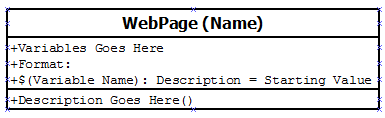
Project 1

Douglas Lueben

# Part 1: Dia

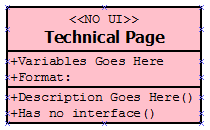
The Dia diagram shows how each web page connects to another. Although a simple Key is on the diagram itself, here is a more detailed reference to my diagram:

### Web PAge



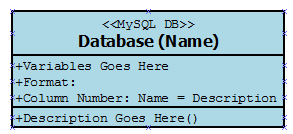
This box represents one page of the website. The file name is at the top, the middle portion stores the important variables it uses from other pages and creates with a short description of what they do (for a more detailed description, refer to Part 6, Variable Descriptions), and the bottom provides a short description of the website

### Technical web Page



The contents of the box are the same as Web Page (see above), except the difference is that it refers to a file with **no user interface.** These pages are used primarily for redirecting, running SQL queries, and making “behind the scenes” calculations.

### Database



The Dia also has a short description of the three databases this project uses. The top portion is the name of the database, the middle portion describes each column, and the bottom portion describes what the database does.

### Note



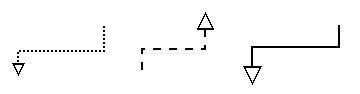
The small white post-it notes provide minor details to the reader. If they’re on top of a line, they represent the button that the user had to press to get directed to a given page.

### Condition

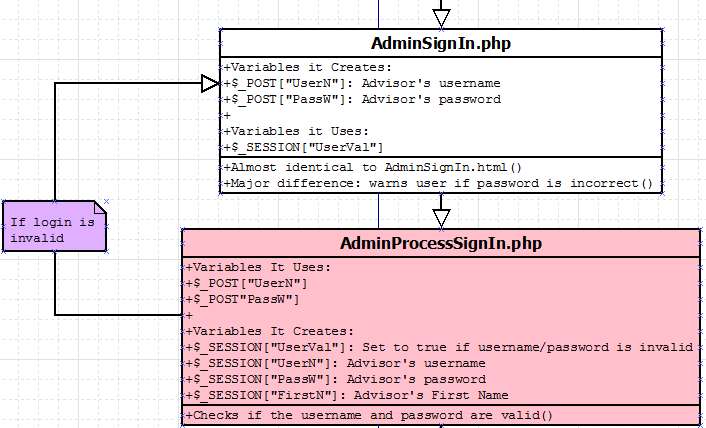


The purple post-it notes tell you when the user is directed down a given path. For example, StudProcessHome.php redirects the user to a part of the web site based on their selection in 02StudHome.php. The condition notes let the reader of the Dia diagram know where the user will be redirected to from StudProcessHome.php based on the value that is passed in from 02StudHome.php

### Lines



A line represents a connection between two web pages. The side with the arrow indicates the direction, whatever page it touches shows the user which page they’re going to. So in this example:



AdminSignIn.php has only **one** output: AdminProcessSignIn.php! AdminProcessSignIn.php has a **conditional** output, that is, if the login is invalid you will go back to AdminSignIn.php

There are three different types of lines because sometimes the lines overlap. I chose three different stroke amounts to make it easy to read which arrows are headed to which pages.

**The Dia diagram will be included in the back of the binder, after the documentation.**

# Part 2: Updated Items (From Rubric)

The following improvements were required by Part 3 of the rubric. Each section corresponds to one of the items in the rubric’s table “List of Additions Required” that I successfully implemented. The changes I make can span multiple pages, so the format for each change is the following:

### FileNAME.extension – Name of Fix

Description of fix goes here.

## Update Options in DB

This was the largest edit to the project, because so many pages reference majors. I didn’t have to change the way majors were held in the database, since VARCHAR will only hold as many characters as there actually are in a given major name, but I had to change the majority of the website to reflect this change.

The idea behind this fix is that I need to replace the name of the major (Computer Science, Mechanical Engineering, etc.) with the equivalent acronym (CMSC, MENG) in the database only, while still showing the full name to the user.

Since “Engineering Undecided” is not a part of this website, I’m assuming it’s a past/future option and have not implemented its acronym.

The rubric says that the only side that needs to see the full major names is the student side, so those are the only pages where I updated the display names.

Here are the pages that were updated to accommodate this:

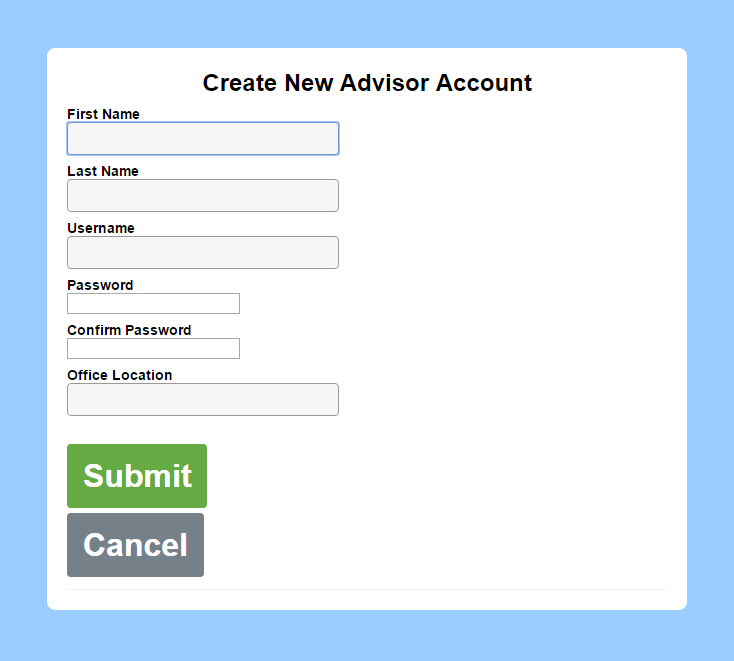
* 01StudSignIn.html
* 06StudEditInfo.php
* AdminScheduleInd.php
* AdminScheduleGroup.php
* 11StudSearchResult.php

## Display where office advisors are located in “Viewing Appointment”

To display where an advisor’s office is located, I had to update the database (To see details on the new database structure vs. the original, refer to Part 4) with an extra column that holds their office location. This means I had to make a number of changes to the website:

### AdminCreateNewADv.php – Add “Office” section to form

When creating a new advisor, I added a section to write their office location down. It now looks like this:



### AdminProcessCreateNEw.php – Convert Office Post to SEssion

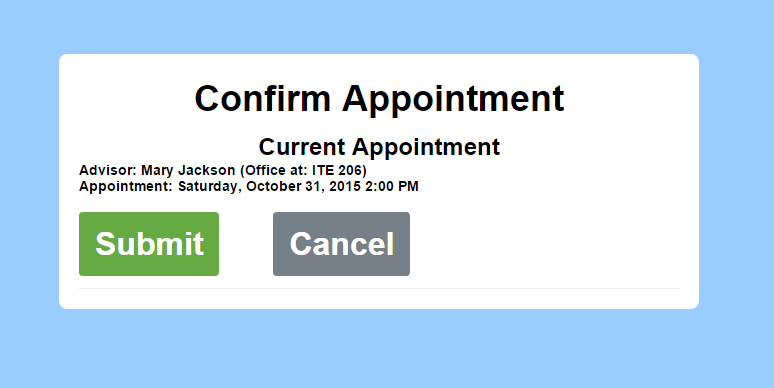
AdminProcessCreateNew.php converts all the post variables from AdminCreateNewAdv.php and turns them into session variables, so I had to update this page to handle converting $\_POST[“OfficeL”] to $\_SESSION[“OfficeL”]

