CpSc 4620/6620: Database Management Systems Project Requirements (Fall 2018)

Important Dates:

Instructor Approval of Topic, Aim & Scope: **Due Date: September 5.**
Proposal Due: **September 12**
Interim Report 1 Due: Thursday, **October 10**
Interim Report 2 Due: Thursday, **November 7**
Final Project Report Due: **Thursday, December 2 (Final)**

Overview:

We will do a semester long project to build a web portal using MySQL, PHP and popular development tools. The project topic is open; you can choose to build something that you really want. However, design and implementation of your project must meet the minimum requirements of depth and scope. You will need to stay focused and ensure that you complete and deliver certain pieces of your project in time. In general, you may not expect any programming help.

Team Work:

You may work individually or in groups of two, unless approved otherwise. Each of two members of a group will receive **identical grade** for the project. We cannot accept any complaints related to team member participation or any such issues; select your team member appropriately. All reports including the proposal must provide the information about the team clearly.

Submission Rule:

- Project reports must be submitted to **Canvas**. One team only needs to submit **one copy** with member name and CU username on it.
- Final implementation must work on production platform (Refer to the section "Deployment Environment" for details).

Work Procedure:

The project is to build a database-driven web portal. Specifically, you will need to complete the following tasks throughout the semester. *Note that different members of a team can work on some of these tasks concurrently.* It is a good idea to keep detailed documentation of each step as you continue working.

1. Your project topic should be rich enough to cover knowledge and skills covered in this class, but should be judiciously limited in scope and/or depth (without sacrificing the required specifications) to meet the deadlines. You can look at the list of possible project ideas; you are encouraged to **come up with your own ideas**. When selecting a topic to build your web portal, keep the following questions in mind:
    a. How do you plan to acquire/generate the data to populate your database? Use of real datasets is highly recommended. You may use program-generated synthetic datasets if real ones are too difficult to obtain (e.g. personal SSN, driver's license code).

   b. How are you going to use the data? What kind of queries do you want to ask? How do you import the data to MySQL database? Your project must include a data model that allows both **queries and updates**.

2. Design the database schema. Start with an ER diagram and convert it to a relational schema. Identify any constraints that hold in your application domain, and code them as database constraints. If you plan to work with real datasets, it is important to go over some **samples** of real data to validate your design (in fact, you should start Task 7 below as early as possible, in parallel to Tasks 3-6). Do not forget to apply database design theory and minimize information redundancies.
3. Create a sample database using a small dataset. You may generate this small dataset by hand (e.g. using MS Excel or Google spreadsheet). You will find this sample database very useful in testing; large data sets make debugging difficult. It is a good idea to write some scripts (PHP, Bash, Perl, or any your favorite language) to create/import/clean the sample database automatically. These script utilities will save you lots of manual database reconstruction time when debugging.
4. Design a user interface for your web portal using **static HTML**. This is a draft version of your web portal. It only shows the expected layout for the portal and is not necessary to include any dynamic content. Think about how a typical user would use your site. Optionally, it might be useful to build a draft version of the site first (i.e., with hard-coded rather than dynamically generated responses), while you refine your website design skills at the same time. Do not spend too much time on refining the look of your interface; you just need to understand the basic "flow" in order to figure out what database operations you need in each step of the user interaction.
5. Write **SQL queries** that will supply dynamic contents for the web pages you designed for Task 4. Also, write SQL code that modifies the database on behalf of the user. You may hard-code the query and update parameters. *Note that you can develop and test these SQL statements using phpMyAdmin or MySQL client command interface.*
6. Develop **PHP code** and **embed** it into previous static HTML pages. Start by implementing a "hello world" type of simple database-driven web application, deploy it in your development environment as well as production environment, and make sure that all parts are working together correctly. The later section will provide the detailed instruction for building/accessing these two environments.
7. Acquire the large "production" dataset, either by downloading it from a real data source or by generating it using your own programs. Make sure the dataset fits your schema. For real datasets, you might need to write programs/scripts to transform them into a form that is appropriate for loading into a database. For program-generated datasets, make sure they contain enough interesting "links" across rows of different tables, or else all your join queries may return empty results.
8. Test the SQL statements you developed for Task 5 in the large database. Do you run into any performance problems? Try creating some **additional indexes** to improve performance.
9. Implement and debug each module of your web portal. **Test** your website with a smaller sample database first. You may need to iterate the design and implementation several times in order to correct any unforeseen problems.
10. Test your website with the production dataset on production environment. Resolve any **performance** problems.

