
My CS Degree Plan

Web Tool

Ana Garcia Ramirez - December 13, 2019

Purpose	3
Development Environment	3
functions.php	4
The Functions	4
header.php	5
setup.php	6
index.php	7
signup.php	7
Checking for username availability	8
checkuser.php	10
login.php	10
Home Page	11
edit_lowerdiv.php	12
edit_core.php	16
edit_math.php	19
edit_free_elect.php	20
edit_sciences.php	21
edit_upperdiv.php	23
edit_techelect.php	25
The Advisor View	27
view_student.php	29

Purpose

Advising students every semester is a time consuming task. Students need to fill out their advising forms and meet their advisors. During this meeting, the student's advisor must manually fill out each student's paper copy of his or her degree plan, taking careful note of the semester and year the student took a specific course, how many times this student has taken this course and the grade earned.

My CS Degree Plan tool is a an electronic version of the degree plan paper copy used in the CS department for student advising. Students can create an account in the web site using their UTEP ID number, edit their degree plans and save them. Advisors can use a special log in account that will display a list of all students in the system. The advisor can then view a student's degree plan, edit it, and save it.

This tool is a web-based system written in PHP 7 to handle the server-side operations. It uses JavaScript, CSS, and HTML for the user interface, and it uses MySQL for database management. At the time this document was written, the CS Department web servers have the same version of PHP and MySQL as listed in the Development Environment section.

Development Environment

My CS Degree Plan was developed in a MacBook with the following specifications:

- Operating System: macOS Mojave Version 10.14.6.
- XAMPP 7.3.9 Rev. 0
- PHP 7.1.23
- MySQL (10.4.6-MariaDB)
- Apache Web Server

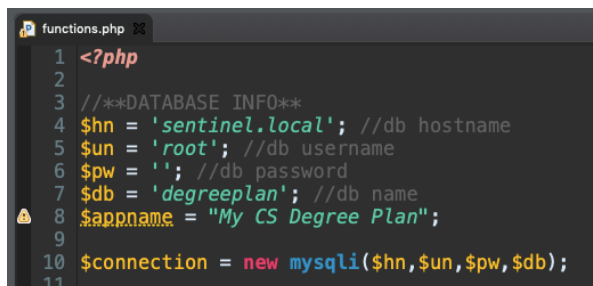
XAMPP is a native installer that installs Apache, PHP and MySQL. Since XAMPP 5.5.30 and 5.6.14 XAMPP uses MariaDB instead of MySQL. However, the commands and tools remain the same for both.

The repository where My CS Degree Plan is stored can be found here:

https://github.com/ajgarcia09/degree_plan

functions.php

The file *functions.php* is the file of main functions used across the project, such as printing the contents of the database and editing a student's degree plan. This file also has the database login details instead of using yet another separate file. It doesn't matter what the name of the database is **as long as it already exists and its credentials are listed** in the first few lines of *functions.php* as shown in Figure 1.

A screenshot of a code editor showing the first 11 lines of a file named 'functions.php'. The code is written in PHP and sets up database connection variables. Line 1: <?php. Line 2: (empty). Line 3: /**DATABASE INFO**. Line 4: \$hn = 'sentinel.local'; //db hostname. Line 5: \$un = 'root'; //db username. Line 6: \$pw = ''; //db password. Line 7: \$db = 'degreeplan'; //db name. Line 8: \$appname = "My CS Degree Plan";. Line 9: (empty). Line 10: \$connection = new mysqli(\$hn,\$un,\$pw,\$db);. Line 11: (empty).

```
1 <?php
2
3 /**DATABASE INFO**
4 $hn = 'sentinel.local'; //db hostname
5 $un = 'root'; //db username
6 $pw = ''; //db password
7 $db = 'degreeplan'; //db name
8 $appname = "My CS Degree Plan";
9
10 $connection = new mysqli($hn,$un,$pw,$db);
11
```

Figure 1: Database log in info

With correct values, line 10 will open a connection to MySQL and select the specified database. *\$appname* holds the name of the tool and it can be edited here.

The Functions

The project uses the following main functions:

- *createTable*
 - Checks whether a table already exists. If it doesn't, it creates it.
- *queryMysql*
 - Issues a query to MySQL, printing an error message if it fails.
- *destroySession*
 - Destroys a PHP session and clears its data to log users out.
- *sanitizeString*
 - Removes potentially malicious code or tags from user input to prevent an SQL Injection.
- *get_firstname*
 - Retrieves a student's first name by querying the database against his or her ID number.
- *get_lastname*
 - Retrieves a student's last name by querying the database against his or her ID number.
- *show_form_edit_example*
 - Shows students an example of how to edit the tables in their degree plan.
- *print_all_students*
 - Custom printing function to display all students registered in the system.

-
- *print_table*
 - Generic function to print each table in the logged in student's homepage.
 - *print_table_for_edit*
 - Custom function to create an editable form in *lowerdiv*, *othermath*, and *upperdiv*.
 - *print_core_table_for_edit*
 - Custom function to create an editable form in *core*.
 - *print_free_elect_table_for_edit*
 - Custom function to create an editable form in *freeelect*.
 - *print_sciences_table_for_edit*
 - Custom function to create an editable form in *sciences*.
 - *print_techselect_table_for_edit*
 - Custom function to create an editable form in *techselect*.
 - *print_student_tables*
 - Prints all tables in a student's degree plan.

header.php

For uniformity, each page of the project needs to have access to the same set of features. I added these features in *header.php*, which is included by all other files and it includes *functions.php*. This means that only a single *require_once* is necessary in each file.

header.php starts by calling the function *session_start*. This function sets up a session that will keep track of certain values stored across different PHP files. When the session has started, the program checks whether the session variable *\$user* is currently assigned a value. If this is the case, then this means a user has logged in and the variable *\$loggedin* is set to TRUE.

Using the value of *\$loggedin*, an *if* block displays one of two sets of menus. The non-logged in menu offers options of Home, Sign up, and Log in, while the logged-in version offers full access to the project's features. Additionally, if a user is logged in, his or her ID number is appended in parentheses to the page title and places after the main heading. *\$user* can be freely referred to wherever displaying the ID number is necessary, because if no user is logged in, the *\$user* variable will be empty and will have no effect on the output.

The styling applied to the header file is in *styles.css* and includes creating a wide heading with an orange colored strip to highlight the app's title as well as turning the links in the header to rounded buttons.

setup.php

With both *functions.php* and *header.php* written, it's time to set up the MySQL tables both of these files will use. This is done in *setup.php*, **which should be typed and loaded into a web browser before calling up any other files**. Failing to do so will produce numerous MySQL errors and **deem the tool unusable**.

The following are the tables created and their columns:

- *students*
 - Username *user*(indexed), password *pass*, first name *firstname*, last name *lastname*
- *lowerdiv*
 - User *user*(indexed), Course Number *coursenum*, Course Name *coursename*, First attempt *one*, Second Attempt *two*, Third Attempt *three*, Grade *GR*, Hours *HR*.
- *core*
 - User *user*(indexed), Course Number *coursenum*, Course Name *coursename*, First attempt *one*, Second Attempt *two*, Third Attempt *three*, Grade *GR*, Hours *HR*.
- *othermath*
 - User *user*(indexed), Course Number *coursenum*, Course Name *coursename*, First attempt *one*, Second Attempt *two*, Third Attempt *three*, Grade *GR*, Hours *HR*.
- *freeelect*
 - User *user*(indexed), Course Number *coursenum*, Course Name *coursename*, First attempt *one*, Second Attempt *two*, Third Attempt *three*, Grade *GR*, Hours *HR*.
- *sciences*
 - User *user*(indexed), Course Number *coursenum*, Course Name *coursename*, First attempt *one*, Second Attempt *two*, Third Attempt *three*, Grade *GR*, Hours *HR*, Hidden Field, *hidden*.
- *upperdiv*
 - User *user*(indexed), Course Number *coursenum*, Course Name *coursename*, First attempt *one*, Second Attempt *two*, Third Attempt *three*, Grade *GR*, Hours *HR*.
- *techelect*
 - User *user*(indexed), Course Number *coursenum*, Course Name *coursename*, First attempt *one*, Second Attempt *two*, Third Attempt *three*, Grade *GR*, Hours *HR*, Hidden Field, *hidden*.

Because the function *createTable* first checks whether a table already exists, *setup.php* can safely be called multiple times without generating any duplicate tables nor other errors. The tables *sciences* and *techelect* have a hidden field column that will be used in the table insertions in MySQL (see the sections *edit_sciences.php* and *edit_techelect.php* for a thorough explanation).

index.php

This file is necessary to give the project a home page. All it does is display the simple welcome message shown in Figure 2.