### AdminCreateNew.php – Update the advisor in the db

I updated the insert query to add the advisor’s office location

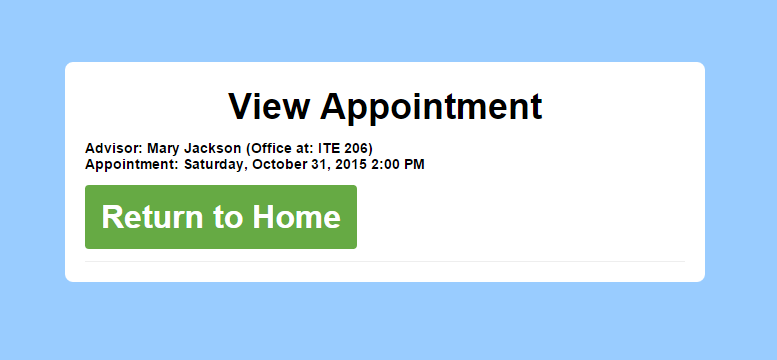
### 10STudConfirmSch.php – ADded the office location to the confirmation

10StudConfirmSch.php shows the student user the details of their appointment, so I added the advisor’s office location to this page. It now looks like this:



### 04StudViewApp.php – Added the office location to review

04StudViewApp.php shows the student user the details of their appointment, so I added the office location of an advisor to this page:



# PArt 3: Additional Items (from Rubric)

The following improvements were required by Part 3 of the rubric. Each section corresponds to one of the items in the rubric’s table “List of Additions Required” that I successfully implemented. The changes I make can span multiple pages, so the format for each change is the following:

### FileNAME.extension – Name of Fix

Description of fix goes here.

## 1 – Find “next Available” for students (with major in mind)

This feature allows a student that isn’t picky get instantly assigned to the next available appointment. To implement this, I made the following changes:

### 03StudSelectType.php - added a “next Available” buttoN

03StudSelectType.php is the first page a student user sees when trying to create an appointment. Originally, it had two buttons: one for group advising, and another for individual advising. The way I interpreted the “Next Available” option is that the user is not picky about the details of the appointment, they just want the soonest one they can get. So, I added a third button to this page for the “Next Available” appointment. When pressed, it lets StudProcessType.php determine the next available appointment and give it to the student user.

### STUDPROCESSTYPE.php – Updated to find Next Available Appointment

I added a section to StudProcessType.php to handle finding the next available appointment if the student user selected the “Next Available” button on the previous page. I run the following SQL query to get the next available appointment:

SELECT \* FROM `Proj2Appointments` WHERE (`Time` > NOW()) ORDER BY `Time` LIMIT 1

And send that information to 10StudConfirmSch.php

The process of selecting the next available appointment is as follows:

|  |  |
| --- | --- |
| **1. Select “Signup for an appointment” on the home page for a given student** |  |
| **2. Select “Next Available”** |  |
| **3. The system will decide when the next appointment is. The appointment information will be listed to make sure that you want to select this appointment or not.** It does *not* distingush between Individual and Group advising, it just selects the next available appointment per the rubric’s requirements. |  |
| **4. Press Submit** |  |
| **5. The appointment has been made.** |  |
| **If for some reason there are NO available appointments, you’ll see this screen instead:** |  |

## 2 – More code commenting and documentation

For commenting, I updated each page of the web site with comments that describe what each chunk of code is trying to do to make it more readable. For HTML forms, I put <!-- --> tags above buttons, form options, and titles to explain what they’re used for.

For Documentation, I created a 6th part to the assignment located below, where I listed every page of the web site with a brief summary, a list of page specific bug fixes, and what variables they used and created. I also created a comprehensive Dia diagram.

## 3 – Move CSS from PHP/HTML file to own stand alone page

The CSS file was, conveniently, already provided with the project. However, there were two issues with it that I fixed:

### MISC. – Replaced Inline CSS with EXTERNAL CSS

Some pages had their own CSS in the file itself. On those pages, I removed the existing CSS and had it reference the provided one instead by adding the following line of code:

<link rel = ‘stylesheet’ type=’text/css’ href=**’css/standard.css’** />

In place of whatever CSS they had in their <head> tags.

Pages that were updated to fix this:

* StudentAdminSignIn.html
* AdminPrintSchedule.php
* AdminEditApp.php
* AdminSearchApp.php
* 12StudExit.php
* 05StudCancelApp.php
* 09StudSearchApp.php

### MISC. – Fixed href in EXTERNAL CSS

Some pages had a reference to the provided CSS, but the code was wrong. Namely, the stylesheet line read:

<link rel = ‘stylesheet’ type=’text/css’ href=**’../css/standard.css’** />

When it should’ve read:

<link rel = ‘stylesheet’ type=’text/css’ href=**’css/standard.css’** />

This is because the CSS folder is in the same directory as the web pages calling it.

Pages that were updated to fix this:

* AdminSignIn.php
* AdminUI.php
* AdminConfirmScheIndApp.php
* AdminScheduleApp.php
* AdminScheduleInd.php
* AdminScheduleGroup.php
* AdminConfirmScheGroupApp.php
* AdminPrintResults.php
* AdminEditGroup.php
* AdminProceedEditGroup.php
* AdminEditInd.php
* AdminConfirmEditGroup.php
* AdminConfirmEditInd.php
* AdminSearchResults.php
* AdminCreateNewAdv.php
* 01StudSignIn.html
* 02StudHome.php
* 03StudSelectType.php
* 07StudSelectAdvisor.php
* 08StudSelectTime.php
* 10StudConfirmSch.php
* 04StudViewApp.php
* 13StudDenied.php
* 11StudSearchResult.php
* 06StudEditInfo.php

## 4 – Merge/Remove reductant SQL queries

Throughout the web site, there were many unnecessary calls to the MySQL server. These have been removed or merged to reduce strain on the server, and decrease load times. Unlike the fixes for standard.css that generally fell into two categories, each merge/removal was a unique case that took serious discretion.

### AdminUI.php - Removed Call to Database/CommonMethods, SQL QUERY

I saw that every time this page was loaded, it called the MySQL database just to get the advisor’s first name. Since this page will be visited frequently, I decided to store the first name in AdminProcessSignIn.php instead so that this page does not need to call the database at all, but instead gets the value from a $\_SESSION variable. This decision was made to reduce the number of SQL queries in the project, and remove unnecessary strain on the database.

