



---

Computer Science Senior Projects

Computer Science

---

12-15-2010

## CAESked: A Class Scheduler for WMU Students

Chris Fruin  
*Western Michigan University*

Jerry Grochowski  
*Western Michigan University*

Follow this and additional works at: [https://scholarworks.wmich.edu/cs\\_seniorprojects](https://scholarworks.wmich.edu/cs_seniorprojects)



Part of the Computer Sciences Commons

---

### WMU ScholarWorks Citation

Fruin, Chris and Grochowski, Jerry, "CAESked: A Class Scheduler for WMU Students" (2010). *Computer Science Senior Projects*. 1.

[https://scholarworks.wmich.edu/cs\\_seniorprojects/1](https://scholarworks.wmich.edu/cs_seniorprojects/1)

This Report is brought to you for free and open access by the Computer Science at ScholarWorks at WMU. It has been accepted for inclusion in Computer Science Senior Projects by an authorized administrator of ScholarWorks at WMU. For more information, please contact [wmu-scholarworks@wmich.edu](mailto:wmu-scholarworks@wmich.edu).



# **Senior Project Final Report**

## **CAESked**

A Class Scheduler for WMU Students

**Project Team:** Chris Fruin and Jerry Grochowski

**Client:** Dr. Karlis Kaugars

CS 4910

Instructor: Dr. Kapenga

Dec. 15, 2010

## **Abstract**

Scheduling classes is a tedious process for WMU students. CAESked, a class scheduling web application, was created to ease this process. The programming languages PHP and JavaScript were used to build a solution which integrates WMU data for upcoming courses into a web application. With features including a weekly calendar, a campus map, and class information all in one place, CAESked provides students with an easy to use tool for adding classes to their schedule.

# Table of Contents

<b>1. Introduction.....</b>	<b>4</b>
<b>2. Background .....</b>	<b>6</b>
<b>3. Design Decisions .....</b>	<b>26</b>
<b>4. Design .....</b>	<b>29</b>
<b>5. Implementation .....</b>	<b>31</b>
<b>6. Testing.....</b>	<b>40</b>
<b>7. Security .....</b>	<b>42</b>
<b>8. Summary.....</b>	<b>44</b>
<b>9. Appendices.....</b>	<b>45</b>
<b>9.1 Appendix A: Spikes.....</b>	<b>45</b>
<b>9.2 Appendix B: Stories .....</b>	<b>47</b>
<b>9.3 Appendix D: Requirements Stories .....</b>	<b>49</b>
<b>9.4 Appendix E: Standards .....</b>	<b>50</b>
<b>9.5 Appendix F: Non-Disclosure Agreement .....</b>	<b>51</b>
<b>9.6 Appendix G: Ownership.....</b>	<b>52</b>
<b>9.7 Appendix H: User Testing Form .....</b>	<b>53</b>
<b>10. Bibliography .....</b>	<b>56</b>
<b>11. Glossary .....</b>	<b>58</b>
<b>12. Index.....</b>	<b>63</b>



## 1. Introduction

This paper describes the CAESked (pronounced “C”-“A”-“E”- “Sked”) project, a web-based class scheduler for Western Michigan University (WMU) students. The Background, Design Decisions, Design, Implementation, Testing, and Security of the project will all be described. Additional details, including proof of concept tests (“Spikes”) and descriptions of features (“Stories”) are included in the Appendices. The remainder of the Introduction will describe in more detail what will be found in each section of this paper. The paper will explain the details, decisions, and process involved in creating CAESked, a web application designed to solve a problem that thousands of WMU students face every semester – scheduling classes.

The Background section will present the problem that led to the decision to create a student scheduling web application. The client who requested the application, Prof. Karlis Kaugars, will be introduced; the current options students have for scheduling will be evaluated; and potential alternatives will be weighed. Additionally, other background information, such as the availability of data, will also be discussed.

The Design Decisions and Design sections will explain the design of CAESked. In the Design Decisions section, important design choices that were made will be discussed. For example, the programming languages chosen, and the reasons for their choice, will be covered. In the Design section, the overall design layout will be discussed. While the Design Decisions section gives specific choices that were made, the Design section will present how the application was structurally composed. For example, the three-tier architecture approach, and how it applied to this project, will be presented in the Design section.

The Implementation section presents the outcome of the design. The finished application and significant aspects of building it will be covered. Also, a guided tour using screenshots will

show the finished application in action.

In the Testing section, several kinds of testing performed on the application will be discussed. These are unit testing, functional testing, user testing, and web browser compatibility testing. Unit testing helped ensure the individual pieces of our application (methods) will function correctly, while functional testing helped ensure that the application as a whole will function correctly. The use of thorough testing helped prevent bugs in the final release and helped verify that the application will be reliable.

In the Security section, the steps which were taken to protect personal data and prevent malicious attacks will be described. This includes the use of HTTPS encryption, and the steps taken to prevent LDAP injection and Cross-Site Scripting attacks.

The Appendices include Spikes, Stories, Requirements Stories, and Standards. Spikes are tests that were used to prove a concept before much time was put into adding a feature to the actual project. This enabled us to limit time invested and risk on potential features before integration into the main application. Stories allowed us to define each feature the user needed. Requirements Stories defined the type of support needed, including web browsers and operating systems. The coding standards defined coding styles to help reduce the chance of errors that poor coding standards cause.

CAESked, the Computer Aided Engineering Center's web scheduler, is an application which required detailed research, discussion, and design to implement properly. The remainder of this paper will discuss in depth the aspects involved in creating a web application that solves the problem of quickly, easily, and accurately producing a student's schedule.

## 2. Background

The Background Section will discuss background information relevant to the scheduling problem and its solution. This includes details about the client and the problems with the present situation. In addition, alternative solutions, custom web applications created by other schools, and data availability from WMU will be covered.

The client for the project is Dr. Karlis Kaugars. He is an Associate Professor of Computer Science at Western Michigan University, as well as the Director of IT and Facilities for the College of Engineering and Applied Sciences (CEAS). One of his major research interests is visualization, and he is known on campus as a GUI expert. Prof. Kaugars has identified a gap in the resources provided by WMU for assisting students in the process of scheduling classes.

Currently, students can use two main resources provided by WMU to decide on a schedule for an upcoming semester. The first is the registrar's online catalog, and the second is the student administrative system ("GoWMU/Banner"). The online catalog only offers course descriptions, including prerequisites and corequisites (at <http://catalog.wmich.edu/>, See Fig. 2.1). Through "GoWMU", students have access to the second major resource: Banner student administrative system (See Fig. 2.2). Registration for classes is done through Banner. Students also have one minor resource to identify what courses should be taken: speaking to an advisor a new student can be given a list of recommended courses to take, but this does not create a schedule, it just helps prioritize what courses should be included in the schedule.

Unfortunately, none of the provided resources are geared towards planning a student's weekly class schedule. Banner is designed for registering - it is not meant to help organize a weekly calendar as a whole. It's also clunky to use, especially for comparing classes, because

it's designed to look at details for only one potential class at a time. Both Banner and the catalog are especially lacking when it comes to identifying scheduling conflicts.

[WMU > Registrar > Academic Catalogs](#)

[WMU A-Z List](#) | [Contact WMU](#)

The screenshot displays the Western Michigan University Academic Catalog website. At the top, there is a navigation bar with links for WMU HOME, ABOUT WMU, ACADEMICS, ADMISSIONS, ALUMNI & FRIENDS, and STUDENT LIFE. Below this is a search bar with a "Search WMU" button and links for "Advanced Search" and "People Search". The main content area shows the date "Apr 26, 2010" and a dropdown menu for "Undergraduate Catalog 2009-10" with a "GO" button. On the left side, there is a "Catalog Search" section with a dropdown menu set to "Courses" and the search term "cs1110". Below this is a "Catalog Search" section with a "GO" button and a "HELP" button. The search results section is titled "Catalog Search" and "Add To Portfolio". It shows "Search Results [Modify search options.]" and "Courses - Prefix/Code Matches". The results are for course prefix "CS" and/or course code "1110". The best match is "CS 1110 - Computer Science I". There are links for "[Add to Portfolio]" and "[Print Course]". The course description for "CS 1110 - Computer Science I" is: "A first course in the science of programming digital computers. Analysis of problems and development of correct procedures for their solution will be emphasized along with the expression of algorithmic solutions to problems in a structured high level computer language. Applications will solve both numerical and non-numerical problems for the computer." The prerequisites and corequisites are: "Prerequisites & Corequisites: Prerequisite: MATH 1180 (or higher) (may be taken concurrently)." The credits are: "Credits: 4 hours". The notes are: "Notes: Enrollment is restricted to undergraduates and those graduate students admitted under the PCS (Permission to take Computer Science) classification." The when offered is: "When Offered: Fall, Spring".

Figure 2.1 - Online Catalog: [catalog.wmich.edu](http://catalog.wmich.edu) has course descriptions and prerequisites.



Back to Student Home Tab



**Subject:** ADA/SPADA  
Acad Talented Youth Program  
Accountancy  
Aeronautical Engineering

**Course Number:**

**Title:**

**Schedule Type:** All  
Independent Study  
Individual Learning  
Lab or Discussion

**Instructional Method:** All  
Compressed Video Semester  
Hybrid  
Online Semester

**Credit Range:**  hours to  hours

**Campus:** All  
English Second Language  
Main  
WMU - Battle Creek

**Course Level:** All  
English Second Language  
Graduate  
Undergraduate

**Part of Term:** Non-date based courses only  
All  
Full Term

**Instructor:** All  
Aardema, Robert J.  
Abbott, Kevin W.  
Abdel-Qader, Ikhlas

**Attribute Type:** All  
Area I: Fine Arts  
Area II: Humanities  
Area III: U.S. Cultures

**Start Time:** Hour  Minute  am/pm

**End Time:** Hour  Minute  am/pm

**Days:**  Mon  Tue  Wed  Thur  Fri  Sat  Sun

Figure 2.2 - GoWMU allows searching for sections.

## Flowchart of a Student’s Scheduling Process

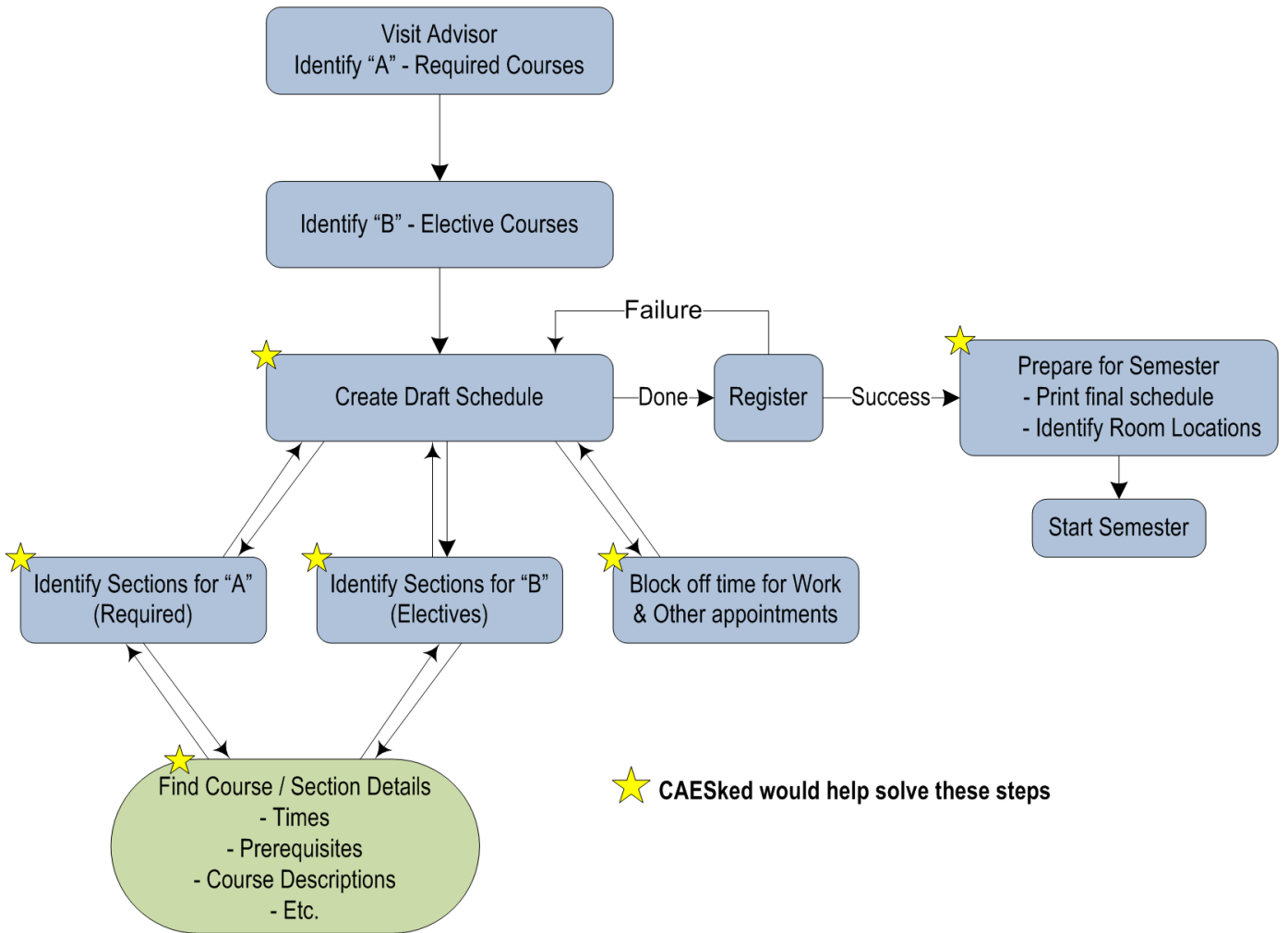


Figure 2.3 – The student class scheduling process.

