IOWA STATE UNIVERSITY

Graduate Student Review System

Group 16: Ashley Montebello, Katie Githens, Wayne Rowcliffe

Advisor: Akhilesh Tyagi

MAY11 4/27/2011

Table of Contents

List of Figures

List of Tables

Definitions

RP	Reporting Period
GProgress	The Department of Computer Science's current graduate student performance review system.
CoE	College of Engineering
ECpE	Electrical and Computer Engineering
ISU	Iowa State University
SAP	Satisfactory Academic Progress

Table 1: Definitions

Executive Summary

The Department of Electrical and Computer Engineering at Iowa State University needs to annually review its graduate students. This process involves self-evaluations, faculty-evaluations, and possibly review by the Graduate Committee. Currently, this process is not automated, and the ECpE graduate secretary must manually coordinate the requesting and compilation of evaluations.
A web-based application could significantly reduce the required effort by automating the task of sending annual notifications to the graduate students and faculty, tracking student progress as identified by both the student and faculty evaluations along with publications and awards the student has received, and selecting which students require additional consideration by the ECpE Graduate Committee.

The Department of Computer Science has a similar graduate review process. Dr. Wallapak Tavanapong and several graduate students have developed an automated system similar to the one desired by the department of ECpE. We have obtained permission from Dr. Tavanapong to use the source from her project as a basis for this project. It is developed in Java using Hibernate for database interaction and Struts for the web-presentation layer.

The College of Engineering maintains information on all current graduate students. If possible, we would like to use this information to populate the project's database. The College of Engineering's database would be polled annually before the start of the review process.

The university already has a login system in place. We would like to use this system rather than the authentication system currently used in Dr. Tavanapong's work. This will help the graduate review system to interface seamlessly with other university web applications.

The expected end product for this project is a solution that satisfies the above problem. It will be based on the previous work done by the Department of Computer Science. Modified versions of the evaluation forms will be created to suit the Department of ECpE's needs. The user interface of the website will fit more closely to ISU's template than the layout currently in place in Dr. Tavanapong's work. User authentication will be provided by the University's login system. Initial population of graduate student info will be obtained from the College of Engineering's record database. The project may be modified to fit other requirements not-yet-discovered.

In addition, we will produce documentation for the system. This will include documentation of the system's architecture and basic design. It will also include documentation on the use of the system for the various classes of users
(e.g. Student and Faculty).

Acknowledgements

We would like to thank the Department of Computer Science in the College of Liberal Arts and Science at Iowa State University for providing a reference implementation of our project. A special thanks to Dr. Wallapak Tavanapong and her team of graduate students who implemented the database and provided extensive documentation. Additionally, we would like to thank Dr. Carl Chang for allowing the Department of Electrical and Computer Engineering to utilize his department's resources.

System Requirements

The expected end product will consist of an online form-based system for end-users and a back-end SQL database for storing all information.

Front-End Requirements

1.1. All Users
- 1.1.1. Edit user account information
- 1.1.2. System must have a login system to authenticate the user

1.2. Administrator
- 1.2.1. Edit faculty information (name, email, students)
- 1.2.2. Edit student information (name, email, professor, program type)
- 1.2.3. Search for a course (by course name or section)
  - 1.2.3.1. System should display course information including: course name, section, and credit hours.
- 1.2.4. Add new courses (name, section/code, credit hours)
- 1.2.5. Remove a course
- 1.2.6. Edit course information (name, section/code, credit hours)
- 1.2.7. Create student accounts
- 1.2.8. Create faculty accounts
- 1.2.9. Edit rating scale used by students and faculty
- 1.2.10. Add and edit reporting periods
  - 1.2.10.1. Choose start/end date for period, deadline for students to enter information, professor deadline for entering information, rating scale.
- 1.2.11. Search for faculty/professor account by name or username
  - 1.2.11.1. System displays name, username, email and students
- 1.2.12. Search for student account by name or username
  - 1.2.12.1. System displays name, username, email, program type, major professor
- 1.2.13. Delete faculty/professor and student accounts

1.3. Students
- 1.3.1. Edit evaluation period information during a set period of time.

1.3.1.1. Courses taken, assistantships, research progress, rating
1.3.2. Submit evaluation for a given period
1.3.3. Edit background information

1.4. Faculty/Professors
1.4.1. Edit advisee rating by RP (Reporting Period)
1.4.2. View advisee rating by RP
1.4.3. Submit advisee rating by RP
1.5. Graduate Committee
1.5.1. View student and professor RP ratings for each student
1.5.2. Sort list of students by years and status

Back-End Requirements

The system must store the information in a MySQL database. MySQL is a multithreaded multi-user open source database management system, which is most widely used as part of web applications. Hibernate is used to map between the relational data in MySQL and Java objects. The database will consist of the tables listed below.

Course

Stores courses given by the ECpE department.

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	Primary	NULL	auto_increment
title	varchar(255)	YES	Unique	NULL	
courseCode	varchar(255)	YES	Unique	NULL	
credits	int(11)	YES		NULL	

Table 2: Course Database Table

Evaluation Period

Stores evaluation period in which students are required to input their progress.

Field	Type	Null	Key	Default	Extra
PID	int(11)	NO	Primary	NULL	auto_increment
period_from	datetime	YES		NULL	
period_to	datetime	YES		NULL	
studentDeadline	datetime	YES		NULL	
facultyDeadline	datetime	YES		NULL	
ratingGroupID	int(11)	YES		NULL	

Table 3: Evaluation Period Database Table

Evaluation Period Information

Stores student's progress in a specific period as well as student self evaluation and faculty ratings.

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	Primary	NULL	auto_increment
facultyComment	varchar(255)	YES		NULL	
comments	varchar(255)	YES		NULL	

comments1	varchar(255)	YES		NULL	
comments2	varchar(255)	YES		NULL	
assistantship	varchar(255)	YES		NULL	
personalSat	int(11)	YES		NULL	
assistantHours	int(11)	YES		NULL	
facultyRating	int(11)	YES		0	
selfRating	int(11)	YES		NULL	
SID	int(11)	YES		NULL	
FID	int(11)	YES		NULL	
PID	int(11)	YES		NULL	
fundingDept	varchar(255)	YES		NULL	
isSubmitted	varchar(10)	YES		NULL	
yearsLeft	varchar(30)	YES		NULL	
facultyComment1	varchar(2047)	YES		NULL	
facultyComment2	varchar(2047)	YES		NULL	
facultyComment3	varchar(2047)	YES		NULL	
facultyRecommend	bit(1)	NO		false	
facultyGradContact	bit(1)	NO		false	
overagedCourses	bit(1)	NO		false	

Table 4: Evaluation Period Info Database Table

Rating Scale

Contains the rating scale(s) used to rate student performance.

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
ID	int(11)	NO	Primary	NULL	auto_increment
rating	varchar(255)	YES		NULL	
meaning	varchar(255)	YES		NULL	
flag	varchar(255)	YES		NULL	
groupID	int(11)	YES		NULL	

Table 5: Rating Scale Database Table

Major Professor

Maps students to their major professor. This allows the professor to complete their evaluation for their students.

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
SID	int(11)	NO	Primary		
FID	int(11)	NO	Primary		

Table 6: Major Professor Database Table

Student Background Information

Stores background information on the student.

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
id	int(11)	NO	Primary	NULL	auto_increment
entryYear	datetime	YES		NULL	
careerObjective	varchar(255)	YES		NULL	

cmplCourse	varchar(255)	YES		NULL	
cmplCommitteeApproval	varchar(255)	YES		NULL	
cmplCommitteeApprovalDate	varchar(255)	YES		NULL	
cmplCourseSemester	varchar(255)	YES		NULL	
cmplCourseExplain	varchar(255)	YES		NULL	
cmplPOS	varchar(255)	YES		NULL	
cmplPOSDate	varchar(255)	YES		NULL	
phdProficiency	varchar(255)	YES		NULL	
phdProficiencySemester	varchar(255)	YES		NULL	
phdProficiencyExplain	varchar(255)	YES		NULL	
cmplPhdPrelims	varchar(255)	YES		NULL	
cmplPhdPrelimsSemester	varchar(255)	YES		NULL	
cmplPhdPrelimsExplain	varchar(255)	YES		NULL	
cmplPhdDefense	varchar(255)	YES		NULL	
cmplPhdDefenseSemester	varchar(255)	YES		NULL	
cmplPhdDefenseExplain	varchar(255)	YES		NULL	
researchCompleted	varchar(255)	YES		NULL	
noFirstAuthor	int(11)	YES		NULL	
noCoAuthor	int(11)	YES		NULL	
expectGradDate	datetime	YES		NULL	
progress	varchar(255)	YES		NULL	
progressExplain	varchar(255)	YES		NULL	
researchArea	varchar(255)	YES		NULL	
thesisTitle	varchar(255)	YES		NULL	

thesisURL	varchar(255)	YES		NULL	
contact	varchar(255)	YES		NULL	
url	varchar(255)	YES		NULL	
releaseInfo	varchar(255)	YES		NULL	

Table 7: Student Background Database Table

Student Account Information

Stores account information about the student. Student background information is mapped to the student using the stdBackground field.