### AdminConfirmScheIndApp.php – Replaced Foreach loop

AdminConfirmScheIndApp.php had a foreach loop that checked to see if an appointment existed for each appointment that the advisor tried to create, and if not inserted it into the database. This puts a heavy load on the server, causing number of queries to the server to have O(n), where n is the number of appointments that the advisor tried to create. This is the section of code that causes this inefficiency:

foreach($datetimes as $dt){  
 $sql = "SELECT \* from `Proj2Appointments` where `Time` = '$dt' and `AdvisorID` = '$id'";  
$rs = $COMMON->executeQuery($sql, $\_SERVER["SCRIPT\_NAME"]);  
$row = mysql\_fetch\_row($rs);  
echo date('l, F d, Y g:i A', strtotime($dt)), " <br> Majors: ", $majorPrint;  
if($row){  
echo "<br><span style='color:red'>!!</span>";  
}  
else{  
$sql = "insert into Proj2Appointments (`Time`, `AdvisorID`, `Major`, `Max`) values ('$dt', '$id', '$majorDB',1)";  
$rs = $COMMON->executeQuery($sql, $\_SERVER["SCRIPT\_NAME"]);  
}  
echo "<br><br>";  
}

Every time the loop above runs, it executes two MySQL queries. Instead, I modified the page to only run two queries: a single SELECT query, and a single INSERT query. This reduces the number of queries from O(n) to O(2). The two queries are shown below:

Select Query:

SELECT \* FROM `Proj2Appointments` WHERE (`Time` = $dt AND `AdvisorID` = $id) OR ...

Insert Query:

INSERT INTO `Proj2Appointments` (`Time`, `AdvisorID`, `Major`, `Max`) VALUES (...Sample values go here...),(...Sample values go here...),...

Even though this increases the amount of code in the .php file, I justify this by being able to take a load off of the MySQL server by reducing the number of SQL queries.

A similar edit was made to AdminConfirmScheGroupApp.php

### AdminConfirmScheGroupApp.php – Replaced Foreach loop

AdminConfirmScheGroupApp.php had a foreach loop that checked to see if an appointment existed for each appointment that the advisor tried to create, and if not inserted it into the database. This puts a heavy load on the server, causing number of queries to the server to have O(n), where n is the number of appointments that the advisor tried to create. This is the section of code that causes this inefficiency:

foreach($datetimes as $dt){  
 $sql = "SELECT \* from `Proj2Appointments` where `Time` = '$dt' and `AdvisorID` = '0'";  
 $rs = $COMMON->executeQuery($sql, $\_SERVER["SCRIPT\_NAME"]);  
 $row = mysql\_fetch\_row($rs);  
 echo date('l, F d, Y g:i A', strtotime($dt)), "<br>Majors included: ", $majorPrint;  
 echo("<br>Number of seats: $studentLimit");  
 if($row){  
 echo "<br><span style='color:red'>!!</span>";  
 }  
 else{  
 $sql = "insert into Proj2Appointments (`Time`, `AdvisorID`, `Major`, `Max`) values ('$dt', '0', '$majorDB','$studentLimit')";  
 $rs = $COMMON->executeQuery($sql, $\_SERVER["SCRIPT\_NAME"]);  
 }  
 echo "<br><br>";  
}

Every time the loop above runs, it executes two MySQL queries. Instead, I modified the page to only run two queries: a single SELECT query, and a single INSERT query. This reduces the number of queries from O(n) to O(2). The two queries are shown below:

Select Query:

SELECT \* FROM `Proj2Appointments` WHERE (`Time` = $dt AND `AdvisorID` = 0) OR ...

Insert Query:

INSERT INTO `Proj2Appointments` (`Time`, `AdvisorID`, `Major`, `Max`) VALUES (...Sample values go here...),(...Sample values go here...),...

Even though this increases the amount of code in the .php file, I justify this by being able to take a load off of the MySQL server by reducing the number of SQL queries.

A similar edit was made to AdminConfirmScheIndApp.php

### 06StudEditInfo.php – Removed While Loop/Inefficient SQL query

Towards the top of 06StudEditInfo.php, the page tries to get the student’s information from the database by looking up their student ID and pulling their information. Unfortunately, instead of only pulling the row that contains the student’s user ID, the original query actually pulls all students from Proj2Students, and then sifts through them with a while loop to see if any of them match the current user’s student ID. This is the original code that caused the problem:

$sql = "select \* from Proj2Students";  
$rs = $COMMON->executeQuery($sql, $\_SERVER["SCRIPT\_NAME"]);  
while($row = mysql\_fetch\_row($rs)){  
 if($row[3] == $\_SESSION["studID"]){  
 $\_SESSION["firstN"] = $row[1];  
 $\_SESSION["lastN"] = $row[2];  
 $\_SESSION["email"] = $row[4];  
 $\_SESSION["major"] = $row[5];  
 }  
}

This code causes inefficiency on the MySQL server (by having it return ALL students, especially on such a large campus) and on the php script (having to loop through EVERY student to find a match, and not stopping once one if found). Instead, I replaced it with this:

//Prepare the SQL query  
$sql = "SELECT \* FROM `Proj2Students` WHERE (`StudentID` = '";  
$sql .= $\_SESSION["studID"];  
$sql .= "')";  
  
//Fetch the row  
$rs = $COMMON->executeQuery($sql, $\_SERVER["SCRIPT\_NAME"]);  
  
//Parse the row for information  
$row = (mysql\_fetch\_row($rs));  
$\_SESSION["firstN"] = $row[1];  
$\_SESSION["lastN"] = $row[2];  
$\_SESSION["email"] = $row[4];  
$\_SESSION["major"] = $row[5];

## 5 – Map overall setup of pages in project In Dia

This was finished in Part 1, listed above. Please refer to that section for all Dia-related information.

# Part 4: Personal Database

**Note:** Although this is project 1, I did not change the format of all the database names throughout this project to Proj1\*\*\*\*\* from Proj2\*\*\*\*\*\*, since it appears so many times throughout the website. Keeping the names the way significantly reduced the amount of work it would take to make this web site.

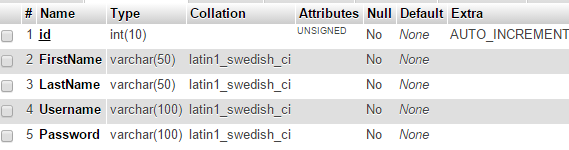
## Challenges

There weren’t that many challenges with the databases, because they were already set up to work with the website. The one difficulty I had that was sort of database related was a time where for some reason, when creating Individual Appointments as an admin, the “Max” number of students would be set to zero. Otherwise, I’ve worked with phpmyadmin before, so it was smooth sailing.

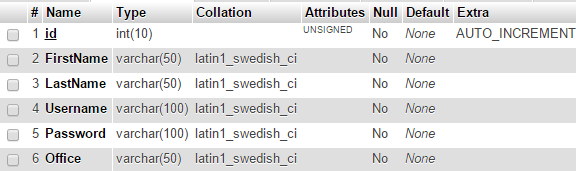
## Proj2Advisors

Purpose: Holds information on all of the academic advisors

Original Structure:



New Structure:



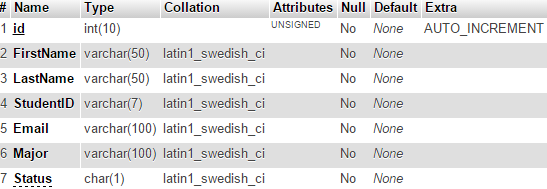
Column Documentation:

* id
  + Primary Key for the table
* FirstName
  + Advisor’s First Name
* LastName
  + Advisor’s Last Name
* Username
  + Advisor’s username
* Password
  + Advisor’s password
* Office
  + The location of the advisor’s office

## Proj2Students

Purpose: Holds all student information

Structure:



This database was *not* edited in any way from the original.

