Project 4 – Database Interaction

Overview

Since the goal of CSC390 is to teach you to program more than it is to teach theory, this project is your final exam. This project is worth 325 points, or 32.5% of your grade. This is a large project and will take you longer than previous projects so make sure to plan for this!

One of the most common uses for PHP is to interact with databases. This project will build on concepts used in the previous project and add database interaction and, optionally, AJAX. Your task is to create a website where people may create an account and then login. Once logged in, they will be able to create and modify to-do lists.

Since this isn't a database course, I have provided the SQL script to generate the database. You will be responsible for writing database queries though.

Feature Requirements

This project is similar to how projects work in the real world. You are given a set of required features below and it is your job to create an application with those features. How you accomplish this is up to you. This allows you to be creative, but also challenges you like you'll be challenged in the real world.

Account Registration

- Users must create an account in order to log in. An account should just have name, email, and password.
- If a user attempts to access certain pages without being logged in, you should redirect them to the login page.
- Users should be able to log in using their email/password combination.
- You should store all user accounts in a database.

To-Do List

- Once a user is logged in, they should be able to see their to-do lists. On this screen they can create a new to-do list or click any of the lists to see its full details. Every to-do list has a title.
- After clicking on a to-do list, it will take them to a page which shows all the items on the to-do list. There should be a place to add new tasks to the list.
- For each item on the list, there should be a "Delete" and "Complete" button or link. The "Delete" button will delete the task and the "Complete" button will complete the task.
- When a task is completed, you should record (in the database) the date and time of completion. When showing completed tasks on the page, you should show this completed date.
- Just to be clear, a user can have many to-do lists and a to-do list can have many tasks.
- Remember, each user should only see their lists! If one user logs in and can see another user's lists and tasks then something is horribly wrong!

Technical Requirements

In addition to the feature requirements, there are some technical requirements that you must meet.

- You MUST store passwords for accounts hashed! This means using the `password_hash()` and `password_verify()` functions built into PHP as of version 5.5 (we're using 7.3).
- All user accounts and tasks will be stored in a MySQL database using a schema I provide for you. You may modify this schema if you would like to add additional features, but other than that you must use it as is.
- All database queries MUST be done using "prepared statements" with bound parameters. This means using the PHP library called PDO.
- Your site must be styled with CSS. You should have a single CSS file for all your pages. Styling is still worth points and you should put some effort into it. Try to make the site look appealing and like something you'd want to use in the real world.
- **Extra Credit:** When adding a new task to a list, do an AJAX call to create the task and then use JavaScript to add the new task to the list. This will allow users to add items to the list without having to reload the entire page.
- It is important your project works correctly on Thor! This is where I will be testing them, so if your site isn't at the correct path and doesn't work on Thor you will lose points!

Tips and Hints

- Install XAMPP so you can work locally!
- Become comfortable with a database administration tool (phpMyAdmin or MySQL Workbench) for creating, editing, and viewing your database.
- If you have common functionality, consider putting that in a class in a separate file and including that when needed.
- Every user will have an ID. Store this ID on the session so you can keep track of which user is currently logged in and know what lists they are allowed to view.
- Try to separate logic and presentation. In class I've gone over the concept of a 'template file' which may be useful.
- MySQL stores dates in the format YYYY-MM-DD HH:MM:SS (that is, year-month-day then hours:minutes:seconds). You can create a timestamp in this format using PHP's `date()` function.
- To do AJAX use jQuery, Axios, or JavaScript's Fetch API. I can help with these.

Plan of Attack!

Do not attempt to do this project all at once. This project is larger than past projects and will take longer. The best thing for you to do is to do this project piece-by-piece. Do one part of it, once you have that part finished then move on to the next part. Repeat that until your project is done! Here's how I would tackle this project.

1. **Start on this project immediately.**
2. Set up some a database administration tool to make it easier to interact with your database.
3. **Ask me for help if you need it!**
4. Using the SQL file I provided, create the database.
5. Create the user registration page, just the actual HTML form.
6. Get the form submitting and when it submits have PHP read the form data and write a record to the 'user' table doing proper password hashing.
7. **Ask me for help if you need it please?**
8. Using project 3 as a guide, get your login page made and login functionality working. This should be very similar to project 3, you'll just be checking their login credentials against a database instead of hardcoded values.
9. Once all the login functionality is working, create a page with a form which allows them to create a new to-do list. Once you have this form made, make the submit program which reads this form data and creates a new record in the list table.
10. On the page made in step 9, do a database query to get and then display all a user's to-do lists (just the title and date created for now).
11. **I'm not kidding ask me for help if you need it!**
12. Create the page to allow them to view the tasks in an individual to-do list. Add a form to allow them to add items to the to-do list.
13. Add the 'Delete' link on each task which will delete that task.
14. Create the page which deletes a task.
15. Add 'complete' link on each task which will complete the task.
16. Create the page which completes a task.
17. You're done at this point! There may be a few small features left to add, but you're feature complete!
18. Now style your to-do app using CSS to make it look better.
19. **You're totally done! Stop unless you want extra credit.**
20. **Get ready to ask me for help with the AJAX stuff. It can get complicated.**
21. Once you have all this working, now attempt to get the add task form working using AJAX. This means using JavaScript to get the form data, do an AJAX call, get a response from the server, and then adding a new item to the to-do list.

Grading

The assignment is worth 325 points. You will be graded on the following criteria:

User Registration & Login work	70
View and create to-lists with titles	70
View, add, and delete tasks on the to-do list	70
All database interaction done properly	30
CSS styling	30
Site works on Thor	30
Code quality/header comment blocks	25
Total	**325**

Submitting your work

Your work should be submitted as an archive (.zip, .7z, .rar) on Moodle. This archive should include your all files needed for your project.

This project is due Tuesday, November 24th at 10:25am.

Late submissions will not be accepted!

(yes this is 5 minutes before the start of our final class)

We will do live demos of all projects during the final exam period.