

Online Education Through Shared Resources

by

Randall Graebner

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

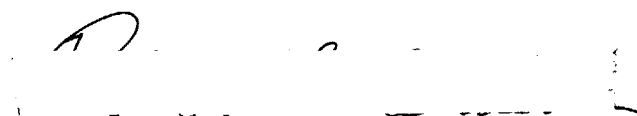
May 2000

[June 2000]

© Randall Eugene Graebner 2000. All rights reserved.

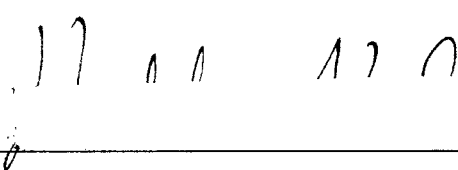
The author hereby grants to M.I.T. permission to reproduce  
and distribute publicly paper and electronic copies of this thesis  
and to grant others the right to do so.

Author



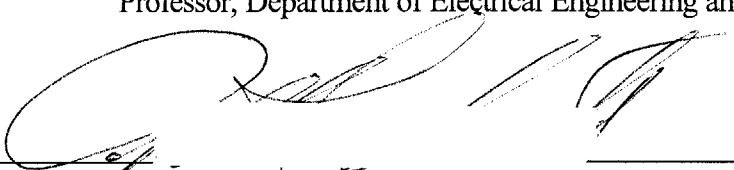
Department of  
Electrical Engineering and Computer Science  
May 2000

Certified by



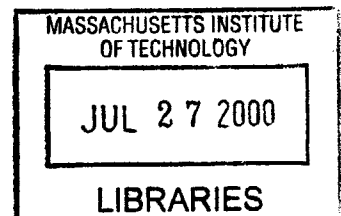
Harold Abelson  
Professor, Department of Electrical Engineering and Computer Science  
Thesis Supervisor

Accepted by



Arthur C. Smith  
Chairman, Departmental Committee on Graduate Students

ENG



# Online Education Through Shared Resources

by

Randall Graebner

Submitted to the Department of  
Electrical Engineering and Computer Science  
on May 2000, in partial fulfillment of the  
requirements for the degree of  
Master of Engineering  
In Electrical Engineering and Computer Science  
Abstract

## Abstract

This paper details the research into, design of, and implementation of a system for Collaborative Learning and Administration of Students and Staff (CLASS) to complement the traditional brick-and-mortar classroom. The system is centered on the user with the goal of providing significant new capabilities to both students and faculty alike. The modeling of classes and user groups significantly helps promote collaboration between users as these user communities can remotely interact with other users within the same community, have asynchronous access to information, and manage their individual roles and contributions as a community member. One of the primary goals of CLASS is to leverage its integrated data model so information can be exchanged between classes and departments. The CLASS system shows that it is possible to build a user-centric system that is capable of complementing traditional classes. It also shows that modeling classes and departments as online communities, when correctly implemented, can save time for both the students and instructors.

Thesis Supervisor. Harold Abelson

Title. Professor, Department of Electrical Engineering and Computer Science

## Acknowledgements

I would first like to thank Doctor Philip Greenspun for suggesting and supervising this project. I would also like to thank Aileen Tang, a fellow Masters of Engineering student, for sharing this project with me. Without her dedication and organization, this project would not have been nearly the success that it is. In addition, I would like to thank Hal Abelson for his strong feedback and for making this project possible.

I also owe thanks to the people at ArsDigita for allowing me to use the ArsDigita Community System as well as helping me efficiently adapt the software to meet the educational needs of this project. Special thanks go to Richard Li for putting up with my trivial questions and for helping out during crunch time.

Finally, this thesis is dedicated to my fiancée, Kimberly Doucette. Her constant support and encouragement have allowed me to make it through these past four year.

# Contents

<b>1 Introduction</b>	<b>7</b>
1.1 Motivation	7
1.2 The idea	8
1.3 Overview	9
1.4 Division of project	12
1.5 The paper	13
<b>2 A user-centric system</b>	<b>15</b>
2.1 User-centric versus class-centric	15
2.2 Use cases	17
2.2.1 The student	17
2.2.2 The teaching assistant	17
2.2.3 The professor	19
2.2.4 The department administrator	23
2.2.5 The system administrator	24
<b>3 Related Work</b>	<b>25</b>
3.1 Blackboard	26
3.2 WebCT	27
3.3 Command	28
3.4 eCollege.com	29
3.5 Eduprise.com	31
3.6 Virtual-U	32
3.7 Serf	32
<b>4 Goals and Design</b>	<b>35</b>
4.1 Goals	35
4.2 System Design	38
4.2.1 Users and user groups	38
4.2.2 Classes, Subjects, and Departments	40
4.2.3 Assignments, Exams, and Class Projects	42
4.2.4 Content Management and Distribution	44
4.2.5 Security and Permissions	45
4.2.6 Collaboration and Coaching	47
4.2.7 System Portal	49
4.2.8 Permanent information access	50
<b>5 Implementation</b>	<b>51</b>
5.1 Technology and implementation decisions	51
5.1.1 Information storage	51
5.1.2 Web server	53
5.1.3 Community building	55
5.2 System organization and data model	56

5.2.1	Users, user groups, and roles	57
5.2.2	Departments and subjects	58
5.2.3	Classes	59
5.2.4	Sections and Teams	61
5.2.5	Assignments, Exams, and Class Projects	61
5.3	Security	62
5.3.1	User identification	62
5.3.2	General methods	64
5.3.3	File permissions	65
5.3.4	Auditing information	66
5.4	Usability	67
5.4.1	The Portal: a customizable view of the system	67
5.4.2	Calendar	68
5.4.3	Permanent information access	69
<b>6</b>	<b>Feedback</b>	<b>70</b>
<b>7</b>	<b>Potential effects of the system</b>	<b>75</b>
6.1	Competitive Analysis	75
6.2	Why this system is different	76
6.3	Use by universities	78
<b>8</b>	<b>Conclusions</b>	<b>79</b>
8.1	Future work	79
8.1.1	Enhanced security	79
8.1.2	Increased functionality for departments	81
8.1.3	Increased coaching and collaboration	80
8.1.4	Reports	83
8.1.5	Usability	83
8.1.6	Templating	84
8.1.7	Online testing	85
8.1.8	Increased educational functionality	85
8.2	Initial system complete	86
<b>9</b>	<b>Appendix</b>	<b>89</b>
A	Competitive Analysis Tables	90
A.1	Developmental Features	90
A.2	Instructor Tools	92
A.3	Instructional Features	93
A.4	Student Tools	94
A.5	Administrator Tools	95
B	Data model	96
<b>10</b>	<b>Bibliography</b>	<b>128</b>

# List Of Figures

2.1: Screenshot of a group bulletin board	17
2.2: Screenshot of an Administrator's view of an assignment page	18
2.3: Screenshot of a class homepage	19
2.4: Screenshot of office hours scheduling page	20
2.5: Screenshot of a list of students in a class	21
2.6: Screenshot of a student information page, including grades	22
2.7: Screenshot of marks given by a grader	23
2.8: Screenshot of System Administrator home page	24
5.1: Figure showing organization of system	56
5.2: Screenshot of a portal page	68

# Chapter 1: Introduction

## 1.1 Motivation

The past decade has witnessed an ongoing information revolution that has allowed information to spread faster than ever before. This ability to immediately share information has spurred a dramatic growth in online communities, as it is now possible for people in different geographic regions to easily share their thoughts and ideas. This newfound ability to instantaneously share information has allowed us to bring more and more of our lives online. People are using the Internet to do everything from reading the news to planning for their retirement. The number and variation of services available to people is unprecedented. However, even with all of this growth, educators have not yet fully harnessed the power of the Internet.

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 even many different services available to host class websites. Many, in fact, do an adequate of using the web as a collaboration tool by providing tools such as real-time chat, virtual whiteboards, and threaded discussion forums. However, the one thing missing in virtually all of these standalone web sites 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 standalone class online they were designed using a class-centric paradigm rather than a user-centric one [Pau92b]. Therefore, these services exist as a large collection of standalone class websites.

The area of online education desperately needs a centralized, integrated, user-centric system that is extensible and customizable enough to handle the needs of large universities while still being simple enough to be used by a single professor. Current software solutions in the area of online-education have been built to be usable from the perspective of the professor, not from the perspective of the student.

## **1.2 The idea**

Numerous studies have shown that the ability to collaborate is the key to allowing the web to enhance the student's learning experience [Woo97]. Without collaboration, the web becomes nothing more than an easily accessible file storage system. The web presents the perfect environment for collaboration as it allows students to communicate with each other both in real time through chat rooms and streaming audio and asynchronously through 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 write more and to learn from one another through critiquing each other's writings. In addition, teachers from a wide variety of areas have found that web-based discussions can lead to greater participation in class discussions by all students involved [Hil90, Mor95, Cli90].

Building a collaborative web site can be a fairly straightforward task. However, people will not use the web site unless it is both easy to use and provides users with significant new capabilities that have the potential to make their lives easier. A web site must be easy to use because most users will not spend a large amount of time trying to



figure out exactly how to use it. In addition, it must provide the user with new capabilities otherwise potential users will keep doing things the way they currently are.

[Gre99]

The system for Collaborative Learning and Administration of Students and Staff (CLASS) described by this thesis attempts to provide all classes of users with significant new capabilities while keeping the site simple and easy to use. In order to accomplish this large task, the system has been designed using a user-centric approach rather than the traditional class-centric approach. The reasoning behind this is simple; the most powerful way to enhance the learning and teaching experiences is through online communities and collaboration.

### **1.3 Overview**

The CLASS system attempts to provide significant new capabilities to three types of users: system administrators, instructors, and students. The system reduces the maintenance responsibilities of the system administrators by delegating responsibility to the instructors. For the system to run properly, the administrators will need to make sure that there are nightly backups and the machines are running correctly. The only other responsibility they have is to create departments when the system is first installed and classes when they do not belong to any departments. Delegating the authority of creating most classes to the department administrators potentially decreases the amount of work required of the system administrator. User registration is completely automated and department and class administrators are able to control who is and is not in their groups.

Class instructors benefit to a much larger degree than system administrators. The online education system provides instructors with the ability to collaboratively develop documents, share a single grade book, monitor the progress of individual students, issue assignments and receive answers online as well as the ability to easily reuse past course material. In addition, the inclusion of bulletin boards and chat rooms make it possible for students to help other students learn the material, thus removing some of the burden of answering questions and explaining the material normally placed solely on the instructors of the class.

The system helps instructors by providing them with a package of automated tools to teach the class. In the same way, the CLASS system is able to help out students. Bulletin boards, for instance, foster collaboration by allowing students to post questions whenever they want. In addition, other students in the class have the ability to answer the question before the instructor. In this way, the students asking questions obtain the desired information quicker than they would have otherwise and the answering student learns by explicitly explaining the answer to the question. This forum is more helpful to students than word of mouth help because if the student posting an answer is wrong, other students can easily point this out. In addition, since users can view all postings made by all other users, the user posting the question can read other postings made by the user that answered the question. This capability allows users to check the credibility of one another before blindly following a suggestion.

To complement the bulletin boards, students are provided with chat rooms. Thus, students not only get asynchronous help from postings on the bulletin boards but they are

also able to ask questions and get immediate feedback from instructors and other students without everyone having to be in one centralized, physical location. Finally, and most importantly, the CLASS system provides users with a unified portal page that allows users to view important information for all of their classes on a single page rather than having to visit four or five individual class homepages.

In order to provide these new capabilities, the CLASS system models the classroom as a database-backed web service where members of each class belong to well-defined groups of users. As a result, the members of these communities can remotely interact with other users within the same community, have asynchronous access to information, and manage their individual roles/contributions as a community member.

How does this translate to the system's application to education? Students can obtain help from the teaching staff remotely during online office hours. In addition, they can share knowledge and help answer questions asked by fellow students when the teaching assistants (TAs) are not available. The teaching staff can collaboratively develop course materials and distribute them via a centralized mechanism. Instead of collecting student grades from many TAs, each maintaining their own records separately, professors have immediate access to all student data in a single format, managed by one database. Finally, all users have access to a reliable archive of all news, announcements, and activities that are relevant to them.

## **1.4 Division of project**

This thesis is a collaborative effort between Aileen Tang and Randy Graebner. Their collaborative efforts have been in data modeling, high-level system design, design and implementation of the pages used by a single class, bug fixes, usability and communicating with MIT's Sloan School of Management. On the individual level, Randy has been working in designing and implementing the overall security and permissions of the system. In addition, he has made extensive revisions to the file storage system to facilitate permission records on file versions rather than only the files themselves. Randy has written this entire paper, although he does use some of Aileen's work in sections 2.2, 4.2.2, 4.2.4, 4.2.7, 5.3.4, and 5.4.1. Aileen has been working on customizing the portal system to provide students with a single access point to all relevant information about their classes. She has also worked to integrate the bulletin board and chat modules into the system.

## **1.5 The paper**

Now that an overview of the project has been presented, the paper discusses further design and implementation issues present in the system.

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 so the user can better understand how the system described by the rest of the paper can be used.

Chapter 3, Related Work, discusses previous work done in this area. 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 this system.

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

Chapter 5, Implementation, presents the technology used to implement the educational system, comparing these choices to the goals of the system.

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

Chapter 7, Potential effects of the system, compares this project to the related work presented in chapter 2. In addition, it talks about potential users of the system and how they will use it.

Chapter 8, Conclusions, discusses the success of this project and outlines future enhancements that will be made to the system to make it an enterprise level solution.

# Chapter 2: A user-centric system

The main difference between the CLASS system and other software packages is that CLASS attempts to provide a broad range of functionality to the user while most other systems try to provide a large number of tools to create a class. This chapter discusses some of the key differences between user-centric and class-centric systems. It then concludes with several scenarios describing how different types of users can each use the system.

## 2.1 User-centric versus class-centric

A user-centric system is one that tries to provide the user with easy access to as much information as possible in a clear and organized fashion. User-centric systems try to make it easy for users to use, personalize and manipulate a large set of tools that can be used to facilitate teaching a class online. Class-centric systems, in contrast, try to provide the largest toolset to facilitate the teaching of a single class online. For the most part, class-centric systems are customizable on the class level rather than the individual level. Class-centric systems are usually a subset of user-centric systems as class-centric systems provide a large set of tools presented in a single standard format while user-centric systems provide most of the same tools in a user interface customized to meet the needs of the individual user rather than the needs of the class as a whole.