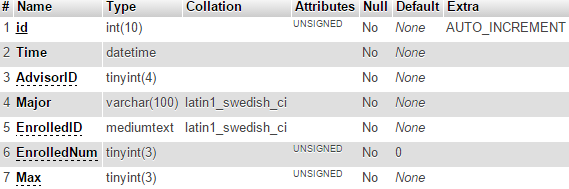
Column Documentation:

* id
  + Primary Key
* FirstName
  + Student’s first name
* LastName
  + Student’s last name
* StudentID
  + Student’s UMBC ID
* Email
  + Student’s email address
* Major
  + Student’s declared major
* Status
  + The status of the student’s current appointment, if they have one

## Proj2Appointments

Purpose: Holds all appointments in the system

Structure:



This database was *not* edited in any way from the original.

Column Documentation:

* id
  + Primary Key
* Time
  + The date + time of the appointment
* AdvisorID
  + The id of the advisor that’s in charge of the appointment
* Major
  + The majors allowed in this appointment
* EnrolledID
  + The UMBC Id of any students registered with this appointment
* EnrolledNum
  + The number of students signed up for this appointment
* Max
  + The maximum number of students allowed at the appointment

# PArt 5: Presentation Video

Here is a link to my presentation video:

<https://youtu.be/Y0dvqnA-H2o>

# Part 6: Page Documentation

## How to read the Documentation

This is the format that I used to document this website:

Purpose: This section is a brief, one-sentence summary of this web page’s purpose in the scheme of the entire website.

UI: This is a true or false value, simply stating whether the web page has a user interface or not. Web pages that have “True” for this value interact with the end user in some way, through a form or by displaying information. Web pages that have “False” for this value do things “behind the scenes” – that is, performing SQL queries, checking $\_POST variable values, and redirecting the user

Important Variables Created: Any important variables that the page creates. Important variables are $\_POST variables, $\_SESSION variables, and any variables that contribute to core functionality of the web page. Variables that aren’t considered important are any temporary variables, any variables that store SQL queries, and any other variables that could fall into similar categories. These can also be found in the Dia diagram.

Refer to the “Variable Descriptions” category below to see what the variables do.

Important Variables Used: Any important variables that the page uses that come from other pages. These are only $\_POST and $\_SESSION variables. These can also be found in the Dia diagram.

Refer to the “Variable Descriptions” category below to see what the variables do.

Level of Editing: The amount of this website that I had to edit to fit the functionality of the rubric. These options are:   
“Comments Only” – No changes were made besides commenting  
“Minimal” – I fixed a few lines of code  
“Moderate” – I had to alter a portion of the page  
“Major” – I had to alter a majority of the page

Description: This is a description of the web page in detail.

Changes I Made: This covers all of the changes I made to this page from the original provided version. To keep this document readable, this does not include commenting, as this change was made in every single page.

Changes that were required by the rubric are **not** listed in this section. To see if a page was affected by one of these changes, please **refer to their respective sections above - Part 2 and Part 3.**

All changes that affect external files (such as CommonMethods.php and standard.css) that affect more than one web page will **not** be listed in the changes section for the altered pages. Instead, the change will listed in the section of the documentation for the **external file**. (i.e. If every page had an issue connecting to CommonMethods.php, I would put that change in the section for CommonMethods.php). This decision was made to reduce the size of this documentation.

The format for each change is as follows:

### NAME OF CODE CHANGE

Change description goes here

## Variable Descriptions

This is a “master list” of all the important variables in the website. Important variables are defined as $\_POST variables, $\_SESSION variables, and any variables that contribute to core functionality of the website. Some variables may be used in different parts of the project (Student, Admin) and have different functionality for each page. All functionalities will be listed below.

* $conn
  + Both: Stores the connection to a MySQL database
* $debug
  + Both: Store whether or not CommonMethods should run in debug mode
* $\_POST[“UserN”]/$\_SESSION[“UserN”]/$\_SESSION[“AdvUN”]
  + Admin: The username of the current advisor logged in
* $\_POST[“PassW”]/$\_SESSION[“PassW”]/$\_SESSION[“AdvPW”]
  + Admin: The password of the current advisor logged in
* $\_SESSION[“UserVal”]
  + Admin: Whether or not the username & password combination the user tried is a valid advisor log in. If this is set to true, then the login was invalid. If the login is valid, this variable shouldn’t be initialized at all.
* $\_SESSION[“FirstN”]/$\_POST[“firstN”]/$\_SESSION[“AdvF”]
  + Admin: The advisor’s first name
  + Student: The student’s first name
* $\_POST[“lastN”]/$\_SESSION[“AdvL”]
  + Admin: The advisor’s last name
  + Student: The student’s first name
* $\_POST[“ConfP”]
  + Admin: This variable is used when creating an advisor account, to confirm that the password is correct. To elaborate, the page AdminCreateNewAdv.php has two password fields. One is for the admin to put in the new account’s password, the other is to “confirm” it, by making sure they put the same password in again. This is done to make sure that the password isn’t saved with a typo in it.
* $\_POST[“OfficeL”]/$\_SESSION[“OfficeL”]
  + Admin: The office location of the administrator
  + Student: The office location of the administrator
* $\_SESSION[“studID”]
  + Admin: The student’s UMBC ID
  + Student: The student’s UMBC ID
* $\_POST[“next”]
  + This variable is used many times throughout the web site, and it holds a different meaning every time that it is used. Generally, it stores a single value to pass to the next page to determine how to be redirected. To see exactly what $\_POST[“next”] is used for on a given page, refer to the Dia diagram.
* $\_SESSION[“advisor”]
  + Admin: The name of the advisor for a given appointment
* $\_POST[“Date”]/$\_POST[“date”]
  + Admin: The date of an appointment
* $\_POST[“time”]/$\_SESSION[“appTime”]
  + Admin: The time of an appointment
  + Student: The time of an appointment
* $\_POST[“major”]
  + Admin: The type of major that this appointment is available to. For example, if the value is CMSC, then the appointment is only available to Computer Science majors.
  + Student: The major of the student signing up for the appointment
* $\_POST[“repeat”]
  + Admin: What days of the week this appointment should repeat for. For example, a given appointment may be available Mondays and Wednesdays
* $\_POST[“stepper”]
  + Admin: How many weeks an appointment should repeat for. For example, an appointment that’s available 8am Mondays and Wednesdays should be available for 5 weeks straight.
* $\_POST[“stepper1”]
  + Admin: The maximum number of students allowed at a group advising appointment.
* $\_POST[“type”]
  + Admin: The type of appointment being referred to. The options are Individual or Group.
  + Student: The type of appointment being referred to. The options are individual or Group.
* $\_POST[“GroupApp”]/$\_SESSION[“GroupApp”]
  + Admin*:* The group appointment to be edited.
* $\_SESSION[“Delete”]
  + Admin: Whether or not the selected appointment is to be deleted or edited.
* $\_SESSION[“advisor”]/$\_POST[“advisor”]
  + Admin: The name of the advisor (first and last). Could also be “Group”
  + Student: The name of the advisor (first and last). Could also be “Group”
* $POST[“IndApp”]
  + Admin: The individual appointment to be edited.
* $\_POST[“studLN”]
  + Admin: The last name of a student
* $\_POST[“filter”]
  + Admin: Whether an appointment is open or closed. As in, whether or not the appointment can be taken by a student or whether it is full
* $\_SESSION[“PassCon”]
  + Admin: If this is true, then the passwords didn’t match when trying to create a new advisor.
* $\_POST[“email”]
  + Student: The student’s e-mail address
* $\_POST[“selection”]
  + Student: Used to determine what action the student user would like to do
* $\_SESSION[“studExist”]
  + Student: Whether or not the student user exists in the db
* $advisorName/$advName
  + Student: The advisor’s first and last name. For Group appointments, this simply says “Group”
* $\_POST[“finish”]
  + Student: Whether the user cancels the appointment while creating it or not. This is set to true only if the user cancels while creating an appointment, on pages like 08StudSelectTime.php
* $\_SESSION[“resch”]
  + Whether the student is overwriting their current appointment
* $\_SESSION[“status”]
  + The status of the appointment. This lets the next page know whether or not the appointment was made successfully, if one was overwritten, or if it didn’t go through.

