CoursesHelper

by Kate Ho (siho)

Feb 6, 03

Description:

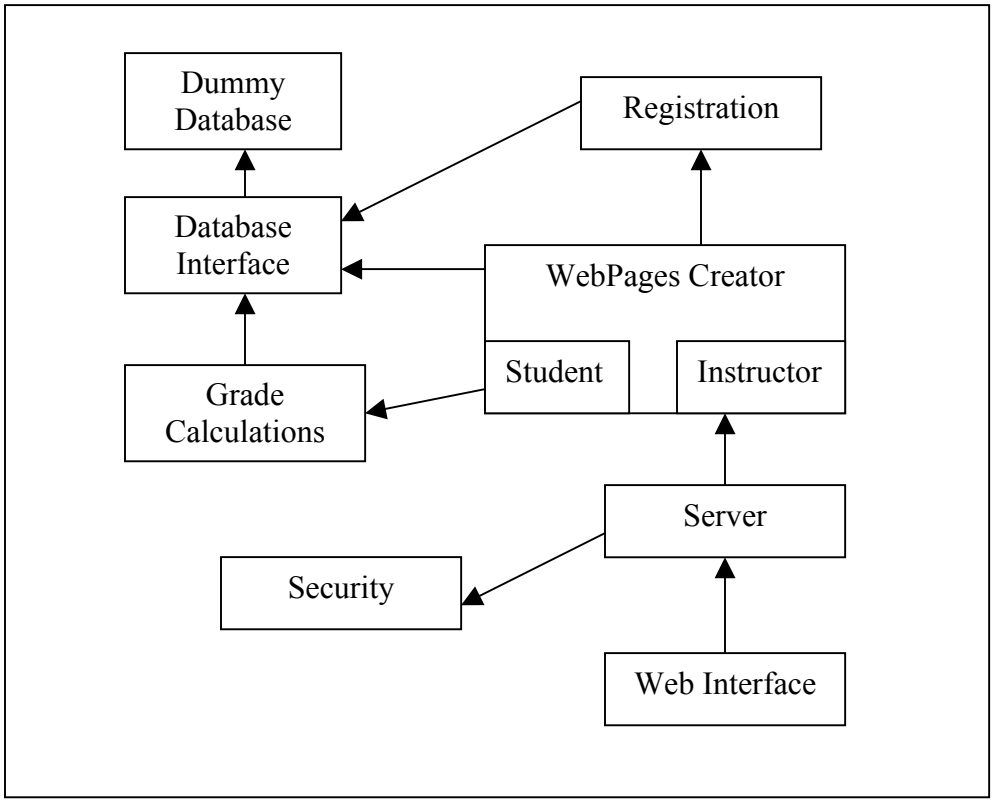
Although Internet technology has become everyday life these days, Brown students could not enjoy the convenience of registering and managing their courses online yet. Moreover, grades are often one of the most concerned subjects for students, and so additional tools for estimating course performance could be valuable.

This project will implement a secure web interface for course registration and grades viewing. It would also provide tools for professors and teachers to help determine the final grade of a student. The goal is to reduce workload in Registrar's Office, and provide a more convenient way for professors and students to manage their courses.

Target Users:

Target users are mainly Brown University Registrar's Office, professors, and students. However, it should also be general enough to use in other universities or schools.

System Model Diagram:



Annotations and Descriptions:

Dummy Database: A database storing student and course information.
Student Information: Student ID, name, semester standing, concentrations (maximum 3), courses registered and taken in each semester, final grades
Course Information: Department, course number, section, instructor name, meeting time, exam group, location, description, student IDs and names of students enrolled, student grades (final and sub-scores for each assignment/quiz/exam), information about sub-scores. There could be 20 sub-scores maximum, each link with weight in final grade (ie, grade distribution), passing score, and a boolean indicating whether passing this sub-score is required to pass the whole course.

Data Interface: An interface for extracting information from database, so any college or university could easily use this software without changing its current database by overriding methods here.