User-centric systems provide users with features such as a personalized interface and a centralized log in point for all of their classes. In addition, user-centric systems provide users with the ability to customize the default page they view when the log on to

the system as well as the ability to view specific class information with a single click after logging in. Users of these types of systems have access to personal calendars that provide them with the ability to keep track of personal items on the same calendar as class and university wide events. In addition, these systems foster a sense of community among users by providing features such as an accessible class directory so that users can easily locate information about other users within the system. This information can range from the user's name and email address to a portrait and a list of all postings a user has ever made to a particular bulletin board. In addition, user-centric systems allow users to keep a record of all classes they have participated in as well as a repository of all of the work they have completed for those classes.

Class-centric systems provide users with a large amount of functionality on a per class basis but lack the personalization provided by a user-centric system. For instance, class-centric systems lack centralized login pages and personalized calendars. In addition, many class-centric systems lack fundamental collaboration tools such as bulletin boards and chat rooms. And, those that do provide these tools do not provide most of the features normally found with them. For instance, bulletin boards in class-centric systems typically do not allow users to perform actions such as viewing all of the postings made by a particular user. In addition, class-centric systems normally do not provide a mechanism for classes to relate to one another. This is undesirable for several reasons. First, there is no way to see if one class is a prerequisite of another. Second, there is no way to determine if two classes are actually the same class offered at different times. Finally, this inability to link classes means that every time a professor wants to

create the class web page for the next semester, they have to start from scratch instead of simply modifying a copy of the web page from the previous term.

## **2.2 Use cases**

One of the goals of the CLASS system is to provide its users with easy access to information and simple navigation around the site. After logging in, users arrive at a portal page (discussed in section 5.4.1) that gives them one-click access to most class material relevant to their roles so that they may conveniently plan, track, and complete their tasks. The CLASS system uses knowledge about the user to give the user important information in a timely manner. In order to provide the reader with a better understanding of some of the ways the CLASS system can be used, this section walks through how a student, teaching assistant, professor, department administrator, and system administrator can each use the site.

### **2.2.1 The Student**

Ben is a student in 6.840. He logs on to the CLASS system via the main server index page. After logging on, he is taken to the portal where he sees that he has two assignments due and one quiz this week. Clicking on the link for the problem set due on Thursday takes him to the assignment's information page, where he uploads his 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 (which is referred to as his workspace) and selects the class homepage for 6.916. From the 6.916 index page, Bob is reminded by the 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 discussion forums relevant to the class. From the Q&A forum, he can view all the threads about the discussion of their project, the Site for Hunger, and posts an answer to the discussion about their data models (Figure 2.1).

## 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](#)

Full Text Search:

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](#).

---

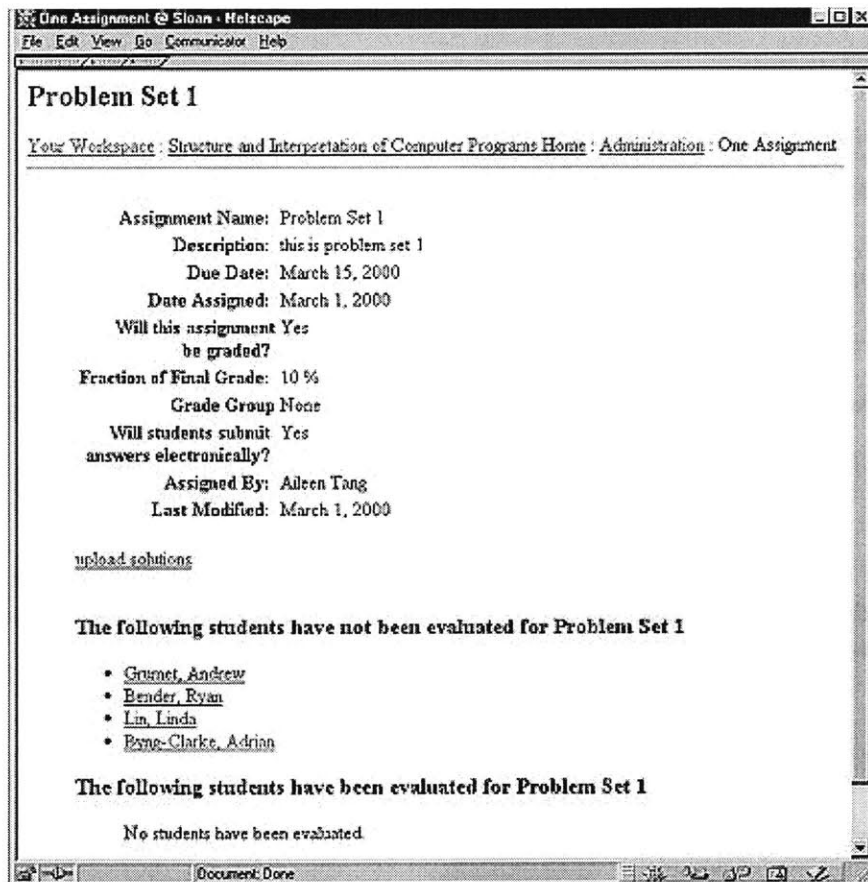
[bob@mit.edu](mailto:bob@mit.edu)

**Figure 2.1: 6.916 Project discussion Q&A forum for the Site for Hunger Team. Notice that since Bob created the forum for his team, his email appears at the bottom of the page.**

### 2.2.2 The Teaching Assistant

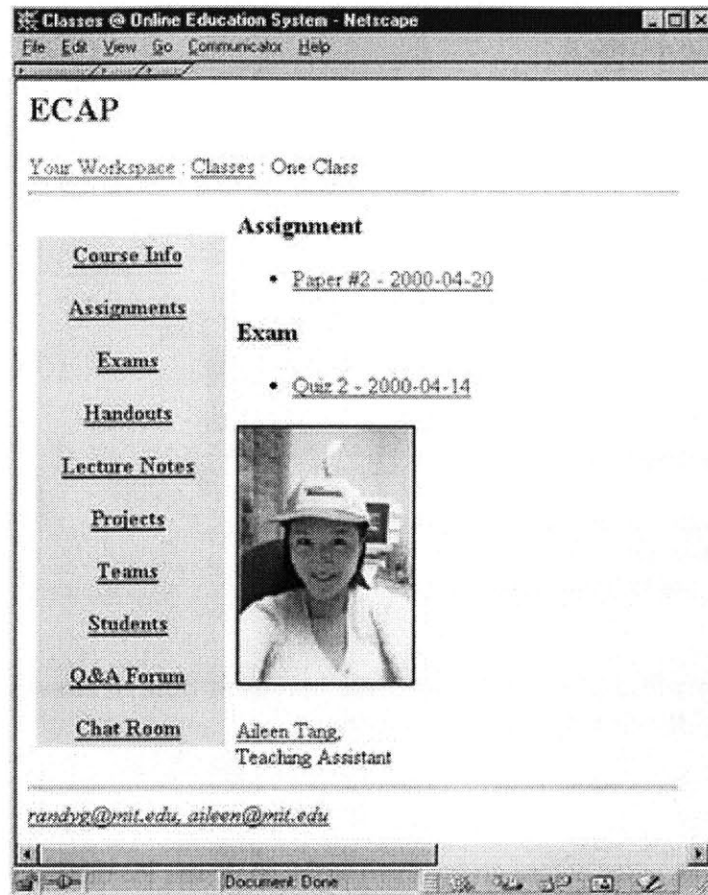
Alyssa is a graduate teaching assistant for 6.001 who is also taking a class called E-Commerce Architecture Project (ECAP). From her workspace, she clicks on the link to March 23 and adds a 6.001 problem set due on that day. Then she clicks on the link to the 6.001 administration page and looks up assignment submissions for Problem Set 1. She sees that 4 of her students have not been evaluated for Problem Set 1 (Figure 2.2), so she downloads the answers for each student, updates their submissions with her comments and uploads them again. She then fills out the student evaluation form for Problem Set 1

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 1 she can see the grades for all students who have been evaluated for the problem set.



**Figure 2.2:** This shows information about an assignment as well as a list of students who have and have not been graded on the assignment.

Alyssa 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, and projects that are due within the week (Figure 2.3). She finds and downloads a new handout that has been uploaded since her last login.



**Figure 2.3: The course homepage shows exams and handouts for the coming week as well as a picture of a randomly chosen student or faculty member.**

### 2.2.3 The Professor

Professor Smith is a lecturer for ECAP. He logs on to the CLASS system and from the portal page 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 2.4). He then returns to his workspace and enters the ECAP administration page. He uploads lecture notes for both yesterday's and next week's lecture. He grants all users read permission for yesterday's lecture notes and he only grants teaching assistants read permission for next week's lecture notes because he wants the TA's to

## Add Office Hours

Your Workspace : [ECAP Home](#) : [Administration](#) : Add Office Hours

---

The screenshot shows a web form for scheduling office hours. The fields are as follows:

- Location:** A text input field containing "3-402".
- Date:** A date picker showing "April 19 2000".
- Start Time:** A time picker showing "11:00 AM".
- End Time:** A time picker showing "4:00 PM".
- How long would you like each appointment to last?:** A dropdown menu showing "15 Minutes".
- Do you want to allow the class to see the names of people signed up for a given time slot?:** Radio buttons for "Yes" (selected) and "No".
- Continue:** A button at the bottom of the form.

**Figure 2.4: The professor is able to schedule blocks of time to allow students to automatically sign up for office hours.**

proof read the lecture notes before giving them to the students. 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. After adding this information, Professor Smith remembers that he needs to schedule a meeting with Brian Berns, a student, to discuss his performance in another class taught by Professor Smith, 6.916. But, he does not remember Brian's email address so he goes to the list of students and sorts the page by the student's last name (Figure 2.5). While he is there, he decides to view Brian's performance in the class. By clicking on Brian's name, he is able to view all of Brian's grades and the comments left by the graders (Figure 2.6). In addition, he sees a picture of Brian so he is now able to put a face with the name. After reviewing Brian's grades, he notices that Brian did not do well on Problem Set 2. But, since he did well on everything

else, Professor Smith clicks on the name of the TA and realizes that the TA has graded everyone harshly (Figure 2.7).

<u>Name</u>	<u>Email</u>	<u>Account Number</u>	
<a href="#">Adler, Jordan</a>	jsadler@mit.edu	95	
<a href="#">Agboh, Peter</a>	pmagboh@mit.edu	186	
<a href="#">Anand, Ishan</a>	ianand@mit.edu	89	Dropped
<a href="#">Artz, Michael</a>	slyph@mit.edu	181	
<a href="#">Baekkelund, Christian</a>	draco@mit.edu	183	Dropped
<a href="#">Bender, Ryan</a>	jrbender@mit.edu	87	
<a href="#">Bletsas, Aggelos</a>	aggelos@media.mit.edu	185	Dropped
<a href="#">Bonnet, Mike</a>	mbonnet@mit.edu	90	
<a href="#">Brown, chad</a>	yandros@mit.edu	64	
<a href="#">Byng-Clarke, Adrian</a>	adrianbc@mit.edu	98	
<a href="#">Chak, Dan</a>	chak@mit.edu	91	
<a href="#">Chang, Peter</a>	psc@uclink4.berkeley.edu	82	
<a href="#">Chao, Tony</a>	tchao@mit.edu	93	
<a href="#">Chen, Wei</a>	wychen@uclink4.berkeley.edu	188	
<a href="#">Chu, Michael</a>	mchu@mit.edu	100	
<a href="#">Cox, Rick</a>	rick@rescomp.berkeley.edu	81	

Figure 2.5: Professors can view a list of all students in the class. The list can be sorted by the student's name, email address, or account number.

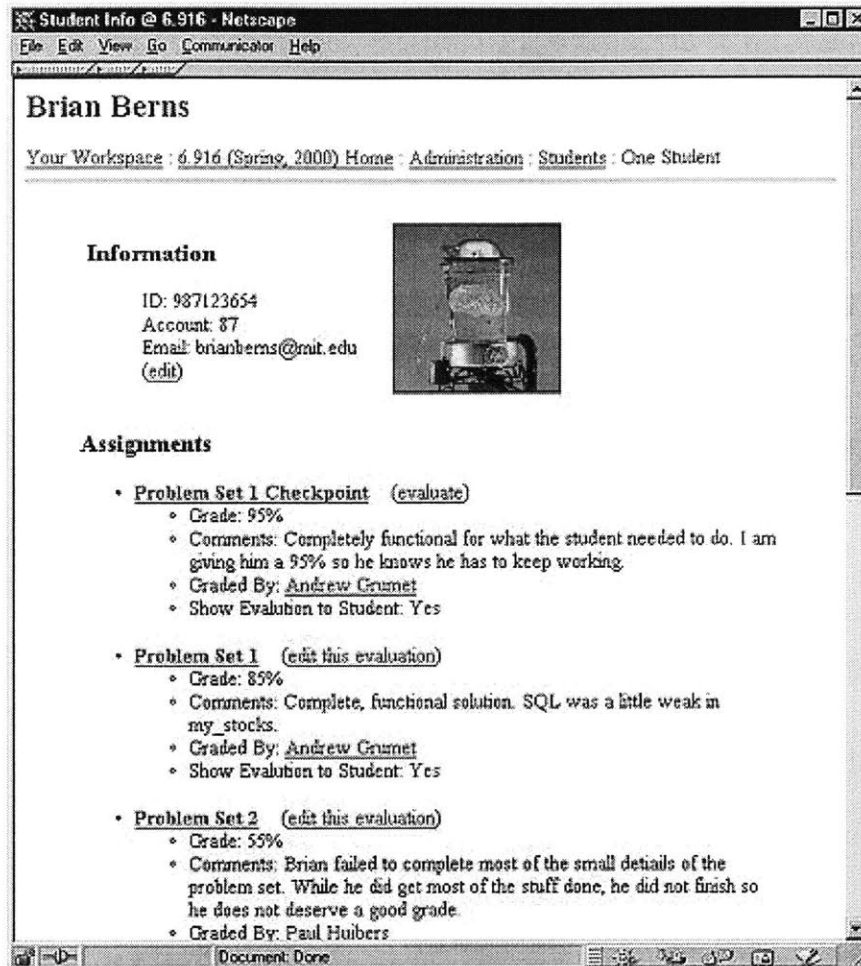


Figure 2.6: Professors and teaching assistants can view the grades and comments received by students. They can also see who did the grading and click through to see how that person has graded other students.