## Standard.css

Purpose: Provide style to the web site

UI: N/A

Important Variables Created: N/A

Important Variables Used: N/A

Level of Editing: Minimal

Description: The stylesheet for the website

### Updated Button.LARGE

When flipping StudentAdminSignIn.html to external CSS, I realized that standard.css’s styling for button.large didn’t actually make large buttons much larger than a standard button. So, I updated these lines:

.button.large {

font-size: 16px;

height: 32px;

padding: 0 16px;

position: center;

}

To this:

.button.large {

**font-size: 32px;**

**height: 64px;**

padding: 0 16px;

position: center;

}

The values that I updated button.large with are the same values from StudentAdminSignIn.html’s inline CSS. I felt this change was necessary to make the transition from inline CSS to standard.css cause as few changes to the existing user interface as possible.

## CommonMethods.php

Purpose: Provides functions to the various web pages to connect to the MySQL database

UI: No

Important Variables Created:  
$conn  
$debug

Important Variables Used: None

Level of Editing: Moderate

Description: This page stores a class called “Common” with a collection of functions for connecting and accessing the MySQL database for this project. It helps to centralize the project, because if a single part needed to be changed (i.e. the database’s server name) you only have to update it in CommonMethods.php instead of all throughout the web page.

### Fixed all calls to CommonMethods.php

Originally, this website had the following line of code throughout its pages:

include('../CommonMethods.php');

Which caused problems because CommonMethods.php was in the same directory as the files calling it, so I changed all of the calls to this:

include('CommonMethods.php');

Which works fine.

Files that were affected by this fix, in no particular order:

* 02StudHome.php
* 09StudSearchApp.php
* 06StudEditInfo.php
* 11StudSearchResult.php
* StudProcessEdit.php
* 07StudSelectAdvisor.php
* 08StudSelectTime.php
* 10StudConfirmSch.php
* StudProcessSch.php
* 04StudViewApp.php
* 05StudCancelApp.php
* AdminUI.php
* AdminSearchApp.php
* AdminProcessSignIn.php
* AdminConfirmScheIndApp.php
* AdminConfirmScheGroupApp.php
* AdminConfirmEditInd.php
* AdminPrintResults.php
* AdminEditGroup.php
* AdminEditInd.php
* AdminCreateNew.php
* AdminConfirmEditGroup.php
* AdminProceedEditGroup.php
* AdminSearchResults.php
* StudProcessCancel.php

### Updated Database information

The original CommonMethods.php had values that pointed to a database I don’t have access to (I’m assuming, the actual advising database?) so I changed the values to my studentmaria database and login info. This was necessary because otherwise I couldn’t test any changes I make to this website.

## index.php

Purpose: Redirects the user to the student/admin selection page

UI: No

Important Variables Created: None

Important Variables Used: None

Level of Editing: Major

Description: Redirects the user to the Student/Admin selection page.

### Changed redirection target to StudentADminSignIn.html

Originally, the page redirected you to the student log in. I uncommented the line:

header(“Location: StudentAdminSignIn.html”);

and commented this line:

header(“Location: 01StudSignIn.html”);

Now the first thing the user sees is the page that lets them select if they’re an admin or not. I don’t see the harm in doing this, because if the end user is a student, they can’t access the administrator side without a valid login anyways. This only makes it easier for the advisors to access their side of the website. Not to mention, it makes it easier for the programmers to test the site.

## StudentAdminSignIn.html

Purpose: Acts as a portal, allowing the user to access the student and admin sides of this website

UI: Yes

Important Variables Created: None

Important Variables Used: None

Level of Editing: Major

Description: A page that separates the admin and student sides of the website. There is a button to sign in as either. Clicking one sends you to the appropriate page. Note that admins can’t sign in as students and vice versa, so any attempt to “sneak in to the admin side” shouldn’t work without the correct log in information.

### Included Greeting

I felt that the page was a little bare, so I chose to add a greeting message that reads “Welcome to Advising” above the buttons for this page. I justify this by making it more enjoyable for the end user.

### Removed Unnessecary Div Tags

The following <div> tags were included in the web page:

<div class =”field”>  
</div>

They provided absolutely no functionality to the page and took up unnecessary space, so I removed them.

### Added Center Tags to the login buttons

The inline CSS that was previously in this page had a “dirty” way of centering the buttons. Instead of actually centering them, the inline CSS added a margin to the left of the buttons that pushed them towards the center.

Since this is bad practice, I felt that I had to change this. I removed the inline CSS and replaced it with standard.css, so I couldn’t make a page-specific fix. To not affect the entire website, I wrapped the buttons in <center> tags to just fix this instance of centering buttons. There may be other, more efficient ways to do this by altering the CSS, but to save time and code complexity, I felt this was the best option.

### Changed the destination of Admin Sign In BUTTON

Originally, clicking the button that reads “Sign in as an Admin” would send you to AdminSignIn.html. However, I have removed that page from the website, since it is a less useful copy of AdminSignIn.php. Since that page no longer exists, I changed the destination of the “Sign in as an admin” button to AdminSignIn.php

## AdminSignIn.HTML

**This page was removed from the site**

This page was almost identical to AdminSignIn.php, the only difference is that AdminSignIn.php has a way to handle incorrect password combinations. In fact, if you get one login attempt wrong, you get redirected to AdminSignIn.php anyways. So to reduce redundancy, I removed AdminSignIn.html. This is important to not only save space, but if someone wanted to add a feature to the sign in page in the future, they’d have to edit two pages instead of just one.

## AdminSignIn.php

Purpose: Provides functions to the various web pages to connect to the MySQL database.

UI: Yes

Important Variables Created:  
$\_POST[“UserN”]

$\_POST[“PassW”]

Important Variables Used:

$\_SESSION[“UserVal”]

Level of Editing: Minimal

Description: This page is a user interface for logging in as an advisor. Once a proper login is passed, it will let the user into the advisor UI. The page will also inform the user if their login is incorrect (i.e. not the same username and password combination as stored in the Proj2Advisors database).

## AdminProcessSignIn.php

Purpose: Checks whether the login information is valid or not.

UI: No

Important Variables Created:  
$\_SESSION[“UserN”]  
$\_SESSION[“PassW”]  
$\_SESSION[“UserVal”]  
$\_SESSION[“FirstN”]

Important Variables Used:

$\_POST[“UserN”]  
$\_POST[“PassW”]

Level of Editing: Minimal

Description: This page simply checks to see if the username and password match an advisor in the database Proj2Advisors. If they do not, then the user is redirected back to AdminSignIn.php with a warning message that they put in the wrong information.

### Created $\_SESSION[“FirstN”]

In the next page, AdminUI.php, there is an unnecessary call to the MySQL database *just* to get the advisor’s first name, so I decided to store the advisor’s first name as a session variable on this page and remove the call to the database in AdminUI.php. This was simple to add, because we were already pulling an entire row from Proj2Advisors that already has the advisor’s name. The reason I chose to do this is so I could reduce the number of SQL queries to the database.

