Simple Bowler Registration

In order to complete this module's homework exercises you must successfully complete the readings for **Chapter 11** within the textbook.

In this assignment you will create a simple form that will collect the first name, last name, age, and bowling average and write the information to a text file. You will also be drawing on your knowledge from previous chapters, let's get started!

Section 1.0 - Getting Started

You will be using the same form layout that you've done in previous assignments so this should look familiar. Feel free to tweak the colors.

1. In Canvas, navigate to **Module 10** and click on the Homework 10 assignment link.
2. Download the **homework10_student.zip** file and extract it. You will find the following files in the folder:
    a. A PHP file (registerbowlers.php)
    b. A CSS file (styles.css)
2. Rename the folder to **homework10** or a folder name of your choice.
    a. Remember, **do not include** any spaces in your folder name.
3. Open **registerbowlers.php** within your editor.
4. Add a title of **Bowler Registeration** within the <title> tag.
5. Link the external CSS file you were provided with to your HTML document using the **<link>** tag directly below the comment tag on Line 6.
6. Jump down to the **<footer>** tag area and hyperlink the **Course Homepage** text to link to your course homepage.
7. Remember to include the title attribute within the anchor tag that reads **Course Homepage**.
8. Add in the appropriate **unique validation URLs** for this assignment. If your CSS file is external, a unique link should be present for the CSS validation.
9. Save your work.

All the coding for this assignment will take place within the **registerbowlers.php** file. The completed web page should look similar to the screenshot below.

