A Database-backed Education System
for Collaborative Teaching and Learning

by

Aileen Tang

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 12, 2000

© 2000 Aileen Tang. All Rights Reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author

Department of Electrical Engineering and Computer Science

May 12, 2000

Certified by

Harold Abelson
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Departmental Committee on Graduate Students
A Database-backed Education System
for Collaborative Teaching and Learning

by

Aileen Tang

Submitted to the Department of
Electrical Engineering and Computer Science

May 12, 2000

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The System for Collaborative Learning and Administration of Students and Staff (CLASS) is a database-backed Web service designed for the purposes of teaching and learning. It creates online communities of people and provides the appropriate tools for them to perform their roles, whether they are professors, students, teaching assistants, course administrators, or class secretaries. Although many education systems already exist on the Web today, our system distinguishes itself by providing a user-centric and open source solution that is highly extensible and customizable to suit any educational purpose, be it academic, corporate, or personal. Our primary goal is to leverage the system’s integrated data model so information can be exchanged across classes and departments. The user interface design is intuitive and pushes information that is relevant to the user in a timely and effective manner. The education system’s first release includes a personalized portal page, a searchable file archive, and tools for the management and administration of departments and classes. Several schools have already planned to begin using our system to support their academic programs in Fall 2000.

Thesis Supervisor. Harold Abelson
Title: Professor of Computer Science and Engineering
I would like to thank Professor Hal Abelson for his generous support and guidance, for his insightful comments, which greatly improved the content of this thesis. I also express sincere gratitude towards Philip Greenspun PhD, for his influential introduction to the world of database-backed Web services. He gave me the experience and cultivation necessary to become a “Webhead.” Without Philip or his company’s product, the ArsDigita Community System, this thesis could not be accomplished with the same efficiency and robustness. I owe thanks to Randy Graebner, my “partner in crime” for this project. Randy has been a significant contributor, a great teammate, and a supportive friend. Thanks, Randy, for being 100% reliable and 100% fun to work with. 🙏 I owe it to you for submitting this thesis on the date printed on the title page and not a month later.

Ever since my first month here at MIT, Charlie Lee has been my best friend, always being there for me throughout the good and bad times. He has truly helped me grow in every way possible - academically, socially, in both personality and character. I am so lucky to walk out of college with a friend like him. I dedicate this thesis to Charlie and my parents who made possible my existence, the fundamental requirement for the writing of this thesis.
Table of Contents

1. Introduction ........................................................................................................... 7
   1.1. Motivation ........................................................................................................ 8
   1.2. The Idea ........................................................................................................ 8
   1.3. Overview ......................................................................................................... 9
       1.3.1. Site-wide administrators ........................................................................ 9
       1.3.2. Class instructors ................................................................................... 9
       1.3.3. Students ............................................................................................... 10
       1.3.4. Summary of the system’s capabilities .................................................. 10
   1.4. Division of Project .......................................................................................... 11
   1.5. The Paper ....................................................................................................... 11

2. A User Centric System ......................................................................................... 13
   2.1. User-Centric vs. Class-Centric .................................................................... 13
   2.2. Example Usage Scenarios ............................................................................. 14
       2.2.1. Student—Obtaining course material and collaboration ...................... 15
       2.2.2. Teaching assistant—Evaluating students ............................................ 16
       2.2.3. Professor—Managing the class ............................................................ 17
       2.2.4. The department administrator—Creating subjects ......................... 21
       2.2.5. The site-wide administrator—Creating departments ....................... 22

3. Related Work ........................................................................................................ 23
   3.1. Blackboard.com .......................................................................................... 23
   3.2. WebCT .......................................................................................................... 24
   3.3. Command ...................................................................................................... 26
   3.4. eCollege.com ............................................................................................... 27
   3.5. Serf ................................................................................................................ 28
   3.6. Eduprise.com ............................................................................................... 29
   3.7. Virtual-U ...................................................................................................... 30

4. Design .................................................................................................................... 31
   4.1. Representing Classes .................................................................................... 31
   4.2. User Administration ..................................................................................... 32
   4.3. Content Management and Distribution ...................................................... 33
   4.4. Collaboration and Coaching ....................................................................... 34
   4.5. Customizable Education Portal .................................................................. 35
   4.6. Security and Permissions ............................................................................ 36

5. Implementation ..................................................................................................... 37
   5.1. System Architecture and the ACS ............................................................. 37
       5.1.1. ACS overview ..................................................................................... 37
       5.1.2. The architecture .................................................................................. 37
       5.1.3. ACS modules ...................................................................................... 38
List of Figures

2.1: Personal Education Portal ................................................................. 15
2.2: Project discussion Q&A Forum for the Site for Hunger Team ................. 16
2.3: Assignment Info and Student Evaluations ........................................ 17
2.4: Scheduling office hours for online sign-ups .................................... 18
2.5: Class list with student information .................................................. 19
2.6: Information page about a student including grades .......................... 20
2.7: Evaluations given by a single grader .......................................... 21
2.8: Site-wide administrator home page ................................................. 22
5.1: Hierarchical view of the data model ................................................. 40
5.2: Course homepage with news and current assignments/exams ............ 43
Chapter 1: Introduction

1.1 Motivation

Over the past decade, we have witnessed an ongoing information revolution brought about by the Internet. We have learned to harness the Internet for more efficient communication and ubiquitous information. These benefits have enabled us to bring many aspects of our lives online. In media and broadcasting we have sites like http://cnn.com; in finance and trading we have http://www.etrade.com; in business and commerce we have http://www.amazon.com. Yet we have not seen a major impact of the Internet in the field of education. Granted, people have developed services that host classes online, and many courses have their own homepages, supplementing the traditional classroom experience. But we are missing a centralized, integrated system for online education that is extensible and customizable enough to be adopted by classes taught both at institutions and by individuals.

A significant number of traditional classes use the Web to supplement lectures and recitations by posting relevant information such as recent news and handouts. There are, in addition, many products offering a variety of services to host class websites. Many, in fact, do an adequate job of using the Web as a collaboration tool by providing real-time chat, virtual whiteboards, and threaded discussion forums. However, the one thing missing in virtually all of these standalone websites and commercial Web services is the interaction of classes with each other. Since most of these services have been developed to either complement a traditional brick-and-mortar class or teach a stand-alone class online, they were designed using a class-centric paradigm rather than a user-centric one [Pau92]. The two paradigms differ in their data design focus. Since the education systems are all data-driven, having a user-centered data model ensures that the data follows the user rather than being confined within a class. Within a user-centric system, the user is presented a personalized set of data that is relevant to him/her. Many of the class-centric services, on the other hand, exist merely as a large collection of standalone class websites (Section 2.1 presents the above discussion in more detail).
Many existing education software solutions have been built to be usable from the perspective of the professor rather than that of the student. The area of online education desperately needs a centralized, integrated, and user-centric system that is extensible and customizable enough to handle the needs of large universities while simple enough to be used by a single professor.

1.2 The Idea

Without collaboration, the Web becomes little more than an easily accessible file storage system. Numerous studies have shown that the ability to collaborate is the key benefit from using the World Wide Web to enhance the student's learning experience [Woo97]. It provides a collaborative environment where students communicate with each other both synchronously (chat rooms and video conferencing) and asynchronously (bulletin boards and shared file storage systems). English teachers, for example, have found that a collaborative writing environment provides an effective means of encouraging students to learn from one another by critiquing each other's writings. In addition, instructors from a variety of areas have found that Web-based discussions can lead to greater in-class participation by all students involved [Hi190, Mor95, Cli90].

The problem of building a collaborative website can be fairly straightforward. However, people will not use it unless it is both intuitive to use and provides significant new capabilities to assist them with everyday tasks. Most users will not invest a large amount of time learning how to use the service before enjoying any of its benefits. In addition, the system must convince users to break away from habitual ways of performing tasks by providing features that are significant improvements over the existing solutions [Gre99].

The goal of the System for CLASS is to provide all types of users with significant new capabilities while maintaining a simple and intuitive user interface. To accomplish this task, we have adopted a user-centric approach in designing the system (see Section 2.1). The reasoning behind this is simple; users represent the most diverse source of
knowledge and provide the most valuable content in a collaborative system. As an education system, the most important role for CLASS is a broker that brings users seeking knowledge to those who provide knowledge. Thus, the most powerful way to enhance the learning and teaching experience is by providing highly accessible and effective means of collaboration between user communities.

1.3 Overview

Our system provides significant new capabilities to three types of users: system administrators, instructors, and students (see Section 2.2 for example usage scenarios).

1.3.1 Site-wide Administrators

The system reduces maintenance burden of the system administrators by delegating their responsibility over individual classes and departments to the respective instructors and staff members. Delegation also increases customizability, suiting the specific needs of each of the system’s subcomponents. Separation of administrative tasks results in a more streamlined set of site-wide administrative duties. To ensure system consistency and coherence, our design must ensure that the subcomponents are modularized and distributed over a controlled delegation. This is similar to a “thread safe” design, where the delegated tasks are isolated and run locally without unexpected global impact on the system. In other words, all tasks that have global impact remain assigned to the site-wide administrator, whereas the administration of individual classes will be under the jurisdiction of their respective staff members. For the system to run properly, the site-wide administrator is responsible for nightly backups and keeping the servers running. The only other system administrative task is initial system configuration/creation of academic departments and subjects. If applicable, the site-wide administrator also needs to create classes that do not belong to any departments.

1.3.2 Class Instructors

Since user registration is completely automated, the staff members of individual departments and classes have local control over user management and group admission. Instructors can collaboratively develop documents, share a single grade book, monitor the
progress of individual students, issue assignments and receive answers online, and locate past course material. In addition, bulletin boards and chat rooms provide a controlled means for students to help each other learn. An open question and answer forum not only helps students obtain answers more quickly but also encourages them to ask questions they are less inclined to ask in front of an instructor (e.g. being afraid to ask a “stupid question”). One important issue, however, is the accuracy of answers provided by students to their peers. This problem has been presented and solved in other systems, for example, by either monitoring the bulletin board/chat room discussion or flagging answers that are not provided from the instructor or teaching assistant.

1.3.3 Students

To students, CLASS is both a coaching and course management system. The education portal integrates content from the student’s classes and displays reminders such as assignment due dates and exams. As mentioned previously, the system provides collaborative tools so students can obtain timely help instead of waiting for the next class or office hour. The distribution of course material online via a customizable interface means a greater penetration of information that is both convenient and beneficial to students. Instead of managing a large “bible” of course material each semester, the student can now shift some or most of the mass storage to his CLASS account online. Via this account, he can access his problem sets, exams, grade history, and all course-related handouts and announcements.

1.3.4 Summary of the System’s Capabilities

By applying the benefits of the Internet to teaching and learning, the system helps the teaching staff perform their duties more efficiently and enhances the students' learning experiences by making a wider range of resources readily accessible. The System for CLASS models the classroom as a database-backed Web service where members of each class belong to well-defined groups of users. As a result, a user can remotely interact with other members of the same community, access information asynchronously (e.g. shared file storage system), perform tasks specific to his/her role, and make contributions as a community member.
What does this translate to the system's application to education? Students can obtain help from the teaching staff remotely during online office hours. They can share knowledge and help answer questions asked by fellow students when the teaching assistant (TA) is not available. The teaching staff can collaboratively develop course materials and distribute them via a centralized medium. Instead of collecting student grades from many TAs, each maintaining his or her records separately, professors have immediate access to all student data in a single format, managed by a centralized database. Finally, all users have access to a reliable archive of all announcements, course material, and communication that are relevant to them.

1.4 Division of Project

This thesis is a joint effort between Randall Graebner and myself. Our collaborative work include data modeling, designing and coding the administrative and user pages in the main class system, communicating with the MIT Sloan School about project requirements, and bug fixes. Our individual efforts are focused on integrating specific ArsDigita Community System (ACS) modules into the project and updating them as necessary to satisfy our project requirements. I had primary responsibility over the education portal (Sections 4.5, 5.5) and an effective design in the user interface of the class pages (Section 5.2), whereas Randy's individual work focused on the file system module (Section 5.8) and the overall security and permissions of the system (Sections 4.6, 5.7). Sections that were primarily Randy's work (security, file storage system) will be documented in this paper but to lesser detail. In the written part of this thesis, our collaborative work exists in Section 1.1, 1.2, Chapter 3, Sections 6.1, and Section 8.2. I made use of Randy's work in Sections 8.1 and Section 5.1.4.

1.5 The Paper

Now that an overview of the project has been laid out, the remaining chapters present the system's design and implementation issues in further detail.
Chapter 2, A User-Centric System, defines the term user-centric and outlines some of the differences between user-centric and class-centric systems. The chapter concludes with several use cases to better demonstrate how our system can be used.

Chapter 3, Related Work, discusses previous work done in the area of online education. There are several companies offering enterprise level software solutions to this problem. Their work is examined so that a clear contrast can be drawn between their work and the CLASS system.

Chapter 4, Design, discusses the high-level goals and structure of the system as well as several specific design issues.

Chapter 5, Implementation, presents the implementation details and the technology used, comparing these choices to our design goals.

Chapter 6, Feedback, discusses some of the feedback we have received from pilot classes, how we modified our design based on this feedback, and the lessons we learned from it.

Chapter 7, Accomplishments, compares this project to the related work presented in Chapter 3. In addition, it describes several schools that have planned to adopt the CLASS system.

Chapter 8, Conclusions, discusses the success of this project and outlines future enhancements that are required to make CLASS an enterprise level solution.
Chapter 2: A User Centric System

In the Introduction, I mentioned that CLASS is a user-centric system, whereas most of the related work existing in the mainstream market today is class-centric. Section 2.1 defines user-centric in more detail and presents key differences between user-centric and class-centric systems. Section 2.2 provides example scenarios of how the CLASS system functions for several types of users.

2.1 User-Centric vs. Class-Centric

All Web-based education systems provide a large set of tools to host and administer classes online via a browser interface. These database-backed systems share the characteristic of being data-driven. The difference between user-centric and class-centric systems lies in the high-level approach of how the relational data is handled with respect to the system’s functionalities. In order to provide a feature-rich tool set to the user, the tool sets’ functionalities need to be keyed from a centralized piece of the data model. Whereas class-centric systems associate defined sets of features to each class, our system stems all features from the user and user group data model.

More generally, class-centric system toolsets are configured on the class level, whereas user-centric systems can be further customized for each individual user. This difference is especially important because it provides greater flexibility to systems that do not exhibit a one-to-one mapping between classes and users. The user-centric design is a more generalized version of the class-centric system by providing a more flexible mapping of the system’s toolset to the individual members of a class rather than to the class as a whole. This is not just limited to the “personalization” aspect of a system. For example, as the system grows via open-source development, the existing user-centric design can be propagated easily from the skeleton based on users and user groups (Section 4.2), where the system’s existing set of functionalities (e.g. bulletin boards, assignment distribution, etc) currently extends. For example, rather than associating the bulletin board with a class
(as in class-centric systems), it is associated with a group of users. As a result, users have different levels of access to a particular discussion topic depending on their roles within the group.

A significant functionality of our user-centric system is the personalized portal page. The portal provides one centralized entry-point to all of the classes and departments that the user belongs to, regardless of whether he/she is a professor, teaching assistant, or student. In addition, the portal serves as a personalized management tool with features like an integrated calendar and mini-tables that scrape content from foreign sites (e.g. stock quotes, weather, movie listings, custom news, etc). Since the portal is a skeleton that hosts plug-in content, the set of content requirements is highly customizable to the individual organizations that adopt our system. The portal’s web-based ADP interface for defining modules is designed for the purpose of easy content integration.

Interfaces of user-centric systems are personalized, “pushing” only relevant and timely data to the user. Once the user’s identity is verified (via a single log-in or client side cookies), the system pulls all information from the database keyed by the user’s ID. A user-centric system also prioritizes community-building features, a prominent focus of the Arsdigita Community System, on which CLASS is built. One rather unique feature of our system is the class/department directory with controlled access so authorized users can locate information about other users within the system. This information ranges from the user’s name and email address to a portrait and a list of all postings he/she has ever made to a particular bulletin board. In short, our system’s data is primarily user-centric, making implementations such as keeping a repository of a user’s class work over his college career a relatively trivial task.

2.2 Example Usage Scenarios

To give a concrete description of what we have discussed so far, this section presents several scenarios that demonstrate how the system can be used by various types of users.
2.2.1 Student – Obtaining course material and collaboration

Bob is a student in 6.001. He logs on to the education system via the main server index page. After logging on, he is taken to the portal (Figure 5.1), where he sees that he has 2 assignments due and one quiz this week.