Field	Type	Null	Key	Default	Extra
SID	int(11)	NO	Primary	NULL	auto_increment
univID	int(10)	YES		NULL	unsigned
lastname	varchar(255)	NO			
name	varchar(255)	YES		NULL	
username	varchar(255)	YES	Unique	NULL	
email	varchar(255)	YES		NULL	
password	varchar(255)	YES		NULL	
sec_question	varchar(255)	YES		NULL	
sec_answer	varchar(255)	YES		NULL	
program_type	varchar(255)	YES		NULL	
status	varchar(255)	YES		NULL	
stdBackground	int(11)	YES	Unique	NULL	
stdCV	int(11)	YES		NULL	USING BTREE

Table 8: Student Info Database Table

Student Course

Stores courses that the student has taken for each evaluation period. Courses are mapped to the student by the SID, mapped to the course details by the CID, and mapped to the evaluation period using the PID.

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	Primary	NULL	auto_increment
SID	int(11)	NO			
PID	int(11)	NO			
CID	int(11)	YES		NULL	

Table 9: Student Course Database Table

Book Chapter

Stores details of book chapters written by students, if any. Book chapters are mapped to students by the student ID (SID) as a foreign key.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	Primary	NULL	auto_increment
chapterTitle	varchar(255)	YES		NULL	

authors	varchar(255)	YES		NULL	
bookName	varchar(255)	YES		NULL	
publisher	varchar(255)	YES		NULL	
bookEditors	varchar(255)	YES		NULL	
date	datetime	YES		NULL	
SID	int(11)	NO			

Table 10: Book Chapter Database Table

Student Software

Stores information about software written by students, if any. Students are mapped to their developed software by their SID.

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
ID	int(11)	NO	Primary	NULL	auto_increment
title	varchar(255)	YES		NULL	
publicationDate	datetime	YES		NULL	
softwareAbstract	varchar(255)	YES		NULL	
SID	int(11)	NO			

Table 11: Student Software Database Table

Technical Presentation

Stores information on technical presentations done by students, if any. Students are mapped to their technical presentations by their SID.

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
ID	int(11)	NO	Primary	NULL	auto_increment
conference	varchar(255)	YES		NULL	
title	varchar(255)	YES		NULL	
presentAbstract	varchar(255)	YES		NULL	
presentDate	datetime	YES		NULL	
SID	int(11)	NO			

Table 12: Technical Presentation Database Table

Technical Report

Stores information on technical reports written by students, if any. Students are mapped to their technical reports by their SID.

Field	**Type**	**Null**	**Key**	**Default**	**Extra**
ID	int(11)	NO	Primary	NULL	auto_increment
title	varchar(255)	YES		NULL	
type	varchar(255)	YES		NULL	
reportAbstract	varchar(255)	YES		NULL	
publicationDate	datetime	YES		NULL	
conferenceJournal	varchar(255)	YES		NULL	
SID	int(11)	NO			

Table 13: Technical Report Database Table

Previous Degree

Stores information on student's previous degrees. Students are mapped to their previous degrees by their SID.

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	Primary	NULL	auto_increment
degree	varchar(255)	YES		NULL	
major	varchar(255)	YES		NULL	
gpa	double	YES		NULL	
year_received	datetime	YES		NULL	
university	varchar(255)	YES		NULL	
SID	int(11)	NO			

Table 14: Previous Degree Database Table

Publication

Stores information on any publications written by the student. Students are mapped to publications using their SID.

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	Primary	NULL	auto_increment
title	varchar(255)	YES		NULL	
type	varchar(255)	YES		NULL	
authors	varchar(255)	YES		NULL	
issue	varchar(255)	YES		NULL	
pages	int(11)	YES		NULL	
volumn	int(11)	YES		NULL	
referred	varchar(255)	YES		NULL	
first_author	varchar(255)	YES		NULL	
conferenceJournal	varchar(255)	YES		NULL	
acceptDate	datetime	YES		NULL	
submitDate	datetime	YES		NULL	
printDate	datetime	YES		NULL	
SID	int(11)	NO			

Table 15: Student Publication Database Table

Faculty/Staff Account Information

Stores account information for faculty. Faculty can be marked as administrators and/or a member of the Graduate Committee.

Field	Type	Null	Key	Default	Extra
FID	int(11)	NO	Primary	NULL	auto_increment
univID	int(10)	YES		NULL	unsigned
lastname	varchar(255)	NO			
name	varchar(255)	YES		NULL	
username	varchar(255)	YES		NULL	
email	varchar(255)	YES		NULL	
password	varchar(255)	YES		NULL	
sec_question	varchar(255)	YES		NULL	
sec_answer	varchar(255)	YES		NULL	
status	varchar(255)	YES		NULL	
url	varchar(255)	YES		NULL	

isGradCommittee	bit(1)	NO		false	
isAdmin	bit(1)	NO		false	

Table 16: Faculty/Staff Account Info Database Table

Non-Functional System Requirements

1. Security
    1.1. System will use IAState Login system for login credentials
2. Extensibility
    2.1. Integrate with College of Engineering database of graduate students

Software Design

This section describes the overall software design of the system for each process. All functions of the system follow the same process for information retrieval, presentation, and updating.

Apache Struts

In order to follow the typical client-server system architecture, our system will follow the MVC (Model-View-Controller) architecture.

Model

The model contains the core of the application's functionality and encapsulates the state of the application. Its only functionality is to maintain the state of the application at various instances and has no knowledge about the view or controller.

View

The view provides the presentation of the model. It can access the model getters, but it has no knowledge of the setters. In addition, it knows nothing about the controller. The view should be notified when changes to the model occur.

Controller

The controller reacts to the user input. It creates and sets the model.

Struts falls under the category of technologies that provides a unified framework for deploying JavaServer Pages (JSP) and servlet applications that use the MVC architecture. It also offers a variety of utility classes to handle web application development in the form of custom tag libraries for HTML forms.

The various components of a Struts application involve JSP, which serve as the "View" component, Controller Servlet which function as the "Controller" and Action Forms, ActionForm Beans, which serve as the "Model". The diagram below depicts a conceptual view of the struts architecture.

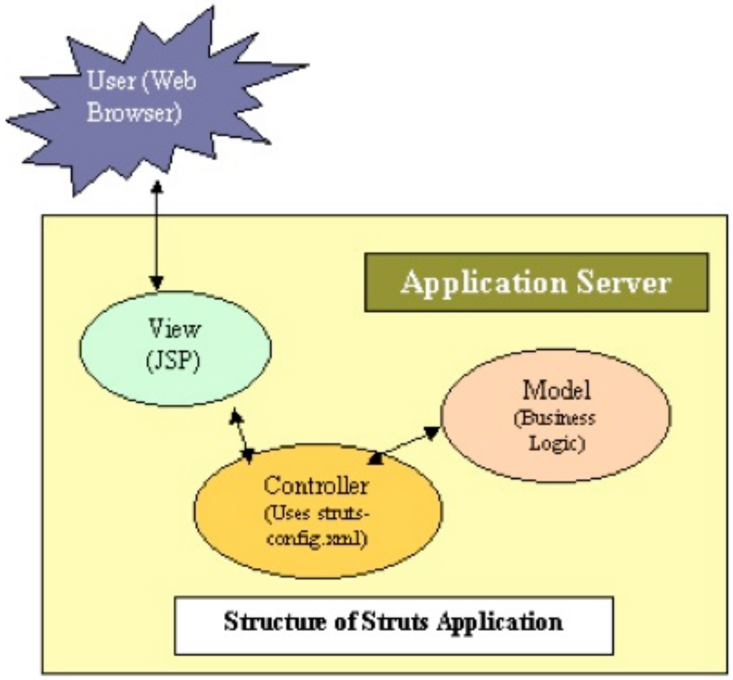


Figure 1: System Structure

Struts manage the communication between the various actions and action forms through a mapping scheme stored in “struts-config.xml” and “web.xml”. A typical application involving Struts would use JSP pages to receive user data, validate the data using an Action Form Bean, process the data and interact with various external resources such as network, database using an Action and display the results back to the user through a JSP or a HTML page.