Figure 2: index.php

signup.php

This module enables students to create an account to record and update their degree plans. It also sets up the database for each user. The ending HTML block is a simple form that allows a username, password, first name, and last name to be entered. The empty `` given the id of `info` is the destination of the Ajax call at the start of the program that checks whether a desired username is available and prints a message for the user (Figure 3).

The screenshot shows a web page titled "My CS Degree Plan (Guest)" in an orange header bar. Below the header, there are four buttons: "Home", "Sign Up", "Log in", and "Log out". A green message box states "⇒ You must be logged in to view this page." Below this, the form instructions are: "Please enter your UTEP ID number as your username" and "And enter the last 4 digits of your UTEP ID as your password." The form fields are: Username (8000400), Password (****), First Name (Tony), and Last Name (Stark). A green message next to the Username field says "✓ This username is available". A "Sign up" button is at the bottom.

Figure 3: signup.php

Checking for username availability

At the start of *signup.php* there is a block of JavaScript that starts with the function *checkUser*. This is called by the JavaScript *onBlur* event when focus is removed from the username field of the form. It sends the contents of the span item mentioned above to an empty string in case it previously had a value. A request is then made to the program *checkUser.php* which reports whether the username *user* is available. The result returned by the Ajax call is then placed into the span item. This message is shown in green text with a green checkmark next to the username field in Figure 3.

Upon the successful user creation of a user with the following credentials:

Username: 8000400

Password: 0400

First Name: Tony

Last Name: Stark

The program calls the following functions:

- *queryMysql*
 - Adds the obtained user credentials in the students table (Figure 4).

```
MariaDB [degreeplan]> SELECT * FROM students;
+-----+-----+-----+-----+
| user | pass | firstname | lastname |
+-----+-----+-----+-----+
| starlord | 1914 | NULL | NULL |
| 8000400 | 0400 | Tony | Stark |
+-----+-----+-----+-----+
2 rows in set (0.002 sec)
```

Figure 4: students table after student signup

- *setup_lowerdiv*
 - Inserts the predefined Lower Division Requirements values into the *lowerdiv* table (Figure 5).

```
MariaDB [degreeplan]> SELECT * FROM lowerdiv;
+-----+-----+-----+-----+-----+-----+-----+-----+
| user | coursenum | coursename | one | two | three | GR | HR |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 8000400 | CS 1401+ | Intro. to Computer Science | NULL | NULL | NULL | NULL | 4 |
| 8000400 | CS 2401+ | Elem. Data Struct./Algorithms | NULL | NULL | NULL | NULL | 4 |
| 8000400 | MATH 2300+ | Discrete Mathematics | NULL | NULL | NULL | NULL | 3 |
| 8000400 | CS 2302+ | Data Structures | NULL | NULL | NULL | NULL | 3 |
| 8000400 | EE 2369+ | Digital Systems Design I | NULL | NULL | NULL | NULL | 3 |
| 8000400 | EE 2169+ | Digital Systems Design I Lab | NULL | NULL | NULL | NULL | 1 |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.002 sec)
```

Figure 5: lowerdiv table after student signup

- *setup_core*
 - Inserts the predefined Core Curriculum values into the *core* table (Figure 6).

```
MariaDB [degreeplan]> SELECT * FROM core;
```

user	coursenum	coursename	one	two	three	GR	HR
8000400	RWS 1301+	Rhetoric and Composition 1	NULL	NULL	NULL	NULL	3
8000400	RWS 1302+	Rhetoric and Composition 2	NULL	NULL	NULL	NULL	4
8000400	MATH 1411+	Calculus I	NULL	NULL	NULL	NULL	3
8000400	L Phil & Cult+	NULL	NULL	NULL	NULL	NULL	3
8000400	Creative Arts+	NULL	NULL	NULL	NULL	NULL	3
8000400	Soc & BehSc+	NULL	NULL	NULL	NULL	NULL	3
8000400	Comp Area Opt1+	NULL	NULL	NULL	NULL	NULL	3
8000400	Comp Area Opt2+	NULL	NULL	NULL	NULL	NULL	3
8000400	HIST 1301+	History of U.S. to 1865	NULL	NULL	NULL	NULL	3
8000400	HIST 1302+	History of U.S. since 1865	NULL	NULL	NULL	NULL	3
8000400	POLS 2310+	Introduction to Politics	NULL	NULL	NULL	NULL	3
8000400	POLS 2311+	American Government & Politics	NULL	NULL	NULL	NULL	3

12 rows in set (0.000 sec)

Figure 6: *core* table after student signup

- *setup_thermath*
 - Inserts the predefined Other Required Mathematics Courses values into the *othermath* table (Figure 7).

```
MariaDB [degreeplan]> SELECT * FROM othermath;
```

user	coursenum	coursename	one	two	three	GR	HR
8000400	MATH 1312+	Calculus II	NULL	NULL	NULL	NULL	3
8000400	MATH 3323+	Matrix Algebra	NULL	NULL	NULL	NULL	3
8000400	MATH 4329	Numerical Analysis	NULL	NULL	NULL	NULL	3
8000400	STAT 3320	Probability & Statistics for CS	NULL	NULL	NULL	NULL	3

4 rows in set (0.000 sec)

Figure 7: *othermath* table after student signup

- *setup_freeelect*
 - Inserts the predefined Free Electives values into the *freeelect* table (Figure 8).

```
MariaDB [degreeplan]> SELECT * FROM freeelect;
```

user	coursenum	coursename	one	two	three	GR	HR
8000400	NULL	NULL	NULL	NULL	NULL	NULL	3

1 row in set (0.000 sec)

Figure 8: *freeelect* table after student signup

- *setup_sciences*
 - Inserts the predefined Life & Physical Sciences values into the *sciences* table (Figure 9).

```
MariaDB [degreeplan]> SELECT * FROM sciences;
```

user	coursenum	coursename	one	two	three	GR	HR	hidden
8000400	PHYS 2420	Introductory Mechanics	NULL	NULL	NULL	NULL	4	NULL
8000400	NULL	NULL	NULL	NULL	NULL	NULL	4	science1
8000400	NULL	NULL	NULL	NULL	NULL	NULL	4	science2

3 rows in set (0.000 sec)

Figure 9: *sciences* table after student signup

- *setup_upperdiv*
 - Inserts the predefined Upper Division Requirements values into the *upperdiv* table (Figure 10).

```
MariaDB [degreeplan]> SELECT * FROM upperdiv;
```

user	coursenum	coursename	one	two	three	GR	HR
8000400	CS 3195	Jr. Professional Orientation	NULL	NULL	NULL	NULL	1
8000400	CS 3331+	Adv. Object-Oriented Programming	NULL	NULL	NULL	NULL	3
8000400	CS 3350	Automata/Computabi/Formal Lang.	NULL	NULL	NULL	NULL	3
8000400	CS 3360	Design/Impl. Programming Languages	NULL	NULL	NULL	NULL	3
8000400	CS 3432+	Comp. Arch I:Comp/Org Design	NULL	NULL	NULL	NULL	4
8000400	CS 4310+	Software Eng: Requirements Eng.	NULL	NULL	NULL	NULL	3
8000400	CS 4311	Software Eng: Design & Implmnt.	NULL	NULL	NULL	NULL	3
8000400	CS 4375	Theory of Operating Systems	NULL	NULL	NULL	NULL	3

8 rows in set (0.000 sec)

Figure 10: upperdiv table after student signup

- *setup_techelect*
 - Inserts the predefined Technical Electives values into the *techelect* table (Figure 11).

```
MariaDB [degreeplan]> SELECT * FROM techelect;
```

user	coursenum	coursename	one	two	three	GR	HR	hidden
8000400	NULL	NULL	NULL	NULL	NULL	NULL	3	elect1
8000400	NULL	NULL	NULL	NULL	NULL	NULL	3	elect2
8000400	NULL	NULL	NULL	NULL	NULL	NULL	3	elect3
8000400	NULL	NULL	NULL	NULL	NULL	NULL	3	elect4
8000400	NULL	NULL	NULL	NULL	NULL	NULL	3	elect5

5 rows in set (0.000 sec)

Figure 11: techelect table after student signup

checkuser.php

This file looks up a username in the database and returns a string indicating whether it has already been taken. Since *checkuser.php* relies on the functions *sanitizeString* and *queryMysql*, the program includes the file *functions.php*.

