UNIVERSITY OF WISCONSIN-LA CROSSE

Graduate Studies

AUTOMATING CURRICULUM MANAGEMENT TASKS

A Manuscript Style Capstone Project Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Software Engineering

Rajarajan Vijayakumar

College of Science and Health

December, 2006

AUTOMATING CURRICULUM MANAGEMENT TASKS

By Rajarajan Vijayakumar

We recommend acceptance of this manuscript in partial fulfillment of this candidate's requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

Dr. Kasi Periyasamy.,Ph.D Examination Committee Chairperson	Date
Dr. Kenny Hunt., Ph.D Examination Committee Member	Date
Dr. Thomas Gendreau., Ph.D Examination Committee Member	Date
Dr. Vijendra Agarwal Director, University Graduate Studies	Date

ABSTRACT

Vijayakumar, Rajarajan, "Automating Curriculum Management Tasks", Master of Software Engineering, August 2006, Advisors: Dr. Kasi Periyasamy and Dr. Thomas Gendreau.

This manuscript describes a web based tool that assists in automating curriculum proposal and change procedures, curricular data maintenance and course catalog publication at the University of Wisconsin-La Crosse. Curriculum proposals are initiated by departments, approved by curriculum committees and finally maintained by Records and Registrations department. The current manual system is paper intensive, inconsistent, confusing and follows a manual workflow. The tool provides the following features: 1.) Using this tool, a faculty member will be able to create new course and program proposals through a web based application. 2.) The current database (namely catalog master) consists of unnormalized tables and consists of neither constraints nor relationships; so a new database has been developed to tackle the problem of redundancy and inconsistency 3.) The tool allows the creation of several reports.

This manuscript describes the development of the curriculum management tool; in particular, it describes the activities performed in each phase, the challenges encountered, the issues that arose, the current status of the project, and its limitations and continuing work.

ACKNOWLEDGEMENTS

My foremost thank goes to my project advisors Dr. Kasi Periyasamy and Dr. Thomas Gendreau. Without them, this dissertation would not have been possible. I thank them for their patience and encouragement that carried me on through difficult times, and for their insights and suggestions that helped to shape my design and programming skills. Their valuable feedback contributed greatly to this dissertation.

I want to express a special thanks to Mr.Josh Anderson (Web Developer - ITS) and Michael A. Taylor (Database Administrator – ITS). Their contributions to this project have been immense.

I would like to thank my project sponsors Ms.Von Ruden Janis and Ms.Diane Schumacher for taking the time to meet regularly and provide valuable feedback during the project. There are no words to express my gratitude to my parents. Although they did, as the saying goes, make me what I am today, that pales next to the wealth of love, support, advice and practical assistance they have given throughout this project.

Finally, I would also like to express my thanks to the Computer Science Department, Records and Registration Department and the University of Wisconsin-La Crosse for providing the computing environment for my project.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

APACHE STRUTS

Struts is a web application framework that simplifies the building of web applications based on MVC design principles

CSS

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document.

CLOB

A Character Large Object (or CLOB) is a collection of character data in a database management system, usually stored in a separate location that is referenced in the table itself. CLOBs are typically text exceeding 4000 characters in length. CLOBs usually have very high size limits, on the order of 2GB or more

DEPLOYMENT DESCRIPTOR

A Deployment Descriptor (DD) is a component in Java 2 Platform, Enterprise Edition applications that describes how a web application should be deployed. It directs a deployment tool to deploy a module or application with specific container options and describes specific configuration requirements that a deployer must resolve.

In Java 2 Platform, Enterprise Edition applications, XML is used for the syntax of the Deployment Descriptor file. It must be called web.xml, and it must be placed in a sub directory called WEB-INF, directly under the web application root.

EAR (Enterprise ARchive)

An Enterprise ARchive, or EAR, is a file format used by Java EE for packaging one or more modules into a single archive so that the deployment of various modules onto an application server happens simultaneously and coherently. It also contains XML files called deployment descriptors which describe how to deploy the modules on an application server

HTML

HyperText Markup Language. A markup language designed for creating web pages and other information to view in a web browser.

IEEE (Institute of Electrical and Electronics Engineers)

An international organization whose constitution describes their purpose as "scientific and educational, directed toward the advancement of the theory and practice of several engineering fields including computer science."

JAR (Java Archive File)

Java Archive file. An archive (like a ZIP file) containing Java class files and images. JAR files are used to package Java applications for deployment.

JAVASCRIPT

JavaScript is a scripting language used in Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.

JDBC

JDBC (Java Database Connectivity) is a Sun Microsystems standard defining how Java applications access databases.

JDK (Java Development Kit)

A software development package from Sun Microsystems that implements the basic set of tools needed to write, test and debug Java applications and applets.

JSP

Java Server Pages is a Java technology that is used to develop dynamic web pages. Java Server Pages (JSP) are indeed HTML pages with embedded Java code. A JSP compiler is used to generate a Servlet from the JSP page.

LDAP (Lightweight Directory Access Protocol)

The Lightweight Directory Access Protocol, or LDAP ("ell-dap"), is a networking protocol for querying and modifying directory services running over TCP/IP. An LDAP directory usually follows the X.500 model: it is a tree of entries, each of which consists of a set of named attributes with values. While some services use a more complicated "forest" model, the vast majority use a simple starting point for their database organization.

MODEL-VIEW-CONTROLLER (MVC)

Model-view-controller (MVC) is a software architecture that separates an application's data model, user interface, and control logic into three distinct components so that modifications to one component can be made with minimal impact to the others.

MVC is often thought of as a software design pattern. However, MVC encompasses more of the architecture of an application than is typical for a design pattern.

PDF

An Adobe technology representation for formatting documents so that they can be viewed and printed using the Adobe Acrobat reader.

SMTP

Simple Mail Transfer Protocol is the standard way to send emails across a network.

SQL

Structured Query Language is a popular computer language used to create, modify and query databases.

SOFTWARE PROTOTYPING

Prototypes provide software developers with a "working model" for demonstration or use by customers, quality-assurance, business analysts, and managers to confirm or make changes to requirements, help define interfaces, develop collaborating components, and to provide proof of incremental achievement of scheduled contractual agreements. Software Prototyping serves any and all of these purposes in practice.

SRS

Software Requirements Specification. A document format supplied by the IEEE for specifying the requirements of a software system.

XML (Extensible Markup Language)

A W3C recommendation for creating special-purpose markup language and is widely used to exchange data, store configuration data, along with many other uses.

UML

Unified Modeling Language™ (UML) is an industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems standardized by the Object Management Group. UML simplifies the complex process of software design by using “blueprints” for software construction.

WAR(WEB Archive File)

A WAR file is simply a snapshot of your web app structure, in a nice portable, compressed form (it is really just a JAR file).

1. BACKGROUND INFORMATION

Curriculum changes are initiated by academic departments. These changes include new course proposals, new program proposals, modification and deletion of existing courses, and modification and deletion of existing programs. To start with, departments are required to fill up some forms depending on the type of changes requested and submit them to the curriculum committees (undergraduate or graduate). The change requests must be approved by the respective academic units (department and college) before they are submitted to the corresponding curriculum committees. The committee in turn will discuss the changes and approve them. After the committee approves the change requests, the Records and Registration office will make the changes visible (or available) to everyone.

At present, the curriculum change process uses paper forms and manual processes. Some of the forms used in this process are listed below:

LX-138: Basic form used to record any curriculum change. There are two versions of this form, one for courses (LX-138C) and one for programs (LX-138P).

- **LX138C**: Form used to make **any** change to a course including description, prerequisite(s), credits, title, grading pattern, instructional pattern, and course number. This form is also used to delete a course and to create a new course.

- **LX138P:** Form used to make changes to majors, minors, concentrations, emphases and college core. Changes include credits, required courses, electives, and other program requirements. It is also used for submitting the curricula of new programs to the curriculum committee.

LX-139: Form used to provide course objectives, outline and requirements. It is never used by itself; it accompanies the LX138C form. It must be submitted for all new courses.

LX-140: Form used to provide additions, deletions or changes to the General Education Program.

A faculty member requests curriculum proposal/curriculum change by filing the appropriate LX Forms. The completed forms are approved by that department and the dean of the appropriate college. The original forms, along with 26 photocopies, are then submitted to Records and Registration. The Records and Registration office in turn numbers the proposals, prepares a curriculum committee agenda from submitted materials, prepares packets of materials and sends them to all the UCC/GCC members. After the proposals are approved, the committee chair, provost's representative and registrar all sign the original forms. The committee minutes are prepared, approved and distributed. The information from the LX forms are coded into the catalog and program master tables in the Enterprise database. This information is also updated in the UWL Catalog document. Finally, copies of LX forms are sent to the department, library archive and the catalog materials are distributed to departments for proofing. The proofed copy is submitted to the publications office where it is converted to a quark document first and then it is converted back to a Microsoft WORD document. It is also posted on the web as a static document. The quark file is sent off-campus to a publisher for printing.

The current manual system has the following disadvantages:

Paper intensive: The LX forms are available through email, but all major and course information must be re-keyed every time a change is requested. Each form is copied 26 times for the committee members. Changes requested at the committee level can result in an additional 26 copies made for the second reading. All agendas and minutes are manually keyed into WORD and the data must be manually keyed into the enterprise system and the catalog document.

Inconsistent: Faculty members do not often know the valid data to enter in various fields. This results in inconsistent and incorrect data. The paper forms allow limited space for entering the course description and program requirements details. Often the space available in the paper form is not sufficient.

Confusion: Sometimes faculty members do not know which forms are required for the changes they are requesting. They often have to call for more information to figure out what forms to complete.

Historical record: It is often difficult to find the paper documents for each change requested. Some changes are not recorded or are written in on the paper documents. Sometimes the paper records are lost or misfiled.

Workflow: The workflow is completely manual. The faculty members complete the forms, and then physically walk them to the chairs and the deans for signatures. After that they manually copy the forms for distribution by the Records Office. The Committee members get the forms and review them before the meetings.

The current project was attempted to automate many of the manual tasks described above and also to provide electronic forms. This will greatly reduce the chances of errors and also will speed up the approval process. In addition, the current project is also expected to lay the foundation for automating other processes such as report generation from Records and Registration, preparation of undergraduate and graduate catalogs and timetables. Moreover, the application is expected to be designed as a web-based application so that it is easy to access and easy to use by the entire university community.

The current project was implemented using prototyping approach. The next chapter briefly describes the two major life cycle models – waterfall model and prototyping model to emphasize the differences.

2. A BRIEF INTRODUCTION TO SOFTWARE LIFE CYCLE MODELS

Systems Development Life Cycle is a detailed and specific set of procedures, steps, and documents that carry a project through its technical development. It includes several phases such as Initiation, Planning, Functional Design, System Design, Development, Integration and Testing, Installation and Acceptance, and Maintenance. The various stages and activities performed in each stage may vary depending on the life cycle model chosen. There is no single life cycle that is best suited for every project. Different software life cycles may complement each other on the same project during different stages of development of a product. Successful use of a particular life cycle model does not ensure success or a best fit for a new project. Choosing the most appropriate model for a particular software project is challenging. Many factors such as team size, project size, available resources, deadlines or milestones, team skills and experience, business policies, and the application domain may influence the choice of a life cycle model. Perhaps the most common life cycle model is the *waterfall model*. It describes a set of stages of software development in a strictly sequential manner. There are several variations of waterfall model, but they all contain the following common stages: requirements analysis and definition, design, implementation, testing and integration, and maintenance. An in-depth discussion of the waterfall life cycle model is beyond the scope of this report; interested readers can refer to any Software Engineering book.

One of the evolutionary software life cycle models is called *software prototyping* that can be used as a complement to the waterfall model in some instances. Software customers and end users usually find it very difficult to precisely express their real requirements at the beginning of the development process. It is almost impossible to predict how a system should be developed and

what user operations should be automated. Careful requirements analysis along with systematic reviews of the requirements help to reduce the uncertainty about what the system should do. However, there is no real substitute for trying out a requirement before committing to it. This is possible if a prototype of the system to be developed is available.

A function described in a specification may seem useful and well defined. However, when that function is used with others, users often find that their initial view was incorrect or incomplete. System prototypes allow users to experiment with requirements and to see how the system supports their work. Prototyping is therefore a means of requirements validation. Users discover requirements errors or omissions early in the software process.

The benefits of developing a prototype early in the software process are the following: (1) Misunderstanding between software developers and users may be identified as the system functions are demonstrated. (2) Missing user services may be detected. (3) Difficult to use or confusing user services may be identified and refined. (4) Software development staff may find incomplete and /or inconsistent requirements as the prototype is developed. (5) A working, albeit limited, system is available quickly to demonstrate the feasibility and usefulness of the application to management. (6) The prototype serves as a basis for writing the specification for a production quality system.

Once a prototype is completed and the customer has given feedback, the requirements for the product must further be developed using a more formal SRS document. The SRS is a valuable communication tool that provides input into other phases, such as design and testing.

An alternative process model which combines the advantages of evolutionary prototyping with the control required for large scale development is known as

incremental development model. It involves developing the requirements and delivering the system in an incremental fashion. Thus, as a part of the system is delivered, the user may experiment with it and provide feedback to later parts of the system. Incremental development avoids the problems of constant change that characterize evolutionary prototyping. An overall system architecture is established early in the process to act as a framework. System components are incrementally developed and delivered within this framework.

The current project used the incremental prototyping model because the initial requirements were not clear. Records and Registration department preferred to see the product in incremental stages to get used to the system early enough and to give constructive feedback to the developer. The next chapter explains the interactions occurred between the developer and Records and Registrations department.

3. VARYING TYPE PROTOTYPE MODEL

During the initial stages of requirements analysis, it became clear in working with the sponsor of the current project that using prototyping model to develop this system would be of great value to both the sponsor and the developer. It helped in identifying and refining requirements. Firstly, we examined the existing manual system which was used to create proposals. Secondly, we identified all the data involved in this manual process. Most of the data was stored in a legacy database which did not follow a relational model. Some of the other data was stored in the form of WORD documents, HTML document and text files.

The requirements were not well defined and could not be identified initially due to the complexity of the project and due to the fact that existing data collected and stored through the manual process was distributed in various places. In order to identify the initial requirements we decided to use a throw away prototyping model to create user interfaces that helped the sponsor to visualize how an online automation system might be designed. Form based user interface prototypes were created and were shown to the project sponsor and to some of the possible end users. This gave end-users a concrete impression of the system capabilities and helped in establishing and validating systems requirements in an interactive way. Once all the basic requirements were identified, a formal software requirements documents was created that defined the functional and non functional requirements and the constraints under which the system must operate.

In principle, the functional requirements definition of a system should be both complete and consistent. Completeness means that all services required by the

user should be defined. Consistency means that requirements should not have contradictory definitions. Since the current project was large and complex, it was practically impossible to achieve requirements consistency and completeness. The reason for this was partly because of the inherent system complexities and partly because different viewpoints expressed by end users have inconsistent needs. After a deeper analysis of the requirements we discovered several problems during review meetings and later in the design phase.

Due to the above shortcomings, we decided to use incremental development model to implement our project. Once the system was delivered, the end users experimented with it and provided feedback to later parts of the system. An overall system architecture was established early in the process to act as a framework and the system components were incrementally developed and delivered within this framework using an evolutionary prototyping model.

4. THE DEVELOPMENT OF AUTOMATED CURRICULUM MANAGEMENT SYSTEM

The Automated Curriculum Management System was developed as a web based project. The first phase of development involved requirements gathering. In the second phase requirements were analyzed and a detailed specification was written. In the third phase, the product was implemented using the incremental prototype model. In the fourth phase the implemented tool was tested thoroughly by the developer and then by few prospective end users. Bugs were Identified and fixed in this phase too.

4.1. Requirements Analysis and Specification

In this phase the functional and non functional requirements were identified and analyzed. A vast majority of the requirements for the Automated Curriculum Management tool was captured by analyzing the existing legacy database system. Later, rapid prototypes were developed to communicate with the sponsor and to quickly discover additional requirements as well as changes to the existing requirements. The prototypes were tested by few end users to see if the tool was easy to use and to understand. Few modifications were made in the requirements specification after gathering feedback from these end users.

Microsoft Front Page 2003 and Macromedia Dream weaver was used to quickly develop prototype UI screens. JSP server-side scripting language was used to connect to an access database and prototype a small amount of non-GUI functionalities. This prototype included user logins, and simple record queries to establish communication between JSP pages and the database. The prototype implemented most of the functionalities in Curriculum Management Tool

including most of the screen shots. The sponsor's feedback was positive and the designing process started.

The following list summarizes the most important requirements of the Curriculum Management Tool.

- Security – Authentication; All users would need to login with a UWL 8.4 domain account username and password. Unlike some web-based software systems that automatically logs out a user when the user closes the web browser, the curriculum management tool requires its users to explicitly log out from the system.
- Security – Authorization; The system administrator must be able to add, update, and remove users and user roles, assign users to roles, limit the accessibility to each role including allowing read-only pages as well as pages that cannot be seen by specified user roles.
- The system needs to work with Oracle 9i as the primary database platform, but it also needs to be portable to other databases such as SQL Server, and Microsoft Access.
- Eliminate Inconsistent Data – Sometimes faculty members do not know the valid data to be entered in various fields. This causes inconsistent and incorrect data. The curriculum management system should have adequate help information and provide users the option to select appropriate data without having to type most of the information.
- Allow Large Objects - The paper forms allow limited space for entering the description and program requirement details. The Curriculum management tool should allow users to enter large amount of information.
- Workflow – When a user approves or disapproves a course/program proposal, it must be transferred to the inbox of the next respective user who is supposed to approve or make modification to this proposal.

• Email notifications – An automated email must be sent to the next respective user notifying awaiting approvals of course and program proposals.

• Scalability and Extensibility – The system must facilitate adding new modules in the future without major re-work.

4.2. User Classifications and Characteristics

Automating Curriculum Management Tasks tool is a multi-user web-based application. An authorization level refers to what functionalities a particular user can access. Automating Curriculum Management Tasks application has several types of users with different authorization levels. Following is the list of user classification.

- Administrator
- General Users
- Restricted Administrator

A user can login to this system using their UWL 8.4 username and password. Lightweight Directory Access Protocol, or LDAP (pronounced as "ell-dap") protocol is used to authenticate a user when they try to login to use this system. LDAP is a networking protocol for querying and modifying directory services running over TCP/IP. An LDAP directory usually follows the X.500 model: it is a tree of entries, each of which consists of a set of named attributes with values.