![Sloan Portal: Bob Thomas]

**Figure 2.1: Personal Education Portal**

Clicking on the link for the problem set due today takes him to the assignment’s information page, where he uploads a submission for the problem set. Bob then decides to find his teammates for the 6.916 project to discuss their data model. He returns to the portal page (a.k.a. workspace) and selects the class homepage for 6.916. From the 6.916 index page, Bob sees a new announcement that there will be a guest speaker from Oracle in next week’s lecture. He clicks on the link to Q&A Forums on the left hand side of the page and selects his team’s private Q&A Forum from the list of 6.916 discussion forums. From the Q&A forum (Figure 5.2), he sees all threads about the discussion of their
project, the Site for Hunger, and posts an answer to the discussion about their data models.

**Site for Hunger**

*Your Workspace : Online Education System Discussion Forums : Site for Hunger*

- [Ask a Question | Search | Unanswered Questions | New Answers]
- Data Model Discussion
- Site design

This forum is maintained by Bob Thomas. You can get a summary of the forum's age and content from the statistics page.

If you want to follow this discussion by email, click here to add an alert.

*bob@mit.edu*

Figure 2.2: Project discussion Q&A Forum for the Site for Hunger Team

### 2.2.2 Teaching Assistant – Evaluating students

Susan is a graduate teaching assistant for 6.001 and is also taking a class called E-Commerce Architecture Project (ECAP). From her workspace, she clicks on the link to May 3 and adds a 6.001 problem set due for that day. Then she clicks on the link to the 6.001 administration page and looks up assignment submissions for Problem Set 5. She sees that 6 of her students have not been evaluated for Problem Set 1 (Figure 5.3), so she downloads the answers for each student and updates their submissions with her comments and uploads them again. She then fills out the student evaluation form for Problem Set 5 for each student and submits the form with the grades and comments, checking the radio button to make the evaluation viewable to the students. These evaluations go into the database, and from the grades list for Problem Set 5, she can see the problem set grades for all students who have been evaluated.
The following students have not been evaluated for Problem Set 1

- Ting, Tiffany
- Janet, Andrew
- Yen, Jenny
- Lin, Linhs
- Ping, Clara, Adam
- Alwin, Fred

The following students have been evaluated for Problem Set 1

- Graham, Randy

Figure 2.3: Assignment Info and Student Evaluations

Susan then returns to her workspace and clicks on the link to the ECAP homepage. The right hand side of the course homepage displays all new announcements posted since her last login and any assignments/exams/projects that are due within the week. She finds and downloads a new handout that has been uploaded since her last login – a related reading material from yesterday’s lecture.

Jeff, one of Susan’s 6.001 students logs onto the 6.001 homepage and sees in the news announcement that his problem set has been graded. He clicks on the link in the announcement to view his grades and Susan’s comments and downloads the graded assignment that was a version of his uploaded answer, modified by the TA.

2.2.3 Professor – Managing the class

Professor Smith is a lecturer for ECAP. From the portal, he clicks on April 14 to schedule office hours with each student in the class to discuss their term project. From the office hours scheduling page, he specifies office hours in 15-minute increments from 11am to
4pm, with noon-1pm reserved for lunch (Figure 5.4). Then he returns to his workspace and enters the ECAP admin page. He uploads lecture notes for both yesterday and next week's lectures. He sets read permission to all for yesterday’s lecture notes and read permission to teaching assistants only for next week’s lecture notes because he wants the TA’s to proofread the lecture notes beforehand. He then spams students in the class about signing up for office hours on April 14. Then he enters the ECAP staff only bulletin board and posts an answer to the thread about course content development for the second half of the semester. To schedule the next staff meeting, he checks the calendar for all three teaching assistants in the class and proposes next Friday, when the TA’s appear to be most available.

**Add Office Hours**

*Add Office Hours*

![Office Hours Form]

**Figure 2.4**: Scheduling office hours for online sign-ups.
### All Students in Software Engineering of Innovative Web Services (Spring, 2000)


<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
<th>Account Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adler, Jordan</td>
<td><a href="mailto:jsadler@mit.edu">jsadler@mit.edu</a></td>
<td>95</td>
</tr>
<tr>
<td>Agbobh, Peter</td>
<td><a href="mailto:pmagbobh@mit.edu">pmagbobh@mit.edu</a></td>
<td>186</td>
</tr>
<tr>
<td>Anand, Ethan</td>
<td><a href="mailto:ianand@mit.edu">ianand@mit.edu</a></td>
<td>89</td>
</tr>
<tr>
<td>Artz, Michael</td>
<td><a href="mailto:slyph@mit.edu">slyph@mit.edu</a></td>
<td>181</td>
</tr>
<tr>
<td>Backskind, Christian</td>
<td><a href="mailto:draco@mit.edu">draco@mit.edu</a></td>
<td>183</td>
</tr>
<tr>
<td>Bender, Ryan</td>
<td><a href="mailto:jrbender@mit.edu">jrbender@mit.edu</a></td>
<td>87</td>
</tr>
<tr>
<td>Bleras, Aggelos</td>
<td><a href="mailto:aggelos@media.mit.edu">aggelos@media.mit.edu</a></td>
<td>185</td>
</tr>
<tr>
<td>Bonnet, Mike</td>
<td><a href="mailto:mbonnet@mit.edu">mbonnet@mit.edu</a></td>
<td>90</td>
</tr>
<tr>
<td>Brown, Chad</td>
<td><a href="mailto:yandros@mit.edu">yandros@mit.edu</a></td>
<td>64</td>
</tr>
<tr>
<td>Byrne-Clarke, Adrian</td>
<td><a href="mailto:adrianboc@mit.edu">adrianboc@mit.edu</a></td>
<td>98</td>
</tr>
<tr>
<td>Chak, Dan</td>
<td><a href="mailto:chak@mit.edu">chak@mit.edu</a></td>
<td>91</td>
</tr>
<tr>
<td>Chang, Peter</td>
<td><a href="mailto:psc@uclink4.berkeley.edu">psc@uclink4.berkeley.edu</a></td>
<td>82</td>
</tr>
<tr>
<td>Chao, Tony</td>
<td><a href="mailto:tchao@mit.edu">tchao@mit.edu</a></td>
<td>93</td>
</tr>
<tr>
<td>Chen, Wei</td>
<td><a href="mailto:wychen@uclink4.berkeley.edu">wychen@uclink4.berkeley.edu</a></td>
<td>188</td>
</tr>
<tr>
<td>Chu, Michael</td>
<td><a href="mailto:micha@mit.edu">micha@mit.edu</a></td>
<td>100</td>
</tr>
<tr>
<td>Cox, Rick</td>
<td><a href="mailto:rick@restomp.berkeley.edu">rick@restomp.berkeley.edu</a></td>
<td>81</td>
</tr>
</tbody>
</table>

Figure 2.5: Professors can view a list of all students in the class, sorted by the student’s name, email address, or account number.
Figure 2.6: Professors and teaching assistants can view grades and evaluations for each student. They can also click on the grader’s name and see how that person has graded other students.
2.2.4 The department administrator—Creating subjects

Jim Moore is in charge of information technology for the chemistry department. He has recently been notified that they will be offering a new course, 5.688. So, he logs on to the department's administration page. From there, he sees that there is not yet a subject named 5.688 so he adds one, giving the appropriate permissions to the class instructor to administer the subject and class Web pages. Since the professor can now take over and create the class Web page, Jim's work for the day is complete.
2.2.5 The site-wide administrator—Creating departments

Ben Jones is the system wide administrator of the system. He logs on to the CLASS system and sees that there are 14 different departments and 37 total subjects in the system (Figure 2.8). He creates a Management department for course 15, giving the Department Head permission to administer the department. He then remembers that he needs to add a summer term, thereby allowing the system to host classes over the summer.

![Figure 2.8: System Administrators get a top-down view of the system's departments, users and subjects.](image-url)
Chapter 3: Related Work

3.1 Blackboard.com

Blackboard.com is one of the most popular websites for online education. It offers My Blackboard, which is both a personalized user interface and a centralized login point to the various offerings of the system. Their system is built on technologies such as mySQL, Apache, mod_perl, and Java. With features like My Blackboard, the company claims to change “the product use paradigm from a course-centric model to a user-centric model.” From the My Blackboard page, users can view course and campus announcements, access direct links to course websites, view personal calendar and to-do lists, personalize the appearance of the page, and access links to campus departments, bookstore, library, registrar, Web resources, etc.

As a commercial product, Blackboard.com offers a wide range of features to suit the needs of a large user base. The main site allows instructors to post course documents, staff information, assignments, multimedia course material in any file format. Its assessment features measure student progress, customize lessons, administer quizzes, and perform surveys. In addition to the ability to administer classes and develop course materials online, Blackboard.com provides a variety of tools for both synchronous and asynchronous collaboration and communication. The asynchronous collaboration tools include bulletin boards, announcements postings, email, and audio/visual media files. The synchronous collaboration tools include chat rooms and virtual white boards. Blackboard.com also provides other functionalities like a calendar for time management and electronic blackboard for note taking. In addition, the Blackboard Resource Center provides a library of supplementary course material for a set of disciplines.

The advantage about Blackboard.com is it provides a pretty complete set of services to host and develop course material online and tools to help students collaborate both with the teaching staff and among themselves. However, users of the system do not appear to
take full advantage of these collaboration tools because they are not easily accessible. As discussed in Section 2.1, this is an artifact of a class-centric system. Once you log into the class Web page, it takes 3 more levels to access the virtual chat, for example. In order for students to collaborate online rather than through more conventional ways such as telephone, face-to-face, or even email, collaboration tools must be readily accessible within the class site. They should be closely coupled with other features of the system where students most likely would need collaboration, and these can include assignments, team projects, and exams.

The user-centric approach allows us to achieve vast improvement over Blackboard.com in integration. Blackboard.com hosts many disjoint classes that do not share data among each other. However, a student (especially one at a university) is highly likely to take more than one class, and a powerful education system should be able to host all of his/her classes via one Web service. This Web service should be capable of integrating information from all of his/her classes and display it in a useful way. For example, the system should display assignment due dates, upcoming exams, and team meetings for all classes that a student is taking in one single calendar. The student should also be able to view news and announcements for all of his/her classes via one page rather than having to logging into the homepage of each class.

3.2 WebCT

WebCT http://www.webct.com/, like Blackboard.com, allows instructors to create complete online courses or material supplementing classroom-based courses. Like Blackboard.com, WebCT supports a Web-based educational environment with a range of standard collaboration and performance assessment tools along with support for user management and course development. Similar to the Blackboard Resource Center, WebCT has an extended “e-Learning Community” that contains a library of resources and a collection of discussion forums separated by course topic. Unlike Blackboard.com, WebCT also offers customizable viewing tools to help students create personalized views of the course material. One tool, dubbed “learning Paths,” allow an instructor to create a
guide through the course content for the students. Another significant difference from Blackboard.com is the modularization of the system design. WebCT offers a set of components (modules) that instructors can select and customize for their specific course. Modules include functionality like quiz, glossary, my progress, content compiler, and file manager. For example, the Content Compiler is a module that allows students to compile a collection of content from the course website for convenient printing and viewing.

WebCT’s modularized toolsets results in greater flexibility in creating course websites, as tools can be added/deleted from a course website depending on the specific need of the course without clogging the interface with useless features that confuses the user. Similar to our education system, WebCT uses a portal page as a centralized entry point to all of a student’s courses. This portal offers a view of course-related links and customizable personal links. However, it does not push course content like the most recent assignments or quiz dates to the portal.

There are two options for hosting a class: 1. hosted by WebCT. It requires no extensive systems knowledge but has limited customizability. Students enter course site through WebCT’s website. 2. hosted locally via a larger software package that supports the instructor running the course site from a local server. This option provides more customizability and requires a licensing fee. WebCT runs on Perl and CGI, which result in noticeable performance penalties. Overall, WebCT consists of commercialized features that seeks to please an as wide a user base and course content as possible. However, the rich set of features also make hosting and developing course content a complicated process for instructors, who have to understand the functionality of many tools before implementing them. The trade-off of a simplified, customizable interface from the students’ view is a more complicated administration view for the instructors.

Despite the rich set of features, WebCT can improve in the following areas. The system does not offer an easy way to group students, for example, into teams working on the same project. The course calendar is not very user friendly. Like Blackboard.com, there is no one integrated calendar accessible from the portal page, and it does not display anything unless the user clicks on a specific day. Courses also do not appear to have
associated status, which allows them to be inactivated after an expiration date or obsoleted by newer offerings of the course. The system displays the list of courses in a long list organized alphabetically by name.

### 3.3 Command

Command ([http://command.mit.edu](http://command.mit.edu)) is a Web-based course management and delivery system developed at MIT in conjunction with industry partners like Intel, IBM, and Lotus. Users enter the system via a list of courses organized by MIT departments. The advantage of Command is that it is MIT centric and currently involves participation from many MIT courses. It offers MIT courses a set of existing course page functionality such as course material organization into assignments, solutions, and lectures and posting of new announcements. These templates eliminate the hassle of building a course website from scratch, managing it at the html level, and organizing the course file system.

Command models after the traditional course homepage, which provides a repository for course documents and information. As a result, it is heavy on course information management and delivery but does not provide enough functionality for personalization, user management, or collaboration. There is no entry point where students gain speedy access to their course websites. Instead, one must either bookmark each course Web page individually or select his/her course from the list organized by department. Lack of integration of the courses result in separate username and password authentication for each course the student enters.

The system is not user-centric, which results in poor usability. For example, it requires the user to log in separately for each class. The calendar, a highly demanded and useful feature, is buried three levels deep. In addition, Command does not provide support for grades and student evaluation, and the user interface has low customizability. There is a lack of file versioning, which is important for collaborative course development. Defining course material to just lectures, assignments, and readings, along with the lack of support for grades assessment, limits the system to traditional classroom courses on
academic topics. In addition, the system does not provide enough ability for professors to develop course material, manage course staff, and keep track of user stats. User management tools are basically non-existent, except for a list of students who are registered for the course and their email addresses.

In summary, Command is a collection of what we currently have as course homepages in one central place but without integration/communication among them. The system is not user-centered, i.e., it does not take advantage of knowing the user's identity and displaying the information that is catered to his/her roles.

3.4 eCollege.com

eCollege.com offers a large set of tools for use by universities, faculty, and students. Their Campus Gateway product provides all of the standard online educational tools such as bulletin boards, chat rooms with archiving, syllabus creation, announcements, collaborative grading, online testing, and automatic email archiving. Professors have access to general usage statistics on the bulletin boards and chat rooms in addition to integrating the syllabus with the online calendar. Students can create a personal homepage on the server, view their grades in the classes they are taking, and are provided with useful links to outside sites such as the local credit union and other non-university related websites [Eco00].

CampusPortal, also offered by eCollege.com, takes the Campus Gateway idea one step further. This product includes all of the features of the Campus Gateway in addition to the ability to tie the system with a university's existing database as well as other outside resources. CampusPortal also includes additional tools for use by the professor to aid in creating the website. The CampusPortal software package provides the ability to create an entire online campus. However, it lacks the ability to associate courses with each other and it lacks a unified view of all classes for the students. The lack of course association means that, among other things, there is no easy way to say that two classes are actually the same class offered during two different semesters. In addition, both CampusPortal
and Campus Gateway are implemented with a class-centric approach. While they both provide adequate tools for students taking a single class, they lack personalization on the user level. This deficiency makes it difficult for users to use the system when they are taking multiple classes.

3.5 Serf

Server-side Educational Records Facilitator (Serf) is a Web based distance education developed by Fred Hofstetter, a professor at the University of Delaware, in collaboration with PBS [Ser00]. Serf offers the smallest feature set out of all of the software packages reviewed. The Serf environment provides students with access to standard features such as a course syllabus, bulletin board, chat room, and mailing lists. In addition, it provides a collaborative grade book so that many instructors can make entries into the same grade book. From the students' perspective, they can view their grades on the website thus giving them an idea of how they are doing in the given class. Also, they are able to submit their assignments online and view the syllabus in the form of a calendar [Icu00].

On the whole, Serf has been designed with the goal of being easy to use. The pages are laid out nicely and links are self-explanatory. The system has been developed with the user experience in mind and tries to give the instructor a lot of flexibility in designing both the content and user interface for a class while still keeping a consistent interface for use by the students. For the most part, Serf succeeds at providing an interface that is intuitive and easy to navigate. However, there are some major areas where it could be improved. For instance, students must wade through the course catalog to get to the class home page. This can take a lot of time, especially if the student is using the system for multiple classes. A better approach, used by CLASS, is to present information and links for all of the classes a student is registered for on the first page seen by the student after logging in.