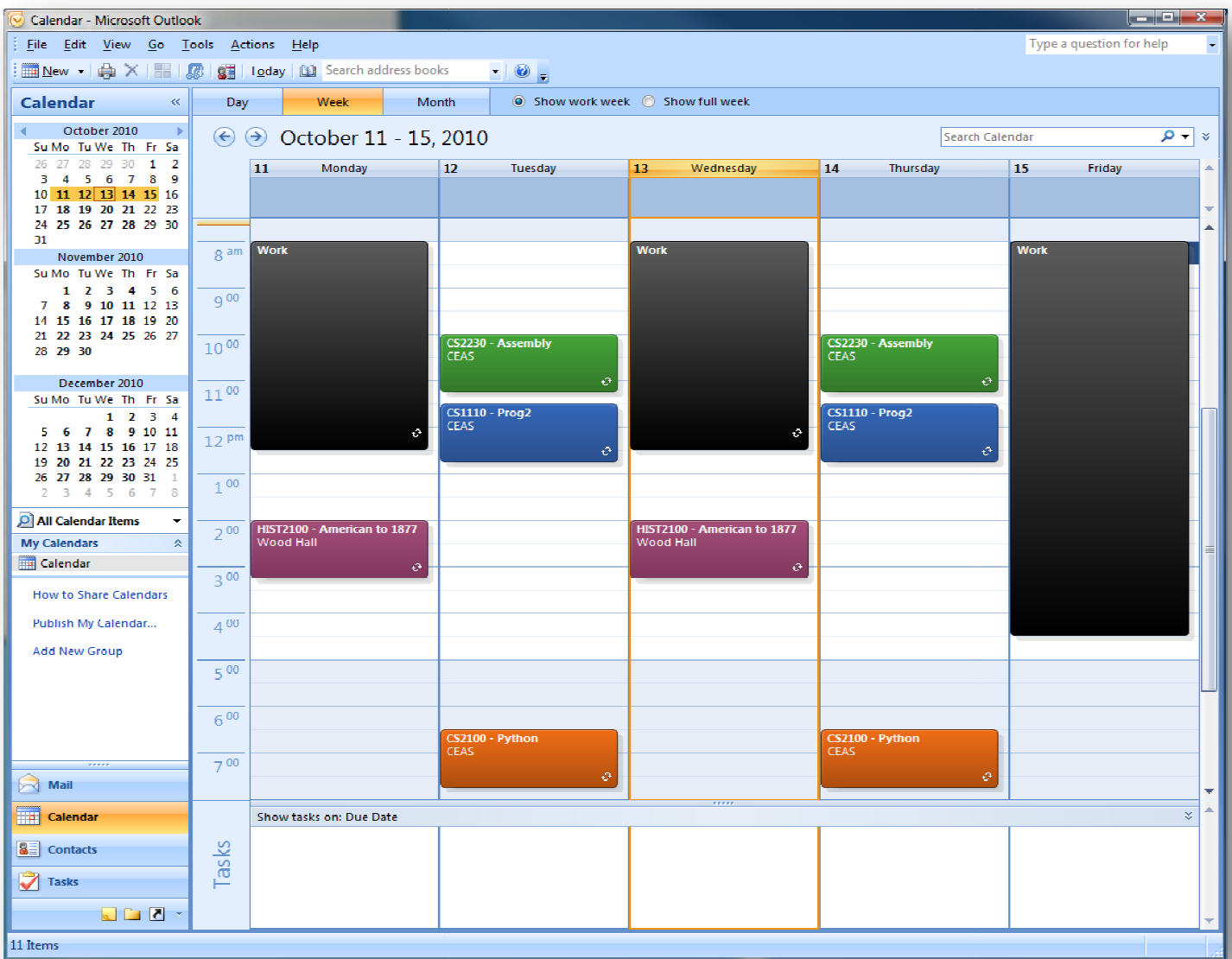
The typical steps a student may take when scheduling classes is shown in Fig. 2.3. (Note that this is a simplified diagram and different students will take different steps, but this diagram illustrates many of the necessary processes.) First, a student should see an academic advisor and identify the required courses they should take next for their major and minor. Then, a student should identify the elective courses they wish to take to round out their schedule. After identifying the possible courses a student wants to take (possibly through the registrar’s catalog),

they begin creating a draft weekly schedule. They may wish to block of time for work or other weekly appointments. Then they should identify sections and section times for each of their required courses (normally using GoWMU/Banner). Following that, they may identify sections from their electives that fit in their remaining open time. There is a lot of give and take in creating a draft schedule, making changes as necessary to avoid conflicts. But after a student has created a draft schedule, they may then try to register for those sections.

The diagram in Fig. 2.3 demonstrates that drafting a weekly schedule is central to scheduling classes, but WMU doesn't offer a weekly scheduling tool. This is a hole that needs to be addressed. (In fact, each of the starred items in Fig. 2.3 can be made easier with CAESked.)

Currently, one approach that students take to identifying scheduling conflicts is to simply record the prospective times with paper and pencil and layout times on a hand-drawn weekly calendar. This approach is less than ideal. If changes need to be made to the schedule, it leads to a lot of erasing. Also transferring data from what the student looks up on GoWMU and copying it down to the paper can lead to errors. Another fault of the paper system is, the paper copy can be damaged, or poor handwriting can make it difficult to read the schedule when the student tries to register for the courses. A process that thousands of students undertake should not be so tedious.

A second approach is for students to try to adapt scheduling software for their classes. The software used includes Google Calendar, Microsoft Outlook Calendar, and similar applications (See Fig. 2.4). The software approach is more flexible: it overcomes the problem with making changes, but there is no integration with WMU's class data. One still has to track down course times and fill them into the software for each course, which is still tedious and leads to errors. Furthermore, it doesn't automatically identify whether prerequisites have been met.



*general scheduling for*

Several features are needed which aren't presently available:

- Detailed information for multiple classes at the same time
- A modifiable week's schedule
- Automated scheduling conflict warnings
- Automated retrieval of classes already signed up for
- Campus map for locating classes
- Prerequisite Checking



The key feature for a potential solution would be a simple weekly calendar, allowing time for classes to be blocked off. The solution should also provide detailed information about sections. If a student chooses a class and section, they should be able to add it to their calendar with the application automatically identifying scheduling conflicts. For example, if two courses are within 15 minutes of each other, but on separate campuses, a flag should be raised. The application, if possible, should also: determine if course prerequisites have been met, retrieve already enrolled classes to automatically populate the calendar, and allow students to block off time for prior engagements (e.g. work). Additionally, a campus map could be generated displaying where a student's classes are located. In fact, there are a number of features that a potential solution could provide to make the scheduling process easier and more effective for students.

To meet these needs, several alternatives were considered, including new software and upgrades to current software. The most logical option would be to use software already designed for this purpose. The problem is that any application would have to be tailored to WMU to integrate with current systems. That limits the commercial options to the provider of WMU's current system: Sungard Data Systems, Inc. Sungard's Banner student system (Unified Digital Campus) is one of the current resources already used by students (accessed through GoWMU). In addition, WMU also uses Sungard's Luminis Platform to manage the GoWMU student portal. Neither of these software systems offers the necessary scheduling functionality. Luminis does not offer a separate software solution meeting the client's requirements. Other schools have faced some of the same problems and we have examined their solutions.

A number of schools have taken the same path this project is taking. That is, individual schools and CS departments have taken it upon themselves to develop *ad hoc* scheduling

assistance software, available as a web application. Many example schools could be discussed here, but we will only briefly mention three. The examples we will examine are from Rutgers University, Iowa State University, and the University of Colorado Denver.

Rutgers University provides a video tutorial (<http://vimeo.com/5479120>) of their “Course Schedule Planner” (<https://sims.rutgers.edu/csp/>). Its features include automated schedule generation, a calendar, and a list view (See Fig. 2.5). Iowa State University calls their solution a “Class Schedule Planner” (<http://planner.iastate.edu/planner.jsp>). It includes section selection and calendar generation as well (See Figs. 2.6 and 2.7). The University of Colorado Denver has created a similar “Course Schedule Planner” (<http://courses.cudenver.edu/>, Figs. 2.8, 2.9, and 2.10). Each of these planners provides functionality similar to what the client has in mind, but a solution tailored to meet WMU's individual needs and to work with its specific data systems is required.

**RUTGERS** Course Schedule Planner (CSP)

Select Courses | Select Sections | Build Schedule (0 Found) | Saved Schedule (0 Saved)

Find Courses

Select a Campus / Location ---

Select a Level of Study ---

Select a Subject Area ---

SEARCH -- Add from Wish List

Selected Courses Credits: 0

No Courses Selected

### Courses Available for Fall 2009

Please "Find Courses" first.

#### Instructions

1. ← "Find Courses" available for current semester.
2. Add courses to your "Selected Courses" list.
3. 1 Check "Build Schedule" tab to verify possible schedules.
4. Click "Build Schedules" button to generate schedules OR
5. Go to "Select Sections" tab to specify preferred sections for your "Selected Courses".
6. Review possible schedules by clicking on "Build Schedules" button or "Build Schedule" tab.
7. Browse through possible schedules in calendar view or switch to list view for more details.
8. While browsing, provide a label and "Save" a schedule for future review or registration.

This video highlights some of the features of the Course Schedule Planner

Figure 2.5 - Rutgers' "Course Schedule Planner".

IOWA STATE UNIVERSITY  
**Class Schedule Planner**

Semester

Fall 2010

Clear

Load

Save

Generate Schedules →

Department

CLASSICAL STUDIES  
 COMMUNICATION DISORDERS  
 COMMUNICATION STUDIES  
 COMMUNITY & REGIONAL PLANNING  
 COMMUNITY DEVELOPMENT  
 COMPUTER ENGINEERING  
**COMPUTER SCIENCE**

Class

COM S 101 ORIENTATION  
 COM S 103 COMPUTER APPLICATNS  
 COM S 104 INTRO TO COMPUTERS  
 COM S 107 APP COMPUTER PROGRAM  
 COM S 203 CAREERS IN COM SCI  
**COM S 207 PROGRAMMING I**  
 COM S 227 INTRO OR IT ORIENTED

+ Add Class

Total Credits: 5.0

Selected Classes

Class	Credits	Sections	Prerequisites and Comments
<input type="button" value="View Sections"/> <input type="button" value="Remove"/> <a href="#">COM S 104</a>	2.0	6	
<input type="button" value="View Sections"/> <input type="button" value="Remove"/> <a href="#">COM S 207</a>	3.0	3	PREREQUISITE: MATH 150 OR PLACEMENT INTO MATH 140/141/142 OR HIGHER COMMENT: CREDIT FOR EITHER COM S 207 OR COM S 227, BUT NOT BOTH, MAY BE APPLIED TOWARD GRADUATION

COM S 104 Sections

Include	Sec	Ref #	Seats	Dates	Days	Start	Stop	Section Comments
<input checked="" type="checkbox"/>	7	2995045	2	08/23-10/15	M W F R	4:10 4:10 4:10	5:00 5:00 6:00	
<input type="checkbox"/>	10	2995060	0	10/18-12/17	M W F R	4:10 4:10 4:10	5:00 5:00 6:00	
<input type="checkbox"/>	11	2995065	0	10/18-12/17	M W F R	4:10 4:10 8:00	5:00 5:00 9:50	

Include Open

Include All

Exclude All

Class information last updated: 2010-10-23 01:35:09

Copyright © 2010, Iowa State University, Ames, Iowa 50011. All rights reserved.

Contact: Office of the Registrar

IOWA STATE UNIVERSITY  
**Class Schedule Planner**

← Back Load Save Print

Reserve Times Sections Schedules

	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9
M							COM S 207				COM S 104					
T																
W			COM S 207				COM S 207				COM S 104					
R											COM S 104					
F							COM S 207				COM S 104					
S																

← ←10 ← 1 of 1 → 10 → →

Seats	Reference #	Course	Section	Title	Days	Times	Meeting dates	Class Comments
14	9235005	COM S 207	1	PROGRAMMING I	M W F W	12:10-1:00pm 8:00-8:50	08/23 to 12/17	CREDIT FOR EITHER COM S 207 OR COM S 227, BUT NOT BOTH, MAY BE
2	2995045	COM S 104	7	INTRO TO COMPUTERS	M W F R	4:10-5:00pm 4:10-5:00pm 4:10-6:00pm	08/23 to 10/15	

Class information last updated: 2010-10-23 01:35:09  
 Copyright © 2010, Iowa State University, Ames, Iowa 50011. All rights reserved.  
 Contact: Office of the Registrar

The screenshot shows the 'UNIVERSITY OF COLORADO DENVER' logo and 'COURSE SCHEDULE PLANNER' text. At the top right, there are links for 'Back', 'Instructor Login', and 'Manage My Plan'. Below the logo, there are four main navigation buttons: 'Search for Courses', 'Review Results', 'View My Plan', and 'Register for Courses'. The 'Search for Courses' button is active. Underneath, there are two search tabs: 'Basic Search' and 'Advanced Search'. A link says 'Looking for more options? Use [advanced search](#).' The search form is divided into several sections: 'Term: Required' with a dropdown set to 'Spring 2010'; 'Day(s) the Courses May Meet' with checkboxes for Su, Mo, Tu, We, Th, Fr, Sa, all of which are checked; 'Format' with checkboxes for 'On Campus', 'Online', and 'Hybrid', all checked; 'Continuing Education' with radio buttons for 'Show All Courses' (selected), 'Only Continuing Ed Courses', and 'Exclude Continuing Ed Courses'; 'College' with a dropdown menu showing 'All Colleges', 'Business School', 'College of Architecture and Planning', and 'College of Arts & Media'; 'Subjects' with a dropdown menu showing 'All Subjects', 'ACCT - Accounting', 'ANTH - Anthropology', 'ARCH - Architecture', and 'BIOL - Biology'; 'Course Level' with a dropdown set to 'All'; and 'Keyword' with a text input field. At the bottom right of the form are two buttons: 'Clear Search' and 'Find Courses'. The footer contains the copyright notice: '© 2006 The Regents of the University of Colorado'.