Registrar and Records Administrator come under the Administrator category and they will be given administrative rights and privileges. Users who come under administrator category can perform the following tasks:

1) Create new Course/Program proposal
2) View/Update/Delete course/program proposal created by any end users
3) View/Add/Update/Delete Department, College, Degree Type, Grade Pattern Information

4) Update Agenda Date for Course/Program proposal
5) Change the status of a proposal
6) View/Add/Delete/Update user information.

Department Chairs, Deans, Provost, UCC Committee chairs, GCC Committee chairs come under restricted administrator category. Users who come under restricted administrator category can perform the following tasks:

1. Create new Course/Program proposal.
2. View/ course/program proposal awaiting agenda date
3. Approve a proposal awaiting approval

Undergraduate curriculum committee members (UCC) and graduate curriculum committee (GCC) come under general user category. Users who come under general user category can perform the following tasks.

1. View/ course/program proposal awaiting agenda date

4.3. User Interface Design

The project sponsor wanted to implement the automation of curriculum management task tool as a web based application for the following reasons:

- Centralized data is secure and easy to backup
- Updates can be made quickly and easily.
- Information is accessible to a wide audience anywhere in the world.
- Access time - 24 hours a day, 7 days a week.
- .Familiar interface encourages use.
- Online training can be completed at user's own time and pace.
- Web-applications make collaboration easy, as basically everyone is using one “instance” of an application.

- Web applications need less people working on it. Desktop applications require several groups of people - programmers, people fixing up releases and distributions, people doing ports for different platforms, maintainers of older versions etc. Web applications require much smaller teams.
- No special configuration or changes needed on users PCs

There are disadvantages as well when using a web application. The disadvantages were also analyzed and listed to plan for future risks. Below is the list of disadvantages of a web user interface:

- Security risks: There are basically three overlapping types of risk which should be considered before designing a web based application- **(1)** The misuse of personal information knowingly or unknowingly provided by the end-user. **(2)**Interception of network data sent from browser to server or vice versa via network eavesdropping.**(3)** Active content that crashes the browser, damages the user's system, breaches the user's privacy, or merely creates an annoyance. **(4)**Bugs or misconfiguration problems in the Web server that allow unauthorized remote users to hack in to the webserver.
- It generally takes more time to develop a web-based application than a similar desktop application. Also it is more difficult for an application that needs lots of editing and ease-of-use features. No drag and drop features are available in web-based development toolkits without additional programming effort, intellisense, auto-completion, and context-sensitive help.
- There are some inconsistency issues with browsers since not all browsers render everything the same way. This is apparent in older versions of the browsers.
- No real standards

- Post backs can be distracting as they are “round trips” to the server, which can cause the screen to blink while all of the controls are refreshed with new information arriving from the server
- Responsiveness can be an issue depending on bandwidth and graphics, and other factors
- Session/State management can be an issue, for example, doing things in the middle of some operation
- Disconnected operations are a problem if the network is unavailable

A careful study of the technical expertise of the various types of users revealed that the user-interface for the system must be relatively easy to learn and use. Users with no computing experience should be able to learn to use the interface after a brief training session. The project sponsor emphasized that the users must have multiple screens for systems interaction. Switching from one task to another should be possible without losing sight of information generated during the first task.

5. DESIGN

The project sponsors wanted to develop the curriculum management tool using the existing design patterns and tools that were used by the ITS department in their previous web projects. The curriculum management tool was intended to be maintained and enhanced by the ITS department in the future, and so it was necessary to use the software tools and adhere to certain standards that were followed by the ITS web development team.

The developer met with the ITS web development team and got information about the software tools used by them to develop web application. The ITS department was using Java Server Pages (JSP) with Struts framework to create all of their web projects and oracle database management system as a backend. Struts is a free open-source framework for creating Java based web applications. Web applications differ from conventional websites, in that web applications can create a dynamic response. Many websites deliver only static pages. A web application can interact with databases and business logic engines to customize a response. Web applications based on JSP sometimes commingle database code, page design code, and control flow code. In practice, we find that unless these concerns are separated, larger applications become difficult to maintain. One way to separate concerns in a software application is to use a Model-View-Controller (MVC) architecture. The *Model* represents the business or database code, the *View* represents the page design code, and the *Controller* represents the navigational code. The Struts framework is designed to help developers create web applications that utilize MVC architecture. Struts provide the foundation, or framework, for building an MVC oriented application along with libraries and utilities.

5.1. High - Level architectural design

The high-level architectural design of the Curriculum Management Tool is presented in Figure 1.0. Users interact with Curriculum Management Tool through JSP web pages with a browser from a client machine for all functionalities of the system. This includes navigation, requesting information, submitting data, viewing reports and performing various other actions. The web pages reside on a web server running Apache Tomcat web server. A collection of java classes stored in the apache tomcat web container contains the code for various Curriculum management tool functionalities, and most of them interact with oracle 9i database.

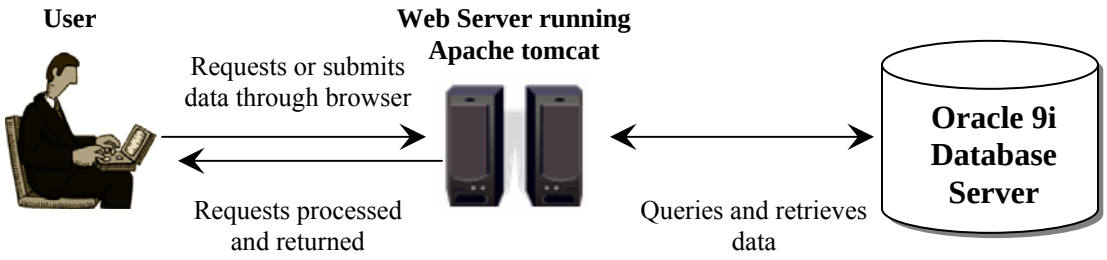


Figure 1: High Level Architecture of Curriculum Management Automation tool

Java Server Pages (JSP) is the preeminent Java technology for building applications that serve dynamic Web content. JSP brings the "write once, run anywhere" paradigm to interactive Web pages; others appreciate the fact that it is fairly simple to learn and lets them wield Java as a server-side scripting language. The biggest advantage of using JSP is that it helps effectively separate presentation from content. The curriculum management tool follows the Model 2 architecture.

Model 2 Architecture

In the MVC architecture, a central servlet, known as the controller, receives all requests for the application. The controller then processes the request and works with the model to prepare any data needed by the view (which is usually JSP) and forwards the data to JSP. The JSP then uses the data prepared by the controller to generate a response to the browser. In this architecture the business logic and presentation logic are separated from each other. This separations accommodates multiple interfaces to the application be they web, wireless or GUI[Swing]) and provides excellent reuse of code

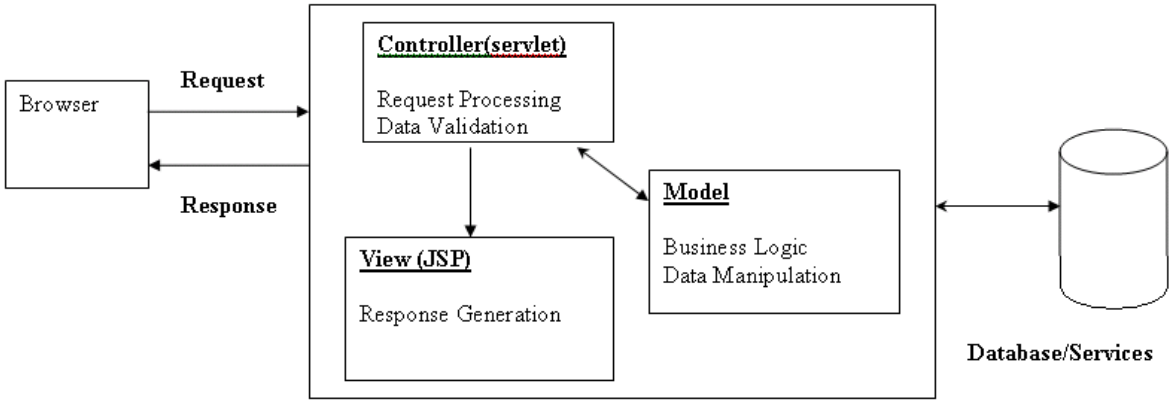


Figure 2. Model II Architecture

Model 1 architecture

Model 1 architecture ties together the business logic with presentation logic of the application. A request is made to a JSP or Servlet; the JSP or Servlet then handles all responsibilities for the request including, 1.) processing request, 2.) validating data 3.)handling business logic, and 4.) generating a response

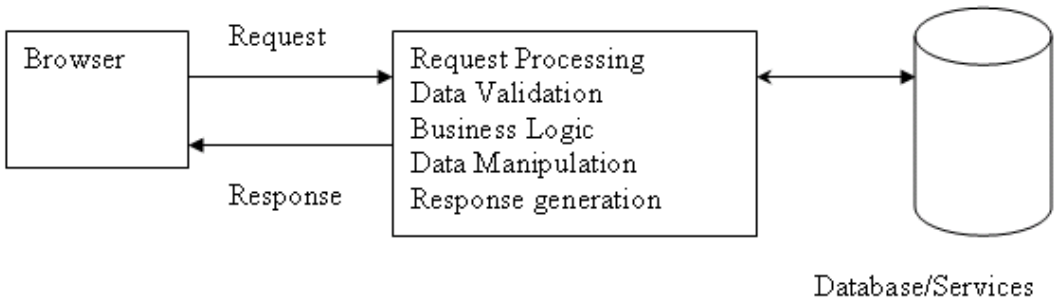


Figure 3. Model I Architecture

5.2. Detailed Architecture of Curriculum Management Tool

As stated earlier, all users of the Curriculum Management Automation Tool interact through web pages. Each web page represents a distinct functionality of the system. Most functionalities typically provide the ability to add, update, delete, and query records to and from the Course Catalog Database. Furthermore, some of the web pages may themselves be run-time generated pages in which the user controls or selection data are "generated" by querying the database for corresponding users. This is an advanced JSP technique by which an empty web page can be used as a template for the "real" web pages at run-time. This technique allows for extensibility as the presentation layer is further decoupled from the application layer. It is also a security feature as a user cannot simply browse to a web page without logging in because the template page would be

empty. In this project, considerable amount of validation and verification is done to maintain consistency between data stored in the Course Catalog Database.

The Curriculum Management Automation Tool revolves around two main objects namely, the Course and Program Objects. The course and program objects are interrelated and share similar data fields. These objects contain a complex set of attributes, most of them have single values but some of them have multiple values. In order to reduce the complexity of these objects, the developer separated the Course object into sub categories and created an object for each sub category. For example, the course object was separated into the following sub categories Course Justification, Non Course Prerequisites, Prerequisites, Co-requisites, Basic Course Information, Course Objective, Course Outline, Course affiliation, Program affiliation, Course description, Cross listed department list, Slash course prerequisites, Slash course co-requisites and Slash course information.

While adding a new course, the following conditions are verified:

1.) Do not add a course if another course with the same course number and course abbreviation is active in the Course Catalog Database.
2.) Do not add a course if another course with the same course number and course abbreviation is available in the Course Catalog Database in the past 10 years, even if it is inactive.
3.) Do not add a course if another course with the same slash course number and slash course abbreviation is active in the Course Catalog Database.
4.) Do not add a course if another course with the same slash course number and slash course abbreviation is available in the course catalog database in the past 10 years even if it is inactive.
5.) Do not add a course if another course with the same course number and course abbreviation exists in an existing cross listed department list.

The following two figures show the UML class diagram for the course catalog web application.

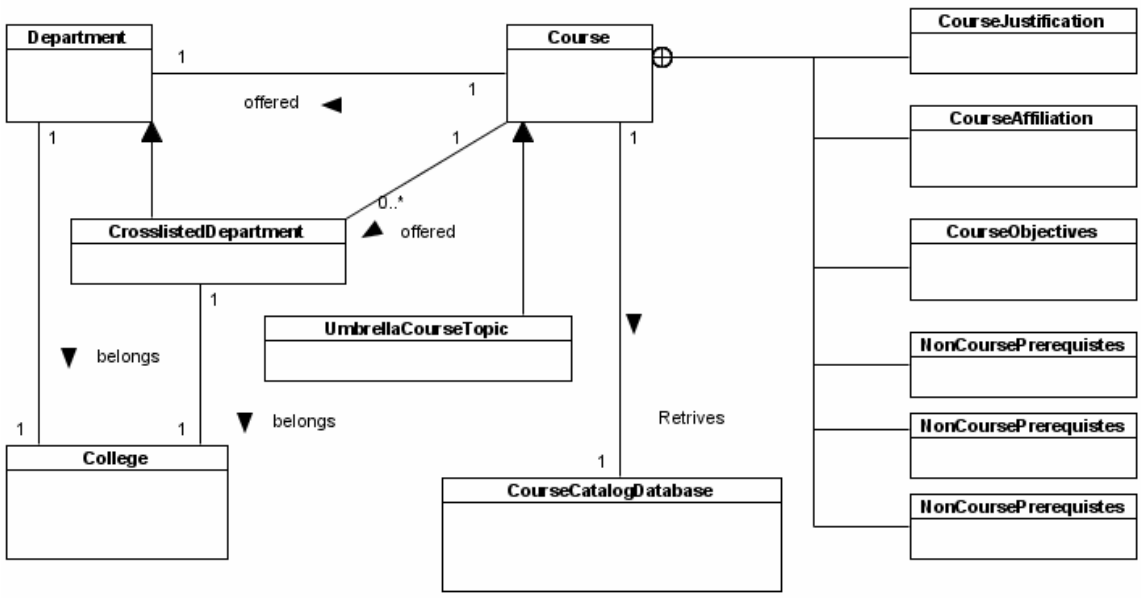


Figure 4. Class Diagram for Course Proposal System

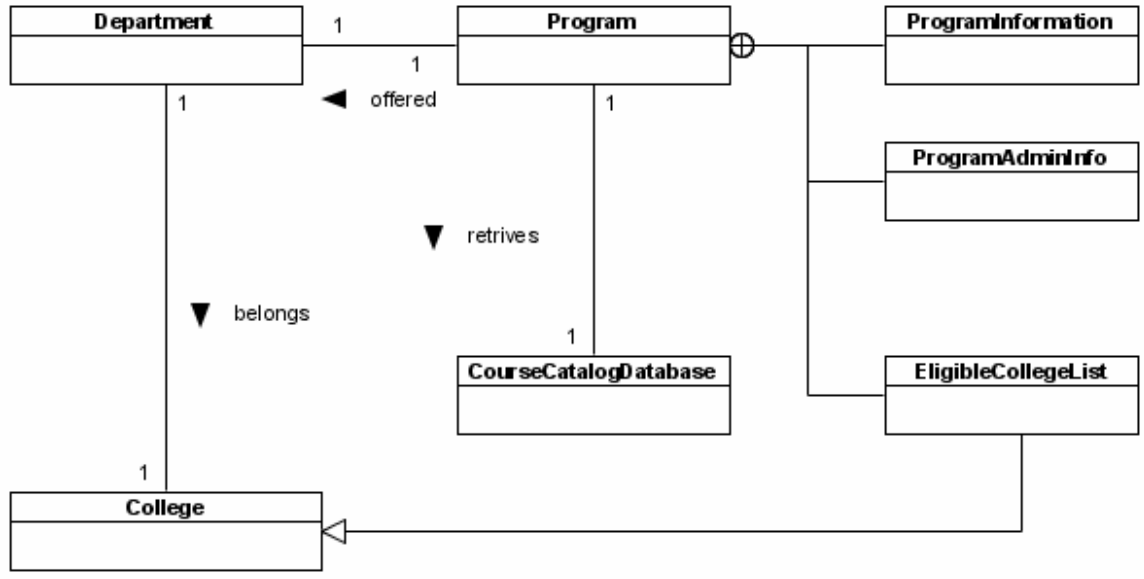


Figure 5. Class Diagram for Program Proposal System

The Curriculum Management Automation Tool itself is composed of multiple tiers or layers, including a presentation layer, an application logic layer, and a database layer. Due to space limitations in this manuscript, a detailed design is

shown for only the subset of the core functionality which consists of course and program information as shown in Figure 5.

The Curriculum management tool is composed of several core functionalities including the following:

Systems Administration - This functionality provides the administrator to add, modify, delete and view administrative information. Administrative information is a collection of data that is required to be entered in the Course Catalog Database before any user can start using the Curriculum Management Automation Tool. Examples of administrative information include Department, College, Degree Type, Grade Pattern, and Special Users with Restricted Admin rights Information.

Login authentication – This functionality allows an end user to login to the curriculum management tool using their existing UWL 8.4 Login ID.

New Course proposals - This functionality provides an end user the ability to add, modify, delete or view new course proposal information.

Existing course - This functionality provides an end user the ability to modify, delete or view course proposal information for an existing course.

New program - This functionality provides an end user the ability to add, modify, delete or view new program proposal information.

Existing program - This functionality provides an end user the ability to modify, delete or view program proposal information for an existing program.

A Brief Introduction to Web-Page Programming and its Challenges

The World Wide Web is intrinsically stateless because each request for a new Web page is processed without any knowledge of previous pages requested. This

is one of the main drawbacks to the HTTP protocol. Because maintaining state is extremely useful, programmers have developed a number of techniques to add state to the World Wide Web. By using session variables, cookies and hidden form variables the previous state information can be preserved. HTTP cookies, sometimes known as web cookies or just cookies, are parcels of text sent by a server to a web browser and then sent back unchanged by the browser each time it accesses that server. HTTP cookies are used for authenticating, tracking, and maintaining specific information about users, such as site preferences and the contents of their electronic shopping carts. The Session variable is used to store information about, or change settings for a user session. Variables stored in the Session object hold information about one single user, and are available to all pages in one application.

In a production system, when an exception is thrown it is likely that the end user is unable to process his or her request. When such an exception occurs, the end user normally expects the following:

- A clear message indicating that an error has occurred
- A unique error number that he can use upon accessing a readily available customer support system
- Quick resolution of the problem, and the assurance that his/her request has been processed, or will be processed within a set time frame

The customer service team or the developer should receive immediate error notification when an error occurs, so that the service representative is aware of the problem before the customer calls for resolution. Furthermore, the service representative should be able to cross-reference a user's unique error number and the production logs for quick identification of the problem -- preferably up to the exact line number or the exact method. In order to provide both the end user and

the support team with the tools and services they need, you must have a clear picture, as you are building a system, of everything that can go wrong with it once it is deployed.