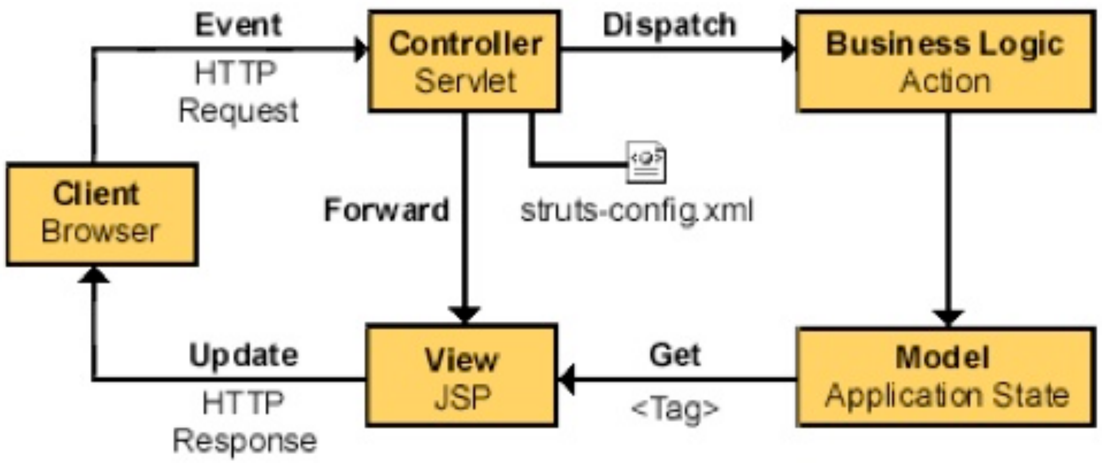


Figure 2: Struts Architecture

Hibernate

An open source object-relation mapping (ORM) solution that maps Java objects to relational databases. Hibernate relieves the developer from significant amount of common data persistence- related programming tasks. It provides data query and retrieval facilities and significantly reduces time spent on manually handling SQL and JDBC calls.

For each table in the database, Hibernate requires a JavaBean that match the attributes of the table. For each attribute, there should be a getter and a setter. Also, a constructor with no parameters is required. Finally, an id attribute is created for each class, so that Hibernate can assign an id for each object. Each JavaBean class is mapped to the corresponding table using an XML mapping file.

XML files are used to configure Hibernate. Namely, the file "hibernate.cfg.xml" describes the connection driver (JDBC in this case), the database URL, the username and password used to access the database and the references all XML mapping files used by Hibernate.

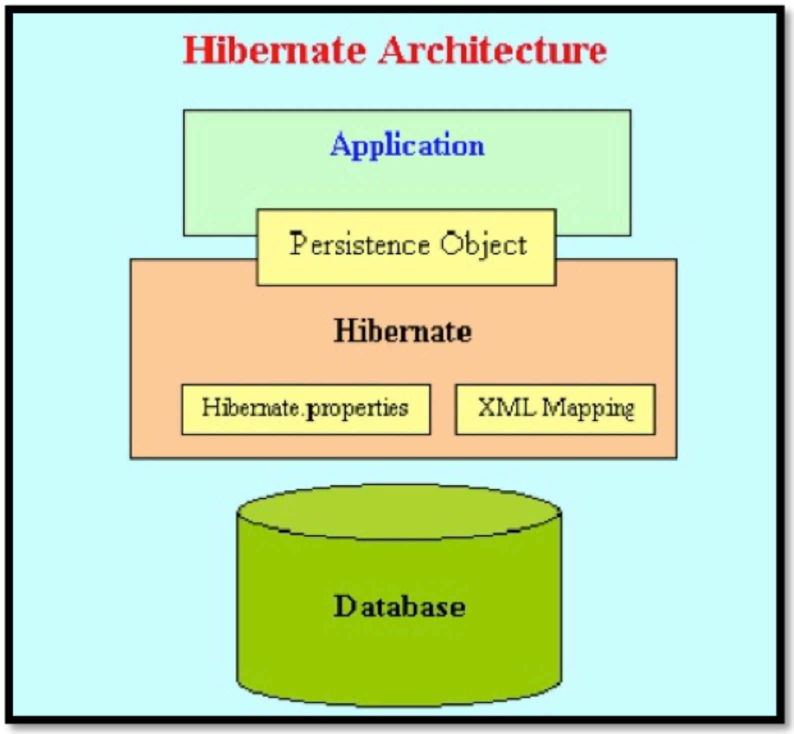


Figure 3: Hibernate Architecture

Hibernate has three main components:

Connection Management

Hibernate Connection management service provide efficient management of the database connections.

Transaction Management

Transaction management service provides the ability to the user to execute more than one database statements at a time.

Object Relational Mapping

Object relational mapping is technique of mapping the data representation from an object model to a relational data model. This part of hibernate is used to select, insert, update and delete the records form the underlying table.

In order to use Hibernate in Java, a Hibernate session is created. Objects can be loaded, deleted, created, and saved to the database using Hibernate commands. Also, Hibernate has its own query language, called HQL, which can be used along with SQL.

Apache Tomcat

A web container developed at the Apache Software Foundation. It provides an environment for Java code to run in cooperation with a web server. Tomcat includes its own HTTP server internally, so it also can be considered as a standalone web server. XML files are used to manage and configure Tomcat.

MySQL

An open source, multithreaded multi-user database, which is mostly used as a web database. Latest version of MySQL (MySQL 5.0) supports SQL 1992 standard, hence considered purely relational. As stated above, hibernate is used to map Hibernate to MySQL.

Input/Output Specifications

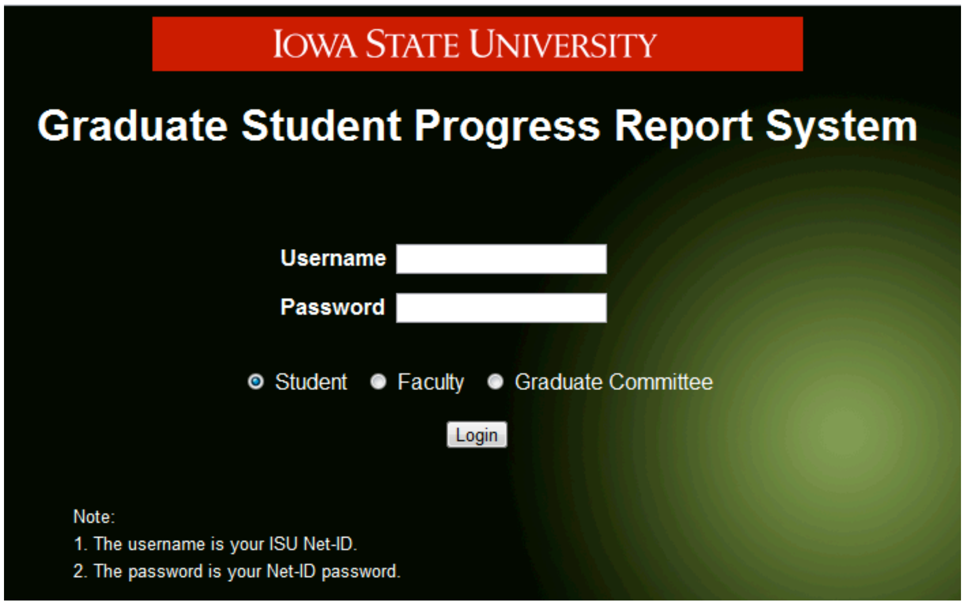
This section describes the input and output functionality of the system. The input/output methods are also given.

1. Input Specifications
   1.1. Form Input
   1.2. File Input
2. Output Specifications
   2.1. Display Output
      2.1.1. Information will be displayed to the user based on the page that the user is viewing. The necessary information for the page will be found within the database, formatted, and displayed to the user.
   2.2. Database Output
      2.2.1. Database output will be obtained by the system and used to generate the JSP pages.