If the `$_POST` variable *user* has a value, the function looks it up in the database and, depending on whether it exists as a username, outputs either "Sorry, this username is taken" next to an "x" (defined by the HTML entity `✘`) of "This username is available" next to a checkmark (defined by the HTML entity `✔`).

login.php

After a successful signup, the user is prompted to log in. The sign-up and login modules are separate from each other as of this release to not over complicate the code, but automatically logging in a newly created user could be easily implemented if desired.

login.php provides the code needed to let the students log in. This file is very similar to the sign-up page since it provides a simple HTML form and some basic error checking as well as using *sanitizeString* before querying the MySQL database.

The main takeaway of *login.php* is that upon successful verification of the username and password, the session variables *user* and *pass* are given the username and password values. **As long as the current session remains active, these variables will be accessible by all the programs in the project, allowing them to automatically provide access to logged-in users.** Figure 12 shows how *index.php* looks when loaded in a browser.

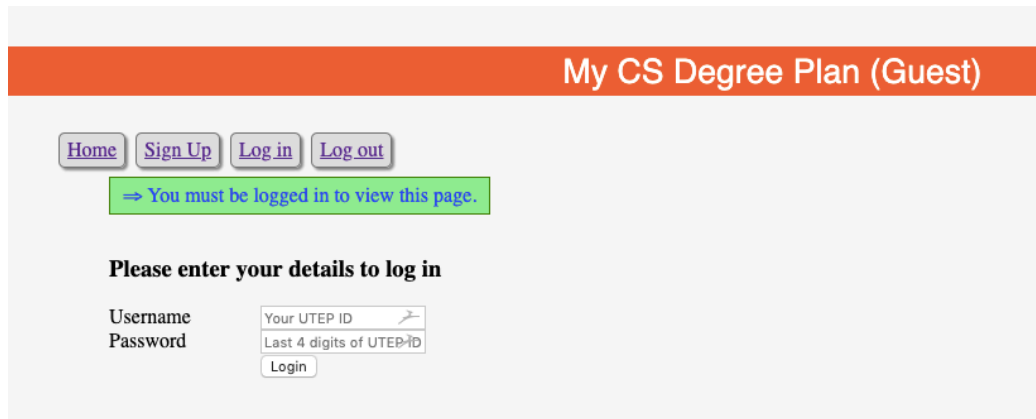
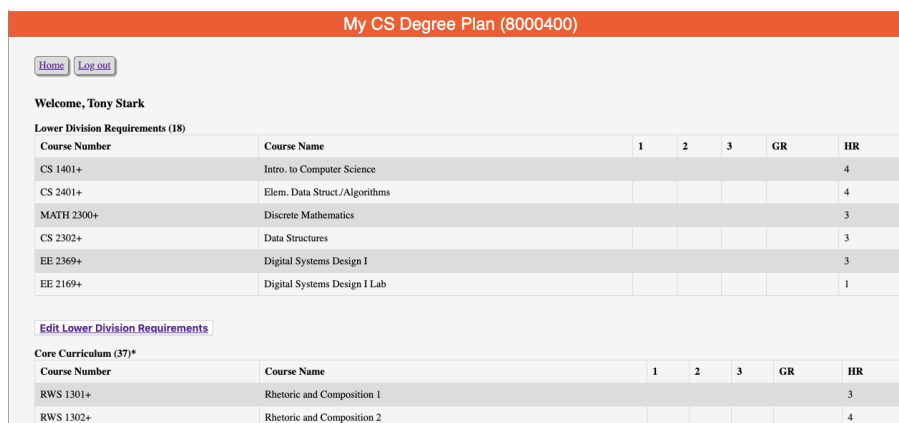


Figure 12: *login.php*

Home Page

When the user successfully signs in, he or she is greeted by a Welcome message next to the user's first name and last name. Also notice in Figure 13 that the words *Guest* in brackets have been replaced by the user's username.



My CS Degree Plan (8000400)						
Home Log out						
Welcome, Tony Stark						
Lower Division Requirements (18)						
Course Number	Course Name	1	2	3	GR	HR
CS 1401+	Intro. to Computer Science					4
CS 2401+	Elem. Data Struct./Algorithms					4
MATH 2300+	Discrete Mathematics					3
CS 2302+	Data Structures					3
EE 2369+	Digital Systems Design I					3
EE 2169+	Digital Systems Design I Lab					1
Edit Lower Division Requirements						
Core Curriculum (37)*						
Course Number	Course Name	1	2	3	GR	HR
RWS 1301+	Rhetoric and Composition 1					3
RWS 1302+	Rhetoric and Composition 2					4

Figure 13: *student homepage*

Every table in the database (except the *students* table) is printed in the home page in an HTML table. Each table is accompanied by an Edit Button, which allows the student to enter his or her course information.

edit_lowerdiv.php

edit_lowerdiv.php allows the user to enter course data into the *lowerdiv* table and update the user's entries in the *lowerdiv* table in the database. Figure 14 shows how the page looks when it is loaded on a browser after clicking the "Edit Lower Division Requirements" button.

My CS Degree Plan (8000400)

[Home](#) [Log out](#)

How to log your courses

Please use the following notation to log your 1st, 2nd, and/or 3rd attempt:
 Fall = FAXx
 Spring = SPxx
 Summer = SUxx
 Maymester = MYxx
 Wintermester = WLxx
 Where xx are the last 2 digits of the year you took the course.

Example: I took Intro to CS during Summer 2015

Course Number	Course Name	1	2	3	GR	HR
CS 1401+	Intro. to Computer Science	SU15			A	4

Lower Division Requirements (18)

Course Number	Course Name	1	2	3	GR	HR
CS 1401+	Intro. to Computer Science	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	4
CS 2401+	Elem. Data Struct./Algorithms	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	4
MATH 2300+	Discrete Mathematics	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	3
CS 2302+	Data Structures	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	3
EE 2369+	Digital Systems Design I	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	3
EE 2169+	Digital Systems Design I Lab	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	1

[Save Changes](#) [Reset](#) [Return to Home](#)

Figure 14: *edit_lowerdiv.php*

The first few lines in this file open a connection to the database and call the functions:

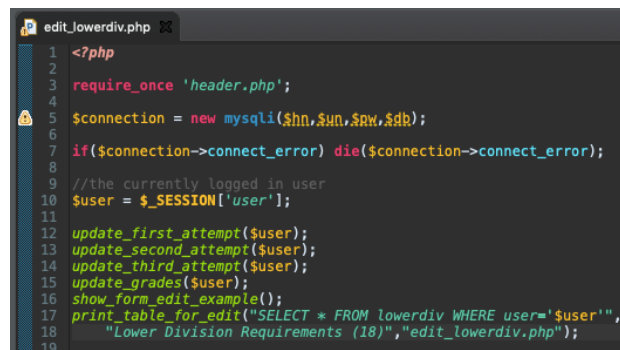
- *update_first_attempt*
- *update_second_attempt*
- *update_third_attempt*
- *update_grades*
- *show_form_edit_example*
- *print_table_for_edit*

as shown in Figure 15.

show_form_edit_example prints a simple, one-row table with an example of how to enter the appropriate information, and *print_table_for_edit* prints *lowerdiv* as it is printed in the home

page, except this representation of *lowerdiv* includes editable text fields in the entries that can be edited (First attempt, second attempt, and grade). At the bottom of the table, there are three buttons, “Save Changes,” which submits the form to the server for handling and updating of the database, “Reset,” which resets each text field to its previously saved value, and “Return to Home,” which disregards any changes that might have been input into the fields and takes the user back to the home page.

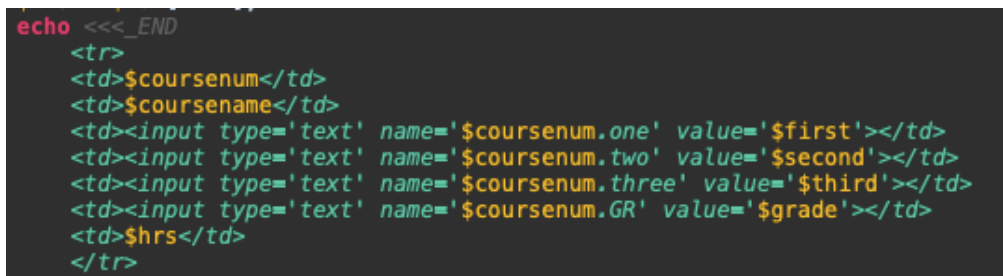
The first 4 function calls check if the table form has been submitted. If it has, then the `$_POST` array contains the values the user desires to add to the table. If any of the `$_POST` values are set, the appropriate function will make the appropriate changes, or it will skip values that aren’t set.