Figure 2.8 -Colorado Denver’s “Course Schedule Planner”.

[Home](#)
[Instructor Login](#)
[Manage My Plan](#)

## UNIVERSITY OF COLORADO DENVER

COURSE SCHEDULE PLANNER

[Search for Courses](#)
[Review Results](#)
[View My Plan](#)
[Register for Courses](#)

Last updated on Friday, February 19, 2010 at 21:11

**You Searched For**

Term  
**Spring 2010**

Days  
All Days

Subject(s)  
BIOL

[Start a New Search](#)

---

**Refine My Search**

Keyword Search

Course Level  
All

Start Time  
Anytime

End Time  
Anytime

Instructor  
All Instructors

Credit Hours  
All Credit Hours

Only show courses that:

Core Courses

Has Extended Info

Showing Courses 1-20 of 103

Show me  10  20  50 results per page

Sort By



BIOL 1550 section: OLI		Call Number: 14016
 ADD	<p><a href="#">Basic Biol: Ecology &amp; Diversity Of Life:gt-sci</a>                      College/School: College of Liberal Arts and Sciences                      Offered: Online                      Instructor: <a href="#">Regier, Kimberly F.</a></p>	<p><a href="#">Live outside of Colorado?</a>                      Format: Online                      Credit Hours: 4                      Status: <b>Closed</b>                      Seats Allowed: 27                      Seats Available: 0</p>
Note: For students who are not majoring in biology, biology and health career majors should not take this course. ONLINE SECTION COURSE. ADDITIONAL \$100 FEE APPLIES. FOR MORE INFORMATION SEE WWW.COONLINE.EDU OR CALL 303-315-3700.		
BIOL 1560 section: L01		Call Number: 10492
 ADD	<p><a href="#">Lab</a>                      College/School: College of Liberal Arts and Sciences                      Offered: M 10:30a-12:30p at <a href="#">SJ</a> 2115                      Instructor: <a href="#">Fees, Colby P.</a></p>	<p>Format: On Campus                      Credit Hours: 0                      Status: <b>Closed</b>                      Seats Allowed: 24                      Seats Available: 0</p>
	<p><a href="#">Basic Biol: From Cells To Organisms: Gt-sci</a>                      Section: 001                      Offered: M W 01:00p-02:15p at <a href="#">NC</a> 1130                      Instructor: <a href="#">Regier, Kimberly F.</a></p>	<p>Format: On Campus                      Credit Hours: 4                      Status: <b>Closed</b>                      Seats Allowed: 216                      Seats Available: 0</p>
Note: For students who are not majoring in biology, biology and health career majors should not take this course.		

Figure 2.9 - Colorado Denver's "Course Schedule Planner" - Selecting a section.

**BIOL 1560** [Lab](#)  [Remove](#)  
 Section: L04 Call# 10495  
 Meeting: Tu 08:00a-10:00a @ SI 2115

Credits: 0

**BIOL 1560** [Lab](#)  [Remove](#)  
 Section: L07 Call# 10498  
 Meeting: Tu 03:30p-05:30p @ SI 2115

Credits: 0

Total Credits: 4

[Return to Search Results](#)

[I'm Ready to Register](#)

**My Personal Events**

 You have not yet added personal events to your plan for this term.

**Add a Personal Event**

Title  Su Mo Tu We Th Fr Sa Start Time End Time Add Event

07:00am 07:00am

	SUN	MON	TUE	WED	THU	FRI	SAT
7:00							
8:00			BIOL 1560 L04 SI 2115 08:00a				
9:00							
10:00							
11:00							
12:00							
1:00		BIOL 1560 L07 SI 2115 03:30p		BIOL 1560 L04 SI 2115 08:00a			
2:00							
3:00			BIOL 1560 L07 SI 2115 03:30p				
4:00							
5:00							

Figure 2.10 - Colorado Denver's "Course Schedule Planner" - Viewing a schedule.



	Simple Calendar w/ Events	Detailed Info for Courses	Easy to Use for Scheduling	Error Free	Integrated w/ WMU	PreReq Checking	Campus Map of Classes
<b>Banner</b>	No	Yes	No	Yes	Yes	Yes	No
<b>Pencil &amp; Paper</b>	Yes	No	Yes	No	No	No	No
<b>General Calendar Apps</b>	Yes	No	Yes	Yes	No	No	No
<b>Other Schools ad hoc solutions</b>	Yes	Yes	Yes	Yes	No	Yes	No
<b>CAESked</b>	Yes	Yes	Yes	Yes	Yes	Possibly	Yes

Table 2.1- Features Comparison.

A detailed comparison of all available options can be seen in Table 2.1. As seen in the table, “CAESked” should contain all the features of the other scheduling options, possibly excluding prerequisite checking (discussed below), making it a superior scheduling application.

Currently, much of the data necessary for implementing a scheduling web application is available through WMU's LDAP directory service (Lightweight Directory Access Protocol – See Fig. 2.11). A student’s Bronco NetID credentials provide access to details of their record, including their previous and current courses. The same login also allows access to details of upcoming courses, including the date and time of individual sections. All of this is already available through LDAP. In fact, any student could download a free LDAP browser client and browse through this data without anything more than their Bronco NetID and password (and a little knowledge of LDAP). The amount of data that LDAP provides depends on the privileges assigned to the user. Prof. Kaugars has provided us with server access giving more information than a student's credentials allow.

Browser root

- AddressBook
- CSAttendance
- JYV8326
  - o=wimich.edu
    - ou=WMUCourses
      - ou=20101
        - wmuCallNumber**

- LoadClasses
- ZMichigan
- ZOpenLDAP
- ZUnizeto CERTUM

Name	Value
wmuEndDate	1-May-2010
wmuCourseEnrolled	10
wmuStartDate	11-Jan-2010
wmuCallNumber	12675
createTimestamp	20100201011011Z
modifyTimestamp	20100201011011Z
wmuCreditsMin	3.0
wmuCourseCapacity	30
wmuCourseNumber	4900
wmuCourseMeeting	<days>MWF</days><start>11:30</start><end>12:20</end><location>C0141 Parkview Campus B
wmuCourseSubentry	cn=schema
wmuCurricula	Computer Science
wmuCurriculaCode	CS
wmuCreditType	Fixed
wmuInstructorName	John A Kapenga
wmuCourseType	Lecture
wmuCampus	Main Campus
wmuOrgUnitDN	ou=Computer Science, ou=College of Engineering and Applied Sciences, ou=Academic Affairs, o=WMU
title	Soft Sys Dev 1: Reqmts & Dsgn
wmuTerm	Spring 2010
objectClass	top
creatorSimpleName	uid=psynchronization,ou=special,ou=people,o=wimich.edu,dc=wimich,dc=edu
modifierSimpleName	uid=psynchronization,ou=special,ou=people,o=wimich.edu,dc=wimich,dc=edu
objectClass	wimichCourse
wmuInstructorDN	wmuuid=KAPENJAGES6,ou=People,o=wimich.edu,dc=wimich,dc=edu

Successfully connected to dir.wimich.edu  
 Schema has been cached. Using cache...

LDAP Syntaxes: Total: 12 Invalid: 0 Duplicated: 0  
 AttributeTypes: Total: 1964 Invalid: 0 Duplicated: 0  
 LDAPObjectClasses: Total: 368 Invalid: 0 Duplicated: 0  
 MatchingRules: Total: 303 Invalid: 0 Duplicated: 0  
 MatchingRuleUses: Total: 0 Invalid: 0 Duplicated: 0

Ready. For Help, press F1

wmuuid=jyv8326,ou=people,o=wimich.edu,d/Schema loaded

There are two main entry types of interest available through LDAP: students and courses. Detailed information for a student is available through their Bronco NetID (wmuUID). And detailed information for a course section is available through CRN's (wmuCallNumber). See Figures 2.12 and 2.13 for the paths through LDAP to these entries. Inside each entry are attributes containing information details. For example, Fig. 2.14 shows the attributes for a wmicourse object (which is included in a wmuCallNumber entry). However, there are some pieces of data that aren't available through LDAP, even with increased server access.

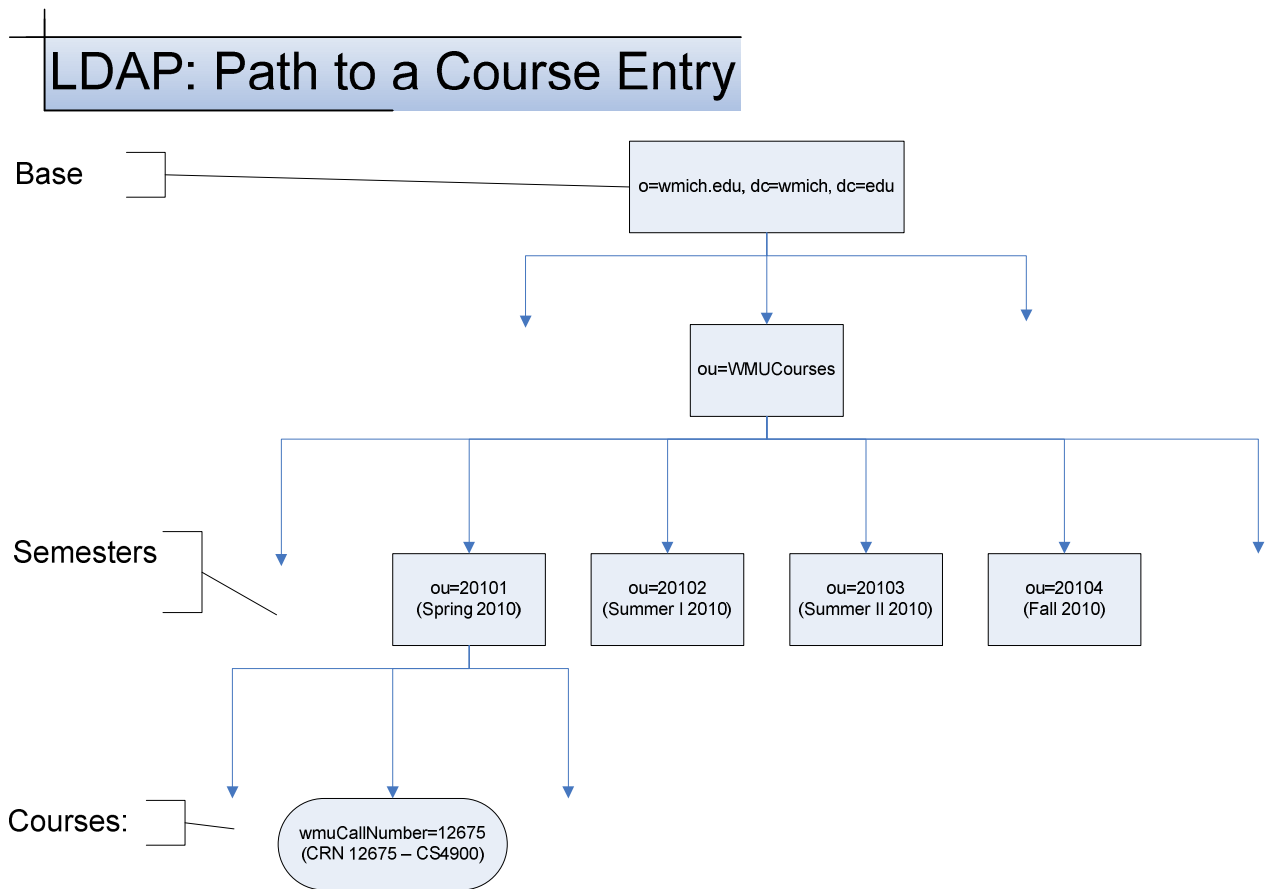


Figure 2.12 - LDAP: Path to a course entry.