Exceptions are classified in different ways. Following is the classification of exceptions into three broad categories:

- **JVM exceptions**: This type of exception is thrown by the Java Virtual Machine (JVM). "OutOfMemoryError" is one common example of a JVM exception. There is nothing one can do about JVM exceptions. They indicate a fatal situation. The only graceful exit is to stop the application server, maybe beef up the hardware resources, and restart the system.

- **Application exceptions**: An application exception is a custom exception thrown by the application or a third-party library. These are essentially checked exceptions; they denote that some condition in the business logic has not been met. Under these conditions, the caller of the method can gracefully handle the situation and take an alternative path.

- **System exceptions**: Most often system exceptions are thrown as subclasses of RuntimeException by the JVM. A NullPointerException, or an ArrayOutOfBoundsException, for example, will be thrown due to a bug in the code. Another type of system exception occurs when the system encounters an improperly configured resource such as a misspelled database connection string. In this case, it will throw a checked exception. It makes a lot of sense to catch these checked system exceptions and throw them as unchecked exceptions. The rule of thumb is, if nothing could be done about an exception, it is a system exception and it should be thrown as an unchecked exception

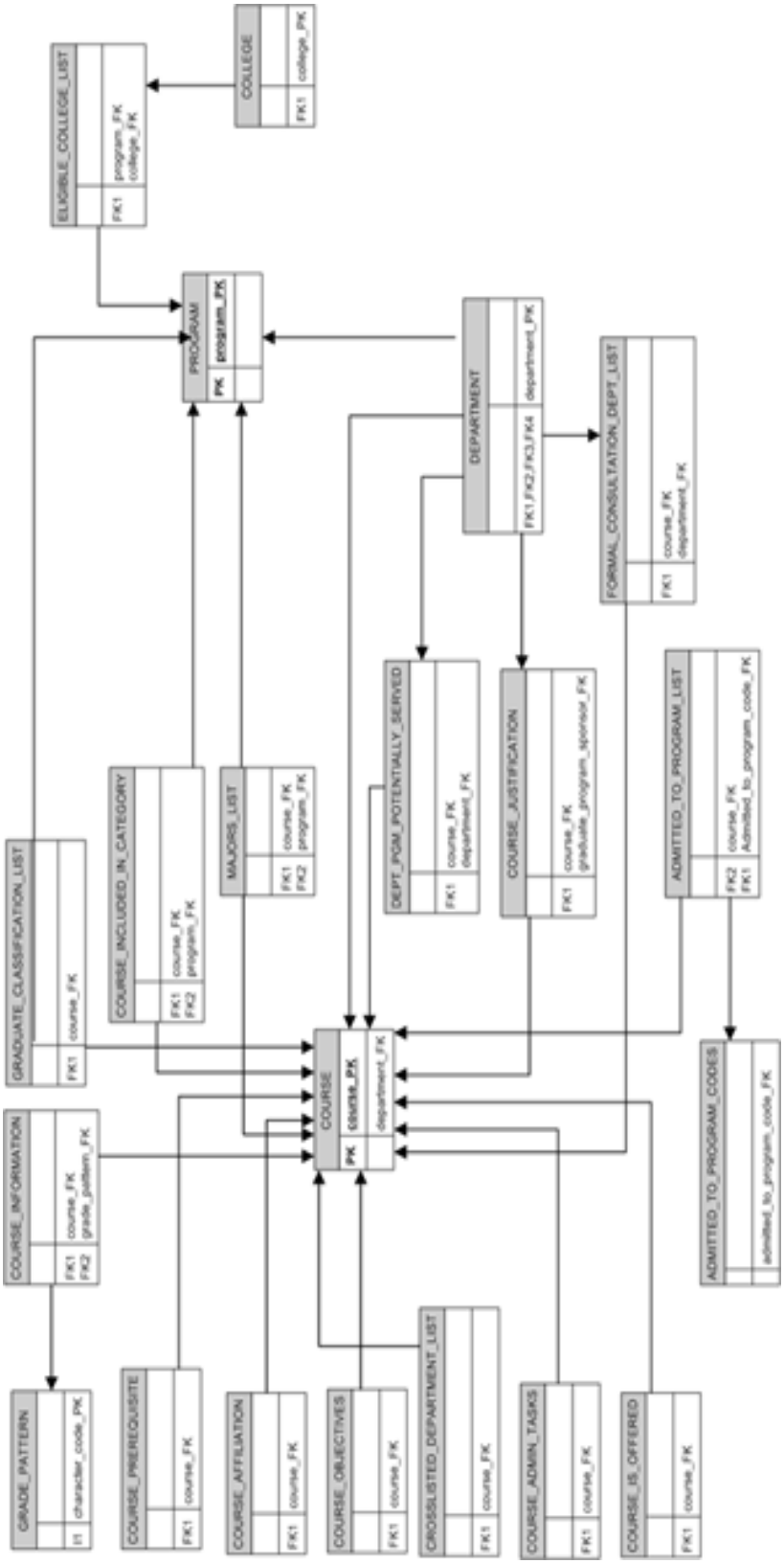
The developer has adhered to the above mentioned best practices in exception handling. The developer has designed the system in such a way that whenever an

exception arises in any part of the web application, the exception is caught and gracefully handled by notifying the user about the error in simple terms and creating a detailed expectation log in the server location for future reference. The exception log contains detailed information about the user logged in, the time of the exception occurrence, the line number of the code which caused the exception and stack trace information. Using the above information, the developer has the ability to identify and fix any bugs or errors that might occur in the future while the customer is using the curriculum management automation tool. Object oriented programming methodology extensively supports designing the exception handling mechanism.

5.3. Database Design of Curriculum Management Automation Tool

Figure 6 shows the database design for the project using an Entity Relationship Diagram (ERD). In this design, each entity represents a table in the database along with the primary key (PK), foreign keys (FK), and uniqueness constraint (U) for each table.

Figure 6: Database Entity Relationship Diagram


COLLEGE
college_PK
FK1
ELIGIBLE_COLLEGE_LIST
program_FK
college_FK
FK1
PROGRAM
program_PK
PK
DEPARTMENT
department_PK
FK1,FK2,FK3,FK4
FORMAL_CONSULTATION_DEPT_LIST
course_FK
department_FK
FK1
GRADUATE_CLASSIFICATION_LIST
course_FK
FK1
COURSE_INCLUDED_IN_CATEGORY
course_FK
program_FK
FK1
FK2
MAJORS_LIST
course_FK
program_FK
FK1
FK2
DEPT_PGM_POTENTIALLY_SERVED
course_FK
department_FK
FK1
COURSE_JUSTIFICATION
course_FK
graduate_program_sponsor_FK
FK1
ADMITTED_TO_PROGRAM_LIST
course_FK
Admitted_to_program_code_FK
FK2
FK1
COURSE
course_PK
department_FK
PK
COURSE_INFORMATION
course_FK
grade_pattern_FK
FK1
FK2
ADMITTED_TO_PROGRAM_CODES
admitted_to_program_code_FK
GRADE_PATTERN
character_code_PK
PK
COURSE_PREREQUISITE
course_FK
FK1
COURSE_AFFILIATION
course_FK
FK1
COURSE_OBJECTIVES
course_FK
FK1
CROSSLISTED_DEPARTMENT_LIST
course_FK
FK1
COURSE_ADMIN_TASKS
course_FK
FK1
COURSE_IS_OFFERED
course_FK
FK1

For the current project, requirements analysis and data modeling were done at the same. As information was collected from the sponsor, data objects were identified and classified as entities, attributes, or relationship; assigned names; and, defined using terms familiar to the end-users. The objects were then modeled and analyzed using an ER diagram. The diagram was reviewed by the developer and the end-users to determine its completeness and accuracy. Using the feedback from the sponsor, the model was corrected and modified, which required additional information to be collected. The review and edit cycle continued until the model was complete.

One of the most important requirements of the curriculum management tool was to create a new database which will accommodate most of the existing data fields from the legacy database and to introduce new data fields. During the requirements gathering phase, the data requirements of the database in terms of primitive objects were determined first. Secondly, rules governing the integrity of the data were identified. Thirdly, the relationship among the objects were identified and classified. Finally the types of transactions that will be executed on the database and the interactions between the data and the transactions were described.

The data within our databases was expected to follow the rules and constraints placed within the data models. Without constraints the data would hold no meaning. The reliability and integrity of that data would always be in question and the users of such data would always question the validity of it. If a database were to neglect the rules and overstep the boundaries that were placed upon it, there would be mayhem within the database and that database would cease to exist for any value. Any table in the database that has a primary key or unique key can be referenced by another table by setting up a rule that relates those tables and governs the relationship. In this relationship there are two tables - parent table and child table. The child table uses a foreign key to reference the parent table's

primary key or unique key. If an attempt to alter or delete data in the parent table and there are rows in the child table, the transaction is not allowed. If the parent data is modified, then all the children's table data are also modified. If the parent is deleted then all the children are deleted. A lot of effort was devoted to the planning and analysis of the course catalog database. Each table in the Course Catalog Database is related to another table. Constraints were also clearly defined while building the database initially

The legacy database consists of a table with 128 attributes and most of the attributes were named with 3 or 4 characters which made it difficult for developers to understand what the data fields meant. The new database consists of 23 tables which accommodate old and new data fields. All the tables in the new database were normalized to prevent redundancy of data. Relationships and constraints were clearly defined in these tables to maintain the consistency of data. The tables and attributes were named in full instead of using abbreviations to eliminate ambiguity. This made it easy for the developers and the database administrators to understand what type of data is stored in the tables instead of checking an index document which had details about each table.

Another challenge related to a web based application is synchronizing concurrent access to objects from multiple web transactions. For example, consider the situation when two users connected to the curriculum management tool application and work on the same record. If one of the users waits just a minute to change a field in the record and the other one meanwhile modifies that field. This condition creates an inconsistent state of data. Dealing with concurrency issue in a web based application is very complex due to the fact that web applications are stateless in nature. To deal with this issue, the developer had to create a data field named ‘sync_lock’ in all the database tables which contains integer values. Whenever a user retrieves a set of data from a specified table, the

sync_lock value is also transmitted to the client. When a client tries to update a specified record the system checks if sunc_lock value is the same and if true the sync-lock value is incremented to denote an update. In a scenario where two users are trying to modify the same record the sync_lock value will be incremented when the first user updates the record. When the second user tries to update the same record the sync_lock value stored in the client machine will not match the value stored in the database which in turn will thrown an exception notifying the user that this record was modified by another user recently and sends the refreshed data which was modified back to the user.

The developer used the JDBC API to interact with the Oracle 9i database and java. The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database [20]. JDBC helps a developer to write java applications that manage these three programming activities:

1.) Connect to a data source, like a database

2.) Send queries and update statements to the database

3.) Retrieve and process the results received from the database in answer to your query

The java business logic code contains a set of classes that takes care of the database interaction functionality. A class named DatabaseConnection sets the database connection string and also takes care of open, close, commit and rollback operations related to a database transaction. The class named CourseCatalogDatabase contains all methods that deal with inserting, deleting, updating and retrieving data stored in the database. A `PreparedStatement` object is used in all the methods to send SQL statements to the database. The developer chose to use PreparedStatement over the normal SQL Statement object because it was more convenient to use and easy to debug when there is an error in a SQL Statement.

In the Curriculum management tool there was a need to store large text objects into the database. In Oracle 9i, a varchar() datatype can only store a maximum of 4000 characters but the program description, course description and course objective data fields in the curriculum management tool can contain more than 4000 characters depending on the course or program complexity. In order to accommodate these large data fields the developer decided to use Oracle's Character large object CLOB to tackle this problem. A Character Large Object (or CLOB) is a collection of character data in a database management system, usually stored in a separate location that is referenced in the table itself. CLOBs are typically text exceeding 4000 characters in length. CLOBs usually have very high size limits, on the order of 2GB or more. In oracle 9i the size limit of CLOB is 4 GB.

5.4. Deploying Curriculum Management Automation Tool

Curriculum Management Automation Tool is deployed by installing the application to a web server running Tomcat. The web server must also have the Java Run Time System and JDK (Java SE Development Kit) installed on it so that the tomcat web container can compile JSP pages and servlets. If JDK is installed in c:\ drive, the following JAR (Java Archive file) files must be present in the C:\jdk1.5.0_07\jre\lib\ext folder.

- Activation.jar
- Mail.jar
- Ojdbc14.jar

Ojdbc14.jar contains a set of classes and interfaces written in Java to allow other Java programs to send SQL statements to a relational database management system. Activation.jar and Mail.jar contains classes which are used to send emails during the workflow process.

Curriculum Management Automation Tool was created using NetBeans IDE. The easiest way to deploy this web application is to open the Course Catalog Web Application project from NetBeans 5.0 and pressing the F6 button; this will in turn start the built-in Jakarta tomcat web server, build all the source code and open this web application in the default internet browser.

A web deployment descriptor file (web.xml) is used to tell the web container how to run the servlets and JSPs. The deployment descriptor (DD) provides a declarative mechanism for customizing your web application without touching the source code. It also lets the user adapt the application to different resources (like databases), without having to recompile and test any code and makes it easier to maintain dynamic security information such as access control lists and security roles.

Instead of hard coding the real path and filename into all the JSPs and other HTML pages, the developer had mapped the JSP filename in the web.xml file. This makes the web application more flexible and enables the developer to move things around without having the maintenance nightmare of tracking down and changing client code that refers to old location of the servlets files. Another reason for using servlet/JSP mapping was to make the web application more secure by hiding the structure of the web application files. If an end user attempts to navigate directly to the servlet / JSP file without going through the right pages or forms, then the end user will be able to see the real path and he/she can type it onto the browser and try to access it directly.

Deploying JSP Web Application in another web server

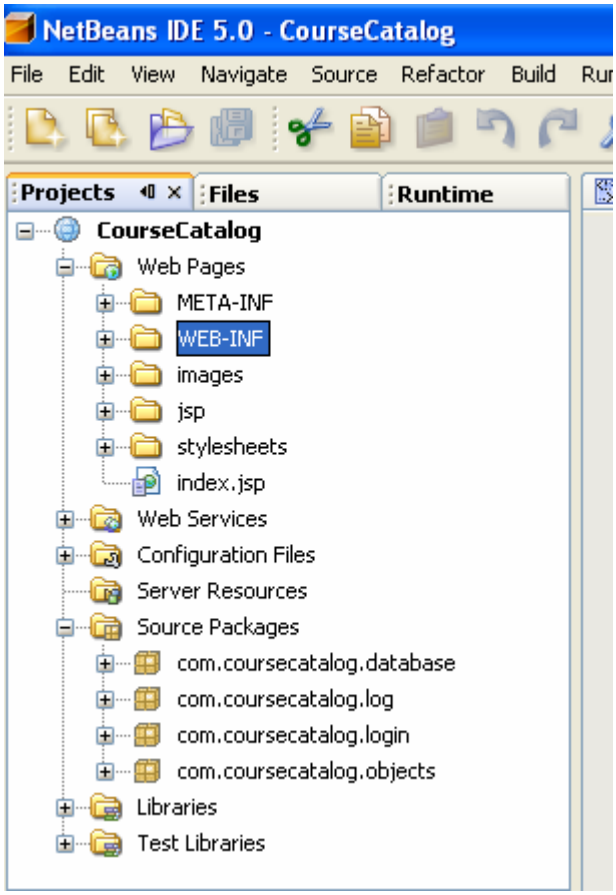
NetBeans 5.0 creates a WAR file automatically when a web application is built. A WAR file is simply a snapshot of web app structure, in a nice portable, compressed form (it is really just a JAR file). This WAR file can be used to deploy the web application in any web servers that supports JSP and Java.

WAR file structure -

- The static HTML files and JSPs are stored in the top level directory.
- The Servlet and related Java[tm] technology class files must be stored in the `WEB-INF/classes` directory.
- Any auxiliary library JAR files must be stored in the `WEB-INF/lib` directory.
- The deployment descriptor is stored as a file named `web.xml` in the `WEB-INF` directory.

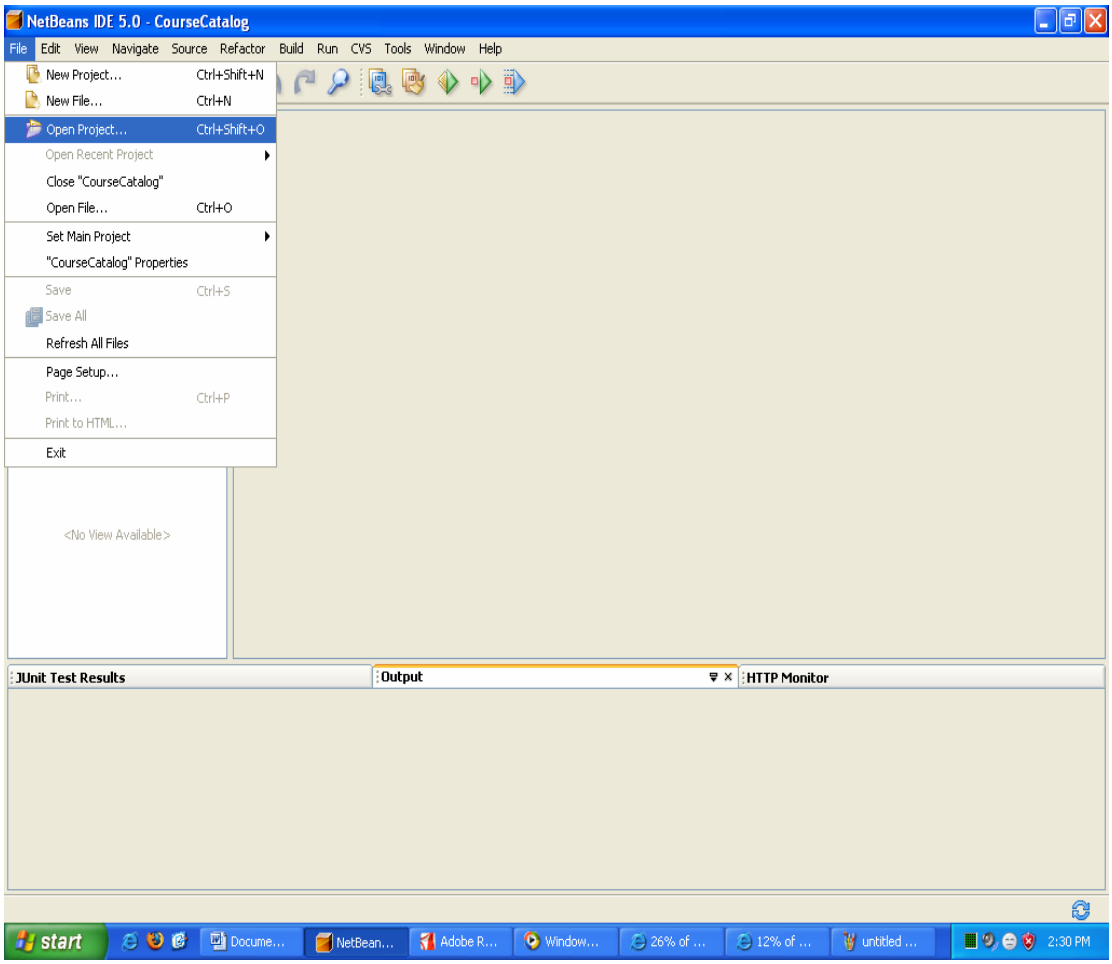
Netbean IDE is a GUI workbench for developing code, featuring facilities like symbolic debugging, version control, and data-structure browsing. Netbean enables software developers to create cross-platform Java desktop, enterprise and web applications. Netbean IDE Runs on Windows, Linux, MacOS, as well as Solaris. It is easy to install and use, works right out of the box - and it is open-source and free. IDE stands for Integrated Development Environment.

