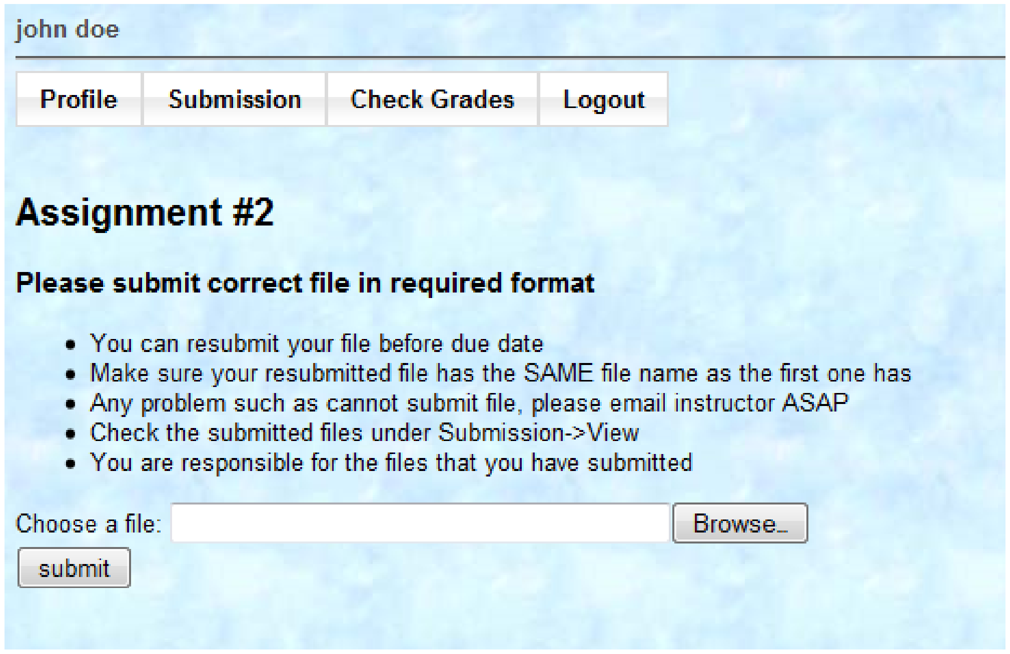
(Click assignment name to submission page)

Assignment #1

Assignment #2

Assignment #3

Figure 10: Assignment List Page



Assignment #2

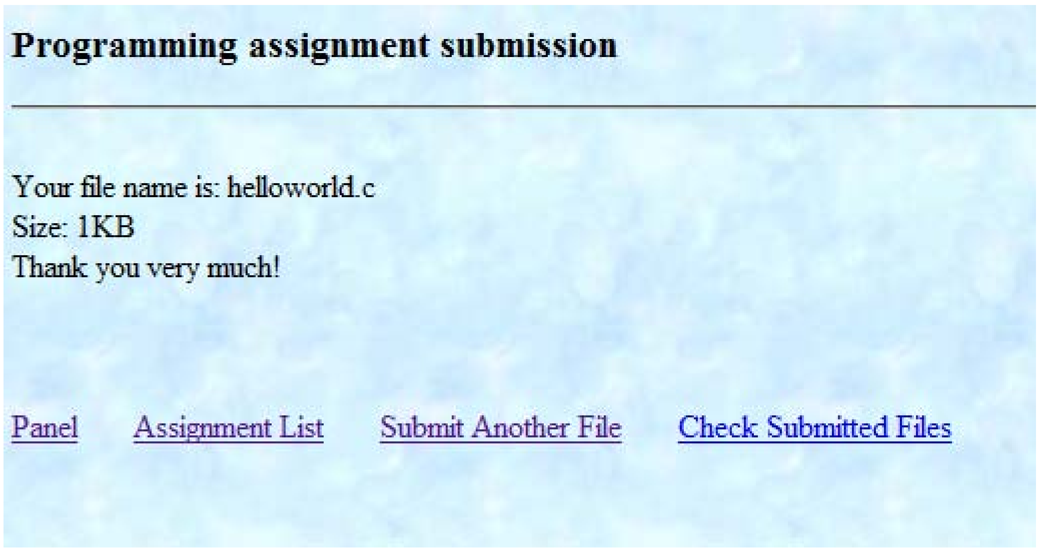
Please submit correct file in required format

- You can resubmit your file before due date
- Make sure your resubmitted file has the SAME file name as the first one has
- Any problem such as cannot submit file, please email instructor ASAP
- Check the submitted files under Submission->View
- You are responsible for the files that you have submitted

Choose a file: Browse...

submit

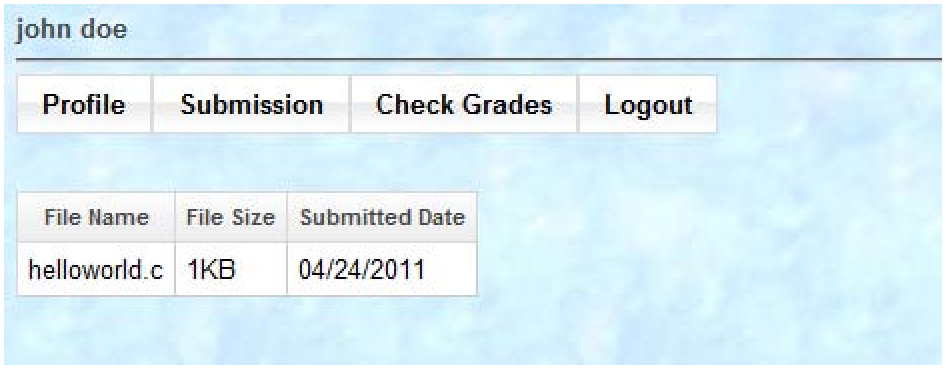
Figure 11: Submission Page



Programming assignment submission

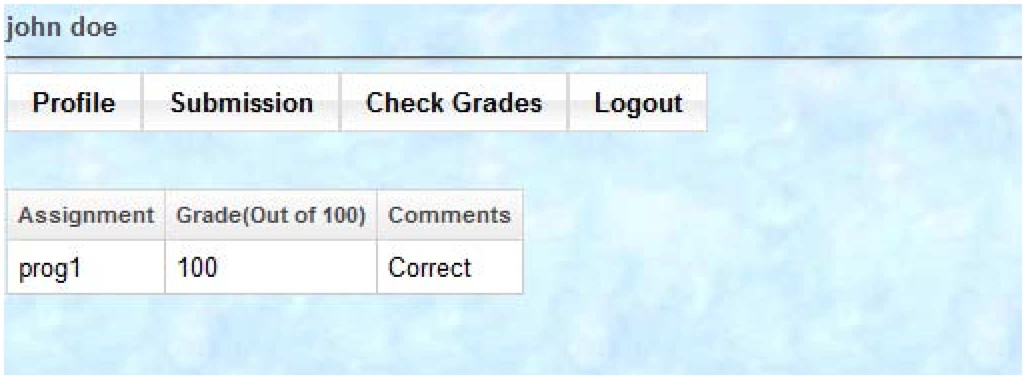
Your file name is: helloworld.c
Size: 1KB
Thank you very much!

Figure 12: Submission Finished



File Name	File Size	Submitted Date
helloworld.c	1KB	04/24/2011

Figure 13: Submission History



Assignment	Grade(Out of 100)	Comments
prog1	100	Correct

Figure 14: Students Check Grades

4.6. Grading Logic

Figure 15 is the flow chart for grading process.

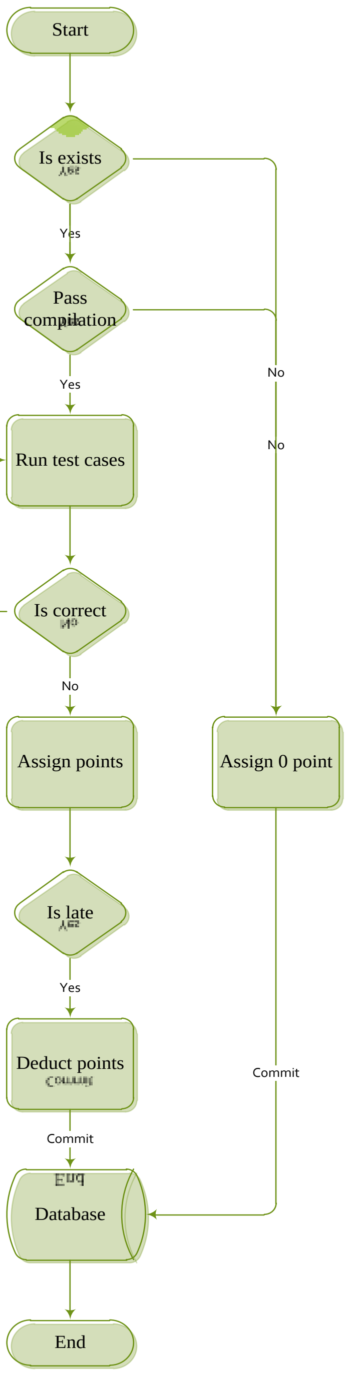
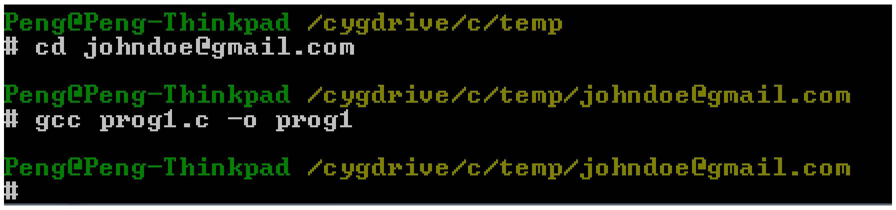


Figure 15: Grading Process Flow Diagram

There are several procedures we need to consider before actually grading the programs.