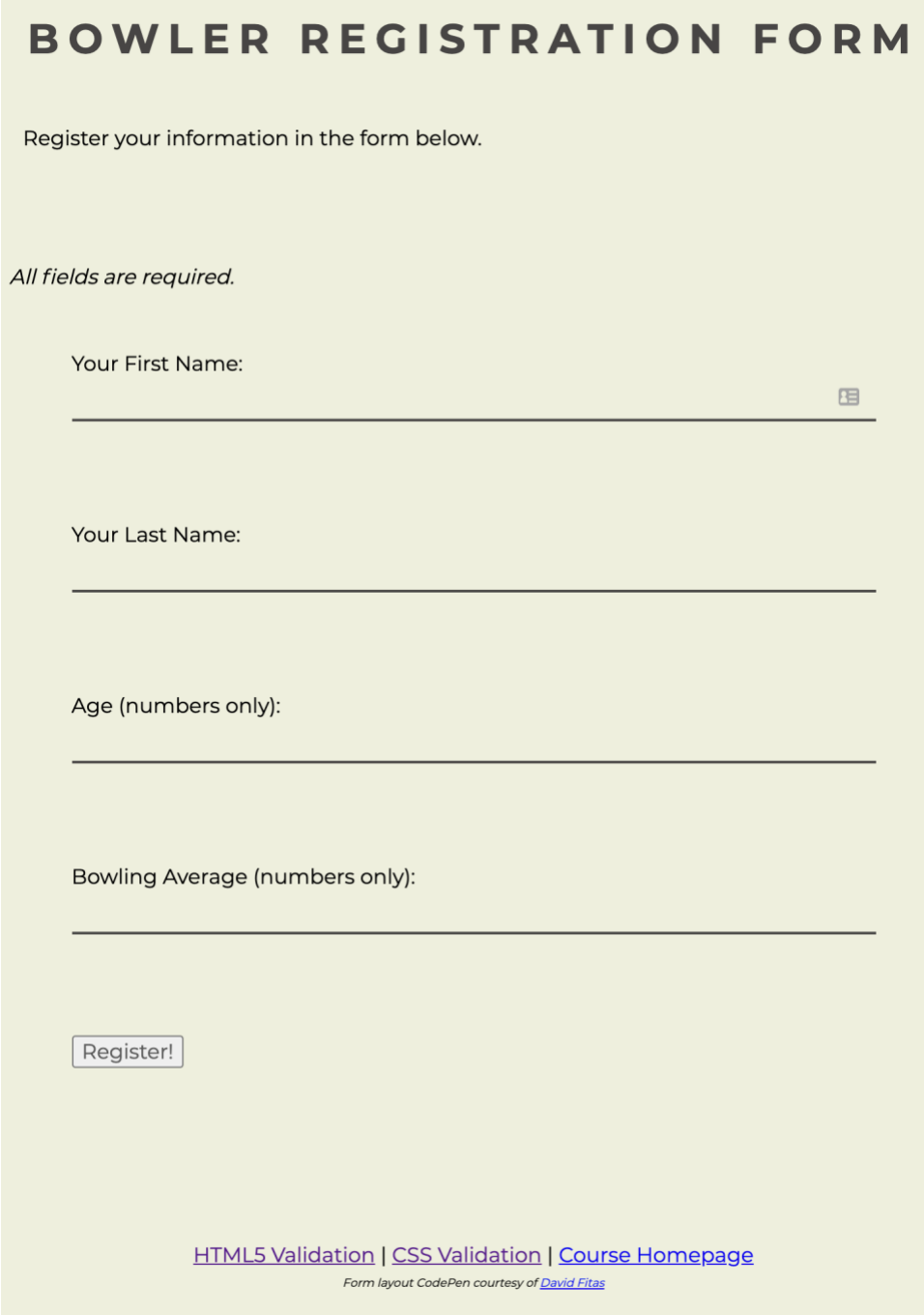

***Figure 1**: The registerbowlers.php file viewed within a web browser (Chrome – Mac)*

Section 1.1 – Creating the Form

Let's start by creating the form so bowlers can register their information. This form will be "sticky" so you may wish to review the Chapter 8 section **on Making Forms Sticky** (page 220).

1. In your editor, located the Start form comment tag, below that add a **<form>** container tag.

2. The opening form tag should include the following:
    a. An **action** attribute that references **registerbowlers.php**
    b. A post **method**
3. With the form tag, add the following code to create the input fields:

```
<p><label>Your First Name:</label><input type="text" name="first_name" size="20" autofocus></p>
<p><label>Your Last Name:</label><input type="text" name="last_name" size="20" ></p>
<p><label>Age (numbers only):</label><input type="text" name="age" size="20" ></p>
<p><label>Bowling Average (numbers only):</label><input type="text" name="average" size="20"></p>
<p><input type="submit" name="submit" value="Register!"></p>
```

4. Using the value attributes below as a reference, add the value to its appropriate input field to make your form "sticky".

```
value="<?php if (isset($_POST['first_name'])) { print htmlspecialchars($_POST['first_name']); } ?>"

value="<?php if (isset($_POST['last_name'])) { print htmlspecialchars($_POST['last_name']); } ?>"

value="<?php if (isset($_POST['age'])) { print htmlspecialchars($_POST['age']); } ?>"

value="<?php if (isset($_POST['average'])) { print htmlspecialchars($_POST['average']); } ?>"
```

5. Save your work.

Section 1.2 – Creating the Script

1. Locate the comment **Start PHP Script** and below that comment add the **PHP delimiters**.
2. **Using PHP multiline commenting** describe the purpose of the script.
    a. You can go back when you've finished coding the script and tweak the description if needed.
3. Add another PHP comment that reads **Print introductory text.**
4. Using either **print** or **echo**, add the following text:
    a. A heading 1 tag that reads: Bowler Registration Form
    b. A paragraph that reads: Register your information in the form below. Please note that all fields are **required** for a successful registration.
        i. Don't forget to make the word required **bold**.

Now onto the good stuff…

5. Add a PHP comment that reads **Identify the file to use:**
6. Type the following code below the comment:

```
$file = 'bowlers.txt';
```

You created a variable named **$file** and identified the text file named **bowlers.txt** as the file that

the information will be written to and stored.

7. Add a PHP comment that reads **Create variables:**
8. Create and add the following **variables:**

```
$form_info = $_POST['first_name'] . " " . $_POST['last_name'] . ", " . $_POST['age'] . ", " . $_POST['average'];
$min = 1;
$max = 300;
```

The **$form_info** variable is going to be used further on in your script to write its contents to the bowlers.txt file, this just simply organizes the form information into a single variable. The **$min** and **$max** variables is going to be used for validating whether the bowler entered an average between 1 and 300.

9. Save your work.

Section 1.3 – Time to Validate!