NetBeans 5.0 Web Application Directory Structure

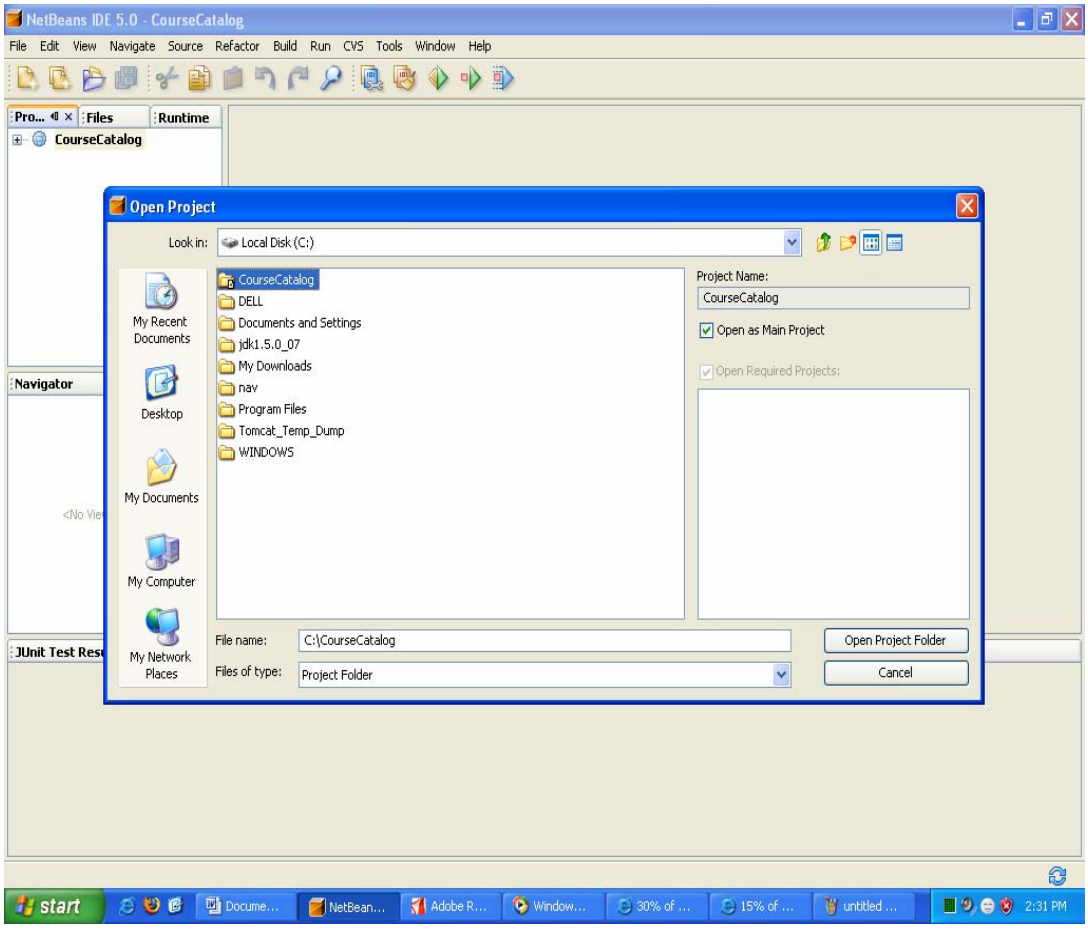


CourseCatalog is the webApplication name and the root folder
Web Pages/ WEB-INF : folder consists of the web.xml (Deployment Description) file.
Web Pages/JSP : folder consists of the JSP file
Web Pages/stylesheets : folder consists the external cascading style sheet file.
Web Pages/images : folder consists if image files used in the web application
Source Packages : folder lets the developer to create packages and java classes.
Following steps describes a way to deploy the webapplication from netbeans.

Step 1: Open NetBeans 5.0 IDE and click on the open project option. CourseCatalog web project consist of various files of the following type – .jsp, .java, .class, .jar, .xml etc. Netbean IDE understands and places the files with different extension in separate folders in a organized manner.



Step 2: Select the Course Catalog Project and Click on the run options or press F6 Button to run the web application. Netbean IDE compiles the web project , creats a build , finally creates a war file and finally opens the initial main page in the browser.



5.5. Web User Interface Design

The Curriculum Management Tool contains a web-based graphical user interface. Web-based user interfaces accept input and provide output by generating web pages which are transported via the Internet and viewed by the user using a web browser program. This interface consists of many JSP web pages through which the users interact. JSP (Java Server Pages) are normal HTML with Java code pieces embedded in them. Java Server Page (JSP) is a technology for controlling the content or appearance of Web pages through the use of servlet's, small programs that are specified in the Web page and run on the Web server to modify the Web page before it is sent to the user who requested it. Sun Microsystems, the developer of Java, also refers to the JSP technology as the Servlet application program interface (API). JSP is comparable to Microsoft's Active Server Page (ASP) technology.

The code for the web page is written using an object-oriented programming language (Java) using classes, methods, attributes, and so on. The code contains classes which define various objects like Course, Department, Student, Faculty, Database, etc. Code may also be embedded directly into HTML but the developer made sure that the code is separated to enforce the model 2 web architecture. The separation of view layer and the code layer allows for better encapsulation, easier maintenance, better readability, easier debugging and easier reusability. Regardless of the design pattern methods used for creating the web pages, all JSP code is processed on the web server and is replaced with HTML before the response is sent back to the requesting client. This allows multiple browsers like Internet Explorer and Netscape Navigator as well as multiple versions of the browsers to interact with the system, even though the client browsers may be on non-Windows platforms or older versions of the Windows operating system.

The most important requirement of this tool was to design a easy-to-use and at the same time, easy to extend GUI. A noteworthy aspect to the Curriculum management tool user interface is that the look and feel of the entire user interface is modifiable without additional programming. Properties related to the appearance of the application are exposed through a Cascading Style Sheet (CSS) including background colors, button styles, fonts and colors for various text and buttons, and so on.

In some web pages the users had to specify the date and time value. Instead of using a conventional text box field to get the date value from the user, the developer used a date picker component which allowed the users to make a quick and straightforward date and time selection. A Date picker is a JavaScript component for pop-up calendar style date selection on a web page. Visually it replicates a typical Windows® interface, presenting the user with a familiar 'look and feel,' making them more efficient and effective. The use of a date picker eliminates confusion over the date format (US, European, etc) and it reduces the keyboard-to-mouse hand movement.

The user interface design principles were followed strictly to make this tool simple and easy to use. All needed options and materials for a given task were made visible without distracting the user with extraneous or redundant information. The users were informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users. The user interface was organized in a meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another.

The user interface looks similar to all the existing web applications created by the ITS department. These existing web applications are used by most of the faculty and non faculty staff of University of Wisconsin - Lacrosse. The curriculum management tool was in designed in such a way so that the faculty and non faculty members can use this tool without any new additional training. Forms validation was done on the client-side using JavaScript to save time and bandwidth by minimizing request/response roundtrips back to the web server. Client side form field validation gives the developer more options to point out to the user where they've gone wrong in filling out the form. Future enhancements or implementations to the user interface should utilize AJAX, Microsoft .NET, Java Server faces or similar technologies to provide real-time control in a separate program, eliminating the need to refresh a traditional HTML based web browser.

6. LIMITATIONS

The Curriculum Management Automation Tool is a good start in the direction of its intended goals. It automates creation and modification of course proposals and program proposals using web forms, and automates the workflow involved in processing the proposals. Despite this good start, there are several significant enhancements required to make it directly usable – some of these can be seen in the next section of this manuscript. Besides these missing features, general limitations will be listed here.

One limitation of the Curriculum Management Automation Tool is that each method in the database class uses separate connections to access the database. Creating a connection to the database is a very resource intensive operation, and therefore takes a relatively long time to complete. If a program services many intermittent requests for access to the database, creating a new connection for each request is obviously undesirable because it consumes more time. Rather than create a new connection for each database request, a better way is to initially create one connection to the database; this connection may then be used whenever the program needs to temporarily access the database. This is known as connection pooling.

Curriculum Management Automation Tool requires system administrators to be aware and cautious in providing adequate security by appropriately configuring the application during installation. The application, by default, does not provide the security measures. The TCP/IP protocol was not designed with security in mind; hence it is vulnerable to network eavesdropping (to be explained shortly). When confidential documents are transmitted from the web server to the browser, or when an end-user sends private information back to the server inside a fill-out form, someone may be listening in.

Following are a list of threats while using an online web application

1.) The misuse of personal information knowingly or unknowingly provided by the end-user.
2.) Interception of network data sent from browser to server or vice versa via network is known as eavesdropping. Eavesdroppers can operate from any point on the pathway between browser and server, both included.

It is important to realize that "secure" browsers and servers are only designed to protect confidential information against network eavesdropping. Without system security on browser and server sides, confidential documents are vulnerable to interception. The single most important step is to increase the site security by responding to suspected security breaches by following certain protocols like SSL and secure Http (S-HTTP). Short for Secure Sockets Layer, a protocol developed by Netscape for transmitting private documents via the Internet. SSL uses a cryptographic system that uses two keys to encrypt data − a public key known to everyone and a private or secret key known only to the recipient of the message. Both Netscape Navigator and Internet Explorer support SSL. Many web sites use the protocol to obtain confidential user information, such as credit card numbers. By convention, URLs that require an SSL connection start with https: instead of http. SSL creates a secure connection between a client and a server, over which any amount of data can be sent securely.

Another limitation is whenever an administrator creates users or modifies user information the program does not validate if the username is a valid UWL 8.4 user ID. The creation and modification of user information should be tied up to the LDAP server to maintain consistency. The curriculum management tool has an email remainder feature which keeps the workflow automated. If there is any inconsistent data while creating a user then the respective user might not receive the automated email remainders due to wrong information stored in the database.

Text Area form element has been used extensively in various web forms to gather voluminous data from the end user while creating a course or program proposal. Unfortunately a text area does not have the ability to add bold, color, various fonts and font sizes. The project sponsor wanted to add these feature in future releases of this tool in order to provide users the ability to format the large amount of text which is typed or pasted by the users in the text area field. Allowing formatting of text will make the data easy to read and will allow the users to paste text from a word document.

7. CONTINUING WORK

Due to the size of the project, some of the functionalities listed in the requirements document were not completed. The list of requirements and enhancements to be implemented in the future are as follows

- Create of database tables to accommodate approved course and program proposal information.
- Automate process to move a course/program from the proposal database to the Approved database.
- Create procedures for Umbrella Course Topic.
- Create web forms and protocol/procedures for General Education submission
- Automate the generation of the undergraduate and graduate academic course catalog by retrieving necessary data from the database.
- Generate administrative and user level reports.
- Populate the approved course/program database (information must be drawn from this database into the proposal system to allow a user to modify existing courses and programs).
- Delete course/program proposal from the database when the proposal is approved.
- Create procedures for Modify Course protocols.
- Create procedures for Modify Program protocols.
- View and print existing program details.
- View and print umbrella course topic proposals.
- View and print existing course topics details.
- View, update and delete and existing program.
- View and print existing course details
- View, update and delete and existing course

- Generate reports for meeting agenda, meeting minutes, course proposals, program proposals, umbrella course topics, catalog copy material.
- Storing prerequisite course information in an organized manner so that validation of these prerequisite courses is possible using predicate logic.

8. CONCLUSION

Curriculum management automation tool is a multi-tired web based application. This application can be used by all UWL faculty staff members, Records and Registration Staff members and some of the students who are members of UCC or GCC committees. This tool automates curriculum proposal and curricular data maintenance. It also allows the information management to be performed through a single system versus many disparate systems. Previously curricular data was stored in different media such as Microsoft WORD documents. Html documents, text files and legacy database which did not follow the relational database model. The current database is normalized and stored in tables that follows a relational database model.

The current system supports the following functionalities through a web-based user interface:

Core Functionalities:

- Create a new course proposal
- View, update, and delete an existing course proposal
- Create a new program proposal
- View, update, and delete an existing program proposal
- View, update, and delete an existing umbrella topic course proposal
- View, update and delete and existing umbrella topic course
- View, Add, update, delete Prerequisites List
- View and print new course proposals
- View and print new program proposals
- Approve/Disapprove course/program proposal by the creator
- Approve/Disapprove course/program proposal by a Department Chair
- Approve/Disapprove course/program proposal by a Dean
- Approve/Disapprove course/program proposal by Registrar

- Send emails to respective end users who are supposed to approve a proposal.

System Administration Functionalities:

- Add, update, delete, view Department
- Add, update, delete, view College
- Add, update, delete, view Degree Type Information
- Add, update, delete, view Admitted to Program codes information
- Add, update , delete, view Grade Pattern Information
- Add, update , delete, view users Information
- Update agenda date for a course/program proposal.
- View list of course/program proposals in initial state
- View list of course/program proposals awaiting agenda date
- View list of approved course/program proposals

General Capabilities:

• Relational database storage

• LDAP(Lightweight Directory Access Protocol) authentication mechanism used to

let University of Wisconsin – La Crosse users login

• Logout with automatic session timeouts.

• Multi-tier architectures supported.

• Highly extensible application.

• Object oriented design.

9. BIBLIOGRAPHY

[1] Lannes L. Morris-Murphy ., Oracle 9i: SQL with an introduction to PL/SQL ., Thomson Course technology.,

[2] Bryan Basham, Kathy Sierra and Bert Bates., Head First Servlets and JSP., O'Reilly Media 2005

[3] Herbert Schildt., The Complete Reference, J2SE 5 Edition

[4] Kevin Loney, George Koch ., Oracle 8i The Complete Reference., 2000

[5] Jason Price, Oracle 9i JDBC Programming,. 2000

[6] Chuck Cavaness , Programming Jakarta Struts, 2nd edition

[7] James Holmes., Struts: The Complete Reference., (Osborne Complete Reference Series)

[8] Jason Gilliam, Charlton Ting, R.Allen wyke. Pure JavaScript., Sams Publishing 1999

[9] Eric Freeman & Elisabeth Freeman., Head First Design Patterns. O'Reilly Media 2004

[10] Philip M.Lewis, Arthur Bernstein, Michael Kifer., Databases and Transaction Processing, An application-oriented Approach., Addison Wesley., 2002

[11] Subrahmanyam Allamaraju, Cedric Beust, Marc Wilcox, and Sameer Tyagi., Professional Java Server Programming J2EE, 1.3 wrox press.,Sep 2001

[12] Thomas Bishop , Glenn E. Mitchell , John Bell , Bjarki Holm , Danny Ayers , Carl Calvert Bettis , Sean Rhody , Tony Loton , Michael Bogovich , Mark Wilcox , Lin Kelly Poon , Nitin Nanda , Rick Grehan , Matthew Ferris , Kelly Lin Poon .,

Professional Java Data., Wrox Press., 2001

[13] Hans Van Vliet., Software Engineering Principles and Practice., John Wiley & Sons

Ltd., Second Edition., 2002

[14] Ian Sommerville., Software Engineering, 5th edition, Addison Wesley, 1996

[15] Edited by Diane Schumacher and Jeri Anibas, Records and Registration

Undergraduate Catalog 2005 – 2007., May 1, 2005

[16] Bert Bates, Kathy Sierra., Head First Java: Your Brain on Java - A Learner's Guide.,

O'Reilly Media., 2003

[17] http://en.wikipedia.org., Computer Definitions

[18] www.w3schools.com., JavaScript , Html , SQL Tutorial

[19] Ben Galbraith, Peter den Haan, Lance Lavandowska, Sathya Narayanan Panduranga,

Krishnaraj Perrumal, Erick Sgarbi, Beginning JSP 2.0., Wrox Press Ltd 2003

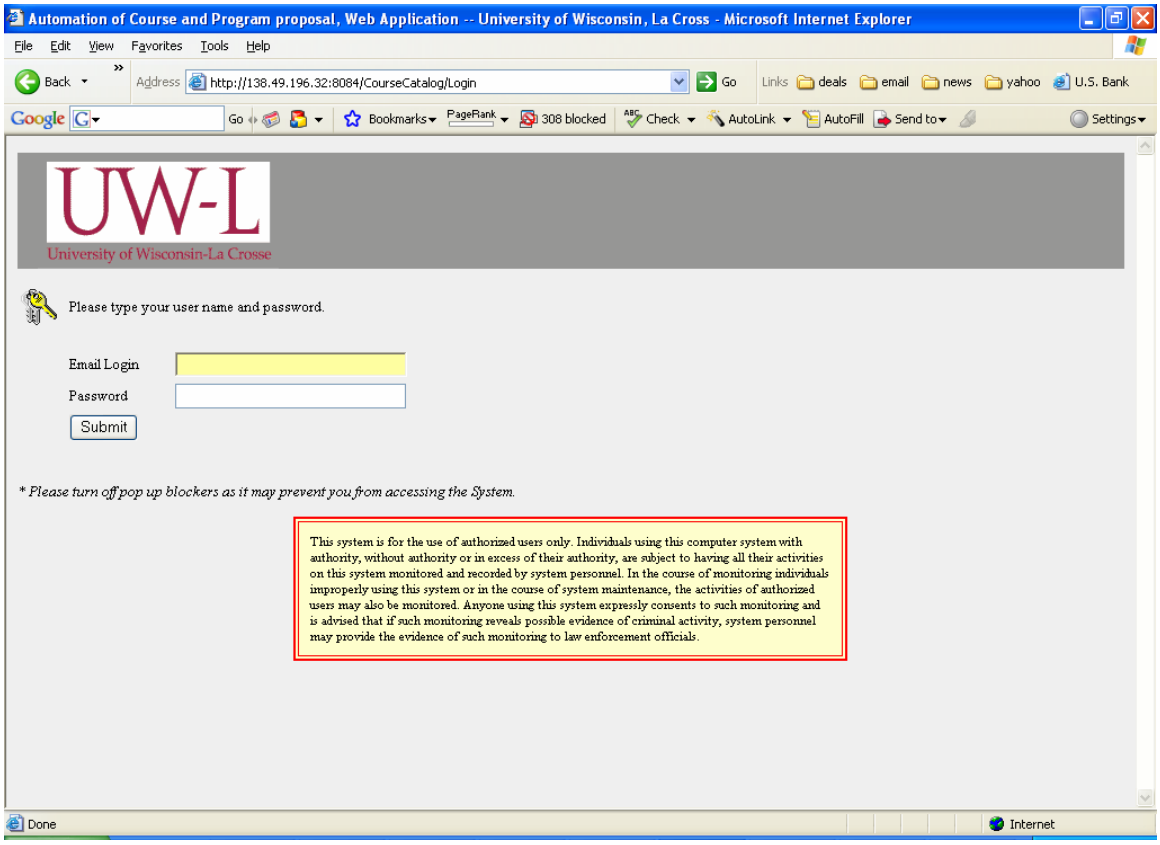
[20] Http://java.sun.com/docs/books/tutorial/jdbc/overview/index.html., JDBC Tutorial

[21] Http://www.netbeans.org/., Netbean IDE reference.

