City University of Seattle

Registration System
Project Assessment

Provided by

Codemutt.com

Kevin McGaughey

8 June 2007

City University CS 600 Applied Research/Capstone Course
Spring 2007 Barbara Endicott-Popovsky

Revision History			
Date	Version	Description	Author
June 3, 2007	0.0.1	Initial In-House Draft started.	K. McGaughey
June 8, 2007	1.0.0	Draft Completed.	K. McGaughey

Table of Contents

I. Introduction

This project was created based upon a statement of work provided by the City University of Seattle to create an online student course registration system. This report will assess the level of completion in accordance with the requirements set forth in the Project Proposal.

The grand overview of the project was to create an online course registration system that was easy to use, looked good, and provided functionality to the user depending on their status. For students, they were to able to access the system, view their information, be able to add, change or delete course requests, and view their scheduled course. For instructors, they were to be able to access the system, view their information and view a course roster once the instructor was scheduled against a course. For administrators, they were to be able to access the system, add, change, or delete classes, courses, instructors, and students, be able to view various reports, and be able to schedule the courses automatically.

The requirements of this system as stated in the Project Proposal, sections V. Scope of Work, VII. Requirements, VIII. Business Rules/Assumptions and IX. Assessment Criteria, and approved by the customer will be reviewed in this report. In addition, this report will cover how well those requirements were satisfied, how well the planning and execution met internal timelines, and a brief look into possible improvements to the current system and possible enhancements for future use.

II. Managerial & Technical Lessons Learned

Managerial Lessons Learned

This project was designed to be completed within a ten week time frame. Due to the compressed time frame, the scope of the project was limited to meet the desired deliverables dates. This required planning to be minimized to reduce complexity of the system in order to meet the requirements of the Statement of Work. The lessons that can be learned from this project are:

- The upfront planning time needs to be emphasized. There are several cases within this project where items from the Project Proposal and the Technical Specification were modified to be more useful than originally planned.
- The scheduling of the different phases of the project did not allow for extra flexibility in delaying a deliverable. In order to meet the fixed deadlines, planning of critical phases of the project were minimized. This allowed the documentation to not completely reflect the actual product in some cases and required more design work during coding.
- I don’t like doing paper work. Can I just be a coder?

Technical Lessons Learned

From the beginning, the plan was to use MySQL and PHP for this project as they were fresh in my mind from the last quarter. The database itself was unremarkable due to the stipulation to develop my own constraints. I kept the requirements simple to keep the system scope under control and reachable. The automated scheduling function was the hardest part, but mainly from my lack of recent experience with SQL. As I proceeded into the second attempt at the scheduler, the SQL I used was more complex, but still nothing extreme. The automated function could have been significantly streamlined with more complex SQL statements. The basic logic structure of the scheduler wasn't challenging, just the inclusion of the SQL to retrieve the data.

The PHP coding did not pose too many challenges overall. When a problem came up during coding involving include files, instead of troubleshooting extensively, I decided to dump them in favor of full code files to prevent wasted time.

More time was wasted during design and requirements collection due to experimentation with different database manager utilities. None of the tested products were able to connect to the database even with significant attempts and scouring the web for solutions. The end result was the use of MyPHPAdmin through a web interface.

III. Success Criteria Comparison

Scope of Work

Requirement: A web interface that is user friendly, accessible, and aesthetic.

Measurement: The website is currently running 24/7 depending on server reliability, currently 99%. The site is accessible to all registered students, instructors and the administrator. As for aesthetics, the site uses the City University of Seattle public website graphics, cascading style sheets, javascript, color scheme, and navigation scheme. Since it is inferred that the City University of Seattle's public website is aesthetic pleasing, then I have met that requirement by emulating their site.

§

Requirement: Allows administrators to add/change/delete the classes for a term, add/change/delete instructors and their information for the term, add/change/delete classrooms for the term.

Measurement: These functions are fully implemented within the administrator module of the Registration System. Included in the administrator module is the ability to add/change/delete students, instructors, courses, and classrooms, plus the automated scheduling function and the resulting class rosters.

§

Requirement: Allows students to view courses being offered for a term, add/change/drop a course for the term.

Measurement: The student module of the system allows students to view their information, the courses available for the term, the courses they request for the term, and finally the class that they get scheduled for after the automated scheduling is complete.

§

Requirement: Allows instructors access to class rosters

Measurement: The instructor module of the system shows instructors the course that they are capable of instructing, and after the scheduling is done, the course they are scheduled to instruct and a link to the class roster for that course.

Requirements

Requirements: Student functions must offer the ability to securely log onto the website, view an available course listing, add, change or drop a course, view a course schedule.

Measurement: All users must log onto the website in order to view their respective modules. The student module has a link for viewing available courses, a link to a page to add, change and delete course requests, and finally after the schedule is complete, the ability to see what course, instructor and class they are scheduled for.

§

Requirements: Instructor functions must offer the ability to securely log onto the website and view a course roster.

Measurement: All users must log onto the website in order to view their respective modules. The module has a link to the class roster, which is populated after the scheduling is complete. The instructor module shows the instructor a course ability list and when scheduling is done, their course information. (Above Scope)

§

Requirement: Administrator functions functions must offer the ability to securely log onto the website, Add, change or delete a course, add, change or delete an instructor, add, change, or delete a student, view an available course listing, view a course to classroom listing, view a course to instructor listing, view a course schedule.

