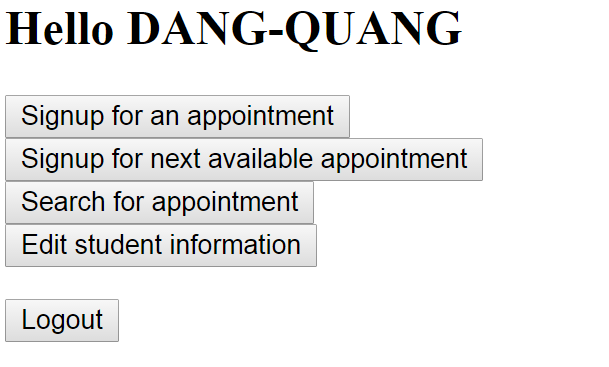
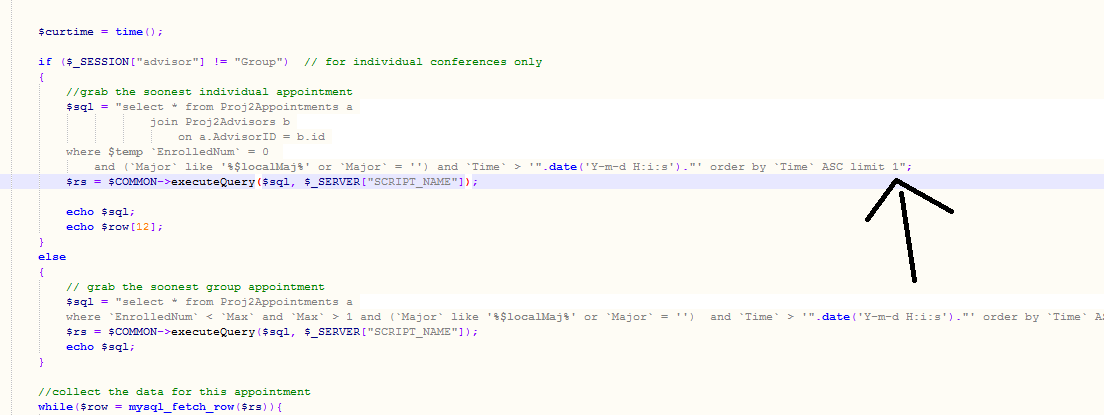
**TASK 1 – Find next available for students**

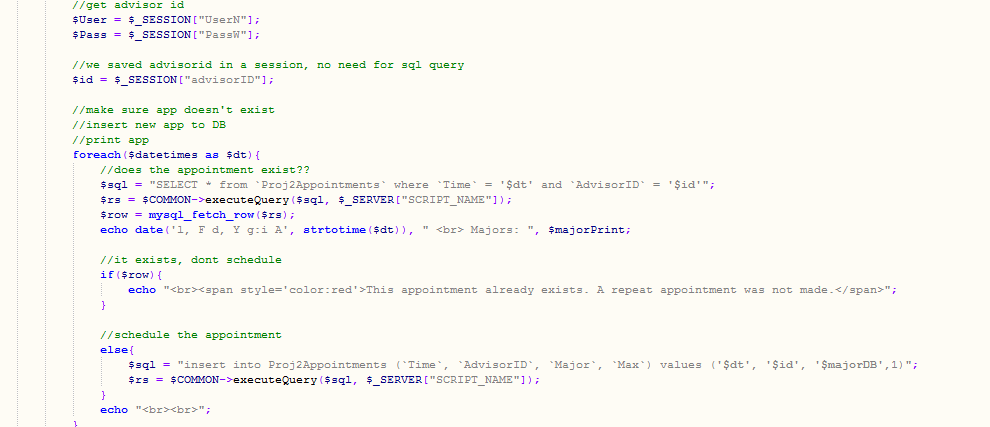
The key step, besides adding a new option on the student home page, was to create a new php file to handle processing of finding the next available appointment. This file was as simple as copying the old file to find available times available, and changing the SQL query to limit 1. The results were already ordered by time in ascending order.