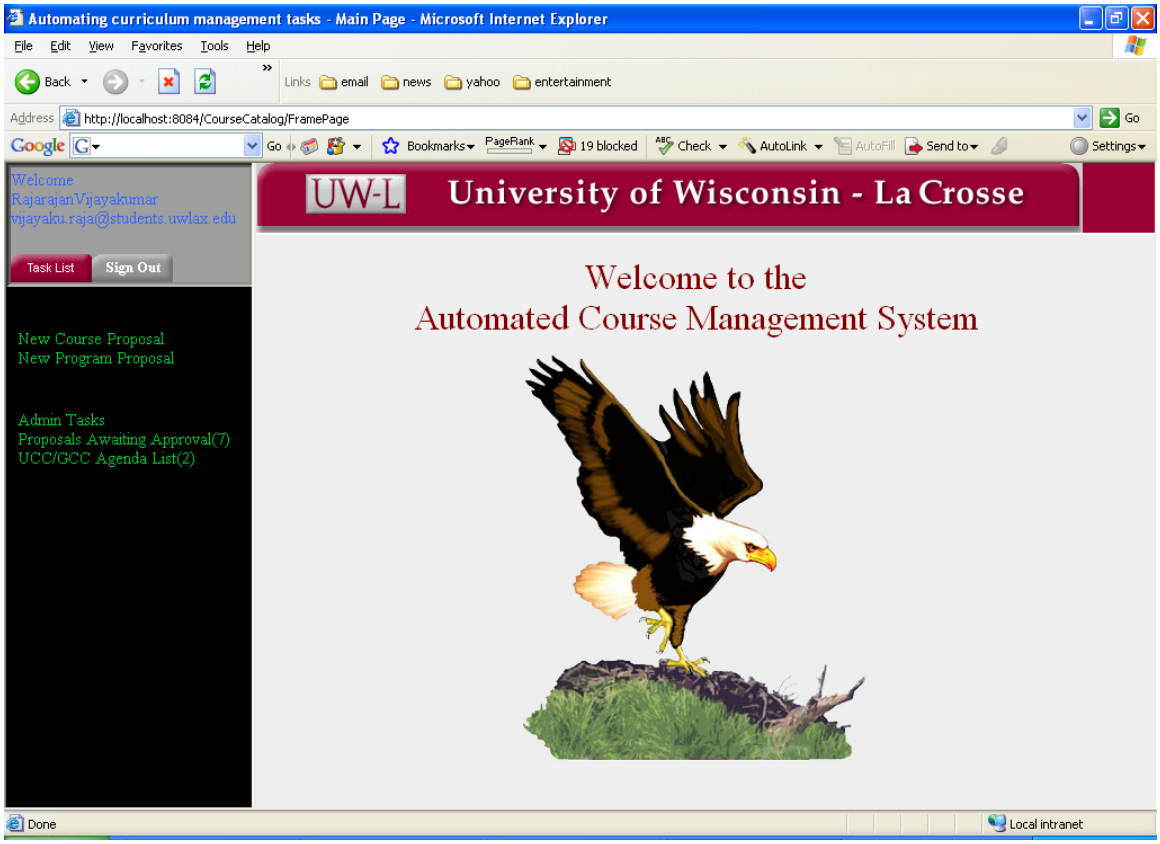
[22] Http://java.sun.com/., Java reference

APPENDIX [A]

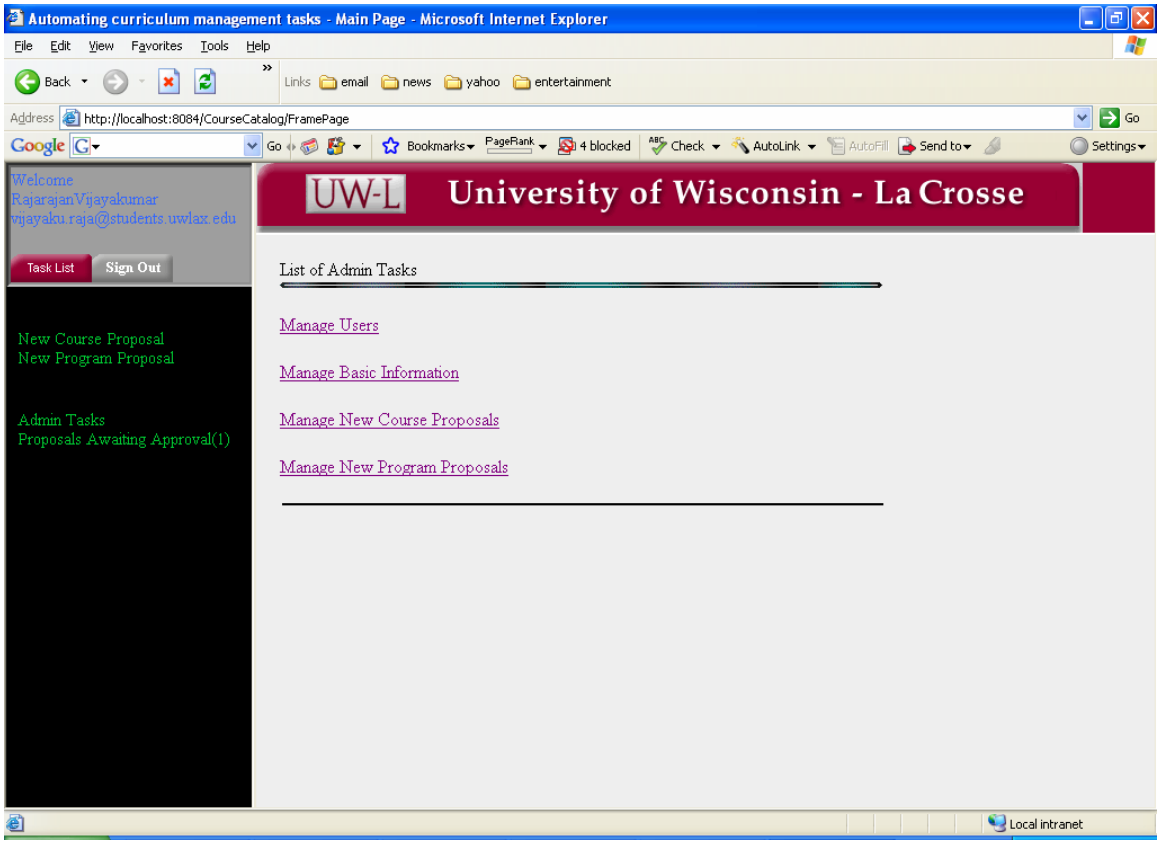
[SELECTED CURICULUM MANAGEMENT AUTOMATION TOOL SCREENSHOTS]



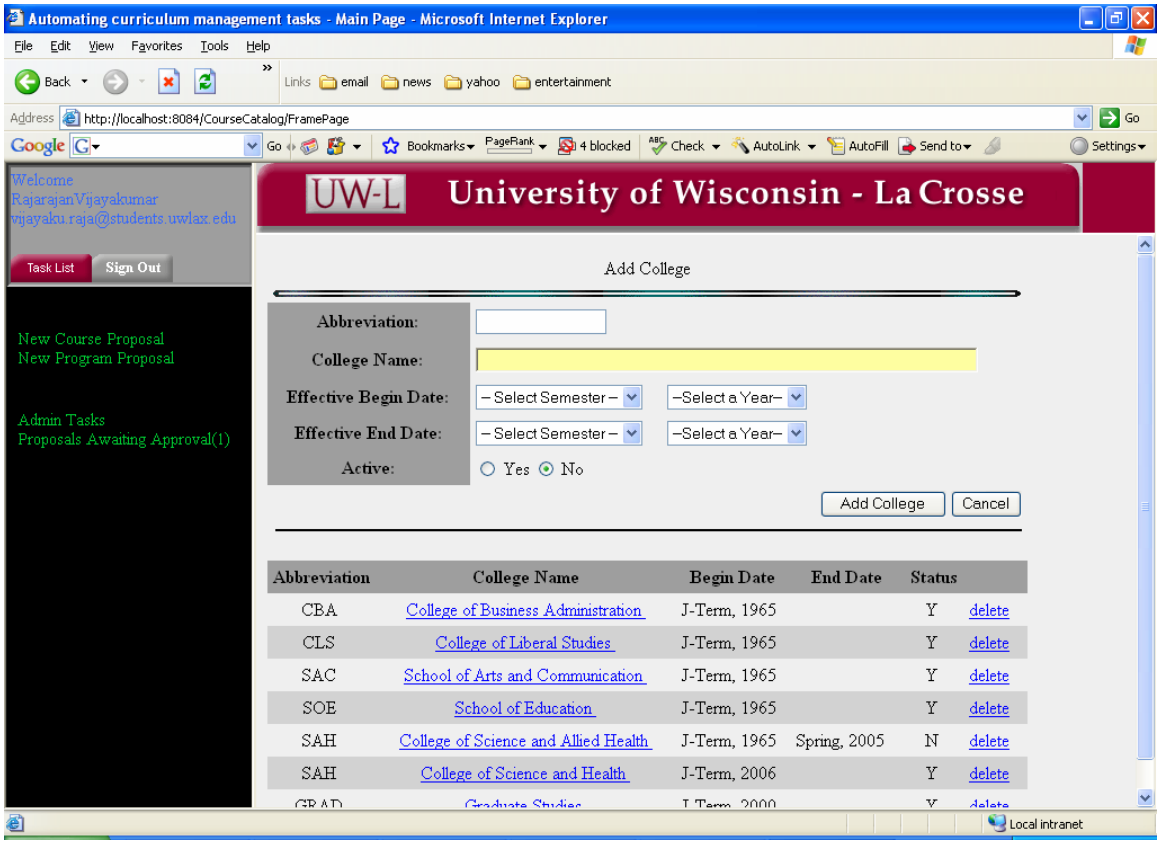
Curriculum Automation tool - Before Login. Login Screen: Enter UWL 8.4 username and password



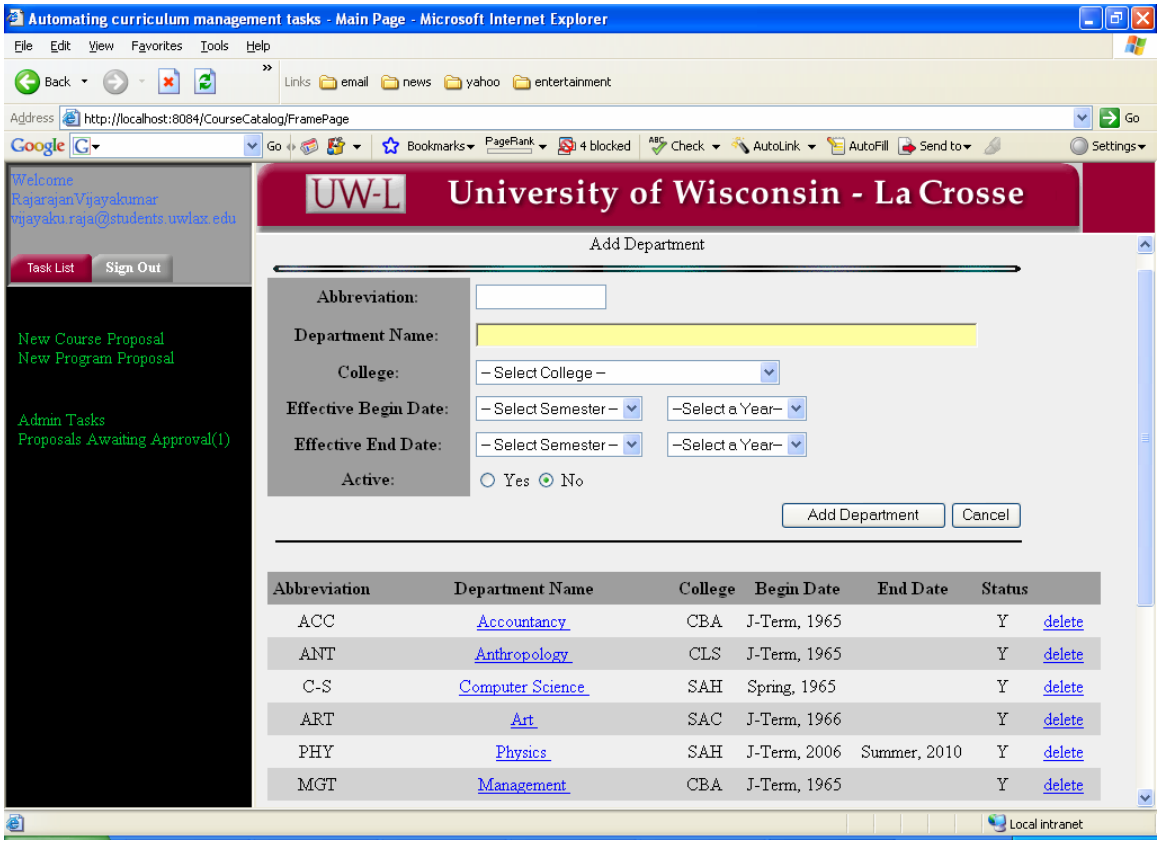
Curriculum Automation tool - After Login


Automating curriculum management tasks - Main Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Links email news yahoo entertainment
Address http://localhost:8084/CourseCatalog/FramePage
Go
Google
Bookmarks PageRank 4 blocked Check AutoLink AutoFill Send to Settings
Welcome
RajarajanVijayakumar
vijayaku.raja@students.uwlax.edu
Task List
Sign Out
New Course Proposal
New Program Proposal
Admin Tasks
Proposals Awaiting Approval(1)
UW-L
University of Wisconsin - La Crosse
List of Admin Tasks
Manage Users
Manage Basic Information
Manage New Course Proposals
Manage New Program Proposals
Local intranet