# LDAP: Path to a Student Entry

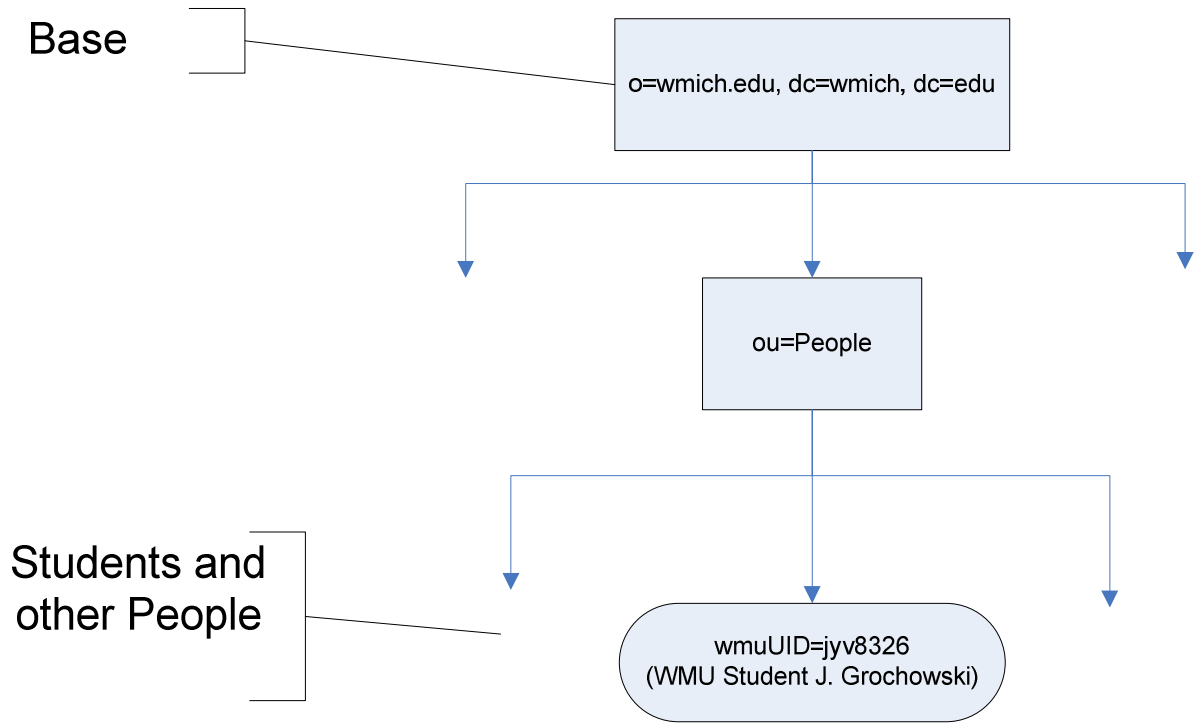


Figure 2.13 - Path to a student entry.

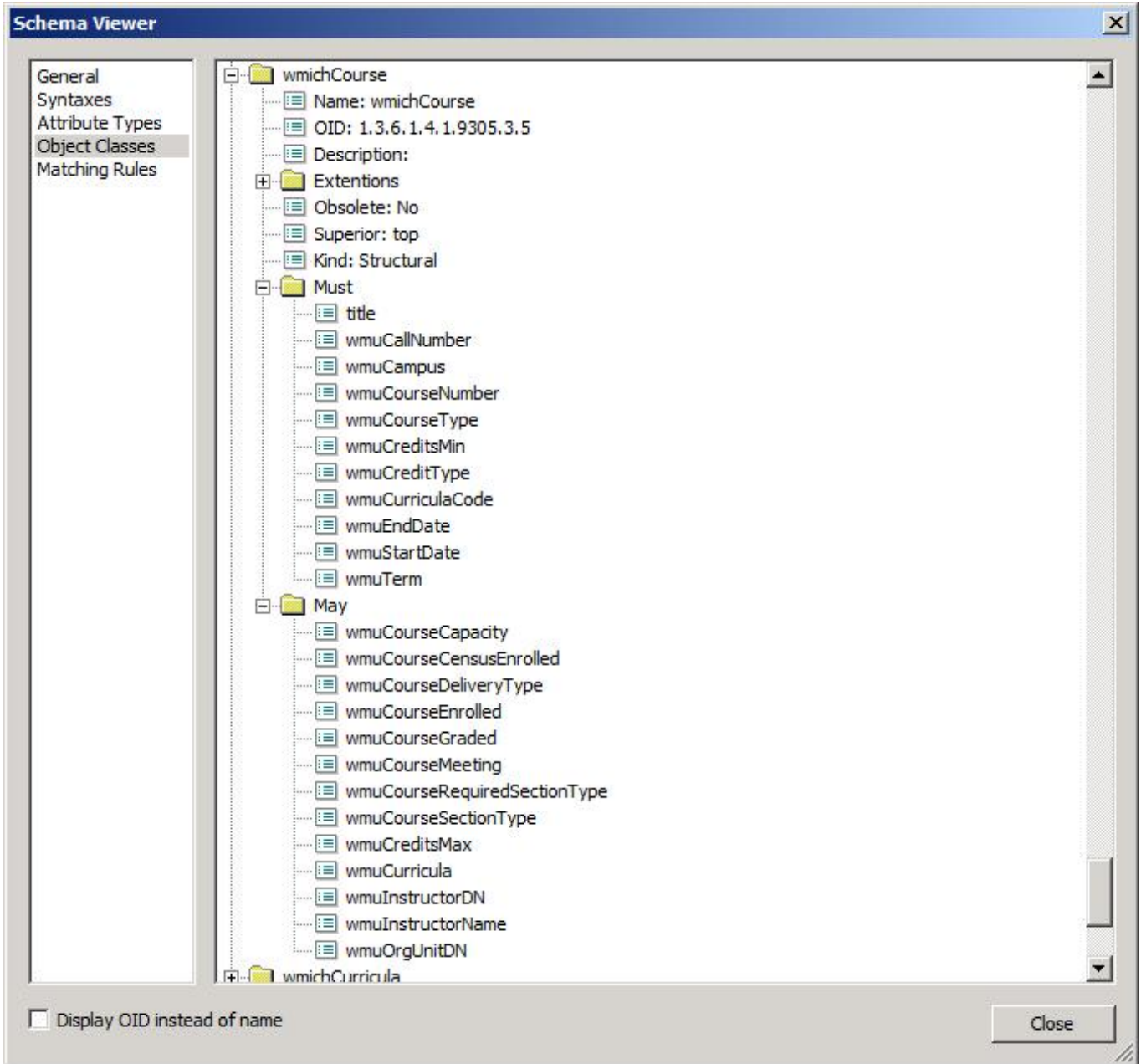


Figure 2.14 - LDAP: attributes for a wmicourse.

Some of the data we would like, especially for verifying prerequisites, is controlled by the Registrar's office through OIT (Office of Information and Technology). For example, LDAP doesn't include the prerequisites for each course. Although that data could be scraped from the

Registrar's online course catalog, it would be more reliable to access the data directly. Also, LDAP doesn't store a student's transfer classes or their class grades. Both are necessary for automatically determining whether a student has met prerequisites. A request to the Registrar was made for access to this information. Although the request has been denied, many of the needed features can be implemented with the available data. There is also the possibility of obtaining the data a different way. It is possible to write a program to go through every course in the catalog and gather the data from the displayed webpages. There are a few drawbacks with this scenario. The written program depends on the format of the catalog webpage; if the page were to change, then the program would no longer work. Also, if the catalog webpage changed the data gathered from the catalog webpage would not update to reflect the change. It would be hard to ensure data accuracy.

The client has many resources to assist in implementing the proposed features. Hardware resources are available through the College of Engineering's IT department. Prof. Kaugars has identified a web server, "Dolby", for this projects use. It runs CentOS Linux and has the necessary software installed. (Current software versions are listed in Appendix D.) In addition to Apache web server software, it has PHP. (The decision to use PHP is discussed further under Design Decisions.) The server has abundant processing power (AMD Opteron 2210) and memory (3.5 GB). Concerning future maintenance of CAESked, the CAE's IT department employs several students as Systems Administrators. These employees are already trained in the upkeep of Linux and Apache. The Systems Administrators will assist Prof. Kaugars in maintaining CAESked, the supporting software, and hardware.

### 3. Design Decisions

The Design Decisions section will outline the choices which were made to guide and build the application. This includes implementation decisions such as programming languages used and data transmission formats, as well as specific decisions such as using LDAP for the main data source.

PHP and JavaScript were chosen as the main programming languages. Prof. Kaugars preferred PHP (“PHP: Hypertext Preprocessor”) for creating dynamic web pages rather than other alternatives such as ASP.NET or JavaServer Pages. PHP was preferred because the client’s office already supports it. Prof. Kaugars also recommended JavaScript. JavaScript is frequently used for building dynamic client-side web applications, such as the front-end for Gmail. JavaScript was used to implement features on the user’s browser such as the weekly calendar. (While PHP creates dynamic web pages to be sent to the user from the *server*, JavaScript allows the page in the user’s *browser* to also respond dynamically, without resending pages from the server.) JavaScript was also used in order to allow sending and receiving data through “AJAX”.

AJAX (Asynchronous JavaScript and XML) allows web applications to function smoothly, like a desktop application. Traditionally, web browsers have required page reloads from the web server every time a web page needs to be updated. But AJAX allows a web page to be updated without reloads. The data requests are handled asynchronously in the background, with the necessary page updates done on the client side via JavaScript in the user’s web browser. (See Fig. 3.1) This allows applications to be built on the web, giving the quick response times users expect.

JSON was chosen for the data interchange format used by AJAX. Traditional AJAX web

applications use XML for data interchange, but our client requested the use of JSON. The advantages of JSON include: it is easier for developers to program with, and it creates faster web applications.

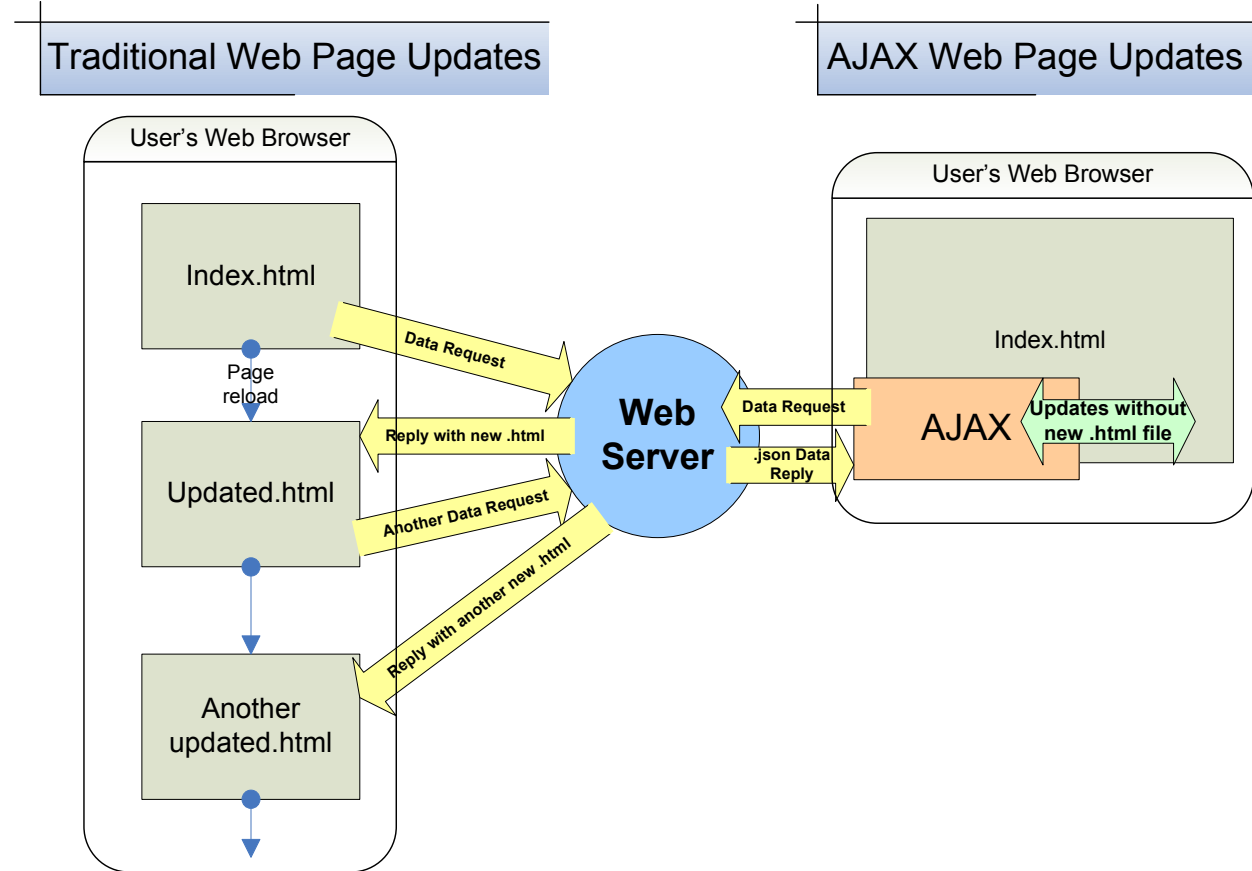


Figure 3.1 - AJAX web applications avoid traditional page reloads.

Prof. Kaugars also requested the application support all modern web browsers. This includes Internet Explorer 7+, Mozilla Firefox 2+, Google Chrome and Safari 3.1+. These browsers make up more than 89% of the browsers currently in use (Refsnes Data, 2010). CAESked has been successfully tested on the above listed web browsers on Windows, Macintosh, and Linux machines (see the Testing section).

LDAP was chosen as the primary data source for Release 1, but MySQL may also be



used in the future. Most of the underlying data originates from WMU’s LDAP directory service. Since data access to the databases used by the Registrar was denied, LDAP was the main resource available. Prof. Kaugars recommended that LDAP be used as much as possible without an additional MySQL database; the primary reason being that keeping an additional database up to date requires a considerable amount of maintenance. A MySQL database may still be needed if additional features are added in the future. For example, a “Save” feature – to allow users to view a saved schedule – would require storing user data in a MySQL database. But for Release 1, LDAP has proven sufficient.