## AdminUI.PHP

Purpose: The central interface for advisors to access and manipulate their appointments

UI: Yes

Important Variables Created: None

Important Variables Used:

$\_SESSION[“FirstN”]

$\_SESSION[“UserN”]

Level of Editing: Moderate

Description: This is the page that has all of the commands for an advisor, such as viewing their schedule, creating an appointment, cancelling an appointment, etc.

### Removed Unnessecary Div Tags

The following <div> tags were included in the web page:

<div class =”field”>  
</div>

They provided absolutely no functionality to the page and took up unnecessary space, so I removed them.

## Logout.php

Purpose: Ends a session with a user

UI: No

Important Variables Created: None

Important Variables Used:

$\_SESSION[“studID”]

Level of Editing: Comments Only

Description: This page destroys all session variables, and logs out a user so that they need to log in again if they wish to access the website. This page is accessed by both students and advisors.

## AdminProcessUI.php

Purpose: Redirects the user to one of the advisor pages

UI: No

Important Variables Created: None

Important Variables Used:

$\_POST[“next”]

Level of Editing: Comments Only

Description: This page redirects the user based on what button they chose in AdminUI.php.

## AdminScheduleApp.php

Purpose: Lets the advisor schedule an appointment

UI: Yes

Important Variables Created:  
$\_POST[“next”]

Important Variables Used: None

Level of Editing: Comments Only

Description: This page lets the advisor select which type of appointment they want to schedule: a Group Appt. or an Individual Appt.

## AdminProcessSchedule.php

Purpose: Redirect the user based on their selection in AdminScheduleApp.php

UI: No

Important Variables Created:

$\_SESSION[“advisor”] (ONLY if the previous selection was “Group”. Otherwise, this is created later)

Important Variables Used: None

Level of Editing: Comments Only

Description: Redirects the user to the group and individual appointment scheduling pages.

## AdminScheduleInd.php

Purpose: Lets the advisor create an individual advising appointment

UI: Yes

Important Variables Created:  
$\_POST[“Date”]  
$\_POST[“time”]  
$\_POST[“major”]  
$\_POST[“repeat”]  
$\_POST[“stepper”]

Important Variables Used: None

Level of Editing: Minimal

Description: A form for an advisor to create an appointment. This page must be updated every semester to change the min availability date for appointments to be scheduled.

### Removed Max Appointment Date

Originally, the following code prevented advisors from creating appointments past October 30, 2015:

<input id="Date" type="date" name="Date" placeholder="mm/dd/yyyy" min="2015-08-01" max="2015-10-30" required autofocus> (mm/dd/yyyy)

I understand that the student side of the web site shouldn’t be allowed to make appointments past the current semester, but the advisor side is an administrator side and should be allowed to make appointments on any day they want to. By removing the maximum allowed date for appointments, it saves the webmaster from having to update the page every semester, and frees the advisor from having difficulties making appointments past the set date.

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code, and is not present in its sister page AdminScheduleGroup.php. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## AdminConfirmScheIndApp.php

Purpose: Shows which appointments were created, and which already existed.

UI: Yes

Important Variables Created: None

Important Variables Used:

$\_POST[“Date”]  
$\_POST[“time”]  
$\_POST[“major”]  
$\_POST[“repeat”]  
$\_POST[“stepper”]  
$\_SESSION[“UserN”]  
$\_SESSION[“PassW”]

Level of Editing: Major

Description: This page lists all of the appointments that the advisor tried to make on the previous page (AdminScheduleInd.php). If an appointment was successfully created, then it was listed normally. If an appointment couldn’t be created because it already existed or had some other conflict, it is listed with two red exclamation points next to it.

## AdminScheduleGroup.php

Purpose: Lets the advisor create a group advising appointment

UI: Yes

Important Variables Created:  
$\_POST[“Date”]  
$\_POST[“time”]  
$\_POST[“major”]  
$\_POST[“repeat”]  
$\_POST[“stepper”]  
$\_POST[“stepper1”]

Important Variables Used: None

Level of Editing: Minimal

Description: A form for an advisor to create an appointment. This page must be updated every semester to change the min availability date for appointments to be scheduled.

### Removed Max Appointment Date

Originally, the following code prevented advisors from creating appointments past October 30, 2015:

<input id="Date" type="date" name="Date" placeholder="mm/dd/yyyy" min="2015-08-01" max="2015-10-30" required autofocus> (mm/dd/yyyy)

I understand that the student side of the web site shouldn’t be allowed to make appointments past the current semester, but the advisor side is an administrator side and should be allowed to make appointments on any day they want to. By removing the maximum allowed date for appointments, it saves the webmaster from having to update the page every semester, and frees the advisor from having difficulties making appointments past the set date.

## AdminConfirmScheGroupApp.php

Purpose: Shows which appointments were created, and which already existed.

UI: Yes

Important Variables Created: None

Important Variables Used:

$\_POST[“Date”]  
$\_POST[“time”]  
$\_POST[“major”]  
$\_POST[“repeat”]  
$\_POST[“stepper”]  
$\_POST[“stepper1”]  
$\_SESSION[“UserN”]  
$\_SESSION[“PassW”]

Level of Editing: Major

Description: This page lists all of the appointments that the advisor tried to make on the previous page (AdminScheduleInd.php). If an appointment was successfully created, then it was listed normally. If an appointment couldn’t be created because it already existed or had some other conflict, it is listed with two red exclamation points next to it.

## AdminPrintSchedule.php

Purpose: Displays all appointments on a given day

UI: Yes

Important Variables Created:  
$\_POST[“type”]  
$\_POST[“date”]

Important Variables Used: None

Level of Editing: Minimal

Description: This page lets an advisor view all of their appointments for a given day. They can narrow the search results by separating Group and Individual appointments

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## AdminPrintResults.php

Purpose: Shows all appointments on a given day for an advisor

UI: Yes

Important Variables Created: None

Important Variables Used:

$\_POST[“type”]

$\_POST[“date”]

$\_SESSION[“UserN”]

Level of Editing: Minimal

Description: This page shows the results of AdminPrintSchedule.php, based on the criteria that the user specified. It lists all of the appointments the currently logged in advisor has on any given day, and what students are currently signed up for the appointments.

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## AdminEditApp.php

Purpose: Allows the user to choose what type of appointment they want to edit

UI: Yes

Important Variables Created:  
$\_POST[“next”]

Important Variables Used: None

Level of Editing: Minimal

Description: This is the first step in editing an appointment. On this page, the advisor chooses whether they want to edit one of their group appointments or one of their individual appointments.

## AdminProcessEdit.php

Purpose: Redirects the user based on what type of advising they want to alter

UI: No

Important Variables Created: None

Important Variables Used:   
$\_POST[“next”]

Level of Editing: Comments Only

Description: This page simply redirects the user to the page that lets them edit group or individual appointments, based on their selection in AdminEditApp.php

## AdminEditGroup.php

Purpose: Allows the user to alter a group appointment

UI: Yes

Important Variables Created:  
$\_POST[“GroupApp”]

Important Variables Used: None

Level of Editing: Minimal

Description: Lists all of the group appointments in Proj2Appointments. When the user selects the radio button next to the appointment they want to alter, they can then select between two buttons. One of the buttons allows the user to delete the appointment from the system, while the other lets the user edit the details of the appointment.

