Project 4 – Database Interaction

Overview

This project brings all of the skills learned in CSC390 together. 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 see and maintain a list of video games they'd like to play or have played (game backlog)

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.

You may work in groups of up to 3 people. Please let me know if you'll be working in a group.

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.

Game Backlog

- Once a user is logged in, they should be able to see their video game backlog. It should show all the games in the backlog. There should be a place to add new games to the list.
- For each game on the list, it should be possible to remove the game, mark the game as "started", and mark the game as "completed".
- When a game is started or completed, you should record (in the database) the date and time. When showing started or completed games on the page, you should show this completed date.
- At the top of this page, give options to filter the backlog in three different way: All games, started games, and completed games. The "started games" filter should only show games that have been started but not completed.
- Remember, each user should only see their list! If one user logs in and can see another user's list and 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 8.1).
- All user accounts and games 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 game to a backlog, do an AJAX call to create the database record and then use JavaScript to add the new game to the list. This will allow users to add games to the backlog without having to reload the entire page. You could do other AJAX functionality if desired.
- 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 JavaScript's Fetch API or Axios. 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 that does a database query to get and then display the user's game backlog.
10. Add a form to this page which allows them to add a new game to their backlog. Once you have this form made, make the submit program which reads this form data and creates a new record in the backlog table.
11. **I'm not kidding ask me for help if you need it!**
12. Add the 'Delete' link or button for each game which will delete that game.
13. Create the page which deletes the game.
14. Add 'started' link or button for each game which will mark that game as started.
15. Create the page which marks the game as started.
16. Add 'completed' link or button for each game which will mark that game as completed.
17. Create the page which marks the game as completed.
18. Add the ability to filter the list to "all", "started", and "completed"
19. You're done at this point! There may be a few small features left to add, but you're basically feature complete!
20. Now style your app using CSS to make it look better.
21. **You're totally done! Stop unless you want extra credit.**
22. **Get ready to ask me for help with the AJAX stuff. It can get complicated.**
23. Once you have all this working, now attempt to get the add game 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 game to the list on the page.

Grading

The assignment is worth 300 points. You will be graded on the following criteria:

User Registration & Login work	60
View games on the backlog	60
Add and delete games from the backlog	40
Mark games as started or completed	20
Ability to filter backlog	30
All database interaction done properly	30
CSS styling	20
Site works on Thor	20
Code quality/header comment blocks	20
Total	**300**

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 Monday, December 9th at 12:55pm.

Late submissions will not be accepted!

(yes this is 5 minutes before the start of our final time so we can demo the projects in class)