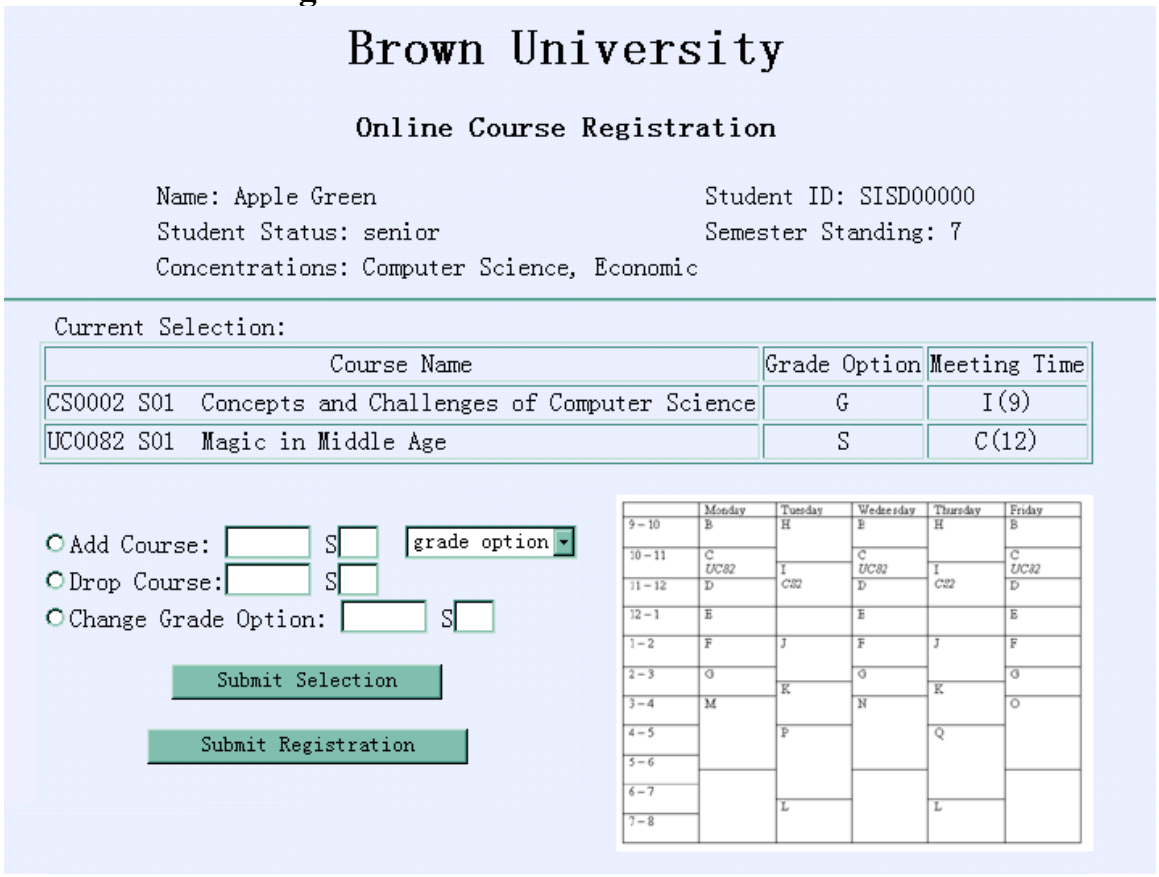
Grade Calculations: Calculate anything related to grade. It could be further divided into two parts:
Course grades: This is a grading tool for instructors. Functions: calculate final scores of students based on sub-scores; calculate mean, standard deviation, medium, highest score, number of students receiving each grade, for each sub-score, and final score; draw graphs of score distributions for any score set (ie, sub-scores or final scores, in whole class), with mean and medium line drawn; determine student grades automatically by score distribution (default), or by inputted score boundaries; predict final grade of each student; submit final grades (ie, editing student and course database for final grades).
GPA: This is a GPA calculation tool for students. Functions: calculate total GPA, concentration GPA, GPA of last 4 semesters.