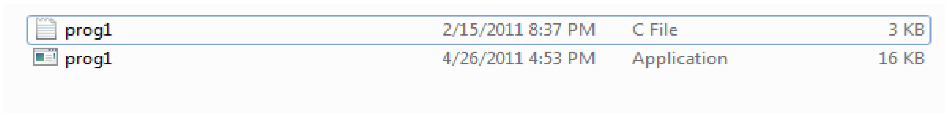
- GCC in Windows

  Because we have Cygwin, a Unix/Linux like environment, in Window, C programs can be compiled by GCC within Cygwin, figure 15 shows the command of calling gcc in Cygwin, figure 16 shows the compiled file in Windows Explorer.



```
Peng@Peng-Thinkpad /cygdrive/c/temp
# cd johndoe@gmail.com

Peng@Peng-Thinkpad /cygdrive/c/temp/johndoe@gmail.com
# gcc prog1.c -o prog1

Peng@Peng-Thinkpad /cygdrive/c/temp/johndoe@gmail.com
#
```

Figure 16: Call GCC in Cygwin

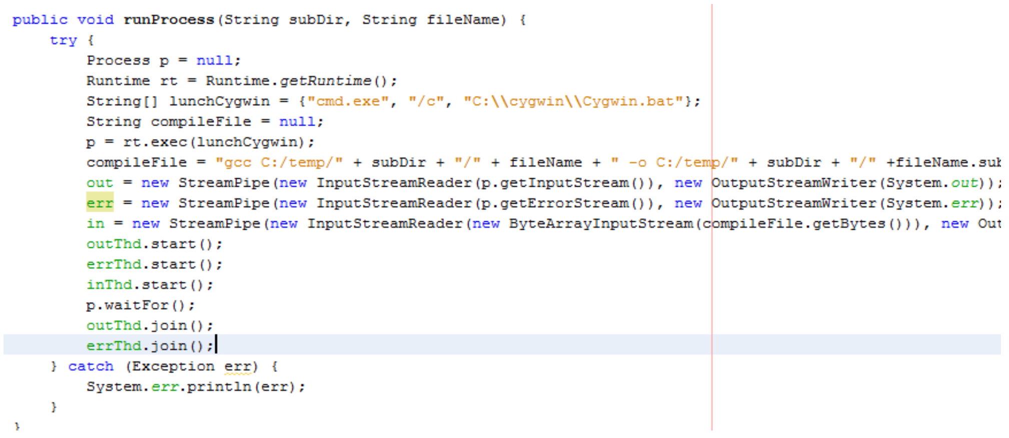


```
prog1        2/15/2011 8:37 PM    C File         3 KB
prog1        4/26/2011 4:53 PM    Application   16 KB
```

Figure 17: Compiled file in Windows Explorer

  So far, multi-processed C program, multi-threaded C program and C program using semaphores and pthread are tested fine in Cygwin environment; however, GCC in Cygwin has some problems occasionally, a few compiled files run with errors in Windows, but they run totally fine on Unix/Linux after they were compiled by GCC in Unix/Linux.

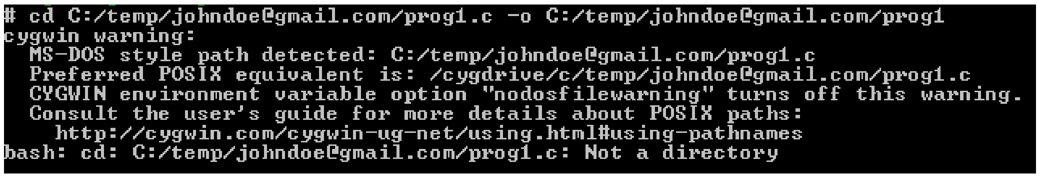
- Methods for calling GCC in Grading logic



```
public void runProcess(String subDir, String fileName) {
    try {
        Process p = null;
        Runtime rt = Runtime.getRuntime();
        String[] lunchCygwin = {"cmd.exe", "/c", "C:\\cygwin\\Cygwin.bat"};
        String compileFile = null;
        p = rt.exec(lunchCygwin);
        compileFile = "gcc C:/temp/" + subDir + "/" + fileName + " -o C:/temp/" + subDir + "/" +fileName.sub
        out = new StreamPipe(new InputStreamReader(p.getInputStream()), new OutputStreamWriter(System.out));
        err = new StreamPipe(new InputStreamReader(p.getErrorStream()), new OutputStreamWriter(System.err));
        in = new StreamPipe(new InputStreamReader(new ByteArrayInputStream(compileFile.getBytes())), new Out
        outThd.start();
        errThd.start();
        inThd.start();
        p.waitFor();
        outThd.join();
        errThd.join();
    } catch (Exception err) {
        System.err.println(err);
    }
}
```

Figure 18: Method for calling GCC

Even though the result of compilation is very important to PAMS, the process of compilation is also import to instructor for the purpose that instructor can observe the details of compilation. Figure 19 shows an error was found in compilation.



```
# cd C:/temp/johndoe@gmail.com/prog1.c -o C:/temp/johndoe@gmail.com/prog1
cygwin warning:
  MS-DOS style path detected: C:/temp/johndoe@gmail.com/prog1.c
  Preferred POSIX equivalent is: /cygdrive/c/temp/johndoe@gmail.com/prog1.c
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
bash: cd: C:/temp/johndoe@gmail.com/prog1.c: Not a directory
```

Figure 19: Error in compilation

Therefore, we need a mechanism to display these messages, in other words, we not only want to input commands to console, but also we would like to see the responses from console. So a class called StreamPipe is deployed in grading logic of PAMS. The class creates an input stream and an output stream which take care of the communication between the PAMS and GCC. Any method that wants to use StreamPipe class should provide two threads for input stream and output stream to ensure that two streams work individually without influencing the other one.

- Test the result

Once the GCC finish the compilation, the compiled files which are executable files in Windows are created in the same directory of source code, the easiest way of check the results of these programs is running them in windows command line (cmd) directly. PAMS do this for instructors as well. It uses Process object in Java to iteratively run the programs, and it records the output of each program by storing it in a Map object with a user id as key. After PAMS catches all the results, it iteratively matches the result to the correct answer, and it records all the grades simultaneously.

- Methods for grading programs
    - First of all, system checks submission and maintain a score table in memory.

      ```
      //Use UserInfo session bean to retrieve all student objects.
      List scoreTable = getUserInfoFacade().findAll;
      ```

      If Student does not submit the assignment or his/her source code cannot pass compilation, PAMS gives them a score and update the score table.
    - Secondly, system checks the correctness of source code. This requires administrator set up some test cases, and their corresponding correct results. Administrator could specify the exact result for one test case, or administrator could specify the key words which could be ordered or disordered he/she is look for.

      PAMS is able to match the correct result and output of source code, or it searches for the key words in the output of source code. Then PAMS gives a grade base on the weight of each test case that set up by administrator. After that, PAMS updates the score table.

  - Next step is that PAMS check the assignment submission date, it deducts the points that set up by administrator from the grade that student gets in score table if the assignment is overdue.
  - At last, PAMS would submit the score table to database to make the changes permanently.

4.7. Tools for Design and Implementation

- Netbeans 6.9.1[15], Netbeans is an Integrated Development Environment for developing Java, PHP, Ruby and other projects [16].
- MySQL 5, MySQL is an open source relational database management system [17].
- ICEfaces plugin for Netbeans, ICEfaces is an open source Ajax framework for Java EE application, which can create server-based rich Internet application.

Chapter 5
PERFORMANCE EVALUATION

PAMS was used for CSc 139 during spring 2011, the system was running fine for both students and administrator; however, some bugs was found during test run.