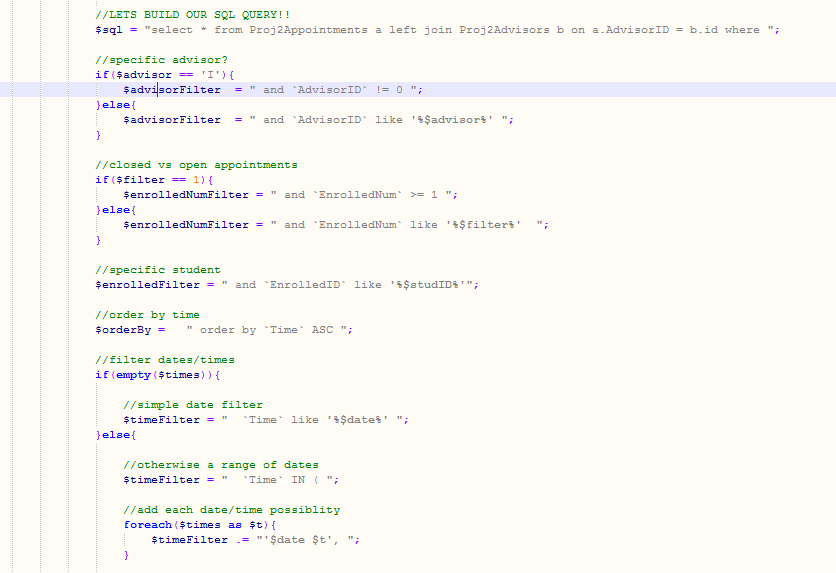
 



**TASK 2 – Code Commenting and Documentation**

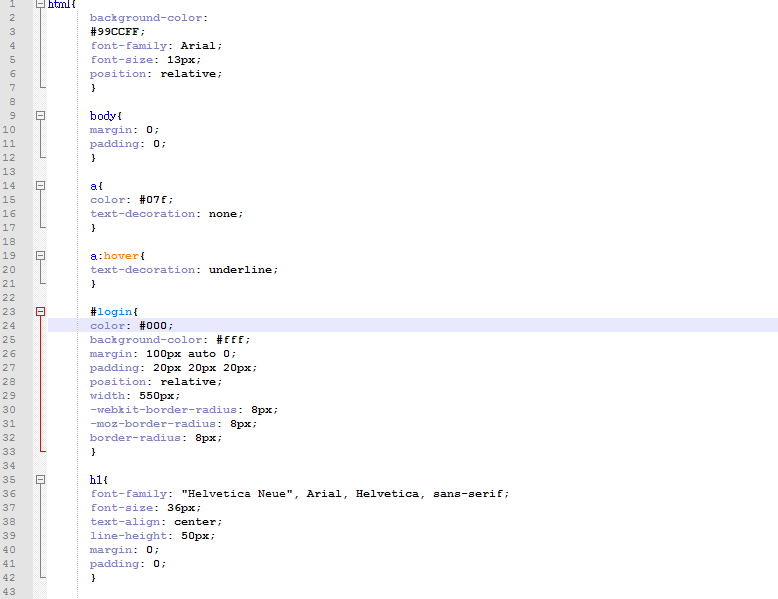
This task is slightly more difficult to provide, without displaying the commenting in every single page. In general, the key areas to document were SQL queries, php loops, and php if statements, as these are the most complex and most likely components to cause confusion.

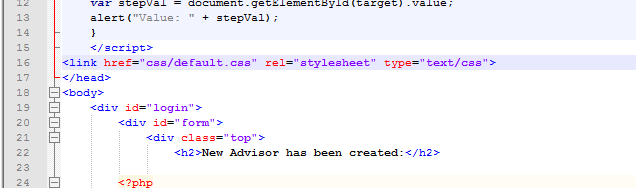




**TASK 3 – Move css to a separate file**

This task was a key but large trivial step to execute. I simply copied out all of the identical css pieces, and pasted them in a single css file for reference in each location that it was used.



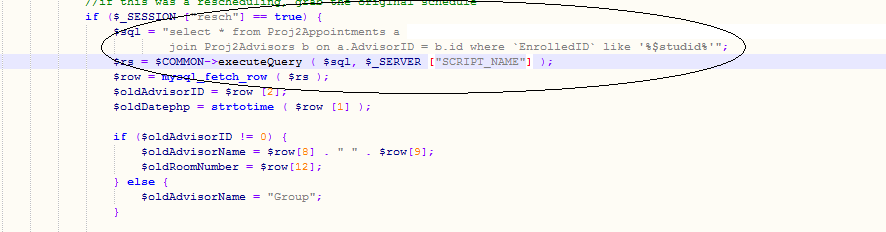


**TASK 4 – merge/remove redundant SQL queries**

This task was the most difficult by far, and also left the most freedom in terms of execution. There were three strategies that I implemented in order to merge and remove redundant SQL queries.

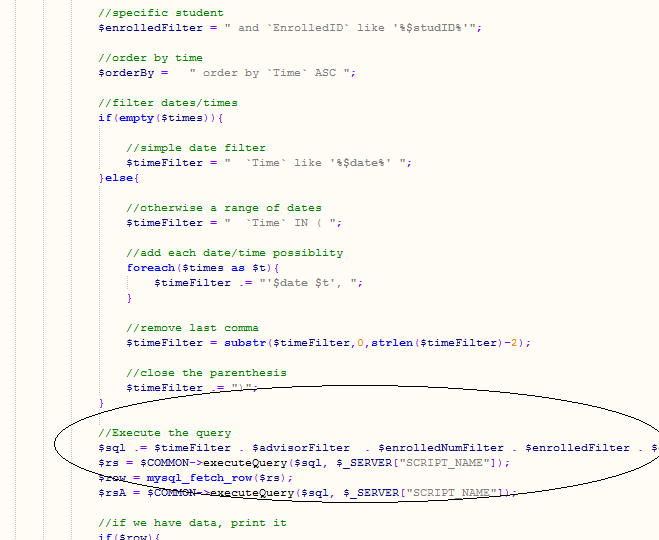
1. Use Joins to prevent the need for multiple queries on separate tables.

In several instances, the original code would query appointments, and then for each advisor, would issue another query for each appointment to determine the advisor’s name. In these instances, I replaced the multiple queries with a single join query which cut down on the number of database operations.



1. Use Dynamic SQL queries to prevent rewriting queries with slight adjustments.

In several instances, the original code would have 5-10 different sql queries for various user scenarios, with much of the query duplicated. In these instances, I used only one query variable and used dynamic filter conditions to capture the cases necessary.



1. Use PHP Session Variables to save and prevent re-querying for data that was just queried a page before.

In several instances, the original code would reissue the same queries on every page to repopulate all the data. In reality, we can just query for all the data at login and save it and then just reference whatever is needed as the user requests it.