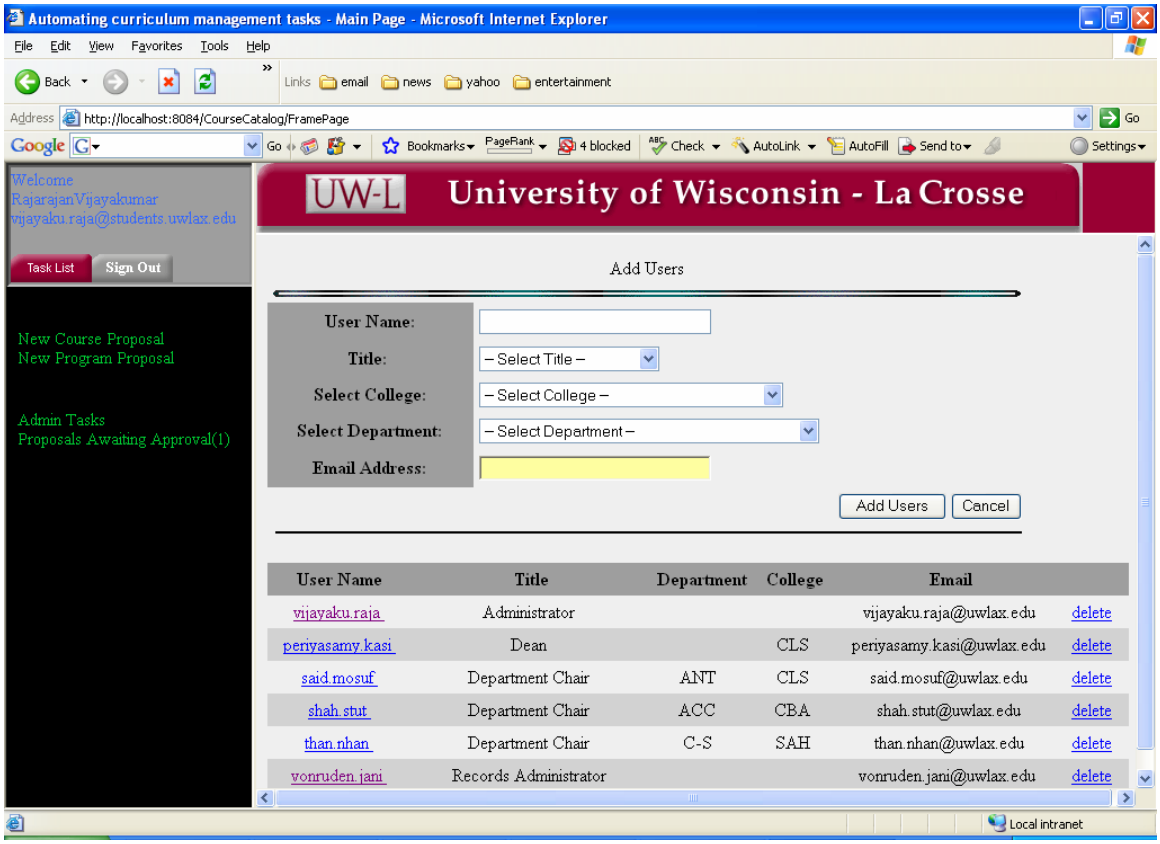
Admin Tasks



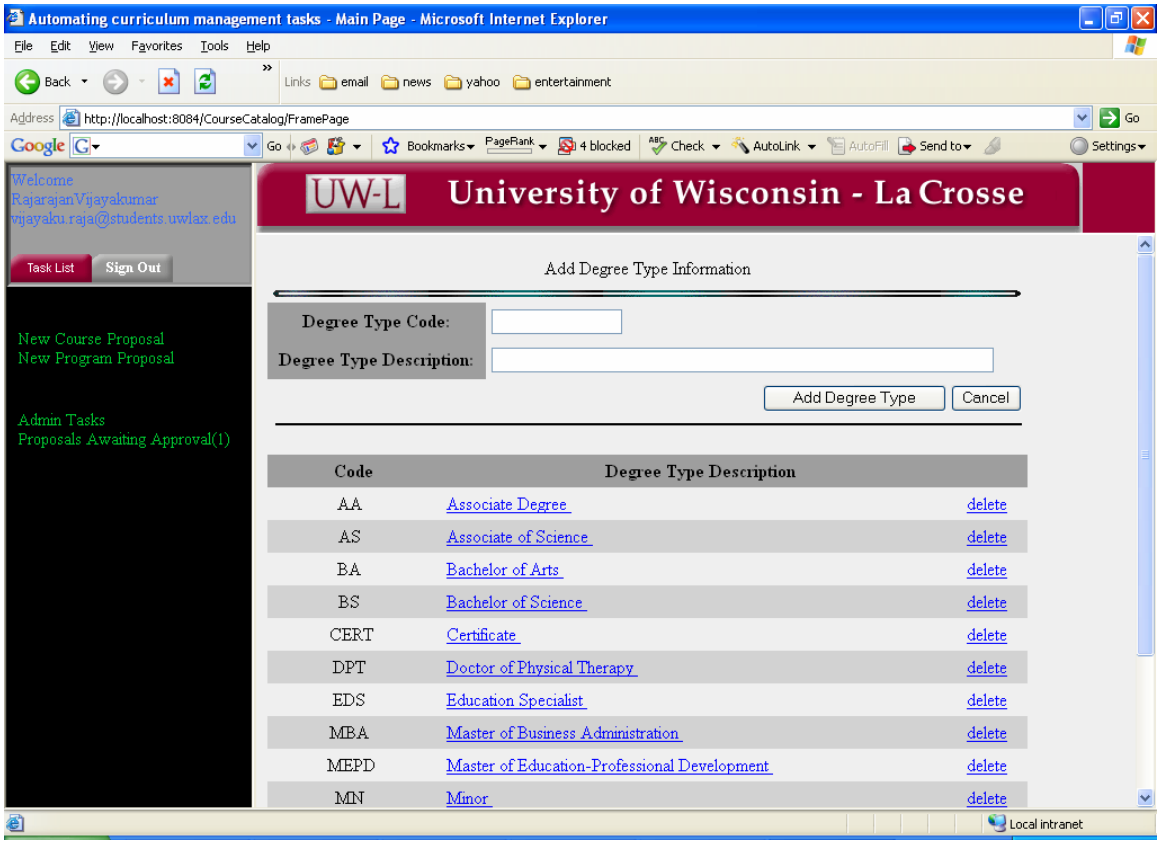
View, Add, Update and Delete college information



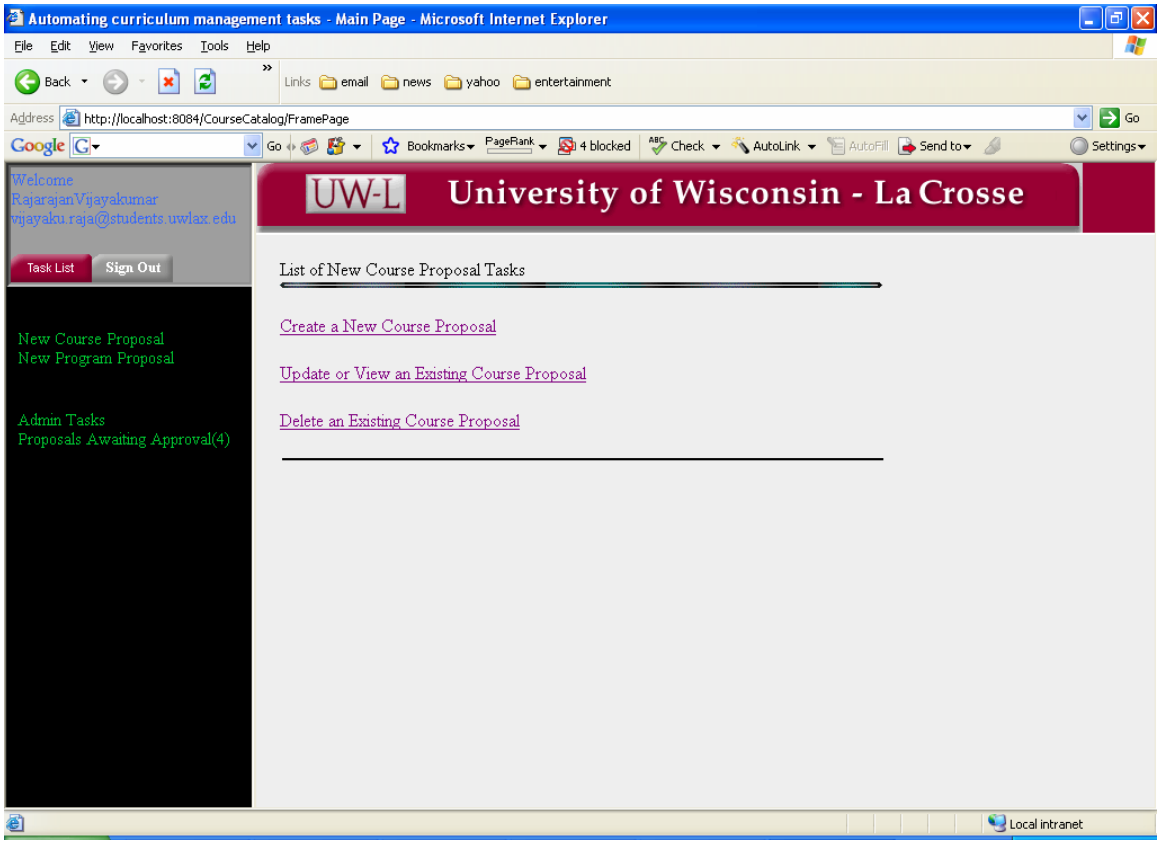
View, Add, Update and Delete department information



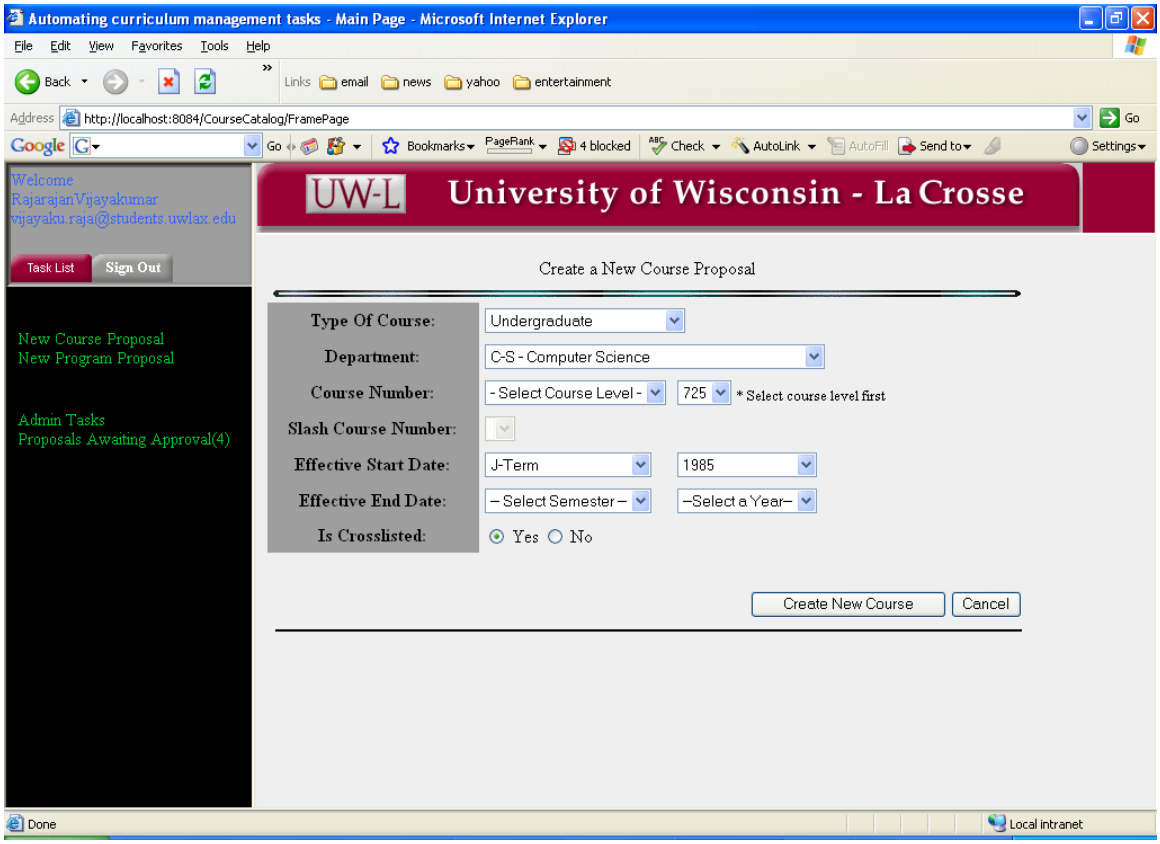
View, Add, Update and Delete user information



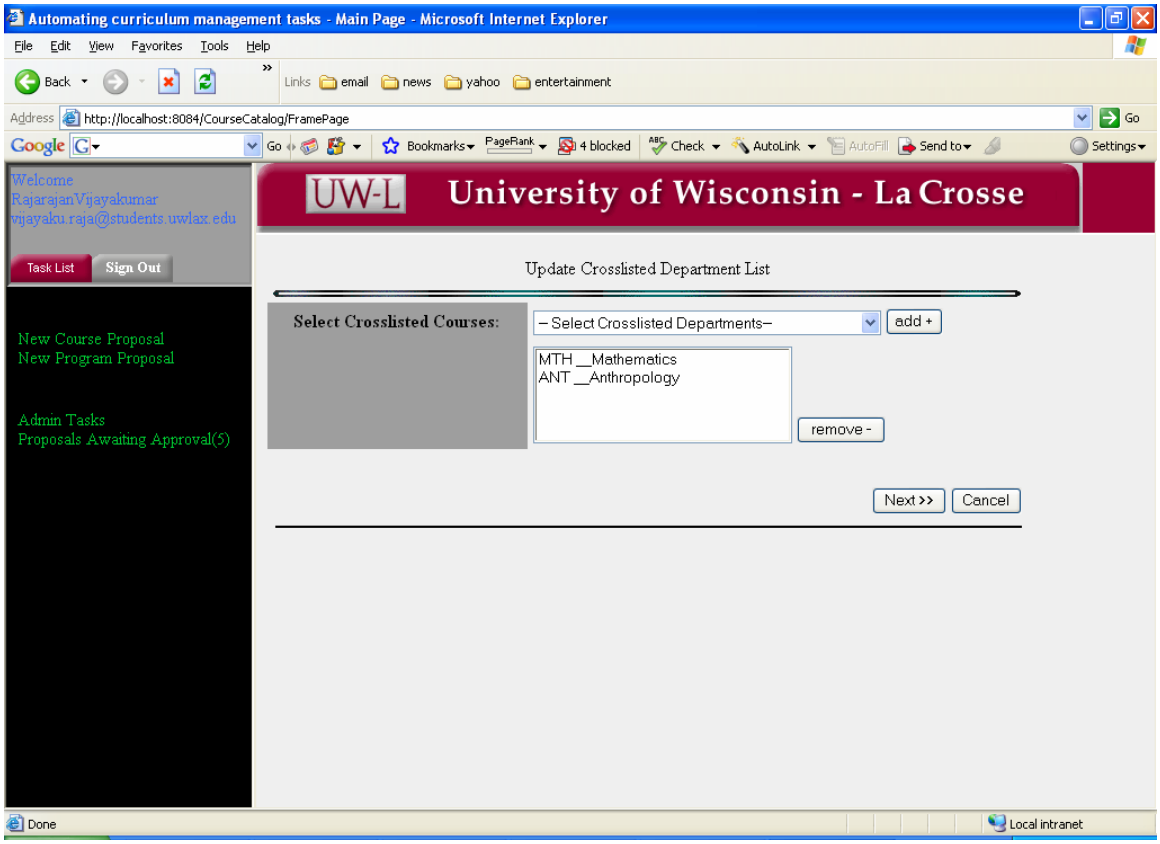
View, Add, Update and Delete degree type information