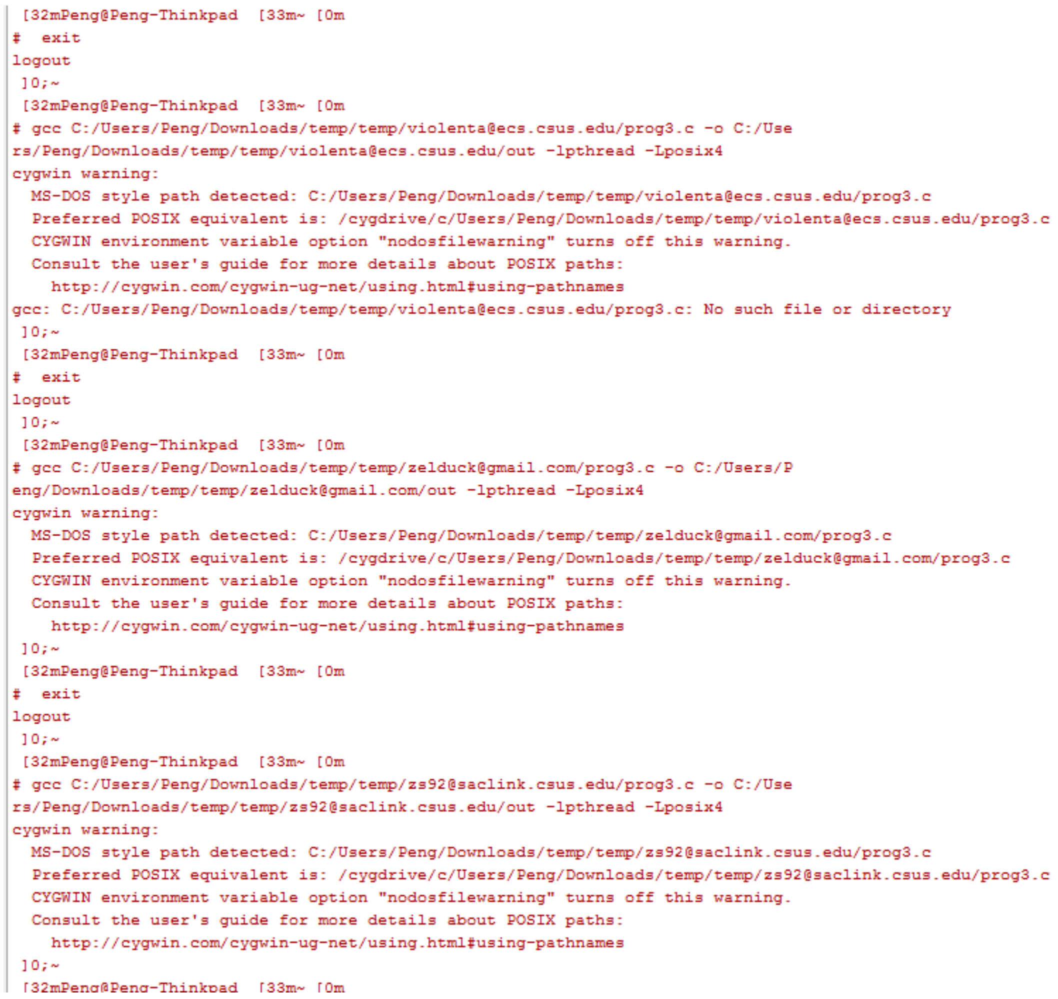
First assignment is requiring students to write a multi-processed C program for matrix multiplication. The command for compiling this program is: gcc prog1.c –o prog1.

Sample code for this command in PAMS:

```
compileFile = "gcc C:/temp/" + subDir + "/" + fileName + " -o C:/temp/" +
            subDir + "/" +fileName.substring(0, fileName.indexOf(".")) + parameter + "\n exit\n";
```

Usually, directory C:/temp/ is default directory for storing submissions. Because each student has his/her own folder in this directory, subDir is the folder name which is student's email address in PAMS. The variable fileName stands for the name of source code, instructor specifies the name of source code for each assignment, for example, prog1.c is the file name for assignment 1. The variable parameter is the parameter that used in compilation when there is required special library function. For example, if an assignment requires using Posix thread, the command should include –lpthread as parameter.

Figure 20 shows the some contents of the log file generated by PAMS during the compilation.



```
 [32mPeng@Peng-Thinkpad  [33m~ [0m
#  exit
logout
 ]0;~
 [32mPeng@Peng-Thinkpad  [33m~ [0m
# gcc C:/Users/Peng/Downloads/temp/temp/violenta@ecs.csus.edu/prog3.c -o C:/Use
rs/Peng/Downloads/temp/temp/violenta@ecs.csus.edu/out -lpthread -Lposix4
cygwin warning:
  MS-DOS style path detected: C:/Users/Peng/Downloads/temp/temp/violenta@ecs.csus.edu/prog3.c
  Preferred POSIX equivalent is: /cygdrive/c/Users/Peng/Downloads/temp/temp/violenta@ecs.csus.edu/prog3.c
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
gcc: C:/Users/Peng/Downloads/temp/temp/violenta@ecs.csus.edu/prog3.c: No such file or directory
 ]0;~
 [32mPeng@Peng-Thinkpad  [33m~ [0m
#  exit
logout
 ]0;~
 [32mPeng@Peng-Thinkpad  [33m~ [0m
# gcc C:/Users/Peng/Downloads/temp/temp/zelduck@gmail.com/prog3.c -o C:/Users/P
eng/Downloads/temp/temp/zelduck@gmail.com/out -lpthread -Lposix4
cygwin warning:
  MS-DOS style path detected: C:/Users/Peng/Downloads/temp/temp/zelduck@gmail.com/prog3.c
  Preferred POSIX equivalent is: /cygdrive/c/Users/Peng/Downloads/temp/temp/zelduck@gmail.com/prog3.c
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
 ]0;~
 [32mPeng@Peng-Thinkpad  [33m~ [0m
#  exit
logout
 ]0;~
 [32mPeng@Peng-Thinkpad  [33m~ [0m
# gcc C:/Users/Peng/Downloads/temp/temp/zs92@saclink.csus.edu/prog3.c -o C:/Use
rs/Peng/Downloads/temp/temp/zs92@saclink.csus.edu/out -lpthread -Lposix4
cygwin warning:
  MS-DOS style path detected: C:/Users/Peng/Downloads/temp/temp/zs92@saclink.csus.edu/prog3.c
  Preferred POSIX equivalent is: /cygdrive/c/Users/Peng/Downloads/temp/temp/zs92@saclink.csus.edu/prog3.c
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
 ]0;~
 [32mPeng@Peng-Thinkpad  [33m~ [0m
```

Figure 20: Contents in log during compilation

Second assignment is writing a multi-threaded C program to do matrix multiplication using pthread, therefore, instructor have to specify the parameter “-lpthread” in compile command. PAMS could finish grading this assignment nicely except that some programs cannot display results correctly in Windows environment. Fortunately, just a few programs suffer from this problem. Instructors have to manually run these programs on Unix/Linux machine.

Third assignment is writing a multi-threaded C program to copy file using semaphores. PAMS did a good job with having any problem.

- ***Problem 1***: Both students and administrator need logging out function. Originally, PAMS doesn't provide any log out function, student and administrator have to wait until the session expire.

  Solution: Add a new method called logout() in managed bean

  ```
  public String logout() {
          Map<String, Object> sessionMap =
  FacesContext.getCurrentInstance().getExternalContext().getSessionMap();
          sessionMap.remove("id");
          sessionMap.remove("currentUser");
          return "index?faces-redirect=true";
  }
  ```

- ***Problem 2***: Students need an assignment list page. Assignment list page could show how many assignments are available and how many assignments have been done.

  Solution: Add an assignment list page.

- ***Problem 3***: Some programs cannot display correctly in Windows environment.

  Solution: Manually run these programs in Unix/Linux environment. Another solution is making PAMS platform-independent; it is not difficult to do this since PAMS is written in Java. This is part of future work.

- ***Problem 4***: Preventing system user from visiting certain page by typing in URL address.

  Solution: Add a method called forwardToLoginIfNotLoggedIn() in authentication managed bean.

```
public void forwardToLoginIfNotLoggedIn(ComponentSystemEvent cse) {
    FacesContext fc = FacesContext.getCurrentInstance();
    String viewId = fc.getViewRoot().getViewId();
    if (!isUserLoggedIn()) {//if not loggedin, back to the login page
      fc.getApplication().getNavigationHandler().handleNavigation(fc, null,
          "/index?faces-redirect=true");
    }
}
```

- ***Problem 5***: Browser compatibility issue. PAMS has issues of rendering UI Components on Microsoft Internet Explorer.

  Solution: There is no solution for this problem right now, probably; next version of ICEfaces framework could perfectly work with Microsoft Internet Explorer.

Chapter 6

CONCLUSION

Since we can barely find a tool that satisfies the requirement of grading C program assignments automatically and managing student information, PAMS is the best choice for instructors who assign C programming assignments, additionally, because of the intuitive interface that PAMS provides, students do not feel strange or uncomfortable when they are using PAMS.

PAMS could dramatically improve the efficiency of grading a programming assignment written in C, so that instructors do not have to spend a lot of time to grade same programming assignment repeatedly. Also, PAMS is a good system for managing student information including his/her grades, name and email address. To compete with other teaching tools, PAMS is designed to be a customizable, user-friendly, reliable and secure system. There is no doubt that both students and instructors could obtain good user experience in using PAMS.

6.1. Future Work

There are a few improvements that probably could make PAMS better:

- Making PAMS platform-independent – This only requires some modification in grading logic module.
- Adding email function – Undoubtedly, using email is one of the easiest ways to contact a person today, adding email function would help PAMS to publish information or broadcast news quickly.
- Adding text message reminder for mobile devices – Mobile devices are very popular today, PAMS would become more user-friendly, if it could send reminders to student’s iPhone or Android devices.

- Grading programming assignments written in other languages – This requires a compiler for other languages. Even though PAMS only includes C compiler currently, it does provides APIs of calling Java compiler.
- Fixing browser compatibility issue – This could be done when next version of ICEfaces framework renders UI Components perfectly on IE browser.

APPENDIX A

User Manual

Student

- Login
  1. Type in URL http://servername:8080/pams in address bar of browser, then hit enter button.
  2. In login page, type in user name and password correctly, and then click on login button.
- Dashboard

  There is a horizontal navigation bar located near the top of the page, it has four menu items: profile, submission, check grades and logout. Above the navigation bar, your name should appear here if you updated your profile before.
- Update profile

  If you do not see your name appears on the top of the dashboard page, please follow these steps to update profile:

  1. Point to the Profile button in the navigation bar, and click on view button.
  2. You could see your user name, first name, last name and course number on your profile page, click on update button.
  3. You can choose to change your password, and you have to type in your first name and last name in corresponding text box, then hit the confirm button, system will redirect you to the login page.