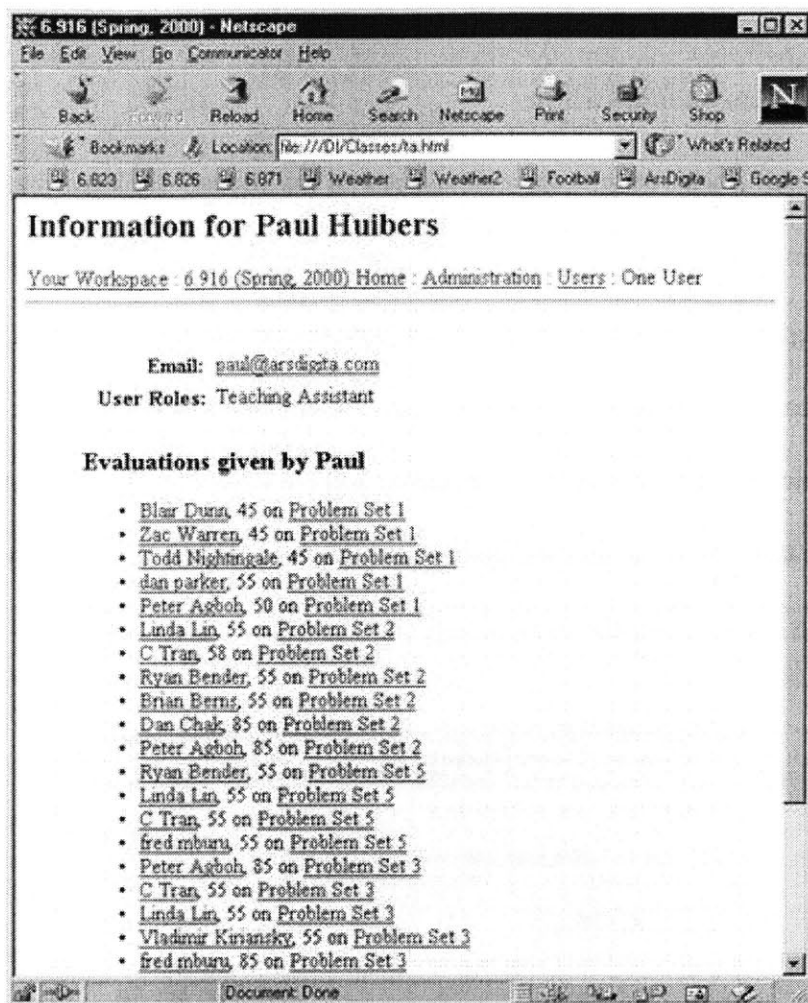


Figure 2.7: Professors and teaching assistants can see how graders grade across the board.

#### 2.2.4 The Department Administrator

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 and goes 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 instructor of the class permission 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 System Administrator

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 know that the Management department is course 15 so he creates the department, giving the department head permission to administer the department. He then remembers that he needs to add the capability for classes to be given over the summer so he goes and adds a summer term. When he is done with that, he is done with his duties for the day and logs out.

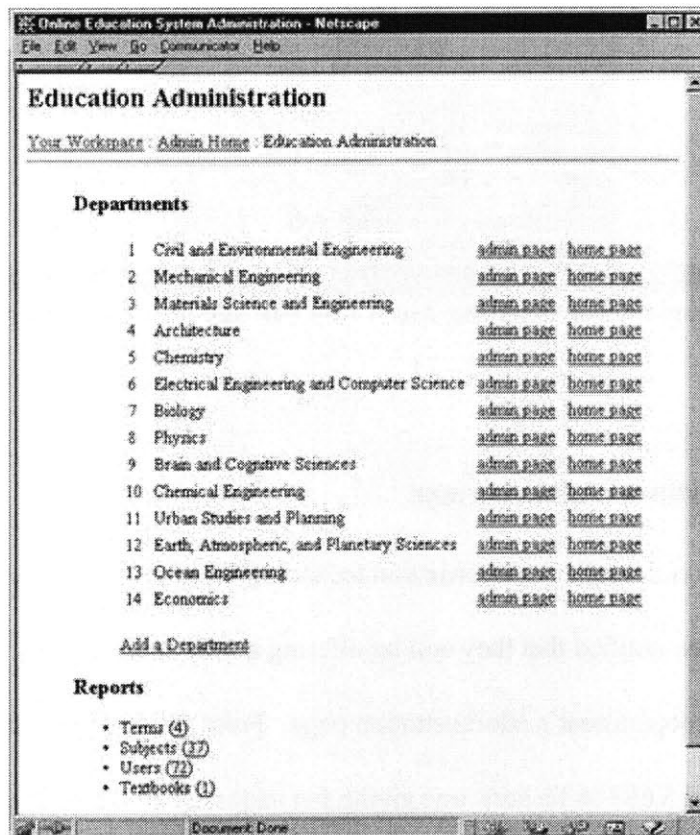


Figure 2.8: System Administrators get a top down view of the system. Here, they can view all of the departments as well as users and subjects within the system.



# Chapter 3: Related Work

There are currently many different commercial systems available that can be used to enhance the learning experience through bringing course materials and collaboration to the World Wide Web. This chapter presents a description of some of these products so that the reader will have an idea of the type of work that has already been done and why this project is different.

Almost all commercial software solutions for online education come with standard tools to facilitate both synchronous and asynchronous collaboration and communication. The synchronous communication tools include items such as virtual whiteboards and chat rooms while the asynchronous communication tools include bulletin boards, file postings, email, and audio/visual media files. Most software packages do an adequate job providing these tools but their design philosophy is flawed as most designs are centered on the class rather than the user.

Previous work done in this area has been in creating software packages to facilitate using a class on the Internet. In order to do this, most designers have created class-centric systems. For example, most packages do not allow the user to see information about multiple classes on the same page. And, a surprisingly large percentage of software solutions do not even allow professors to manage groups of users [Com00].

This chapter reviews several of these software packages, pointing out how they are class-centric in design. In addition, the discussions relate back to the use cases

presented in chapter two so that the user can begin to see the differences between these systems and the CLASS system described by this paper.

### **3.1 Blackboard**

Blackboard.com is a company that was founded with the vision of “transforming the Internet and other online networks into powerful environments for teaching and learning.” [Bla00b] And, to a large extent, the software that they have produced has the ability to do this. Their Blackboard CourseInfo is a standalone software package that can be used to facilitate the online teaching needs of a university. It is written in Java and mod\_perl accessing a MySQL database, runs on Apache and provides the standard tools of chat, bulletin boards, email, online testing, audio/visual streaming files, and a virtual whiteboard [Bla00a].

The Blackboard CourseInfo Enterprise Edition is one of the few online education packages that try to use a user-centric model rather than a course-centric one. This fact is most apparent in their MyBlackboard page that provides a personalized user interface and centralized login point for all of the tools offered by the system. From the MyBlackboard page, a student can access all of their course home pages and receive all campus wide announcements.

The MyBlackboard page, while slightly simplifying the interface for the average user, falls far short of changing the product from course-centric to user-centric. In order to successfully do that, the system would need to further personalize the portal with a page like the portal page provided by the CLASS system. The CLASS portal page

includes information such as relevant news for every class, a single calendar displaying information for all of the user's classes, and links to the relevant collaboration tools. Currently, the portal provided by MyBlackboard falls short of shifting the software from a class-centric to a user-centric paradigm as it only offers generic services such as a centralized login location, campus wide announcements, links to off campus merchants, and a personal, standalone calendar [Bla00a].

## 3.2 WebCT

World Wide Web Course Tools (WebCT) is a multi-lingual tool that facilitates the creation of web based educational environments. It uses three main methods of accomplishing this goal. First, it allows the course instructor to customize the user interface by setting custom color schemes and page layouts. Second, it provides a set of standard teaching tools such as bulletin boards, chat rooms, online quizzes, whiteboards, and presentation areas. Finally, it provides a set of tools that can be used by the instructor such as collaborative grading and course layout templates [Web00].

WebCT provides a large amount of functionality to make life easy for the student. For instance, when students read a multi-page handout, they can mark where they are and come back to it later. In addition, it is possible for students to make permanent, private notes and associate any such notes with a page of course content for future reference. Students can also use built in self-evaluation tools to judge their progress in the class.

WebCT provides a large variety of useful tools for creating a web site. However, it has several disadvantages. First, it is written in Perl using a proprietary database.

While the database does provide an API for interfacing with it, it would be much easier for the programmer if a standard database were used. Second, there is no easy way to relate courses. It is not possible to say that two classes are the same but are offered at different times. Third, WebCT is course-centric. While it does offer portal page, the portal is only personalized to a small degree and does not include a combined calendar nor does it provide links to relevant information for the classes for which a user is registered. Finally, and most importantly, there is no convenient way to group students together. This makes group projects much more difficult to manage in addition to lacking the ability to easily send email to a subset of the students in the class.

### **3.3 Command**

Command is a course-centric Web-based course management and delivery system developed at MIT in conjunction with industry partners such as Intel, IBM, and Lotus [Com00]. It is modeled after the traditional course homepage, which provides a repository for course documents and information. Command has made good progress towards its goal of storing information as it is possible for professors to easily upload assignments and their solutions and students can easily upload their answers. However, in reaching this goal, the developers have used a class-centric model and have, to a large degree, neglected collaboration tools and usability.

Command fails to be user-centric in many ways. This fact is most apparent when it is noticed that feature development was centered on making life easier for the instructors while ignoring the needs of the students. For instance, there is no entry point where students can gain speedy access to their course websites. Instead, students must

either bookmark each course web page individually or find the course in the class listings every time they enter the site. This is in clear contrast to systems like CLASS and Blackboard that provide the student with a unified entry point to all of the classes. The course calendar provided by the CLASS system on the first page after logging in is buried 3 pages deep in the Command system.

The lack of collaboration tools is perhaps the largest hole and in the Command package and it is the single largest reason the Command is classified as a class-centric system. The system lacks the file versioning that is essential for collaborative document development. In addition, the system's only means of facilitating student-to-student collaboration is through a single class bulletin board. The system does not even have a simple chat room. Usability of the system is not much better. The calendar, an often-used feature, requires the user to go three pages into the system to see it and even when the student reaches the calendar, it only displays information relevant to the particular class instead of information for all of the classes for which the student is registered. This is in contrast to the user-centric CLASS system that provides the student with a single calendar of all a user's courses on the first page after the user logs in.

### **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 home page 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 web sites [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 web site.

The CampusPortal software package provides the ability to create an entire online campus. However, it lacks key features such as the ability to associate courses with each other and 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. This is made most apparent by the fact that they both lack personalization on the user level. When users log in, for example, they do not see a consolidated view of their classes. In addition, where student teams can easily be assigned to projects in the CLASS system, the eCollege suite of tools does not allow an instructor to easily assign tasks or grades to groups of students.

### **3.5 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. This means that a student will have to go to every home page and then to the calendar. That is a total of eight pages that a user taking four classes must visit. This is in stark contrast to the user-centric design of the CLASS system that allows users to view a consolidated calendar and assignment list on a single page.

### **3.6 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 easily be able to see if one class is a prerequisite for another or if the class has ever been offered before.

### **3.7 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 web site thus giving them an idea of how they are doing in the given class. Also, students are able to submit their assignments online and view the syllabus in the form of a calendar [Icu00].

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 being easy for the student 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, one used by the CLASS system described in chapters 4 and 5, 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 web sites, 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.

# Chapter 4: Goals and Design

The overarching goal of the CLASS system is to make life easier for all of the people involved in learning and teaching. This is an ambitious goal that will have to be reached by taking small steps. The first step to reaching this goal is to convince faculty and students to use the system. However, it may be difficult to obtain the cooperation of the faculty that must use the software. What might cause difficulties in adopting such a system? One main obstacle is that most teachers are content using whatever method they are currently using. Whether they are using one of the software packages described in chapter 3 to manage their website, they are managing their own site, or they do not have a website at all, it could prove difficult to convince the faculty that the CLASS system will be worth the effort of switching from their current system. To overcome this obstacle, the system must make the transition from their current method to this system as painless as possible. In addition, the system must provide an easy way for users to maintain and update their web sites. This chapter discusses some of the high level goals of the CLASS system followed by a discussion about the design of several key sections of the system.

## 4.1 Goals

The main goal of the CLASS system is to supplement classroom learning by providing all of its users with significant new capabilities. Any system that is going to be adopted must meet several criteria. First, the system must provide substantial improvement above and beyond the tools currently used by the teaching faculty. These new capabilities can include, among many other things, easy distribution of files, a shared grade book, and viewing reports of student progress. If it does not, the faculty will allow

inertia to play its role and will not start using the new system. Second, the system must provide significant new capabilities to the student including easy access to information for all of their classes as well as the ability to ask questions at any time that is convenient for them. If students do not have a good reason to use it then, like the faculty, they will continue doing what they are used to doing. Third, and most importantly, the system must foster collaboration. If students and faculty do not collaborate with each other then the system is nothing more than a fancy file storage system. In addition, collaboration is the foundation for many of the tools that can ultimately make life easier for the users of the system. Finally, the system must be easy to use. If either students or faculty have a difficult time using the system, they will not use it. Unless both groups of users consistently use the system, it will not effectively serve its purpose.

If the CLASS system is going to meet these criteria, it must provide at least the following minimal feature set:

- Faculty must be able to add collaboration tools such as bulletin boards and chat rooms by simply giving them a name, setting the permissions, and pressing a button to create it.
- Faculty must be able to comment on the performance of any student or group within the class and view comments left by other faculty. This includes the grades for students on all of their assignments. This collaborative grade book must be accessible to the faculty in no more than 2 pages. If it is much deeper, the faculty may be reluctant to use the system.

- Faculty and students alike must be able to easily collaborate using bulletin boards and chat rooms for online office hours and question and answer sessions.
- Students must be no more than a two clicks away from the information they need. This means that if a student is viewing a problem set for one class, he should be able to view the assignments for another class in no more than two page views.
- The system must be designed in such a way that no single user or administrator can be the bottleneck for the creation of new classes or the addition of any collaboration tools or users to a given class web page.
- Faculty and staff must be able to view information about classes as a whole as well as information about particular problem sets or particular problems on problem sets.
- The system must do no harm. That is, the system must not make some features more difficult to do while other features easier. For instance, most professors like to reuse material from semester to semester. The system must make it easy for professors to do this by providing a mechanism to “copy” a web page, assignment, or handout from a previous semester. If it did not then it would be causing harm as creating a new class on the system would be more difficult for the professor than if he was not using the system.

This feature set is the absolute minimum the CLASS system can start with. In order to be fully functional, it must provide additional features such as the easy creation and online tests as well as customized reports from the data already present within the database.