While Serf does provide most of the core functionality of most distance learning websites, including online exams, it is lacking some core collaborative tools. For
instance, it lacks the ability to group students into teams or sections. In addition, it does not provide an area for students to upload personal files or files that can be shared among different students. Many of the problems found in Serf stem from the fact that it was designed under a class-centric paradigm rather than a user-centric one. These deficiencies, combined with its requirements that the system only works on servers running Microsoft NT and Microsoft SQL Server make this solution less than optimal.

3.6 Eduprise.com

Eduprise.com is an "e-learning solution" [Edu99] originally developed at the University of North Carolina. The developers wanted to create an easy to use system to facilitate teaching over the Internet. The system, which runs only on the Microsoft NT and Macintosh operating systems, provides a large set of tools while allowing for easy customization of the site. In addition, the site has been built such that professors can choose whether they want to teach a class entirely over the Web or if they want to use it to simply complement a traditional class. Eduprise not only provides bulletin boards and chat rooms but it also provides statistics on their usage, timed and untimed online tests, the ability to include streaming audio and video as part of the class, the ability to group lectures and problem sets into lessons, personal home pages for students and more.

Eduprise, like many of its competitors, has been designed around using the system for a single class. While the system succeeds in providing most features desired by the faculty, this class-centric model makes it difficult for students in multiple classes to use the site. For instance, it does not have simple functionality such as showing students the classes for which they are currently enrolled. Also, while the system does provide a calendar view of the term for a given class, if a student is taking four classes, they have to view four different calendars in order to see all of their assignments.
3.7 Virtual-U

Developers at Simon Fraser University have been working in conjunction with several companies to produce Virtual-U, a customizable virtual campus for distributed learning. Virtual-U provides a fairly complete set of services to the end user, including many of the standard online educational tools such as bulletin boards, chat rooms, online assignment submission, online tests, collaborative grading, and course structuring materials. In addition, the system has been designed to be multi-lingual so users can use the site in their choice of Spanish, French, Portuguese, or English [Vir00].

Virtual-U is a self-contained module that can be completely customized as a single web service by any university or individual. This means that universities using the software can customize it so that it looks like a continuous part of the pre-existing computing system. However, a major drawback to the system is that it has been built with the implicit assumption that students will only be taking one class at a time. For instance, in order to see the calendar for a single class, a student must find the class in the course listings, click to the home page and then click on the calendar link. Only then can students view the calendar (or syllabus) for the class. Forcing students to repeat this for every class is not a user-friendly feature. In addition, Virtual-U does not have any way to relate classes to one another. Therefore, it is not possible for a student to see if one class is a prerequisite for another or if the class has ever been offered before.
Chapter 4: Design

The System for CLASS's goal is to provide students, teaching assistants, and instructors a “one-stop” shopping for all their course-related needs. It provides them with tools to communicate with other users. From the portal launch-pad, they have access to all class material relevant to their roles so they can conveniently plan, track, and complete their tasks. Our system is user-centric by leveraging knowledge about the user and his/her login time to “push” important information to the user in a timely and effective manner.

4.1 Representing Classes

Classes are modeled as groups consisting of users each with distinctive membership roles (e.g. students, TA’s, instructors, etc.). Each class is a child of a subject, which owns properties such as name, semester, description, professor in charge, but does not associate with any users. Thus, we are able to model class offerings during different semesters as instances of the same subject. This design allows us to build a simple interface between different offerings of the same subject and an easy migration of a subject offering from one semester to the next. At the beginning of a semester, the current offering of 6.001, for example, automatically “obsoletes” the 6.001 class offered in the previous term. We can therefore preserve all previously offered versions of a class while only the current class is considered “alive” in the system.

The relationship between subjects and classes is analogous to the object oriented programming concept of classes and objects. Specifically, Java classes define a set of properties and behavior. Objects that are instantiations of the class all inherit the same set of class properties and methods. In our system, an academic subject is the concept of a “course,” whereas classes are instances of a subject that contains users and “live data” relevant to that instance. A new class created under that subject automatically obsoletes all previous instances of the same subject.
Departments, on the other hand, consist of both subjects and users. (For detailed discussion about how users are organized within each department and class, see Section 4.2, User Administration.) To handle the common case where subjects are joint offerings between two or more departments, a many-to-many mapping table links subjects to departments by their primary keys. This mapping table carries information that is specific to a subject-department pair (e.g. subject number, level of credit).

4.2 User Administration

We want to build an education system that can be adopted by individuals and organizations of any size, from a single class to an academic department or a university consisting of several academic departments. As discussed in Section 2.1, a benefit of our system is the characteristic of being user-centric instead of class-centric. Users can be members of several classes and/or departments and have specific roles (e.g. professor, student, and teaching assistant) that vary with each membership. The appropriate actions that a user can perform as a member of a group (e.g. a class, team, or department) depend on his/her role within the group. The system manages all information relevant to each role, so it knows the teaching assistants' office hours and the students' account information. As group member, the user has appropriate permissions to access bulletin boards, new announcements, grades, and handouts of his/her classes. Since the system is user-centric, it takes advantage of its knowledge of the user's identity and provides the user with efficient and comprehensive access to information that is relevant to him/her.

In addition to managing user information, making this information easily and effectively accessible is important for a system that is both an educational and a collaborative community. For example, the professor can find out which students are doing well or badly in the class, who has been most actively helping other students on bulletin boards and in chat rooms, and schedules of other teaching staff so he/she can schedule meetings at a time that the most people can attend. On the other hand, students can easily lookup information about other students, their interests, and availability as potential teammates. At the department level, a Sloan School administrator can view reports that show, for
example, how much effort a TA has put forth in helping students online, and how student enrollment varies in a particular class over time. A student's academic advisor can have access to detailed information about the students’ academic enrollment during a semester.

Accessing academic information via the Web presents obvious privacy issues regarding who has permission to view which information under what circumstances. We built our system based on the assumption that the user's identity has been authenticated. Once we know the user’s ID, we determine whether the user with a role of X has permission to do Y (for more details, see Section 4.6). Since privacy is primarily a social issue, we have chosen to not address it completely in our initial implementation. The best way to address privacy is perhaps via a combination of customizability and notification. Customizability allows information access to be controlled at a finer granularity, where a student may specify whether anyone other than the class instructor or teaching assistant should have access to his class grades. On the other hand, notification (in the most basic form, a private policy) helps relieve privacy concerns by giving the user sufficient warning about who has access to his/her academic information.

4.3 Content Management and Distribution

The system comes with a file storage system that organizes and serves all course material and student assignment submissions. These can include multimedia lecture demos, problem set handouts, research papers, and student projects. The interface for uploading and distributing these files should be easy to use and come with the appropriate permissions control to support the following scenario:

A week before the quiz, the professor uploads a draft solution to the quiz and notifies the teaching staff to proofread and comment on it. A teaching assistant downloads the solution, corrects some typos, and uploads the modified version. Later, a recitation instructor downloads what is the updated version and adds an alternative solution to one of the problems. After the quiz is administered, the professor goes back, changes the permissions on the solutions file to make it viewable by students in the class.
In addition, the system serves as a central place for the development, distribution, and submission of assignments. The professor should be able to upload an assignment and set its due date. The student should be able to upload his/her solutions to the assignment before the due date, and the teaching assistant can download the student's submission, grade it, and enter an evaluation of the student's answers into the database.

4.4 Collaboration and Coaching

The traditional classroom experience encompasses two important components: collaboration among classmates and coaching by the instructor. Our system should not only imitate but also enhance the educational experience beyond the classroom.

Via the System for CLASS, a student should be able to interact with other students through both asynchronous and synchronous collaboration. Asynchronous collaboration is a more traditional concept in Web-based communities, where users interact with each other through non-real-time forms of communication. Asynchronous collaboration is widespread today via online question and answer forums (e.g. bulletin boards) and email forms. Synchronous collaboration, on the other hand, is gaining more presence as real-time interaction becomes more crucial in the way people are using the Internet today. For our system, synchronous collaboration means students can interact with other students or the class instructor in real-time via a Web browser. A chat forum allows students to discuss an assignment with other students or ask a teaching assistant on duty to explain a concept. Our system design should allow easy integration of additional plug-ins (such as video conferencing and Java-based white boards) as they become available.

An indelible aspect of the classroom experience is coaching. An instructor should follow a student's progress closely, remind the student if he/she is falling behind, and give detailed feedback about his/her current performance. The System for CLASS makes coaching more efficient for the instructor and more ubiquitous and effective for the student. An instructor can configure the system to send periodic reminders to the student about their progress on a term project. It can coach students about where they should be in the class readings so they do not fall behind. It automatically issues warning flags (cc'd
to his/her academic advisor) to the student who is in danger of failing the class. Generally, the system acts as the most responsible instructor who tracks each student's individual progress and devotes more attention to those who are falling behind.

4.5 Customizable Education Portal

We designed the entry point to the system as an information-packed portal. This allows us to present a consistent launching pad to other areas of the system while “pushing” important information so they are readily accessible from the top level. The portal is a page that our users would want to visit several times a day. Our users should be willing to use this page as a centralized management location for their academic (extensively) and (to a lesser degree but as much as possible) non-academic life. The portal imitates the features and benefits of My Yahoo:

- **Ease of use** - pieces of the portal can be added/deleted/moved around very easily, and a large user base has already gotten used to this interface.
- **Diversity of content** – in addition to standard features: stock quotes, current weather, my links, etc, the portal can be extended with new modules very easily.
- **Integration** - the portal works well in context of the rest of the education system. For example, the portal displays announcements made by all of his/her classes since a user’s last log in. It contains a calendar that displays the due dates for all assignments, projects, and exams that are relevant to the user.
- **Personalization** – the portal modules can be customized whenever possible.
- **Convenience** - everything is accessible within 2 clicks from the top-level portal page. This is very important, as people want to access information quickly and easily.

Some basic pre-packaged content for the portal includes stock quotes, current weather, and other system integrated information such as class calendar, bulletin boards, recent news, announcements, assignments, and projects. The portal will be as closely integrated with the system as possible, which means that a student will see a calendar that is
automatically populated with important dates from his/her classes and any relevant new announcements made since he/she last logged in.

4.6 Security and Permissions

Now that the system knows who the user is and has information about both users and classes, it needs to make sure that users are only allowed to see what they have permission to view. The system must also ensure that it is possible for users to privately create a document and then, when it is finished, make it public to all users. There are several ways that the system could do this. First, it could allow all users to see everything, allowing everyone to collaborate. This is obviously undesirable, as we do not want to let the public see the grades for every student nor let students see the solutions before the assignment is due. A second option is to limit information access to only authorized groups of users. A third option is to only allow users with a certain role or roles within certain groups to view the content.

CLASS uses the third approach. The roles for a class are professor, teaching assistant, or student. When the group is a department of a university, the roles can be administrator and member. Using this approach, the permissions become conceptually simple. For instance, the system can be set up so that only professors and teaching assistants can see the grades of students. In addition, this approach allows professors to decide on a class-by-class basis which groups of users with which roles should be able to perform certain tasks. There are many different cases where this feature is useful. For instance, one professor may want to allow the general public to download assignments while another professor does not let the general public see anything. Or a professor does not want teaching assistants to be able to upload assignments but allows them to see grades, whereas another professor lets teaching assistants upload assignments but not see the grades.
Chapter 5: Implementation

5.1 System Architecture and the ACS

5.1.1 ACS overview

The Arsdigita Community System (ACS) started as a set of tools Philip Greenspun wrote to build his website, http://photo.net. Later, the ACS was extended into a package of open source modules for building database-backed Web services that provide collaboration and user tracking tools quickly and easily. From the open source software came the company, Arsdigita, which continues to extend the open source community system with new releases about once a month. The community system provides a useful and flexible mechanism for Web services to be built quickly with a small group of 3-5 programmers without having 30 programmers labor for 8 months either building things from scratch or integrating existing solutions that may not be fully compatible or extensible. These qualities are especially important in the Internet market today where many startups are facing the challenges of time-to-market and limited resources to sell their business ideas.

5.1.2 The architecture

The ACS runs on the AOLserver, which is an application server developed by AOL now primarily used for their internal high-traffic Web hosting. The advantage of using AOLserver as an application server is that it provides a convenient set of APIs (written in C and tcl) that allow dynamic pages to communicate with the server and the database. More importantly, AOLserver proves a better choice over the traditional Common Gateway Interface (CGI) mechanism, because it layers directly on top of the database with a persistent connection to the database pools. Especially on high-traffic Websites, this results in drastic performance improvement over CGI, which must fork a separate database process to handle each request.

In the back end, an Oracle database hosts relational data that the websites run on. Oracle is the most robust and powerful relational database available in the market today. In addition to traditional SQL queries, Oracle provides a set of PL-SQL functions that run
within the database to quickly generate data that require more processing than SQL queries alone. Oracle 8 also allows Java to be executed directly within the database, thus providing even more power and flexibility for high demand data generation and manipulation.

A user accesses pages on our Web service by requesting a URL via his/her browser (the client) that is the address of a tcl page. (If necessary, we can easily circumvent this by serving an html URL that is really a dynamic tcl page) When a client requests a tcl page, AOLserver executes the tcl script, usually resulting in html text to be returned and displayed on the client browser. When the request also involves database access, AOLserver processes the appropriate commands from the tcl script, usually involving taking one of the persistent database handles and passing the appropriate SQL query to Oracle. Oracle processes the SQL query, running any necessary database procedures (e.g. PL-SQL) and returns the relevant dataset. The tcl script would then instruct AOLserver to extract data from the dataset into appropriate data structures so the script can process the information and display it to the client.

This 3-layer architecture is fast and reliable. The advantage of using tcl is twofold. Since AOLserver has a tcl version API, we can interface with the server and access the database conveniently. In addition, the advantage of using tcl for website development is that the language is interpreted and allows for fast execution and "tweaking" without the complications of debugging more complicated compiled languages like C or Java and the latency of compile time.

5.1.3 ACS modules

Each ACS module consists of two main parts: tcl scripts and sql data models. The tcl scripts include administration pages, user pages (these are often customized for individual sites), and a set of tcl procedures that are loaded onto the AOLserver at startup time and available as APIs for all dynamic pages in the system. The modules each occupy its own directory and have standard naming conventions so they are easy to integrate and extend. The second part of ACS consists of sql data models that can be loaded directly into the
Oracle database. These data models create tables to support the functionality of each module. For example, the user group module provides the capability to create and manage groups of users. The user group data model includes, among others, a table for customizable user group information and a table that maps users to groups, providing many-to-many users-to-groups mappings. The tcl portion of the user groups module contains administration pages to create new user groups, set the group’s admission policy, and add/delete members to each group. In addition, it provides tcl procedures for individual pages to require a user be logged on and check whether that user belongs to the appropriate group before granting him/her access to information that is private to the group.

Currently ACS modules range from organizing traditional Web communities and fostering online collaboration to hosting E-commerce services and tracking customer behavior. It has also recently expanded to include online polls and file system management and continues to grow with new modules in each release. The education system represents a new venture for the ACS from a general community/collaboration tool or the too specific E-commerce application into a system with new educational focus. The System for CLASS will harness existing tools in the ACS to create applications of the system’s current capabilities and extend it with more specific functionality that are unique to education, like the administration of online classes, new tools for academic collaboration, and mechanisms for effective Web-based teaching.

5.1.4 System organization and data model

The CLASS system is complex and contains many different pieces, including users, groups, and group-based roles. There are classes, subjects, departments, assignments, exams, and projects in addition to everything provided by the ArsDigita Community System (Figure 5.1). Departments can have multiple subjects and subjects can belong to multiple departments. A subject can have multiple classes but a class can only belong to one subject. The departments and classes are built on top of core ACS features. The system can be adopted at any level of the diagram. Finally, every class contains announcements, exams, projects, sections, and teams.
5.2 Administration and User Pages

The system is divided into the following sections: department/subject/class administration and department/class user pages. Administration pages provide functionality to perform tasks such as content and user management, setting class/department properties, and permissions configurations. User pages, on the other hand, display information that is available to general members of the department/class, or if the user pages setting is public, to anyone who visits the website. Parallelism between the administration and user pages is preserved whenever possible to create a consistency of functionality and user interface within a class or department.

