CCT395, Fall 2010, Topics in CCIT: Databases

The Final Project

Proposal due on November 3, 2010, at the beginning of class.
Final report due on November 24, 2010, at the beginning of class.

1. The Basics

In this final homework assignment you will design and implement an information system using a MySQL database as a backend and PHP for the user interface.

1.1. Teams

This project can be done in teams of up to 3 students. If you prefer, you can work individually. However, students who choose to work individually should keep in mind that **all submissions will graded the same way regardless of the number of people in the team**. In other words, if you choose to work individually, you will need to do alone what other students will be doing in pairs or teams of three.

1.2. Due Dates and Deliverables

This assignment consists of **two submissions**. The first one is a proposal, worth 5% of your grade and due on at beginning of class on November 3. The second is a final report, worth 25% of your grade and due on November 24. The final report refer to an *actual running system* and include the SQL code that was used to build it.

1.3. Scope

It is up to you to decide what kind of functionality your system will offer, subject to the following constraints:

- Your system should not violate University of Toronto's Appropriate Use Policy or applicable laws.
- The system should be entirely your work. (Open source is a wonderful thing, but not for this project.)
- The implemented system should be comparable in complexity to the “Madame Z's Fortune Telling Shop” example (or *slightly* simpler). Your ER diagrams should consist of somewhere between 5 and 10 entities and your database should consist of 5 to 10 tables. This could either be a complete system or a prototype of a larger system. In other words, if you start off with an idea that turns out to be too complex, you can specify that you are building a simpler version of it for this project. In this case, make sure to clearly explain in what ways the prototype differs from the larger system.
- The system should have a clear purpose. (We cannot evaluate if you build it right if it is not clear what it was *supposed* to do.)
- Your system can replicate the functionality of an existing system, with some differences, as long as you do not have access to this systems' design documentation or code.
- Your system should not be too similar to the cases discussed in the class or in the book.

Here are some examples of systems that you could try to build:

- A product catalog.
- A database of recipes that allows users to find recipes by ingredients, type of cuisine, preparation method, etc.
- A online discussion forum, where users can start threads and post replies.
- A photo collection, allowing users to browse photos by album, time, place, etc. and to attach tags to them.
- A system where users can search for and find rooms for rent, either individually or jointly with others.

1.4. Basic and Advanced Features

Each database project must have *all* of the basic features listed below as well as *three* of the advanced features.

2. Basic Features

Your database must implement all of the following features.

2.1. An SQL Database on Yoda

You must set up the database on the Yoda database server. The database must be populated with enough data to enable a demo of all of the queries.

2.2. A PHP Front-End for Three Queries

You must implement a PHP front-end that would allow users to make at least **three** different kinds of queries against the database, returning HTML to the user. (You can use more than three queries to satisfy those requirements.)

- Each table in your database must be used for at least one query.
- All queries should be accessible from the project's first page by clicking or entering data into HTML forms.
- At least two of your queries must use joins.
- At least two of your queries must show multiple results, sorted by some column.
- At least one query must use an HTML form.
- ~~At least one query must be accessible by clicking on a link from the result of another query.~~
- The queries must not not allow SQL injection.

3. Advanced Features

*Your database should implement **two** of the following advanced features.*

3.1. User Accounts

Your system would allow users to login with a username and a password. Upon logining in the user would be able to either perform queries that are not available to anonymous users or to get customized results.

3.2. Enabling Modification of the Data by the Users

You system would allow the contents of the database to be modified through the web interface. (Examples of this would include posting comments, assigning tags, updating the values, etc.)

3.3. Backups

You would have a way to backup the content of the database to a set of files) and to restore the database from the backup. (You will need to explain the exact steps that would be necessary to recreate your database – with all the data – after dropping all tables, and include the necessary code. Your explanation should show clearly that you have actually gone through the process of backup and restoring.)

3.4. A Query with XML Output

Your system would have at least one query returnibng the data as XML rather than HTML. (Note: this should be proper XML rather than just XHTML.)

3.5. Transactions

Your system would involve a case where a single action by the user results in multiple queries and handle this situation using an SQL transaction. (This feature only makes sense if you are also doing 3.2.)

3.6. Stored Procedures

Your system would implement one of the queries using a stored procedure.

3.7. Publicly Available Data

Your system would incorporate a substantial quantity of publicly available data. ("A substantial quantity" means it should be more data than you would be able to enter by hand. Rather, you need to find a way to import the data.) For some examples of publicly available data see http://www.toronto.ca/open/catalogue.htm. (If you use such data, make sure that you comply with the licensing requirements.)

4. The Project Proposal

Your first deliverable is a project proposal, due at the beginning of class on November 3, 2010. Please submit **two copies**. The proposal should have the following parts:

4.1. The Project Name

Specify what your project will be called. Pick both a long name (a few words) and a short name (one word). The short name will be used for the project's URL. (For example, the long name could be "Madame Z's House of Fortune" and while short name would be "madame_z" or "fortune.")

4.2. The List of Team Members

List the members of the team with student numbers.

4.3. An Abstract

Provide a brief (100-200 words) summary of what your system will do.

4.4. A Functional Specification

Provide a more detailed (800-1500 words) description of the specific functionality that your system will and will not offer. Please state which three of the advanced features you intend to implement.

4.5. An ER Diagram and Data Dictionary

Provide an ER diagram for your system and a data dictionary, consisting of a brief explanation of each entity and a table documenting and explaining the attributes of each entity.

4.6. The Implementation Plan

Describe the schedule for the implementation and testing of the system. State clearly who is responsible for each piece and by when they will need to finish it. Do not forget to allocate time for testing and revisions!

4.7. Signatures

The proposal needs to be signed by all members of the team.

5. The Final Report

The final report is due at the beginning of class on November 24 and should contain the following parts:

5.1. The Name of the System

State the name of the system.

5.2. The Members of the Team

List the names and student numbers of all members of the team at the time of the submission. If this list is different from the one shown in your proposal, attach a footnote explaining why. (If you expect a change in team composition, please discuss this with the instructor *immediately*.)

5.3. The URL of the Implemented System and the Name of the Database

State the URL from which we can access the system. If the system requires login, please provide the login information. Additionally, please specified the name of the database used by your system.

5.4. An Abstract

Provide a brief (100-200 words) summary of what your system actually does.

5.5. Any Revisions to The Functional Specification

If your final system does not implement everything described in the functional specification or implements things not described in it, please describe the differences.

5.6. The Final ER Diagram and Data Dictionary

Include an ER diagram and data dictionary for your final system (as it was implemented). The final data dictionary should include and identify all necessary primary and foreign keys and should satisfy the conditions for 1NF, 2NF, and 3NF.

5.7. An Explanation of Basic Features

For each bullet point in section 1.2 provide a brief explanation (50-200 words) of how your system satisfies requirement. (Your explanation should make it clear to us how to verify the feature. For example, you could say: “From the first page you can enter the name of an ingredient, for example, 'cabbage', and click on 'Search by ingredient' to perform a query for all recipes using 'cabbage'. This query uses a join.)

5.8. An Explanation of Advanced Features

For each of the three advanced features that you choose, provide an explanation of 100-300 words of how your system satisfies it.

5.9. The SQL

Attach a print out of all the SQL used in your system, including the SQL for queries used inside PHP.

5.10. Credits

Describe briefly the contribution of each team member.

5.11. Signatures

The final report needs to be signed by all members of the team.