User Interface Specifications

Login View

The existing GProgress login system, which consists of two separate logins, will be condensed into a single login that will require the user to enter his/her ISU Net-ID and password. Based on the entered information, the system will check to see whether the user exists in the system. If the user is found, then the system will proceed to check the ISU LDAP server to authenticate the Net-ID and password to see if there is a match. If the provided Net-ID and password match, the user will be redirected to their respective home page depending on their user type within the system.

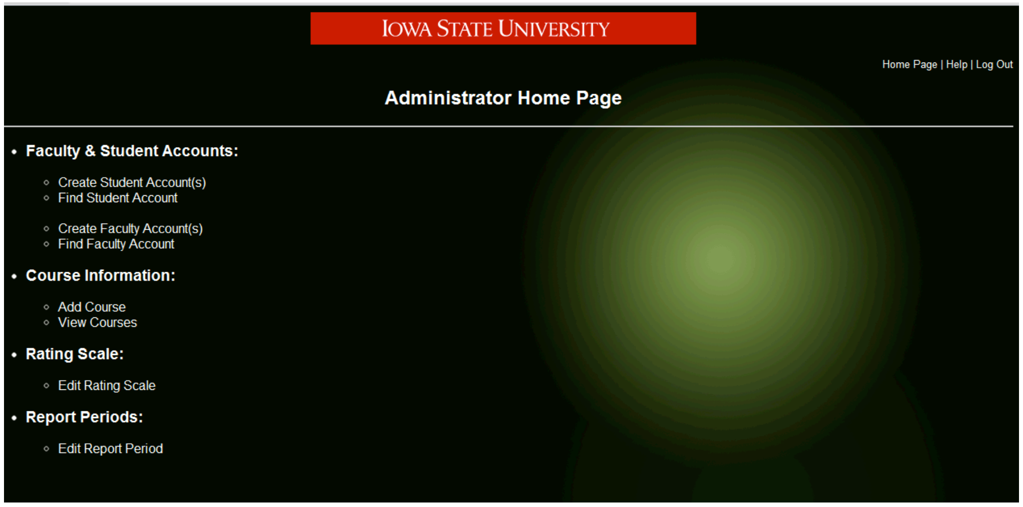


Note:
1. The username is your ISU Net-ID.
2. The password is your Net-ID password.

Figure 4: Login Page

Administrator View

The screenshot below shows an example screen that a faculty/staff administrator would see when they log in to the system. The system will check whether the user has students and/or is part of the Graduate Committee. If so, the administrator will also have access to faculty and/or graduate committee items.



IOWA STATE UNIVERSITY

Home Page | Help | Log Out

Administrator Home Page

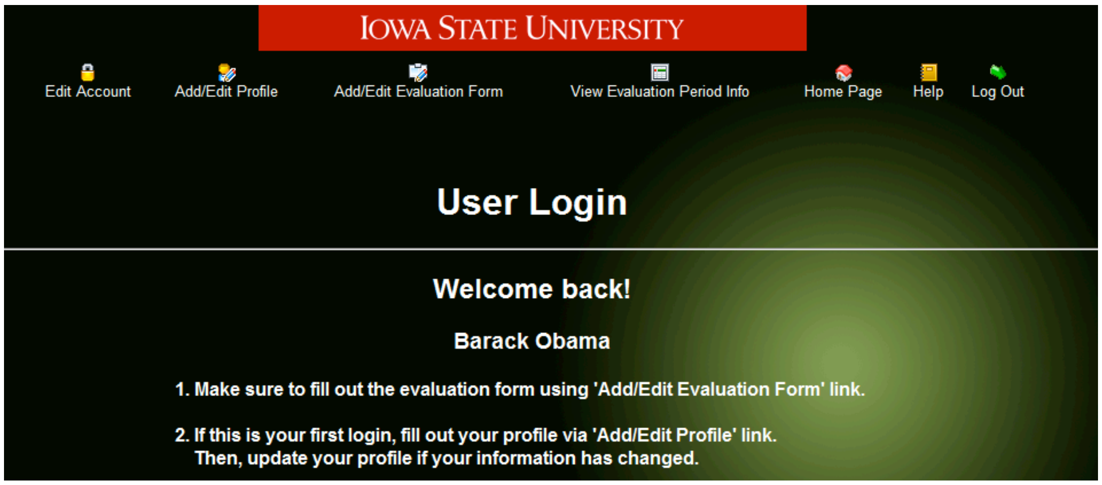
- **Faculty & Student Accounts:**
  - Create Student Account(s)
  - Find Student Account
  - Create Faculty Account(s)
  - Find Faculty Account
- **Course Information:**
  - Add Course
  - View Courses
- **Rating Scale:**
  - Edit Rating Scale
- **Report Periods:**
  - Edit Report Period

Figure 5: Administrator View

Student View

The screenshot below shows an example of the student home page. Students will have access to various menu items based on functionality outlined in the functional requirements.

Students will need to fill out an evaluation for each evaluation period. An example evaluation form is shown below.



IOWA STATE UNIVERSITY

Edit Account Add/Edit Profile Add/Edit Evaluation Form View Evaluation Period Info Home Page Help Log Out

User Login

Welcome back!

Barack Obama

1. Make sure to fill out the evaluation form using 'Add/Edit Evaluation Form' link.

2. If this is your first login, fill out your profile via 'Add/Edit Profile' link.
   Then, update your profile if your information has changed.

Figure 6: Student View

Faculty View

An example of a faculty view is shown below. The system will check whether a faculty member is also an administrator and/or part of the Graduate Committee. If so, the faculty member will also have access to administrator or graduate committee items. However, the faculty will need to specify whether or not they would like to login as a faculty or graduate committee member.

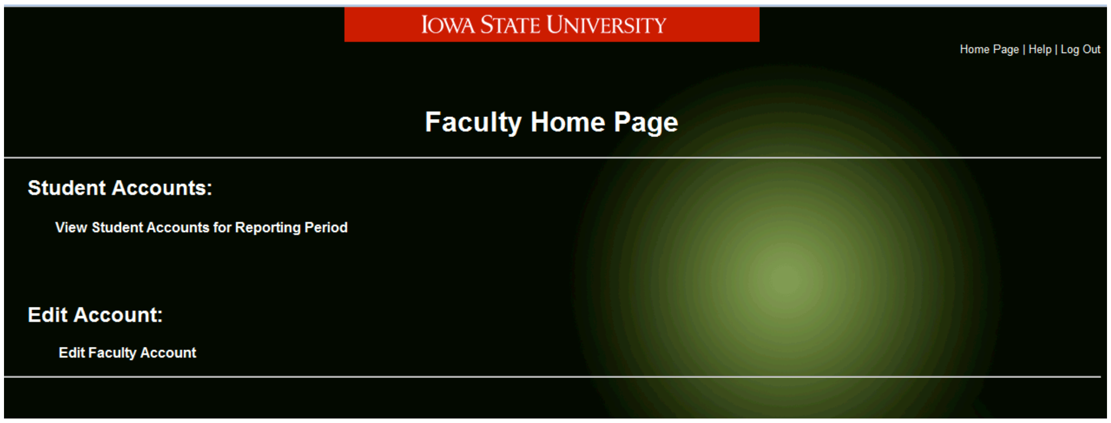
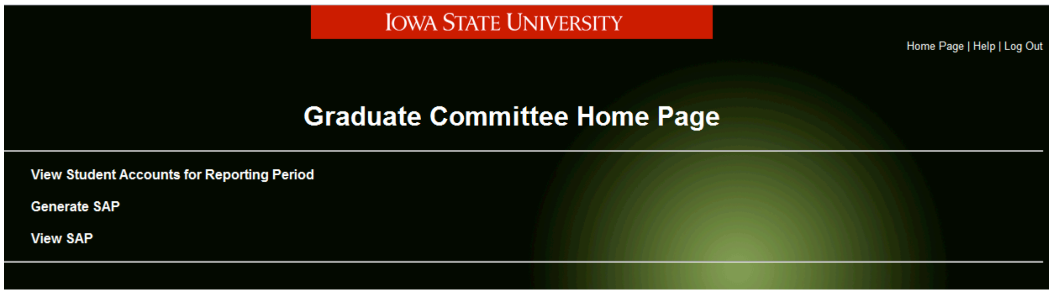


Figure 7: Faculty View

Graduate Committee View

The graduate committee will have three options upon login to the system. The options will be the following:

1. View Student Accounts for Reporting Period
   a. Graduate Committee member will select the desired reporting period and the student review and information along with the faculty rating will be shown.
2. Generate SAP (Satisfactory Academic Progress)
   a. User will upload a .csv file containing the desired format which will trigger the generation of the SAP report.
3. View SAP (Satisfactory Academic Progress)
   a. Committee will select a reporting period to view the SAP. The report will be displayed containing necessary fields and will highlight students that have an "unsatisfactory" grading.