The sections below present a more detailed look at the implementation of class administration and user pages. Department and subject pages are analogous to those of the class’s, where administration privileges are granted to users who have administrator roles for the respective department or subject. For implementation of security on administration and user pages, see Section 5.7, Security.
5.2.1 Class administration

Class administration pages are restricted to access by the course instructors, administrators, and TA’s. Class administration includes functionalities for tracking all users, which fall under four categories: professor, teaching assistant, student, or dropped. For further discussion about these categories see Section 5.3 User Management. From the administration pages, professors/TA’s can view all students (including those who dropped from the class) and see the evaluations of their performance on assignments/projects/exams. They can view students by the teams they are assigned to and/or by their recitation/tutorial groups. Upon viewing a student’s evaluations, a TA or instructor may also see what assignments the student has not been evaluated for and submit an evaluation for that assignment.

5.2.2 Student pages

From the class user pages, students can view contact information and pictures of other students in the class. They can also find out from the collaboration tools which students seem to be knowledgeable and willing to answer other students’ questions. Since all user data is managed via the ACS users and user groups modules, they easily integrate with our system’s collaboration, permissions, and file management features. The user interface design for student pages is similar to the design of systems like Blackboard.com and Command, as discussed in Chapter 3. The main feature set of the course homepage is divided into links listed on the left hand side of the course index page. These links include collaboration tools, course material, general course information, lists of students and student groups (e.g. project teams and recitations), and projects/assignments/exams. The grouping of the main feature set is intended to be intuitive to the student rather than to the system. As we have seen in some other systems, a shortcoming in the user interface is the grouping of links, such as lumping all class-related handouts under a link to Course Materials (WebCT) or grouping all solutions under the link to Solutions (Command). Such grouping may be reasonable to the system or data model, but is not intuitive to a student, who tends to categorize course material by the type of the task it requires of him/her. For example, task types include problem sets,
papers, quizzes, lecture notes, etc. Furthermore, students perceive solution handouts as associated with the assignment/exam they are for rather than as separate entities. As a result, the design strategy for user pages is to provide a grouping of links/feature sets that is natural to the student. Important material such as problem sets, lecture notes, exams, project teams, Q&A forums are never buried more than 1 level below the class index page.

Another issue we have noticed among existing education systems is the lack of use of the right hand side of the class index page, which is either blank (e.g. Command) or just shows a list of important announcements (e.g. Blackboard.com). None of the existing systems take the user-centric approach of “pushing” timely important information to the top level. How does the system know what “timely” information to push? Our guess is that the user would most likely be interested in the new announcements and handouts posted since his/her last login. Also, most students plan their academic schedule by the week, so they would also be interested in seeing any assignments/projects/exams that are coming up for the current week. Another interesting feature we implemented in our class index page is the display of one randomly selected member of class. The example in Figure 4.1 below features a picture of the professor and a link to his contact info. This feature gives the course homepage an up-to-date feel while “pushing” users to randomly meet fellow classmates online.
Figure 5.2: Course homepage with news and current week’s assignments/exams.

5.3 User Management

5.3.1 User groups

To solve the problem of managing what a user can do within a particular area of the system, we utilized the ACS’s User Groups module to model all users in our system. Many ACS modules that our system integrates support the concept of user groups. We can thus easily group users, manage them within the existing ACS user groups framework, and apply a consistent permissions/security system across all users within a group. The user groups concept allows us to manage users by leveraging the flexibility of a relational data model. It easily integrates implementations of the user groups data model with existing functionality in the ACS, all without sacrificing generality that should be preserved in a software module. Each user can be a member of one or more of the following groups: academic departments, classes, and teams. These user groups fall under a hierarchical relationship where a department contains one or more classes, and a class can consist of one or more student teams. There is no restriction on the relationship
between teams and departments, so members of a department can be grouped into teams if desired.

5.3.2 User roles

We do not assume that all classes/departments have standard ways of naming their instructors. For example, a person known to the system as having a "professor" role may not be a Professor to the University. Therefore, he should be more appropriately referred to on the class homepage as an instructor. The names of user roles are a customizable feature. From the department and class administration pages, users can specify what actual title to associate with each of the four roles defined in the system: professor, teaching assistant, student, and dropped. When a student drops the class, he remains in the system with an updated role from "student" to "dropped." All relevant information is kept in tact, but the role of "dropped" makes the student inactive most of the time unless the student is specifically requested to be included in an operation (for example, statistics of a quiz that was administered before drop date).

Another feature to ensure flexibility in handling user roles is an additional mapping table that defines information relevant to a particular role. For example, information we wish to ask about professors may include office hours, office room, phone number. A user who is a student, on the other hand, should be asked to input any applicable information such as student ID, student account name, etc. A graduate student who is both teaching and taking classes in our system will have two mapping table entries keyed by his user ID: one with information relevant to his role as a teaching assistant and the other relevant to his role as a student. Another benefit of the flexible mapping table is that the system can be easily extended with additional roles such as course administrator, lab assistant, and recitation instructor. If such distinction is desired, a system administrator can define additional roles via the subject administration pages.

5.3.3 Managing permissions

We can distinguish permission control for users within the same group by his/her particular role within that group. For example, the permission level for a TA should be a superset of the students' in that class plus extra permissions to perform TA duties.
Associated with each role is a relative permission level. This is used for features such as bulletin boards, where a bulletin board accessible by students is also accessible by the professor, because the professor role is defined to have a greater permission level than that of a student’s. This frees the system from making any assumptions about existing and newly added roles and instead operates solely on information captured by the data model.

5.4 Student Tasks

We model all exams, quizzes, assignments, and projects as the general concept of student tasks. This allows the system to be readily extensible with new task types if necessary (for example, lab experiments), and flexibility in interfacing all task types with the grading system. The grading system should not distinguish between assignments, exams, and projects. Instead, each task has an associated custom-defined grade type, the instructors decide the grade weighing of each task and automatically have the system clearly communicate the grading policy on the class homepage. Instead of creating a separate table for assignment, projects, and exams, we distinguish them by the task_type column in the table called edu_student_tasks. The tcl interface that displays/edits information on these tasks can therefore also be standardized.

Each task has a column grade_id that references the table edu_grades, which records information about the class’s grading policy. All rows in the edu_grades table under the same grade_id should have weights that add up to 100 (percent). So all tasks that are graded should associated with a grade_id that has a weight and a grade_name (e.g. 6.001 is 50% qualitative and 50% quantitative, where grade_name='qualitative' and weight=50). When we calculate a student’s grades, we join the student_tasks with their corresponding row from edu_student_evaluations by student_id along with the edu_grades table by grade_id. We then obtain a weighted sum of all grades from the edu_student_evaluations table with that student_id. The system does not yet support complicated grading formulas like dropping the lowest exam score or project grades that drop exponentially as a function of time past due. There is perhaps no simple solution to
supporting complex grading schemes. An intermediate solution is to allow users to disable the system’s built-in grade calculator and just provide an Excel-like spreadsheet to manage grades. On the other hand, simple statistical calculations such as mean, median, and standard deviation still prove useful to any class.

Students submit their solutions to the tasks online via the class pages. To verify that a student has submitted their solutions, we check for the existence of the row in edu_student_answers with the appropriate student_id and task_id. Furthermore, students may work in teams (modeled as user_groups) on any task, which we support by the column team_id in edu_student_answers. Although students in the same team submit collaborative solutions to a task, their grades are not shared by default. When the instructor or teaching assistant views the information page for a task from the administrative site, he/she also sees all students who have submitted answers to the task and have/have not been evaluated. The instructor/TA then has the option to evaluate that student (or group of students), and the grade is entered in the edu_student_evaluations table, keyed by the task_id and either student_id or team_id. After a student has been evaluated, the instructor can choose whether to make this evaluation viewable by the student. If viewable, the student can see his/her grades from the class homepage by clicking on that assignment. Otherwise, the student will only see that his/her submission has been evaluated.

5.5 The Portal

5.5.1 Customizability

Currently, the education portal contains the following set of pre-defined modules: stock quotes, current weather, calendar, class homepages and news/announcements. Additional modules can be added easily by defining the preloaded tcl procedure in the education-portal.tcl file and/or by implementing the ADP code via the portal module’s Web interface. We recognize that needs for the portal differ greatly not only among organizations but also among different users within the same organization. Therefore we should not make assumptions about what modules are most useful to the users. The portal
is designed with the idea that one or two programmers act as consultants to organizations that adopt our system and help them customize the system if necessary. For example, if adopted for the MIT Sloan School of Management, the portal can provide sets of pre-defined modules that suit the different needs of Sloan staff, students, and alumni. This can be done with some lightweight tcl/adp programming via the portal’s web-based interface. The advantage of our portal system over other closed-source systems like WebCT is that it can be fully customized and extended to fit the needs of any individual, organization, or university that implements our system.

5.5.2 Modifications to the ACS portal module

The education portal is built on top of the ACS portal module, with several significant modifications. Since our portal imitates My Yahoo in having modules with information scraped from remote sites (e.g. stocks, weather), it is necessary to implement row-by-row memoization. Each row in the portal sql tables is indexed by user id and identifies the information that needs to be scraped from the remote sites. For example, in the portal_stocks table each row containing the columns, user_id, stock_symbol translates to one ns_geturl request sent to


5.5.3 Performance tuning

Scraping from remote websites is slow, so we optimize the portal’s loading time by memoizing the unique ns_geturl request for a specified length of time. The next time our system runs ns_geturl with the same url, the memoized html page is returned instead of fetching the page again from quotes.nasdaq-amex.com. For stock quotes the memoization timeout is set to 20 minutes (since most quotes are 20-minute delayed). For more stable information such as current weather, the timeout is set to one hour. Currently only table modules that depend on data from external sites implement row-by-row memoization. If the portal page needs to be further optimized, additional portal modules can implement memoization. However, performance in gathering local data such as class announcements and calendar may be bottlenecked by the complex database queries joining class and user information. Therefore, optimization may involve tuning these SQL queries, for example, by indexing the user_groups table. Query performance tuning becomes increasingly
important as the size of the edu_classes table grows, especially for a large department or university with many users and classes.

5.5.4 The calendar

The portal calendar is an integrated version based on the ACS calendar module. Updates to the original calendar include support for both user_group and individual user-based calendar entries. This enables the calendar to display not only class-related events/due dates gathered from class database tables but also personal events from user input. The current calendar displays class events/due dates with links directly to the information page about that particular assignment/quiz/news announcement. This feature acts as a convenient one-click entry point to the course homepage. In addition, since people think of their tasks as organized by days and weeks rather than by classes, our calendar system integrates dates directly with appropriate material from the class websites. For the instructors, the calendar is a convenient planning tool, where he/she can click on the day of the calendar, enter a new assignment/task due on that day or schedule a meeting for that day. Additional interface is being built to allow a user with sufficient permissions (e.g. a professor) to view all entries of members in the class that are designated as non-private. This is particularly useful for the instructor who is planning an exam review or a last-minute staff meeting.

5.6 Collaboration Tools

We have tied several ACS collaboration/communication tools to the education system. These include a bulletin board in the form of threaded discussions, a real-time chat program, and the email module that supports spamming all users in the class/department or a subset of users within those groups. The ACS bulletin board module had to be rewritten to recognize users based on their roles and support having bulletin boards associated at the user group level. This allows a class to create one or more of the following: a course Q&A forum for students, a staff-only forum for the professors and TA’s, team forums for each of the teams, and recitation bulletin boards for the recitation sections.
The Group Spamming module provides mailing list functionality while eliminating the hassle of setting up and maintaining separate mailing lists. Like the bulletin board module, group spamming was also updated to with group-level, role based spamming to support sending email to all students in a recitation section, for example, or emailing all TA’s in the class. Adding more collaboration tools to the system is fairly easy provided that they interface with our role-based user/user groups schema.

5.7 Security

The System for CLASS contains a large amount of sensitive information such as personal information about every user, student grades, and sensitive handouts such as exams before they are administered. Therefore, we have gone through great measures to verify that the user is who they say they are in addition to keeping sensitive information out of the hands of people who should not see it. Generally we need to verify upon each user request on a department or class page 1) the user’s identity 2) the group_id (i.e. class or department) the request is for 3) whether the user is a member of the group and 4) if so, whether the user’s role within the group has permission to perform the request. Each class has an associated permissions role/action matrix. Each entry in the matrix is configurable from the administration pages and specifies whether a member with that role has permission to perform the action in question. Given the role/action table, two separate procedures perform security checks for administration and user pages. Viewing a user page requires that the user be a member of the class (unless the class homepage is designated as public, in which case anyone can view the page). Requests to perform additional actions within the user pages (e.g. submitting answers to an assignment) passes the specific action into the security procedure, which looks up the class permissions table and returns whether or not the user has permission to perform the action. Security checks on the administration pages are analogous to those for the user pages.
5.8 File Storage System

The file system we have implemented serves as a central place for the development, distribution, and submission of assignments. It ties in the ACS file system with updates to support our system’s specific requirements. Existing features rolled over from the ACS file system include file versioning, repository, and management of uploads and downloads. One feature that was lacking in the ACS file system was the ability to control permissions at the user group level. Since our system was based heavily on user groups, an upgrade to the file system for group permissions was necessary. In addition to controlling permissions by user_id (primary key in the users table), the file system needed to support permission control across groups of users (by group_id – primary key in the user_groups table). By doing so, we can specify file permission not only for particular classes, but also within the same class distinguished by the users’ roles. For example, the 6.001 Quiz 2 file have the following permissions associated with it: read permission for all 6.001 students, read/write permission for all 6.001 TA’s, and read/write/delete permission for all 6.001 professors.

5.9 Auditing Database Tables

In order to track the changes in important database tables, the updates/deletes to those tables must be audited so they can be recovered if necessary. Auditing is done by adding an audit table that keeps the “older” versions of data from their corresponding tables. Oracle automatically inserts data into audit tables upon update/deletion of the row from the corresponding tables via audit triggers (see data model in Appendix Section B). Currently we audit important configuration information about departments, subjects, and classes, in addition to grades information from tables like edu_grades, edu_student_answers, and edu_student_evaluations.
Chapter 6: Feedback

During the course of development, two separate classes tested the CLASS system and provided valuable feedback. The first class, 6.916, Software Engineering of Innovative Web Services, used an early version of the system to facilitate collaboration and grading for the course. The second class, ECAP: E-commerce Architecture Project, started out with the same system but received periodic upgrades as development progressed.

6.1 Software Engineering of Innovative Web Services

The feedback we received from 6.916 was mostly positive. The main features used by the instructors were the Bulk Spamming module with archiving and collaborative grading. Instructors found the system intuitive to use and appreciated the ease of managed collaborative grading. The only other feature they used was project administration, which received negative reviews at first. However, as the semester progressed and the functionality improved, the instructors started liking the system. The chief complaint initially was that the workflow for creating project groups was too long and unintuitive. Initially, it required five page views to create a project instance and then another two page views for every individual that was added to the team. To remedy this, we removed two of the pages from the workflow for creating a project. In addition, we allowed students to sign up for project teams via the website, thereby removing the burden of team creation from the instructors.

The problems with creation and management of project teams taught us two things. First, the number of page views required to accomplish a task should be proportional to the size and significance of the task itself. Users do not want to wait for five separate pages to load for every project team they add. The problem becomes especially cumbersome when the task is repetitive, such as creating 10 project teams for a class. We also learned that our data model should equate project instances with teams instead of associating them via a mapping table. We found that making the projects themselves teams greatly reduced the
complexity of both the data entry page flow and the display of project information. In addition, this change simplified the front-end interface for students to sign up for term projects. We initially associated teams with projects via mapping tables based on the assumption that teams exhibit a one-to-one mapping to projects and assignments. However, from 6.916 feedback, we discovered that in addition to teams, we needed to create an additional entity, which we referred to as project instances. In addition to modeling group final projects with user groups, we also needed to consider special cases where teams start on separate projects and join efforts later or when small teams work on several projects with other teams. To model this, we could either have a one to many project to team map or we could have project instances of user_group type that could contain either users or other user groups. Although both alternatives would have worked, we chose the latter because a shared user_groups data model simplified both the page flow and the system code.

6.2 E-commerce Architecture Project

In contrast to 6.916, the feedback we received from the E-commerce Architecture Project was mostly negative until the end of the semester. This class was a small, discussion-oriented seminar that did not have problem sets or exams. Instead, students wrote a paper every other week and meet with the instructor periodically to discuss past papers and progress on their current paper. Students used the site mainly to track feedback on their papers and download new assignments and handouts. The teaching assistant used the site for distribution of materials and management/evaluation of students in the class. The instructor, who only used the site sporadically, gave us no feedback. Whenever he needed something done with the site, he would ask the teaching assistant to do it.