## 4. Design

The Design section will describe the overall layout of the application. It will discuss the Three-Tier Architectural approach used to guide the overall design, how the actual modules in the application fit together, as well as the hierarchy used to organize source code (“Subversion” / version control).

CAESked was designed using the Three-Tier Architectural approach. Three logical layers were used, separating: 1. User Interface, 2. Business Layer (business logic), and 3. Database Layer (LDAP). (See figure 4.1) Using the Three-Tier model, different tiers can be modified or replaced independently of each other. Keeping the layers modular also assists in testing and security.

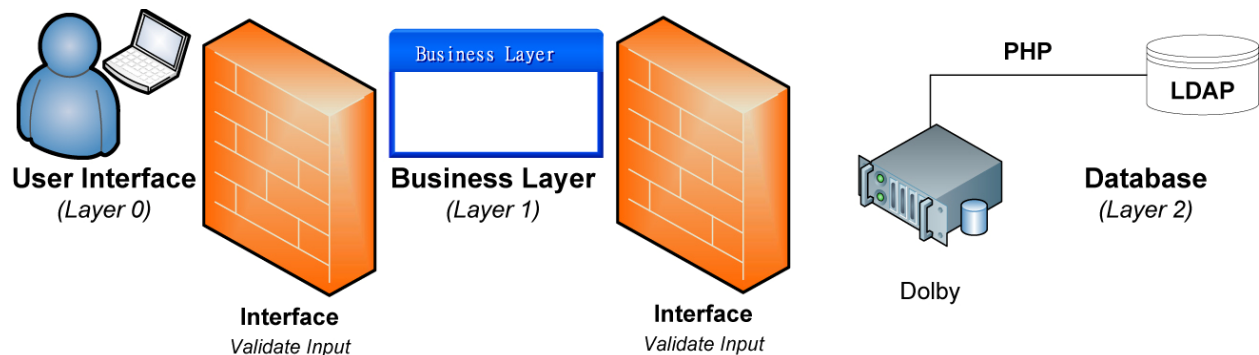


Figure 4.1 – Three-Tier Architecture application layers.

The actual modules of the application fell within the Three-Tier model as follows: The web pages and JavaScript objects seen by the user fell under the user interface. This included the login page (“index.php”), the main page of the application (“caesked.php”), and the calendar itself (“jquery.caeskedweeklycalendar.js”). Processing of class data – loading it into the calendar and user interface – fell under business logic. Business logic was primarily performed by a combination of JavaScript and PHP (including “main.js”, “map.js”, and “backend.php”). The actual calls retrieving data from the LDAP directory service fell under the data tier (including

“sections.php”, “ldapConnect.php”, and “buildings.php”). (The files listed are the main files. Many supporting files are not given here.)

Subversion (SVN) was used for revision control of source code files. Subversion was chosen because its use was covered in CS4900, and because the CS department at WMU offers Subversion accounts. The layout of the project in SVN followed the recommendations given in the course text, “Pragmatic Version Control: Using Subversion, 2<sup>nd</sup> edition” (Mason, 2006). This included using top-level directories for: current main-line work (“trunk”), “branches”, and frozen copies of releases (“tags”). Under “trunk” were directories for documentation (“doc”), source code (“src”), scripts (“util”), and vendor code (“vendor”). Vendor code includes the JavaScript toolkit JQuery. Under the “src” directory were subdirectories for each of the three layers of the application: the user interface (“ui”), business logic (“bl”), and LDAP interface (“ldap”) (See Fig. 4.2, SVN project hierarchy).

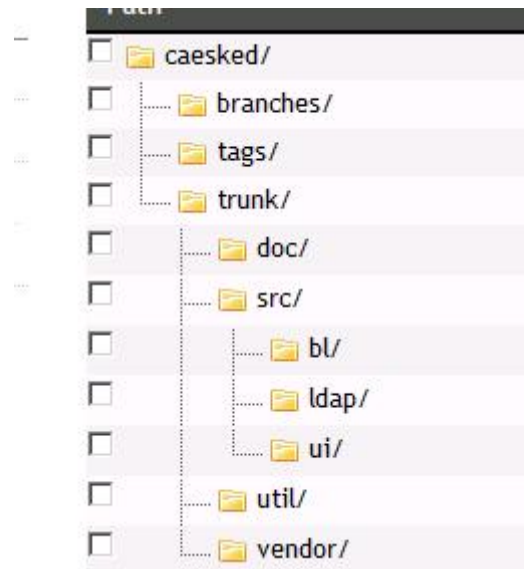


Figure 4.2 - Subversion repository hierarchy.

## 5. Implementation

The Implementation section will provide a guided tour of CAESked, the solution implemented to solve the student scheduling problem. Screenshots will be used to demonstrate features. Along the way, implementation details will be discussed, such as the modification of a calendar plug-in, and PHP code designed to keep the application automatically up to date.

The first screenshot is the login page. Students login with BroncoNet credentials to gain access to their schedule and WMU’s course offerings (See Fig. 5.1).

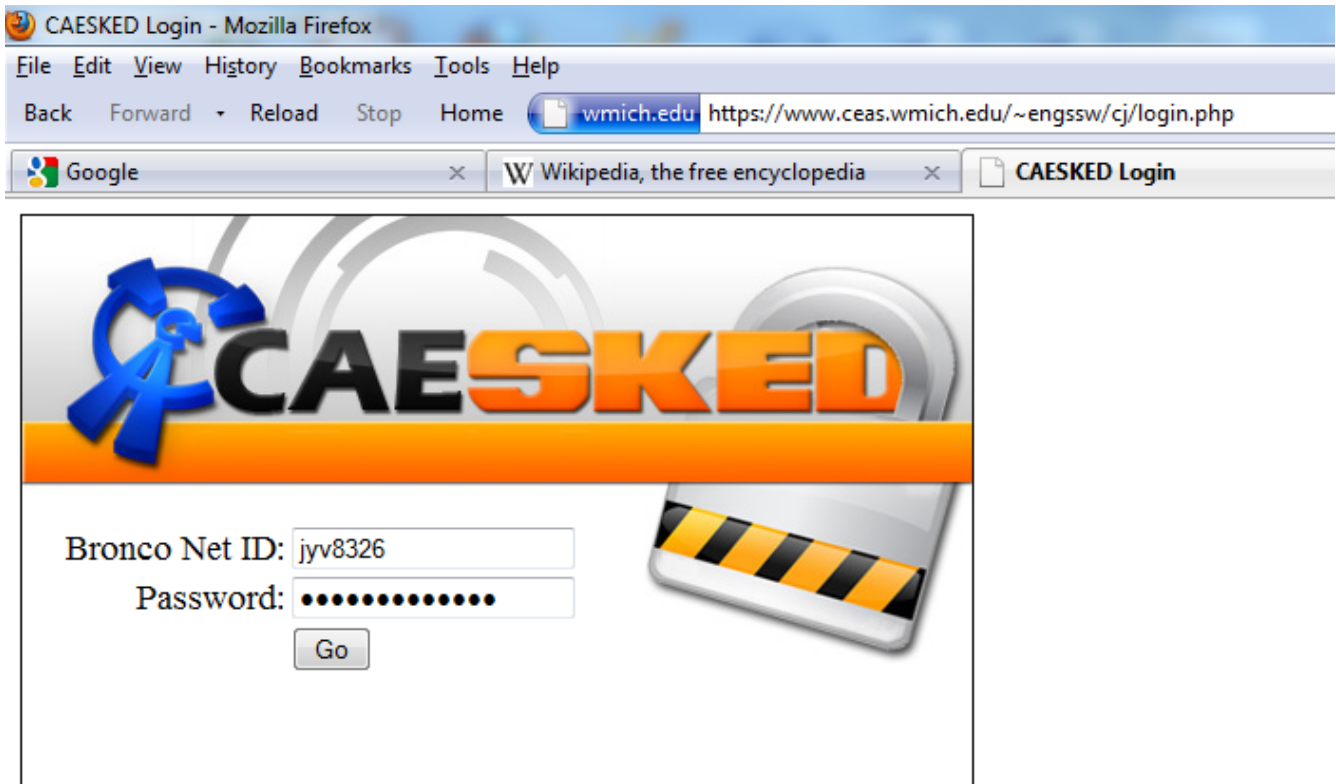


Figure 5.1 – The Login page to the web scheduler, CAESked, seen in the Firefox web browser.

Once logged in, the main page gives access to CAESked’s features. Automatically, the student’s current classes are displayed on a calendar (See Fig. 5.2). “Clicking” on a scheduled class provides details for that class, such as the instructor’s name and the meeting location (See Fig. 5.3). Although more features are available through CAESked’s menu, the main feature is the calendar.



Figure 5.2 – CAESked’s Main page. Note the student’s schedule is automatically loaded into the calendar.



Figure 5.3 – Clicking on the calendar event for CS 4310 brings up class details.

CAESKed’s calendar was implemented using a modified plug-in. The original plug-in, “jquery-week-calendar”, was programmed using the JavaScript toolkit jQuery. However, the original plug-in didn’t quite suit the needs of this project. The original version was modified from a calendar meant to show a week such as Sunday, October 10, 2010, through Saturday, October 16, 2010, into one which showed a generic week, Sunday – Saturday. Since course

meetings are the same each week, only a generic week was desired.

Another feature available through the menu is “Quick Add”. (Note that the menu provides animated toggling of features like “Quick Add” - See Fig. 5.4.) Through this feature, a student can quickly add a section to their schedule if they know the desired course’s subject code and course number (for example, “CS1110”).

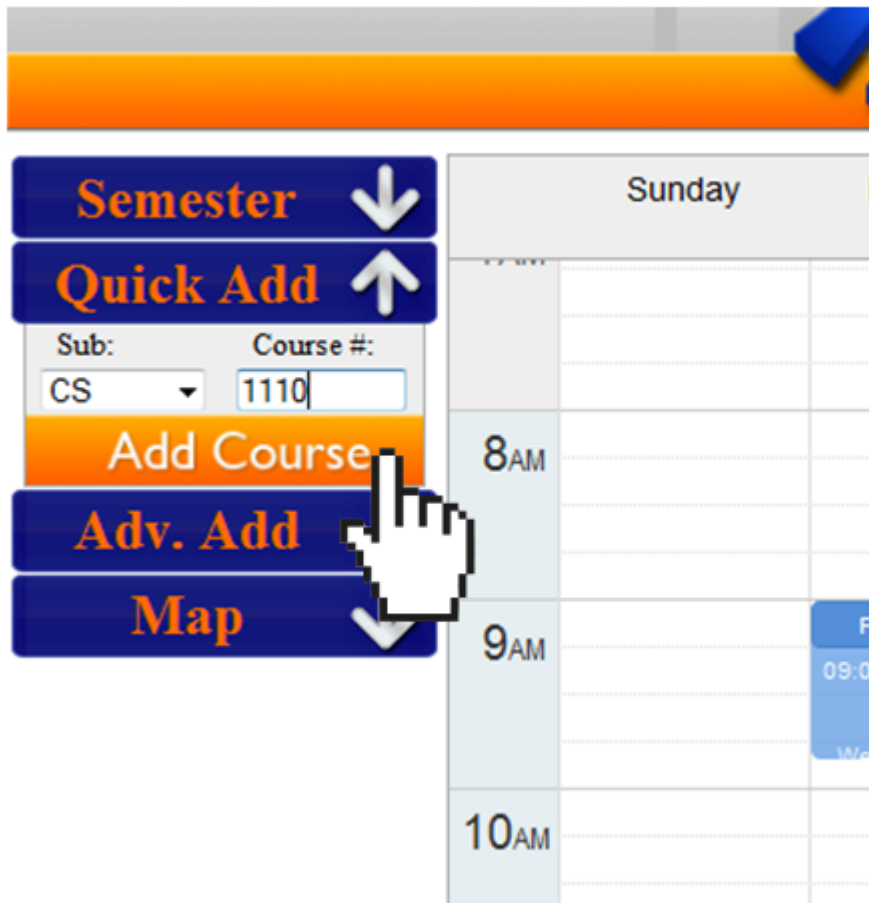


Figure 5.4 – The “Quick Add” feature, adding CS 1110. Note also that menu items like “Quick Add” toggle open and closed when selected.

“Advanced Add” can be used to filter through all courses. It requires first selecting a subject (See Fig. 5.5), then selecting between the courses offered for that subject, and finally selecting a section. Note that before adding to their schedule, a student can quickly compare details between available sections – “hovering” the cursor over a section displays its details (See Fig. 5.6).