## **4.2 System Design**

Administering and participating in an online community can be simplified into two distinct tasks; uploading information to the database and reading it back out. The challenge arises in deciding which information to solicit from the user, how to store that information in the database and how to organize the information on the screen when it is given back to the user. The system must be designed in such a way that users will never question why they are giving a certain piece of information to the system. In addition, users should never be confused about what they are seeing. Finally, the information should be efficiently organized in the database so that the programmer can easily manipulate whatever data is needed without making the user wait for a long period of time. This section discusses some of the design issues that were considered during the development of the CLASS system as well as some of the alternatives.

### **4.2.1 Users and user groups**

The use of the CLASS system begins and ends with its users. These are the people that will be contributing to the online community and making it grow. If the users are not happy or cannot access what they need, they will not return. In general, the end users are either class instructors or students. It may even be the case that some users are instructors for some classes while students in other classes. In addition, the general public, who will not need to identify themselves, may also use the system.

The identity of the user is important in determining whether the user is allowed to perform a specific function. For instance, the system should only allow instructors to grade student homework submissions. And, only students should be prompted to turn in

their homework. In addition, the identity of the user is also important for tracking the user's actions as well as notifying them of particular events. Also, in order for an online community to work effectively, the system must be able to tell other community members who made what posting. Thus, when a user first enters the system, they should be given the chance to identify themselves to the system.

Once the system knows who the user is, it is able to customize the interface appropriately. For instance, it is able to show the user information about appropriate classes and can provide links that are appropriate to the user's role within the system (this is discussed in more detail in sections 4.2.2 and 5.2.1). Also, once the system knows the identity of a user, it can easily collect information about the user such as postings the user has made to bulletin boards and chat rooms. In addition, people with the appropriate permissions can then go in and access any recorded information about a given user. This is convenient in cases where a student is on the border for a particular grade and a professor would like to measure their participation. The information lookup can also work the other way. Students may want to find out when a professor has office hours or information about potential teammates. Or, it may be the case that an academic advisor wants to find detailed information about a given student.

In addition to identifying users, the system needs to be able to identify groups of users. For instance, the system needs to know which subset of users belong to a certain class. Or, which subset of that class belongs to a given team. Knowing this information allows the system to provide links to private discussion forums or folders within the

virtual file system to specific groups. Thus, being able to group the users together is an essential part of collaboration and properly restricting access to certain areas of the site.

#### **4.2.2 Classes, Subjects, and Departments**

Classes are modeled as groups consisting of users each with distinctive membership roles (e.g. students, TA's, instructors, etc.). This approach was chosen because many users make up a single class and these users have distinct roles, each of which have distinct permissions within the system. Modeling classes as groups of users allows the system to easily create class bulletin boards, messages of the day, and chat rooms that are only viewable by members of the class. In addition, it provides an easy way for the programmer to query the database, leaving out information that is not relevant to members of the class.

The alternative to making classes a type of user group is to treat classes as their own entity. This is the approach that is taken by many of the systems discussed in chapter 3. By making classes their own object, we are making classes the center of the system and this design can make it difficult to customize the system for the user. In addition, the ArsDigita Community System (the toolkit that is used as a foundation for the CLASS system) is centered on user groups and all of the standard tools provided for group wide functionality such as bulletin boards and chat rooms. If the system were to treat a class as its own object rather than a user group, the CLASS system would not be able to take full advantage of the many key features of the ACS. Finally, classes are actually just a group of users studying the same material. Therefore, it is only natural to represent a class as a user group.



Each class is a child of a subject, which owns properties such as name, semester, description, and faculty 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 allows for 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 classes offered during previous semesters while only the current class is considered “alive” in the system.

While classes are modeled as groups of users, subjects are not. The reason for this is that there are normally only a few people that have any interest in maintaining the properties of a subject or creating the classes. And, these people are almost always members of the department to which the subject belongs. Therefore, members of the department are responsible for maintaining information within the subject.

The other alternative would be to make subjects user groups. This path was not taken because subjects do not need bulletin boards or chat rooms nor do users ever need to spam members of a subject. In addition, if subjects were treated as a user group, then every time a subject was created or the professor in charge of the subject changes then someone would have to update the membership of the user group. In the case of adding the new subject, the department administrator would have to select a user to be in charge. In terms of switching control of the subject, the professor that used to be in control would have to explicitly add a new faculty member to the group. This is a less an optimal

situation because it may be the case that the departing faculty member is not in charge of the subject any more because they are no longer working for the university. In this case, the department would have to solicit the system wide administrator to go in and fix the permissions. For these reasons and many more, subjects are modeled as a separate entity rather than as a type of user group.

Unlike subjects and much like classes, departments are modeled as user groups. Departments consist of a group of users that are in charge of maintaining the subjects within the departments. Departments will always have multiple members and could greatly benefit from having features such as a group folder in the file system and departmental bulletin boards. 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).

### **4.2.3 Assignments, Exams, and Class Projects**

Every class has assignments and many classes have exams and class projects. And, since this information is going to be frequently requested, it must be stored in the database in an efficient manner. There are many different ways that this can be done. First, it would be easy to create a table for assignments, one for exams and yet another for projects. Storing all of the information in different tables would make it extremely easy for the programmer to know which information to solicit for each type of task. In addition, it would be obvious to a new programmer where assignments, exams, and projects are stored. However, this method has the disadvantages of having to maintain

three separate tables and having to write a separate set of pages for soliciting and displaying information for each type of task.

A second option for storing this information is to store all of the assignments and exams in one table and the projects in a second table. This makes sense because assignments and exams require the same type of information. In addition, projects tend to be separate from exams and assignments because many projects are done in groups and it is often desirable to have extra information stored for each group. However this, like the first possibility, has the disadvantage of requiring extra pages for soliciting the information and more complicated database queries when the system is generating reports.

A third approach, the one taken by the CLASS system, is to store all information about assignments, exams, and projects in a single table. There are many advantages to this approach. First, all three types of tasks require similar information such as a name, when the task is due, who assigned it, and a pointer to a URL or file describing the task. In addition, some classes have group assignments and it does not make sense to treat project teams and assignment teams any differently. Rather, by storing all of the tasks in a single table, we can have a single user group type that can be used to group users working on the same project or assignment. In addition, it greatly simplifies queries used to generate reports. Finally, storing all of the tasks in the same table greatly simplifies the way grades are stored within the system. It is now possible to have the grades table contain a single column referencing the tasks table rather than having three separate columns, only one of which could not be null.

This was a difficult decision because there are many different valid ways that assignments, exams, and class projects could be represented within the database. The CLASS system stores all of the information in a single table because all three tasks are fundamentally the same. The original design of the system called for exams and assignments to be stored in the same table with projects in separate tables. However, as it was realized that all three tasks could be assigned to individuals or groups, the design changed to only use a single table. This design decision simplified both the tasks representation as well as the storing of grades and evaluations within the system.

#### **4.2.4 Content Management and Distribution**

The online education system uses a centralized file storage system to organize and serve all course material and student assignment submissions. This material can include content such as lecture notes, handouts, assignments, research papers, exams, and student projects and can take the form of any type of file from a Microsoft Word document to a JPG image. The interface used for this uploading allows the user to upload either a file from their computer or a URL referencing a remote web page. The interface should be easy to understand and should allow the user to set the permissions on the file when it is uploaded. In addition, it is essential for the file system to allow for file versioning and permissions per version rather than per file. The reasoning for this can be seen in the following scenario:

A week before a quiz, a professor uploads a draft of the quiz solutions and notifies the teaching staff to look over them and fix anything that is not quite correct. A teaching assistant then downloads the solutions, fixes a few typos and then uploads a new version. Later, a recitation instructor downloads the updated

version, adds another possible solution to one of the problems and then uploads the update version. After the quiz has been administered, the professor can then simply change the permissions on this third version so that all the students may now view it.

If the permissions were implemented on a per file basis rather than per version, the students would be able to see old versions of the solutions even though they should not have permission to view them. By using a central file system and implementing permission on a per version granularity, the system promotes collaborative document creation.

This same file system also acts as a centralized location for the development, distribution, and submission of assignments. There are several reasons for taking this approach. First, the professor can collaboratively develop an assignment in the same manor as described in the scenario above. When the document is complete, the professor can make it available to the class. Then, students can upload their solutions to this same repository, allowing the instructors to download and grade the student submissions. In addition, using a central file storage system allows the student to make their submissions available to the public therefore creating a portfolio of their work simply by using the CLASS system. In addition, using a single file system for all files provides an easy way to copy files from one class to another or from one department to another. If separate file systems were used, the transfer could be much more involved.

#### **4.2.5 Security and Permissions**

Now that the system knows who the user is and has information both about users and classes, it needs to make sure that users are only allowed to view 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 the latest version available to all users. To effectively meet these goals, the system must provide several features. First, the system must provide variable security. Some parts of the system should be open for the public to view and others should only be viewable by a single user. Second, the system must provide variable levels of user verification. The system administrator should be able to decide if all access should be over encrypted or unencrypted connections and whether or not to use client certificates (or kerberos) for client authentication. If some sort of client authentication were not performed (and only SSL is used), then it would be possible for an intruder to take over someone's email address and break into his or her account. This is a less than optimal situation when the system is used to store personal information and grades.

There are several ways to meet the goal of providing variable levels of security. First, the system could provide group wide security. This would allow everyone in a particular group to view particular information. This solution will not work if a class is a user group because we do not want to allow students to see the grades of other students. The solution would work, however, if we made professors a user group and students a second user group and a class would be the combination of these two groups. This solution, however, is less than optimal because this makes it difficult to have class-wide collaboration since it would require all of the collaborative tools be to open to multiple groups instead of only a single group. This many to one mapping is not desirable as it would require an extra mapping table where restricting features to a single group only requires an extra column in the feature's table.

A second solution, one used by the CLASS system, is to provide security based on a user's role within the group. The CLASS system uses this approach because it provides the most flexibility. The roles for a class are professor, teaching assistant, student, and dropped. 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 the class. In addition, this approach provides professors with the flexibility of being able to decide on a per 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 want to let the general public see any part of the class web page. Or, it may be the case that a professor does not want teaching assistants to be able to upload assignments but they can see grades and another professor wants to let teaching assistants upload assignments but not see the grades.

#### **4.2.6 Collaboration and Coaching**

A traditional brick-and-mortar classroom experience includes the two key components of collaboration between students and faculty as well as the coaching of students by the instructors. In order to be successful, the CLASS system must try to emulate and enhance these key features.

The CLASS system must provide students with the opportunity to interact with other students and faculty through both synchronous and asynchronous means. Users of the system interact asynchronously in many different areas of the site. The most obvious of these is the class bulletin board. However, students should also be invited to comment on class projects as well as exchange email with each other. For synchronous collaboration, students and instructors should be able to interact in a real-time chat forum. This allows students to discuss assignments with each other as well as probe the instructors for help. The system design should also allow the easy integration of additional plug-ins (such as video conferencing and Java-based white boards) as they become available.

In addition to easy communication and collaboration, the classroom experience also offers the ability for instructors to coach the students towards their goals. One easy way to facilitate this is to provide instructors with up to date information about how individual students are doing in a particular class. The CLASS system can make this process easy for the instructor by using the online grade book to provide a detailed view of each student's progress. That is, professors should be able view all of the grades and comments for a particular student on a single page thus easily being able to see how a student is doing in the class and what assignments have and have not been graded (as discussed in section 2.2.3 and seen in Figure 2.6). In addition, a professor can view a table showing all students and their problem set grades so that it is easy to determine which student performed the best or worst on a particular problem set. And, if desired, the professor can drill down into the page and view the comments for a particular student. These two formats were chosen because they allow the professor to view the grades of



the entire class or the grades for a particular student. This allows the professor to easily find students that are performing noticeably below the class average and then view detailed information specific to that student.

#### **4.2.7 System Portal**

A user-customizable portal page is the centerpiece of the system. When users first log in, they are presented with a customizable portal page that pushes important information in front of the user. The portal page should be a page that users are willing to visit several times a day, using the pages as a centralized management location for their academic and (to a lesser extent) non-academic lives. To be successful in attracting users, the portal imitates the features and benefits of other portal pages such as MyYahoo and MyNetscape.

The CLASS portal page has been designed with several criteria in mind. First, the system must be easy to use. Users can add, delete, and rearrange pieces of the portal page using the same user interface provided by major commercial portals. Second, the portal page must be tightly integrated with the educational system. The portal page can contain announcements for all of a user's classes, a consolidated academic and personal calendar, as well as links to all upcoming and past exams, assignments, and projects. Third, the portal page must be convenient for the user. Users want to be able to access information quickly and easily. Therefore, everything should be only a few clicks from the top-level portal page. Finally, the portal page must be diverse. Not only should it provide information highly relevant to the user's education but it should also provide

customizable information such as stock quotes, current weather conditions, and personal bookmarks.

When a user first logs on to the system, the portal page contains space for class information such as bulletin boards, recent announcements, assignments and projects. In addition, it contains a space for current weather conditions and stock quotes. All of this is customizable at the system wide level. Therefore, if a system administrator wants to allow users to link to streaming versions of their favorite mp3, it would be a simple addition. Providing this flexibility and customization provides the portal page with the greatest chance of being efficiently used.

#### **4.2.8 Permanent information access**

One feature that is often overlooked in web sites is permanent URLs. It is never a good idea to break a link because people tend to bookmark and link to pages. Thus, it is imperative that the online education system keeps permanent URLs for every file ever uploaded to the system. There are many different ways to point to this uploaded file and they all work equally as well as long as all URLs are unique and they do not change. In addition, it would be ideal if a human could determine what the URL was simply by looking at it as this makes the system much more user friendly.

# Chapter 5: Implementation

This chapter includes an overview of the implementation of a user-centric online education system. The chapter starts with an introduction to the tools used to create the site. It then continues with an overview of how the system is organized and a discussion of the data model. This is followed by an overview of how security and authorization is handled. Finally, the chapter concludes with a discussion of the usability of the system.

## 5.1 Technology and implementation decisions

Achieving the ambitious goals outlined in chapter four requires the careful selection of tools. If the instruments used to create this system are chosen poorly then the system will never be adopted. This section briefly discusses which tools are used by this system and why they were chosen.

### 5.1.1 Information storage

The CLASS system must be able to store a tremendous amount of information about the users and classes of the system. In addition to storing basic user and class information, it must be able to keep records of all of the bulletin board and chat postings as well as storing all of the files uploaded as assignments or student solutions. The system must also be able to access all of its information quickly and easily. In addition, it must be able to search through all course material for key words or patterns that may be present within any of the information. Since course material could include anything from Microsoft Office documents to PDF files to images, the storage location must be able to be searched using many different techniques. Another requirement for the CLASS

system is that it must be easy to create backups of the data stored within the system and it must be easy to transfer the data from its current location to different location and possibly even into a different storage method.