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

### Removed Premature initialization of $\_SESSION[“DELETE”]

At the top of the page beneath session\_start(), the page initialized $\_SESSION[“Delete”] to false, without referencing it again in the page. This is unnecessary, because AdminProcessEditGroup.php does the same thing at the top of its page. So to reduce redundancy, I deleted this line.

## AdminProcessEditGroup.php

Purpose: Redirects the user to delete or edit the selected appointment

UI: No

Important Variables Created:  
$\_SESSION[“Delete”]  
$\_SESSION[“GroupApp”]

Important Variables Used:  
$\_SESSION[“advisor”]  
$\_POST[“next”]

Level of Editing: Comments Only

Description: Redirects the advisor based on whatever they chose in AdminEditGroup.php. Sets up session variables for either page.

## AdminProceedEditGroup.php

Purpose: Allows the user to edit the details of a group appointment

UI: Yes

Important Variables Created: None

Important Variables Used:   
$\_SESSION[“GroupApp”]

Level of Editing: Minimal

Description: Lets the advisor edit the details of the appointment they selected in AdminEditGroup.php

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## AdminConfirmEditGroup.php

Purpose: Allows the user to edit the details of a group appointment

UI: Yes

Important Variables Created: None

Important Variables Used:   
$\_SESSION[“GroupApp”]  
$\_SESSION[“Delete”]

Level of Editing: Minimal

Description: Lets the advisor edit the details of the appointment they selected in AdminEditGroup.php

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## AdminEditInd.php

Purpose: Lists all appointments you can delete

UI: Yes

Important Variables Created:  
$\_POST[“IndApp”]

Important Variables Used: None

Level of Editing: Minimal

Description: This page lists all the appointments for the advisor that aren’t group appointments.

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## AdminConfirmEditInd.php

Purpose: Lists all appointments you can delete

UI: Yes

Important Variables Created:

Important Variables Used:   
$\_POST[“IndApp”]

Level of Editing: Minimal

Description: Shows the deleted appointment information

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## AdmiNSEARCHAPP.php

Purpose: Allows the advisor to search for their appointments

UI: Yes

Important Variables Created:  
$\_POST[“date”]  
$\_POST[“time”]  
$\_POST[“advisor”]  
$\_POST[“studID”]  
$\_POST[“studLN”]  
$\_POST[“filter”]

Important Variables Used: None

Level of Editing: Minimal

Description: Provides an interface to the user that lets them search for appointments based on different criteria.

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## AdminSearchResults.php

Purpose: Shows the result of an advisor’s search for appointments

UI: Yes

Important Variables Created: None

Important Variables Used:   
$\_POST[“date”]  
$\_POST[“time”]  
$\_POST[“advisor”]  
$\_POST[“studID”]  
$\_POST[“studLN”]  
$\_POST[“filter”]

Level of Editing: Minimal

Description:

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## AdminCreateNewAdv.php

Purpose: Allows an advisor to create a new account for a new advisor

UI: Yes

Important Variables Created:  
$\_POST[“firstN”]  
$\_POSt[“lastN”]  
$\_POST[“UserN”]  
$\_POST[“PassW”]  
$\_POST[“ConfP”]  
$\_POST[“OfficeL”]

Important Variables Used: None

Level of Editing: Minimal

Description: Lets an existing advisor create a new advisor account.

### Removed Commented out Code

The following code was commented out in the head tag for the web page:

<script type="text/javascript">

// window.onload = function () {

// document.getElementById("PassW").onchange = validatePassword;

// document.getElementById("ConfP").onchange = validatePassword;

// }

// function validatePassword(){

// var pass2=document.getElementById("ConfP").value;

// var pass1=document.getElementById("PassW").value;

// if(pass1!=pass2)

// document.getElementById("ConfP").setCustomValidity("Passwords Don't Match");

// else

// document.getElementById("PassW").setCustomValidity('');

// //empty string means no validation error

// }

// </script>

It looks like there was a time where the web page checked that your passwords match BEFORE going to AdminProcessCreateNew.php but was taken out. Nevertheless, I removed this because it was wasting space.

## AdminProcessCreateNEW.php

Purpose: Prepares to create a new advisor account

UI: No

Important Variables Created:   
$\_SESSION[“AdvF”]  
$\_SESSION[“AdvL”]  
$\_SESSION[“AdvUN”]  
$\_SESSION[“AdvPW”]  
$\_SESSION[“OfficeL”]  
$\_SESSION[“PassCon”]

Important Variables Used:  
$\_POST[“firstN”]  
$\_POST[“lastN”]  
$\_POST[“UserN”]  
$\_POST[“PassW”]  
$\_POST[“ConfP”]  
$\_POST[“OfficeL”]

Level of Editing: Comments Only

Description: If the passwords the admin provided don’t match, then send the user back to AdminCreateNewAdv.php page. If they do match, send them to AdminCreateNew.php

## AdminCreateNew.php

Purpose: Creates the new advisor in the database

UI: Yes

Important Variables Created: None

Important Variables Used:   
$\_SESSION[“AdvF”]  
$\_SESSION[“AdvL”]  
$\_SESSION[“AdvUN”]  
$\_SESSION[“AdvPW”]  
$\_SESSION[“OfficeL”]

Level of Editing: Minimal

Description: Displays the advisor that was just created, or lets the user know that the advisor exists already.

### Removed Unnecessary javascript code

Towards the top of the page, there was a small chunk of javascript code that seemed to print the value of a user-defined portion of the web page. This seems to be leftover debug code. Since removing it affects nothing, and no pages I could find call this function, it has been removed. These are the lines that were removed:

<script type="text/javascript">

function saveValue(target){

var stepVal = document.getElementById(target).value;

alert("Value: " + stepVal);

}

</script>

## 01StudSignIn.HTML

Purpose: The log in page for students to sign up for advising appointments

UI: Yes

Important Variables Created:  
$\_POST[“firstN”]  
$\_POST[“lastN”]  
$\_POST[“studID”]  
$\_POST[“email”]  
$\_POST[“major”]

Important Variables Used: None

Level of Editing: Minimal

Description: The interface for users to log in as students.

### Removed Majors from the drop-down box that were commented out

Non-STEM majors were commented out in the drop down box. Since they were taking up space, I removed them.

## StudProcessSignIn.php

Purpose: Prepares the student information for the next page

UI: No

Important Variables Created:  
$\_SESSION[“firstN”]  
$\_SESSION[“lastN”]  
$\_SESSION[“studID”]  
$\_SESSION[“email”]  
$\_SESSION[“major”]

Important Variables Used:  
$\_POST[“firstN”]  
$\_POST[“lastN”]  
$\_POST[“studID”]  
$\_POST[“email”]  
$\_POST[“major”]

Level of Editing: Comments Only

Description: Converts the post variables from 01StudSignIn.html to session variables, then redirects to 02StudHome.php

## 02StudHome.php

Purpose: The main user interface for the user

UI: Yes

Important Variables Created:  
$\_SESSION[“studExist”]  
$\_POST[“selection”]

Important Variables Used:   
$\_SESSION[“firstN”]  
$\_SESSION[“studID”]

Level of Editing: Minimal

Description: This page has buttons that give the user different options to schedule and edit advising appointments.