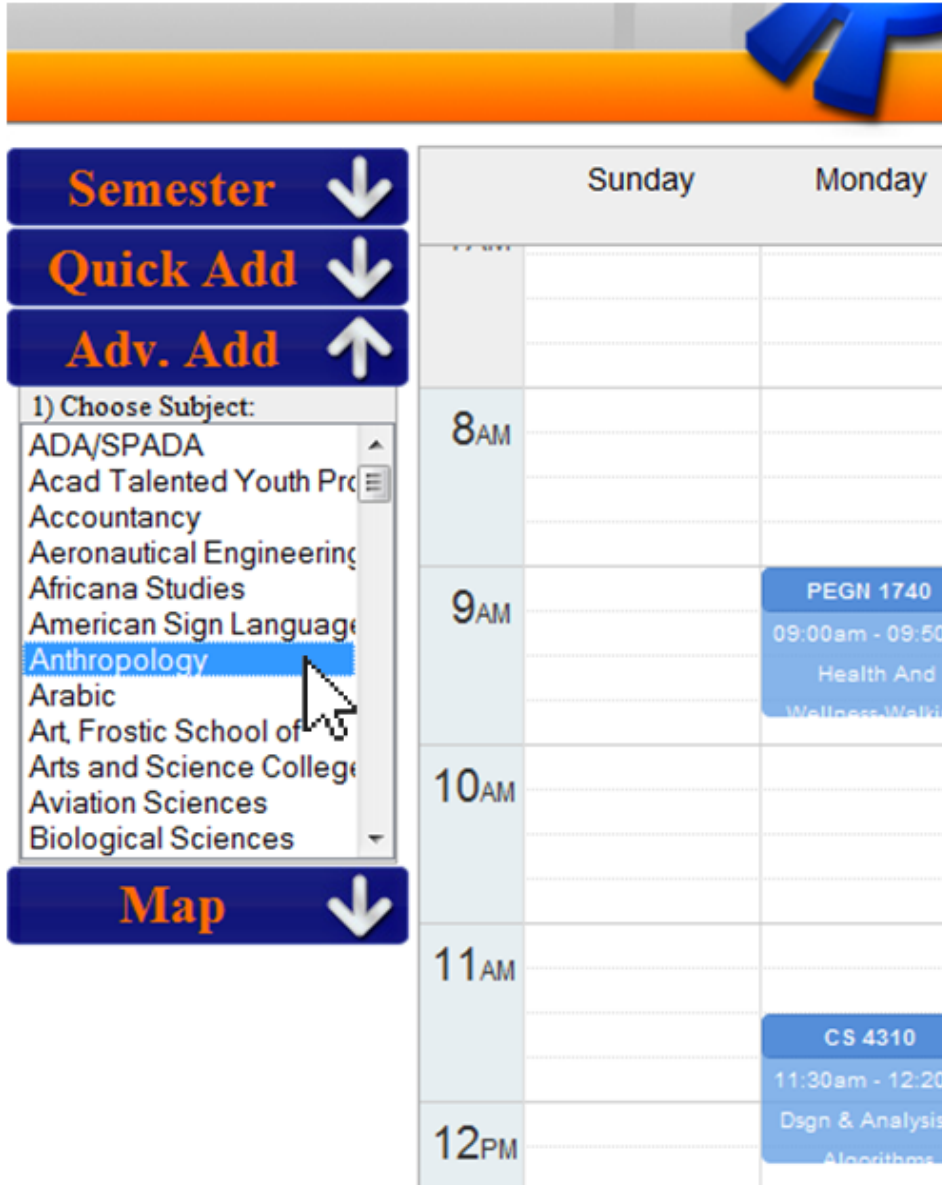


Figure 5.5 – The “Advanced Add” feature. The first step in filtering through courses is selecting a subject.



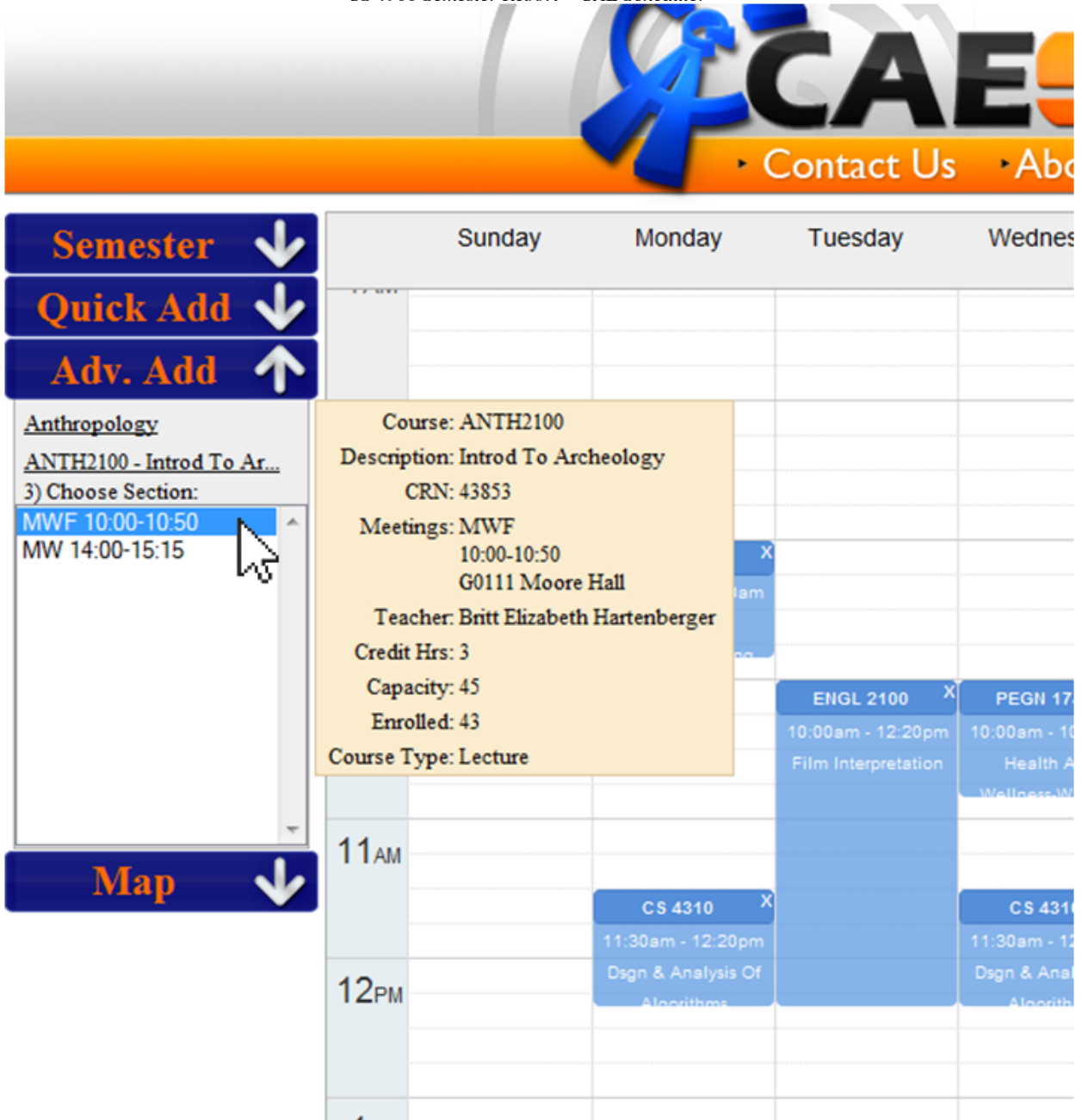


Figure 5.6 – The “Advanced Add” feature, continued. After selecting a subject (“Anthropology”), and a course (“ANTH 2100”), the sections for that course are displayed. Hovering the cursor over a section quickly displays its details.

The final main feature is the “Map”: the locations of scheduled classes are shown on a campus map. The map feature utilizes Google Maps. It provides features that Google Maps

users are accustomed to, such as zooming in and panning the map. Class locations can be seen using markers on the main campus in Fig. 5.7. Note that for Parkview campus, the individual room locations are shown for the College of Engineering and Applied Sciences, rather than just the building (See Fig. 5.8).

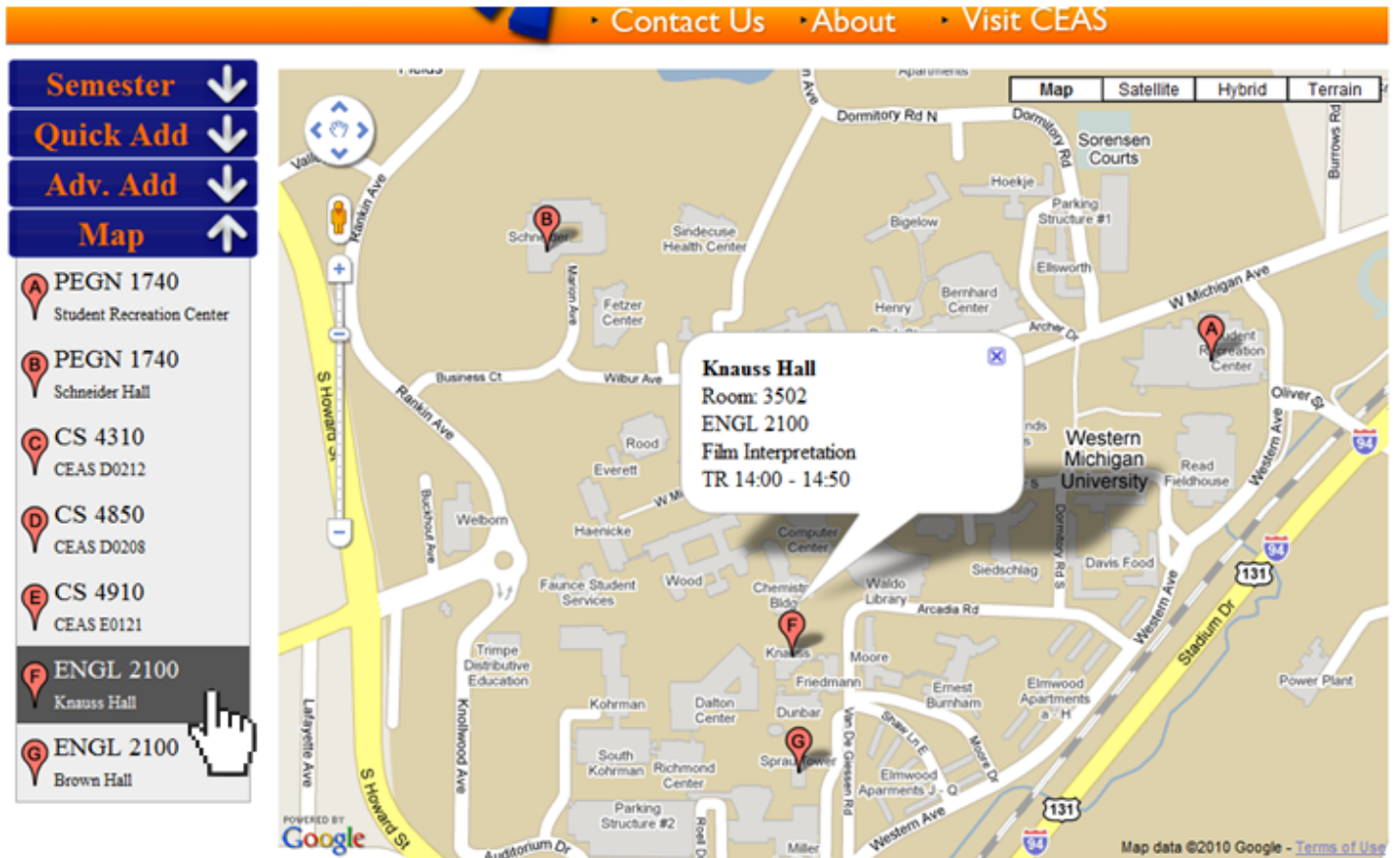


Figure 5.7 – The “Map” feature, displaying markers for scheduled classes on the main campus.

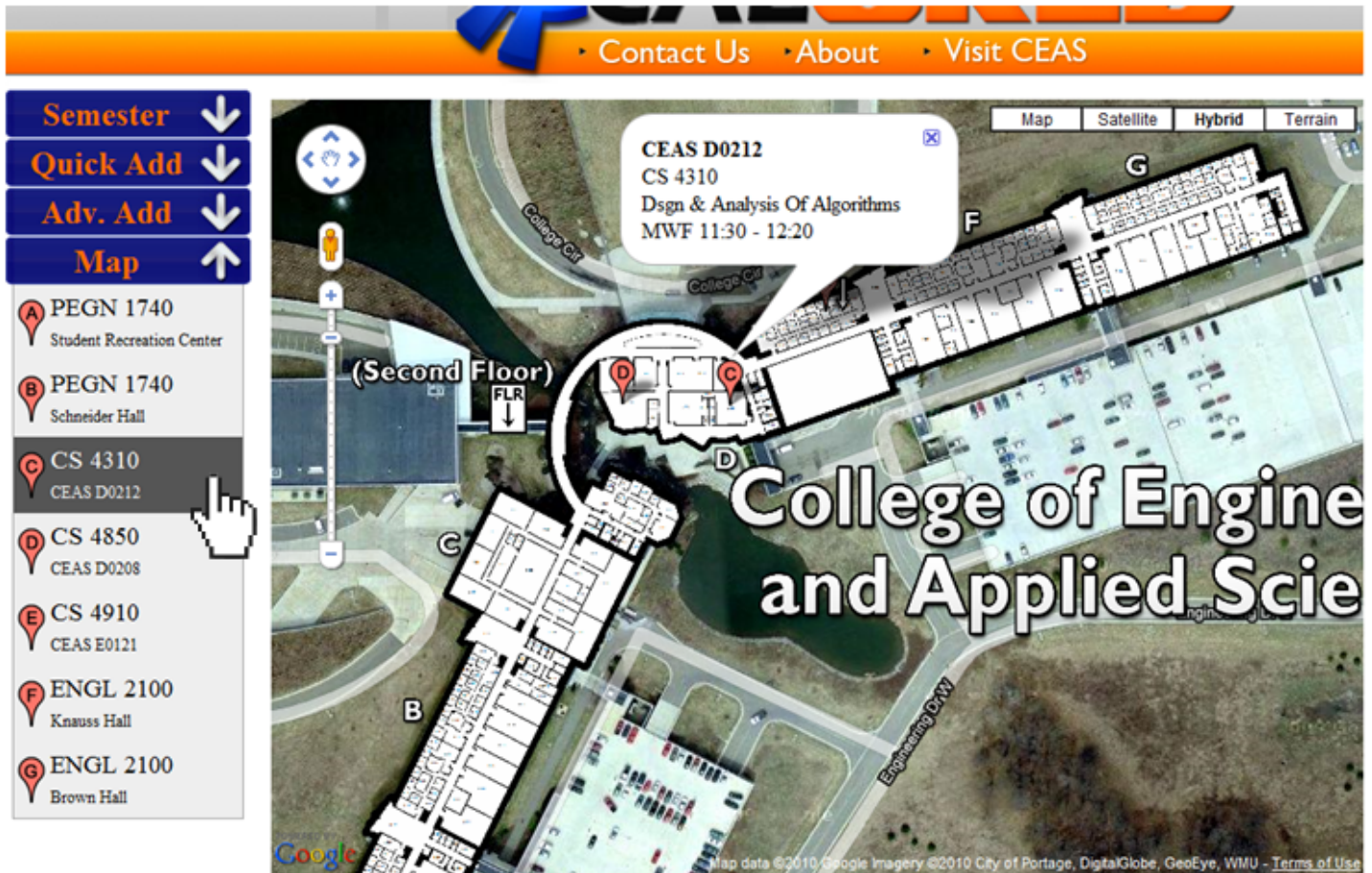


Figure 5.8 – The “Map” feature (continued). Notice that for classes at CEAS, room locations are shown.