ECAP provided us with valuable feedback, which helped shape CLASS into a more well-rounded system. The first thing we learned was that our original design was heavily biased towards technical classes with weekly problem sets and not broad enough to support small, seminar-based classes. Not only did the site fail to foster collaboration as well as we had hoped, some of the core features desired by ECAP were also missing. For
instance, the original system did not provide a mechanism for students to sign up for office hours online. For large technical classes, posting the time and location for office hours is enough, because the number of students that would show up makes scheduling individual appointments unpractical. However, since this class was small, the professor allowed students to schedule thirty-minute appointments with him. Another feature that the class wanted was the ability for students to post comments about projects done by other students. The twist was that the owner of the project should see the comments as anonymous, the teaching assistant and professor should see all comments, while all students should only see their own comment or those relating to their project. Although we never entertained this scenario in our design, it seemed reasonable when it was brought up.

The second lesson we immediately learned from this class was that users who are not familiar with computers or do not speak English well had a difficult time navigating the site. We learned this from students who fit one of the categories and the teaching assistant, who fell into both categories. The main problem was that the functionality provided by the links was not clearly communicated on the user pages. For instance, the teaching assistant would add an assignment and upload a corresponding file. Then, to change the file, she would go into the file system and get lost instead of going to the assignment info page from the course website and editing the file from there. She got lost in the file storage system because it contained permissions, file type, time stamps, and versioning information, all of which was previously hidden to her. The teaching assistant repeatedly made this same mistake for assignments, lecture notes, and handouts. As a result, one of the first modifications we made was removing the link to the file storage system. Once this was done, she was able to find the appropriate assignment and edit it with no problem. Another area that confused both the teaching assistant and the instructor was the use of student teams. This observation was consistent with the feedback we received from 6.916, and the remedy was the same.

One final thing the teaching assistant requested was for all grades for an assignment to be initially hidden to the students. When all students were graded for that assignment, she
should be able to reveal all grades with one press of a button. We actually included this feature in our original design but did not get around to implementing it. From this feedback, we realized that this feature was useful and should enjoy higher priority.

In contrast, we did not receive much informative feedback from the students. The only strong opinion they expressed was easier access to their grades. The original design only provided links to grades from the individual assignment’s information pages. However, students wanted the ability to view all of their grades from one single page that also showed statistics such as class distribution and averages.

6.3 Lessons Learned

As we made improvements, user feedback steadily became more positive. Changes we made based on feedback included adding online scheduling of office hours, removing the link to the raw file storage system, and rewording much of the text on user pages. In addition, we made extensive modifications to the workflow for adding teams and projects. Finally, we removed the requirement that all assignments have to associate with an uploaded file.

Having two drastically different pilot classes proved extremely valuable. First, the feedback has taught us that simply making the system user-centric does not guarantee usability or flexibility. In addition to being user-centric, the system must be general enough to meet the needs of most users and classes while specific enough to be user-friendly and intuitive. Although achieving the right balance is difficult, it should nevertheless be an important design consideration. Secondly, since different classes are structured differently, they do not all fit nicely into a general mold. Therefore, our system must be versatile to serve 90% of the classes and extensible to suit the remaining 10%. We learned that getting feedback early and often from real users is a good way to keep the design from being too narrowly focused and biased towards a particular type of class. Finally, we learned that simplicity is best. If providing a really neat feature that is only somewhat useful greatly complicates the user interface, then it is best to not include the
feature at all. We found a good example of this problem by exposing the end user to the raw file storage system.
Chapter 7: Accomplishments

7.1 Competitive Analysis

Many online education systems have been created before this one, and many more have yet to come. How does our system compare in functionality and usability to other systems in its market space? Appendix A provides side-by-side comparison of our system's functionality to that of fourteen commercial systems. As seen from the comparison, CLASS displays a significant portion of the functionality provided by other commercial systems. Commercial education systems have to present the most diverse set of features possible to capture the largest customer base. However, due to time constraints, our goal was to create a system with a core set of functionalities and flexible enough to be later extended via open source or in-house development. Much of the future work is detailed later in Section 8.1.

7.2 Our Accomplishments

We have built a user-centric system designed to suit the users' individual needs rather than lumping all tools together for each class. In addition, CLASS helps reduce site-wide administrative tasks by delegating the work down to the department and class level. This divide and conquer approach reduces the possibly for error while shortening the system's response time. To request a class on the Command system, the instructor or TA submits a form to the site-wide administrator, who verifies the information is correct and approves it. This process may not be a big deal if the approval only involves someone going to a page and pressing a button. However, if a single user had to do this for hundreds of classes during a one or two-day period, it would become an extremely cumbersome task. The CLASS system, on the other hand, delegates the responsibility to create new classes to administrators of individual departments. For the creation of 300 classes over six departments, the work is delegated to the administrative staff of six separate departments as opposed to a single site-wide administrator. Once a professor has taught a class, they
also have the authority to create a new instance of the class for the next semester. After several semesters, administrative burden gradually trickles down the organizational hierarchy, reducing the tasks required of the high level administrators.

Whereas CLASS only displays a link if it leads to useful data, other software packages like Command show links that often lead the user to a dead end. Finally, CLASS is built on top of the ACS, a software package that primarily focuses on the creation and management of online communities. As a result, it fosters a sense of community in unique ways. For example, the class homepage displays the name and picture of a randomly selected student or faculty member, therefore "forcing" people to meet each other in the class. When a student or faculty member looks up information about another user, in addition to name, email, and portrait, they will also find all contributions the user has ever made as a community member. Most other commercial products do not offer this functionality.

The portal page is one feature that CLASS does better than any of the systems reviewed in Chapter 3. The education portal provides features such as an integrated calendar, all new announcements since the user’s last login, and direct access to class pages, bulletin boards, and chat rooms. None of the existing education systems provide as much personalized information on a single page; instead, most force the user to visit all of the class homepages to retrieve this information. The selection of portal tables can be expanded or restricted by the site-wide administrator. Therefore, if the system were to add a streaming video section or a real time white board, the administrator could elect to make that information available on the portal page. Maximum flexibility is achieved by shifting the bulk of content decision from administrators to end-users. The trade-off is a possibly more complicated user interface and the danger of distracting users too often with decision-making. Again, we have chosen to balance this trade-off using a user-centric approach. Leveraging knowledge of the user’s identity and role within the user group, the system only "pushes" relevant information, resulting in a customized and less clobbered user interface.
7.3 **Use by Universities**

No Web service is considered successful unless it has users. With that said, the success of any Web service is evaluated by the quality and quantity of its user experience. The initial feature set of our education system has initially been tailored for MIT’s Sloan School of Management, which has requested a highly customizable and collaborative educational environment to replace their existing system, Virtual Campus. Sloan’s basic requirements include course websites to supplement traditional lectures, a personalized portal page, and a centralized document repository to manage the School’s publications and course material, which include documents as well as media and image files. In addition to the Sloan School, Dartmouth’s Tuck School of Management, Olin University, and the University of San Francisco have also expressed interest in using CLASS to enhance the student learning experience. Finally, the ArsDigita University will also be using this software to administer all of its classes. MIT Sloan, Tuck School of Management, and ArsDigita University plan to adopt our software for the Fall 2000 semester.
Chapter 8: Conclusions

After examining many commercial educational packages and then designing and building our own, two points of interest remain: additional work required to turn this system into a robust enterprise solution suitable for major universities mentioned in section 6.3 and the lessons we learned from building a system of this size.

8.1 Future Work

The future work described here gives an overview of a few of the many enhancements needed to convert this educational system into a world-class enterprise solution. Some of the proposed changes, such as online testing, personalization, and the data mining will require major additions to the current system and may prove to be technically challenging. Other changes, such as providing the ability for an instructor to customize text or reveal all student grades at once will not require fundamental changes or additions to the current system but will bring the CLASS system closer to its goal of being highly flexible and usable.

8.1.1 Enhanced security

The system currently relies on the user logging in by providing their email and a password. This password is then encrypted and checked against the encrypted password also stored in the database. If the passwords match, the user’s cookie is set. While this method, when used with the other measures, is secure, it does not guarantee authentication. The current implementation does not prevent a person from registering as someone else and gaining access to private information. The use of client certificates or some other authentication process with trusted root does guarantee that users are who they claim they are. Therefore, using digital certificates is a better way to handle security when they are feasible. Many large universities already use digital certificates so asking the members of the university to use them for the site will not be a problem. However, when universities want to invite alumni and the general public who may not be familiar
with certificates, the issue becomes more complicated. Certificates are not as desirable when users are not familiar with them because obtaining them may be enough of a deterrent for many users to never use the site. Therefore, a future enhancement will be to incorporate the use of digital certificates in a way that the administrator of the system can determine whether or not they should be used.

8.1.2 Increased functionality for departments

The current system uses departments as a means to group subjects together. In reality, however, departments are much more complicated than just a group of subjects. One place to significantly improve the functionality of departments is to add more roles within the user group. Currently, the system only provides two distinct roles, administrator and member. Instead of using these roles, the system should try to model a real department by adding roles for department heads, secretaries, research scientists, tenured and non-tenured faculty, undergraduate and graduate students majoring in the department, and students taking classes offered by the department. This new hierarchy will allow the system to provide significant new capabilities to all members of the department. Adding roles will allow users to send email to all users of any given role. In addition, each role will be able to have their own bulletin boards and chat rooms.

Finally, and most importantly, adding new roles to departments will allow tighter security within the system. Currently, all members of a department can view information about all classes in the department. In addition, they all have permission to edit properties of the department and add new subjects. These are actions that can easily be made more secure by specifying more roles in the department user groups. The one drawback to adding more roles is that it increases the complexity of system management. And, as we learned from the E-commerce Architecture Project class, more complexity can lead to a less user-friendly system. Therefore, when the new roles are added, we must be careful to strike a balance between functionality and usability.

8.1.3 Increased coaching and collaboration

CLASS was built with the goal of providing an easy to use collaborative environment to complement physical classes. However, only a limited set of collaborative tools has
actually been incorporated into the system. There are many more tools that could be
integrated or developed to enhance this system's collaboration capabilities.

The ACS has many different tools that the system has not even begun to tap into. For
instance, community members should be able to comment on exams or projects. If an
instructor has an addendum to an assignment or students have questions, the related
discussions should be located directly on the assignment information page. In addition,
the system should be tied into the ACS wimpy point module, which allows users to
collaboratively create presentations. Finally, the system should be integrated with the
ACS coaching module so that students can receive automatic email reminders regarding
problem sets or exams.

To aid the instructors in coaching, the system should take advantage of the data it has to
provide the professor with useful information. For instance, the system should tell the
professor which students are not doing well or that students did not understand a
particular section of a lecture. Defining and implementing these new features for the
teaching staff will prove challenging but will also provide large benefits to the end user.

In addition to the ACS-related tools and data mining, CLASS should also include a real-
time whiteboard so that students and faculty alike can communicate with diagrams as
well as chat. This would greatly enhance online office hours as students and instructors
can easily convey equations to each other. And, if students are allowed to take "snap
shots" of the white board, they will have an instant record of exactly what the instructor
wrote on the board. In some instances, this could be a significant improvement over
having the student frantically copy the drawing instead of trying to comprehend the
material.

8.1.4 Reports

CLASS provides new capabilities to the class instructors by providing detailed
information about each user. However, it does not currently provide department
administrators or other faculty with similar capabilities. For instance, there is no easy
way for a department administrator to see an enrollment list for the current semester's classes. In addition, department administrators should be able to view the history of a subject, when classes within the subject have been offered, who taught the class, and how many students took it. In addition, the system should generate system-wide reports for each student. These reports are useful for academic advisors who would like to see how a particular student is performing. These are just a few examples of the many ways useful data could be extracted from the database and displayed to the user in an attempt to make the site more useful.

8.1.5 Usability

As mentioned previously, people are not going to use the system unless it is extremely easy to use and it makes their life easier. Usability testing on study groups can greatly improve the system’s usability. In addition, more classes using the system will lead to significant user interface changes.

In addition to the features we already incorporated based on feedback, the system could also add several features that would make the site much more convenient to use. To complement the reports mentioned in section 8.1.4, the system should provide the ability for administrators to batch upload class-related material. In addition, it would be convenient for instructors if they could easily download student grades into a spreadsheet. Students should be allowed to make inline comments when they are reading articles or other material uploaded by the professor. And, for all of the users, the functionality of the portal site would be greatly increased if the number and variety of tables were increased. For instance, the portal page should also included information such as recent news, movies, and sports.

8.1.6 Templating

Currently, all of the class home pages look identical. The only way to tell them apart is to look at the page title where it says the name of the class. This is an undesirable situation since the needs of every class are different. For some classes, the default setup will work well. However, other classes will want to add graphics or have certain categories they would like to display.
The easiest way to accommodate the needs of creative professors is to place of the files into templates defaulting to the current homepage setup. This allows professors to change the look and feel of their course homepage without affecting the other homepages in the system. A templating mechanism would also allow departments to give all of their pages a unified look and feel so that it will be obvious to the user which course or department they are currently viewing.

8.1.7 Online testing

The CLASS system should include an online testing and self-evaluation package. This would be a significant new capability for students by allowing TAs to create practice tests for students to take online. Students could then use the results for self-evaluation purposes. In addition, online tests should include questions that are essay, short answer, multiple choice, true/false, multiple select, fill in the blank or any combination of those types. All of the questions except for the essay and short answer could easily be graded automatically.

8.1.8 Increased educational functionality

The CLASS system should include a lot more functionality to support explicit educational use. The system described by this paper only implements a few of the many, many different features required to make this a highly usable and flexible system. For instance, the system should employ data mining techniques to help in course management; instructors should have the option of grading tasks on a per problem basis so they can tell that most students did poorly on problem two of assignment six. Another example of data mining is to provide the professor with an automatically generated list of students doing poorly in class or a list of students who did not do well on a particular lesson within the class.

Other improvements include increased support for creating content, better modularity, and interoperability. The system should provide instructors support for creating online tests, assignments, or other course material. In addition, the system should be modular
enough so that programmers wishing to adopt it can easily write their own module and integrate them as “plug-ins.” Finally, it should provide support for syncing the portal calendar with the user’s palm pilot or another portal such as MyYahoo.

8.2 Initial System Complete

Although many online educational resources are available today, not many of these are free and open source, while even fewer are user-centric. This project has successfully met the goal of creating a system that sets itself apart from the existing set of solutions for online education.

Most of the high level goals of the system have been met. We have shown that it is possible to build a user-centric, collaborative system capable of complementing a class and integrate it with a set of collaborative tools. We have provided important information to users with minimal number of page loads and decentralized administrative responsibility onto the individual department and class administrators. However, one goal we have not met is creating strong security. Rather than spending time on implementing client certificates in AOLserver, we chose to spend our time adding core features to the system. Not achieving this goal is not a major setback because stopgap measures could be taken to reach this goal. For instance, the system could run mod_AOLserver within Apache, which provides client certificates for authentication.

The goal of fostering collaboration within a user-friendly system is much harder to evaluate, as success is difficult to measure without subjecting the system to widespread use. The system has the potential to foster collaboration through the many collaborative tools it provides. Usability of the system, however, still requires a great deal of work. As discussed in Section 8.1, many features that could be added to enhance the user’s experience. As discussed in Chapter 7, a pilot class at MIT’s Sloan School of Management found some areas of the system intuitive and some cryptic. While we have tried to explain the cryptic sections more clearly, there is no way to reliably test the effectiveness of our user interface until the system finds a large user base.
Despite these shortcomings and the long list of future work, the project is a success at creating a user-centric, open source system capable of supporting a large set of academic departments and classes. We have also found that anyone designing a user-centric system in the future must make sure that every feature of the system is built upon a skeletal backbone of the user and user_groups data model. If this property is not preserved, the system immediately shifts towards a class-centric design, and a significant amount of user capabilities are lost. This was most apparent when our initial implementation of class projects were mapped to the class rather than modeled as user groups. We had great difficulty creating a user-friendly workflow with this model. Once we modeled project teams as groups, the workflow became naturally improved.

Although significant amount of work is required to turn this into a highly usable and flexible system, future development can build upon the solid user-centric base we have designed and developed. We hope to complete many additional desired features within the coming months so that schools such as MIT's Sloan School of Management, Dartmouth's Tuck School of Management, and ArsDigita University can adopt the System for CLASS this fall.
Appendices

A. Competitive Analysis Tables

This appendix reviews a series of software systems, including the one created for this thesis. The number corresponding to each system below represents the numbers for the column headings found in each of the following tables. A column noted with a "Y" represents a feature that the product possesses. Please note that just because two or more systems possess the same feature does not mean that those features function in the exact same manner. Appendix B has been broken up into 6 tables: Developmental Features, Instructor Tools, Instructional Features, Student Tools, Technical Support, Administrator Tools. With the exception of columns 1 and 12, most of this information can be found in a study conducted at Marshall University [Com00]. Reprinted with permission.