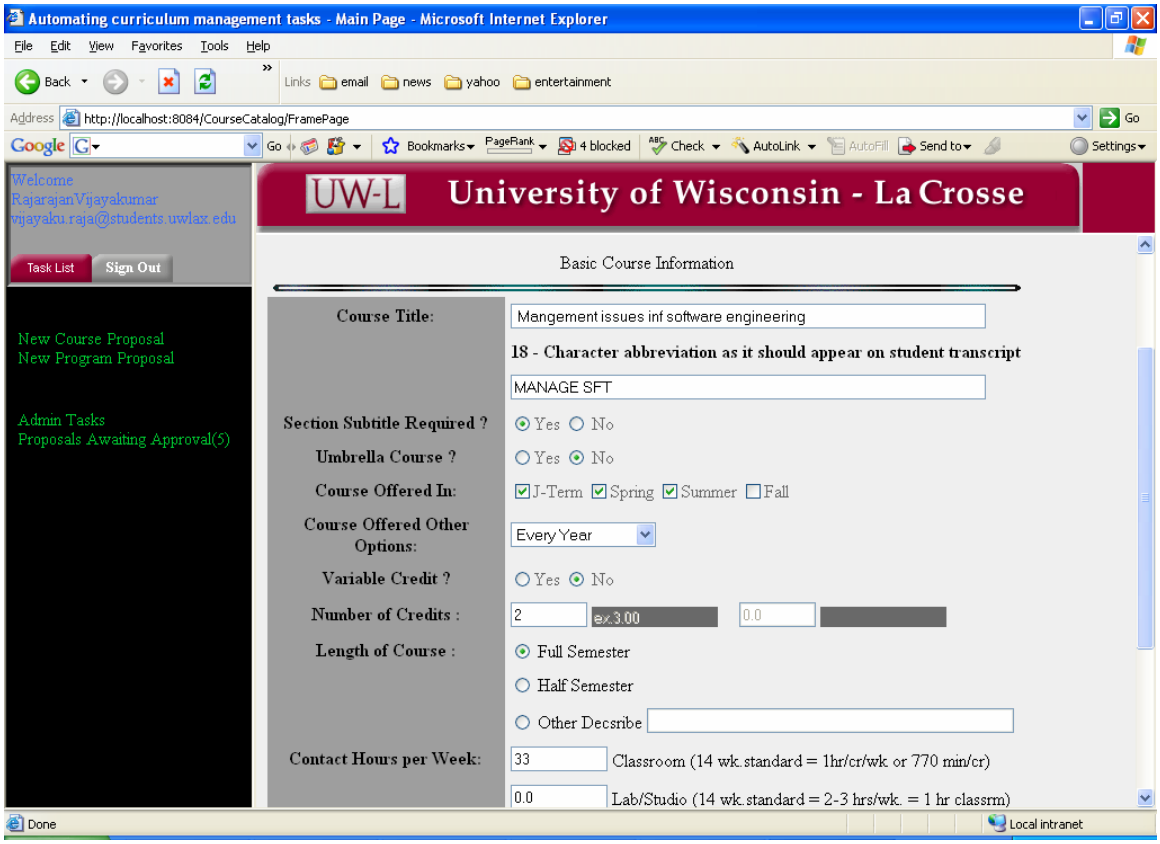
Course proposal options



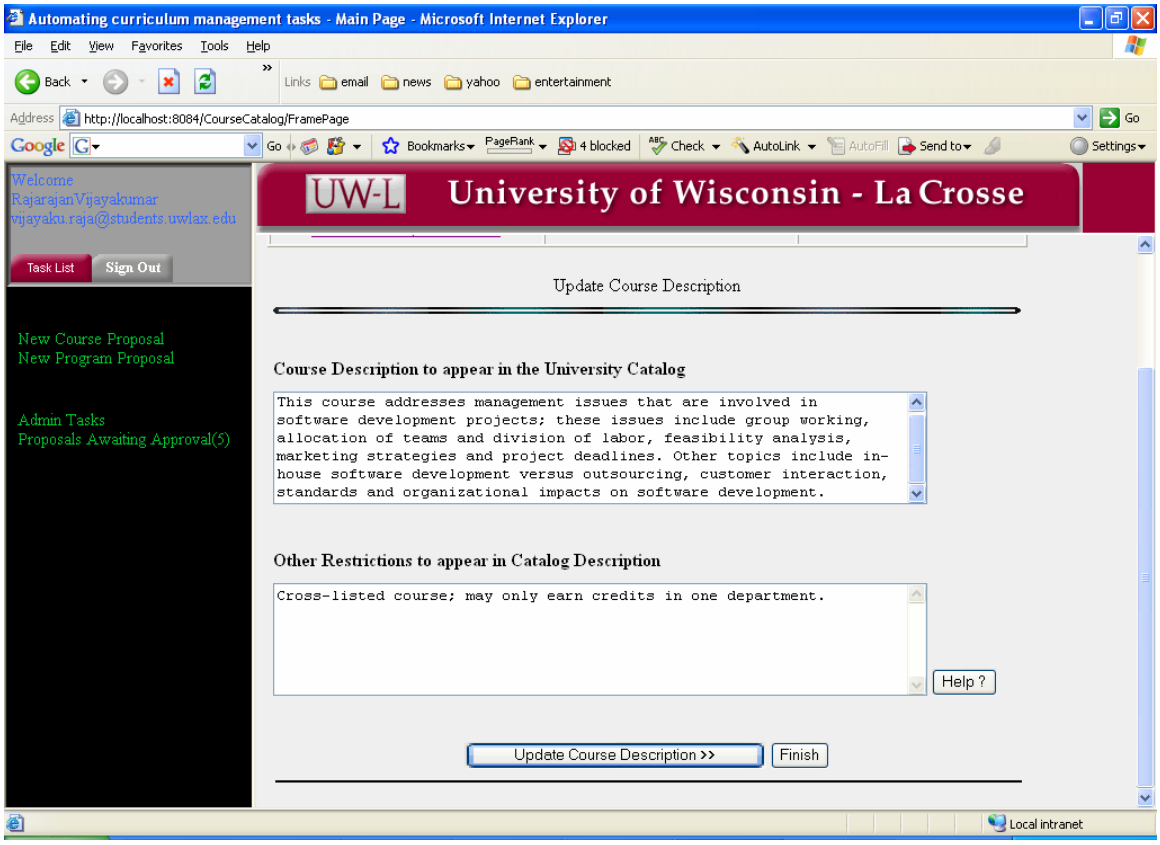
Create a new course



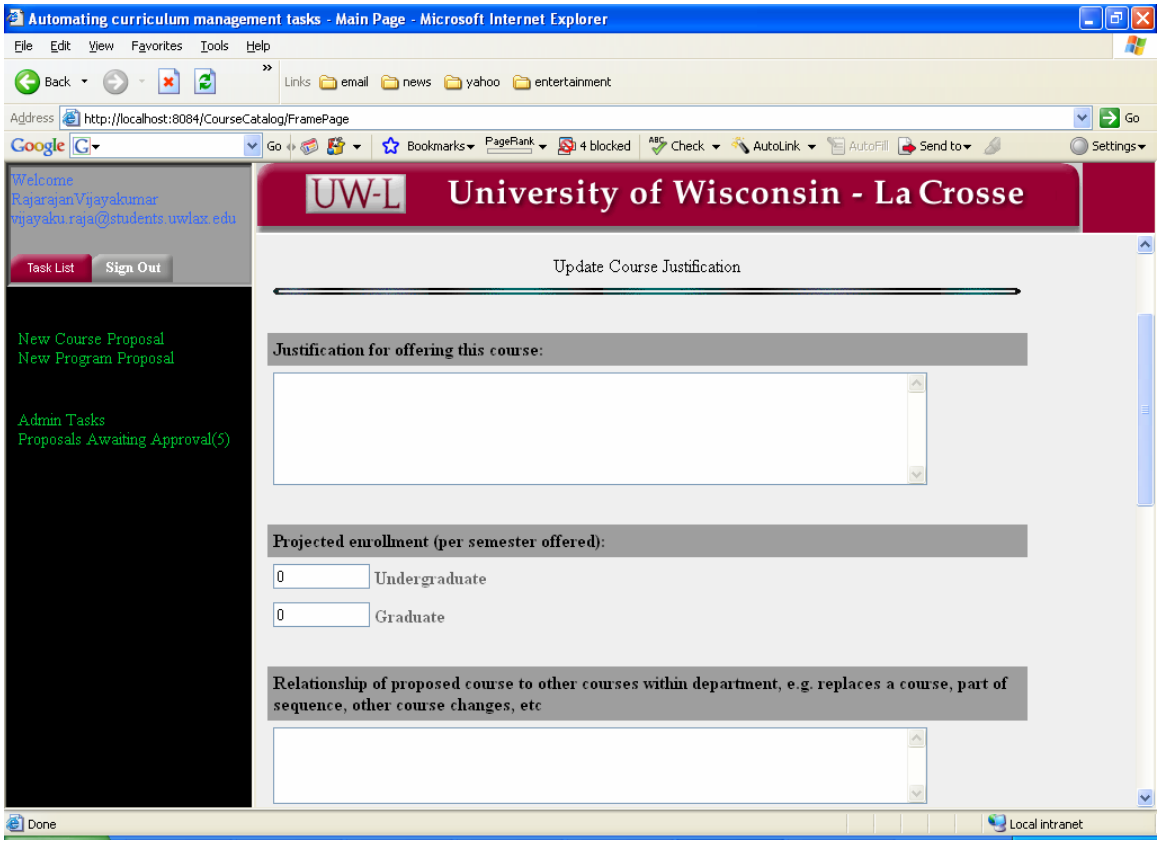
Select crosslisted department list



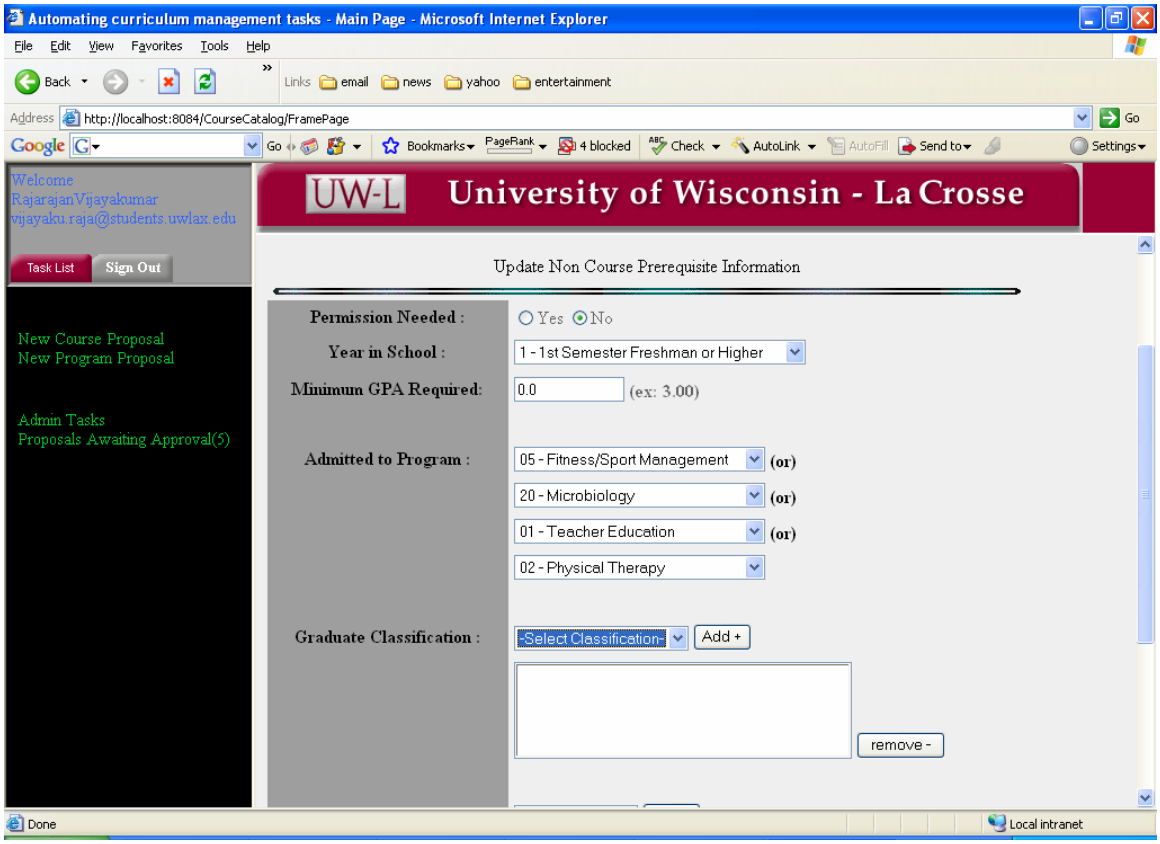
Update basic course information



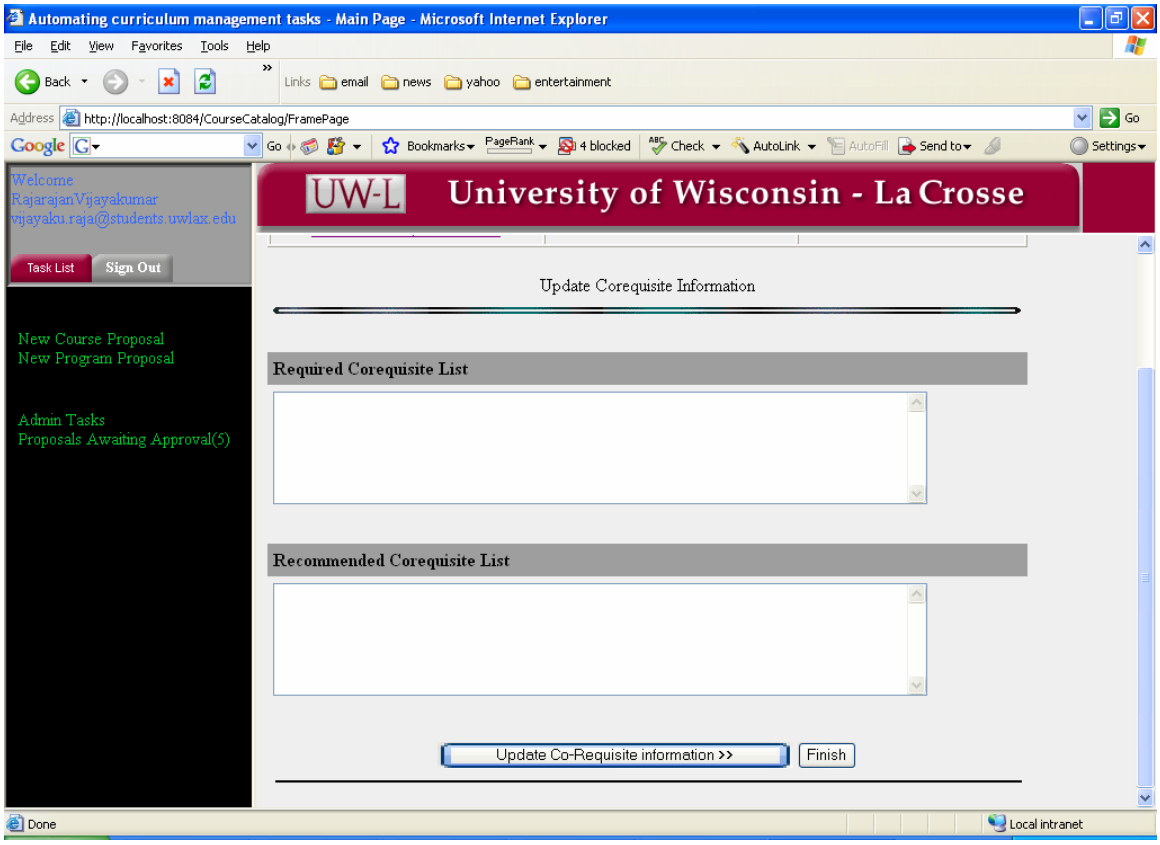
Update course description



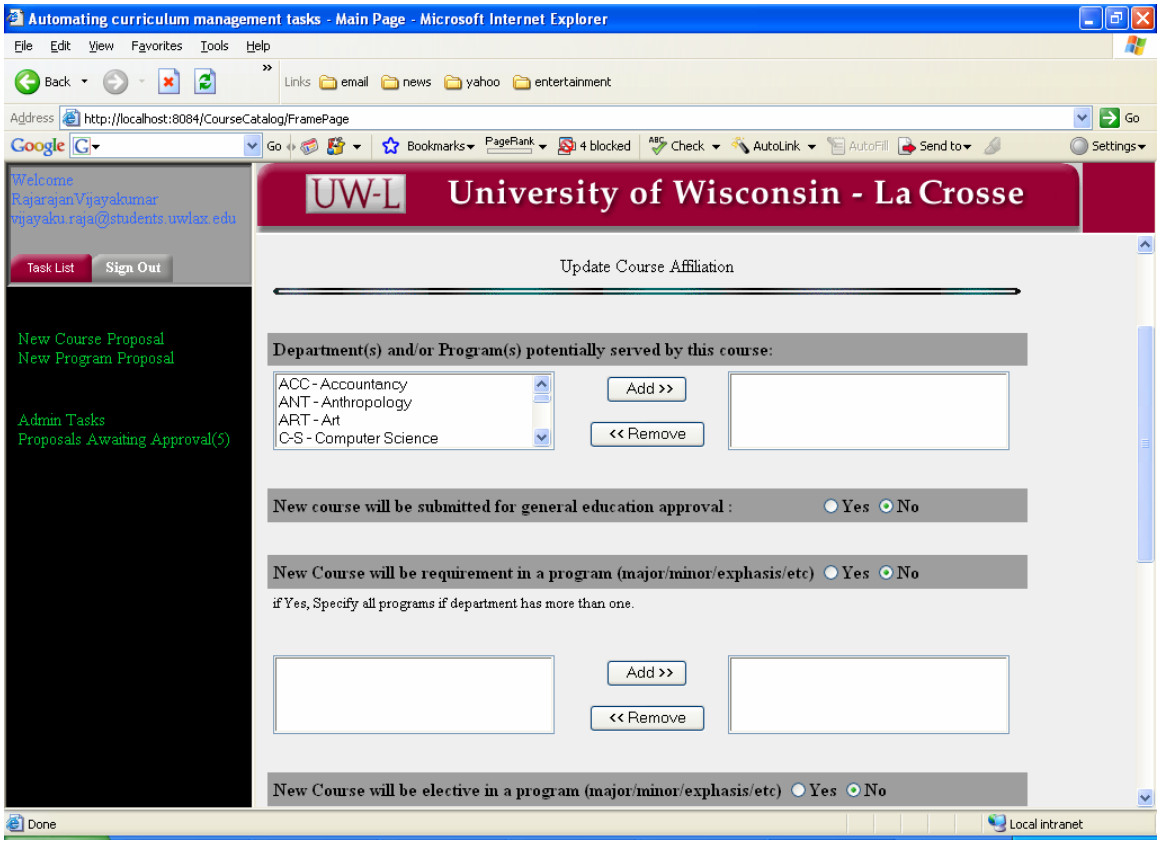
Update course justification



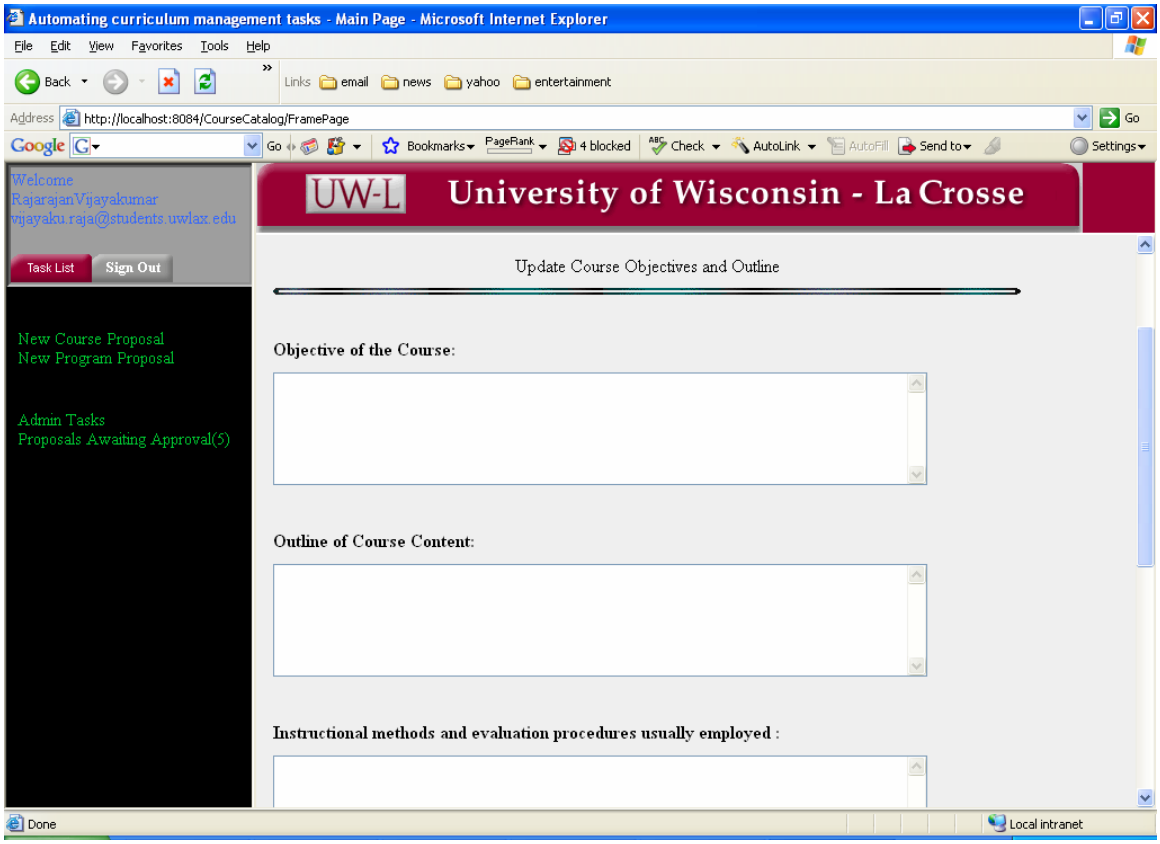
Update non course prerequisite



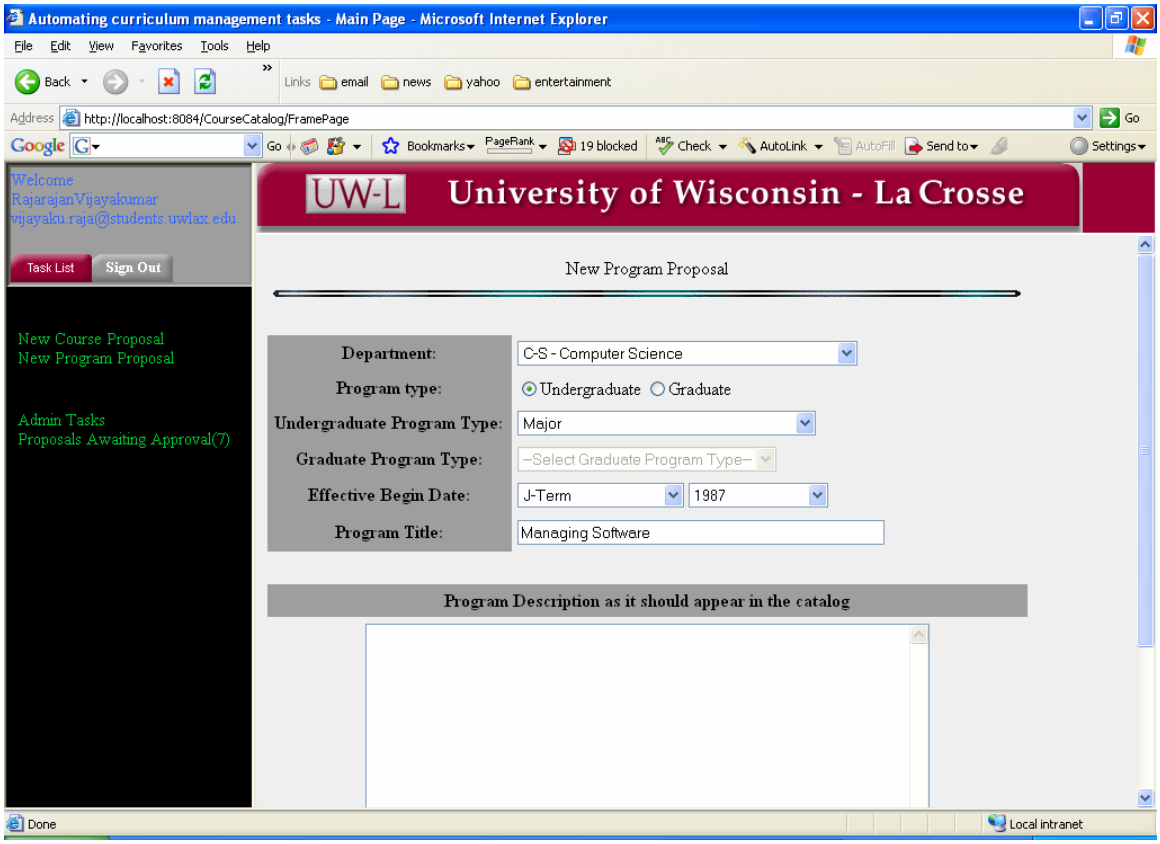