First off, let's add in some PHP validation using some if conditionals and add a few **print** statements in case there are errors. If you wish to **echo** instead of **print**, that is totally up to you. Much of this code (minus a few tweaks) can be found on pages 220 – 227, please feel free to use your book as a reference for further explanations.

1. Add the PHP comment **Check if the form has been submitted:**
2. Let's check to see if the form has been submitted and add a variable that we will initially assign a false value to indicating that there aren't problems. Type the following code below the comment you created:

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {

    $problem = false;
```

3. Add the PHP comment **Check for each value…**
4. Using the code within the script located on **page 222** as a reference for validating if the **first** and **last name fields** contain content and if not, display a message add the following code below the comment you just created:

```
if (empty($_POST['first_name'])) {
        $problem = true;
        print '<p class="error">Please enter your first name!</p>';
    }

if (empty($_POST['last_name'])) {
        $problem = true;
        print '<p class="error">Please enter your last name!</p>';
    }
```

5. Now, check to see if the user entered their **age** and if they entered a number into the field. If not, display an error message. Note the use of the **&&** Logical Operator (review PHP's Operators on page 135 for a refresher) and the use of the **!** Logical Operator. You want both conditions to be true in this case:

```
if ($_POST['age'] !== null && !is_numeric($_POST['age'])) {
        $problem = true;
        print '<p class="error">Please enter your age (numbers only).</p>';
    }
```

6. Using the code example above, check to see if the user entered a number for their **average** and if not, print an appropriate error message. I used the message **Please enter your current bowling average (numbers only)**.
    a. Remember to apply the error class to the paragraph tag so the error message displays in red.

Remember the **$min** and **$max** variables you created at the start of the script? It's now time to use them. You will use them to validate if the user entered a bowling average between 1 and 300. Note that we are assuming the bowler has an established bowling average.

7. Add the following code:

```
if ($_POST['average'] < $min || $_POST['average'] > $max) {
    $problem = true;
    print '<p class="error">Your average must be 1 and 300.</p>';
    }
```

Note the use of the **||** Logical Operator and the greater than **>** and less than **<** Comparison Operators.

8. Save your work.

Section 1.4 – Writing the Information to a File

We are going to use the **is_writable()** function to return a Boolean value of either true or false that will determine if you can "write" data to the bowlers.txt file (see page 304). If we can write to the file and there aren't any problems or errors, then we are going to use the **file_put_contents()** function (see page 303) to add the information based on the variables we created at the start of the script and the form information added by the user.

1. Add the following code to your script:

```
if ((is_writable($file)) && (!$problem)) {

    file_put_contents($file, $form_info . PHP_EOL, FILE_APPEND);
```

Note the use of the **&& and !** Logical Operators once again. $file is referencing the bowlers.txt file and **!$problem** essentially means that there aren't any problems so we can move ahead and information to the text file.

The **arguments** added to the **file_put_contents()** function do the following:

- **$file** references the file we are writing information to, so in our case it's **bowlers.txt**.
- **$form_info** references the variable that gathers the first name, last name, age, and bowling average from the form that user filled out. A comma separates the name variables, age, and average.
- **PHP_EOL** is a PHP constant that represents the correct end-of-line character based on the current operating system (see page 304).
- **FILE_APPEND** is another PHP constant that allows new data and information to be appended to what's already there (see page 303).

Let's print a few messages letting the user know that their information was successfully registered, a link that allows us to view the bowlers.txt file so check to see if the information was correctly added, and a link to register another bowler.

2. Add the following code (you can use either print or echo) and include the closing curly brace which closes the if conditional in Step 1:

```
    // Print a message:
    print '<p class="success">You are now registered ' . ucfirst($_POST["first_name"]) . " " . ucfirst($_POST["last_name"]) .'! </p>';
    print '<p><a href="bowlers.txt" target="_blank">View Registrations</a></p>';
    print '<p><a href="registerbowlers.php">Register another bowler.</a></p>';
}
```

Note the use of the **ucfirst()** function, that's new huh? This function turns the first letter of in a text string to upper case.

3. Add another closing curly brace **}** below the code you just added that closes the main if conditional where you check to see if the form has been submitted.
4. Save your work.

Section 1.6 – Creating the Text File

There is one final thing left to do before you begin testing and that is to create the text file and set the permissions for the file. You will use FileZilla for this.

1. Please watch the video at https://bit.ly/3isqSuC to see how to create the text file and set the file permissions.
2. After you've created the bowlers.txt file, test the functionality of the form within a web browser and verify that information is being written to the text file.

Watch the Bowler's Registration Form Walkthrough video at https://bit.ly/3DkThLi for what to test for and how the completed form and script should function.

Section 1.7 – Error(s) and Help

Please use the **Help** forum for assistance within Canvas. All errors and/or issues should be posted to the forums so the entire class can participate and offer assistance and solutions for correcting issues. Your classmates are a valuable learning resource.

NOTE: If you want to add error reporting to your PHP script as you've done it in previous assignments, please feel free to do so. The web server we use has the display_errors setting enabled so either way, you will see any errors displayed within the web browser. Not all servers have it enabled so it's always nice to include it if you wish.

Section 1.8 - Update & Submit for Grading

Please note that you always need to notify me that your work is ready to be graded through the appropriate assignment area in Canvas each week. Missing notifications will not receive credit.

1. Log in to the Canvas classroom.
    a. Click on the **Home** tab in the side navigation bar.
    b. Click on **Module 10** and locate the **Module 10 – Homework 10** link**.**
    c. In the comments area let me know you are finished and your assignment is ready for grading, "my work is ready for grading." will do just fine.
    d. **Supply me with your course homepage URL,** *e.g., http://youruserID.macombserver.net/itwp2750/home.htm*
    e. Please note that you will not be attaching your completed assignment file(s) to the assignment area but rather uploading it to the web server where I will access it. It is not

necessary to attach any files. **The file attachment option has been disabled since you will be uploading your work to the student web server.**

f. Once you have submitted notification and your work has been graded, you cannot resubmit for a better grade, the original grade remains as is.

Please be sure to test the links to your work. Test access to your assignment from your course homepage to be sure everything is correctly hyperlinked and in working order. If I cannot access your work, I cannot grade it. The result will be a zero for the assignment.

Please review the Weekly Schedule for other required assignment(s) due for this module.

Grades will be assigned within the limits of the grading policy. Points listed within the Value column of the rubric below are the maximum number of points you can receive if your work is 100% correct. Incorrect work may receive zero through the maximum point value range listed within the Value column. ***Late work is not accepted. The appropriate assignment area(s) will be closed at the time of assignment's due date. Email attachments of any assigned submitted work will not be accepted.***

Please review the course Late Policy within Syllabus (First Day Handout).

Homework 10 Grading Rubric

Criterion	Value
Appropriate page title is present within the title tag. A page title of "Untitled" will not receive credit.	1
Added heading and introductory paragraph text using PHP as stated in the instructions. Syntax must be correct.	2
Form added to the web page containing 4 input fields and a submit button. Appropriate attributes are included and form must be "sticky" as stated in the instructions. Syntax must be correct.	5
PHP delimiters are present within all PHP scripts as stated in the instructions. Syntax must be correct.	1
Variables are present as stated in the instructions. Syntax must be correct.	5
An if conditional is present that checks to see if the form has been submitted as stated in the instructions. Syntax must be correct.	1
PHP validation is performed using an if conditions statement and check to see if the inputs fields were completed correctly as stated in the instructions. Syntax must be correct.	5
Information is written to a text file using the is_writable() and file_put_contents() functions as stated in the instructions. Syntax must be correct.	2
Error messages are displayed if the user does not fill out the form properly as stated in the instructions. Syntax must be correct.	3
A success message and hyperlinks are displayed upon filling out the form without errors as stated in the instructions. Syntax must be correct.	3
Correct use of PHP commenting throughout the documents where indicated within the instructions. Syntax must be correct.	1

Correct display of content/output to the Web browser. This includes all PHP, HTML, and CSS (if applicable).	5
The Course Homepage text is hyperlinked to link to your course homepage with the footer tag of the document. A relative path must be used.	1
Successfully validated HTML5 document and CSS (where applicable). Validation must be successful to receive the full points.	5
TOTAL POSSIBLE POINTS:	**40**