4. Log into the system.

- Submit assignment
    1. Hover mouse cursor on the submission button in navigation bar, and then click on submit.
    2. In the assignment list page, choose an appropriate assignment by clicking the assignment name.
    3. In the submission page, click on Browse button, and then choose a file from local machine, click submit button to submit this file.
    4. You could see file name and size of your submitted file if the file was submitted successfully. You can choose to go back to dashboard, check assignment list, submit another file for this assignment and check submission history by clicking the hyperlinks provided on the page.
- Check grades
    1. In dashboard page, click on the check grades button in navigation bar.
    2. You can see a grades table with table headers: assignment, grade and comment.
- Logout

  Click on the logout button in navigation bar in dashboard page

Administrator

- Login
    1. Type in URL http://servername:8080/pams in address bar of browser, then hit enter button.

2. In login page, type in user name and password correctly, and then click on login button.

- Dashboard

  Dashboard contains a horizontal navigation bar, a panel which is composed by five tab panels (submission, grade, student, assignment, and course) and an operation panel.

- View data

  You can view all the submissions, grades, student information, assignment details and course information by simply clicking on corresponding tab names in the center panel of dashboard

- Setup the system

  Before every semester, setting up system is required. There are steps:

  1. Click on set up button in navigation bar in dashboard page.
  2. Input class information including class name, instructor, full credit for this class, semester, section number, and then click on next button.
  3. Manually add students' information to system database by providing student's user name (recommend email address) and class id. Click on more button to add another student, click on done button to finish setup.
  4. There is another way to add students' information, you can create a plain text file which contains student's user name and class id, then name it student.txt and save the file to student folder in PAMS directory (C:\pams\student). The format of data in student.txt should be:

     Student1 user name, class id |

     Student2 user name, class id |

     …

- Start grading process
    1. Click on grading button in navigation bar in dashboard page.
    2. Type in a test case for this assignment and corresponding correctly result or key words.
    3. Choose between matching results and finding key words.
    4. Choose ordered or disordered if you want PAMS to find key words in the output of student’s program. (The order of key words matters if you select this)
    5. If you want to see results of compilation, click on compile button. Then click on test results to see final results, PAMS will automatically update database when this step finished.
    6. If you do not want to see any results right now, just click on compile and test result button at the bottom.
    7. When grading process finished, click on dashboard hyperlink to go back to dashboard.
- Add a student
    1. In dashboard page, click on add a student on the operation panel at the right side of page.
    2. Type in student user name, and select a class id from drop-down list in the pop-up panel.
    3. Click on add button to finish adding a student.
- Delete a student
    1. In dashboard page, click on student tab in the center panel.
    2. Select a student by clicking the corresponding row in the table.
    3. Click on delete this student on the operation panel at the right side of page.

4. Click on confirm in the pop-up panel.

- Add an assignment
    1. Expend the operation panel in dashboard page, click on add an assignment.
    2. Type in information for assignment name, assignment type, due date, cutoff date, weight, full credit and description in the pop-up panel.
    3. Click on add button to finish add an assignment.
- Delete an assignment
    1. Expend the operation panel in dashboard page, click on delete an assignment.
    2. Select the assignment that you want to delete in the drop-down list in the pop-up panel.
    3. Click on confirm to delete this assignment.
- Edit grades
    1. In dashboard page, click on grade tab in the center panel.
    2. Select an assignment, and then click on display button to show grades for selected assignment.
    3. Select a grade by clicking the corresponding row in the table.
    4. Click on edit this record in the operation panel at the right side of dashboard page.
    5. Type in a new grade in the pop-up panel.
    6. Click on update button to finish editing.

APPENDIX B

Source Code

- index.xhml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:icecore="http://www.icefaces.org/icefaces/core"
      xmlns:ice="http://www.icesoft.com/icefaces/component"
      xmlns:ace="http://www.icefaces.org/icefaces/components"
      xmlns:h="http://java.sun.com/jsf/html">

    <body>

        <ui:composition template="./template/StandardTemplate.xhtml
            <ui:define name="content">
                <ice:form>
                    <fieldset style="width: 300px">
                        <legend>Login</legend>
                        <table>
                            <tr>
                                <td>
                                    <ice:outputLabel value="User Name"/>
                                </td>
                                <td>
                                    <ice:inputText value="#{userinfoController.id}" style="width:
200px"/>
                                </td>
                            </tr>
                            <tr>
                                <td>
                                    <ice:outputLabel value="Password"/>
                                </td>
                                <td>
                                    <ice:inputSecret value="#{userinfoController.password}"
style="width: 200px"/>
                                </td>
                            </tr>
                            <tr>
                                <td></td>
                                <td><ice:commandButton value="login"
action="#{userinfoController.login}"/></td>
                            </tr>
                        </table>
                    </fieldset>
                    <br/>
                    <ice:outputText value="#{userinfoController.message}"/>
                </ice:form>
            </ui:define>
        </ui:composition>
    </body>
</html>
```

- panel.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!--
  Document   : panel
  Created on : Jan 18, 2011, 12:34:14 PM
  Author     : Peng
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:icecore="http://www.icefaces.org/icefaces/core"
      xmlns:ace="http://www.icefaces.org/icefaces/components"
      xmlns:ice="http://www.icesoft.com/icefaces/component"
      >
    <h:head>
    <title>Sacramento State - Computer Science</title>
    <link rel="stylesheet" type="text/css" href="./xmlhttp/css/rime/rime.css"/>
    <style type="text/css">
        body {
            background-image: url('resources/pic/icecrystal.jpg');
        }
    </style>
  </h:head>
  <h:body>
    <ice:form>
        <ice:outputLabel value="#{submissionController.fname}  "/>
        <ice:outputLabel value="#{submissionController.lname}"/>
        <hr/>
        <ice:menuBar orientation="horizontal">
            <ice:menuItem value="Profile">
                <ice:menuItem value="View" link="profile.xhtml"/>
            </ice:menuItem>
            <ice:menuItem value="Submission">
                <ice:menuItem value="Submit"
action="#{submissionController.goSubmit}"/>
                <ice:menuItem value="View" action="#{submissionController.goView}"/>
            </ice:menuItem>
            <ice:menuItem value="Check Grades" link="viewGrade.xhtml">
            </ice:menuItem>
            <ice:menuItem value="Logout" action="#{submissionController.logout}">
            </ice:menuItem>
        </ice:menuBar>
        <br/><br/><br/><br/><br/><br/>
        <ul>
            <li>Please update your profile if you don't see your name on the page.</li>
            <li>Please don't forget to log out.</li>
        </ul>
        <br/>
        <br/><br/><br/><br/><br/><br/><br/>

    </ice:form>
  </h:body>
</html>
```

- submission.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!--
  Document   : submission
  Created on : Jan 21, 2011, 12:07:32 AM
  Author     : Peng
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
          xmlns:ui="http://java.sun.com/jsf/facelets"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
      xmlns:icecore="http://www.icefaces.org/icefaces/core"
     xmlns:ace="http://www.icefaces.org/icefaces/components"
     xmlns:ice="http://www.icesoft.com/icefaces/component"
     >
  <h:head>
    <title>Sacramento State - Computer Science</title>
    <link rel="stylesheet" type="text/css" href="./xmlhttp/css/rime/rime.css"/>
    <style type="text/css">
      body {
        background-image: url('resources/pic/icecrystal.jpg');
      }
    </style>
  </h:head>
  <h:body>
    <ice:form>
      <ice:outputLabel value="#{submissionController.fname}  "/>
      <ice:outputLabel value="#{submissionController.lname}"/>
      <hr/>
      <ice:menuBar orientation="horizontal">
```