There are many different ways to provide this type of storage. One common solution is to store all of the information in files within the file system and accessed through Common Gateway Interface (CGI) programs. Yet another common answer is to store all of the user and class information in a relational database and all of the uploaded files in the file system. Another approach is to store all of the information, including the uploaded files, in the database. The two previous approaches allow the programmer to access the data in a variety of ways, including CGI programs as well as other interpreted and compiled languages. Still another solution to this problem is to store persistent objects in serialized form in the file system, accessing them through a related programming language.

The CLASS system stores all of the information in a relational database as this solution comes the closest to meeting the needs of the system. Relational databases are robust yet fast. They are faster than CGI scripts and persistent objects when storing and accessing data. In addition, they are easy to program using the tools outlined below. The relational database chosen for the initial implementation of this system is Oracle 8i [Ora00], a commercial database focused on providing fast and robust access to Internet services. The Oracle relational database is a powerful and robust database that is easy to program. It supports transactions and has a relatively robust recovery mechanism. In addition, it uses mostly ANSI SQL which means that even after the system is written, it

will be possible to port it on to another database if the need arises. Oracle has been designed so that it can talk to other databases. So, if the registrar's office of a university using this educational system is running Microsoft SQL Server the system will still be able to communicate with the database of the registrar. Oracle also possesses a full text search engine that allows the user to search through text strings and all files stored in the database. Since it is desirable for users to be able to search through uploaded files, the system stores all files in the database instead of the file system.

The CLASS system will run Oracle on a Unix operating system for two reasons. First, since many people will be relying on the system working at all times, we want the operating system to be reliable. Unix is one of the most stable and reliable operating systems currently available. Second, Oracle is developed on Unix. This means that new releases of Oracle and bug fixes for current releases are available first on Unix. It also means that Oracle is tested the most thoroughly on Unix which means that Oracle is going to be the most stable when running on Unix.

### **5.1.2 Web server**

There are many different web servers available on the market. The task is to decide which server to use. Apache is the most popular solution to the problem and is widely supported by a large user group. Oracle web server is the option that would provide the system with the fastest server and would remove a layer of complexity from the system as it is built directly in to the Oracle database. However, the CLASS system uses AOLserver [Ame95].

AOLserver was chosen for several reasons. First, AOLserver has a proven track record as it serves the billions of hits a day received by <http://aol.com>. This is in stark contrast to the java server included within Oracle which is in only its second release and has not been tested on large volume web sites. Second, AOLserver is multithreaded. That is, it runs using only two processes rather than dozens or even hundreds. In addition, it maintains a pool of open database connections, distributing them when needed. This is in contrast to Apache, which forks a new process whenever another page is requested and if that process needs access to the database, it requests a new connection with the database. These two features combine to force the Apache web server to perform slower than the efficient AOLserver.

In addition to performance issues, development concerns also played a role in selecting a web server. AOLserver provides both a Tcl and a C interface. This is convenient because Tcl can solve most problems encountered in web programming and if Tcl cannot solve it, the programmer can use the C API within AOLserver or the Java API provided within Oracle. AOLserver provides the Tcl interface for the developer for several reasons. First, Tcl is a non-typed language and everything is represented as a string. This is a feature especially suited to use on the web since everything sent to the user's web browser is a string and everything sent to the database is a string. Thus, Tcl does not force the programmer to convert between types. Second, Tcl is an interpreted language and therefore is a fast development tool since it does not need to be compiled. Finally, Tcl is a simple language that is easy to learn so there is not a large learning curve for people programming in Tcl for the first time.

There are three final reasons for selecting AOLserver as the web server. First, like Apache, it is open source so if the system requires a feature not offered by the server, it is possible to add that capability to the server. In addition, if AOLserver follows historical trends, the open sourcing of it will cause it to become even more reliable and robust as additional programmers contribute to its code base. Second, AOLserver was chosen so that the system can be built using the ArsDigita Community System, discussed in the section below. Finally, there is a mod\_AOLserver that allows AOLserver to run within Apache. Thus, if at a later time there were a need to run Apache instead of AOLserver, it would be straightforward to change the web server. A conversion from Apache to AOLserver would not be as simple.

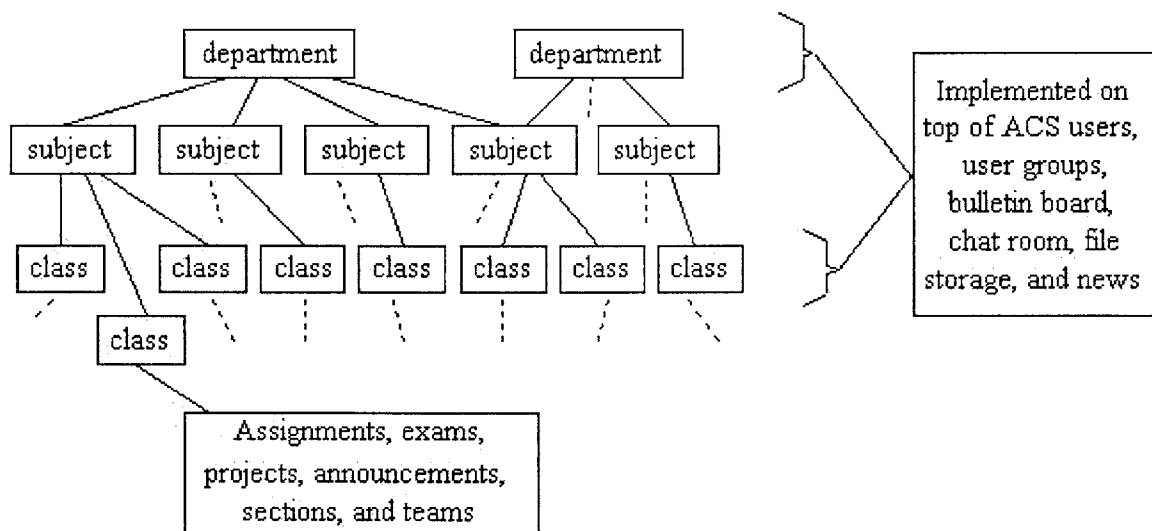
### **5.1.3 Community building**

As was discussed in chapter 4, in order for the CLASS system to meet its goals, it must be a user-centric software package that fosters collaboration. Rather than starting from scratch, the CLASS system is built upon the ArsDigita Community System (ACS) [ARS00]. The ACS provides a lot of the core functionality that the system requires including bulletin boards, chat rooms, and a user-centric data model. In addition, the ACS provides a user groups module that allows users to be placed into categorized groups. These groups, in turn, make up the core of the educational system. The ACS provides a perfect starting point for this project as it uses the same technology chosen for the online education system and is a proven solution for creating high volume, user-centric community web sites.

The ACS already provides many of the essential tools for successfully building a community. Therefore, this thesis is focused on harnessing existing tools in the ACS to effectively create online educational communities and extending the ACS with functionality specific to education. Such functionality includes administration of classes and departments as well as online grading and the collaborative development of documents such as class assignments and solutions.

## 5.2 System organization and data model

The CLASS system is complex and therefore contains many different pieces. There are users that are members of and have roles in groups; there are classes, subjects, and departments; and there are assignments, exams, and projects in addition to everything provided by the ArsDigita Community System (Figure 5.1).



**Figure 5.1: A diagram of the hierarchy representing the structure of the data model. 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 entered at any level of the diagram. Finally, every class contains announcements, exams, projects, sections, and teams.**

This chapter discusses in detail the implementation of many of these parts as well as how the data is modeled. Each section loosely follows the design discussion of chapter 4. The SQL and PL/SQL referred to are located in Appendix A.

### **5.2.1 Users, user groups, and roles**

A user is any person who uses or visits the CLASS system. Thus, any person interacting with the system is a user. Conceptually, a user can be identified by their name, email address, password or any other provided information. Or, if the user decides to not provide any information, they can be identified by their IP address, browser type, and operating system. However, anonymous user tracking only works for a single user session since the system does not know whether or not the machine is a public terminal and thus used by multiple users. In addition, this method does not allow the system to relate anonymous user sessions to each other since there is no way to determine if it is the same user even if all of the above conditions are the same. If a user wishes to access any private information or make any contribution to the site, they must log in using their email address and password.

User groups are simply ways of associating users with one another. Groups can vary in size from having no members to including every single registered member in the system. A user must be registered with the system in order to be part of a group. Examples of groups are classes, recitation sections, teams, and departments. Within these groups, users have roles. These roles reflect the position of the user within the group.

For instance, a group of type class has roles of Professor, Teaching Assistant, Student, and Dropped.

The CLASS system uses tables from the ArsDigita Community System (ACS) to represent users, user groups, and user group membership. There are several reasons for this. First, these tables are a fairly complete data model for storing demographic, personal, and professional information about people using the system and the groups to which they belong. Second, the ACS relies heavily on these tables and if the educational system does not use these tables it would not be possible to fully take advantage of the many user-centric features of the ACS. Third, the ACS already has an extensive permissions system built using these tables. This existing code gives us a strong starting point for implementing security throughout the system. Finally, starting with a well thought out and tested set of tables greatly simplifies the task of creating a user-centric system. When everything is dependant upon a central users table, it is difficult to create a system centered on anything else.

### **5.2.2 Departments and Subjects**

Departments are a main piece of the CLASS system. Departments contain many subjects, instances of which are classes. In an effort to keep the system user-centric, the system treats departments as user groups. That is, a department is simply a group of users along with a little bit of extra information stored in a helper SQL table.

Departments are groups because a department is really just a group of people all striving to teach the same type of material. In addition, there are needs for items such as

department wide bulletin boards, news postings, and chat rooms. The simplest way to do this is to keep the design user-centric by making departments a user group.

A many-to-many mapping table is used to associate departments to subjects. This method was chosen because most departments will offer many subjects and some subjects will be offered by multiples departments. Classes are grouped into subjects because it is often the case that a class is offered every semester and has information that is common to every offering of the class. Currently, anyone in a department has the authority to create and manage any subject within that department.

In contrast to departments and classes, subjects are not user groups. Rather, a single SQL table within the database represents subjects. Subjects are not implemented as user groups because people do not make up subjects. Instead, a group of classes forms a subject. Most people belong to either an instance of the subject, a class, or to the department offering the subject and therefore there is no direct way to map users to a subject.

### **5.2.3 Classes**

Supplementing physical classes is the main purpose of the educational system. The system does this by modeling the classroom as a database-backed web service where every class is simply a group of users. This model promotes collaboration because it allows every class to have its own bulletin boards, chat rooms, and file storage systems. In addition, classes are able to take advantage of the many other built in features of the ACS.

Every member of a class has exactly one of four roles in the group. Professors are the head of the class and have permission to perform every action within the class homepage. They are able to view all official class documents as well as grades and reviews for everyone. In addition, they have the ability to determine the permissions other roles have within the class homepage. Teaching assistants help the professors teach and can do things such as holding online office hours and grading students. Students can view certain class documents and participate in any open class forums. Finally, there are students that have dropped the class. They remain part of the class user group mainly for record keeping purposes.

Dividing the users into roles not only simplifies the implementation of the permissions but it also allows the system to collect certain information about the user depending on the role of the user. For instance, the system requests a student identification number for every student in the class. Normally, this is simply the student's university identification number that can be used by the professor when reporting grades. However, it does not make sense to collect this information for a professor. Instead, if the user is a professor or a teaching assistant, the system will request information such as office hours, phone number, and office location. This is just one of the ways the user interface is customized for the user depending on what his role is within the class.

#### **5.2.4 Sections and Teams**

Sections and teams are two areas of classes that are vital to collaboration. Representing sections within the system allows section leaders to post information specific to that section and it also promotes collaboration in smaller, more intimate groups. And, the easiest way to allow this is by modeling sections as subgroups of classes. They are stored in the database as user groups with their parent group identification number referencing the class to which the section belongs.

Teams are an essential part of a collaborative educational system because team assignments and projects are an integral part of many traditional classes. And, like sections, teams are simply represented as subgroups with the class as the parent. This allows team members to upload private team information as well as private and public status reports. In addition, teams can easily set up private bulletin boards, chat rooms, and file storage folders.

#### **5.2.5 Assignments, Exams, and Class Projects**

As was discussed in section 4.2.3, the CLASS system stores all information pertaining to assignments, exams, and class projects in the same database table. For an individual assignment, the information is stored in the table and the system knows what to do from there. However, when the assignment or project is to be done by a team, a helper table is needed. This helper table was originally a table that mapped user groups (teams) to projects and held a small amount of extra information such as a description of the particular project the team is working on. However, we ran into several problems with this implementation. Mainly, we realized that it is sometimes the case that multiple

teams work on the same project. This would mean that our mapping table of class projects to teams would have to be many to one. However, that would require storing a lot of redundant information. So, the final decision was to make each project group a user group with some extra information. This had several benefits. First, if a team of three people were working on the project, it is straightforward to copy that group into the project group. Second, if two teams merged for whatever reason, the second team can simply be copied into the project group and all comments about individual groups before the merge stay in tact. Finally, this scheme provides a cleaner implementation for displaying and processing grades in the case where the same team worked on multiple projects.

## **5.3 Security**

The online education system contains a large amount of sensitive information such as personal information about every single user, student grades, and private handouts such as exams before they are administered. Therefore, the system goes through great measures to verify that users are who they say they are so that it may keep sensitive information out of the hands of people who should not see it. This section outlines how some of the security is implemented.

### **5.3.1 User identification**

In order for the education system to be secure, the system must be completely confident that the user requesting a page has been properly identified. When the user first enters the site, they are asked to input their email and a password. This information is

then sent over a secure (SSL) connection. When the system receives this information, the password is encrypted and checked against the encrypted password stored in the database. If the two match, the system grants the user access.

Once the user has access, the system must be able to save this state so that it does not repeatedly ask the user for their email and password. To do this, the server stores a cookie on the user's machine. This cookie contains a random length token string and is only sent over SSL. This cookie is only sent to a client once and the token string is different every time a user logs in. Thus, there is no possible way that two users could end up with the same token string. A second, insecure token is used for user access when the connection is not encrypted. Thus, if the session is not encrypted, the system assumes that the information is not vital and so allows the token identification strings to be passed in the clear. When the session is secure, the secure token string is used to identify the user thus guaranteeing that the secure token is only sent over an encrypted connection when it is sent back to the server.