Registration: A tool for course registration. Functions: add course (input: department, course number, section, grade option); drop course; change grade option; produce weekly schedule base on current selection; submit registration (ie, editing database).

WebPages Creator: Extract information and create webpages. It can be divided into two parts: student and instructor. These webpages will be used to interact with users. This module will also be responsible for processing input data.

Server:	Establish connections with web browsers, create threads, and control traffic. Transfer data between web browsers and WebPages Creator. It acts as the gate between users and the whole system, and should be secure and robust. Data transfer between server and web browsers should be encrypted (with kerberos tickets), and should be decrypted before sending to WebPages Creator. It should be able to support at least 500 users at the same time, and should not crash even when the number of users exceeded maximum limit.
Security:	Authenticate users using Kerberos, assists server to encrypt and decrypt messages, and validates user identity during the whole session.
Web Interface:	Web browsers of users. This is NOT part of this project.

User Interface Diagrams:



This is the online course registration interface. It prints out all the basic information about the student, current course selection, and a weekly schedule corresponding to the current course selection. User can add a course, drop a course, or change grade option of a course by selecting the appropriate radio button and filling in the textboxes. User completes registration by clicking “Submit Registration” button, in which case a confirmation window will pop-up.

Brown University

View Grades and Registration

Name: Apple Green Student ID: SISD00000
Student Status: senior Semester Standing: 7
Concentrations: Computer Science, Economic
GPAs:
Total: 3.3 Concentration: 3.5 Last Two Years: 3.1

Grades:

Fall 2000:		Spring 2001:	
CS0015	A	CS0016	B
MA0010	B	MA0017	A
EN0003	S*	MU0040	C
EN0009	S	MU0055	B
Fall 2001:		**Spring 2002:**	
CS0031	A	CS0032	A
MA0018	B	MA0052	A
EL0011	S*	EC0011	C
AB0001	S	CI0030	B

Registered:

Course Name		Grade Option	predict grade	View Sub-Scores
CS0002 S01	Concepts and Challenges of Computer Science	G	predict	view
UC0082 S01	Magic in Middle Age	S	S	view

Sub-score: UC0082 S01 Magic in Middle Age

Paper1: 10%	Paper2: 25%	Final Paper: 40%	Final Exam: 25%
65/100	75/100		

This is the grade-viewing interface. Students can checks their grades here, as well as calculated GPAs, and the courses they have registered for. User can predict his/her grade in a class by clicking “predict” button, and view the sub-scores of a class by clicking on “view” button.

Brown University

Course Management

Course Number: UC0082 S01 Semester: Spring 2000
Course Name: Magic in Middle Age
Instructor: ABC DEF Number of Students enrolled: 5
Meeting Time: C Exam Group: 12
Location: Sayles 101

Add New Column: Weight: % add

	Paper1: 10%	Paper2: 25%	Final Paper: 40%	Final Exam: 25%	Final Score	Final Grade
John White	50/100	65/80				
Jane Green	73/100	72/80				
Jone Black	45/100	/80				
Apple Green	82/100	/80				
Dave Blue	90/100	/80				
Mean	68/100					
S.D.	15.3					
Medium	73					
Highest	90					
A:	80-100: 2					
B:	65-80: 2					
C:	45-65: 2					
NC:	0-45: 0					
	Cal. Statistics	Cal. Statistics	Cal. Statistics	Cal. Statistics	Cal. Statistics	
	Est. Boundaries	Est. Boundaries	Est. Boundaries	Est. Boundaries	Est. Boundaries	
	Draw Graph	Draw Graph	Draw Graph	Draw Graph	Draw Graph	
req. to pass?	N	N	Y	Y		

Submit Changes

This is the course management interface. It contains all the grading tools for instructors. It prints out all the basic information about a class, and instructors can input students' sub-scores here. Instructors can change the value of anything (blue) in the screen by clicking on it. Instructor can add new sub-score column, change weights of sub-scores, input and change grades, input and change grade boundaries, input final grade, and view all statistics here. At the bottom of each column, there are 3 buttons: calculate statistics – clicking it will start the statistics calculations, and print result on screen; estimate boundaries; and draw graph – clicking it will cause a new window with score distribution graph to pop up. If "require to pass" is "Y", a student is required to pass (ie, not NC) in that sub-score to pass the whole course. Clicking on "Submit Changes" will submit the change to database.