Measurement: All users must log onto the website in order to view their respective modules. The administrator module has links to additional pages, one for each section, students, instructors, classes, courses, each of which contains the ability to view a list of that item, the ability to add more items, change an item, or delete an item. There is also the ability to run the automated scheduling function and display class rosters for all the courses scheduled. (Above scope)

Business Rules/Assumptions

Requirement: Access to this site will be via the internet

Measurement: The website is accessible at cs600.codemutt.com

§

Requirement: Access will be available around the clock except for scheduled maintenance

Measurement: The website is located on a leased server and available 24/7 with a current reliability of 99%.

§

Requirement: Access will be limited by password controlled functionality

Measurement: Each user has their own login and password which defines their functionality. Each code file has code to prevent unauthorized access by a different type of user.

§

Requirement: The administrator will select the courses for the term

Measurement: The only user with the ability to add courses is the administrator.

§

Requirement: Courses will only be offered once per term

Measurement: The system only supports one term, and the database uses the class id as a primary key with no duplicates, thus preventing the addition of an additional course.

§

Requirement: Not all courses will be offered each term

Measurement: The addition of courses is accomplished by the administrator. There isn't a master list of courses in the system.

§

Requirement: Each classroom will host only one class per term

Measurement: The system uses the classroom id as a primary key with no duplicates, thus preventing the addition of the same classroom.

§

Requirement: Each instructor will teach only one course per term

Measurement: The automated scheduler queries the catalog of classes scheduled against instructors to prevent one instructor being assigned to two courses.

§

Requirement: Each student can only take one course per term

Measurement: The automated scheduler queries the enrollment table of scheduled students to prevent one student from being scheduled for two courses.

§

Requirement: Each course will have an instructor and classroom assigned before students can be assigned

Measurement: The automated scheduler creates a catalog of scheduled courses as its first step that combines courses, classes and instructors.

§

Requirement: Students who are seniors or in the major concentration will have priority for degree required courses

Measurement: Priority is given to seniors in the major track, then seniors, then juniors in the major track, and then everyone else by the automated scheduler.

§

Requirement: Only pre-registered students and instructors will be allowed to access the system

Measurement: The system requires users to login and the usernames and passwords are created by the administrator during the addition of students and instructors.

§

Requirement: Various reports will be made available; class rosters, course lists, classroom lists …

Measurement: Each module provides different reports. The student module shows a list of available courses, then a class list of scheduled classes for the student after the scheduling is complete. The instructor module shows a class roster after the scheduling is complete. The administrator module has lists of classrooms, courses, instructors, students and then a course list and class rosters once the scheduling is complete.

Assessment Criteria

Requirement: The interface should maintain a consistent feel and look

Measurement: The site uses the look and feel of the City University of Seattle public website for every page that is shown to the users based upon the stylesheets and formatting of the website. The navigation menu changes based on the module the user log in but do not change the look or feel of the site.

§

Requirement: The interface should be easy to navigate

Measurement: The site uses the navigation system used on the City University of Seattle public website and thus inferred to be easy to navigate.

§

The proposal for this project did not include any requirements for the size of the data pool or the automated scheduler. The data pool currently consists of 28 instructors, 28 classrooms, 29 courses, and over 200 students. The tables can hold over 10,000 entries without anticipating any considerable increase in query times. The current version of the scheduler accomplished over 620 database queries in approximately one second. The automated scheduler was developed to save the administrator the time required to determine which instructor matched which course and the subsequent assignments of students. While not mentioned in the proposal, the scheduler is considered vital to the proper operation of the system and not considered above the scope.

IV. Client Roles

The clients of the Registration System are the developers, the school administrators, and the end users, students, instructors and system administrators.

The developer’s stake in this system is to continue to expand the capabilities of the system without changing the look and feel or functionality currently implemented.

The school administrator’s responsibility to this system is to fully support it, advertise it, and integrate the system into the current servers and databases that the school maintains.

The student’s role is to use this system to request their courses for the term, make appropriate changes to their requests (add, change or drop) and see a schedule of their courses.

The instructor’s abilities in this system are limited to the ability to see what they have been scheduled for and a class roster.

The administrator’s responsibility is to make the appropriate changes to the system to reflect the correct information in each section, students, instructors, classes, and courses so that the other users have accurae information to make informed decisions regarding courses.

V. Future Enhancements

This system lacks a lot of potential functionality based on the limited time frame to create and implement this project. To fully implement such a system, the databases of the university need to be integrated into the system to access course lists, class lists, student information, instructor information and other useful data. Some potential enhancements that would increase the functionality of this system are:

Add the ability to schedule multiple courses per term for students and instructors and multiple terms.

Add the ability to track student courses completed as well as courses remaining in a degree program.

Add better security to the system with secure passwords.

Add email notification of scheduled courses for students and class rosters for instructors.

Add better database query error capturing. Unless the current tables crash, the queries will work correctly.

Add better error trapping during data input. The basic error trapping used does not take into account unusual entries or special symbols.

Consider implementing ASP or AJAX coding to make the number of pages a lot smaller.

VI. Conclusion

This system required some very rusty skills. This program took nine quarters, with one off for intensive work training. Some of the skills used were learned a couple of decades earlier and revisited during this program. This capstone project was a good visit in a potentially typical website or application development project. While I enjoyed the coding, I have never been fond of the paperwork involved. I did not mind working solo, but I think the course would have been more beneficial as a team effort. With a team effort, the scope of the project could have been enlarged to provide more functionality.

The End!