```
            <ice:menuItem value="Profile">
                <ice:menuItem value="View" link="profile.xhtml"/>
            </ice:menuItem>
            <ice:menuItem value="Submission">
                <ice:menuItem value="Submit"
action="#{submissionController.goSubmit}"/>
                <ice:menuItem value="View" action="#{submissionController.goView}"/>
            </ice:menuItem>
            <ice:menuItem value="Check Grades" link="viewGrade.xhtml">
            </ice:menuItem>
            <ice:menuItem value="Logout" action="#{submissionController.logout}">
            </ice:menuItem>
        </ice:menuBar>
        <br/><br/><br/>
    </ice:form>
    <h2>#{redirectSubmission.subName}</h2>
    <h3>Please submit correct file in required format</h3>
    <ul>
        <li>You can resubmit your file before due date</li>
        <li>Make sure your resubmitted file has the SAME file name as the first one
has</li>
        <li>Any problem such as cannot submit file, please email instructor ASAP</li>
        <li>Check the submitted files under Submission->View</li>
        <li>You are responsible for the files that you have submitted</li>
    </ul>
    <form name="upload" enctype="multipart/form-data" method="post"
action="/pams/Upload">
        Choose a file: <input type="file" name="in_file" size="40"/>
        <br/>
        <input type="submit" value="submit"/>
        <br/><br/><br/><br/>
        <h:link value="Back" outcome="assignmentList.xhtml"/>
    </form>
  </h:body>
</html>
```

- profile.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!--
  Document   : profile
  Created on : Feb 8, 2011, 11:34:56 PM
  Author     : Peng
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
```

```
          xmlns:h="http://java.sun.com/jsf/html"
        xmlns:icecore="http://www.icefaces.org/icefaces/core"
       xmlns:ace="http://www.icefaces.org/icefaces/components"
       xmlns:ice="http://www.icesoft.com/icefaces/component"
       >
    <h:head>
        <title>Sacramento State - Computer Science</title>
        <link rel="stylesheet" type="text/css" href="./xmlhttp/css/rime/rime.css"/>
        <style type="text/css">
            body {
                background-image: url('resources/pic/icecrystal.jpg');
            }
        </style>
    </h:head>
    <h:body>
        <ice:form>
            <ice:outputLabel value="#{submissionController.fname}  "/>
            <ice:outputLabel value="#{submissionController.lname}"/>
            <hr/>
            <ice:menuBar orientation="horizontal">
                <ice:menuItem value="Profile">
                    <ice:menuItem value="View" link="profile.xhtml"/>
                </ice:menuItem>
                <ice:menuItem value="Submission">
                    <ice:menuItem value="Submit"
action="#{submissionController.goSubmit}"/>
                    <ice:menuItem value="View" action="#{submissionController.goView}"/>
                </ice:menuItem>
                <ice:menuItem value="Check Grades" link="viewGrade.xhtml">
                </ice:menuItem>
                <ice:menuItem value="Logout" action="#{submissionController.logout}">
                </ice:menuItem>
            </ice:menuBar>
            <br/><br/><br/><br/>
            <ice:panelGrid columns="2">
                <ice:outputLabel value="User Name"/>
                <ice:outputText value="#{viewProfile.uid}"/>

                <ice:outputLabel value="First Name"/>
                <ice:outputText value="#{viewProfile.fname}"/>

                <ice:outputLabel value="Last Name"/>
                <ice:outputText value="#{viewProfile.lname}"/>

                <ice:outputLabel value="Course"/>
                <ice:outputText value="#{viewProfile.cid}"/>
            </ice:panelGrid>
            <ice:commandButton value="Update" action="#{viewProfile.redir}"/>
            <br/><br/><br/>
            <h:link value="Back" outcome="panel.xhtml"/>
        </ice:form>
    </h:body>
</html>
```

- dashboard.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!--
    Document   : dataBase
    Created on : Mar 23, 2011, 12:57:08 AM
    Author     : Peng
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:icecore="http://www.icefaces.org/icefaces/core"
      xmlns:ace="http://www.icefaces.org/icefaces/components"
      xmlns:ice="http://www.icesoft.com/icefaces/component"
      >
    <h:head>
        <title>Dashboard</title>
        <link rel="stylesheet" type="text/css" href="./xmlhttp/css/rime/rime.css"/>
        <style type="text/css">
            body {
                /*background-color: #ADD8E6;*/
                background-image: url('resources/pic/brick_small.jpg');
            }
        </style>
    </h:head>
    <h:body>
        <ice:form id="main">


            <ice:panelGrid columns="1">
                <ice:menuBar orientation="horizontal">
                    <ice:menuItem value="Set up">
                        <ice:menuItem value="Preset system" link="regclass.xhtml"/>
                        <ice:menuItem value="Change settings" link="changeSetting.xhtml"/>
                    </ice:menuItem>
                    <ice:menuItem value="Database" link="dataBase.xhtml">
                    </ice:menuItem>
                    <ice:menuItem value="Grading" link="gradeAssignments.xhtml"/>
                    <ice:menuItem value="Statistic" link=""/>
                    <ice:menuItem value="Logout" action="#{submissionController.logout}">
    </ice:menuItem>
</ice:menuBar>

<!--
Floating Box
-->
<ice:panelPopup rendered="true" draggable="true" style="z-index: 1000; top:
64%; left: 1%; position: absolute; width: 200px; height: 100px;">
    <f:facet name="header">
        <ice:panelGrid width="100%" cellpadding="0" cellspacing="0"
columns="1" style="text-align: center;">
            <ice:outputText style="color:black;" value="Reminder"/>
        </ice:panelGrid>
    </f:facet>
    <f:facet name="body">
        <ice:panelGrid width="100%" cellpadding="0" cellspacing="0"
columns="1">
            <ice:outputText value="1. Assignment #3 due date: 3/29."/>
            <ice:outputText value="2. Grade assignment #3 after break."/>
        </ice:panelGrid>
    </f:facet>
</ice:panelPopup>
<ice:panelPopup rendered="true" draggable="true" style="z-index: 1000; top:
64%; left: 18%; position: absolute; width: 200px; height: 100px;">
    <f:facet name="header">
        <ice:panelGrid width="100%" cellpadding="0" cellspacing="0"
columns="1" style="text-align: center;">
            <ice:outputText style="color:black;" value="Sticky notes"/>
        </ice:panelGrid>
    </f:facet>
    <f:facet name="body">
        <ice:panelGrid width="100%" cellpadding="0" cellspacing="0"
columns="1">
            <ice:outputText value="Only 'Add an assignment' is available now"/>
        </ice:panelGrid>
    </f:facet>
</ice:panelPopup>
<!--
Floating Box End
-->
<ice:panelBorder renderCenter="true" renderEast="true">
    <f:facet name="center">
        <ice:panelTabSet width="800" height="500">
            <ice:panelTab label="Submission">
                <ice:dataTable id="view_upload_table"
value="#{viewUpload.allList}" var="file" rows="10">
                    <ice:rowSelector value="#{viewUpload.selectedFile}"
selectionAction="#{viewUpload.rowSelectionListener}"/>
```

```
                        <ice:column>
                          <f:facet name="header">
                            <ice:commandSortHeader columnName="File Name"
arrow="true">
                              <ice:outputText value="File Name"/>
                            </ice:commandSortHeader>
                          </f:facet>
                          <ice:outputText value="#{file.fname}"/>
                        </ice:column>
                        <ice:column>
                          <f:facet name="header">
                            <ice:outputText value="File Size"/>
                          </f:facet>
                          <ice:outputText value="#{file.fsize}"/>
                        </ice:column>
                        <ice:column>
                          <f:facet name="header">
                            <ice:outputText value="Submission Timestamp">
                            </ice:outputText>
                          </f:facet>
                          <ice:outputText value="#{file.fts}">
                            <f:converter converterId="com.pz.converter.DateTime"/>
                          </ice:outputText>
                        </ice:column>
                        <ice:column>
                          <f:facet name="header">
                            <ice:outputText value="Student"/>
                          </f:facet>
                          <ice:outputText value="#{file.userinfo.id}"/>
                        </ice:column>
                        <ice:column>
                          <f:facet name="header">
                            <ice:outputText value="Student First Name"/>
                          </f:facet>
                          <ice:outputText value="#{file.userinfo.fname}"/>
                        </ice:column>
                        <ice:column>
                          <f:facet name="header">
                            <ice:outputText value="Student Last Name"/>
                          </f:facet>
                          <ice:outputText value="#{file.userinfo.lname}"/>
                        </ice:column>
                      </ice:dataTable>
                      <ice:dataPaginator id="uploadPaginator"
                                  for="view_upload_table"
                                  paginator="true"
                                  fastStep="3"
                                  paginatorMaxPages="4"
```

```
                    >
                  <f:facet name="first">
                     <ice:graphicImage
                        url="./xmlhttp/css/xp/css-images/arrow-first.gif"
                        style="border:none;"
                        title="First Page"/>
                  </f:facet>
                  <f:facet name="last">
                     <ice:graphicImage
                        url="./xmlhttp/css/xp/css-images/arrow-last.gif"
                        style="border:none;"
                        title="Last Page"/>
                  </f:facet>
                  <f:facet name="previous">
                     <ice:graphicImage
                        url="./xmlhttp/css/xp/css-images/arrow-previous.gif"
                        style="border:none;"
                        title="Previous Page"/>
                  </f:facet>
                  <f:facet name="next">
                     <ice:graphicImage
                        url="./xmlhttp/css/xp/css-images/arrow-next.gif"
                        style="border:none;"
                        title="Next Page"/>
                  </f:facet>
                  <f:facet name="fastforward">
                     <ice:graphicImage url="./xmlhttp/css/xp/css-images/arrow-
ff.gif"
                                   style="border:none;"
                                   title="Fast Forward"/>
                  </f:facet>
                  <f:facet name="fastrewind">
                     <ice:graphicImage url="./xmlhttp/css/xp/css-images/arrow-
fr.gif"
                                   style="border:none;"
                                   title="Fast Backwards"/>
                  </f:facet>

               </ice:dataPaginator>
            </ice:panelTab>
            <ice:panelTab label="Grades">
               <ice:selectOneMenu value="#{viewGrades.assignmentName}">
                  <f:selectItem itemLabel="Assignment #1" itemValue="csc13911"/>
                  <f:selectItem itemLabel="Assignment #2" itemValue="csc13912"/>
                  <f:selectItem itemLabel="Assignment #3" itemValue="csc13913"/>
                  <f:selectItem itemLabel="Assignment #4" itemValue="csc13914"/>
                  <f:selectItem itemLabel="Assignment #5" itemValue="csc13915"/>
                  <f:selectItem itemLabel="Assignment #6" itemValue="csc13916"/>
```