11. Polish the web interface using JavaScript and CSS. You may add logos and banners if you like; they are **optional**; they are not the focus of this course.

Project Requirement:

Although, as described in the syllabus, students can design their own project topics, here is a minimum set of functionalities that students must implement in their projects is presented here. **CpSc4620** students must design and implement all **basic functions**, and could optionally implement advanced features. **CpSc6620** students must implement both **basic functions and advanced features**. The requirement includes:

1. User account management. The web portal needs to provide a registration module to create a user account for accessing all implemented functions and features.
    a. Basic functions: New user registration module (user name, password and confirmation), sign-in module (user name, password and verification), and profile update module (First Name, Last Name, Age, Telephone Number, Email Address, Mailing Address).
    b. **Advanced feature**: The portal must validate the provided password should include **at least one capital letter** when registering a new account.
2. Data management. A signed-in user should be able to use a web interface to view, query and update their personal data sets (not profile data).
    a. Basic functions: This web portal should allow users to **view** category-level data sets related to your project topic (e.g. a list of user name, a list of in-stock item and so on). Users are able to **query** the data sets over specified category (e.g. tiger card balance, transaction records on a given date). Users should be able to **update** existing record (e.g. deposit/withdraw cash on bank account, change in-stock and so on).
    b. **Advanced features**: In the view page, the portal should automatically **split** a large query result into multi pages (e.g. display 25 items per page). Add some advanced **filters** or **sorting** features on query page (e.g. records in recent 7 days, transaction amount is over $500 and so on).
3. Database administration. The web portal should provide an administration account to maintain the underlying database.
    a. Basic functions: View/query/update user accounts. **Backup** selected tables related to web portal.
    b. **Advanced feature**: **Recover** database using the backup.

Development Tools:

➔ You can use either text editor:
- Vim: http://www.vim.org
  cheat sheet: http://people.csail.mit.edu/vgod/vim/vim-cheat-sheet-en.png
- Sublime Text: http://www.sublietext.com
- Atom: https://atom.io/

➔ or IDE (Integrated Development Environment):
- PhpStorm: https://www.jetbrains.com/phpstorm
  Note: you can apply student free version at: https://www.jetbrains.com/student/
- Others: https://cube3x.com/ides-for-php-development

Deployment Environment:

1. Test platform:
    a. Virtual machine: https://www.virtualbox.org/
    b. Linux image: https://www.virtualbox.org/wiki/Linux_Downloads (e.g. Ubuntu 14.04 i386)
    c. LAMP (Linux-Apache-MySQL-PHP): https://help.ubuntu.com/community/ApacheMySQLPHP
2. Production platform:
    a. Web page document root. You can access this directory at any one of lab machines. /web/home/{YOUR_CU_ACCOUNT}/public_html/4620/project [6620 students should replace 462 by 662 in the path]
    b. MySQL database. You can create a production MySQL database on our department server by using following link: https://buffet.cs.clemson.edu
    c. Web portal URL: http://people.cs.clemson.edu/~{YOUR_CU_ACCOUNT}/4620/project [6620 students should replace 4620 by 6620 in the URL]
    d. Deployment steps:
        i. Access any of the CS Linux lab machines.
        ii. Create the following folder if it does not exist: /web/home/{YOUR_CU_ACCOUNT}/public_html/4620/project. The directory /web/home/username already exists with a 2.0 GB quota for students.
        iii. This directory and any subdirectories must have permissions 711 or 755. 711 is recommended for security reasons (755 allows a user to get directory listing if an index.html file does not exist.)
        iv. Static files need 644 permissions.
        v. PHP scripts need 600 permissions.
        vi. CGI scripts need 700 permissions.

Project Reports:

Proposal. You should write a detailed description of what you propose to develop and design. Try to be precise, professional and not verbose. The objective is that the reader will get a clear idea of the basic outlines of your proposed project. Submit a project proposal to Blackboard including team member name.

Interim Report 1. You should have completed Tasks 1-5 and have started thinking about 6 and 7. If you plan to work with real data, you should also have made significant progress on Task 7 (you should at least ensure that it is feasible to obtain the real dataset, transform it, and load it into your database). Submit interim report to Canvas including the following components:

1. Name of team member and their CU user account.
2. A brief description of your web portal.
3. A plan for getting the data to populate your database and a snippet of sample data.
4. A list of assumptions that you are making about the data being modeled.
5. An ER diagram for your database design (draft version).
6. A schema for your database design: a list of tables with keys declared (draft version).
7. A description of the interface of the web portal (draft version).