1. System described in thesis
2. Blackboard
3. Convene
4. Embanet
5. eCollege.com
6. IntraLearn
7. TopClass
8. WebCT
9. Web Course In A Box
10. Integrated Virtual Learning Environment (IVLE)
11. LUVIT
12. Serf
13. Virtual-U
14. Eduprise.com
15. Command

A.1 Developmental Features

<table>
<thead>
<tr>
<th>DEVELOPMENTAL FEATURES</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content format will allow for simple transfer to/from another vendor's platform</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Platform uses open data standard so that it can communicate with existing university database applications</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Content can be authored on PCs running Windows 95/98/NT</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Content can be authored on</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Feature</td>
<td>PC</td>
<td>Mac</td>
<td>DEC Unix</td>
<td>Windows NT</td>
<td>IMS</td>
<td>AICC</td>
<td>HTML</td>
<td>Content Creation</td>
<td>Server Software</td>
<td>Authoring Software</td>
<td>Reporting Features</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>----</td>
<td>-----</td>
<td>----------</td>
<td>------------</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>------------------</td>
<td>------------------</td>
<td>--------------------</td>
<td>-------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Courses can be taken using a PC running Windows 95/98/NT</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Courses can be taken using a Macintosh running OS 7.5 or greater</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform provider is supportive of implementing IMS standard within product</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform provider is supportive of implementing AICC standards within product</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform utilizes standard HTML for content creation</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform is structured so students can view all of their current courses when they log on</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform's server software will run on DEC Unix</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform's server software will run on Windows NT</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple choice questions can be created/scored with platform's authoring software</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True/False questions can be created/scored with platform's authoring software</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matching questions can be created/scored with platform's authoring software</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short answer questions can be created/scored with platform's authoring software</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Essay questions can be created/scored with platform's authoring software</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Essay database for management of test questions</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platforms supports reporting features for test questions</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform supports Microsoft</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-----</td>
<td>----</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internet Explorer 4.x and newer browsers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform supports testing stage for courses to debugged before</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>making them live to students</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform allows author to view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>course as student without</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>logging out</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform has built-in threaded</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>discussion list capabilities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform has built-in chat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>capabilities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform can be integrated with</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real networks video and audio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>products</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform can be integrated with</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Macromedia Shockwave products</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vendor provides development services</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management component will create reports for tracking student progress</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform has a feature to import existing test questions in a tab-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>delimited format</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### A.2 Instructor Tools

<table>
<thead>
<tr>
<th>INSTRUCTOR TOOLS</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course planning</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Course managing</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Fast course revising</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Course monitoring</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Instructional designing</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Presenting information</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>On-line testing</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>On-line grading</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Managing records</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>No HTML knowledge required</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Customization of student curriculum</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Student tracking</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Automated grading</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Level of control over design</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Instructor can assign specific course material to individual or group of students</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Multiple choice self test tutorial questions - (automatic marking)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>&quot;Fill in the blank&quot; self test tutorial questions - (automatic marking)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Customized feedback to tutorial questions</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Redirect path of tutorial depending on question answers</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Timed quizzes (graded with permanent mark retention)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
### A.3 Instructional Features

<table>
<thead>
<tr>
<th>INSTRUCTIONAL FEATURES</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faculty to student asynchronous communication is possible</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Faculty to student synchronous communication is possible</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Faculty can make their own changes to content</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Training is provided for faculty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Courses can have consistent interface</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Platform includes an internal e-mail client</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Platform has e-mail management capabilities for students</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform has e-mail management capabilities for faculty</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform supports multiple instructors for a single course</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
## A.4 Student Tools

<table>
<thead>
<tr>
<th>STUDENT TOOLS</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bookmark management</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Multimedia support</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Private e-mail</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>File submissions</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Threaded discussions</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Course Chat rooms</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Logged chat</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Whiteboard</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Self-assessing</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Progress tracking</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Desktop based file management for</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>uploading to server</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Study skill building</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Un-timed quizzes</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>One question-at-a-time function</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bulletin board/conferencing tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Image database</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Student access to own grades</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Access to course grade distribution</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Automated glossary tool</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Automated index tool</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Online assistance</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Search tool for course content</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Student presentations area</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

- Allows user to view all class events on one consolidated page once logged in.
- Student can make private annotation of material.
### A.5 Administrator Tools

<table>
<thead>
<tr>
<th>ADMINISTRATOR TOOLS</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Client/Web interface</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Authorization tools</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Logout feature</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Resource monitoring</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Remote access tools</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Crash recovery tools</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Student support tools</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Instructor support tools</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Administrator support tools</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Built-in file management tools</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Ability to export raw data</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Customization of text messages</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Resume session function</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Security access</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Variable level of security</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Online registration</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Registered markers</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Batch upload to register students</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Guest account creation</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Instructors can create groups of students</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

### B. Data Model

--
-- /www/doc/sql/education.sql
-- by randyg@mit.edu and aileen@mit.edu on September 29, 1999
-- with much help from philg@arsdigita.com on October 28, 1999
-- the system is centered around the classes

---

72
-- as it is too difficult to have 2 types of queries to support
-- individual users

-- instead of having a classes table, we just define a user group
-- type of "edu_class"

-- for the class name we'll use "group_name" (a default field from
-- the user_groups table); everything else will have to be in
-- edu_class_info; this is a bit tricky since we need to
-- keep the definitions for the helper edu_classes_info table in
-- sync with what we insert into user_group_type_fields (used
-- to generate UI)

-- we don't store much contact info in the classes table; if we need
-- to send out a report on system usage, we send it to all the people
-- with the admin role in this user group

-- this table holds the terms for the classes (e.g. Fall 1999)

create sequence edu_term_id_sequence start with 1;

create table edu_terms (    term_id integer not null primary key,
    term_name varchar(100) not null,
    start_date date not null,    end_date date not null
);

-- we want the above table to automatically start with a term that
extends over all time
-- (or at least 100 years) for classes that people take at their own
pace

insert into edu_terms (term_id, term_name, start_date, end_date)
select edu_term_id_sequence.nextval, 'No Term', sysdate,
add_months(sysdate,1200)
from dual
where 0 = (select count(*) from edu_terms);

-- for a multi-department university, we need to this to sort courses
-- by department; we're going to want private discussion groups, etc.
-- for people who work in departments, so we make this a user group

-- to find the department head and other big staffers, we look at
people with
-- particular roles in the user_group_map

create table edu_department_info (    group_id integer primary key references user_groups,
    department_number varchar(100),
    -- we'll generate a home page for them but if they have one
    -- we can provide a link
    external_homepage_url varchar(200),
    mailing_address varchar(200),
)

73
-- we want to audit the department information

cREATE TABLE edu_department_info_audit (  
group_id   integer,  
department_number varchar(100),  
external_homepage_url varchar(200),  
mailing_address varchar(200),  
phone_number varchar(20),  
fax_number varchar(20),  
inquiry_email varchar(50),  
description clob,  
mission_statement clob,  
last_modified date,  
last_modifying_user integer,  
modified_ip_address varchar(20)  
);

-- we create a trigger to keep the audit table current

CREATE OR REPLACE TRIGGER edu_department_info_audit_tr  
before update or delete on edu_department_info  
for each row  
begin  
insert into edu_department_info_audit (  
group_id,  
department_number,  
external_homepage_url,  
mailing_address,  
phone_number,  
fax_number,  
inquiry_email,  
description,  
mission_statement,  
last_modified,  
last_modifying_user,  
modified_ip_address)  
values (  
:old.group_id,  
:old.department_number,  
:old.external_homepage_url,  
:old.mailing_address,  
:old.phone_number,  
:old.fax_number,  
:old.inquiry_email,  
:old.description,
-- now, lets create a group of type department and insert all of
-- the necessary rows to generate the user interface on the /admin

declare
  n_departments_group_types integer;
begin
  select count(*) into n_departments_group_types from user_grouptypes
  where group_type = 'edu_department';
  if n_departments_group_types = 0 then
    insert into user_grouptypes
      (group_type, pretty_name, pretty_plural, approval_policy,
       default_new_member_policy, group_module_administration)
    values
      ('edu_department', 'Department', 'Departments', 'wait', 'open', 'full');
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'department_number', 'Department Number', 'text',
       'varchar(100)', '', 1);
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'external_homepage_url', 'External Homepage URL',
       'text', 'varchar(200)', '', 2);
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'mailing_address', 'Mailing Address', 'text',
       'varchar(200)', '', 3);
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'phone_number', 'Phone Number', 'text',
       'varchar(20)', '', 4);
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'fax_number', 'Fax Number', 'text',
       'varchar(20)', '', 5);
  end if;
end;
/
show errors

-- now, lets create a group of type department and insert all of
-- the necessary rows to generate the user interface on the /admin

declare
  n_departments_group_types integer;
begin
  select count(*) into n_departments_group_types from user_grouptypes
  where group_type = 'edu_department';
  if n_departments_group_types = 0 then
    insert into user_grouptypes
      (group_type, pretty_name, pretty_plural, approval_policy,
       default_new_member_policy, group_module_administration)
    values
      ('edu_department', 'Department', 'Departments', 'wait', 'open', 'full');
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'department_number', 'Department Number', 'text',
       'varchar(100)', '', 1);
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'external_homepage_url', 'External Homepage URL',
       'text', 'varchar(200)', '', 2);
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'mailing_address', 'Mailing Address', 'text',
       'varchar(200)', '', 3);
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'phone_number', 'Phone Number', 'text',
       'varchar(20)', '', 4);
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
    values
      ('edu_department', 'fax_number', 'Fax Number', 'text',
       'varchar(20)', '', 5);
    insert into user_group_type_fields (group_type, column_name,
      pretty_name, column_type, column_actual_type, columnextra, sort_key)
values
    ('edu_department', 'inquiry_email', 'Inquiry Email', 'text',
    'varchar(50)', '', 6);

insert into user_group_type_fields (group_type, column_name,
    pretty_name, column_type, column_actual_type, column_extra, sort_key)
    values
    ('edu_department', 'description', 'Description', 'text', 'clob', '', 7);

insert into user_group_type_fields (group_type, column_name,
    pretty_name, column_type, column_actual_type, column_extra, sort_key)
    values
    ('edu_department', 'mission_statement', 'Mission Statement', 'text',
    'clob', '', 8);

end if;
end;
/

-- now we want to create a view to easily select departments
create or replace view edu_departments
as
    select
        user_groups.group_id as department_id,
        group_name as department_name,
        department_number,
        external_homepage_url,
        mailing_address,
        phone_number,
        fax_number,
        inquiry_email,
        description,
        mission_statement
    from user_groups, edu_department_info
    where user_groups.group_id = edu_department_info.group_id
    and group_type = 'edu_department'
    and active_p = 't'
    and approved_p = 't';

-- we model the subjects offered by departments
create sequence edu_subject_id_sequence;

-- we don't store the subject number in edu_subjects because a joint subject
-- may have more than one number
create table edu_subjects
(
    subject_id integer primary key,
    subject_name varchar(100) not null,
    description varchar(4000),
    description_html_p char(1) default 'f'
)
constraint edu_sub_desc_html_p_ck
  check(description_html_p in ('t', 'f')));

-- at MIT this will be a string like "3-0-9"
credit_hours varchar(50),
prerequisites varchar(4000),
professors_in_charge varchar(200),
last_modified date default sysdate not null,
  last_modifying_user not null references users,
modified_ip_address varchar(20) not null
);

-- we want to audit edu_subjects
create table edu_subjects_audit (subject_id integer,
  subject_name varchar(100),
  description varchar(4000),
    description_html_p char(1),
  credit_hours varchar(50),
  prerequisites varchar(4000),
  professors_in_charge varchar(200),
  last_modified date,
  last_modifying_user integer,
  modified_ip_address varchar(20)
);

-- we create a trigger to keep the audit table current
create or replace trigger edu_subjects_audit_trigger
before update or delete on edu_subjects
for each row
begin
  insert into edu_subjects_audit (
    subject_id,
    subject_name,
    description,
    description_html_p,
    credit_hours,
    prerequisites,
    professors_in_charge,
    last_modified,
    last_modifying_user,
    modified_ip_address)
values (
  :old.subject_id,
  :old.subject_name,
  :old.description,
  :old.description_html_p,
  :old.credit_hours,
  :old.prerequisites,
  :old.professors_in_charge,
  :old.last_modified,
  :old.last_modifying_user,
  :old.modified_ip_address);
end;
/

show errors

create table edu_subject_department_map (  
department_id integer references user_groups,  
subject_id integer references edu_subjects,  
-- this would be the full '6.014' or 'CS 101'  
subject_number varchar(20),  
grad_p char(1) default 'f' check(grad_p in  
('t','f')),  
primary key (department_id, subject_id)  
);

-- now we create classes. A class is a particular subject being taught  
in a particular  
-- term. However, we can also have special cases where a class is not  
associated with  
-- a term and we can even have classes that stand by themselves and  
aren't associated with  
-- subjects, e.g., an IAP knitting course  
-- the PL/SQL statement cannot create the table so we do it here.  
-- create a table to hold the extra info for each group of type  
-- 'edu_classes'

create table edu_class_info (  
group_id integer not null primary key references  
user_groups,  
term_id integer references edu_terms,  
subject_id integer references edu_subjects,  
-- if the class doesn't start or end on the usual term boundary,  
fill these in  
start_date date,  
end_date date,  
description varchar(4000),  
-- is the description in html?  
description_html_p char(1) default 'f'  
constraint edu_class_desc Html_p_ck  
check(description_html_p in ('t','f')),  

-- at MIT, something like 'Room 4-231, TR 1-2:30'  
where_and_when varchar(4000),  
-- I still don't agree with this column. I think we should use  
-- the file system to hold this and just keep a pointer to the  
-- syllabus. That way we would have versioning which we do not  
-- have now (randyg@arsdigita.com, November, 1999)  
syllabus_id integer references fs_files,  
-- we keep references to the class folders so that we can link to  
them directly  
-- from various different parts of the system.  
assignments_folder_id references fs_files,  
projects_folder_id references fs_files,  
lecture_notes_folder_id references fs_files,
handouts_folder_id references fs_files,
exams_folder_id references fs_files,
-- will the class Web page and the documents on it be open to the public?
public_p char(1) default 'f' check(public_p in ('t','f')),
-- do students receive grades?
grades_p char(1) default 'f' check(grades_p in ('t','f')),
-- will the class be divided into teams?
teams_p char(1) default 'f' check(teams_p in ('t','f')),
exams_p char(1) default 'f' check(exams_p in ('t','f')),
-- does the class have a final exam?
final_exams_p char(1) default 'f' check(final_exams_p in ('t','f')),
last_modified date default sysdate not null,
last_modifying_user references users,
modified_ip_address varchar(20)
);

-- this table audits edu_class_info
create table edu_class_info_audit (group_id integer,
term_id integer,
subject_id integer,
start_date date,
end_date date,
description varchar(4000),
description_html_p char(1),
where_and_when varchar(4000),
syllabus_id integer,
assignments_folder_id integer,
projects_folder_id integer,
lecture_notes_folder_id integer,
handouts_folder_id integer,
exams_folder_id integer,
public_p char(1),
grades_p char(1),
teams_p char(1),
exams_p char(1),
final_exams_p char(1),
last_modified date,
last_modifying_user integer,
modified_ip_address varchar(20)
);

-- we create a trigger to keep the audit table current
create or replace trigger edu_class_info_audit_trigger
before update or delete on edu_class_info
for each row
begin
insert into edu_class_info_audit (}
group_id,
term_id,
subject_id,
start_date,
end_date,
description,
description_html_p,
where_and_when,
syllabus_id,
assignments_folder_id,
projects_folder_id,
lecture_notes_folder_id,
handouts_folder_id,
exams_folder_id,
public_p,
grades_p,
teams_p,
exams_p,
final_exam_p,
last_modified,
  last_modifying_user,
  modified_ip_address)
values (  :old.group_id,
  :old.term_id,
  :old.subject_id,
  :old.start_date,
  :old.end_date,
  :old.description,
  :old.description_html_p,
  :old.where_and_when,
  :old.syllabus_id,
  :old.assignments_folder_id,
  :old.projects_folder_id,
  :old.lecture_notes_folder_id,
  :old.handouts_folder_id,
  :old.exams_folder_id,
  :old.public_p,
  :old.grades_p,
  :old.teams_p,
  :old.exams_p,
  :old.final_exam_p,
  :old.last_modified,
    :old.last_modifying_user,
    :old.modified_ip_address);
end;
/
show errors

declare
  n_classes_group_types integer;
begin
  select count(*) into n_classes_group_types from user_group_types where
group_type = 'edu_class';
  if n_classes_group_types = 0 then
    insert into user_group_types