A screenshot of a code editor showing the first 19 lines of a PHP file named `edit_lowerdiv.php`. The code includes a PHP opening tag, a `require_once` statement for `header.php`, a MySQL connection setup with `mysqli`, an error check, a session variable `$user` retrieval, and several function calls: `update_first_attempt`, `update_second_attempt`, `update_third_attempt`, `update_grades`, and `show_form_edit_example`. It also includes a `print_table_for_edit` function call with a SQL query and file path.

```
1 <?php
2
3 require_once 'header.php';
4
5 $connection = new mysqli($hn,$un,$pw,$db);
6
7 if($connection->connect_error) die($connection->connect_error);
8
9 //the currently logged in user
10 $user = $_SESSION['user'];
11
12 update_first_attempt($user);
13 update_second_attempt($user);
14 update_third_attempt($user);
15 update_grades($user);
16 show_form_edit_example();
17 print_table_for_edit("SELECT * FROM lowerdiv WHERE user='$user'",
18 "Lower Division Requirements (18)","edit_lowerdiv.php");
19
```

Figure 15: Opening lines of `edit_lowerdiv.php`

Figure 16 shows a portion of the function `print_table_for_edit`, which prints the HTML table with the appropriate text fields.

A screenshot of a code editor showing a portion of the `print_table_for_edit` function. It displays an `echo` statement with a `<<<_END` comment, followed by an HTML table structure. The table has four columns: `$coursenum`, `$coursename`, and two text input fields. The first text field is named `$coursenum.one` and has a value of `$first`. The second text field is named `$coursenum.two` and has a value of `$second`. The third column is named `$coursenum.three` and has a value of `$third`. The fourth column is named `$coursenum.GR` and has a value of `$grade`. The table is closed with `</tr>`.

```
echo <<<_END
<tr>
<td>$coursenum</td>
<td>$coursename</td>
<td><input type='text' name='$coursenum.one' value='$first'></td>
<td><input type='text' name='$coursenum.two' value='$second'></td>
<td><input type='text' name='$coursenum.three' value='$third'></td>
<td><input type='text' name='$coursenum.GR' value='$grade'></td>
<td>$hrs</td>
</tr>
```

Figure 16: HTML table with editable fields to print *lowerdiv*

Each text field is given a name, which in this case, is the name of the course at the current row concatenated with the keywords *one*, *two*, *three*, or *GR*. For example, the text field at row CS 1401+, column 1 will have a name of **CS 1401+.one**. The text field at row CS 1401+, column 2 will have a name of **CS 1401+.two**, and so on. Figure 17 is an illustration of how the text fields in rows CS 1401+ and CS 2401+ are named.

Lower Division Requirements (18)						
Course Number	Course Name	1	2	3	GR	HR
CS 1401+	Intro. to Computer Science	<input type="text" value="CS 1401+.one"/>	<input type="text" value="CS 1401+.two"/>	<input type="text" value="CS 1401+.three"/>	<input type="text" value="CS 1401+.GR"/>	4
CS 2401+	Elem. Data Struct./Algorithms	<input type="text" value="CS 2401+.one"/>	<input type="text" value="CS 2401+.one"/>	<input type="text" value="CS 2401+.one"/>	<input type="text" value="CS 2401+.one"/>	4

Figure 17: Demonstration of text field naming conventions

These text field names will be stored in the `$_POST` array when the user clicks on “Save Changes” and submits the form with his or her changes. Each field name in the form will be an index of the post array. For instance, if the value at CS 1401+.one is SU15 it will be saved in the post array as follows:

```
$_POST['CS_1401+_one'] = 'SU15'
```

Notice how the field’s name went from being **CS 1401+.one** to **CS_1401+_one** in the `$_POST` array. **This is because PHP replaces whitespace and dots with an underscore in super global array indices.** All of the tables in this website follow the same naming conventions for each of their editable fields and their respective indices in the `$_POST` array. If you wish to use the values in the editable tables for future changes, pay close attention to the naming conventions, **it is easy to make a mistake, or render an incorrect index.**

To better illustrate the naming conventions, take for example that user enters the values shown in Figure 18. The `$_POST` array will store them like this:

```
$_POST['CS_1401+_one'] = 'SU15'
$_POST['CS_1401+_two'] = ''
$_POST['CS_1401+_three'] = ''
$_POST['CS_1401+_GR'] = 'A'
$_POST['CS_2401+_one'] = 'FA15'
$_POST['CS_2401+_two'] = ''
$_POST['CS_2401+_three'] = ''
$_POST['CS_2401+_GR'] = 'B'
$_POST['MATH_2300+_one'] = 'FA15'
$_POST['MATH_2300+_two'] = 'SP16'
$_POST['MATH_2300+_three'] = ''
$_POST['MATH_2300+_GR'] = 'B'
$_POST['CS_2302+_one'] = 'SP16'
$_POST['CS_2302+_two'] = ''
$_POST['CS_2302+_three'] = ''
$_POST['CS_2302+_GR'] = 'A'
```

```

$_POST['EE_2369+_one'] = 'SP16'
$_POST['EE_2369+_two'] = ''
$_POST['EE_2369+_three'] = ''
$_POST['EE_2369+_GR'] = 'A'
$_POST['EE_2169+_one'] = 'SP16'
$_POST['EE_2169+_two'] = ''
$_POST['EE_2169+_three'] = ''
$_POST['EE_2169+_GR'] = 'A'

```

Course Number	Course Name	1	2	3	GR	HR
CS 1401+	Intro. to Computer Science	SU15			A	4
CS 2401+	Elem. Data Struct./Algorithms	FA15			B	4
MATH 2300+	Discrete Mathematics	FA15	SP16		B	3
CS 2302+	Data Structures	SP16			A	3
EE 2369+	Digital Systems Design I	SP16			A	3
EE 2169+	Digital Systems Design I Lab	SP16			A	1

Save Changes Reset Return to Home

Figure 18: Editing the lowerdiv table

update_first_attempt, *update_second_attempt*, *update_third_attempt*, and *update_grades* all work in the exact same way; they check for any set or edited values in the table and retrieve them from the *\$_POST* array. Figure 19 shows the first few lines of *update_first_attempt*.

```

31 function update_first_attempt($user){
32     if(isset($_POST['CS_1401+_one']) &&
33         $_POST['CS_1401+_one'] != queryMysql("SELECT one FROM lowerdiv WHERE user='$user' AND coursenum='CS 1401+';")){
34         $first = sanitizeString($_POST['CS_1401+_one']);
35         queryMysql("UPDATE lowerdiv SET one='$first' WHERE user='$user' AND coursenum='CS 1401+';");
36     }
37     if(isset($_POST['CS_2401+_one']) &&
38         $_POST['CS_2401+_one'] != queryMysql("SELECT one FROM lowerdiv WHERE user='$user' AND coursenum='CS 2401+';")){
39         $first = sanitizeString($_POST['CS_2401+_one']);
40         queryMysql("UPDATE lowerdiv SET one='$first' WHERE user='$user' AND coursenum='CS 2401+';");
41     }

```

Figure 19: Snippet of *update_first_attempt*

update_first_attempt takes the current *\$user* as a parameter. The first condition statement checks if there is a value in the *\$_POST*['CS_1401+_one'] array position. Then, the database is queried to retrieve the value stored at the row corresponding to the current user (8000400), the current course (CS 1401+) and the column 1. As shown in Figure 5, the current value is NULL, so the comparison *\$_POST*['CS_1401+_one'] != NULL holds and the program enters the code in the curly brackets.

The value in *\$_POST*['CS_1401+_one'] is passed to the *sanitizeString* function to ensure that this value is safe to use in a MySQL query. The variable *\$first* now holds the safe string from the *\$_POST* and it can be used in the query on the next line to update the appropriate row and column in the database.

These steps are repeated through the remainder of *update_first_attempt* and through the 3 remaining *update* functions. Once the changes are saved and the user returns to the home page, the changes made are rendered correctly as shown in Figure 20.