Update co requisite information



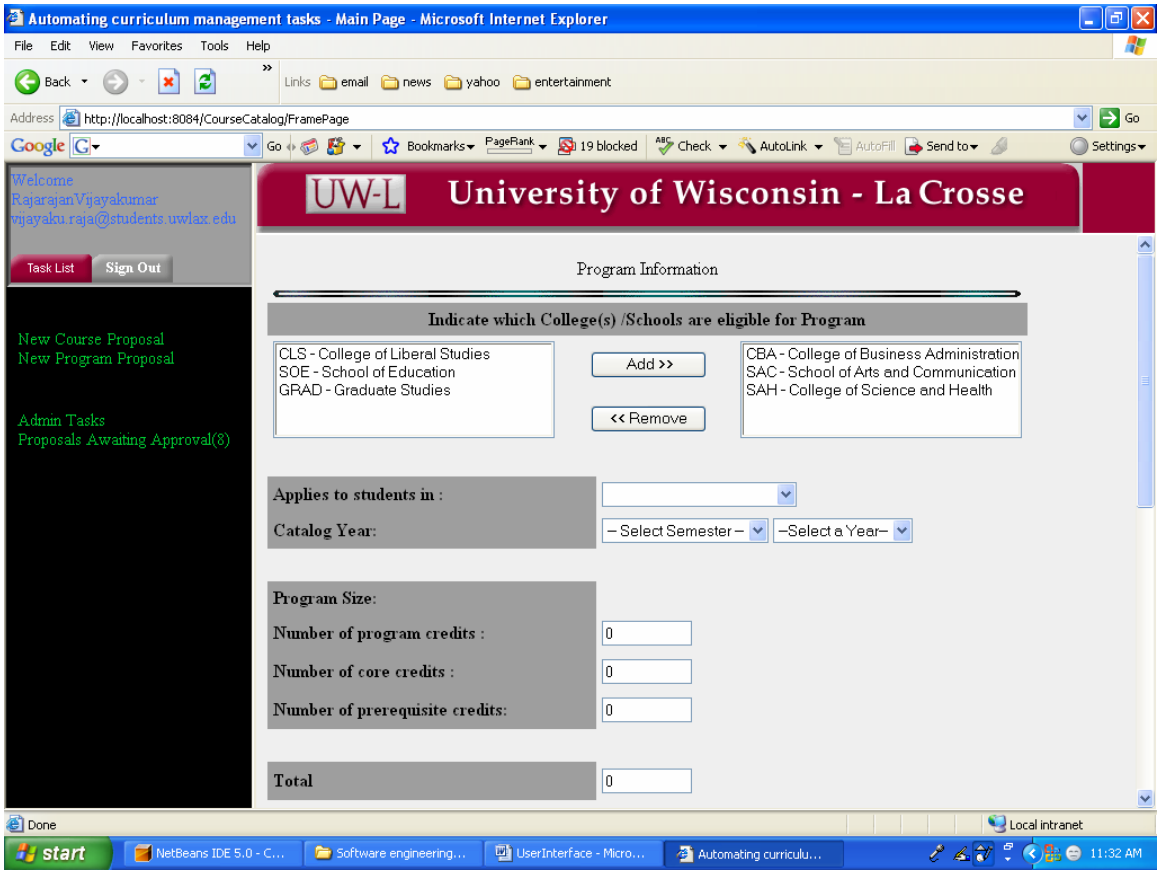
Update course affiliation information



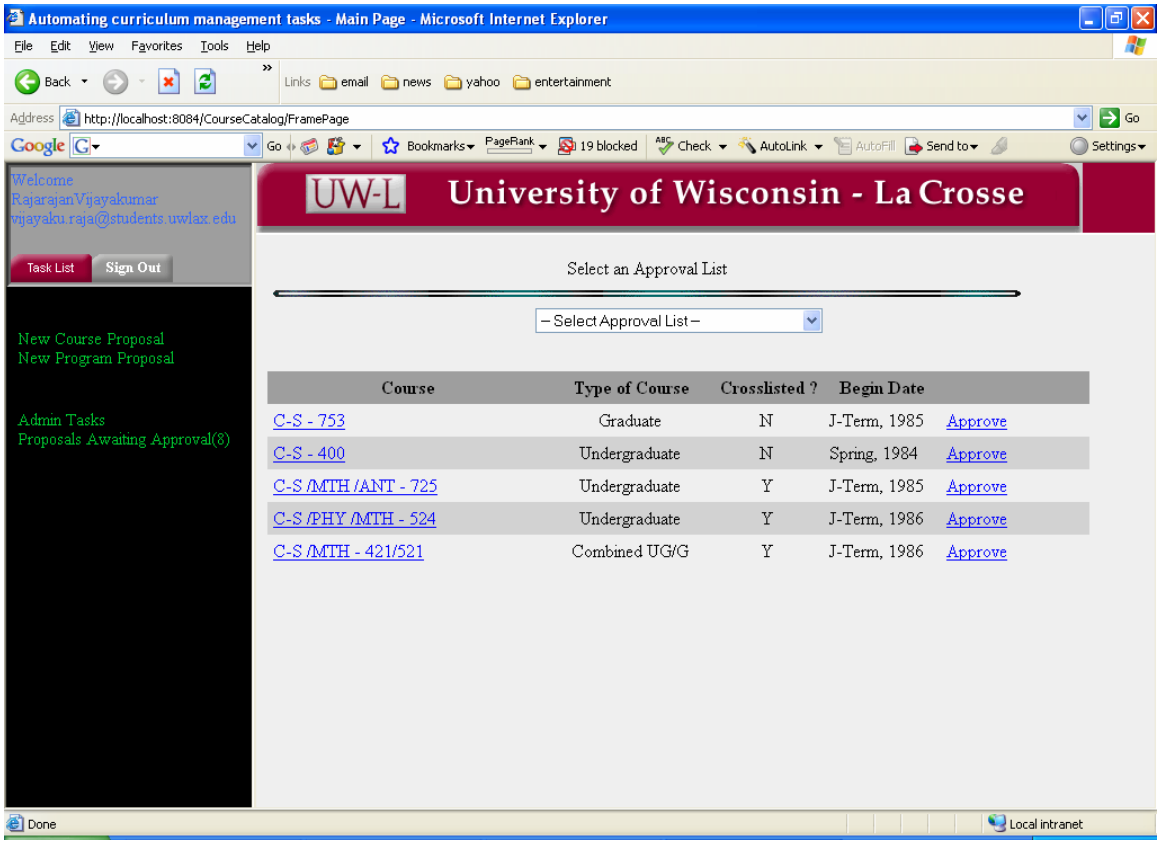
Update course objective and outline information



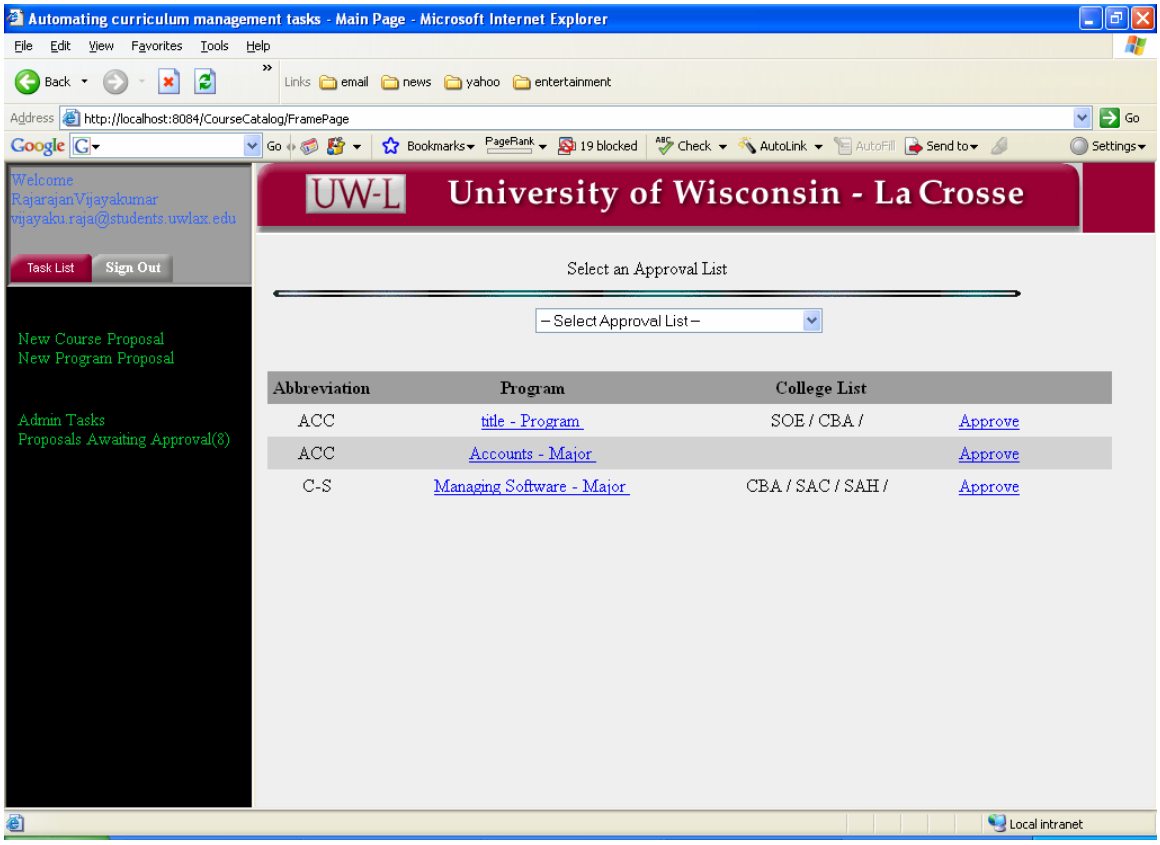
Create a new Program



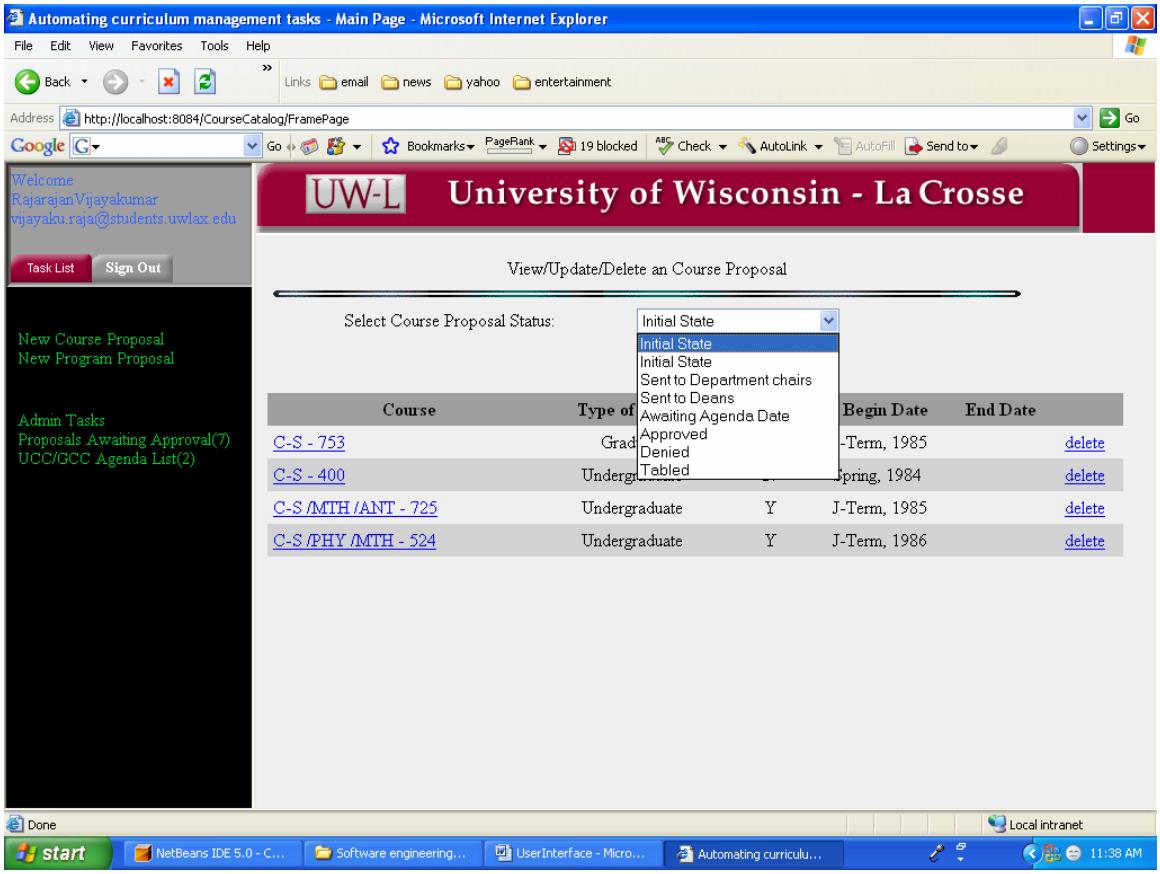
Update eligible college list and other program proposal information



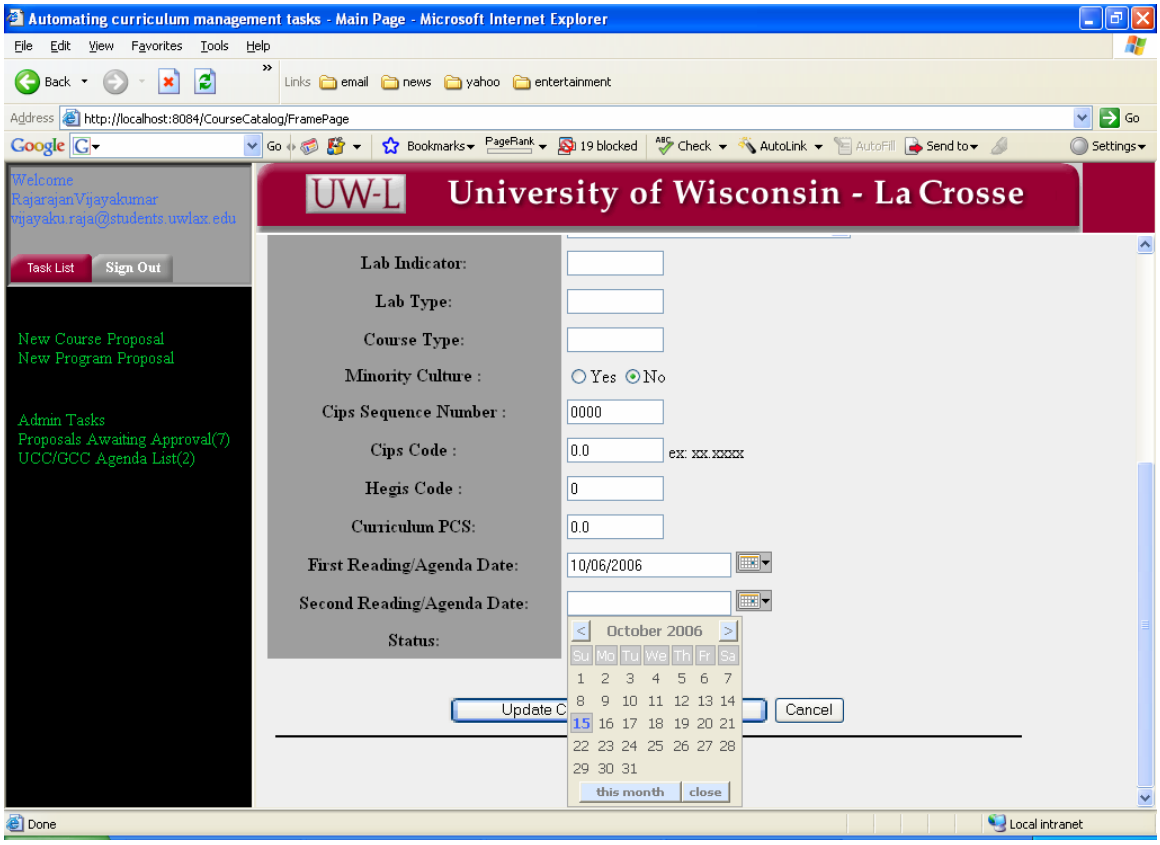
List of Course Proposals awaiting approval



List of program proposals awaiting approval



Select proposals in different state



Select Agenda Date