One final step is taken to ensure correct user identification. Every 20 minutes (this number is configurable), the system will issue another token string and invalidate the original string. Thus, even if someone were to sniff the insecure token, it would only work for less than 20 minutes.

The major weakness of this scheme is that, while information sent during a session is secure, the user verification mechanism is less than ideal. The CLASS system does not currently prevent anyone from using someone else's email address for a short period and registering as him or her. In that case, the intruder could potentially have

access to confidential information (especially if the intruder posed as a department administrator). To achieve a high level of security, the CLASS system will need to be upgraded to use an authentication process with a trusted root such as the Kerberos principal at MIT. This is discussed more section 8.1.1.

### **5.3.2 General methods**

Now that the system knows who the user is, it must also be able to determine which class or department the user is trying to access. This information is stored in an SQL table that is keyed off of the token string used to identify the user. This way, the system does not have to issue a cookie for every group for which the user is a member. In addition, storing this information in the database means that the pages will not have to pass this identification number through the URLs.

When a user requests a class or department page, the system obtains the user identification number as well as the associated group identification from the database. Then, every page takes this information, along with a specified action, and performs a check to make sure that the user has permission to perform a given action. For instance, one action is “View Student Grades.” In order for a user to see the grades of all of the students in the class, the user has to have a role within the class that has explicit permission to perform the “View Student Grades” action. The procedure executing on every page takes the action, user identification number and group identification number as arguments and verifies that the user has the correct role in the group to perform the specified action. If they do not, an appropriate error message is displayed.



A second type of verification that is performed uses row level permissions. That is, every secure row in the database has a separate permissions record. Specifically, it is possible for the creator of a document to determine who can and cannot see it by setting the permissions record. The default user interface allows the row administrator to grant other read, write, comment, or administrative privileges to individual users, groups of users, or users with a certain role within a group. This is convenient when the system needs to determine the permissions of users based on what information they are trying to retrieve rather than what page they are trying to view.

### **5.3.3 File permissions**

The CLASS system wants to promote user collaboration among friends taking the same class as well as among instructors and all other users. One major part of this collaboration is the file system. Collaborative document development is one of the important benefits of using this system. As such, the system needs to make sure that the authors of the document can privately collaborate on the document, only making it public when it is complete.

There are several different approaches that can be taken to solve this problem. First, every single document uploaded can be considered a different file. This is not an ideal solution because when a document is developed collaboratively, there are many versions of the same document and the database and user interface should reflect this fact. The second option is to place permissions on every file. Again, this is not a good idea as it is not normally the case that authors want to allow all of the people viewing a final draft to also be able to view the first draft. A third solution, one taken by the

CLASS system, is to have a separate permissions record for every version of every file uploaded to the system. This way, it is possible to collaborate on a document with only a few people while showing the final version to the general public.

Now that every version has its own permissions record, the problem is determining how to provide users with a simple interface. The CLASS system has greatly simplified the user interface for use by instructors. When instructors upload files, they are only asked to set read and write permissions. And, they are only give the options of “Only Professors,” “Professors and Teaching Assistants,” “All members of this class,” and “General Public.” This way, users of the CLASS system are not exposed to the complexity of groups and roles. In addition, if a professor wishes to collaborate on a document with someone that is not in the class, the professor still has the option to go into the class folder in the file storage system and manually give the user permission to edit the document.

#### **5.3.4 Auditing information**

In a collaborative development environment, documents evolve and information changes. In order to better track mutating state in these important database tables, the updates and deletes of information in several tables must be audited so they can be recovered if necessary. When information in the database changes, it is useful to know which user changed what information when. Therefore, in addition to copying all of the old data and recording when the change was made, the IP address and user identification number of the modifying user are also recorded in the separate audit table. Oracle automatically inserts data into audit tables upon update and deletion of a row from the

corresponding tables via audit triggers (see data model in Appendix Section A). Currently we are auditing 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.

## **5.4 Usability**

One of the main goals of this system is to provide a usable, personalized environment for the user. This section describes several steps that were taken to try to make the system more user friendly.

### **5.4.1 The Portal: a customizable view of the system**

The central feature of the online education system is the portal page. This is the page that is seen by all users when they first log in. The page contains information such a customizable stock quotes, current weather conditions, a customized calendar (discussed in section 4.2.7), links to class homepages, discussion boards, and chat rooms, and class news. Since all types of users use this portal, every user can decide which of the above items are included on their personal page. In addition, every user can customize the stock and weather sections so that they receive information relevant to them. This portal page, which has been built on top of a core ArsDigita Community System module, can easily be extended to contain more custom information. For instance, if the Sloan School of Management wanted to include a section of headlines from the magazine *The Economist* then it would only take one person a few hours to write a program to retrieve the headlines from the magazine's online site and display them on the portal page. This is

one large advantage this system has over proprietary systems such as WebCT. WebCT cannot be customized because the end user cannot view the code. However, the system described by this thesis is open source so anyone with some time can extend and customize this page.

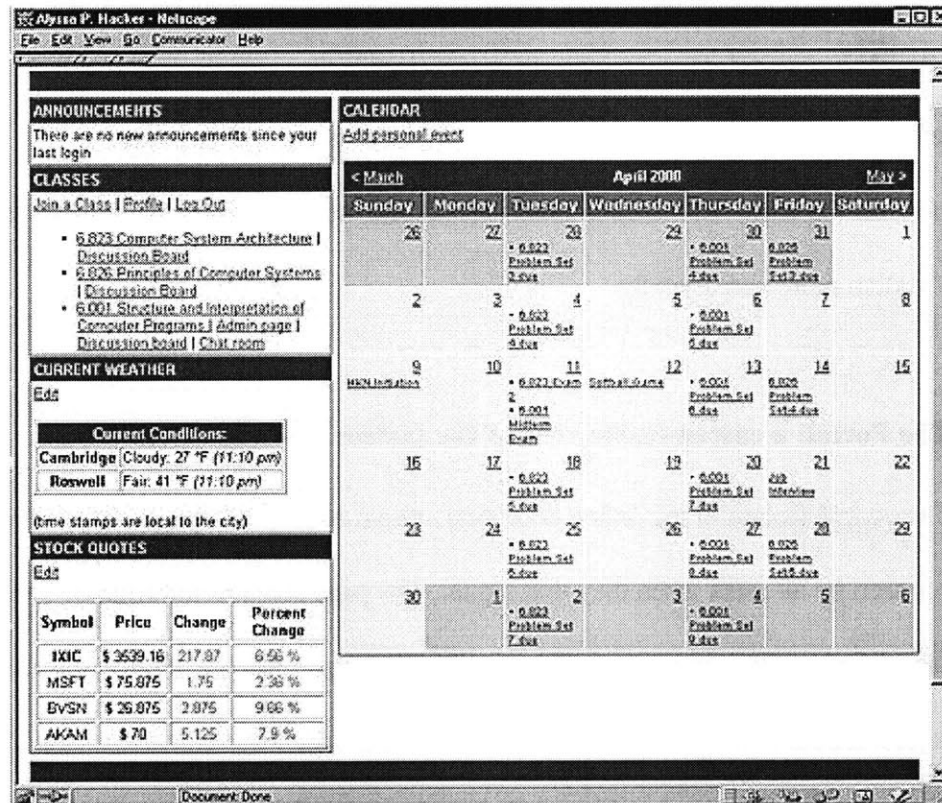


Figure 5.2: A personal education portal. Note that the user is a teaching assistant for 6.001 and therefore has a link to the administration pages. However, since the user is only a student in 6.823 and 6.826, there are not links to the administration pages.

## 5.4.2 Calendar

A customized calendar is a key feature of a user's portal page. The calendar is a modified version of the ArsDigita Community System's calendar module. The modifications that were made enable the calendar to display class-related events and due dates for every class a user is registered for. In addition, a user can add personal events

to this calendar, thus having a central page with all personal and academic information. The calendar acts as an easy entry point to information about assignments and person events. In addition, it allows class administrators to easily add assignments and exams for any day simply by clicking on the number in the calendar.

### **5.4.3 Permanent information access**

One easy way to add usability to a web based system is to provide permanent URLs. A stable Internet requires a stable set of links. Therefore, this system creates user-readable unique permanent URLs for every version of every file uploaded. The CLASS system does not use the scheme provided by the PURL [Onl00] software mechanism because the CLASS system does not require the URL parsing or indirection provided by the PURL software. However, there is no reason that the system could not use the PURL software directly.

Currently, the CLASS system simply requires all file requests to go through a URL filter that parses the URL and returns the appropriate file. The CLASS system assigns a unique integer to every uploaded version. Then, when a user wants to retrieve the version, the URL is simply the name they gave to the file followed by the file extension with the single variable passed to the file being the version identification number. This method provides a permanent, human-readable, unique, able to be bookmarked URL for every file ever uploaded to the system.

# Chapter 6: Feedback

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

The feedback received from 6.916 was mostly positive. The main features used by the instructors were the bulk spamming module with archiving and collaborative grading. The instructors found the system intuitive to use and appreciated the collaborative ease of grading. The only other feature they attempted to use was project administration. This section of the system 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 not intuitive and was very long. 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 automatically sign up to be in their own teams thus removing the burden of having the instructors create the teams.

These problems with creating and management of project teams taught us two things. First, five page views to create one item is too many. Users do not want to have to wait for five separate pages to load in order to add one item to the system. This is

especially true when the user has to go through this process several times in a row. The second thing that we learned from this experience is that projects should be teams rather than mapped to teams. We found that making the projects themselves teams greatly reduced the complexity of the page flow and it also reduced the complexity of pages displaying project information. In addition, this change made it much easier to allow students to sign up for term projects. The initial reasoning for linking teams to projects was that the purpose of a team is to complete a project or an assignment. However, from the feedback gathered from 6.916, we found that there is a need for teams and project groups to exist as separate entities. The reasoning for this is simple. Many classes have group final projects. To model this, we need user groups. However, there are also some classes that have teams start on projects and then join efforts and there are also classes that have small teams that 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 a projects type of user group. We chose the latter because it simplifies the page flow as well as grade calculations although both of the alternatives would have worked.

In contrast to 6.916, the feedback we received from the Ecommerce and Architecture class was mostly negative until the end of the semester. This class was a small, discussion-oriented seminar that did not have problem sets or tests. The class was set up such that the students had to write a paper every couple of weeks and meet with the instructor during the other weeks to discuss past papers as well as the student's progress on their current paper. Students used the site mainly to view their grades on assignments as well as viewing new assignments and downloading handouts. The teaching assistant used the site for distribution of materials as well as some grading. The instructor only

used the site sporadically and gave us no feedback. Whenever he needed something done with the site, he would ask the teaching assistant to do it.

The Ecommerce and Architecture class provided us with valuable feedback about our design and has helped us shape the CLASS system into a better-rounded system. The first thing that we learned was that our original design was heavily biased towards technical classes with weekly problem sets and it was not broad enough to support small, seminar based classes. Specifically, the site was not nearly as fostering of collaboration as we would have liked. Some of the core features desired by the class were missing. For instance, the original system did not have a way for students to automatically register for office hours. For large technical classes, posting a time and room is enough because so many students that show up that it is not feasible to have appointments. However, this class was small and the professor allowed the students to schedule thirty-minute appointments with him. Another feature that this 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 be able to see the comments without the name of the student that posted the comment, the teaching assistant and professor should be able to see all comments and all names and other students should only be able to see their own comment. We never even entertained this scenario in our design although once it was brought up it seemed reasonable.

The second lesson we immediately learned from this class was that users that are not familiar with computers or do not speak English well had a difficult time navigating the site. We learned this because the teaching assistant fell into both categories while



some students fell into each. The main problem was that the functionality provided by the links was not clear enough. For instance, the teaching assistant would add an assignment and upload a corresponding file. Then, to change the corresponding file, she would go into the file system and get lost instead of finding the assignment and using that interface to perform the change. She was getting lost in the file storage system because it contains permissions information as well as information about when the file was uploaded and the file type; all information that was previously hidden to her. The teaching assistant repeatedly made this same mistake for assignments, lecture notes, and handouts. Therefore, one of the first modifications we made to the system was removing the link to the file storage system. Once this was done, she was able to find the appropriate assignment and easily edit it. Another area that confused both the teaching assistant and instructor was the use of teams. This feedback was consistent with the feedback we received from 6.916 and the remedy was the same.

One final thing we learned from the teaching assistant is that she wanted a way to allow all grades for an assignment to be hidden to the students until all students were graded at which point she would be able to, with one press of a button, reveal all of the grades. This is actually a feature that we included in our original design but did not get around to implementing. From this feedback, we found that this is actually a useful feature.

The feedback we received from the students was not nearly as informative. The only strong opinion they expressed was that they wanted easier access to their grades. That is, the original design of the system required that to see a grade, you had to view

information about an assignment. However, students wanted the ability to view all of their grades, along with class distribution and averages, on a single, consolidated page.

As we started incorporating the feedback into the design, user feedback steadily became more positive. We added things such as automatic scheduling of office hours, we removed the link to the raw file storage system, and we reworded a lot of the messages a user saw. In addition, we modified the workflow for adding teams and projects. Finally, we removed the requirement that all assignments have an associated file.

The experience of having two drastically different pilot classes taught us a lot. First, the feedback has taught us that making a user-centric system does not guarantee that the system will be usable. In addition to being user-centric, the system needs to be general enough to meet the needs of most users and classes while specific enough to be useful. This is a difficult medium to achieve but is definitely something that must be considered when the system is being designed. Second, we found that different classes are structured in different ways and that there is not a general mold that all classes will nicely fit into. We learned that getting feedback early and often from potential users is a good way to keep the design from being too biased towards one type of class. And, we learned that simplicity is best. If providing a really neat feature that is only somewhat useful to the end user greatly complicates the user interface then it is best to not include the feature at all. This was especially apparent with exposing the end user to the raw file storage system.

# Chapter 7: Potential effects of the system

## 7.1 Competitive Analysis

As was outlined in chapter three, there have been many different online educational systems created before this one and there will be many more created after this one. How does this system compare in functionality and usability to the many other systems on the market? Appendix A provides several tables comparing the functionality of this system to the functionality of fourteen commercial systems. As can be seen, the CLASS system possesses a large amount of the functionality provided by the commercial systems. However, due to time constraints, it does not include all of the possible functionality. For instance, the CLASS system does not include large blocks of functionality such as an online testing system. In addition, there are many features within the system that could be made customizable that currently are not.

