Honors Research Project Database

Introduction

As I began the daunting task of selecting an honors project to take on, I wanted something that would build on skills I would use in the “real world”. In considering my options, I found that web programming seemed to be the way things were going, and decided to go that route. I started searching through the countless binders of past projects desperately looking for direction, when Dr. Mugler stepped in. Within fifteen minutes, he had expressed the need for an online database of honors projects. I had found my calling.

This project was to combine many useful aspects of programming, including database access, web programming techniques, and the whole software engineering experience from start to finish. I had been working with JSP and databases in my database programming class, and it seemed powerful enough to get me going. Now that I had the basics down, I was ready to begin.

Planning and Design

The first thing I worked on was an overall plan for the project. Before I even typed a line of code, I sat down for a couple of “brainstorming” sessions and drew up a rough design. I drew a layout for each page I could think of at the time, and sorted out what database values / tables I would need. Project management, including the ability to add, edit, and delete projects, would be important. There would also need to be a way to view projects and search / sort by different parameters. Some sort of access control would also be required for the site. Getting everything sorted out on paper saved me a lot of time with coding later on.

Another part of the planning stage included researching different technologies to use. I had already decided upon JSP. Since JSP can work with many different database management systems, I went with the free MySQL system. I figured it would be an easy change to swap to a different system if the need arose. I spent a week or so configuring these services on my fiance’s old computer and set up an account on logmein.com to access the machine remotely.

Implementation

I started implementing the site by creating the user accounts functionality. I added a 'users' table that held a username, password, and admin bit. Admin accounts would have special abilities, such as adding, editing, and deleting other accounts and managing projects. In this layout, the honors
college would have an admin account, and have complete control of the content, and who accessed it. Rather than have them add a separate log in for each person, I figured they could create an account for the semester (Ex. Fall10) that would be replaced every semester with a new password. Log ins would be handled through sessions, so the user would be logged out when closing the browser, further preventing unauthorized access.

When a user would try to access a page, and a session had not yet been created, the user would be redirected to a log in page. The user is then prompted to enter credentials, which are checked against the database. If they are legitimate, a session attribute is set and the user is redirected to the page they were trying to access. Each page that has separate functionality for a regular user and an admin would check if the user is an admin or not and show or hide additional admin functions accordingly.

When the user wishes to log out, there is a log out link (which simply invalidates the session) or the user can simply close the browser (which also invalidates the session).

Account management was the next hurdle, so I created an admin only page to add and delete accounts. It displayed a list of information on accounts, including the username, when it was created, and who created it. The list also contained a ‘delete’ button to easily remove accounts. At the bottom of the list, there was an ‘add account’ button that would start the account creation process. The first step was a simple web form to enter the desired account username and password. From there, the administrator could either ‘cancel’ or continue by clicking ‘add’. The next page displayed the information just entered and prompted for verification. The user could then ‘Add’ or ‘Edit’ the information, or ‘cancel’ to return to the main accounts page without adding. This process became the standard for later adding projects and students to the database.
Next on my list was setting up the projects database and pages. I decided to create a separate “semesters” table since it is the one field that would be duplicated the most if left in the projects table (many projects completed per semester). I finished the projects table, and then began on the pages. I created a page for adding projects, using the add accounts functionality as a jumping off point, and the ability to search projects by many different attributes.

Searching would be handled with a drop down for what to search by, and a text box for the value to search for. As long as the ‘search for’ box wasn’t empty, a search would be done. This was then translated into what would be the ‘WHERE’ clause in the query.

*Ex: ‘Search By Major’ is selected and a value of ‘Computer Science’ is entered.*

\[
\text{filter} = \text{“WHERE Major LIKE ‘%Computer Science%’”};
\]

Another feature that would be necessary as the database grew in size was paging. Although my test database only had a few projects in it, as the number grew into the hundreds and eventually thousands, these would ideally not all be on one page. This was a little tricky at first, but turned out to be pretty trivial to implement. The basic query format I used was:

\[
\text{SELECT * FROM Projects LIMIT 5, 10;}
\]

This basically grabs 10 rows from the ‘Projects’ table, but skip the first 5. The numbers would be calculated based on a page number variable, and ‘next’ and ‘previous’ displayed if needed. Pages would be 0 based, so page 0 would not skip any database rows. So, after adding in the filter string, the query became:
After a user finds the project they are looking for, they would want to view details about that project. This created the need for a “details” page. The page showed the title, student, major, and completed semester of the project. I would also need a link to the actual project file for users to view.

The next step was figuring out how to store the project files. I found that the best way to handle the actual project files would be to treat them as attachments. Since a project may have multiple files, just having functionality for one file was not acceptable. The solution I settled on was to have a folder with the same name as the project id that would store all files for that project. The files would be viewed from the details page and uploaded by admins there as well. I took some time to
research uploading a file to a server, which proved to be relatively simple, and worked it into the site.

Logged in as admin, logout

Upload attachment for project:
Browse... Upload Cancel

Upload page

The result was a pretty sleek looking details page with plenty of flexibility for project files. I tested out a lot of the functions of the site, cleaned up bugs, and prepared for a meeting with the honors college to see if anymore functionality was required.

Overall, the meeting with Honors College went well. I connected to my home computer through the logmein service that I had set up when starting my project. From there, I was able to demo the current site, and ask for input. I left the meeting with a good feeling, and just a little functionality to add, that being the ability for more than one student to be associated with a project.

This latest requirement would require breaking the student information out into a “students” table and adding a “studentprojects” table to link the students and projects tables. When initially creating the project, one student would be associated with it. Afterwards, admins could add more students to the project from the details page. To accomplish this, I added a new ‘Add Student’ page and an ‘Add Student’ button below the listed students.

Closing

At the company where I am interning I have been doing a lot of web programming lately. In working closely with the customers, the company recognized the demand for a web based system. Coming from a company whose main product is a Windows application, moving towards a web system says a lot. Because of the mobility of the web, and ease of deployment, web based systems, such as my project, are becoming far more common.

Taking this project from inception to completion has proved to be an invaluable experience for me. Gaining experience in all stages of the software development cycle has increased my confidence as a programmer and, I believe, made me more employable. Gaining initial requirements, designing, coding, demoing, and maintaining the system has given me skills that no one class could encompass.
Appendix

Database Schema

**Projects** (*id, title, semesterID)

**StudentProjects** (*studentID, *projectID)

**Students** (*id, fname, lname, major)

**Semester** (*id, season, year)

*denotes key