Welcome, Tony Stark						
Lower Division Requirements (18)						
Course Number	Course Name	1	2	3	GR	HR
CS 1401+	Intro. to Computer Science	SU15			A	4
CS 2401+	Elem. Data Struct./Algorithms	FA15			B	4
MATH 2300+	Discrete Mathematics	FA15	SP16		B	3
CS 2302+	Data Structures	SP16			A	3
EE 2369+	Digital Systems Design I	SP16			A	3
EE 2169+	Digital Systems Design I Lab	SP16			A	1

Figure 20: Changes to *lowerdiv* in database are successfully saved and rendered in the home page.

edit_core.php

Editing the Core Curriculum is very similar to editing the Lower Division Requirements. The *core* table has a key difference from *lowerdiv*. There are five free courses in *core*, they must only fall under the subjects of *L Phil & Cult*, *Creative Arts+*, *Soc & BehSc+*, *Comp Area Opt1* and *Comp Area Opt2*. Input checking and insertions into the database remain the same, with the only difference being that *L Phil & Cult*, *Creative Arts+*, *Soc & BehSc+*, *Comp Area Opt1* and *Comp Area Opt2* are added as indices to the `$_POST` array. The contents of the form in Figure 21 are stored in the `$_POST` array. The built-in PHP function `var_dump` prints readable information about one or more variables like the value and type of the variables. Calling `var_dump($_POST)` produces the following output:

```
[ "RWS_1301+_one" ]=> string(4) "SU16"
[ "RWS_1301+_two" ]=> string(0) ""
[ "RWS_1301+_three" ]=> string(0) ""
[ "RWS_1301+_GR" ]=> string(1) "A"
[ "RWS_1302+_one" ]=> string(4) "SP17"
[ "RWS_1302+_two" ]=> string(0) ""
[ "RWS_1302+_three" ]=> string(0) ""
[ "RWS_1302+_GR" ]=> string(1) "A"
[ "MATH_1411+_one" ]=> string(4) "SP15"
[ "MATH_1411+_two" ]=> string(0) ""
[ "MATH_1411+_three" ]=> string(0) ""
[ "MATH_1411+_GR" ]=> string(1) "A"
[ "L_Phil_&_Cult+" ]=> string(15) "Intro to Ethics"
[ "L_Phil_&_Cult+_one" ]=> string(4) "SP16"
```

```
["L_Phil_&_Cult+_two"]=> string(0) ""
["L_Phil_&_Cult+_three"]=> string(0) ""
["L_Phil_&_Cult+_GR"]=> string(1) "A"
["Creative_Arts+"]=> string(8) "MUS 1317"
["Creative_Arts+_one"]=> string(4) "SP15"
["Creative_Arts+_two"]=> string(0) ""
["Creative_Arts+_three"]=> string(0) ""
["Creative_Arts+_GR"]=> string(1) "A"
["Soc_&_BehSc+"]=> string(10) "PSYCH 1310" [
"Soc_&_BehSc+_one"]=> string(4) "SP15"
["Soc_&_BehSc+_two"]=> string(0) ""
["Soc_&_BehSc+_three"]=> string(0) ""
["Soc_&_BehSc+_GR"]=> string(1) "A"
["Comp_Area_Opt1+"]=> string(9) "UNIV 1301"
["Comp_Area_Opt1+_one"]=> string(4) "FA14"
["Comp_Area_Opt1+_two"]=> string(0) ""
["Comp_Area_Opt1+_three"]=> string(0) ""
["Comp_Area_Opt1+_GR"]=> string(1) "A"
["Comp_Area_Opt2+"]=> string(7) "CS 1310"
["Comp_Area_Opt2+_one"]=> string(4) "SP15"
["Comp_Area_Opt2+_two"]=> string(0) ""
["Comp_Area_Opt2+_three"]=> string(0) ""
["Comp_Area_Opt2+_GR"]=> string(1) "A"
["HIST_1301+_one"]=> string(4) "FA14"
["HIST_1301+_two"]=> string(0) ""
["HIST_1301+_three"]=> string(0) ""
["HIST_1301+_GR"]=> string(1) "A"
["HIST_1302+_one"]=> string(4) "SP15"
["HIST_1302+_two"]=> string(0) ""
["HIST_1302+_three"]=> string(0) ""
["HIST_1302+_GR"]=> string(1) "A"
["POLS_2310+_one"]=> string(4) "FA14"
["POLS_2310+_two"]=> string(0) ""
["POLS_2310+_three"]=> string(0) ""
["POLS_2310+_GR"]=> string(1) "A"
["POLS_2311+_one"]=> string(4) "SP16"
["POLS_2311+_two"]=> string(0) ""
["POLS_2311+_three"]=> string(0) ""
["POLS_2311+_GR"]=> string(1) "A"
```

Once the changes are saved and the user returns to the home page, the changes to *core* are rendered correctly as shown in Figure 22.

Core Curriculum (37)						
Course Number	Course Name	1	2	3	GR	HR
RWS 1301+	Rhetoric and Composition 1	SU16			A	3
RWS 1302+	Rhetoric and Composition 2	SP17			A	4
MATH 1411+	Calculus I	SP15			A	3
L Phil & Cult+	Intro to Ethics	SP16			A	3
Creative Arts+	MUS 1317	SP15			A	3
Soc & BehSc+	PSYCH 1310	SP15			A	3
Comp Area Opt1+	UNIV 1301	FA14			A	3
Comp Area Opt2+	CS 1310	SP15			A	3
HIST 1301+	History of U.S. to 1865	FA14			A	3
HIST 1302+	History of U.S. since 1865	SP15			A	3
POLS 2310+	Introduction to Politics	FA14			A	3
POLS 2311+	American Government & Politics	SP16			A	3

[Return to Home](#)

Figure 21: Editing the core table

Core Curriculum (37)*						
Course Number	Course Name	1	2	3	GR	HR
RWS 1301+	Rhetoric and Composition 1	SU16			A	3
RWS 1302+	Rhetoric and Composition 2	SP17			A	4
MATH 1411+	Calculus I	SP15			A	3
L Phil & Cult+	Intro to Ethics	SP16			A	3
Creative Arts+	MUS 1317	SP15			A	3
Soc & BehSc+	PSYCH 1310	SP15			A	3
Comp Area Opt1+	UNIV 1301	FA14			A	3
Comp Area Opt2+	CS 1310	SP15			A	3
HIST 1301+	History of U.S. to 1865	FA14			A	3
HIST 1302+	History of U.S. since 1865	SP15			A	3
POLS 2310+	Introduction to Politics	FA14			A	3
POLS 2311+	American Government & Politics	SP16			A	3

Figure 22: Changes to core in database are successfully saved and rendered in the home page.

edit_math.php

The *othermath* table can be edited by the user and processed by the server in the same way that *lowerdiv* is edited and processed; \$_POST array values are checked, input is sanitized, and appropriate *UPDATE* queries are sent to the database. The contents of the form in Figure 23 are stored as shown below by the var_dump of the \$_POST array:

```
["MATH_1312+_one"]=> string(4) "FA15"  
["MATH_1312+_two"]=> string(4) "FA16"  
["MATH_1312+_three"]=> string(0) ""  
["MATH_1312+_GR"]=> string(1) "A"  
["MATH_3323+_one"]=> string(4) "SU16"  
["MATH_3323+_two"]=> string(0) ""  
["MATH_3323+_three"]=> string(0) ""  
["MATH_3323+_GR"]=> string(1) "C"  
["MATH_4329_one"]=> string(4) "FA17"  
["MATH_4329_two"]=> string(4) "SP18"  
["MATH_4329_three"]=> string(0) ""  
["MATH_4329_GR"]=> string(1) "A"  
["STAT_3320_one"]=> string(4) "SP18"  
["STAT_3320_two"]=> string(0) ""  
["STAT_3320_three"]=> string(0) ""  
["STAT_3320_GR"]=> string(1) "B"
```

Once the changes are saved and the user returns to the home page, the changes to *othermath* are rendered correctly as shown in Figure 24.

Other Required Mathematics Courses (12)						
Course Number	Course Name	1	2	3	GR	HR
MATH 1312+	Calculus II	FA15	FA16		A	3
MATH 3323+	Matrix Algebra	SU16			C	3
MATH 4329	Numerical Analysis	FA17	SP18		A	3
STAT 3320	Probability & Statistics for CS	SP18			B	3

[Save Changes](#) [Reset](#) [Return to Home](#)

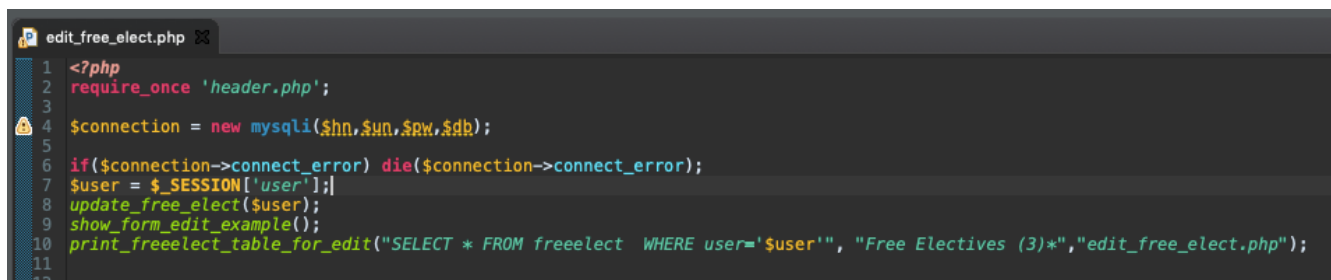
Figure 23: Editing the othermath table

Other Required Mathematics Courses (12)						
Course Number	Course Name	1	2	3	GR	HR
MATH 1312+	Calculus II	FA15	FA16		A	3
MATH 3323+	Matrix Algebra	SU16			C	3
MATH 4329	Numerical Analysis	FA17	SP18		A	3
STAT 3320	Probability & Statistics for CS	SP18			B	3