UI Descriptions:

The user interface will mainly consist of 5 windows, and a login screen. Instructors and students login to the system via the login screen, and they'll get a ticket for authentication during the whole session. After logging in, students will enter into a screen with their basic information, and from there he/she can choose whether to register courses online or view his/her grades. For instructors, they will enter into a screen with their basic information and a list of courses that they teach, and then he/she can choose which course he wants to manage, input grades to, and get score statistics from.

Non-functional Requirements

Performance:

Within Brown network, users should be able to get responses within 20 seconds for any operations during non-busy time (not registration period, not right after finals), and within 2 minutes during busy time.

Testing:

- All components should be tested extensively before integrating.
- For database, add, delete, and update should be functional for all data items.
- For grade calculations, statistics and graphs should be generated correctly, comparing to existing or hand-calculated results; grade estimation should be reasonable, again comparing with existing data, and by teachers' judgments.
- Registration should communicate with database correctly (ie, extracting and changing the right data), and produce a correct weekly schedule.
- WebPages creator should generate the correct interface, comparing with the pictures lists above. It should prints out correct information, and sends the correct data to other components.
- Server should be able to create and manage threads, and handle inputs from multiple web clients at the same time. It should transfer data correctly, and block unauthorized accesses.
- Security should be able to authenticate a user correctly, and assist server to encrypt and decrypt message correctly.

The whole system can be tested extensively by writing 2 programs (one simulate student, another simulate instructor) that input lots of random commands into the server at the same time, with the help of threads. It can then be tested by group members randomly doing various operations through the web.

Reliability:

The system should be able to deal with at least 500 users at one time, and should not crash even when users exceed that amount. It should be able to deal with cases where a student has logged in multiple times and making changes via more than one computer. All data in database should be consistent.

Ease of use:

Users should be able to do any operations within 5 clicks.

Portability:

Users should be able to use this system via web browsers anywhere with Internet connection. Any university, college, or school should be able to use this software by simply changing the database interface, and maybe also minor changes in security and server.

Documentation:

Detail documentations for users should be provided online. Documentations should also be provided for maintainers, on how database interface could be changed so existing database can be used, and how authentication system could be changed if any other method is preferred.

Dependencies:

This project will be depending on Kerberos. It also requires user to have a web browser installed.

Updated Requirements:

High priority:

* Authentication
* Courses registration via Internet
* Grades viewing via Internet
* Grades input by professors via Internet

Mid priority:

* Help professors to determine students' final grade
  - allow professors to set grade distribution for each course
  - calculate students' final grade by summing up the sub-grades/scores
  - allow professors to set requirement that a student must pass certain homework/exams to pass the course
  - allow professors to alter or input final grades manually
  - estimate grades
  - provide statistics for each sub-score and final grade:
    mean, standard deviation, medium
    creating graphs for each score set.
* Final grades could be submitted by professors via Internet, and immediately available for students viewing.
* GPAs (general, concentration, last two years) will be calculated and shown with students' grades.

Low priority:

* allow students to view their sub-scores in each course.
* generate a weekly class timetable
* predict final grades based on existing data

OPTIONAL FEATURES:

* provide professors and students with student's learning curve in each course, based on inputted sub-scores.
* allow students to view courses statistics, such as number of students enrolling in class.
* allow students to find out who else is in the same class, access their classmates basic info if they chose to release them, all via internet.
* allow students to changes courses online
* attach with each student's name with his/her photo
* let students to do course evaluations online, to save papers.
* gather and calculate evaluation statistics for all courses, and allow potential students to view them online.
* automatically generate course webpages with basic info.

Risky Parts:

- Authentication and security might be hard to implement.
- Supporting a large amount of users at a single time using one server might be extremely inefficient
- It requires a lot of maintenance.