While the CLASS system does not provide all of the functionality provided by most other systems, it does make information more accessible to users. The system does this in two ways. First, it provides a portal page that contains links to information relevant to the user for the current month. In addition, it provides direct access to class pages, bulletin boards, and chat rooms. None of the systems reviewed in Chapter 3 contain this much information on a single page. Rather, most current systems force the user to visit each class homepage to retrieve this information. The second way the CLASS system makes information more accessible to users is through the class homepage. This system automatically notifies the user when a new handout or assignment has been added to the homepage. In addition, links to information such as handouts, exams, and quizzes only

show up if there is something to see on the target page. Many of the current software packages show these links even if there is nothing on the other end, thus leading the user to a dead end. A final way the homepage provides more information to the user is it will randomly select a student or faculty member from the class and display their name and picture, therefore "forcing" people to meet other people in the class. This same portrait functionality is used on the student information page. If a student or faculty member in the class wants to find information about another student, they are not only able to view the student's name, email, and portrait but they also see a list of bulletin board items they have posted. Currently, most other commercial products do not offer this functionality.

In addition to trying to provide the user with more easily accessible information, the CLASS system also tries to foster community and collaboration, thus allowing everyone to learn from everyone, not just the teaching assistants. This is one large thing that is missing in many of the commercial systems on the market today. Through the portal page, the system allows the user to access almost any information they need within only a few clicks. This includes giving the student the option to view the most recent postings on class bulletin boards and chat rooms for all of a student's classes. This is a major improvement over most commercial systems that require the user to first go the class home page then to the bulletin board to view the most recent postings.

## **7.2 Why this system is different**

As was discussed in section 3.1, there are many different competing types of online educational systems that provide many different collaborative tools to the internet. The CLASS system, however, is different because it is user-centric. It has been designed

with the user in mind rather than a single class. In addition, it provides many features that help reduce administrative tasks, and increase the collaborative potential of the system. The CLASS system helps reduce administrative tasks by delegating the work down to the department and class level. For instance, to request a class on the Command system, the instructor or TA must fill out a form and then someone must do a check to make sure the information is correct and approve it. Since the approval is probably someone going to a page and pressing a button, it is not a big deal. But, if a single user had to do this for hundreds of classes during a one or two day period, it would become a very time consuming task. The CLASS system, on the other hand, allows each department to administer this information. So, when a class is added, they simply have to click on a few links and the class is there. Currently, this method requires more work to create a single class. However, if there were 300 classes over six departments, this would delegate the work to the administrative staff of six separate departments as opposed to a single one. In addition, once a professor has taught a class one semester, they have the authority to add the class for the next semester. After several semesters, the administrative burden placed on the departments will be minimal.

In addition to trying to slowly delegate administrative tasks to the end user, this system tries to promote collaboration through bulletin boards, chat rooms, and a customizable portal page. The customizable portal page is a feature not available in most other educational systems and sets the CLASS system apart. The unified portal page allows users to customize what information they see as well as where on the page they see it. Some options for the user include a consolidated calendar with hyper linked personal information integrated with class schedules, recent information from class

bulletin boards and chat rooms, as well as a variety of other items such as recent stock quotes and local weather information. The selection of items 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 system administrator could elect to make that information available on the portal page. Placing this information on the portal page provides the user with access to most information about any class within a single click. This feature not only provides the user with a lot of information but it also pushes the bulletin boards and chat rooms in front of the user by showing them the latest information and placing them one click away from joining in on the discussion.

### **7.3 Use by universities**

This educational system has been built for MIT's Sloan school of Management. They have requested a highly customizable, highly collaborative educational environment that will allow their professors to easily create interactive web sites to supplement traditional lectures. In addition, Dartmouth's Tuck School of Management has also expressed interest in using the system to run all of its classes. Finally, ArsDigita University will be using this software to administer all of its classes. All three schools are looking to use the software beginning the Fall 2000 semester. In addition to these three schools, Olin University and the University of San Francisco are also interested in using this online education system to enhance the student learning experience.

# Chapter 8: Conclusions

After examining many existing educational packages and then designing and building our own, two points of interest remain. First, we must discuss what remains to be done to turn this system into a robust enterprise solution suitable for use by the major universities mentioned in section 7.3. Second, we must examine what we have learned from the experience of building such a large educational system.

## 8.1 Future Work

The future work described here gives a broad overview of a few of the many enhancements that need to be made to convert this educational system into a world-class enterprise solution. Some of the proposed changes, such as online testing, personalization of material, and the data mining used in creating reports for both instructors and students 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 as is described in section 5.3.1. While this method, when used with the other measures

described in section 5.3, is secure, it does not guarantee that users are who they say they are. The current implementation does not prevent someone 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 such a way that the administrator of the system can easily determine whether or not they want to use them.

### **8.1.2 Increased functionality for departments**

The current system currently uses departments as a means to group subjects together. In reality, however, departments are much more complicated than just a group of subjects and the system should reflect this. 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 members for the departments. 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 adding roles to department user groups. The one drawback to adding more roles is that it makes managing the system more complex. And, as we learned from the Ecommerce and Architecture 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**

The educational system was built with the goal of providing an easy to use collaborative environment to complement physical classes. However, only a small number of collaborative tools have actually been incorporated into the system. There are many more tools that could be integrated or developed that would make this system an even more fruitful place to learn.

The ArsDigita Community System (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 discussion about it should be located directly on the assignment information page. In addition, the system should be tied into the ACS wimpy point module that 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 before problem sets or tests.

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 be able to tell the professor which students are not doing well or that students did not understand a particular section of a particular lesson. 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, the online education system 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 would easily be able to convey equations to each other. And, if students are allowed to take "snap shots" of the white board then they will have a record of exactly the instructor wrote on the board. In some instances, this could be a significant improvement over having the student frantically copying the drawing while trying to comprehend the material.

#### **8.1.4 Reports**

The system currently tries to provide new capabilities to the instructors of individual classes by providing detailed information about each particular user. However, it does not currently provide department administrators or other faculty with new capabilities. For instance, there is no easy way for a department administrator to see a list of classes for the current semester with the enrollment. In addition, department administrators should be able to view the history of a subject. They should be able to view 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 would be useful for academic advisors that 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 was discussed in section 4.1, people are not going to use the system unless it is extremely easy to use and it makes their life easier. To better increase the usability of the system, study groups could be asked to use it and make suggestions. In addition, as more classes start using the system, there will be user interface changes that will need to be incorporated.

In addition to this invaluable user 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 information concerning classes or students or even class material such as online tests. In addition, it would be convenient for instructors if they could easily download student grades into a spreadsheet. Students should be allowed them 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 certain graphics or will have certain other 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 will provide professors with the ability to easily change the look and feel of their course homepage without changing the look and feel of all of the other homepages in the system. A templating system would also allow departments to easily 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 because it would allow teaching assistants to create practice tests that students could take online. Students could then use the results to evaluate their current knowledge in the class. In addition, online tests provide a significant new capability for instructors, as it would allow them to administer tests online. 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 automatically graded.

### **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 that need to be implemented to make this a highly usable and flexible system. For instance, the system should employ data mining techniques to help in course management. For example, instructors should have the option of grading tasks on a per problem basis. This way, professors could easily tell that most students did poorly on problem two of assignment six. This would be an easy way for the system to help professors improve their assignments. Another example of data mining would be to provide the professor with an automatically generated list of students doing poorly in class or a list of students that 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 the system can easily write their own module and “plug it in.” Finally, the system should provide support for syncing the portal calendar with the user’s palm pilot or even another portal, such as MyYahoo.

## **8.2 Initial system complete**

There are many different online educational resources available today. However, there are not many that are free and open source. And, there are even fewer that are centered on the user. This project has been a success even though it has not completely met all of its original goals.

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 that it is possible to integrate collaborative tools within an educational system. We have been able to provide important information to users with a minimal number of page loads and we have been able to delegate administrative responsibility onto users of the system. We have not been able, however, to meet our goal of strong security. Rather than spending time on implementing client certificates in AOLserver, we have spent our time on developing and designing other features. Not achieving this goal, however, is not a major setback because there are stopgap measures that could be taken to reach this goal.

For instance, the system could run `mod_AOLserver` within Apache, which does have certificates.

The goal of fostering collaboration within a user-friendly system is much harder to evaluate as there is no easy way to evaluate this without the system experiencing widespread use. The system has the potential to foster collaboration through the many collaborative tools provided by the system so this could be considered to be a success. The usability of the system, however, is a place that needs a great deal of work. As discussed in section 8.1, there are many features that could be added that would enhance the usability of the system. In addition to these features, however, we have found that the existing user interface should be modified. As discussed in chapter 7, a pilot class at MIT's Sloan School of Management found some areas of the system intuitive and while they found other areas of the system to be cryptic. While we have tried to make the more cryptic sections more usable, there is no way to reliably test this until the system is used by a larger variety of users.

Despite these shortcomings, and the long list of future work, the project is a success as it shows that it is possible to create a user-centric, open source system that is capable of supporting a simple class. We have also found that anyone designing a future user-centric system must take great care to make sure the user and groups of users are at the center of every feature of the system. If a designer ever stops doing this, the system immediately shifts towards a class-centric design and a significant amount of new user capabilities are lost. This was made most apparent to us when our initial implementation of class projects revolved around projects for a class rather than projects as user groups.

We had great difficulty creating a user-friendly workflow with this model. However, once we modeled project teams as groups, the workflow became apparent.

As outlined in section 8.1, there is still a lot of work to be done to turn this into a highly usable and flexible system. Many of the improvements will be challenging to make but now that a solid user-centric base has been created, future development will have a strong base to build upon. We hope to be able to complete many of these 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 will be able to use this system next fall.



# Appendix

## 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, 12, and 15, 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

<b>DEVELOPMENTAL FEATURES</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>
Content format will allow for simple transfer to/from another vendor's platform	Y	Y	Y	Y	Y	Y						Y	Y		Y
Platform uses open data standard so that it can communicate with existing university database applications	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y		Y
Content can be authored on PCs running Windows 95/98/NT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Content can be authored on Macs running OS 7.5 or greater	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Courses can be taken using a PC running Windows 95/98/NT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Courses can be taken using a Macintosh running OS 7.5 or greater	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Platform provider is supportive of implementing IMS standard within product	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y
Platform provider is supportive of implementing AICC standards within product	Y	Y	Y	Y	Y	Y	Y			Y		Y	Y	Y	Y
Platform utilizes standard HTML for content creation	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Platform is structured so students can view all of their current courses when they log on	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Platform's server software will run on DEC Unix	Y	Y		Y			Y	Y					Y		Y
Platform's server software will run on Windows NT	Y	Y	Y		Y	Y	Y		Y	Y	Y	Y	Y	Y	Y
Multiple choice questions can be created\scored with platform's authoring software		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			Y	
True\False questions can be created\scored with platform's authoring software		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			Y	
Matching questions can be		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			Y	

created\scored with platform's authoring software																	
Short answer questions can be created\scored with platform's authoring software		Y	Y	Y	Y	Y	Y	Y			Y					Y	
Essay questions can be created\scored with platform's authoring software		Y	Y	Y	Y	Y	Y	Y	Y	Y						Y	
Platform supports question database for management of test questions		Y		Y	Y	Y	Y		Y	Y						Y	
Platforms supports reporting features for test questions		Y	Y	Y	Y	Y	Y		Y	Y	Y					Y	
Platform supports Microsoft Internet Explorer 4.x and newer browsers	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			Y	Y	Y	
Platform supports testing stage for courses to debugged before making them live to students	Y	Y	Y	Y	Y	Y	Y	Y			Y			Y	Y	Y	
Platform allows author to view course as student without logging out	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y				Y	Y	Y	
Platform has built-in threaded discussion list capabilities	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			Y	Y	Y	
Platform has built-in chat capabilities	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y	Y		
Platform can be integrated with Real networks video and audio products		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			Y	Y	Y	
Platform can be integrated with Macromedia Shockwave products		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			Y	Y		
Vendor provides development services	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y			Y	Y		
Management component will create reports for tracking student progress	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y					Y	
Platform has a feature to import existing test questions in a tab-delimited format		Y	Y		Y	Y	Y				Y						

## A.2 Instructor Tools

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

### A.3 Instructional Features

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

#### A.4 Student Tools

STUDENT TOOLS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Authentication	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Bookmark management	Y		Y		Y	Y	Y		Y	Y		Y	Y	Y	
Multimedia support		Y	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y	Y
Private e-mail			Y	Y	Y	Y	Y	Y	Y	Y	Y		Y		
File submissions	Y	Y	Y	Y	Y	Y	Y		Y	Y			Y	Y	Y
Threaded discussions	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y
Course Chat rooms	Y	Y	Y	Y	Y		Y		Y	Y		Y	Y	Y	
Logged chat	Y	Y	Y	Y	Y		Y			Y		Y		Y	
Whiteboard		Y	Y	Y	Y		Y			Y					
Self-assessing		Y		Y	Y	Y	Y		Y		Y			Y	
Progress tracking	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y			Y	
Desktop based file management for uploading to server	Y			Y			Y		Y	Y			Y	Y	Y
Study skill building		Y		Y	Y	Y	Y		Y		Y		Y	Y	
Un-timed quizzes		Y		Y	Y	Y	Y		Y	Y	Y			Y	
One question-at-a-time function		Y		Y	Y	Y	Y		Y		Y				
Bulletin board/conferencing tools	Y	Y		Y	Y	Y	Y		Y	Y		Y	Y	Y	Y
Image database	Y			Y	Y		Y							Y	
Student access to own grades	Y	Y		Y	Y	Y	Y			Y	Y	Y	Y	Y	Y
Access to course grade distribution				Y	Y		Y						Y		
Automated glossary tool				Y	Y	Y	Y				Y	Y	Y		
Automated index tool				Y	Y		Y		Y					Y	
Online assistance		Y		Y	Y	Y	Y		Y	Y	Y		Y	Y	
Search tool for course content	Y	Y		Y	Y	Y	Y		Y	Y	Y	Y		Y	
Student presentations area	Y	Y		Y	Y	Y	Y			Y			Y	Y	
Allows user to view all class events on one consolidated page once logged in.	Y				Y			Y			Y				
Student can make private annotation of material.						Y	Y					Y			
Student Homepage Tool.		Y		Y			Y	Y							