IOWA STATE UNIVERSITY
Home Page | Help | Log Out
Graduate Committee Home Page
View Student Accounts for Reporting Period
Generate SAP
View SAP

Hardware/Software Specifications

1. Client Machine
    1.1. Supported Web Browser
        1.1.1. Internet Explorer 8
        1.1.2. Firefox 3
        1.1.3. Google Chrome 6
        1.1.4. Apple Safari 5
2. Server
    2.1. MySQL 5.1 or above
    2.2. Apache Tomcat 5.0 or above
    2.3. JRE 1.5.0 or above

Test Plan

In order to verify that the system we produce meets the requirements of the project, we will use three main forms of testing: unit testing, hands-on testing, and peer code review. Unit testing will be done using the JUnit4 test framework. Hands-on testing of the system will be done by checking all forms and screens on the produced website against any screenshots or descriptions provided in this document. Hands-on testing will also include manual verification that buttons and links behave as expected. To aid the hands-on testing, we may employ a GUI testing framework similar to Abbot and Costello if one can be found to work with html pages. Peer code review will take place throughout the development process to increase the overall quality of the code and eliminate as many bugs as possible.

Below is a list of items that should be tested using the above-described means. This list is not comprehensive, and should be expanded upon as needed during the development process.

1. Login/Logout
    1.1. Make sure login uses the engineering servers correctly
    1.2. Make sure once logged in, a user has the proper privileges
    1.3. Make sure logging out clears the user's information
2. Form Submission
    2.1. Ensure that forms contain the desired information
    2.2. Ensure that the forms are saved to the server correctly
    2.3. Ensure that the user can see the relevant forms
3. Email Notification
    3.1. Ensure that email notifications are sent when evaluation period begins
    3.2. Ensure that faculty are emailed when students submit their self-evaluation
    3.3. Ensure that the graduate committee is emailed when faculty have finished their evaluations
4. Security
    4.1. Test that private data requires the proper login/password to access
    4.2. Test against SQL injection
    4.3. General security concerns

5. Load Testing
    5.1. Test that the system can hold up to the expected load

Schedule

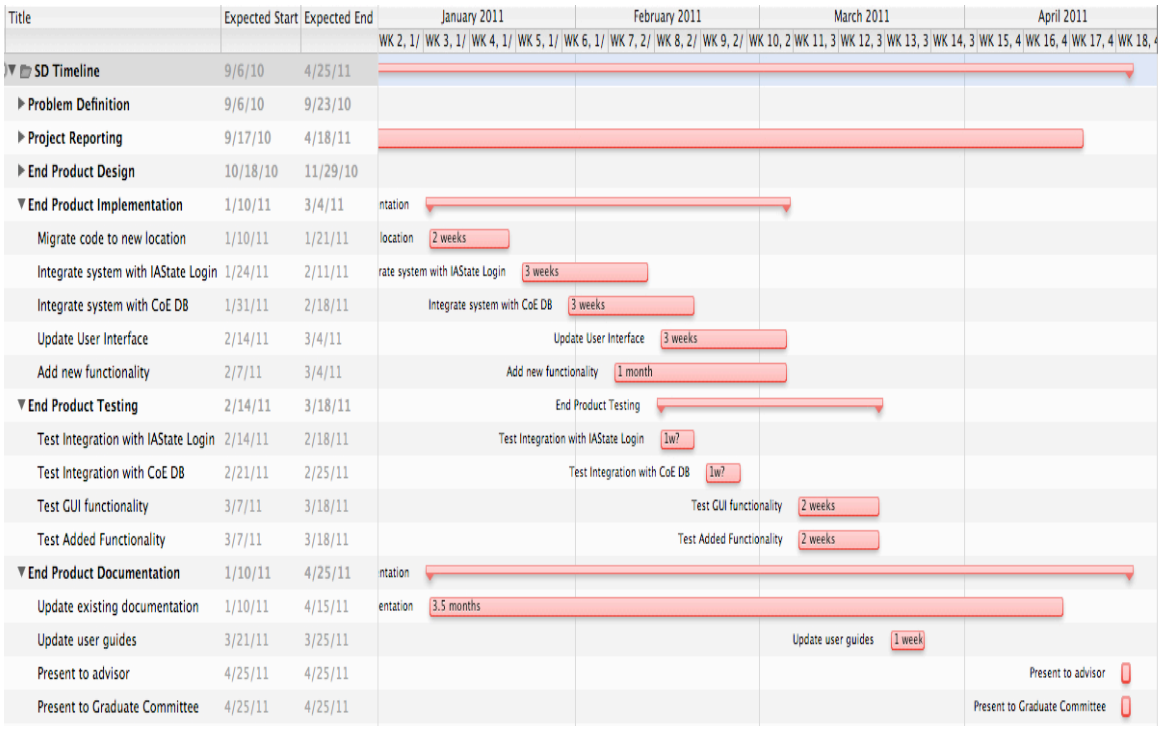
The following are the items that we will need to complete in the Spring 2011 in order to complete the project on time.

1. End Product Design
    1.1. Gain deeper understanding of existing code

2. End Product Implementation
    2.1. Migrate code to new location
    2.2. Integrate with IAState login system
    2.3. Integrate with CoE database
    2.4. Update user interface
    2.5. Add additional functionality

3. End Product Testing
    3.1. Test integration with IAState login system
    3.2. Test integration with CoE database
    3.3. Test user interface functionality
    3.4. Test added functionality

4. End Product Documentation & Demos
    4.1. Create modified version of existing documentation of the GProgress system to match implemented system
    4.2. Update system user guides for GProgress system to match implemented system
    4.3. Present end product to advisor & possibly graduate committee



Title	Expected Start	Expected End
SD Timeline	9/6/10	4/25/11
Problem Definition	9/6/10	9/23/10
Project Reporting	9/17/10	4/18/11
End Product Design	10/18/10	11/29/10
End Product Implementation	1/10/11	3/4/11
Migrate code to new location	1/10/11	1/21/11
Integrate system with IAState Login	1/24/11	2/11/11
Integrate system with CoE DB	1/31/11	2/18/11
Update User Interface	2/14/11	3/4/11
Add new functionality	2/7/11	3/4/11
End Product Testing	2/14/11	3/18/11
Test Integration with IAState Login	2/14/11	2/18/11
Test Integration with CoE DB	2/21/11	2/25/11
Test GUI functionality	3/7/11	3/18/11
Test Added Functionality	3/7/11	3/18/11
End Product Documentation	1/10/11	4/25/11
Update existing documentation	1/10/11	4/15/11
Update user guides	3/21/11	3/25/11
Present to advisor	4/25/11	4/25/11
Present to Graduate Committee	4/25/11	4/25/11

January 2011
February 2011
March 2011
April 2011
WK 2, 1/ WK 3, 1/ WK 4, 1/ WK 5, 1/ WK 6, 1/ WK 7, 2/ WK 8, 2/ WK 9, 2/ WK 10, 2 WK 11, 3 WK 12, 3 WK 13, 3 WK 14, 3 WK 15, 4 WK 16, 4 WK 17, 4 WK 18, 4
2 weeks
3 weeks
3 weeks
3 weeks
1 month
1w?
1w?
2 weeks
2 weeks
3.5 months
1 week

Figure 8: Schedule Spring 2011

Implementation

Form Additions & Updates

In order to customize the existing GProgress system to fit the ECpE department's needs, some changes and additions to existing forms and functionality were required. Updates to forms included adding new fields to existing student and faculty evaluation forms, updating forms to increase system maintainability, adding new functionality to the administrator view, and adding filtering abilities to the graduate committee view.

Form Updates

The following changes were made to system pages and forms:

1. Update all wording to "ECE related instead of "Computer Science"
2. PhD Student Self-Evaluation Questions & Logic
    a. Has student completed PhD qualifying exam? (Yes/No)
        i. If yes, specify date of completion.
        ii. If no, specify number of semesters until completion & explanation.
    b. Has student completed PhD preliminary exam? (Yes/No)
        i. If yes, specify date of completion.
        ii. If no, specify number of semesters until completion & explanation.

        c. Has student completed PhD defense? (Yes/No)
            i. If yes, specify date of completion.
            ii. If no, specify number of semesters until completion & explanation.
3. Form input filtering
    a. Current forms had no input filtering whatsoever; students could enter anything they wanted into the fields.
    b. Updated this to check inputs and make sure they match desired formats before allowing them to submit.
4. System maintainability
    a. Current forms had hard-coded dropdowns for years that students could choose from for various dates. These date ranges were not dynamically populated, so by year 2020 forms would no longer be useful.
    b. Updated all forms found with this functionality to be dynamically populated based on current year.
    c. Moved all CSS declarations into 3 files instead of having same declarations on 100+ forms. Now, if the look and feel of the system needs to be changed, it can be done quite easily.