During implementation, many features were designed with future maintenance in mind. For example, the semesters seen in Fig. 5.9 are not hard-coded; they are generated using PHP files. As the semesters go by, only the semesters for the current year, the previous year, and the next year will be displayed; and the default semester will always be the upcoming semester (or the present semester for its first two months). Other PHP code was also written to automatically keep the application up to date. For example, the subject codes used by “Quick Add” and “Advanced Add” are dynamically generated from the current subjects used in LDAP. If WMU adds a new subject, it will be automatically added.



Figure 5.9 – The available semesters will be updated automatically.

## 6. Testing

The Testing section will discuss the steps which were taken to help ensure the application is reliable. This included unit testing, functional testing, browser compatibility testing, and user testing. The testing frameworks used in this project included: PHPUnit and Selenium. PHPUnit fit this project's needs for unit testing. Selenium was used for functional testing of the web application.

Unit testing was used primarily to test application logic, which was mostly written in PHP. PHPUnit is a common choice for PHP unit testing, and is similar to the JUnit framework discussed in class. Like JUnit, it also is friendly to the eXtreme Programming methodology, which this project followed. Since PHPUnit is, as the name implies, geared towards testing PHP, additional testing was needed for the parts of the application written in JavaScript. Since JavaScript is closely associated with the user interface, functional testing played a large role in its testing.

Selenium fit our needs for functional testing - testing the system as a whole. Selenium includes a plug-in for the Firefox web browser. The plug-in allowed user actions on the webpage to be recorded. Use cases were recorded and played back as retesting was needed, automating the testing process. Selenium was used mostly for testing the user interface, rather than testing the application logic. But only automating unit and functional testing wasn't enough. Browser compatibility tests and user testing was also needed.

Since different web browsers are notorious for display differences, we tested on a number of browsers and operating systems. We tested recent versions of Internet Explorer, Firefox, Safari, and Chrome on various operating systems (See Appendix D). To help insure browser compatibility, we used W3C's page validator to make sure we followed HTML standards (W3C,

2010); and, in addition to individual web browser tests, we also used browsershots.org to see how our pages looked on over 30 web browsers on various operating systems (Browsershots.org).

Especially with web applications, user testing is essential. Prof. Kaugars CS3400 (Graphical User Interface Design) students filled the role of users. A task list was given to the testers and completed (See the User Task Form in Appendix H). The tasks completed by the users ensured that the features of the application worked correctly and that the user interface was intuitive. Testing by a large number of students helped find rare bugs and also provided interface design feedback. The combined approach of user, browser compatibility, unit, and functional testing went a long way toward verifying the stability of our application.

## 7. Security

The Security section will discuss the steps which were taken to secure personal information and prevent attacks against the application. This includes the encryption of personal data and the prevention of LDAP injection and Cross-Site Scripting attacks.

CAESked uses HTTPS encryption for as much of the application as possible. Each page (the login page, the main page, and the map) uses the HTTPS protocol. Using HTTPS, login credentials and personal data relating to a student's schedule are all encrypted. However, there remains one issue with CAESked using HTTPS: the version of Google Maps used for the map feature does not support HTTPS. Because of this, the map data from Google is not encrypted. Although personal data is still encrypted, use of both HTTP and HTTPS on the same page is still a security issue. A crafty attacker could modify some of the JavaScript from Google and compromise the Map feature. This remains an issue, but the choice to use HTTPS on the same page as the map was still considered a plus – encrypting personal data with the possibility of a cracker reading it is better than not encrypting personal data at all.

Although no SQL database was used (and therefore SQL Injection attacks weren't an issue), "LDAP Injection" attacks are still a security concern. LDAP Injection attacks work the same as SQL Injection attacks – a cracker inputs specially crafted strings of characters with the intention of gaining unauthorized information or otherwise causing damaging effects. Unfortunately, in the case of LDAP and PHP, stored procedures and parameterized queries were not a defensive option as they are with SQL. The main defense against LDAP Injections appears to be strong input validation. An approach found at The Web Application Security Consortium's website was followed (LDAP Injection, 2009): characters such as "\*" wildcards used in LDAP injections were identified. All strings that were passed into LDAP Domain Names and Filters



were validated, even strings that did not come directly from the user (such as semester values coming from the semester drop-down list).

The above validation was in addition to user interface validation, which was done before LDAP validation. Validating user input was also done to help prevent the second major type of web exploit: Cross-Site Scripting. One of the most common attack vectors for Cross-Site Scripting is also user input (Cross-site scripting, 2010). Validation of user input helped reduce this threat.



## **8. Summary**

In summary, CAESked is a web application designed to solve a problem; previously WMU had no tool to assist students in planning a weekly class schedule. CAESked combines the benefits of scheduling software, such as Google Calendar, with access to WMU class data. In addition, it provides features such as a campus map automatically showing a student's class locations. The client, Prof. Kaugars, offered excellent support to the project and will be able to maintain the project through the CAE IT department after the team members have graduated. WMU has gained a useful tool from this project, and the project's team members have learned a lot in completing it.

## 9. Appendices

### 9.1 Appendix A: Spikes

Spike solutions are created to figure out answers to tough technical or design problems. A spike solution is a very simple program to explore potential solutions. Spikes are built to only address the problem under examination and ignore all other concerns. The goal is reducing the risk of a technical problem or increasing the reliability of a user story's estimate.

Spikes:

1. Create a potential GUI design for discussion with client.

Status – Completed: This spike enabled us to go through several paper prototypes of interfaces that would meet both the clients and users needs.

2. Create a potential theme for discussion.

Status – Completed: This spike was to get a general feel for the project and the interface that should be represented. The initial theme focused more on WMU and the black and gold colors. This was not what the client wanted so other options were perused. In the end the client wanted a white background with the CAE being prominent in the website. The client agreed to the theme and logo, along with the color scheme.

3. Connect to LDAP and see what data is available.

Status – Completed: This required downloading an LDAP client and verifying all of the connection information provided was valid. After that it included viewing the LDAP tree and seeing what information was available under a user's login and what information needed an administrative login.

4. Retrieve course offerings from LDAP.

Status – Completed: This was a proof of concept to use PHP to connect to the LDAP server and gather course information about the signed in user. This verified PHP could connect and reliably gather data from the LDAP server. Also verified what information about course history could be gathered. Because of this spike we learned that transfer course information is not stored in LDAP.

5. Program small PHP file to test LDAP connectivity and speed of queries.

Status – Completed: This is a modification of the previous spike with connectivity and timestamps included into the program. This allowed us to measure the time for LDAP queries from different locations on different computers and connections. The result was that LDAP was fast enough to query that there is no need to copy the data into a MySQL database on Dolby.

6. Create a webpage integrating PHP, JavaScript, and AJAX/JSON. For example, selecting a course through a series of combo boxes to display course detail information.

Status – Completed: This allowed us to research the usability of JSON and its speed. Because JSON comes in directly as JavaScript objects, it greatly reduced overhead compared to XML which JavaScript must parse every time the object is refreshed.

7. Test candidate JavaScript frameworks for calendar functionality.

Status – Completed: jQuery (<http://jquery.com/>) has a weekly calendar plug-in (jQuery-week-calendar: <http://wiki.github.com/robmonie/jquery-week-calendar/>). This plug-in meets most of our functional needs, and it works well with JSON. It will need modification, but since its source code is included and its license allows it, this shouldn't be a problem.

## 9.2 Appendix B: Stories

User stories serve the same purpose as use cases but are not the same. They are used to create time estimates for the release planning meeting. They are also used instead of a large requirements document. User Stories are written by the customers as things that the system needs to do for them. They are similar to usage scenarios, except that they are not limited to describing a user interface.

Story	Time Estimate	Risk	Status	Notes
1. A WMU student wants to see if his planned classes conflict with each other time-wise. He'd like to see them visually on a calendar (a week) to easily see how much time he has between classes and when he has free time. (Via a web browser)	40 hrs	Medium	Done	Implemented using a jQuery week calendar plug-in and data from WMU through LDAP
2. Details for individual classes and sections for the upcoming semester should also be made available.	30 hrs	Medium	Quick Add - Done  Detailed Add - Done	
3. A student wants to see the courses he's already registered for displayed in a weekly calendar. (Automatically filled calendar after login.)	8 hrs	Low	Done	
4. A student wants to add her work hours to the schedule.	15 hrs	Low	Moved to Release 2	
5. Another student wants to see on a map where their classes will be, i.e. where their rooms are at CEAS and which buildings on the main campus.	35 hrs	High	Done	Currently, the main campus shows bldgs but not rooms

### Future Release Stories:

6. They'd also like to save it in case they decide to make changes later. (Save on the Server - one copy to limit data size)	8 hrs	Med		
7. A student wants help deciding which	25 hrs	High		

electives to take. They have a list of possibilities, but want their schedule to be compact (with no conflicts). (Elective Recommender)				
8. A student would like to know which of their potential classes their friends are in. (This could be implemented using Facebook friends/Facebook's API. Students would need to input their Facebook ID to allow this feature, and it would probably require storing in a MySQL database. Perhaps sections from the QuickAdd/DetailedAdd could be starred if there were friends in it. And the friend's names could be shown when the mouse is hovered over the star.)	40 hrs	High		
9. This student wants to also know if he's met all the pre-reqs for his classes next semester.	10 hrs	High (Data availability issues)		Will probably only be able to list pre-reqs for a course
10. A <i>teacher</i> would like to see his courses on a weekly schedule.	20 hrs	Medium		
11. This teacher would also like an easy way to email all of the students in his respective classes.	5 hrs	Low		
12. A student would like to know what courses are remaining for her major, minor, and gen-ed requirements in order to help her schedule classes.	50 hrs	High		
13. A student would like to register for the courses they've selected in this scheduler with one click. (This feature has all but been ruled out for security reasons, but we include it here for completeness sake.)	8 hrs	High		
14. A student would like to export the schedule they've created in CAESked and use it in Google Calendar and WMU's email calendar.	25 hrs	High		WMU's email calendar may not allow importing.

### 9.3 Appendix D: Requirements Stories

Requirements Stories define non-feature requirements of the client. For example, Requirements Stories may specify the hardware or software that will be used to run the application.

Requirements Stories:

1. Web browser requirements: Per a discussion with Prof. Kaugars, the application should support all modern browsers. This includes Internet Explorer 7+, Mozilla Firefox 2+, Google Chrome and Safari 3.1+.
2. Operating System Support: Per a discussion with Prof. Kaugars, the application should support all modern operating systems. This includes Windows 2000, XP, Vista, 7, Linux, and Apple OS X.
3. Hardware Support: The application will run on a server called "Dolby" that Prof. Kaugars already has running and has support personnel to keep it running in the future. Dolby is a Linux server running CentOS release 5.2 (Linux kernel Ver. 2.6.18-92.1.22.e15). The server already has MySQL (Ver. 5.1.44) installed and Apache (Ver. 2.2.3). The server has abundant processing power (AMD Opteron 2210) and memory (3.5 GB).

## 9.4 Appendix E: Standards

Standards define the programming standards that were used for coding the application.

Coding standards followed:

PHP: <http://pear.php.net/manual/en/standards.php>

JavaScript: <http://javascript.crockford.com/code.html>

## 9.5 Appendix F: Non-Disclosure Agreement

(Email correspondence:)

from **Karlis Kaugars** <karlis.kaugars@wmich.edu>  
to Chris Fruin <chris.fruin@gmail.com>,  
Jerry Grochowski <jerry.grochowski@gmail.com>,  
John kapenga <john.kapenga@wmich.edu>  
date Wed, Feb 10, 2010 at 9:27 AM  
subject Non-Disclosure  
mailed-by wmich.edu

I confirm that I do not require a non-disclosure agreement to be signed for the Web Class Scheduler project being completed by Chris Fruin and Jerry Grochowski.

Karlis.



## 9.6 Appendix G: Ownership

Prof. Kaugars has agreed not to claim ownership of the source code for this project. He's agreed to release it under the GNU General Public License (GPL) Version 3 (<http://www.gnu.org/licenses/gpl-3.0.txt>).