## A.5 Administrator Tools

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

## B Data Model

This is the data model that is used to drive the online education system described by this thesis. Many of the tables reference tables from the ArsDigita Community System. For the following tables, please see [http://software.arsdigita.com/www/doc/sql/users, user\\_groups, fs\\_files, fs\\_versions, user\\_group\\_type, user\\_group\\_type\\_fields, user\\_group\\_type\\_member\\_fields, country\\_codes, states, and portal\\_pages](http://software.arsdigita.com/www/doc/sql/users, user_groups, fs_files, fs_versions, user_group_type, user_group_type_fields, user_group_type_member_fields, country_codes, states, and portal_pages).

```
-- 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,  
    -- for schools like MIT where each department has a number  
    department_number       varchar(100),  
    -- we'll generate a home page for them but if they have one already  
    -- we can provide a link  
    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 default sysdate not null,  
    last_modifying_user     references users,  
    modified_ip_address     varchar(20)  
);
```

-- 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,
    :old.mission_statement,
    :old.last_modified,
    :old.last_modifying_user,
    :old.modified_ip_address);
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 pages

```

```

declare
    n_departments_group_types integer;
begin
    select count(*) into n_departments_group_types from user_group_types where
group_type = 'edu_department';
    if n_departments_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','full');
    end if;
end;

```

```
insert into user_group_type_fields (group_type, column_name,
pretty_name, column_type, column_actual_type, column_extra, 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, column_extra, 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, column_extra, 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, column_extra, 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, column_extra, 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, column_extra, 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),
    -- 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),  
    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,  
        credit_hours,  
        prerequisites,  
        professors_in_charge,  
        last_modified,  
        last_modifying_user,  
        modified_ip_address)  
    values (  
        :old.subject_id,  
        :old.subject_name,  
        :old.description,  
        :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
-- (IAP = MIT's Independent Activities Period)

-- 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),
    -- at MIT, something like 'Room 4-231, TR 1-2:30'
    where_and_when     varchar(4000),
    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_exam_p          char(1) default 'f' check (final_exam_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),
    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_exam_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,
    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.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
        (group_type, 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 (group_type, 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 (group_type, 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
        (group_type, column_name, pretty_name, column_type, column_actual_type,
        column_extra, sort_key)
        values
        ('edu_class', 'start_date', 'Date to Start Displaying Class Web Page', 'date', 'date', "", 3);

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

        insert into user_group_type_fields (group_type, column_name, pretty_name,
        column_type, column_actual_type, column_extra, sort_key)
        values
        ('edu_class', 'description', 'Class Description', 'text', 'varchar(4000)', "", 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', 'where_and_when', 'Where and When', 'text', 'varchar(4000)', '', 6);
```

```
insert into user_group_type_fields (group_type, column_name, pretty_name,  
column_type, column_actual_type, column_extra, sort_key)  
values  
( 'edu_class', 'syllabus_id', '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 recieve 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', 'teams_p', 'Will the class be divided into teams?', 'boolean',
'char(1)', 'default "f" check(teams_p 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', 'exams_p', 'Will the class have exams?', 'boolean', 'char(1)', 'default "f"
check(exams_p in ("t", "f"))', 14);
```

```
insert into user_group_type_fields (group_type, 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_group_type_fields (group_type, column_name, pretty_name,
column_type, column_actual_type, column_extra, sort_key)
values
('edu_class', 'exams_folder_id', 'Exams Folder', 'integer', 'integer', 'references fs_files',
16);
```

```
end if;
end;
/
```

```
-- create a view for current classes whose webpages we should display
-- to students
```

```
create or replace view edu_current_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'
and sysdate<edu_class_info.end_date
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', 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', 'varchar(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', 'varchar(100)', ", 3);
end if;
end;
/

-- we want to create a view to it is easy to retrieve information about sections

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 lets 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;
begin
    select 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 mashed 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 is a percentage (0 to 1)
    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 so we know if anyone changes anything
```

```
create table edu_grades_audit (  
    grade_id          integer,  
    grade_name        varchar(100),  
    class_id          integer,  
    comments          varchar(4000),  
    -- weight is a percentage  
    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
);

```

```

create sequence edu_task_sequence;

```

```

-- includes assignments, projects, exams, and any other 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 the
    -- user pages
    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, etc)
    last_modified     date,
    -- could be date assignment is due, or date of an exam
    due_date         date,
    -- this references the fs_files file_id 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_grade_p char(1) check (requires_grade_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,
date_assigned,
last_modified,
due_date,
file_id,
weight,
requires_grade_p,
online_p as electronic_submission_p
from edu_student_tasks
where task_type='project'
and active_p='t';

```

```

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,
online_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,
online_p as electronic_submission_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 (
    student_id          references users,
    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,
        :old.file_id,

```

```

        :old.last_modified,
        :old.last_modifying_user,
        :old.modified_ip_address);
end;
/
show errors

-- 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               not null references user_groups,
    -- must have student_id or team_id
    student_id             references users,
    team_id                references user_groups,
    task_id                references edu_student_tasks,
    -- there may be several times during the term that the professor
    -- wants to evaluate a student. So, the evaluation_type
    -- is something like 'end_of_term' or 'midterm'
    evaluation_type        varchar(100),
    grader_id              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,
    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
);

-- 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,
        last_modifying_user   integer,
        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,
        :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 project. It is possible  
-- to have one term project but many instances of that project. For  
-- instance, "Final Project for 6.916" is a term project that would  
-- be kept in the edu\_tasks table but ArfDigita.org is a project  
-- instance that would be kept in this table. There is a many to  
-- one mapping

-- we make task\_id not null because every project 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 sequence edu_project_instance_id_seq start with 1;
```

```
create table edu_project_instances (
    project_instance_id      integer not null primary key,
    project_instance_name    varchar(200),
    project_instance_url     varchar(500),
    -- which project is this an instance of?
    project_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 project.
    project_type              varchar(10)
                             default 'team' check(project_type in ('user','team')),
    min_body_count           integer,
    max_body_count           integer,
    -- we want to be able to "delete" project instances so we have active_p
    active_p                  char(1) default 't' check(active_p in ('t','f'))
);
```

```
-- we want to be able to assign students and teams to projects
```

```
create table edu_project_user_map (
    project_instance_id      integer not null references edu_project_instances,
    team_id                  integer references user_groups,
    student_id               integer references users,
    constraint edu_project_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))
);
```

```
-- lets make it easy on Oracle to access this table
```

```
create index edu_project_map_idx on edu_project_user_map(project_instance_id,
team_id, student_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. Thus, 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 should reference user_group_roles(role)
    role              varchar(200),
    -- what the class wants to call the role
    pretty_role       varchar(200),
    pretty_role_plural varchar(300),
    -- sort key for display of columns.
    sort_key          integer not null,
    -- this is to capture info about the hierarchy of role permissions
    priority          integer,
    primary key (group_id, role)
);
```

```
-----
-----
--
-- begin the portal tables
--
-----
-----
```

```
-- the portal mini-tables
```

```
create sequence weather_id_sequence;
```

```
create table portal_weather (
    weather_id        integer not null primary key,
    user_id           not null references users,
    city              varchar(100),
    usps_abbrev       references states,
    zip_code          varchar(10),
    -- 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')),
    current_p         char(1) default 'f' check (current_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 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_id_sequence;

-- the viewable column that specifies whether the calendar
-- entry is viewable by the public and if so, whether we should show the
-- title or something in place of the title (e.g. Busy, Free, Tentative --
-- MS Outlook options). Also, the owner column identifies who
-- the entry is for: so we can display calendars with respect to individual
-- users or groups of users (like in a team)

create table edu_calendar (
    calendar_id     integer primary key,
    category        not null references calendar_categories,
    -- the way we connect calendar entries to users
    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')),
    -- first day of the event
    start_date     date not null,
    -- last day of the event (same as start_date for single-day events)
    end_date       date not null,
    -- day to stop including the event in calendars, typically end_date
    expiration_date date not null,
    -- 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 we want the state code
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
);

```

```

-----
-----
--
--          BEGIN PL/SQL
--
-----
-----

```

```

-- we need a trigger to populate the edu_role_pretty_role_map
-- this is included in case people want to add new roles to
-- the class all they have to do insert into user_group_roles
-- and this will take care of the rest

```

```

CREATE OR REPLACE TRIGGER edu_class_role_update_tr
AFTER UPDATE OF role ON user_group_roles
BEGIN
    -- we want to update the existing row
    update edu_role_pretty_role_map
    set role = :new.role
    where group_id = :old.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 corresponding
-- 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,
            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 delete 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 role = :old.role;

```

```

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_table_id portal_tables.table_id%TYPE;
weather_table_id portal_tables.table_id%TYPE;
classes_table_id portal_tables.table_id%TYPE;
announcements_table_id portal_tables.table_id%TYPE;
calendar_table_id portal_tables.table_id%TYPE;
begin
select table_id into stock_table_id from portal_tables where

```

```

        table_name='Stock Quotes';
select table_id into weather_table_id from portal_tables where
    table_name='Current Weather';
select table_id into classes_table_id from portal_tables where
    table_name='Classes';
select table_id into announcements_table_id from portal_tables where
    table_name='Announcements';
select table_id into calendar_table_id 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_table_id, 1, 'l');
insert into portal_table_page_map
(page_id, table_id, sort_key, page_side)
values
(:new.page_id, weather_table_id, 2, 'l');
insert into portal_table_page_map
(page_id, table_id, sort_key, page_side)
values
(:new.page_id, classes_table_id, 1, 'r');
insert into portal_table_page_map
(page_id, table_id, sort_key, page_side)
values
(:new.page_id, announcements_table_id, 3, 'l');
insert into portal_table_page_map
(page_id, table_id, sort_key, page_side)
values
(:new.page_id, calendar_table_id, 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

- [Aca95] Academic Resources for the E-Generation. July, 1995.  
<<http://www.studyfree.com/>>
- [Ame95] America OnLine. AOLserver. America OnLine, <http://www.aolserver.com>, 1995.
- [Ars00] ArsDigita Corporation. The Arsdigita Community System. 2000.  
<<http://software.arsdigita.com>>
- [Bar94] Barron, Ivers, Sherry. Exploring the Internet. *The Computing Teacher*, 22(2), 14-19. 1994.
- [Bla00a] Blackboard CourseInfo 4.0 Technical White Paper, UNIX Edition, February 2000, Washington, DC <<http://company.blackboard.com/CourseInfo/index.html>>
- [Bla00b] Blackboard.com Home Page <<http://www.blackboard.com/>>
- [Cli90] Clifford, R. Foreign languages and distance education: The next best thing to being there. (ERIC Document Reproduction Service No. ED 327 066). 1990.
- [Com00] Comparison of Online Course Delivery Software Products, Marshall University, April 2000.  
<<http://multimedia.marshall.edu/cit/webct/compare/comparison.html>>
- [Cou98] Course Management and Delivery. Massachusetts Institute of Technology, August 1998. <<http://command.mit.edu/>>
- [Eco00] Ecollege.com Homepage, Denver, Colorado, 2000. <<http://www.ecollege.com/>>
- [Edu99] Eduprise, Inc. Homepage, 1999 <<http://www.eduprise.com/>>
- [Ehr96] Ehrmann, S. Looking Backward: U.S. efforts to use technology to transform undergraduate education. University of California. 1996.  
<<http://eee.uci.edu/programs/auctlt/LookingBack.html>>
- [Ell97] Ellis, B. Virtual Classroom Technologies for Distance Education: The case for On-line Synchronous Delivery. In Proceedings of North American Web Developers Conference, Alberta, Canada. October 1997.  
<<http://www.detac.com/solution/naweb97.htm>>
- [Goo96] Goodwin-Jones, Polyson, Saltzberg. "A Practical Guide to Teaching with the World Wide Web." *Syllabus Magazine*, September 1996.  
<<http://www.umuc.edu/iuc/cmc96/papers/poly-p2.html>>



- [Gre99] Greenspun, P. *Philip and Alex's Guide to Web Publishing*. Morgan Kaufmann Publishers. April, 1999
- [Hil90] Hiltz. Evaluating the virtual classroom. In *Online Education: Perspectives on a New Environment*, ed. L. M. Harasim, 133-184. New York: Praeger. 1990.
- [Icu00] Icubed, LLC Homepage, 2000. Demonstratio site for SerfSoft.  
<<http://www.icubed.org/>>
- [IMS00] IMS Global Learning Consortium, Inc. Homepage, April 7, 2000.  
<<http://imsproject.org/>>
- [Mor95] Morten, Flate, Paulsen. *The Online Report on Pedagogical Techniques for Computer-Mediated Communication*. NKI, Oslo, Norway, August 1995.
- [Mur96] Murray W. Goldberg, Sasan Salari and Paul Swoboda, "World Wide Web Course Tool: An Environment for Building WWW-Based Courses", *Computer Networks and ISDN Systems*, 28 (1996)
- [Onl00] Online Computer Library Center. Permanent uniform resource locators.  
<<http://purl.org>, 2000>
- [Ora00] Oracle 8i Homepage, Oracle Corporation. 2000  
<<http://oracle.com/database/oracle8i/>>
- [Pau90] Paulsen. Organizing an electronic college. In *Proceedings of the Third Guelph Symposium on Computer Conferencing*, 87-97. Guelph, Ontario: University of Guelph. 1990.
- [Pau92a] Paulsen, M. A goal-oriented method for establishing an electronic college. In *Impact of Informatics on the Organization of Education*, eds B. Samways and T.J. van Weert, 113-118. Amsterdam: Elsevier. 1992.
- [Pau92b] Paulsen, M. F. *From Bulletin Boards to Electronic Universities: Distance Education, Computer-mediated Communication, and Online Education*. 1992. University Park, Pennsylvania: The American Center for the Study of Distance Education. Pages: 67. 1992.
- [Sca84] Scardamalia, Bereiter. Computer support for knowledge-building communities. *Journal of the Learning Sciences*, 3(3), 265-283. 1984.
- [Sch96] Schutte, J. *Virtual Teaching in Higher Education: The New Intellectual Superhighway or Just Another Traffic Jam?* 1996.  
<<http://www.csun.edu/sociology/virexp.htm>>

- [Ser00] SerfSoft Corporation Homepage, Fred T. Hofstetter, 2000 <<http://serfsoft.com/>>
- [She96] Sherry, L. Issues in Distance Learning. *International Journal of Educational Telecommunications*, 337-365. 1996
- [Vir00] Virtual Learning Environments, Inc. Homepage, 2000. <<http://www.vlei.com/>>
- [Web00] WebCT Homepage, 2000. <<http://www.webct.com/>>
- [Woo97] Woodruff, Brett, Macdonald, Nason. Participation in knowledge- building communities to promote teaching competency in mathematics. Paper submitted to The Canadian Society for the Study of Education. 1997  
<http://www.oise.utoronto.ca/%7Eewoodruff/cp.htm>