```
                        </ice:selectOneMenu>
                        <ice:commandButton value="Display"
action="#{viewGrades.retrieveResult}"/>
                        <br/>
                        <ice:dataTable id="view_all_grades"
value="#{viewGrades.resultByAssignment}" var="grade" scrollable="true"
scrollHeight="300px" columnWidths="70px, 200px, 100px, 100px, 100px, 100px">
                           <ice:rowSelector />
                           <ice:column>
                              <f:facet name="header">
                                 <ice:outputLabel value="Assignment"/>
                              </f:facet>
                              <ice:outputText value="#{grade.assignment.aname}"/>
                           </ice:column>
                           <ice:column>
                              <f:facet name="header">
                                 <ice:outputLabel value="User Name"/>
                              </f:facet>
                              <ice:outputText value="#{grade.userinfo.id}"/>
                           </ice:column>
                           <ice:column>
                              <f:facet name="header">
                                 <ice:outputLabel value="First Name"/>
                              </f:facet>
                              <ice:outputText value="#{grade.userinfo.fname}"/>
                           </ice:column>
                           <ice:column>
                              <f:facet name="header">
                                 <ice:outputLabel value="Last Name"/>
                              </f:facet>
                              <ice:outputText value="#{grade.userinfo.lname}"/>
                           </ice:column>
                           <ice:column>
                              <f:facet name="header">
                                 <ice:outputLabel value="Grade"/>
                              </f:facet>
                              <ice:outputText value="#{grade.point}"/>
                           </ice:column>
                           <ice:column>
                              <f:facet name="header">
                                 <ice:outputLabel value="Comment"/>
                              </f:facet>
                              <ice:outputText value="#{grade.comment}"/>
                           </ice:column>
                        </ice:dataTable>

                     </ice:panelTab>
                     <ice:panelTab label="Students">
```

```
                                <ice:dataTable id="view_all_students"
value="#{viewStudent.userList}" var="user" rows="10">
                                  <ice:column>
                                    <ice:rowSelector
selectionListener="#{viewStudent.rowSelectionListener}" multiple="false"/>
                                    <f:facet name="header">
                                      <ice:outputText value="User Name"/>
                                    </f:facet>
                                    <ice:outputText value="#{user.id}"/>
                                  </ice:column>
                                  <ice:column>
                                    <f:facet name="header">
                                      <ice:outputText value="Class ID"/>
                                    </f:facet>
                                    <ice:outputText value="#{user.cid}"/>
                                  </ice:column>
                                  <ice:column>
                                    <f:facet name="header">
                                      <ice:outputText value="User First Name"/>
                                    </f:facet>
                                    <ice:outputText value="#{user.fname}"/>
                                  </ice:column>
                                  <ice:column>
                                    <f:facet name="header">
                                      <ice:outputText value="User Last Name"/>
                                    </f:facet>
                                    <ice:outputText value="#{user.lname}"/>
                                  </ice:column>
                                  <ice:column>
                                    <f:facet name="header">
                                      <ice:outputText value="User Password"/>
                                    </f:facet>
                                    <ice:outputText value="#{user.password}"/>
                                  </ice:column>
                                </ice:dataTable>
                                <ice:dataPaginator id="viewStudentPaginator"
                                          for="view_all_students"
                                          paginator="true"
                                          fastStep="3"
                                          paginatorMaxPages="4"
                                          >
                                  <f:facet name="first">
                                    <ice:graphicImage
                                      url="./xmlhttp/css/xp/css-images/arrow-first.gif"
                                      style="border:none;"
                                      title="First Page"/>
                                  </f:facet>
                                  <f:facet name="last">
```

```
                                        <ice:graphicImage
                                            url="./xmlhttp/css/xp/css-images/arrow-last.gif"
                                            style="border:none;"
                                            title="Last Page"/>
                                    </f:facet>
                                    <f:facet name="previous">
                                        <ice:graphicImage
                                            url="./xmlhttp/css/xp/css-images/arrow-previous.gif"
                                            style="border:none;"
                                            title="Previous Page"/>
                                    </f:facet>
                                    <f:facet name="next">
                                        <ice:graphicImage
                                            url="./xmlhttp/css/xp/css-images/arrow-next.gif"
                                            style="border:none;"
                                            title="Next Page"/>
                                    </f:facet>
                                    <f:facet name="fastforward">
                                        <ice:graphicImage url="./xmlhttp/css/xp/css-images/arrow-
ff.gif"
                                                          style="border:none;"
                                                          title="Fast Forward"/>
                                    </f:facet>
                                    <f:facet name="fastrewind">
                                        <ice:graphicImage url="./xmlhttp/css/xp/css-images/arrow-
fr.gif"
                                                          style="border:none;"
                                                          title="Fast Backwards"/>
                                    </f:facet>

                                </ice:dataPaginator>
                            </ice:panelTab>
                            <ice:panelTab label="Assignments">
                                <ice:dataTable value="#{viewAssignment.assignmentList}"
var="assignment">
                                    <ice:rowSelector />
                                    <ice:column>
                                        <f:facet name="header">
                                            <ice:outputText value="Assignment No."/>
                                        </f:facet>
                                        <ice:outputText value="#{assignment.aid}"/>
                                    </ice:column>
                                    <ice:column>
                                        <!--<ice:rowSelector value="" multiple="false"
selectionListener=""/>-->
                                        <f:facet name="header">
                                            <ice:outputText value="Assignment Name"/>
                                        </f:facet>
```

```
            <ice:outputText value="#{assignment.aname}"/>
        </ice:column>
        <ice:column>
            <f:facet name="header">
                <ice:outputText value="Assignment Type"/>
            </f:facet>
            <ice:outputText value="#{assignment.atype}"/>
        </ice:column>
        <ice:column>
            <f:facet name="header">
                <ice:outputText value="Due Date"/>
            </f:facet>
            <ice:outputText value="#{assignment.duedate}">
                <f:convertDateTime type="date" dateStyle="default"/>
            </ice:outputText>
        </ice:column>
        <ice:column>
            <f:facet name="header">
                <ice:outputText value="Cutoff Date"/>
            </f:facet>
            <ice:outputText value="#{assignment.curoffdate}">
                <f:convertDateTime type="date" dateStyle="default"/>
            </ice:outputText>
        </ice:column>
        <ice:column>
            <f:facet name="header">
                <ice:outputText value="Weight"/>
            </f:facet>
            <ice:outputText value="#{assignment.weight}"/>
        </ice:column>
        <ice:column>
            <f:facet name="header">
                <ice:outputText value="Full Credit"/>
            </f:facet>
            <ice:outputText value="#{assignment.fullcredit}"/>
        </ice:column>
        <ice:column>
            <f:facet name="header">
                <ice:outputText value="Description"/>
            </f:facet>
            <ice:outputText value="#{assignment.description}"/>
        </ice:column>
    </ice:dataTable>
</ice:panelTab>
<ice:panelTab label="Courses">
    <ice:dataTable value="#{viewClass.courseList}" var="class">
        <ice:rowSelector />
        <ice:column>
```

```
                    <!--<ice:rowSelector value="" multiple="false"
selectionListener=""/>-->
                    <f:facet name="header">
                        <ice:outputText value="Course Name"/>
                    </f:facet>
                    <ice:outputText value="#{class.cname}"/>
                </ice:column>
                <ice:column>
                    <f:facet name="header">
                        <ice:outputText value="Grader"/>
                    </f:facet>
                    <ice:outputText value="#{class.grader}"/>
                </ice:column>
                <ice:column>
                    <f:facet name="header">
                        <ice:outputText value="Points"/>
                    </f:facet>
                    <ice:outputText value="#{class.totalcredit}"/>
                </ice:column>
                <ice:column>
                    <f:facet name="header">
                        <ice:outputText value="Semester"/>
                    </f:facet>
                    <ice:outputText value="#{class.semester}"/>
                </ice:column>
                <ice:column>
                    <f:facet name="header">
                        <ice:outputText value="Section"/>
                    </f:facet>
                    <ice:outputText value="#{class.session}"/>
                </ice:column>
            </ice:dataTable>
        </ice:panelTab>
    </ice:panelTabSet>
</f:facet>
<f:facet name="east">
    <ice:panelGrid width="200">
        <f:facet name="header">
            <ice:panelGrid width="100%" cellpadding="0" cellspacing="0"
columns="2" style="text-align: center; background-color: #cccccc;">
                <ice:outputLabel value="Operations"/>
            </ice:panelGrid>
        </f:facet>

        <ice:panelGroup menuPopup="menu_selection">
            <ice:panelGrid width="100%" cellpadding="0" cellspacing="0"
columns="1" >
                <ice:panelCollapsible expanded="true">
```