Added Features

To make the system easier to use, reduce the amount of work required to get the system up and running, and make the system easier to maintain and update, some changes were required. These changes included creating imports for student and faculty account creation to automate the process, updating the login process for the graduate committee, and importing course lists from ISU website.

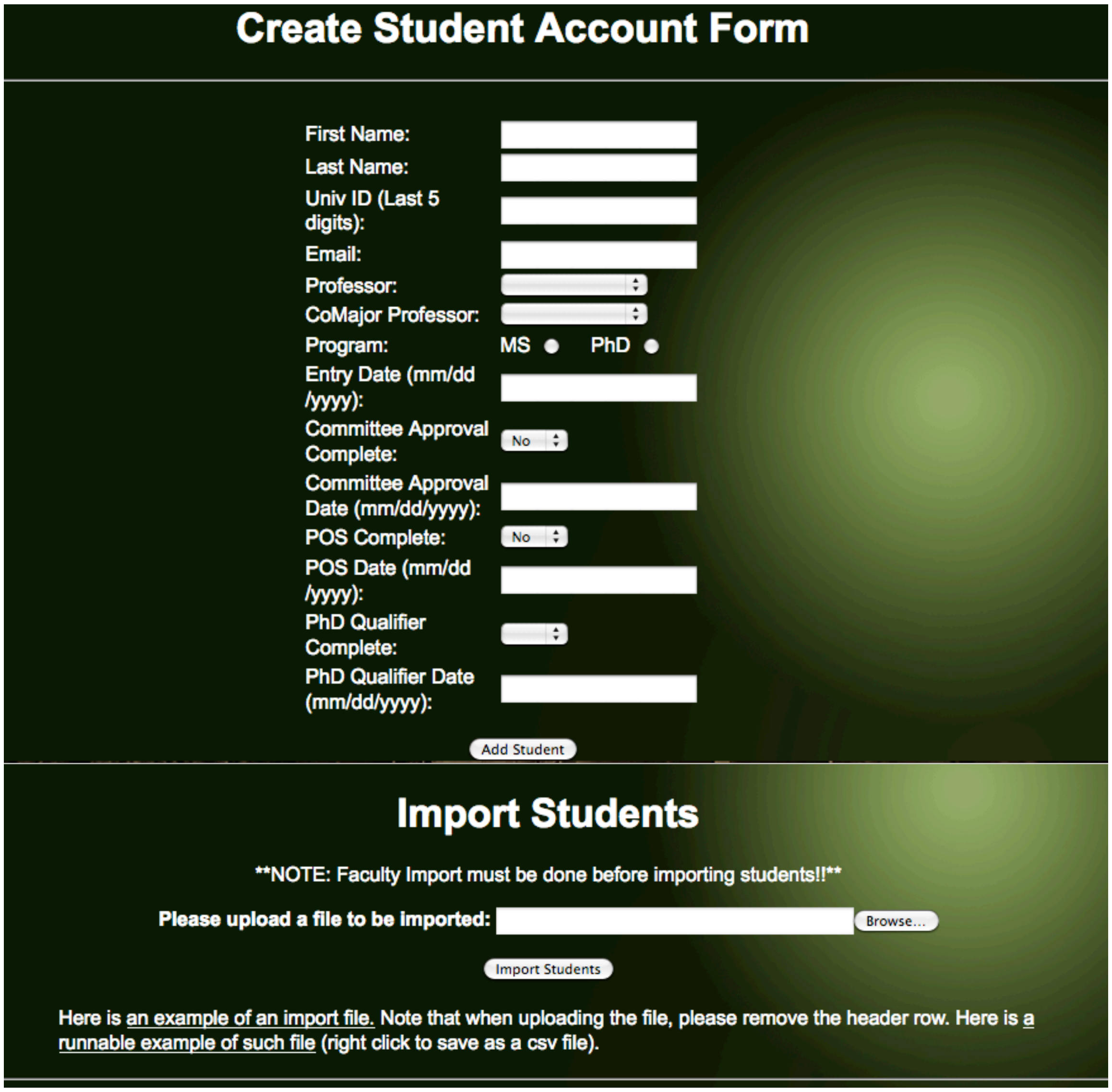
Student & Faculty Imports

To make the student and faculty account creation easier for the system administrator, we decided to create student and faculty import pages. The existing system only allowed the system administrator to manually add each student and faculty member one at a time. This process would be very time consuming since the system will have about 400 users with new users being added each year.

The new functionality allows the administrator to easily take a correctly formatted CSV file and import the information into the system in one step. The system will automatically create the user accounts and populate them with the input data, reducing the amount of input needed from the student and ensuring accurate data.

Once the file has been imported, the system will display a list of all users that were successfully imported to allow the administrator to easily see the results of the import. Also, if the administrator only needs to import a few students, they may also use the manual import and provide the same data contained in the CSV file.

The details of the actual integration with the College of Engineering database will be discussed in a later section of this document. Below are pictures of the import functionality.



Create Student Account Form

First Name:	
Last Name:	
Univ ID (Last 5 digits):	
Email:	
Professor:	
CoMajor Professor:	
Program:	MS ● PhD ●
Entry Date (mm/dd/yyyy):	
Committee Approval Complete:	No
Committee Approval Date (mm/dd/yyyy):	
POS Complete:	No
POS Date (mm/dd/yyyy):	
PhD Qualifier Complete:	
PhD Qualifier Date (mm/dd/yyyy):	

Import Students

NOTE: Faculty Import must be done before importing students!!

Please upload a file to be imported:

Here is an example of an import file. Note that when uploading the file, please remove the header row. Here is a runnable example of such file (right click to save as a csv file).

Figure 9: Updated Create Student Account page allowing manual addition & import

Login Process

The current system also had a separate username and password combination that would be used by all faculty members on the graduate committee. In order to make the system more secure and conform to our new Net-ID login integration, this had to be changed.

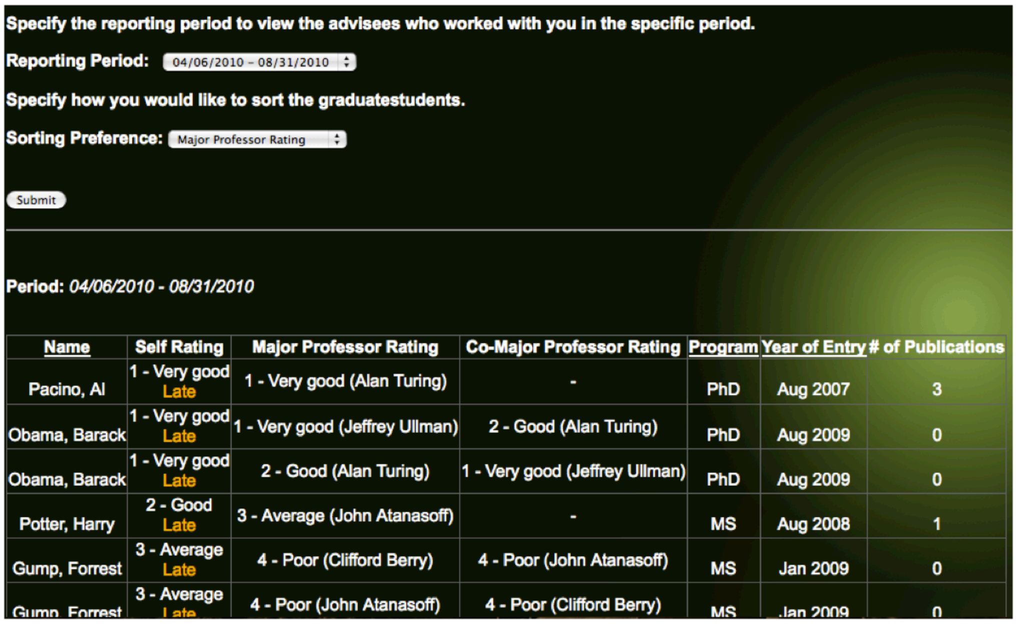
The solution to this problem was to add a new field to the faculty table in the database called “isGradCommittee”. This field can be set to yes or no for each faculty member in the system. The administrator can update members if members change by searching for the faculty to update, and updating this faculty member's graduate committee status.