Figure 24: Changes to othermath in database are successfully saved and rendered in the home page.

edit_free_elect.php

The *freeelect* table is edited slightly different than the rest of the tables, since it is made up of a single row. Printing this table in its editable format is accomplished by calling *print_free_elect_table_for_edit* in the file *functions.php*. This function reduces the amount of code needed to print a single-row table. *edit_free_elect.php* checks for values added to the `$_POST` array in a single function, *update_free_elect* and inserts the changes into the database. *frenum* is the name given to the text field that will hold the course number. *freename* is the name given to the text field that will hold the course name. Figure 25 shows the first lines of *edit_free_elect.php*: considerably fewer function calls are made to edit this table compared to the rest of them.



```
1 <?php
2 require_once 'header.php';
3
4 $connection = new mysqli($hn,$un,$pw,$db);
5
6 if($connection->connect_error) die($connection->connect_error);
7 $user = $_SESSION['user'];
8 update_free_elect($user);
9 show_form_edit_example();
10 print_freeelect_table_for_edit("SELECT * FROM freeelect WHERE user='$user'", "Free Electives (3)*","edit_free_elect.php");
11
12
```

Figure 25: First 10 lines of *edit_free_elect.php*

The values in the *freeelect* form shown in Figure 26 are stored in the `$_POST` array as follows:

`["frenum"]=> string(9) "ECON 2310"`

`["freename"]=> string(31) "Introduction to Macro-Economics"`

`["first"]=> string(4) "SP18"`

`["second"]=> string(0) ""`

`["third"]=> string(0) ""`

`["grade"]=> string(1) "A"`

When the user clicks on “Save Changes” and returns to the home page, the edits made to *freeelect* in the database are rendered in the home page (Figure 27).



Course Number	Course Name	1	2	3	GR	HR
ECON 2310	Introduction to Macro-	SP18			A	3

Save Changes Reset Return to Home

Figure 26: Editing *freeelect*

Free Electives (3)*						
Course Number	Course Name	1	2	3	GR	HR
ECON 2310	Introduction to Macro-Economics	SP18			A	3

Figure 27: Changes to freeelect in database are successfully saved and rendered in the home page.

edit_sciences.php

Making changes to the *sciences* table is slightly different than making changes to other tables. The reason for this, is because inserting course data into the last two rows is a little more complex than inserting to any other row in any of the previously discussed tables. The last two rows of *sciences* have all fields blank except the HR column (Figure 28).

Life & Physical Sciences (12)*						
Course Number	Course Name	1	2	3	GR	HR
PHYS 2420	Introductory Mechanics					4
						4
						4

Figure 28: sciences table as it appears in the user's home page.

Inserting into the first row is easy because there are two already populated, and unique fields, *coursename* which holds the value "PHYS 2420" and *coursenum* which hold the value "Introductory Mechanics." Inserting into *one*, *two*, *three*, or *GR* can easily be accomplished with a MySQL query such as:

```
UPDATE sciences SET GR='A' WHERE coursenum='PHYS 2420';
```

The query above successfully updates the value of *GR* to an A in the PHYS 2420 row.

Attempting a query using the *HR* value to update either of the last two rows would be unsuccessful because it would overwrite all three rows, since all of them have *HR*=4.

Recall from Figure 9 that an additional column named *hidden* is **present in the MySQL table but it is not visible in the student's view of the table**. This behavior is possible because the function *print_table* called in the home page, and the function *print_sciences_table_for_edit* omit printing the *hidden* column, since it is unnecessary for the end user to know it exists.

The second row in *sciences* has *hidden*='science1' which is an identifier meaning "science elective 1" and the third row has *hidden*='science2' which is an identifier meaning "science elective 2".

edit_sciences.php calls the functions *update_coursenum*, *update_coursename*, *update_first_attempt*, *update_second_attempt*, *update_third_attempt*, and *update_grade_attempt*, shown in Figure 29.

```
1 <?php
2 require_once 'header.php';
3
4 $connection = new mysqli($hn,$un,$pw,$db);
5
6 if($connection->connect_error) die($connection->connect_error);
7
8 $user = $_SESSION['user'];
9 //good for debugging
10 var_dump($_POST);
11 update_course_num($user);
12 update_course_name($user);
13 update_first_attempt($user);
14 update_second_attempt($user);
15 update_third_attempt($user);
16 update_grades($user);
17 show_form_edit_example();
18 print_sciences_table_for_edit("SELECT * FROM sciences WHERE user='$user'", "Life & Physical Sciences (12)*","edit_sciences.php");
19
```

Figure 29: Lines 1-18 of *edit_sciences*.

Attempting to edit a column in the second or third row of the table will use their respective hidden fields in a MySQL query. Figure 30, *update_course_num* shows how this is accomplished.

```
20 function update_course_num($user){
21     if(isset($_POST['coursenum_science1']) &&
22         $_POST['coursenum_science1'] != queryMySQL("SELECT coursenum FROM sciences WHERE user='$user' AND hidden='science1';")){
23         $coursenum = sanitizeString($_POST['coursenum_science1']);
24         queryMySQL("UPDATE sciences SET coursenum='$coursenum' WHERE user='$user' AND hidden='science1';");
25     }
26     if(isset($_POST['coursenum_science2']) &&
27         $_POST['coursenum_science2'] != queryMySQL("SELECT coursenum FROM sciences WHERE user='$user' AND hidden='science2';")){
28         $coursenum = sanitizeString($_POST['coursenum_science2']);
29         queryMySQL("UPDATE sciences SET coursenum='$coursenum' WHERE user='$user' AND hidden='science2';");
30     }
31 }
```

Figure 30: *update_course_num*

The values in the *sciences* table form shown in Figure 31 are stored as follows in the *\$_POST* array:

```
["PHYS_2420_one"]=> string(4) "SU16"
["PHYS_2420_two"]=> string(0) ""
["PHYS_2420_three"]=> string(0) ""
["PHYS_2420_GR"]=> string(1) "A"
["coursenum_science1"]=> string(9) "GEOL 1301"
["coursename_science1"]=> string(25) "Intro to Physical Geology"
["first_science1"]=> string(4) "FA15"
["second_science1"]=> string(0) ""
["third_science1"]=> string(0) ""
```

```

["grade_science1"]=> string(1) "A"
["coursenum_science2"]=> string(9) "GEOL 1302"
["coursename_science2"]=> string(18) "Historical Geology"
["first_science2"]=> string(4) "FA16"
["second_science2"]=> string(0) ""
["third_science2"]=> string(0) ""
["grade_science2"]=> string(1) "A"

```

After saving changes to the form and returning to the homepage, Figure 32 shows that the values in *sciences* in the database were successfully updated and rendered in the home page.

Life & Physical Sciences (12)*						
Course Number	Course Name	1	2	3	GR	HR
PHYS 2420	Introductory Mechanics	SU16			A	4
GEOL 1301	Intro to Physical Geol.	FA15			A	4
GEOL 1302	Historical Geology	FA16			A	4

Save Changes Reset [Return to Home](#)

Figure 31: Editing sciences

Life & Physical Sciences (12)*						
Course Number	Course Name	1	2	3	GR	HR
PHYS 2420	Introductory Mechanics	SU16			A	4
GEOL 1301	Intro to Physical Geology	FA15			A	4
GEOL 1302	Historical Geology	FA16			A	4

Figure 32: Changes to sciences in database are successfully saved and rendered in the home page.

edit_upperdiv.php

The instructions used to edit the *upperdiv* table are similar to editing *lowerdiv*, *core*, and *other math*. The printing function for the editable form used is *print_table_for_edit* from *functions.php* since the table form is in the same format as *lowerdiv*, *core*, and *othermath*. The values in the form shown in Figure 33 are stored in the *\$_POST* array as follows:

```

["CS_3195_one"]=> string(4) "FA17"
["CS_3195_two"]=> string(0) ""
["CS_3195_three"]=> string(0) ""
["CS_3195_GR"]=> string(1) "A"
["CS_3331+_one"]=> string(4) "FA17"
["CS_3331+_two"]=> string(0) ""
["CS_3331+_three"]=> string(0) ""
["CS_3331+_GR"]=> string(1) "A"