```
          <f:facet name="header">
             <ice:panelGroup>
                <ice:outputText value="Student"/>
             </ice:panelGroup>
          </f:facet>
          <ice:panelGroup>
             <ice:commandLink value="Add a student"
action="#{viewStudent.addStudent}">
             </ice:commandLink>
             <br/>
             <h:link outcome="index.xhtml" value="Edit student
information"/><br/>
             <h:link outcome="index.xhtml" value="Delete a
student"/><br/>
          </ice:panelGroup>
       </ice:panelCollapsible>
       <ice:panelCollapsible expanded="true">
          <f:facet name="header">
             <ice:panelGroup>
                <ice:outputText value="Grade"/>
             </ice:panelGroup>
          </f:facet>
          <ice:panelGroup>
             <ice:commandLink value="Add a record"
action="#{viewGrades.showPanel}"/><br/>
             <h:link outcome="index.xhtml" value="Edit a record"/><br/>
          </ice:panelGroup>
       </ice:panelCollapsible>
       <ice:panelCollapsible expanded="false">
          <f:facet name="header">
             <ice:panelGroup>
                <ice:outputText value="Assignment"/>
             </ice:panelGroup>
          </f:facet>
          <ice:panelGroup>
             <h:link outcome="index.xhtml" value="Add an
Assignment"/><br/>
             <h:link outcome="index.xhtml" value="Edit assignment
information"/><br/>
             <h:link outcome="index.xhtml" value="Delete an
assignment"/><br/>
          </ice:panelGroup>
       </ice:panelCollapsible>
       <ice:panelCollapsible expanded="false">
          <f:facet name="header">
             <ice:panelGroup>
                <ice:outputText value="Course"/>
             </ice:panelGroup>
```

```
                                </f:facet>
                                <ice:panelGroup>
                                    <h:link outcome="index.xhtml" value="Edit course
information"/><br/>
                                </ice:panelGroup>
                            </ice:panelCollapsible>

                        </ice:panelGrid>
                    </ice:panelGroup>
                </ice:panelGrid>
            </f:facet>
        </ice:panelBorder>
    </ice:panelGrid>
    <ice:menuPopup id="menu_selection">
        <ice:menuItem value="Delete"/>
        <ice:menuItem value="Edit"/>
    </ice:menuPopup>
</ice:form>
<!--
Popup panel for database operations
-->
<ice:form>
    <ice:panelPopup id="add_st" modal="true" rendered="true"
visible="#{viewStudent.showPanelAdd}" draggable="false" style="z-index: 1000; top:
30%; left: 5%; position: absolute; width: 380px; height: 130px;">
        <f:facet name="header">
            <ice:panelGrid width="380px" cellpadding="0" cellspacing="0"
columns="2" style="text-align: center;">
                <ice:outputText style="color:black;" value="Add a student"/>
            </ice:panelGrid>
        </f:facet>
        <f:facet name="body">
            <ice:panelGrid width="100%" cellpadding="0" cellspacing="0"
columns="3">
                <ice:outputText value="User name: "/>
                <ice:inputText id="st_name" value="#{viewStudent.userName}">
                    <f:validator validatorId="com.pz.validator.emailValidator"/>
                    <f:ajax execute="@this" event="blur"/>
                </ice:inputText>
                <h:message for="st_name" style="color:red"/>
                <ice:outputText value="Class ID: "/>
                <ice:selectOneMenu id="cl_id" value="#{viewStudent.classId}">
                    <f:selectItems value="#{viewStudent.courseList}"/>
                </ice:selectOneMenu>
                <h:message for="cl_id" style="color:red"/>
                <ice:outputText/>
                <ice:commandButton value="Add"
action="#{viewStudent.addStudentConfirm}"/>
```

```
                        <ice:commandButton value="Cancel"
action="#{viewStudent.addStudentCancel}"/>
                    </ice:panelGrid>
                </f:facet>
            </ice:panelPopup>
        </ice:form>
        <ice:form>
            <ice:panelPopup id="op_st" modal="true" rendered="true"
visible="#{viewStudent.showPanelOp}" draggable="false" style="z-index: 1000; top:
30%; left: 5%; position: absolute; width: 380px; height: 200px;">
                <f:facet name="header">
                    <ice:panelGrid width="380px" cellpadding="0" cellspacing="0"
columns="2" style="text-align: center;">
                        <ice:outputText style="color:black;" value="Edit Student Info"/>
                    </ice:panelGrid>
                </f:facet>
                <f:facet name="body">
                    <ice:panelGrid width="100%" cellpadding="0" cellspacing="0"
columns="3">
                        <ice:outputText value="User Id: "/>
                        <ice:outputText value="#{viewStudent.selectedUser.id}"/>
                        <ice:outputText/>
                        <ice:outputText value="User Password: "/>
                        <ice:inputText value="#{viewStudent.newPassword}"/>
                        <ice:outputText/>
                        <ice:outputText value="Class"/>
                        <ice:selectOneMenu value="#{viewStudent.classId}">
                            <f:selectItems value="#{viewStudent.courseList}"/>
                        </ice:selectOneMenu>
                        <ice:outputText/>
                        <ice:outputText value="First Name:"/>
                        <ice:inputText value="#{viewStudent.fName}"/>
                        <ice:outputText/>
                        <ice:outputText value="Last Name:"/>
                        <ice:inputText value="#{viewStudent.lName}"/>
                        <ice:outputText/>
                        <ice:commandButton value="Update"
action="#{viewStudent.editStudent}"/>
                        <ice:commandButton value="Delete"
action="#{viewStudent.delStudent}"/>
                        <ice:commandButton value="Cancel"
action="#{viewStudent.editStudentCancel}"/>
                    </ice:panelGrid>
                </f:facet>
            </ice:panelPopup>
            <!--
            <ice:panelPopup id="add_re" modal="true" rendered="true"
visible="#{viewGrades.isShow}" draggable="false" style="z-index: 1000; top: 30%; left:
5%; position: absolute; width: 380px; height: 180px;">
                <f:facet name="header">
                    <ice:panelGrid width="380px" cellpadding="0" cellspacing="0"
columns="2" style="text-align: center;">
                        <ice:outputText style="color:black;" value="Add a record"/>
                    </ice:panelGrid>
                </f:facet>
                <f:facet name="body">

                    <ice:panelGrid width="100%" cellpadding="0" cellspacing="0"
columns="3">
                        <ice:outputText value="Student user name: "/>
                        <ice:selectOneMenu>
                            <f:selectItem itemValue="a" itemLabel="a"/>
                        </ice:selectOneMenu>
                        <ice:outputText value=""/>
                        <ice:outputText value="Select assignment: "/>
                        <ice:selectOneMenu>
                            <f:selectItems value="#{viewStudent.courseList}" var="course"
itemLabel="#{course.cname}" itemValue="#{course.cid}"/>
                            <f:selectItem itemLabel="csc13911" itemValue="csc13911"/>
                        </ice:selectOneMenu>
                        <ice:outputText/>
                        <ice:outputText value="Points"/>
                        <ice:inputText value=""/>
                        <ice:outputText/>
                        <ice:outputText value="Comment"/>
                        <ice:inputText value=""/>
                        <ice:outputText/>
                        <ice:outputText value=""/>
                        <ice:commandButton value="Add"
action="#{viewStudent.addStudentConfirm}"/>
                        <ice:commandButton value="Cancel"
action="#{viewStudent.addStudentCancel}"/>
                    </ice:panelGrid>

                </f:facet>
            </ice:panelPopup>-->
        </ice:form>
    </h:body>
</html>
```

- AuthenticationBean.java

```
package com.pz.authentication;

import java.io.Serializable;
import java.util.Map;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.ComponentSystemEvent;

/**
 *
 * @author Peng
 */
@ManagedBean
@SessionScoped
public class AuthenticationBean implements Serializable {
    private Map<String, Object> sessionMap;

    /** Creates a new instance of AuthenticationBean */
    public AuthenticationBean() {
    }

    public Map<String, Object> getSessionMap() {
        FacesContext context = FacesContext.getCurrentInstance();
        sessionMap = context.getExternalContext().getSessionMap();
        return sessionMap;
    }

    public void forwardToLoginIfNotLoggedIn(ComponentSystemEvent cse) {
        FacesContext fc = FacesContext.getCurrentInstance();
        String viewId = fc.getViewRoot().getViewId();
        if (!isUserLoggedIn()) {//if not loggedin, back to the login page
            fc.getApplication().getNavigationHandler().handleNavigation(fc, null,
                    "/index?faces-redirect=true");
        }
    }

    public boolean isUserLoggedIn() {
        return getSessionMap().containsKey("currentUser");
    }


}
```

- SubmissionController.java

```
package com.pz.controller;

import com.pz.persistence.Userinfo;
import com.pz.sessionbean.UserinfoFacade;
import java.util.Map;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.faces.application.FacesMessage;

/**
 *
 * @author Peng
 */
@ManagedBean
@RequestScoped
public class SubmissionController {

    @EJB com.pz.sessionbean.UserinfoFacade ejbFacade;
    private String id;
    private Userinfo currentUser;
    private String fname;
    private String lname;

    /** Creates a new instance of SubmissionController */
    public SubmissionController() {
    }

    private UserinfoFacade getFacade() {
        return ejbFacade;
    }


    public String getId() {
        Map<String, Object> sessionMap =
FacesContext.getCurrentInstance().getExternalContext().getSessionMap();
        id = sessionMap.get("id").toString();
        return id;
    }