Now instead of requiring graduate committee members to use a separate username and password to login as a graduate committee member, they can continue to use their Net-ID and password to login for both the faculty and graduate committee views. Restricting user login to Net-IDs for all users will help maintain confidentiality of the data and keep outsiders from being able to crack passwords and gain unauthorized access to the system.

Graduate Committee Evaluation Filters

The graduate committee expressed that they would like a way to sort the graduate students by various attributes:

1. Major Professor Rating
    a. Sorts the graduate students according to their major professor rating. This allows the graduate committee to look at the best and the most in need of improvement.
2. Faculty Recommendation
    a. Shows the graduate students that a faculty member has indicated may deserve recognition for their achievements.
3. Faculty Request Committee Intervention
    a. Shows the graduate students that a faculty member would like the graduate committee to address as a potential problem.
4. Over-aged Courses
    a. Shows the students that have been enrolled for more than five years.
5. Program of Study Overdue
    a. Shows the students that have not completed their POS on schedule.



Specify the reporting period to view the advisees who worked with you in the specific period.

Reporting Period: 04/06/2010 - 08/31/2010

Specify how you would like to sort the graduatestudents.

Sorting Preference: Major Professor Rating

Period: *04/06/2010 - 08/31/2010*

Name	Self Rating	Major Professor Rating	Co-Major Professor Rating	Program	Year of Entry	# of Publications
Pacino, Al	1 - Very good Late	1 - Very good (Alan Turing)	-	PhD	Aug 2007	3
Obama, Barack	1 - Very good Late	1 - Very good (Jeffrey Ullman)	2 - Good (Alan Turing)	PhD	Aug 2009	0
Obama, Barack	1 - Very good Late	2 - Good (Alan Turing)	1 - Very good (Jeffrey Ullman)	PhD	Aug 2009	0
Potter, Harry	2 - Good Late	3 - Average (John Atanasoff)	-	MS	Aug 2008	1
Gump, Forrest	3 - Average Late	4 - Poor (Clifford Berry)	4 - Poor (John Atanasoff)	MS	Jan 2009	0
Gump, Forrest	3 - Average Late	4 - Poor (John Atanasoff)	4 - Poor (Clifford Berry)	MS	Jan 2009	0

Figure 10: Graduate Committee Filtering

Future Graduate Committee Evaluation Additions

The distinction between direct-entry PhD and after MS PhD has not been implemented. Future milestones to consider include: prelims and recommendation of committee appointment.

The graduate committee also expressed that they would like to see statistical information included in the future. These statistical additions would include things like percentages of students who complete milestones in a certain time period, percentages of students who complete their degree within a certain time period, etc. to get an overall view of the department's effectiveness in these areas.

Course Import

GProgress had a manual import for courses. We decided to add an automated import for courses as well. You can see a screen shot below:

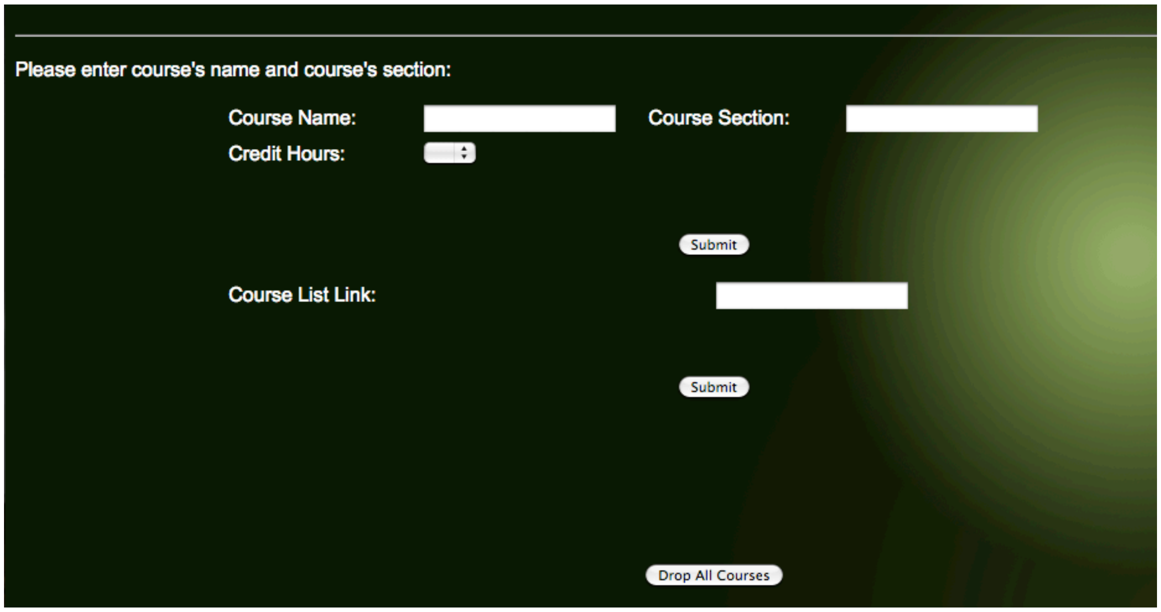


Figure 11: Updated Course Add Page with Import Functionality

The rational for this approach is that students are asked to fill out the coursework they have completed during each evaluation period. For convenience, we would like to have a drop-down list of courses that were offered. This data would be very tedious to enter manually. Fortunately, it is available on the ISU class listing. By scraping this data, we are able to pragmatically obtain the desired course lists, complete with descriptions and credit amounts. This method works for any data returned by the ISU class list search, so results are not limited to one particular department.

We implemented the course scraping approach by writing a parser to scrape out the data desired, given a valid URL to an ISU class list search. This data is then dumped into the database, and made available to the drop-down list that students select from.

System Integrations

College of Engineering Database Integration

When starting the implementation process for this project, we discovered that the College of Engineering already has a graduate database that tracks certain information about students and faculty. In order to reduce the required work to get all users into the system, it was important to utilize the existing database. The existing database will also allow us to pull more information about students directly from the database instead of relying on students to input accurate information. However, as students we were not allowed access to the database and data directly due to security and confidentiality reasons.

The current system required the administrator to manually add students and faculty one by one. For a system that may have up to 400 users, this method would not be realistic. As a solution, we agreed to just do an import from the existing system to our database. To do this, a CSV format was developed for both students and faculty imports. Then a file select was added to the "Create Student Account" and "Create Faculty Account" pages that the system administrator has access to.

Each year, a new CSV export containing the desired information will be pulled from the College of Engineering graduate database and used to populate our own database. The system administrator will then import the produced CSV each year. The College of Engineering hopes to fully integrate the two databases and systems in the future, but due to time constraints and difficulty, the current solution was the most practical.

Net-ID Login Integration

The system we inherited from the Computer Science department used local usernames / passwords for its authentication system. This is inconvenient in several ways.

First, we need to track the user's password. This adds an extra security concern as we then have to worry about securely storing the password locally. Local storage would require that we obfuscate the password in some way, probably by hashing, to keep admins with database access from viewing user's passwords.

Second, the user has to track an additional username / password. In the old system, users had to remember an additional set of login credentials. Since this system is designed to be used only periodically, users would likely forget these credentials by the next evaluation period. This is particularly a concern for any user who prefers to change their password periodically for security reasons. With the Net-ID system, the user's password is guaranteed to be up to date with what they use for most other university services.

Finally, using the Net-ID system makes the application feel more integrated with ISU and its other systems. Web-CT and Cymail both use the Net-ID for authentication. Using the Net-ID login with our system aids in the perception that the graduate evaluation process is part of ISU standard procedure.

To integrate the Graduate Evaluation system with the Net-ID system, we use Kerberos. Kerberos is an authentication protocol that the university supports for validating Net-ID login credentials. Java provides an implementation of Kerberos. Any time authentication is required, we use Java's Kerberos implementation to validate the credentials. In this way, we avoid ever storing the password locally.

Testing

White Box Testing

Since our team has access to all of the system code, we were able to easily test all functionality and interactions between the user's browser and the database. The system implements a log that tracks all actions completed from the browser instructions to the actual "behind-the-scenes" actions that take place. Any problems processing the information will be seen in the log file giving detailed descriptions of the problem so it can be fixed.