```

```

["CS_3350_one"]=> string(4) "SP18"
["CS_3350_two"]=> string(0) ""
["CS_3350_three"]=> string(0) ""
["CS_3350_GR"]=> string(1) "A"
["CS_3360_one"]=> string(4) "SP18"
["CS_3360_two"]=> string(0) ""
["CS_3360_three"]=> string(0) ""
["CS_3360_GR"]=> string(1) "A"
["CS_3432+_one"]=> string(3) "F16"
["CS_3432+_two"]=> string(0) ""
["CS_3432+_three"]=> string(0) ""
["CS_3432+_GR"]=> string(1) "B"
["CS_4310+_one"]=> string(3) "F17"
["CS_4310+_two"]=> string(0) ""
["CS_4310+_three"]=> string(0) ""
["CS_4310+_GR"]=> string(1) "B"
["CS_4311_one"]=> string(3) "F18"
["CS_4311_two"]=> string(0) ""
["CS_4311_three"]=> string(0) ""
["CS_4311_GR"]=> string(1) "A"
["CS_4375_one"]=> string(3) "F17"
["CS_4375_two"]=> string(4) "SP18"
["CS_4375_three"]=> string(4) "FA18"
["CS_4375_GR"]=> string(1) "B"

```

After the user saves the changes to the form and returns to the home page, the table is successfully updated in the database and rendered in the home page (Figure 34).

Upper Division Requirements (23)						
Course Number	Course Name	1	2	3	GR	HR
CS 3195	Jr. Professional Orientation	FA17			A	1
CS 3331+	Adv. Object-Oriented Programming	FA17			A	3
CS 3350	Automata/Computabi/Formal Lang.	SP18			A	3
CS 3360	Design/Impl. Programming Languages	SP18			A	3
CS 3432+	Comp. Arch I:Comp/Org Design	F16			B	4
CS 4310+	Software Eng: Requirements Eng.	F17			B	3
CS 4311	Software Eng: Design & Implmnt.	F18			A	3
CS 4375	Theory of Operating Systems	F17	SP18	FA18	B	3
<input type="button" value="Save Changes"/> <input type="button" value="Reset"/> Return to Home						

Figure 33: Editing upperdiv

Upper Division Requirements (23)						
Course Number	Course Name	1	2	3	GR	HR
CS 3195	Jr. Professional Orientation	FA17			A	1
CS 3331+	Adv. Object-Oriented Programming	FA17			A	3
CS 3350	Automata/Computabi/Formal Lang.	SP18			A	3
CS 3360	Design/Impl. Programming Languages	SP18			A	3
CS 3432+	Comp. Arch I:Comp/Org Design	F16			B	4
CS 4310+	Software Eng: Requirements Eng.	F17			B	3
CS 4311	Software Eng: Design & Implmnt.	F18			A	3
CS 4375	Theory of Operating Systems	F17	SP18	FA18	B	3

Figure 34: Changes to upperdiv in database are successfully saved and rendered in the home page.

edit_techselect.php

The *techelect* table follows the same editing pattern as the *sciences* table since all of its rows are empty and there is no unique identifier on any row to be used in a MySQL query to update a row. Recall from Figure 11 that there is a *hidden* column that holds the following values:

elect1 for the first row (technical elective 1)
 elect2 for the second row (technical elective 2)
 elect3 for the third row (technical elective 3)
 elect4 for the fourth row (technical elective 4)
 elect5 for the fifth row (technical elective 5)

The instructions used to edit and update the database are identical to those instructions used to edit the *sciences* table. The functions *print_table* and *print_techselect_table_for_edit* also omit printing the *hidden* column to the end user. The fields in the form shown in Figure 35 are stored as follows:

```
["coursenum_elect1"]=> string(7) "CS 5375"
["coursename_elect1"]=> string(28) "Software Reverse Engineering"
["first_elect1"]=> string(4) "FA17"
["second_elect1"]=> string(0) ""
["third_elect1"]=> string(0) ""
["grade_elect1"]=> string(1) "A"
["coursenum_elect2"]=> string(7) "CS 4330"
["coursename_elect2"]=> string(30) "Mobile Application Development"
["first_elect2"]=> string(4) "SP18"
["second_elect2"]=> string(0) ""
["third_elect2"]=> string(0) ""
["grade_elect2"]=> string(1) "A"
```

```

["coursenum_elect3"]=> string(7) "CS 5383"
["coursename_elect3"]=> string(33) "Software Requirements Engineering"
["first_elect3"]=> string(4) "FA18"
["second_elect3"]=> string(0) ""
["third_elect3"]=> string(0) ""
["grade_elect3"]=> string(1) "A"
["coursenum_elect4"]=> string(7) "CS 5374"
["coursename_elect4"]=> string(21) "Software Construction"
["first_elect4"]=> string(4) "FA18"
["second_elect4"]=> string(0) ""
["third_elect4"]=> string(0) ""
["grade_elect4"]=> string(1) "A"
["coursenum_elect5"]=> string(7) "CS 5388"
["coursename_elect5"]=> string(18) "Project Management"
["first_elect5"]=> string(4) "FA18"
["second_elect5"]=> string(0) ""
["third_elect5"]=> string(0) ""
["grade_elect5"]=> string(1) "A"

```

Once changes have been saved and the home page has been reloaded, *tech elect* is successfully edited and rendered in the homepage (Figure 36).

Technical Electives (15)*						
Course Number	Course Name	1	2	3	GR	HR
CS 5375	Software Reverse Eng	FA17			A	3
CS 4330	Mobile Application De	SP18			A	3
CS 5383	Software Requirement	FA18			A	3
CS 5374	Software Constructior	FA18			A	3
CS 5388	Project Management	FA18			A	3
<input type="button" value="Save Changes"/> <input type="button" value="Reset"/> <input type="button" value="Return to Home"/>						

Figure 35: Editing techelect

Technical Electives (15)*						
Course Number	Course Name	1	2	3	GR	HR
CS 5375	Software Reverse Engineering	FA17			A	3
CS 4330	Mobile Application Development	SP18			A	3
CS 5383	Software Requirements Engineering	FA18			A	3
CS 5374	Software Construction	FA18			A	3
CS 5388	Project Management	FA18			A	3

Figure 36: Changes to upperdiv in database are successfully saved and rendered in the home page.

The Advisor View

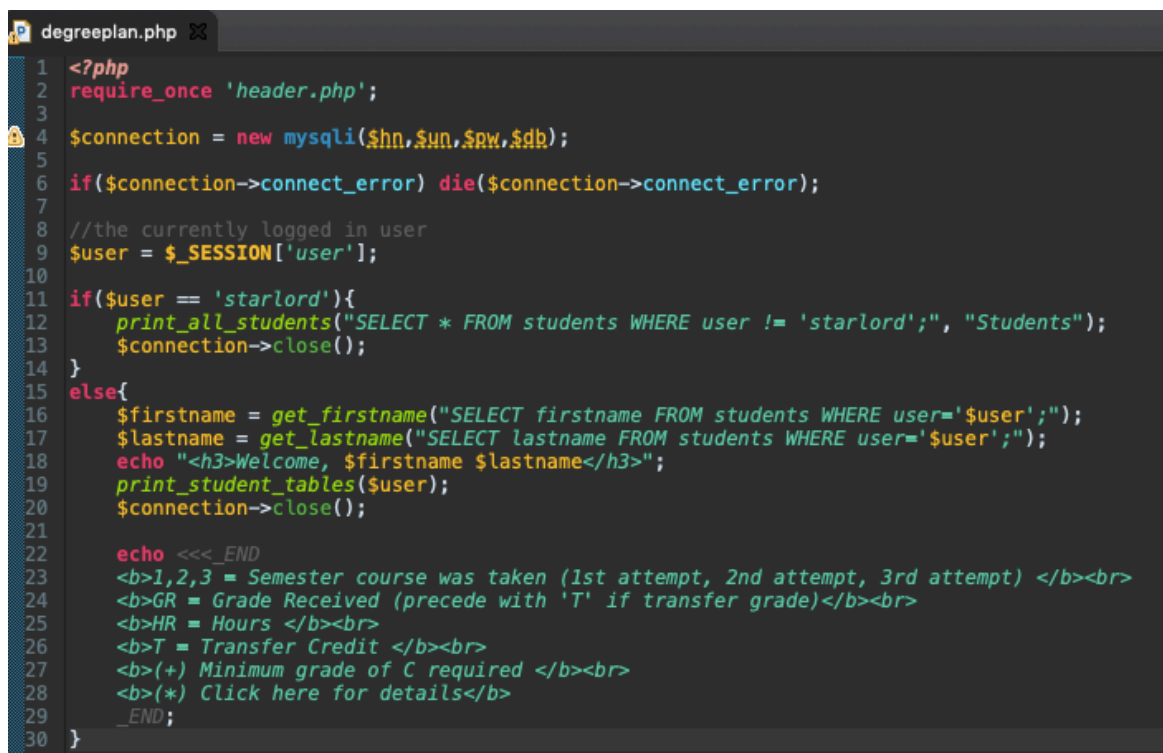
The system provides an Advisor View account that displays a table of all the students currently enrolled in the system and a hyperlink that takes the advisor from his or her homepage to the selected student's current degree plan. This functionality is accomplished by the file that displays the home page, *degreeplan.php* (Figure 37).