Interim Report 2. Submit a project status report to Canvas including the following:

1. Name of team member and their CU user account.
2. Changes/updates to your original proposal (if any).
3. Final version of the ER diagram and the list of database tables with columns and keys.
4. A full version of SQL statements that match the functionality of web portal.
5. Summary of progress so far, e.g., components built, tasks completed, portal preview.
6. A list of tasks to be completed before the final due date.
7. Problem description (if any).

Final Report. Submit a final project report including the following:

1. Name of team member and their CU user account.
2. The problem description, motivation, and survey of related work as in the project proposal, possibly refined.
3. A detailed description of the approach you took, including the final system architecture and any design choices you made involving trade-offs.
4. Clearly list exactly what you have implemented for each of basic functions and/or advanced features and **how we can test it**.
5. Demonstrate success of your project by using appropriate screen snapshot and the detailed description. Note: You must provide a clear evidence on each of basic functions and/or advanced features.
6. Any limitations in your current implementation and thoughts on how you might fix them in the future.
7. A link to your web portal. To get graded, your final project must be hosted on production platform. It must be accessible for **at least two weeks** after the midnight of the final due date. It's your responsibility to keep it online in that period, otherwise a significant penalty is going to be applied on your final grade.
8. **An administration username and password for your web portal.**
9. A link to your source code. A URL of a downloadable tarball file including all your implementation (database dump file, PHP source file, and other related files). The URL must be accessible at the midnight of the final due date and remain accessible for **at least two weeks**. The source code directory should contain a **README** file describing structure of your source code and name of team member with their CU user accounts. You also need to put **the detailed comments** inside each .php file (e.g. explain the return value of echo SQL statement, declare the meaning of variables, show the logic of codebase).

Project Evaluation:

The evaluation of your project is based on completeness, quality and overall efforts on this semester long project. The grade distribution is:

- Reports (60%) [Interim 1: 15%, Interim 2: 15%, Final: 30%]
- Implementation (40%)

The followings are some of examination points we will perform on your project:

1. Does your web portal have all required basic functions and/or advanced features? Does everything work correctly and do all the buttons and links that we click indeed do what they are expected to do? We will see if you are really accessing the database dynamically and are not just "display" the data from some readymade source. We will ensure this by typing ad-hoc queries. Make sure all integrity constraints are preserved. You will also need to make sure that all query

results have column names neatly presented and that data is clearly tabulated.

2. A project report where you summarize all steps of your project in a single document attached snapshots of your web interface. This report can be easily created by cutting-and-pasting all the previous reports that you submitted, making sure you include the ER diagram, the details about normalization, your relational schema and, the queries you created in the project. Include the role of each team member in the project. Clearly identify the scope of everybody's contribution. If there were any problems during the course of the project (due to less than complete participation by some team member), identify them. Include your answer to the next question too. You do not need to submit your reports separately**.**
3. Implement inserts, updates, deletes, etc., or some more complex queries. Make an interface and connect it to the database. In your project report, clearly list exactly what you have implemented for this step and how we can test it.

Possible Project Ideas:

- TigerOne card system
- Library management
- B2C online sales webpage
- Automobile dealer sales system
- Mobile user portal
- Stock system
- Football ticketing system
- Your own idea!

Some Publicly Available Data Sets:

- Data.gov (http://www.data.gov/) has a huge compilation of data sets produced by the US government.
- The Supreme Court Database (http://scdb.wustl.edu/data.php) tracks all cases decided by the US Supreme Court.
- US government spending data (http://usaspending.gov/data) has information about government contracts and awards.
- Federal Election Commission (http://www.fec.gov/disclosure.shtml) has campaign finance data to download; their “disclosure portal” (http://www.fec.gov/pindex.shtml) also provide nice interfaces to explore the data.
- GovTrack.us (http://www.govtrack.us/developers) tracks all bills through the Congress and all votes casted by its members.
- National Institute for Computer-Assisted Reporting maintains a list of datasets of public interest (http://www.ire.org/nicar/database-library/). Use this list for examples of what datasets are “interesting”—they are generally not available to the public, but there may be alternative ways to obtain them.
- Google Fusion Table (http://www.google.com/fusiontables/Home/) hosts quite a number of datasets of public interest. It is a good place to find datasets or data sources to work on, and you can consider using it as a method of hosting your data for public access.