CS 380

Homework 5
Systems of HTML/Javascript/PHP/MySQL
Due: Nov 15, 2020, 11:59PM

Form a group of 2 or 3 people. Ideally these groups would be different from the groups for the first project (while ideal, this is not a requirement). Especially ideal would be for people who worked solo on the first project to not work solo on the second project. Anyone who did not present about the first project will be required to present, either on Nov 16 or Nov 19.

Send email to gtowell380@cs.brynmawr.edu with your group by Oct 31. One email per group is sufficient.

Background

For this assignment you will build all the parts of a system of interacting web pages with server-side support. I describe below two possibilities: a bank and a library. My descriptions are not meant to be exhaustive or even requirements; rather they should be viewed as sketches or outlines. As such, you should feel free to innovate; removing things that you think are not important and adding other things that you think are important. If you have a proposal for a completely different system, please sketch out a proposal (at a level of detail similar to that of my bank and library) and give it to me by Nov 2.

Requirements

The system must have the following elements:

- A login page. Logins then must persist allowing a user to interact with the system for at least 90 minutes without having to log in again.
- A logout button. Terminates logins
- When logged in at least two user "roles"; e.g., "user" and "administrator".
- At least two "activities" (in the form of web pages) for each of the roles. You will also need some way for the user in each role to select the activity.
- A database at the back end that supports the activities. The database should consist of at least 4 tables with primary and foreign key relationships among the tables
- Not precisely a requirement, but all of the HTML pages should both "look nice" and be easily usable. When doing web development, aesthetics matters. Likewise do not bury functions inside of dropdown lists (unless you have such dropdown lists clearly and obviously labelled.)
- The system should have few complete page loads. Rather most of the work should be done by updating page elements rather than the whole page. (So I expect extensive use of the javascript fetch (or its AJAX/JQuery relatives.)
- All communication with the server must be "login protected". Just because you are using fetch does not mean that you should be insecure.
- At least one element in the system must get real-time updates. This can be implemented using either long polling or server side events. You should not use basic "short polling".

A Banking System

A banking system might consist of two types of users: account holders and branch managers. Account holders may have multiple different accounts, and each account might be at a different branch. In addition to logging in, an account holder should be able to:

1. get a report showing the balance for each of their accounts
2. enter a transaction on their account (write a check).
3. Get a report of all the transactions that have occurred on a select account

Branch managers should be able to to at least the following:

1. Get a report of the balance for every account associated with their branch
2. Accept a deposit for some account
3. create a new account

For the real-time update, perhaps the balance report. That is, when ever a new transaction is posted to an account the balance on the screen updates immediately.

Note that creating accounts for branch managers is outside the scope of these interactions. You would manually insert Branch managers into the system by putting their information directly into the appropriate table(s) in the database.

Underlying this would be a set of database tables. I image that at least the following tables would be needed (but your table design could be wildly different from this)

1. User: userid, role, user details
2. Branch: branchid, branchlocation
3. Account: accountid, userid, branchid
4. Transaction: transactionid, accountid, transaction details

A Library System

A library system might consist of two types of users: librarians and borrowers. In addition, there are, of course, books. (Note a single person can create more than one library account) A borrower can do the following:

1. Get a list of the books they have borrowed
2. Borrow a book. This should include assignment of a due date
3. Get a listing of books available to be borrowed that meet some criteria — for instance, by an author

For the real-time update, perhaps the list of checked out books. That way the user can immediately see when book returns are processed, or if someone else is checking out books on their account)

A librarian should be able to do at least the following:

1. Indicate that a book has been returned to the library
2. Add a new book to the library's collection
3. Get a listing of all books that have been borrowed that meet a specific criterion: e.g. by an author or borrowed by a specific borrower, overdue, etc
4. Create a new borrower account

Note that creating accounts for librarians is outside the scope of these interactions. Hence, librarians would need to be manually added to the appropriate database tables.

Underlying this would be a set of database tables. I image that at least the following tables would be needed (but your table design could be wildly different from this)

1. User: userid, role, user details
2. Book: bookID, book details (I can easily image splitting the book table into at least 3 parts)
3. Account: accountid, userid
4. Transaction: transactionid, accountid, bookid, transaction type, due date

(I am not particularly happy with this set of tables, but I they could work.)

What to Hand in

A 1-2 page report. This report should include the following information:

1. The names of the members of the group.
2. Your database design. (Your tables and their relations and why you set up those particular relations). This is likely to be at least 1/2 of the text of the report.
3. The starting URL
4. The login credentials for an existing user in each of the roles in your system
5. A 1-2 paragraph reflection on things that went well or poorly; parts of the assignment that took the most time, etc.

The report should be submitted as a PDF prior to the due date.