The advisor view credentials are:

Username: starlord

Password: 1914

After opening a connection to the database and obtaining the currently logged in user from the `$_SESSION` array, the program checks whether the logged in user is "starlord." If true, the function `print_all_students` from `functions.php` is called. This function prints a table with two columns, the *StudentID* and the *Student Name* (Figure 38). If the logged in user is not starlord, then a student is logged in. The `else` statement executes and produces the home page for the student where he or she can view and edit his or her degree plan.



```
1 <?php
2 require_once 'header.php';
3
4 $connection = new mysqli($hn,$un,$pw,$db);
5
6 if($connection->connect_error) die($connection->connect_error);
7
8 //the currently logged in user
9 $user = $_SESSION['user'];
10
11 if($user == 'starlord'){
12     print_all_students("SELECT * FROM students WHERE user != 'starlord';", "Students");
13     $connection->close();
14 }
15 else{
16     $firstname = get_firstname("SELECT firstname FROM students WHERE user='$user'");
17     $lastname = get_lastname("SELECT lastname FROM students WHERE user='$user'");
18     echo "<h3>Welcome, $firstname $lastname</h3>";
19     print_student_tables($user);
20     $connection->close();
21
22     echo <<<_END
23     <b>1,2,3 = Semester course was taken (1st attempt, 2nd attempt, 3rd attempt) </b><br>
24     <b>GR = Grade Received (precede with 'T' if transfer grade)</b><br>
25     <b>HR = Hours </b><br>
26     <b>T = Transfer Credit </b><br>
27     <b>(+) Minimum grade of C required </b><br>
28     <b>(*) Click here for details</b>
29     _END;
30 }
```

Figure 37: *degreeplan.php*

My CS Degree Plan (starlord)	
Home Log out	
Students	
StudentID	Student Name
8000400	Stark, Tony
87654321	Banner, Bruce
89012345	Parker, Peter

Figure 38: starlord's home page: provides a list of all students registered in the system.

Each student ID number in Figure 38 is a hyperlink that takes the advisor to a page where he or she can view the selected student's degree plan. The function `print_all_students` from `functions.php` (Figure 39) prints the table shown in Figure 38. Notice that in line 126, each student ID number is being printed as a hyperlink to the page `view_student.php?view=student.php`. All items in a URL after a '?' are stored as parameters in the `$_GET` array.

For example, if the advisor wishes to see the degree plan of Tony Stark, he or she would click on the student's ID number, which is 8000400.

The advisor can view Tony's degree plan at the page `view_student.php?view=8000400`.

The ID number is stored in the `$_GET` array as: `$_GET['view'] = 8000400`.

```

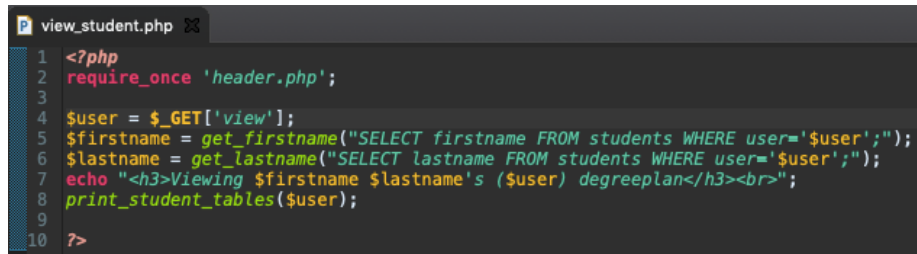
104 function print_all_students($query,$table_title){
105     global $connection;
106     $result = $connection->query($query);
107     if(!$result) die ($connection->connect_error);
108     $rows = $result->num_rows;
109
110     echo "<th><b>$table_title</b></th>";
111     echo <<<_END
112     <table>
113         <tr>
114             <th>StudentID</th>
115             <th>Student Name</th>
116         </tr>
117     _END;
118     for($j =0; $j < $rows; ++$j){
119         $result->data_seek($j);
120         $row = $result->fetch_array(MYSQLI_ASSOC);
121         $student = $row['user'];
122         $firstname= $row['firstname'];
123         $lastname= $row['lastname'];
124         echo <<<_END
125         <tr>
126             <td><a href='view_student.php?view=$student'>$student</a></td>
127             <td>$lastname, $firstname</td>
128         </tr>
129     _END;
130     }
131
132     echo "</table>";
133     echo "<br><br>";
134     $result->close();
135 }

```

Figure 39: `print_all_students`

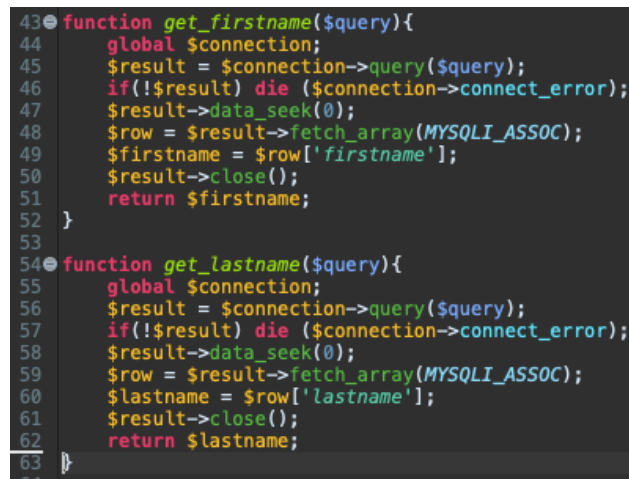
view_student.php

After the advisor clicks on a student's ID number, the file *view_student.php* (Figure 40) retrieves the stored value in the `$_GET` array at index *view*, and stores it in the variable `$user`. Then, the program calls the functions *get_firstname* and *get_lastname* (Figure 41) from *functions.php* which retrieve the specified's student's first and last name to print a "Viewing Tony Stark's 8000400 degree plan" message (Figure 41). The function *print_student_tables* then prints all of the tables in the selected student's degree plan for the advisor to view and edit.



```
1 <?php
2 require_once 'header.php';
3
4 $user = $_GET['view'];
5 $firstname = get_firstname("SELECT firstname FROM students WHERE user='$user'");
6 $lastname = get_lastname("SELECT lastname FROM students WHERE user='$user'");
7 echo "<h3>Viewing $firstname $lastname's ($user) degreeplan</h3><br>";
8 print_student_tables($user);
9
10 ?>
```

Figure 40: *view_student.php*



```
43 function get_firstname($query){
44     global $connection;
45     $result = $connection->query($query);
46     if(!$result) die ($connection->connect_error);
47     $result->data_seek(0);
48     $row = $result->fetch_array(MYSQLI_ASSOC);
49     $firstname = $row['firstname'];
50     $result->close();
51     return $firstname;
52 }
53
54 function get_lastname($query){
55     global $connection;
56     $result = $connection->query($query);
57     if(!$result) die ($connection->connect_error);
58     $result->data_seek(0);
59     $row = $result->fetch_array(MYSQLI_ASSOC);
60     $lastname = $row['lastname'];
61     $result->close();
62     return $lastname;
63 }
64
```

Figure 41: *get_firstname* and *get_lastname*

My CS Degree Plan (starlord)

Home

Log out

Viewing Tony Stark's (8000400) degreeplan

Lower Division Requirements (18)

Course Number	Course Name	1	2	3	GR	HR
CS 1401+	Intro. to Computer Science	SU15			A	4
CS 2401+	Elem. Data Struct./Algorithms	FA15			B	4
MATH 2300+	Discrete Mathematics	FA15	SP16		B	3
CS 2302+	Data Structures	SP16			A	3
EE 2369+	Digital Systems Design I	SP16			A	3
EE 2169+	Digital Systems Design I Lab	SP16			A	1

Edit Lower Division Requirements

Core Curriculum (37)*

Course Number	Course Name	1	2	3	GR	HR
RWS 1301+	Rhetoric and Composition 1	SU16			A	3

Figure 42: Advisor viewing student with ID number 8000400 at `view_student.php?view=8000400`