Time Logger

1) MVC - How you delineated your objects

Model – The model in the time logger is not quite organized. In total there were 33 files that contained code inside of them. Most of the action was does server side through php. Because of the nature of the time logger (being heavily formed based) it made php the ideal language to run the time logger on. About half of the php pages are just html pages which I guess you can count as a view. None of the pages were dynamic, meaning any change in page layout was almost always a change in the page itself. Most of the files are for the administrator section of the project, which governed the edits, additions, and deletions of users, sessions, and classes.

View – The view object is not particularly fleshed out either. For the most part, the webpages were simple forms that were written in html and reused. As you can imagine the deletion of a session, user, or class are all very similar visually, so it was just a matter of changing the labels, names, headers, ids, and titles.

Controller – For the time logger project, most of the behind the scenes actions occur in php. However, most of the script that are useful to us are written in SQL. This is because time logging is an inherently database heavy project. We simply used php to query SQL scripts which ran with INSERT, DELETE, and other SQL keywords to change the database appropriately.

2) Objects - Javascript/PHP - Serialization

Passing info with Javascript Objects to PHP Objects with Cookies

The cookies and sessions are not a significant aspect of this project. Most of the data was stored in local forms which was more than enough for this project. The form would send the data via the POST method and have php handle the rest. However, we did use cookies and Sessions for storing the user id as well and the username. The user id cookie storage became useful when it came time for the user to request the amount of times they have put in. This allows the server to query the time without the need to ask the user what their id is.

3) Reading/Writing Files/Local Storage

JSON – Javascript

There was no utilization of local storage or JSON files in this project. All of the data is stored in a MySQL database. The data contains the username, password. It also has tables for sessions and subjects which are covered in the next question.

4) Databases SQL - Identify Entities, Xref, and Enum Tables

Entities – The database had a session entities and a user entities. The user entities stored the usernames and passwords. The session entities stored the start date and time of the session, the end date and time of the session, the session subject, and the id of the user that attended that section.

Xref – perhaps there was a chance to xref the user to the sessions themselves but I did not think it was necessary for this database.

Enum – The subject list is an enum table and it simply contains all the subjects that are available for selection for the user.

5) Form Validation - Reqular Expressions

My form validation and regex usage came in the form of verifying whether or not the user sign up variables were valid in accordance to the username characters and password length security.

6) User-Admin-Login

The administrator had the ability to add users, add sessions, edit users, delete sessions, and delete users. The admin also could sort the database by session subject and by session id.

The user was only able to view their own sessions and also create their own sessions.

7) Cookies - Sessions - Securing Pages

Cookies and sessions were covered in question #2. However, in this section I can talk about security. As mentioned in my presentation, I found that there was a security flaw in which the user was able to delete the read-only portion of the html code via inspect element and simply add on more hours. Of course, the end session time should be handled via the server, that way the server time is in sync with the real time, rather than an edited time. However, this would also apply to the start time. Perhaps, a use of storing the session during login in a $\_SESSION variable would be the right choice. However, the system would have no way of knowing which class is currently in session.