(grouptype, pretty_name, pretty_plural, approval_policy,
default_new_member_policy, group_module_administration)
values
('edu_class', 'Class', 'Classes', 'wait', 'open', 'full');

insert into user_group_type_fields (grouptype, column_name,
pretty_name, column_type, column_actual_type, column_extra, sort_key)
values
('edu_class', 'term_id', 'Term Class is Taught', 'text', 'integer',
'not null references edu_terms', 1);

insert into user_group_type_fields (grouptype, column_name,
pretty_name, column_type, column_actual_type, column_extra, sort_key)
values
('edu_class', 'subject_id', 'Subject', 'text', 'integer', 'not null
references edu_subjects', 2);

insert into user_group_type_fields
(grouptype, column_name, pretty_name, column_type,
column_actual_type, column_extra, sort_key)
values
('edu_class', 'startdate', 'Date to Start Displaying Class Web
Page', 'date', 'date', '', 3);

insert into user_group_type_fields
(grouptype, column_name, pretty_name, column_type,
column_actual_type, column_extra, sort_key)
values
('edu_class', 'enddate', 'Date to Stop Displaying Class Web Page',
'date', 'date', '', 4);

insert into user_group_type_fields
(grouptype, column_name, pretty_name, column_type,
column_actual_type, column_extra, sort_key)
values
('edu_class', 'description', 'Class Description', 'text',
'verchar(4000)', '', 5);

insert into user_group_type_fields
(grouptype, column_name, pretty_name, column_type,
column_actual_type, column_extra, sort_key)
values
('edu_class', 'where_and_when', 'Where and When', 'text',
'verchar(4000)', '', 6);

insert into user_group_type_fields
(grouptype, column_name, pretty_name, column_type,
column_actual_type, column_extra, sort_key)
values
('edu_class', 'syllabusid', 'Syllabus ID', 'integer', 'integer',
'references fs_files', 7);
insert into user_group_type_fields (group_type, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key) values
('edu_class', 'assignments_folder_id', 'Assignments Folder', 'integer', 'integer', 'references fs_files', 8);

insert into user_group_type_fields (group_type, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key) values
('edu_class', 'projects_folder_id', 'Projects Folder', 'integer', 'integer', 'references fs_files', 8.5);

insert into user_group_type_fields (group_type, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key) values
('edu_class', 'lecture_notes_folder_id', 'Lecture Notes Folder', 'integer', 'integer', 'references fs_files', 9);

insert into user_group_type_fields (group_type, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key) values
('edu_class', 'handouts_folder_id', 'Handouts Folder', 'integer', 'integer', 'references fs_files', 10);

insert into user_group_type_fields (group_type, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key) values
('edu_class', 'public_p', 'Will the Web page be open to the public?', 'boolean', 'char(1)', 'default 't' check(public_p in ('t','f')), 11);

insert into user_group_type_fields (group_type, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key) values
('edu_class', 'grades_p', 'Do students receive grades?', 'boolean', 'char(1)', 'default 'f' check(grades_p in ('t','f')), 12);

insert into user_group_type_fields (group_type, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key) values
('edu_class', 'teamsp', 'Will the class be divided into teams?', 'boolean', 'char(1)', 'default 'f' check(teamsp in ('t','f')), 13);

insert into user_group_type_fields (group_type, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key) values
('edu_class', 'examsp', 'Will the class have exams?', 'boolean', 'char(1)', 'default 'f' check(examsp in ('t','f')), 14);
insert into user_grouptype_fields (grouptype, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key)
values
('edu_class', 'final_exam_p', 'Will the class have a final exam?', 'boolean', 'char(1)', 'default ''f'' check(final_exam_p in (''t'',''f''))', 15);

insert into user_grouptype_fields (grouptype, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key)
values
('educlass', 'examsfolderid', 'Exams Folder', 'integer', 'integer', 'references fs_files', 16);

insert into user_grouptype_fields (grouptype, column_name, pretty_name, column_type, column_actual_type, column_extra, sort_key)
values
('edu_class', 'descriptionhtml_p', 'Description HTML?', 'boolean', 'char(1)', 'default ''f'' check(description_htmlp in (''t'',''f''))', 17);

end if;
end;
/

create or replace view educurrentclasses
as
select
user_groups.groupid as classjid,
groupname as classname,
edu_class_info.term_id,
subject_id,
edu_class_info.start_date,
edu_class_info.end_date,
description,
whereandwhen,
syllabus_id,
lecturenotes_folder_id,
handouts_folder_id,
assignments_folder_id,
projects_folder_id,
exams_folder_id,
public_p,
grades_p,
teams_p,
exams_p,
final_exam_p
from user_groups, edu_class_info
where user_groups.groupid = edu_class_info.group_id
and group_type = 'edu_class'
and active_p = 't'
and existence_public_p='t'
and approved_p = 't'
and sysdate<edu_class_info.end_date

83
and sysdate >= edu_class_info.start_date;

-- create a view for all active classes in the system - these are so
-- professors can access the admin pages even though students don't see
-- these classes

cREATE OR REPLACE VIEW edu_classes
AS
SELECT
  user_groups.group_id AS class_id,
  group_name AS class_name,
  edu_class_info.term_id,
  subject_id,
  edu_class_info.start_date,
  edu_class_info.end_date,
  description,
  where_and_when,
  syllabus_id,
  lecture_notes_folder_id,
  handouts_folder_id,
  assignments_folder_id,
  projects_folder_id,
  exams_folder_id,
  public_p,
  grades_p,
  teams_p,
  exams_p,
  final_exam_p
FROM user_groups, edu_class_info
WHERE user_groups.group_id = edu_class_info.group_id
  AND group_type = 'edu_class'
  AND active_p = 't'
  AND existence_public_p = 't'
  AND approved_p = 't';

-- now, we want to be able to store information about each individual
-- in a class so we create an entry in user_group_type_member_fields

INSERT INTO user_group_type_member_fields
  (group_type, role, field_name, field_type, sort_key)
VALUES
  ('edu_class', 'student', 'Institution ID', 'short_text', 1);

INSERT INTO user_group_type_member_fields
  (group_type, role, field_name, field_type, sort_key)
VALUES
  ('edu_class', 'dropped', 'Institution ID', 'short_text', 2);

INSERT INTO user_group_type_member_fields
  (group_type, role, field_name, field_type, sort_key)
VALUES
  ('edu_class', 'student', 'Student Account', 'short_text', 3);

INSERT INTO user_group_type_member_fields
  (group_type, role, field_name, field_type, sort_key)
VALUES
  ('edu_class', 'dropped', 'Student Account', 'short_text', 2);

INSERT INTO user_group_type_member_fields
  (group_type, role, field_name, field_type, sort_key)
VALUES
  ('edu_class', 'student', 'Institution ID', 'short_text', 1);

INSERT INTO user_group_type_member_fields
  (group_type, role, field_name, field_type, sort_key)
VALUES
  ('edu_class', 'dropped', 'Institution ID', 'short_text', 2);

INSERT INTO user_group_type_member_fields
  (group_type, role, field_name, field_type, sort_key)
VALUES
  ('edu_class', 'student', 'Student Account', 'short_text', 3);
(group_type, role, field_name, field_type, sort_key) values
('edu_class', 'dropped', 'Student Account', 'short_text', 4);

insert into user_group_type_member_fields
(group_type, role, field_name, field_type, sort_key) values
('edu_class', 'ta', 'Office', 'short_text', 5);

insert into user_group_type_member_fields
(group_type, role, field_name, field_type, sort_key) values
('edu_class', 'professor', 'Office', 'short_text', 6);

insert into user_group_type_member_fields
(group_type, role, field_name, field_type, sort_key) values
('edu_class', 'professor', 'Phone Number', 'short_text', 7);

insert into user_group_type_member_fields
(group_type, role, field_name, field_type, sort_key) values
('edu_class', 'ta', 'Phone Number', 'short_text', 8);

insert into user_group_type_member_fields
(group_type, role, field_name, field_type, sort_key) values
('edu_class', 'ta', 'Office Hours', 'short_text', 9);

insert into user_group_type_member_fields
(group_type, role, field_name, field_type, sort_key) values
('edu_class', 'professor', 'Office Hours', 'short_text', 10);

-- we want to be able to divide classes further into sections.
-- this is nice for tutorials and recitations.
-- you can get the class for the section from the parent_group_id from user_groups

create table edu_section_info
(
    group_id integer not null references user_groups,
    section_time varchar(100),
    section_place varchar(100)
);

declare
    n_section_group_types integer;
begin
    select count(*) into n_section_group_types from user_group_types where
    group_type = 'edu_section';
    if n_section_group_types = 0 then
insert into user_group_types
(group_type, pretty_name, pretty_plural, approval_policy,
default_new_member_policy, group_module_administration)
values
('edu_section', 'Section', 'Sections', 'wait', 'open', 'full');

insert into user_group_type_fields (group_type, column_name,
pretty_name, column_type, column_actual_type, column_extra, sort_key)
values
('edu_section', 'section_time', 'Section Time', 'text',
'vearchar(100)', '', 2);

insert into user_group_type_fields (group_type, column_name,
pretty_name, column_type, column_actual_type, column_extra, sort_key)
values
('edu_section', 'section_place', 'Section Place', 'text',
'vearchar(100)', '', 3);

end if;
end;
/

create or replace view edu_sections
as
select
user_groups.group_id as section_id,
group_name as section_name,
parent_group_id as class_id,
section_time,
section_place
from user_groups, edu_section_info
where user_groups.group_id = edu_section_info.group_id
and group_type = 'edu_section'
and active_p = 't'
and approved_p = 't';

declare
n_classes_group_types integer;
begin
select count(*) into n_classes_group_types from user_group_types where
group_type = 'edu_department';
if n_classes_group_types = 0 then
insert into user_group_types
(group_type, pretty_name, pretty_plural, approval_policy,
default_new_member_policy, group_module_administration)
values
('edu_department', 'Department', 'Departments', 'wait', 'open', 'none');
end if;
end;
/

-- we are implementing teams as subgroups so let's create a view to see them

create or replace view edu_teams
as
select
group_id as team_id,
group_name as team_name,
parent_group_id as class_id,
admin_email,
registration_date,
creation_user,
creation_ip_address,
existence_public_p,
new_member_policy,
email_alert_p,
multi_role_p,
group_admin_permissions_p,
index_page_enabled_p,
body,
html_p,
modification_date,
modifying_user
from user_groups
where group_type = 'edu_team'
and active_p = 't'
and approved_p = 't';

-- Create edu_team group type
declare
n_teams_group_types integer;
beginsel ect count(*) into n_teams_group_types from user_group_types where
group_type = 'edu_team';
    if n_teams_group_types = 0 then
        insert into user_group_types
        (group_type, pretty_name, pretty_plural, approval_policy,
default_new_member_policy, group_module_administration)
        values
        ('edu_team', 'Team', 'Teams', 'wait', 'open', 'none');
    end if;
end;
/
create sequence edu_textbooks_sequence start with 1;

create table edu_textbooks
(textbook_id integer not null primary key,
title varchar(200),
author varchar(400),
publisher varchar(200),
-- isbn has to be a varchar and not a number because some ISBNs
have the letter
-- x at the end; ISBN will be just the digits and letters mushed together
-- (no dashes in between), amazon.com style
isbn varchar(50)
);

-- map the textbooks to classes
create table edu_classes_to_textbooks_map (  
textbook_id integer references edu_textbooks,
class_id integer references user_groups,
required_p char(1) default 't' check (required_p in ('t','f')),
comments varchar(4000),
primary key (class_id, textbook_id)
);

create sequence edu_grade_sequence;

-- records the grade types and their relative weights. this table will not
-- capture the qualitative factors, but should take care of the
-- quantitative portion of the final grade
create table edu_grades (  
grade_id integer not null primary key,
grade_name varchar(100),
class_id integer not null references user_groups,
comments varchar(4000),
weight number check (weight between 0 and 100),
last_modified date default sysdate not null,
last_modifying_user not null references users,
modified_ip_address varchar(20) not null
);

-- we want to audit edu_grades
create table edu_grades_audit (  
grade_id integer,
grade_name varchar(100),
class_id integer,
comments varchar(4000),
weight number,
last_modified date,
last_modifying_user integer,
modified_ip_address varchar(20),
delete_p char(1) default ('f') check (delete_p in
('t','f'))
);

-- we create a trigger to keep the audit table current
create or replace trigger edu_grades_audit_trigger
before update or delete on edu_grades
for each row
begin
insert into edu_grades_audit (grade_id, grade_name, class_id, comments, weight, last_modified, last_modifying_user, modified_ip_address)
values (:old.grade_id, :old.grade_name, :old.class_id, :old.comments, :old.weight, :old.last_modified, :old.last_modifying_user, :old.modified_ip_address);
end;
/
show errors

-- we want to be able to easily keep track of lecture notes/handouts
-- note that we do not keep track of author or date uploaded or even
-- a comment about it. We do not because is all kept in the
-- fs_files table, which edu_handouts references. We keep the
handout_name
-- in both places because we will be displaying that a lot and we do not
-- want to always have to join with fs_files
create sequence edu_handout_id_sequence start with 1;
create table edu_handouts (handout_id integer not null primary key, class_id integer references user_groups, handout_name varchar(500) not null, file_id integer references fs_files not null, -- what kind of handout is this? Possibilities include -- lecture_notes and announcement handout_type varchar(200), -- what date was this handout given out distribution_date date default sysdate);

-- we want to be able to keep track of assignments within the class.
create sequence edu_task_sequence;
-- includes assignments, projects, exams, any tasks a student might be
-- graded on

create table edu_student_tasks (
    task_id integer primary key,
    class_id not null references user-groups,
    grade_id references edu_grades,
    -- we have to have a task type so we can categorize tasks in
    task_type varchar(100) check (task_type in ('assignment', 'exam', 'project')),
    task_name varchar(100),
    description varchar(4000),
    -- the date we assigned/created the task
    date_assigned date,
    -- we want to know the last time the task was modified
    -- (the permissions were changed or a new version was uploaded,
    last_modified date,
    -- could be date assignment is due, or date of an exam
    due_date date,
    -- this references the fs_files that holds either the
    -- actual assignment available for download or the url of the
    -- assignment
    file_id references fs_files,
    -- who assigned this?
    assigned_by not null references users,
    -- This column is for projects where students can
    -- assign themselves to teams.
    self_assignable_p char(1) default 'f' check (self_assignable_p
    in ('t','f')),
    self_assign_deadline date,
    -- how much is this assignment worth compared to the others with
    -- the same grade_id (e.g. under the same grade group)?
    -- weight is a percentage
    weight number check (weight between 0 and 100),
    requires_graded_p char(1) check (requires_graded_p in ('t','f')),
    -- whether the task is submitted/administered online
    online_p char(1) check (online_p in ('t','f')),
    -- if an assignment has been deleted we mark it as inactive
    active_p char(1) default 't' check (active_p in ('t','f'))
);

-- views for assignments, exams, and projects
create or replace view edu_projects
as
    select
        task_id as project_id,
        class_id,
        task_type,
        assigned_by as teacher_id,
        grade_id,
        task_name as project_name,
        description,
create or replace view edu_exams
as
select
task_id as exam_id,
task_type,
class_id,
assigned_by as teacher_id,
grade_id,
task_name as exam_name,
description as comments,
date_assigned as creation_date,
last_modified,
due_date as date_administered,
file_id,
weight,
requires_grade_p,
one_p
from edu_student_tasks
where task_type='exam'
and active_p='t';

create or replace view edu_assignments
as
select
task_id as assignment_id,
task_type,
class_id,
assigned_by as teacher_id,
grade_id,
task_name as assignment_name,
description,
date_assigned,
last_modified,
due_date,
file_id,
weight,
requires_grade_p,
one_p
from edu_student_tasks
where task_type = 'assignment'
and active_p='t';

-- we want to be able to post the solutions and associate the solutions
-- to a given file

create table edu_task_solutions (  
task_id references edu_student_tasks,  
file_id references fs_files,  
primary key(task_id, file_id)  
);

