CS 494 Final Project

Overview

This project is going to combine things from the previous assignments and expand on them slightly. You will have asynchronous calls where appropriate and will implement a login system that supports the creation of new user accounts. Finally you will implement some feature that goes beyond the material covered in class. A good option is to use another students how to guide and implement a feature they went over that would work well with your site.

Requirements

User Accounts

You need to have a login page similar to homework 3 that will allow a user to login. If a login attempt fails, it should use an Ajax call (do not refresh the whole page) to display an error message indicating the login was unsuccessful and allow the user to try again or link to a page to create an account.

The account creation page should specify some set of requirements for a username and password. These should be validated (eg. password length of at least 8 characters and a username of at least length 4 characters) on the client side and a warning should be displayed if they do not meet the validation requirements. If an account name is already in use, the user should be notified via an Ajax call that the username was taken and to try a different username.

Error Messages

Any sort of notifications or error messages should be displayed asynchronously as mentioned in the user accounts section. So this means that for most of the operations that require server side processing you should be using Ajax calls so you can handle a failure without redirecting to a new page.

Styling

You should make use of CSS to style and layout your webpage. It need not be flashy, but it should look significantly different than the page would look without CSS. Elements should not all be crammed together in the upper left corner of the page.

Functionality

Once logged in the user should be able to add some data or change some settings. These changes should be stored in a database. It could be uploading files or showing messages just sent to the current user. The website can do whatever you want it to in terms of function so long as it meets the other requirements. It can be a message board, you can make a assignment tracking website to keep track of homework assignments, it can be a dashboard to keep track of selected sports teams. It is up to you.

As usual

All pages should validate error free as HTML5 at validator.w3.org. The most recent version of Chrome should throw no console errors in any situation. (If you use a library that throws css warnings and it is internal to the included files, that is OK) All pages should pass php -l *filename* parsing on the command line.

Deliverables

In the Blackboard submission text box include a url to the login page along two valid username/password combinations that will show us different content in case we are unable to create a new account.

A single pdf containing the following

- Screenshots of the major portions of your website (this is just so we can make sure nothing terrible happened between when you submitted and when we viewed it).
- One code example that shows an instance of displaying an error or redirecting to a new page based on the result of an Ajax call failing or succeeding respectively.
- A description of what you implemented that went beyond the scope of the class. For example, if you used an API to get sports data from ESPN, talk briefly about that and how you used it.
- A screenshot of the main content page (whatever that happens to be) passing validation at validator.w3.org
- A screenshot or console output of all php files passing ‘php -l’ with no errors

A single .zip file that contains all of the source used. We should basically be able to unzip this into a directory on a server and go view the website there.