## 9.7 Appendix H: User Testing Form

This is the form that was used by the user testers (CS 3400 students). Each student was given one form and was asked to complete its steps using CAESked.

### User Evaluation of CAESked - A Class Scheduling Web Application

1. Enter the following URL into a web browser:

<https://www.ceas.wmich.edu/~engssw/cj/login.php>

2. Record the web browser you are using: \_\_\_\_\_

(If at any time you notice an error or receive an error message, please record it. Also note that the changes you make to your “schedule” in this application will not affect your actual schedule. This web application has “read-only” access and is for schedule planning only.)

3. Login using your BroncoNetId credentials.

Were you able to successfully login? Yes \_\_\_ No \_\_\_

4. The main page should automatically load with your classes for the fall semester scheduled on the calendar. Did your schedule automatically load? Yes \_\_\_ No \_\_\_

5. Change the semester to Spring 2011. How difficult was it to change the semester?

(Check one of the following)

Easy \_\_\_ Moderate \_\_\_ Difficult or Confusing \_\_\_ Could not do or Failed \_\_\_

6. Use the “Quick Add” feature to add a Computer Science course you plan on taking to your Spring schedule. How difficult was it to use this feature?

Easy \_\_\_ Moderate \_\_\_ Difficult or Confusing \_\_\_ Could not do or Failed \_\_\_

Record the CRN of the CS course you added: \_\_\_\_\_

7. Use the “Advanced Add” feature to add one general education requirement to your Spring schedule. (If you know of a course you wish to take, try to add it, otherwise you may randomly select a non-Computer Science course). How difficult was it to use this feature?

Easy\_\_\_ Moderate\_\_\_ Difficult or Confusing\_\_\_ Could not do or Failed\_\_\_

Record the CRN of the non-CS course you added: \_\_\_\_\_

8. View your schedule for the Spring semester. (The calendar should display events for at least the two courses you just added.) Click on a calendar event/item . Are details displayed for the calendar event you clicked on? Yes\_\_\_ No\_\_\_

9. Do the detail fields appear correct? (Do the “Meetings” match the scheduled times on the calendar; other than possibly “Capacity”, are there any fields that are blank/undefined/or erroneous?)

Appears Correct\_\_\_ Appears Incorrect\_\_\_

(If Incorrect, indicate the incorrect field(s) and value(s).)

10. Use the “Map” feature to view the locations of your scheduled courses. Does it appear to be working correctly? Yes\_\_\_ No\_\_\_

11. Were any of your course locations not found (does the list on the left, under “Map”, show a question mark icon rather than a red marker?)

All Locations Found\_\_\_ Location(s) Not Found\_\_\_

(If not found, record the building name and room number.)

12. Use the mapped course list (on the left, under “Map”) to center on your CS course. (Click the listed CS course.) Did the map center correctly and display an info balloon for the course?

Yes\_\_\_ No\_\_\_ (if “No”, record the problem - that is, did not center on course, or no info balloon, etc.)

13. Assuming your CS course location is the CEAS building, the map should display the floor plan and the marker should be on the correct room for the course. Does the marker appear to be on the correct room?

Yes\_\_\_ No\_\_\_

14. Try to change the floor displayed for CEAS. How difficult was it to change the floor?

Easy\_\_\_ Moderate\_\_\_ Difficult or Confusing\_\_\_ Could not do or Failed\_\_\_

15. Did the map feature seem slow or sluggish?

Very Slow\_\_\_ A Little Slow\_\_\_ Seemed OK\_\_\_ Was Snappy\_\_\_

16. Delete the CS course from your schedule (from the calendar). How difficult was it to remove the course?

Easy\_\_\_ Moderate\_\_\_ Difficult or Confusing\_\_\_ Could not do or Failed\_\_\_

17. Briefly describe one aspect of the user interface that you did not like or found unintuitive.

18. Briefly describe one feature you would like to see added to this web application.

## 10. Bibliography

Browsershots.org. (n.d.). *Browser Compatibility Test*. Retrieved October 31, 2010, from Browser Shots: <http://browsershots.org/>

Encarta World English Dictionary [North American Edition]. (2009). *corequisite*. Retrieved February 20, 2010, from msn encarta: [http://encarta.msn.com/dictionary\\_1861600329/corequisite.html](http://encarta.msn.com/dictionary_1861600329/corequisite.html)

Los Medanos College. (n.d.). *Course Prerequisites & Advisories*. Retrieved February 20, 2010, from Los Medanos College: [http://www.losmedanos.edu/courses/pre\\_req2.htm](http://www.losmedanos.edu/courses/pre_req2.htm)

Mason, M. (2006). *Pragmatic Version Control: Using Subversion*. Raleigh: The Pragmatic Bookshelf.

PHP. (2009, December 17). *index*. Retrieved February 20, 2010, from PHP: <http://php.net/index.php>

Refsnes Data. (2010). *Browser Statistics*. Retrieved February 21, 2010, from w3schools.com: [http://www.w3schools.com/browsers/browsers\\_stats.asp](http://www.w3schools.com/browsers/browsers_stats.asp)

SunGard. (2010). *Banner Unified Digital Campus*. Retrieved February 20, 2010, from SunGard Higher Education: <http://www.sungardhe.com/Products/Product.aspx?id=832&LangType=1033>

SunGard. (2010). *Luminis Platform*. Retrieved February 20, 2010, from SunGard Higher Education: <http://www.sungardhe.com/Products/Product.aspx?id=1042&LangType=1033>

W3C. (2010). *The W3C Markup Validation Service*. Retrieved October 31, 2010, from W3C: <http://validator.w3.org/>

Western Michigan University. (n.d.). *Home*. Retrieved February 20, 2010, from Computer Aided Engineering Center: <http://www.wmich.edu/engineer/cae/>

Western Michigan University. (n.d.). *About*. Retrieved February 20, 2010, from College of Engineering and Applied Sciences: <http://www.wmich.edu/engineer/>

Western Michigan University. (n.d.). *About WMU*. Retrieved February 20, 2010, from Western Michigan University: <http://www.wmich.edu/about/>

Western Michigan University. (n.d.). *Fast Facts*. Retrieved February 22, 2010, from About WMU: <http://www.wmich.edu/about/facts/>

Western Michigan University. (2008). *Frequently Asked Questions*. Retrieved February 20, 2010, from GoWMU: <https://gowmu.wmich.edu/site/help/faq.html>

Western Michigan University. (2010, January 22). *Information Technology*. Retrieved February 20, 2010, from Office of Information Technology: <http://www.wmich.edu/oit/>

Wikipedia. (2010, February 5). *Ad hoc*. Retrieved February 20, 2010, from Wikipedia, the free encyclopedia: [http://en.wikipedia.org/wiki/Ad\\_hoc](http://en.wikipedia.org/wiki/Ad_hoc)

Wikipedia. (2010, February 22). *Ajax (programming)*. Retrieved February 22, 2010, from Wikipedia, the free encyclopedia: [http://en.wikipedia.org/wiki/Ajax\\_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming))

Wikipedia. (2010, February 21). *Data scraping*. Retrieved February 21, 2010, from Wikipedia, the free encyclopedia: [http://en.wikipedia.org/wiki/Data\\_scraping](http://en.wikipedia.org/wiki/Data_scraping)

Wikipedia. (2010, February 18). *Lightweight Directory Access Protocol*. Retrieved February 20, 2010, from Wikipedia, the free encyclopedia: <http://en.wikipedia.org/wiki/LDAP>

Wikipedia. (2010, February 20). *MySQL*. Retrieved February 20, 2010, from Wikipedia, the free encyclopedia: <http://en.wikipedia.org/wiki/MySQL>

Wikipedia. (2010, February 2). *System testing*. Retrieved February 2, 2010, from Wikipedia: [http://en.wikipedia.org/wiki/Functional\\_testing](http://en.wikipedia.org/wiki/Functional_testing)

Wikipedia. (2010, February 26). *Unit testing*. Retrieved February 26, 2010, from Wikipedia: [http://en.wikipedia.org/wiki/Unit\\_testing](http://en.wikipedia.org/wiki/Unit_testing)

## 11. Glossary

**Ad hoc** – Ad hoc is a Latin phrase which means "for this purpose". It generally signifies a solution designed for a specific problem or task, non-generalizable, and which cannot be adapted to other purposes. The term may refer, for example, to a tailor-made suit, a handcrafted network protocol or a purpose-specific equation. (Wikipedia, 2010)

**AJAX** – Ajax (shorthand for asynchronous JavaScript and XML) is a group of interrelated web development techniques used on the client-side to create interactive web applications. With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. (Wikipedia, 2010)

**Apache** – The Apache HTTP Server, commonly referred to as Apache, is web server software notable for playing a key role in the initial growth of the World Wide Web. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently known as Sun Java System Web Server), and has since evolved to rival other Unix-based web servers in terms of functionality and performance. The majority of web servers using Apache run a Linux operating system.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The application is available for a wide variety of operating systems, including Unix, GNU, FreeBSD, Linux, Solaris, Novell NetWare, Mac OS X, Microsoft Windows, OS/2, TPF, and eComStation. Released under the Apache License, Apache is characterized as open source software. (Wikipedia, 2010)

**Banner** – Banner Unified Digital Campus is the world's most widely used collegiate administrative suite of student, financial aid, finance, human resources, and advancement systems. It is a tightly integrated suite of proven, scalable, enterprise-wide applications on a

single database, designed to support institutions of all sizes and types. Banner runs on the Oracle Relational Database Management System. (SunGard, 2010)

**Branches** – Different lines of development in the project source code repository. For example, after a release is made, it may become its own branch with bug fixes made specifically for its branch.

**CAE** – The Computer Aided Engineering Center is a comprehensive computer center comprising of 7 computer labs equipped with state of the art technology and specific engineering software for The College of Engineering and Applied Sciences at Western Michigan University. The center also provides a service area for troubleshooting hardware and software problems. (Western Michigan University)

**CEAS** – College of Engineering and Applied Sciences, the CEAS is state-of-the-art facility housed in a \$72 million high-tech building with close interaction with business and industry. CEAS is a separate campus located in Kalamazoo Michigan. (Western Michigan University)

**Core** – The first version of the application. Before even Release 1, the core is the smallest version of the application which is still functional.

**Corequisite** – compulsory accompanying course: a course of study that must be taken along with another. (Encarta World English Dictionary [North American Edition], 2009)

**Data Scraping** – A technique in which a computer program extracts data from human-readable output coming from another program. (Wikipedia, 2010)

**Functional Testing** - software or hardware testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. (Wikipedia, System testing, 2010)

**GoWMU** – GoWMU is Western Michigan University's portal for the WMU community to



access a collection of online resources. This includes e-mail, course registration, academic transcript and viewing of grades. (Western Michigan University, 2008)

**LDAP** – The Lightweight Directory Access Protocol is an application protocol for querying and modifying directory services running over TCP/IP. LDAP stores information in a directory with a tree structure. A directory is a set of objects with attributes organized in a logical and hierarchical manner. A simple example is the telephone directory, which consists of a list of names (of either persons or organizations) organized alphabetically, with each name having an address and phone number associated with it. (Wikipedia, 2010)

**Luminis** – The Luminis Platform gives individuals personalized, 24x7 access to information and services previously limited by location or hours of operation. It integrates with Banner, so users can access the information and services they need through a single sign-on. Information from the Banner administrative systems helps to personalize the Luminis Platform, and users can work through Luminis to supply information back to Banner. (SunGard, 2010)

**MySQL** – MySQL is a relational database management system that runs as a server providing multi-user access to a number of databases. MySQL is officially pronounced *My S-Q-L*, but often pronounced *My SeQueL*. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Sun Microsystems, a subsidiary of Oracle Corporation. (Wikipedia, 2010)

**NetID** – The unique login string given to every WMU student to identify them to the Luminis system. It is assigned in the students first semester and does not change.

**OIT** – The Office of Information Technology provides data, computing, telephone and

connection services to WMU students, faculty and staff. OIT also provides classroom support services to faculty. (Western Michigan University, 2010)

**PHP** – PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. (PHP, 2009)

**Prerequisite** – A course or other requirement that must be completed and verified prior to enrollment in a more advanced level course. (Los Medanos College)

**Section** – A section is the identifier to a particular timeslot for a course. Courses may have multiple sessions in a single semester and each session gets a section number to identify which one it is.

**Spike** – A preliminary test for feasibility. Spikes are proof of concept tests to see if something will work before its integration into the project is attempted.

**Story** – A description of how a user will use the application. Stories are used to pinpoint features and should be agreed upon by the client.

**Tag** – A symbolic name for a set of files in the source code repository. Each release will be tagged in order to keep a frozen copy.

**Trunk** – The main line of development in the source code repository. Most active development on the project will take place in the trunk line.

**Unit Testing** - A software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. (Wikipedia, 2010)

**WMU** – Western Michigan University, Western Michigan University is a dynamic, student-centered research university with an enrollment of 25,000. WMU is located in Kalamazoo

Michigan. (Western Michigan University)

## 12. Index

Apache .....	25, 58	MySQL.....	25, 27, 60
Banner .....	6, 12, 58	NetID.....	20, 60
CEAS .....	6, 59	OIT.....	24, 60
Corequisite .....	6, 59	PHP.....	25, 49, 61
Functional testing .....	40, 59	Prerequisite .....	6, 24, 61
GoWMU .....	6, 59	Registrar.....	6, 24
LDAP .....	20, 24, 60	Unit testing .....	40, 61
Linux.....	25	WMU.....	6, 10, 12, 13, 20
Luminis.....	12, 60		