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

    public Userinfo getCurrentUser() {
        return currentUser;
    }

    public void setCurrentUser(Userinfo currentUser) {
        this.currentUser = currentUser;
    }
    public String getFname() {
        currentUser = getFacade().find(this.getId());
        fname = currentUser.getFname();
        return fname;
    }

    public void setFname(String fname) {
        this.fname = fname;
    }

    public String getLname() {
        lname = currentUser.getLname();
        return lname;
    }

    public void setLname(String lname) {
        this.lname = lname;
    }

    public String logout() {
        Map<String, Object> sessionMap = 
FacesContext.getCurrentInstance().getExternalContext().getSessionMap();
        sessionMap.remove("id");
        sessionMap.remove("currentUser");
        return "index?faces-redirect=true";
    }

    public String goSubmit() {
        System.out.println(this.getFname());
        if (this.getFname().equals("")) {
            FacesContext fc = FacesContext.getCurrentInstance();
            FacesMessage fm = new FacesMessage("Please update you profile first.");
            fc.addMessage(null, fm);
            return null;
        } else {
            return "assignmentList?faces-redirect=true";
        }
    }

    public String goView() {
        if(this.getFname().equals("")) {
            FacesContext fc = FacesContext.getCurrentInstance();
            FacesMessage fm = new FacesMessage("Please update you profile first.");
            fc.addMessage(null, fm);
            return null;
        } else {
            return "viewUpload?faces-redirect=true";
        }
    }
}
```

- UserinfoController.java

```
package com.pz.controller;

import com.pz.persistence.Userinfo;
import com.pz.sessionbean.UserinfoFacade;
import java.util.Map;
import javax.ejb.EJB;
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;

/**
 *
 * @author Peng
 */
@ManagedBean
@RequestScoped
public class UserinfoController {

    @EJB
    private com.pz.sessionbean.UserinfoFacade ejbFacade;
    private String id;
    private String password;
    private Userinfo current;
    private String message;

    private Map<String, Object> sessionMap;

    /** Creates a new instance of UserinfoController */
    public UserinfoController() {
    }

    public String getId() {
        return id;
    }

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

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

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

public String getMessage() {
    return message;
}

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

public Map<String, Object> getSessionMap() {
    FacesContext context = FacesContext.getCurrentInstance();
    sessionMap = context.getExternalContext().getSessionMap();
    return sessionMap;
}

public void setSessionMap(Map<String, Object> sessionMap) {
    this.sessionMap = sessionMap;
}

private UserinfoFacade getFacade() {
    return ejbFacade;
}

public Userinfo getCurrent() {
    return current;
}

public void setCurrent(Userinfo current) {
    this.current = current;
}



public String login() {
    current = getFacade().find(this.id);
    if (current.getPassword().equals(this.password)) {
        getSessionMap().put("currentUser", this.current);
        if (current.getRoletype().equals("ad")) {
            getSessionMap().put("id", current.getId());
            return "dataBase?faces-redirect=true";
        } else {
            getSessionMap().put("id", current.getId());
            System.out.println(getSessionMap().get("id").toString());
            return "panel?faces-redirect=true";
        }
        } else {
            this.message = "Please check your password";
            return "index";
        }

    }
}
```

- ViewUpload.java

```
package com.pz.managedbean;

import com.icesoft.faces.component.ext.RowSelectorEvent;
import com.pz.persistence.Fileinfo;
import com.pz.persistence.Userinfo;
import com.pz.sessionbean.FileinfoFacade;
import java.util.Comparator;
import java.util.List;
import java.util.Map;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.faces.model.ListDataModel;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**
 *
 * @author Peng
 */
@ManagedBean
@RequestScoped
public class ViewUpload {

    @EJB
    private com.pz.sessionbean.FileinfoFacade ejbFacade;
    @PersistenceContext
    EntityManager em;
    private List<Fileinfo> allList;
    private List<Fileinfo> result;
    private Userinfo currentUser;
    private List<Fileinfo> selectedFile;

    private String fileName;

    /** Creates a new instance of ViewUpload */
    public ViewUpload() {
    }
    private FileinfoFacade getFacade() {
        return ejbFacade;
    }

    public Userinfo getCurrentUser() {

        return currentUser;
    }

    public void setCurrentUser(Userinfo currentUser) {
        this.currentUser = currentUser;
    }

    public String getFileName() {
        return fileName;
    }

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



    public List<Fileinfo> getSelectedFile() {
        return selectedFile;
    }

    public void setSelectedFile(List<Fileinfo> selectedFile) {
        this.selectedFile = selectedFile;
    }



    public List<Fileinfo> getAllList() {
        allList = ejbFacade.findAll();
        return allList;
    }

    public void setAllList(List<Fileinfo> allList) {
        this.allList = allList;
    }

    public List<Fileinfo> getResult() {
        Map<String, Object> sessionMap =
FacesContext.getCurrentInstance().getExternalContext().getSessionMap();
```

```
        currentUser = (Userinfo) sessionMap.get("currentUser");
        this.result = em.createQuery("select f from Fileinfo f where
f.userinfo=:user").setParameter("user", this.currentUser).getResultList();
        //System.out.println(sid);
        return result;
    }

    public void setResult(List<Fileinfo> result) {
        this.result = result;
    }

    public void rowSelectionListener(RowSelectorEvent event) {
        System.out.println("selected");
        this.selectedFile = event.getSelectedRows();
        System.out.println(selectedFile.get(0).getFname());
        this.fileName = selectedFile.get(0).getFname();
    }
}
```

- ViewAssignment.java

```
package com.pz.managedbean;

import com.pz.persistence.Assignment;
import com.pz.sessionbean.AssignmentFacade;
import java.util.List;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.model.ListDataModel;

/**
 *
 * @author Peng
 */
@ManagedBean
@RequestScoped
public class ViewAssignment {

    @EJB private com.pz.sessionbean.AssignmentFacade ejbFacade;

    private ListDataModel<Assignment> assignmentList;
    private List<Assignment> result;

    /** Creates a new instance of ViewAssignment */
    public ViewAssignment() {
    }

    private AssignmentFacade getFacade() {
        return ejbFacade;
    }

    public ListDataModel<Assignment> getAssignmentList() {
        result = getFacade().findAll();
        assignmentList = new ListDataModel(result);
        return assignmentList;
    }

    public void setAssignmentList(ListDataModel<Assignment> assignmentList) {
        this.assignmentList = assignmentList;
    }

    public List<Assignment> getResult() {
        return result;
    }

    public void setResult(List<Assignment> result) {
        this.result = result;
    }
}
```

- ViewClass.java

```
package com.pz.managedbean;

import com.pz.persistence.Course;
import com.pz.sessionbean.CourseFacade;
import java.util.List;
import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;

/**
 *
 * @author Peng
 */
@ManagedBean
@RequestScoped
public class ViewClass {

    @EJB private com.pz.sessionbean.CourseFacade ejbFacade;

    private ListDataModel<Course> courseList;
    private List<Course> result;

    /** Creates a new instance of ViewClass */
    public ViewClass() {
    }

    private CourseFacade getFacade() {
        return ejbFacade;
    }

    public ListDataModel<Course> getCourseList() {
        result = getFacade().findAll();
        courseList = new ListDataModel(result);
        return courseList;
    }

    public void setCourseList(ListDataModel<Course> courseList) {
        this.courseList = courseList;
    }

    public List<Course> getResult() {
        return result;
    }

    public void setResult(List<Course> result) {
        this.result = result;
    }
}
```

- RunApp.java

```
package com.pz.logic;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

/**
 *
 * @author Peng
 */
public class RunApp {
    public String result = null;
    public String runApplication(String fileDir, String fileName, String parameter) throws
IOException, InterruptedException {

        Process p = null;
        //String result = null;
        String command = "C:\\TEMP\\" + fileDir + "\\" + fileName +" "+ parameter + "\n";
        //String sid = fileName.substring(0, fileName.indexOf("."));
        try {
            p = Runtime.getRuntime().exec(command);
            InputStream fis = p.getInputStream();
            InputStreamReader isr = new InputStreamReader(fis);
            BufferedReader br = new BufferedReader(isr);
            String line = null;
            //StringBuilder sb = new StringBuilder(fileDir + "'s result: ");
            StringBuilder sb = new StringBuilder();
            while ((line = br.readLine()) != null) {
                //System.out.println(line);
                sb.append(line);
            }
            //sb.append("\n\n");
            result = sb.toString();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return result;
    }
}
```

REFERENCE

[1] SacCT, https://online.csus.edu/webct/entryPageIns.dowebct

[2] WASG, Palnitkar Rahul, “A WEB BASED ASSIGNMENT SUBMISSION AND GRADING SYSTEM”, 2003

[3] Cygwin, http://en.wikipedia.org/wiki/Cygwin

[4] Java EE, http://download.oracle.com/javaee/6/firstcup/doc/

[5] GlassFish, http://glassfish.java.net/

[6] JavaServer Face, http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[7] Model-View-Controller, http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[8] Enterprise JavaBean, http://www.oracle.com/technetwork/java/javaee/ejb/index.html

[9] Java Persistence API, http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html

[10] MySQL, http://www.mysql.com/

[11] Stateful session bean, http://en.wikipedia.org/wiki/Enterprise_JavaBean#Stateful_Session_Beans

[12] Stateless session bean, http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/EJBConcepts3.html

[13] Singleton session bean, http://en.wikipedia.org/wiki/Enterprise_JavaBean#Singleton_Session_Beans

[14] JSF life cycle phases, http://download.oracle.com/javaee/1.4/tutorial/doc/JSFIntro10.html

[15] Netbeans, http://netbeans.org/

[16] Other projects, http://en.wikipedia.org/wiki/NetBeans

[17] Relational database management System, http://en.wikipedia.org/wiki/Relational_database_management_system