## StudProcessHome.php

Purpose: Redirects the user based on what they chose in 02StudHome.php

UI: No

Important Variables Created:  
$\_SESSION[“resch”]

Important Variables Used:   
$\_SESSION[“selection”]

Level of Editing: Comments Only

Description: Forwards the user to different parts of the student site based on their selection to 02StudHome.php

## 03StudSelectType.php

Purpose: Page to choose the type of advising the user wants

UI: Yes

Important Variables Created:  
$\_POST[“type”]

Important Variables Used: None

Level of Editing: Moderate

Description: This page lets the user decide whether they want Individual or Group advising

### Added Center Tags to the Buttons

Originally, there were two buttons: “Individual” and “Group”, one on each side of the page. Since I had to add a third button (See Part 3) I decided to stack them vertically and center them so they looked okay.

## StudProcessType.php

Purpose: Redirects the user based on the type of advising they’d like

UI: No

Important Variables Created:  
$\_SESSION[“advisor”]  
$\_SESSION[“appTime”]

Important Variables Used:   
$\_POST[“type”]

Level of Editing: Major

Description: Sends the user to the appropriate page for making an appointment

### referenced and used CommonMethods.php

To implement the “Next Available” option as required by the rubric (For details, refer to Part 3 of the documentation) I had to add CommonMethods.php and connect to the database.

## 07StudSelectAdvisor.php

Purpose: Lets the user choose which advisor they want to meet with

UI: Yes

Important Variables Created:  
$\_POST[“advisor”]

Important Variables Used: None

Level of Editing: Minimal

Description: This page lets the user choose which advisor they want to meet with. This page is only visited for individual appointments, since group appointments have multiple advisors.

## 08STudSelectTime.php

Purpose: Lets the user select a time for their appointment

UI: Yes

Important Variables Created:  
$\_SESSION[“appTime”]  
$\_advisorName

Important Variables Used:  
$\_POST[“advisor”]  
$\_SESSION[“adivsor”]  
$\_SESSION[“major”]

Level of Editing: Minimal

Description: Provides a list of available appointments based on their times that corresponds with the advisor/type of advising that the student prefers.

## 10StudConfirmSch.PHP

Purpose: Makes sure the user wants the appointment

UI: Yes

Important Variables Created:  
$\_POST[“finish”]  
$\_SESSION[“appTime”]

Important Variables Used:   
$\_POST[“appTime”]  
$\_POST[“firstN”]  
$\_POST[“lastN”]  
$\_POST[“studID”]  
$\_POST[“major”]  
$\_POST[“email”]  
$\_POST[“resch”]  
$\_POST[“advisor”]

Level of Editing: Minimal

Description: Double checks that the user wants to make the appointment they set up in the previous pages by displaying the information and providing a chance to cancel. If they already have an appointment, it notifies them that they’re overwriting it and shows them the details of it.

### Added a NULL GUARD

Since there’s now a third option to create an appointment (Next Available, for details refer to Part 3) I had to add a check to see if $\_POST[“appTime”] is set or not before overwriting the value of $\_SESSION[“appTime”] because if next available is selected, $\_SESSION[“appTime”] holds the value of the next available appointment and should not be overwritten.

## StudPRocessSch.php

Purpose: Determine whether the appointment can be made or not

UI: No

Important Variables Created:  
$\_SESSION[“status”]

Important Variables Used:  
$\_POST[“finish”]  
$\_SESSION[“firstN”]  
$\_SESSION[“lastN”]  
$\_SESSION[“studID”]  
$\_SESSION[“major”]  
$\_SESSION[“email”]  
$\_SESSION[“advisor”]  
$\_SESSION[“appTime”]  
$\_SESSION[“studExist”]

Level of Editing: Minimal

Description: This page attempts to create an appointment for the student. If the student already had an appointment, it removes that one first before creating the new one. If someone else registered for the same timeslot while the student was confirming their appointment, it informs the user that the timeslot isn’t available.

### Removed a DEBUG LINE

There was a line meant for debugging only that was causing alot of problems. This is the line:

if(debug) { echo("Advisor -> $advisor<br>\n"); }

The issue it caused is that it would send a header, preventing the web page from redirecting where it needed to. This is the error it would cause:



It would also run despite the fact that I was not in debug mode. I removed this code to make the program work properly, and it provides a nearly useless function for debugging anyways.

## 13StudDenied.php

Purpose: Informs the student that the appointment is no longer available

UI: Yes

Important Variables Created: None

Important Variables Used: None

Level of Editing: Minimal

Description: This page simply tells the user that the appointment is no longer available.

## 12StudExit.php

Purpose: This page informs the user the status of the appointment after they alter it

UI: Yes

Important Variables Created: None

Important Variables Used:   
$\_SESSION[“resch”]  
$\_SESSION[“status”]

Level of Editing: Minimal

Description: Displays a message to the user, letting them know that they’ve reached their final step with what they’re doing to the appointment

## 04StudViewApp.php

Purpose: Let’s the student view their appointment

UI: Yes

Important Variables Created:  
$advisorName

Important Variables Used:   
$\_SESSION[“studID”]

Level of Editing: Minimal

Description: Displays the student’s appointment information

## 05StudCancelApp.php

Purpose: Lets the student cancel their existing appointment

UI: Yes

Important Variables Created: None

Important Variables Used:   
$\_SESSION[“firstN”]  
$\_SESSION[“lastN”]  
$\_SESSION[“studID”]  
$\_SESSION[“major”]  
$\_SESSION[“email”]

Level of Editing: Minimal

Description: Displays the student’s appointment information, and confirms that the user wants to cancel their appointment.

## StudProcessCAncel.php

Purpose: Determines whether to cancel the student’s appointment

UI: No

Important Variables Created: None

Important Variables Used:  
$\_SESSION[“firstN”]  
$\_SESSION[“lastN”]  
$\_SESSION[“email”]  
$\_SESSION[“major”]  
$\_SESSION[“studID”]

Level of Editing: Comments Only

Description: If the user chose to cancel their appointment, this page will remove them from the appointment in the database. If the user chose to keep the appointment, nothing will happen.

## 09StudSearchApp.php

Purpose: Lets the user perform a search for different available appointments

UI: Yes

Important Variables Created:  
$\_POST[“date”]  
$\_POST[“time”]  
$\_POST[“advisor”]

Important Variables Used: None

Level of Editing: Minimal

Description: Lets the user look up appointments based on different criteria. You can’t select appointments from here, but you could look them up for planning purposes or for a friend.

## 11STudSearchResult.php

Purpose: Displays all appointments that match the user’s criteria

UI: Yes

Important Variables Created:  
$advName

Important Variables Used:  
$\_POST[“date”]  
$\_POST[“time”]  
$\_POST[“advisor”]

Level of Editing: Minimal

Description: Shows all appointments that match the user’s criteria from 09StudSearchApp.php. Note that you cannot schedule an appointment from this page.

### Removed commented Code

The following three lines were commented out at the top of the page:

//ini\_set('display\_errors','1');

//ini\_set('display\_startup\_errors','1');

//error\_reporting (E\_ALL);

They are probably left over from an older revision of this website. I removed them since they were taking up space.

## 06StudEditInfo.php

Purpose: Lets the student edit their information in the database

UI: Yes

Important Variables Created: None

Important Variables Used:  
$\_SESSION[“firstN”]  
$\_SESSION[“lastN”]  
$\_SESSION[“email”]  
$\_SESSION[“major”]  
$\_SESSION[“studID”]

Level of Editing: Major

Description: This page lets a student edit all of their information except for their student ID.

### Removed unused majors

The drop-down box that holds all possible majors had many commented out options that aren’t implemented yet. I decided to remove them since they’re unused until one day they are actually implemented.