-- we want a table to map student solutions to assignments  
-- this is what allows students to upload their finished papers, etc.

create table edu_student_answers (  
   -- either student_id or team_id must be non-empty  
   student_id references users,  
   team_id references user_groups,  
   task_id references edu_student_tasks,  
   file_id references fs_files,  
   -- this is the date of the last time the solutions were changed  
   last_modified date default sysdate not null,  
   last_modifying_user not null references users,  
   -- modified_ip_address is stored as a string separated by periods.  
   modified_ip_address varchar(20) not null  
);

create table edu_student_answers_audit (  
   student_id integer,  
   task_id integer,  
   file_id integer,  
   -- this is the date of the last time the solutions were changed  
   last_modified date,  
   last_modifying_user integer,  
   -- modified_ip_address is stored as a string separated by periods.  
   modified_ip_address varchar(20)  
);

-- we create a trigger to keep the audit table current

create or replace trigger edu_student_answers_audit_tr  
before update or delete on edu_student_answers  
for each row  
begin  
   insert into edu_student_answers_audit (  
      student_id,  
      task_id,  
      file_id,  
      last_modified,  
      last_modifying_user,  
      modified_ip_address)  
   values (  
           :old.student_id,  
           :old.task_id,
-- this is where we keep the student grades and the evaluations
-- that students receive from teachers

create sequence edu_evaluation_id_sequence;

create table edu_student_evaluations (evaluation_id integer primary key,
class_id integer not null references user_groups,
    -- must have student_id or team_id
student_id integer references users,
team_id integer references user_groups,
task_id integer references edu_student_tasks,
    -- there may be several times during the term that the prof
    -- wants to evaluate a student. So, the evaluation_type
    -- is something like 'end_of_term' or 'midterm'
evaluation_type varchar(100),
grader_id integer not null references users,
grade varchar(5),
comments varchar(4000),
show_student_p char(1) default 't' check (show_student_p
    in ('t', 'f')),
evaluation_date date default sysdate,
last_modified date default sysdate not null,
    -- modified_ip_address is stored as a string separated by
    -- periods.
last_modifying_user integer not null references users,
modified_ip_address varchar(20) not null);

-- we want to audit the evaluations table

create table edu_student_evaluations_audit (evaluation_id integer,
class_id integer,
    -- must have student_id or team_id
student_id integer,
team_id integer,
task_id integer,
evaluation_type varchar(100),
grader_id integer,
grade varchar(5),
comments varchar(4000),
show_student_p char(1),
evaluation_date date,
last_modified date,
-- we create a trigger to keep the audit table current
create or replace trigger edu_student_answers_audit_tr
before update or delete on edu_student_answers
for each row
begin
insert into edu_student Answers_audit (student_id, task_id, file_id, last_modified, last_modifying_user, modified_ip_address)
values (:old.student_id, :old.task_id, :old.file_id, :old.last_modified, :old.last_modifying_user, :old.modified_ip_address);
end;
/
show errors

-- now, we want to hold information about each task. It is possible
-- to have one term task but many instances of that task. For
-- instance, "Final Project for 6.916" is a term task that would
-- be kept in the edu_tasks table but ArfDigita.org is a task
-- instance that would be kept in this table. There is a many to
-- one mapping
-- we make task_id not null because every task has to be part of
-- some sort of task (either an assignment or a project)
-- we make it a task because all evaluations are done on tasks
create or replace sequence edu_task_instance_id_seq start with 1;
create table edu_task_instances (task_instance_id integer not null primary key, task_instance_name varchar(200), task_instance_url varchar(500), -- which task is this an instance of? task_id integer not null references edu_student_tasks, description varchar(4000), approved_p char(1) default 'f' check(approved_p in ('t','f'))), approved_date date, approving_user references users(user_id), -- we want to be able to generate a consistent user interface so -- we record the type of task. -- (aileen 4/00) renamed this from task_type because task_type is
-- a reserved column name in edu_student_tasks
    team_or_user varchar(10) default 'team'
check(team_or_user in ('user', 'team')),
    min_body_count integer,
    max_body_count integer,
-- we want to be able to "delete" task instances so we have
    active_p char(1) default 't' check(active_p in ('t', 'f'))
);

-- we want to be able to assign students and teams to tasks
-- we use an index instead of a multi-column primary key because
-- team_id and student_id an both be null
create table edu_task_user_map (    task_instance_id integer not null references edu_task_instances,
    team_id integer references user_groups,
    student_id integer references users,
    constraint edu_task_user_map_check check (((team_id is null and student_id is not null) or (team_id is not null and student_id is null)))
);

create index edu_task_map_idx on edu_task_user_map(task_instance_id, team_id, student_id);

-- now create a view for project instances
-- note the renaming of project_type to team_or_user as task_type is a
-- reserved column name that references edu_student_tasks
-- rename to project_type for now to keep code from breaking but we
-- eventually make the code reference team_or_user instead
create or replace view edu_project_instances as
    select
        i.task_instance_id as project_instance_id,
        i.task_instance_name as project_instance_name,
        i.task_instance_url as project_instance_url,
        i.task_id as project_id,
        i.description, i.approved_p, i.approved_date, i.approving_user,
        team_or_user as project_type,
        min_body_count, max_body_count, i.active_p
    from edu_task_instances i, edu_student_tasks t
    where i.task_id=t.task_id and t.task_type='project';

create or replace view edu_project_user_map as
    select
        m.task_instance_id as project_instance_id,
        m.team_id, m.student_id from edu_task_user_map m,
        edu_student_tasks t, edu_task_instances i
    where t.task_type='project' and i.task_id=t.task_id and m.task_instance_id=i.task_instance_id;
-- we want to allow classes to rename their roles. That is,
-- some people want to be called Professor where others want
-- to be called Instructor and still others want to be called
-- Lecturer. We don't want to just use the 'role' column
-- in user_group_roles because then we would not have a way
-- to "spam all professors and TAs" because we would not know
-- which role was a prof and which was a TA. Also, we want to
-- have a sort_key so that we know which order to display these
-- items when they are shown to the user. So, we have the following
-- table

-- so, for the case where a class wants to call the prof a Lecturer,
-- we would have role = Professor and pretty_role = Lecturer

create table edu_role_pretty_role_map (  
    group_id    not null references user_groups,  
    role        varchar(200),  
    pretty_role varchar(200),  
    pretty_role_plural varchar(300),  
    sort_key    integer not null,  
    priority    integer,  
    primary key (group_id, role)  
);

-- we want to allow professors to have office hours and
-- for students to be able to automatically sign themselves
-- up as either an individual or as a team. These items
-- are only used when students can self register. If they cannot
-- then it is just as easy for a prof to put the event into a class
-- event or under a message of the day. This could
-- also be used for advisors to allow advisees to sign up for
-- an appointment on registration day

cREATE SEQUENCE eduappointments_appt_id_seq START WITH 1;

create table edu_appointments (  
    appointment_id integer  
    constraint edu_appts_appt_id_nn not null  
    constraint edu_appts_appt_id_pk primary key,
    user_id    constraint edu_appts_user_id_nn not null  
    constraint edu_appts_user_id_fk references users,
    key,  
    -- who is the meeting with? If this is office hours,
    -- for instance, the user_id would be the TA or professor
    -- if the office hours are restricted to a particular group
    -- then we need a group_id (e.g. if the professor teaches
-- multiple classes he would only want one class to have
-- office hours at a given time
    group_id                 constraint edu_appts_group_id_fk references
user_groups,
    -- where is the meeting (office hours)
    place                   varchar(4000),
    -- Date and time.
    start_time              date constraint edu_appts_start_time_nn not null,
    end_time                date constraint edu_appts_end_time_nn not null,
    -- The number of minutes between appointments
    -- if 'increment' is left null or is longer than
    -- end_time - start_time then there is only one time slot
    -- to fill
    time_increment          integer,
    -- should students be able to see who has signed up in the other
    -- slots (or should they just see something like "Not Available")
    public_schedule_p       char(1)
    constraint edu_appt_public_ck check
(public_schedule_p in ('t','f')),
    check (starttime < end_time)
);

-- now, we want a way to keep track of who is signed up
-- for which appointment at which time
-- if the prof wants a lunch break, they just schedule themselves
-- for that time block

create table edu_appointments_scheduled ( 
    appointment_id       integer
    constraint edu_appt_sched_appt_id_nn not null
    constraint edu_appt_sched_appt_id_fk references
edu_appointments,
    user_id               integer
    constraint edu_appt_sched_user_id_fk references
users,
    group_id              integer
    constraint edu_appt_sched_group_id_fk references
user_groups,
    -- Date and time.
    start_time            date constraint edu_appt_sched_start_nn not null,
    end_time              date constraint edu_appt_sched_end_nn not null,
    constraint edu_appt_sched_nn_ck check ((user_id is not null and
                                          group_id is null) or (user_id is null and
                                          group_id is not null))
);

create index edu_appt_schedule_group_idx on
edu_appointments_scheduled(appointment_id, group_id);
create index edu_appt_schedule_user_idx on
edu_appointments_scheduled(appointment_id, user_id);
CREATE OR REPLACE TRIGGER edu_class_role_update_tr
AFTER UPDATE OF role ON user_group_roles
FOR EACH ROW
BEGIN
    -- we want to update the existing row
    update edu_role.pretty_role_map
    set role = :new.role
    where group_id = :new.group_id
        and role = :old.role;
END;
/
show errors

-- for every row that is inserted into the user_group_roles, if
-- the group is of type edu_class then we want to insert a
-- role into edu_role.pretty_role_map

CREATE OR REPLACE TRIGGER edu_class_role_insert_tr
AFTER INSERT ON user_group_roles
FOR EACH ROW
DECLARE
    v_class_p integer;
BEGIN
    select count(group_id) into v_class_p
    from user_groups
    where group_type = 'edu_class'
        and group_id = :new.group_id;
    IF v_class_p > 0 THEN
        insert into edu_role.pretty_role_map (group_id, role, pretty_role, pretty_role_plural, sort_key, priority)
        select :new.group_id, :new.role, :new.role, :new.role || 's', nvl(max(sort_key),0) + 1,


```sql
nvl(max(priority), 0) + 1
from edu_role.pretty_role_map
where group_id = :new.group_id;

END IF;
END;
/
show errors

-- if a role is deleted from user_group_roles and the group
-- is of type edu_class then we also want to delete it from
-- edu_role.pretty_role_map

CREATE OR REPLACE TRIGGER edu_class_role_delete_tr
BEFORE DELETE ON user_group_roles
FOR EACH ROW
BEGIN
    delete from edu_role.pretty_role_map
    where group_id = :old.group_id
    and lower(role) = lower(:old.role);
END;
/
show errors

-- the portal mini-tables

create sequence weather_id_sequence;

create table portal_weather (weather_id integer not null primary key,
user_id integer not null references users,
city varchar(100),
usps_abbrev references states,
zip_code varchar(10) references zip_codes,
-- the type can be: next day forecast, 5 day forecast, current
conditions
    five_day_p char(1) default 'f' check (five_day_p in
('t', 'f')),
    next_day_p char(1) default 'f' check (next_day_p in
('t', 'f')),
```
create table portal_stocks {
    user_id not null references users,
    symbol varchar(10) not null,
    default_p char(1) default 'f' check(default_p in ('t','f'))
};

create sequence portal_stocks_sequence;

--- we're currently using the calendar module and not edu_calendar
--- because the features have not been fully implemented
-- this is taken from the intranet calendar
create table edu_calendar_categories (  
    category varchar(100) primary key,
    enabled_p char(1) default 't' check(enabled_p in ('t','f'))
);
create sequence edu_calendar_categories_sequence;

-- updates from intranet/doc/sql/calendar.sql:
-- the addition of a viewable column that specifies whether the
-- entry is viewable by the public and if so, whether we should show
-- title or something in place of the title (e.g. Busy, Free, Tentative
-- MS Outlook options). also, addition of owner column that identifies
-- the entry is for: so we can display calendars with respect to
-- users or groups of users (like in a team)
create table edu_calendar (  
    calendar_id integer primary key,
    category not null references calendar_categories,
    owner not null references users,
    title varchar(100) not null,
    body varchar(4000) not null,
    -- is the body in HTML or plain text (the default)
    html_p char(1) default 'f' check(html_p in ('t','f')),
    start_date date not null, -- first day of the event
    end_date date not null, -- last day of the event (same as
    start_date for single-day events)
    expiration_date date not null, -- day to stop including the
event in calendars, typically end_date
    -- viewable as public means the title will be displayed. private
    -- means the entry will be invisible unless viewed by the
    -- owner. busy, free, or tentative will be displayed instead of
title
    -- to viewers other than owner
    viewable varchar(100) default 'public' check(viewable in
    ('public', 'busy', 'free', 'tentative', 'private'))
    event_url varchar(200), -- URL to the event
    event_email varchar(100), -- email address for the event
-- for events that have a geographical location
country_code references country_codes(iso),
-- within the US
usps_abbrev references states,
-- we only want five digits
zip_code varchar(10),
approved_p char(1) default 'f' check(approved_p in ('t','f')),
creation_date date not null,
creation_user not null references users(user_id),
creation_ip_address varchar(50) not null
);

-- start an class category so classes can have events (e.g. exam review) of
-- "group" scope
create or replace trigger calendar_cat_upon_new_class
after insert on user_groups
for each row
begin
IF :new.group_type='edu_class' THEN
    insert into calendar_categories (category_id, scope, group_id, category, enabled_p)
    values
    (calendar_category_id_sequence.nextval, 'group', :new.group_id, 'Class', 't');
    END IF;
end;
/
show errors

-- create default tables for each portal
-- start a personal category so the user can enter personal events of
-- "user" scope
create or replace trigger portal_page_upon_new_user
after insert on users
for each row
begin
    insert into portal_pages
    (page_id, user_id, page_number)
    values
    (portal_page_id_sequence.nextval, :new.user_id, 1);

    insert into calendar_categories (category_id, scope, user_id, category, enabled_p)
    values
    (calendar_category_id_sequence.nextval, 'user', :new.user_id, 'Personal', 't');
end;
/
show errors

-- the opposite of the above trigger -- for deleting users
create or replace trigger portal_remove_upon_user_delete
before delete on users
  for each row
  begin
    delete from portal_pages
    where user_id=:old.user_id;
  end;
/
show errors

create or replace trigger portal_setup_upon_page_insert
after insert on portal_pages
for each row
declare
  stock_tableid portal_tables.table_id%TYPE;
  weathertableid portal_tables.table_id%TYPE;
  classestableid portal_tables.table_id%TYPE;
  announcementstableid portal_tables.table_id%TYPE;
  calendartableid portal_tables.table_id%TYPE;
begin
  select table_id into stock_tableid from portal_tables where
table_name='Stock Quotes';
  select table_id into weathertableid from portal_tables where
table_name='Current Weather';
  select table_id into classestableid from portal_tables where
table_name='Classes';
  select table_id into announcementstableid from portal_tables
where
table_name='Announcements';
  select table_id into calendartableid from portal_tables where
table_name='Calendar';
  insert into portal_table_page_map
    (page_id, table_id, sort_key, page_side)
  values
    (:new.page_id, stock_tableid, 1, '1');
  insert into portal_table_page_map
    (page_id, table_id, sort_key, page_side)
  values
    (:new.page_id, weathertableid, 2, '1');
  insert into portal_table_page_map
    (page_id, table_id, sort_key, page_side)
  values
    (:new.page_id, classestableid, 1, 'r');
  insert into portal_table_page_map
    (page_id, table_id, sort_key, page_side)
  values
    (:new.page_id, announcementstableid, 3, '1');
  insert into portal_table_page_map
    (page_id, table_id, sort_key, page_side)
  values
    (:new.page_id, calendartableid, 2, 'r');
end;
/
show errors

-- the opposite of the trigger above -- upon deleting a page for portal
-- table we also want to delete the entries from portal_table_page_map
create or replace trigger portal_update_upon_page_delete
before delete on portal_pages
  for each row
  begin
    delete from portal_table_page_map where page_id=:old.page_id;
  end;
/
show errors
Bibliography

<http://www.studyfree.com/>


<http://eee.uci.edu/programs/auctlt/LookingBack.html>

<http://www.detac.com/solution/naweb97.htm>

<http://www.umuc.edu/iuc/cmc96/papers/poly-p2.html>


[9] Murray W. Goldberg - book chapter "Using a Web-Based Course Authoring Tool to Develop Sophisticated Web-Based Courses", Book Title: "Web Based Instruction (WBI)", Editor Badrul H Khan, University of Texas at Brownsville