All pre-existing functionality as well as added functionality was tested to ensure proper functionality for all users and use cases. Our team also did some cross-browser and cross-platform testing on the system since our team members all have different platforms and browsers. This testing included Mac OS, Linux, and Windows running Firefox, Safari, and Internet Explorer among others.

Functionality was tested by inputting information for various users into the system, testing it was properly stored to the database, gave proper notice to the user that information had been updated and that the newly stored information was being displayed back to the user on subsequent views.

Testing was also done to make sure students could only see their own information and information related to their faculty member's reviews. The faculty member view had to be tested to ensure faculty could only access information for their own students and complete reviews for their own students. Graduate committee members in the updated login system had to be set by the administrator so testing had to be done to ensure only faculty members set by the administrator could gain access to graduate committee functions.

Stability Testing

Since the system was obtained from the Computer Science department, we inherited a tested, functioning system to begin with. This system had already been in use in the computer science department for a few years so the functionality had already been tested thoroughly from their testing before deployment and through normal use by Computer Science department students and staff.

Usability Testing

For usability testing, two demos were given to the graduate committee and faculty that would be using the system. One demo was given during the first semester of this project to get an idea of what features needed updates and any new functionality that the committee wanted. At the time, the committee did not have any input on the functionality so we based our decisions on our advisor's input.

Towards the end of our implementation phase, another presentation was given to the graduate committee and faculty to demo the new and updated functionality. The graduate committee had more input this time on what they would like changed and what they wanted added to our implementation. Most changes were minor additions or updates that were easily completed. Overall they appeared to be happy with the progress and functionality that the system provided. The one addition to functionality that they wanted for the system was statistical information on student progress, which is discussed in the implementation section of this document.

Security Testing

Security testing includes testing the new Net-ID login functionality as well as that users only have access to the proper information and pages once they have logged in to the system.

The Net-ID testing involved using our own team member Net-IDs to test with since we cannot get any test data for this purpose. All team members were able to successfully login to the system using the integrated Net-ID login from various system setups (Mac, Linux, Windows) to ensure all users can successfully login to the system from different systems and browsers.

User role testing was done for each user class: administrator, student, faculty and graduate committee. Through this testing we confirmed that the user sessions were working properly and users could only access pages that their user class had access to. This testing also was done to ensure the confidentiality of student information and their reviews. Although the information contained in the system is not anything critical to a student's files, leaked information would still violate a student's right to have their information be confidential.

As an additional security measure, the system might be given to a CprE 531 class in the future to do another thorough testing of all security aspects of the system. This will confirm the security of the system before it is put in use by the department.

Integration Testing

Net-ID Login

To make sure that the Net-ID Login System worked with the existing system, we tested that it authenticates correctly against the server. We also tested the authentication privileges of the user, making sure that the users only gain access to data they have privileges to view after they authenticate correctly. Additionally, we tested to make sure that the session maintains the logged in status the same as it did in the previous system.

Load Testing

To make sure the system could handle the expected load, we looked at the existing system data that we were given. The existing system contained around 400 users between faculty and students. Since our given user base will be approximately 400 concurrent users and the existing system had been in use with about the same amount of users, we could reasonably assume that the system could handle at least that amount or more. Our system has also been given a server with the College of Engineering for hosting purposes. This server has been set up by the IT department specifically for our system and will be able to handle the desired amount of users and traffic.

User Manual

System Setup for Local Machine

1. Download code to computer
2. Start MySQL server
3. Run csgradstudenteval.sql file to load tables and data
4. Create Apache Tomcat server on your machine
   a. Add gradEvalSys to to the right location to run on the Tomcat server
5. Use browser to navigate to login pages for admin & user
   a. http://localhost:8080/gradEvalSys/forms/userLogin.jsp
   b. http://localhost:8080/gradEvalSys/forms/adminLogin.jsp
6. Login using given account information if using administrator login, otherwise login using your own ISU email and Net-ID password.

Overall Process

1. Administrator will need to login and create a new Evaluation Period for the current year. Administrator will need information on the dates for the period, dates for students to complete their self-evaluation and dates for faculty to complete their evaluations.
2. Administrator will update course list by going to "Edit Courses".
3. Administrator should then add or import any faculty accounts needed for the system by going to the "Create Faculty Account" page.
4. Administrator will then need to add or import any student accounts needed for the system by going to the "Create Student Account" page.

How to Update Student Imported Fields

1. If field(s) to add require changes to underlying database start here, else skip to step 2.
    a. Update "Web Content/WEB-INF/classes/database/entitiesConfig/Background.hbm.xml"
        i. Add new fields to this xml
        ii. Field names must match field names from the database
    b. Update associated java file "Web Content/WEB-INF/classes/database/entities/Background.java"
        i. Add new fields
        ii. Create getter & setter for each field
2. Update "Web Content/WEB-INF/classes/objects/RowOfStudentTable.java"
    a. Add new private String variableName for the field(s) to add
        i. All should be String type for this java file
    b. Update constructors with new variables
    c. Create getter & setter for new field(s)
3. Update "Web Content/WEB-INF/classes/coreservlets/CreateStudentFormBean.java"
    a. Update NUM_COL and NUM_INPUT_COL
        i. Add # of new fields to existing value
    b. Add private String variableName for the field(s) to add
    c. Create getter & setter for new field(s)
    d. Go through file and find where student Background or Student info is being set.
        i. Use setter methods to set values equal to values that are imported
4. Update "Web Content/forms/Create_New_Student.jsp"
    a. Add new field(s) to the manual create
        i. Property value must match the variableName chosen in step 2b)
    b. Update sample files that are linked to to show new examples of input files to use
5. Update "Web Content/WEB-INF/results/Create_New_Student.jsp"
    a. This page shows the results
        i. Add table spots to display for each field
6. Rebuild project to update .class files

491 Team Review

Reviewers: Charles Ristau, Mohd Azmy, Yiksen Tan, Theron Worthington

Team Reviewed: May 2011 Group 16
Project: Graduate Student Review System

Members:
Ashley Montebello-Cpre
Katie Githens-SE
Wayne Rowcliffe-SE

What is the high-level objective of the project? (2-4 sentences)

The objective of the project is to create a review system for ECpE department which will allow students and faculty to fill our and view performance evaluations. It will also allow the graduate review committee to easily review graduate student performance.

What are the key functional requirements of the system? (2-4 requirements)

There are four main functional requirements. The first is to allow students the ability to update profile and enter progress/self-review for each review perod. The second functional requirement is to give faculty the ability to review their students, recommend students for awards, and to view necessary student information. The graduate committee will have the ability to view student progress based on milestones, ratings, and award recommendation, etc. Lastly, the administrator will be able to update students, faculty, review periods, and courses.

What has been actually implemented? (1-2 paragraphs)

Provide a brief description of the functioning of the implemented system, the hardware, software, and other equipments used in the system.

So far, the team has made many changes to existing code, including changing hard-coded selects to dynamic(to get rid of extra code and repetition). All of the forms/code have been implemented and tested. The team has permission to use the college of engineering graduate database.

The team is still going through the code to find more things to improve. The project uses ISU NetID Login and Kerberos authentication to implement the login part of the system. A csv is created to communicate with their system. The project will be hosted by the College of Engineering after the project is complete.

How to setup the system? (~1 page in bulleted list form)

Provide a step-by-step instruction on how to setup/demo and test the operational system
???????????

Test results observed? (one paragraph)

The test results passed with dummy data. The real data can only be observed when the system is actually implemented because of privacy issues with login IDs. However, the dummy data is very similar to the real data, so the real data should easily work.

Critique of the project

Provide at least one strength and at least one weakness of the implemented system (2-4bullets)

- A strength is that it seems to work at this point, given the test demonstration our group was given.
- One weakness is that at this point, there are continual updates which makes testing the code a little difficult. Only some netids will start the system, but this is an update issue.

Does the implementation meet the specification? In either case, provide a brief discussion.
(2-3 sentences)

Yes, many of the specifications were created by the project team, and all of those specs have been met including the ones given. The login and review system seems to be fully operational and gives the users their intended abilities.

Suggestions
The group has made excellent progress, no large changes need to be made. The choice of Java as a platform was a good because it can run on many types of machines.

References

Graduate Student Performance Evaluation System (GProgress) Documentation
Department of Computer Science – Mohammed Alabsi, Srikanth Krithivasan, Melissa Yahya, Lexin Liu