

Name: Terrence Jackson

Date: 10.08.2024

Course : CMSC 340 Web Programming

Week: 8

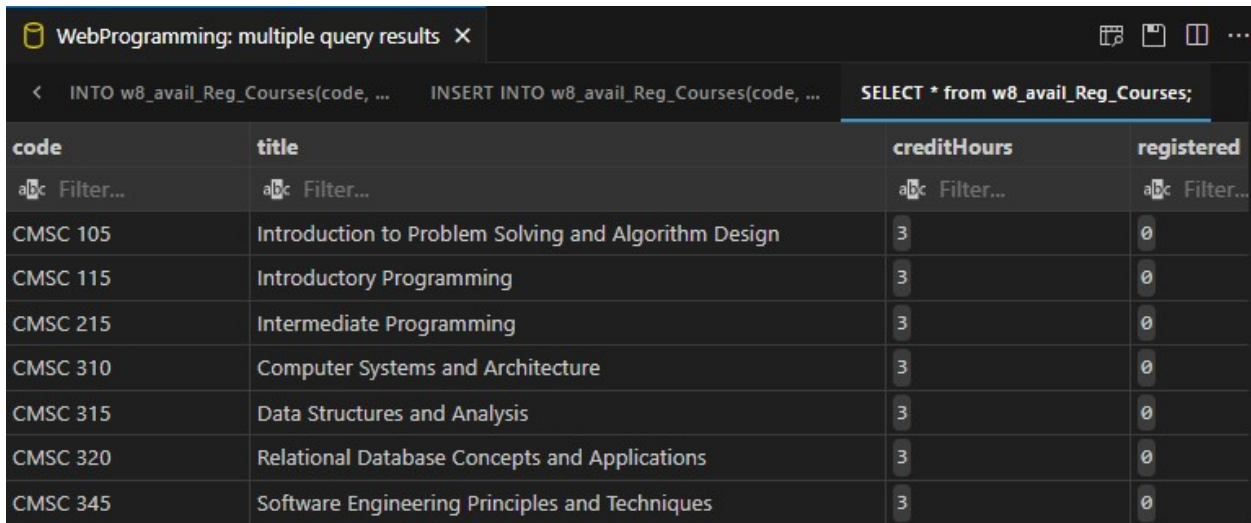
Assignment: Modify a Database-driven Web Application that Registers Students for Courses to Add both JavaScript and PHP Validations

Rubric Criteria:

Number of correctly implemented specifications of confirming operation of given web application without validation out of four (4) specifications (10%)

Insert here a screenshot(s) of the result of confirming operation of given web application without validation:

1. Run the SQL-Script-W8-A1-JS-PHP-Register-for-Courses.sql to load the data needed by the web application.



The screenshot shows a database query results window titled "WebProgramming: multiple query results". The query executed is "SELECT * from w8_avail_Reg_Courses;". The results are displayed in a table with the following columns: code, title, creditHours, and registered. The table contains 7 rows of data.

code	title	creditHours	registered
CMSC 105	Introduction to Problem Solving and Algorithm Design	3	0
CMSC 115	Introductory Programming	3	0
CMSC 215	Intermediate Programming	3	0
CMSC 310	Computer Systems and Architecture	3	0
CMSC 315	Data Structures and Analysis	3	0
CMSC 320	Relational Database Concepts and Applications	3	0
CMSC 345	Software Engineering Principles and Techniques	3	0

2. Screenshot of browser window displaying the result of Register-for-Courses.php.

localhost/W8-A1/Register-for-C X +

localhost/W8-A1/Register-for-Courses.php ☆

Register for Next Term Courses Form

Select a course from these available courses for registration:

CMSC 105 Introduction to Problem Solving and Algorithm Design

CMSC 115 Introductory Programming

CMSC 215 Intermediate Programming

CMSC 310 Computer Systems and Architecture

CMSC 315 Data Structures and Analysis

Register for the Selected Course

These are the course you registered for thus far:

Total credit hours thus far:

9

3. Screenshot of browser window displaying the result of registering for only three (3) non-duplicate courses.

localhost/W8-A1/Register-for-C X +

localhost/W8-A1/Register-for-Courses.php ☆

Register for Next Term Courses Form

Select a course from these available courses for registration:

CMSC 105 Introduction to Problem Solving and Algorithm Design

CMSC 115 Introductory Programming

CMSC 215 Intermediate Programming

CMSC 310 Computer Systems and Architecture

CMSC 315 Data Structures and Analysis

Register for the Selected Course

These are the course you registered for thus far:

CMSC 105 Introduction to Problem Solving and Algorithm Design

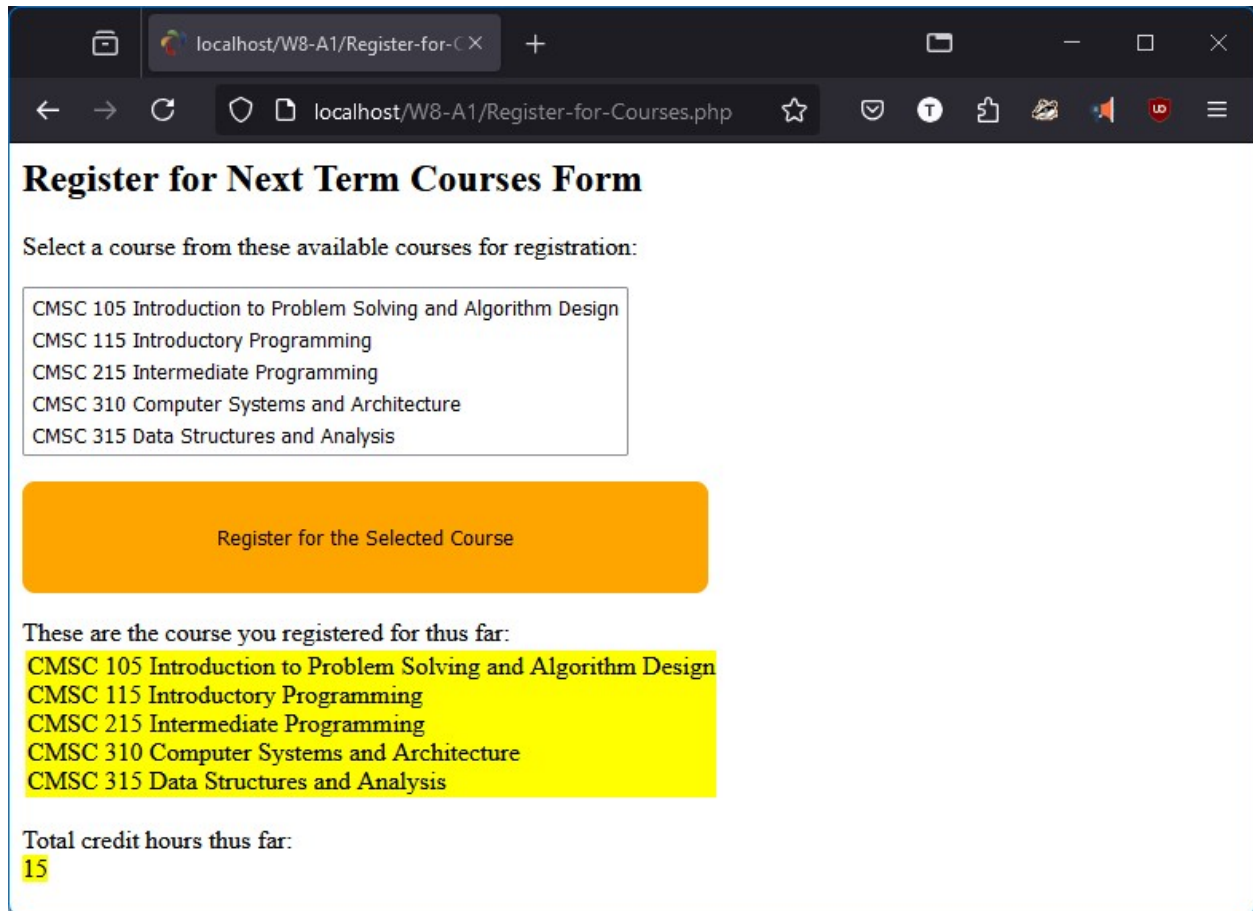
CMSC 215 Intermediate Programming

CMSC 315 Data Structures and Analysis

Total credit hours thus far:

9

4. Take a screenshot of your browser window displaying the result of registering for more than three (3) non-duplicate or duplicate courses.



Rubric Criteria:

Number of relevant explanations out of three (3) explanations of approach, design, and steps of completing this part of the assignment (10%)

Your response here:

Approach

For this part of the assignment, my goal was to verify the functionality of the provided web application without applying any registration rules validation. This involved checking that the SQL script correctly loaded the necessary data into the MySQL database and that the PHP files were properly deployed on my AMPPS web server. My focus was on ensuring that the web application displayed the registration form and allowed me to select courses, register them, and view the results in the browser, all before any modifications or advanced validation.

Design

The design of the provided code centers around a course registration form that interacts with the MySQL database. The form uses an HTML <select> element, dynamically populated with available courses fetched from the database, allowing users to select and register for courses. Each course selection updates a list of registered courses and their total credit hours, which is displayed to the user. The design ensures the basic flow of course selection, registration, and information display works, without enforcing any restrictions on the number of courses registered.

Steps

I began by running the provided SQL script to load the course data into the MySQL database. I then deployed the given web application files to my web server, ensuring they were all placed in the correct directory. I navigated to the Register-for-Courses.php file using my web browser and verified that the course registration form displayed correctly with data from the database. Then I tested the registration process by selecting three non-duplicate courses, confirmed that the registered courses and total credit hours updated. Finally, I registered for more than three courses to confirm there were no registration rules applied.

Rubric Criteria:

Number of correctly implemented specifications of JavaScript client-side validation out of five (5) specifications (15%)

Embed here (or submit separately) a copy of your JavaScript script file Validate.js:

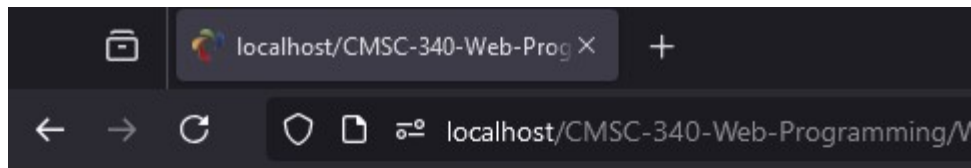
Submitted separately.

Copy and paste here the text of your two JavaScript validation functions:

```
function validate_NoMoreThan9CreditHours() {  
    currentCreditHours = document.getElementById("currentCreditHoursElementID");  
    if (parseInt(currentCreditHours.textContent) >= 9) {  
        return "You cannot register for more than 9 credit hours per term.\n";  
    }  
    return "";  
}  
  
function validate_NotPreviouslyRegistered() {  
    // Get the text from the select element to find the selected course  
    select = document.getElementById("selectCourseElementID");  
    selectIndex = select.selectedIndex; // find the index of the selected element  
    selectedCourse = select.options.item(selectIndex).text; // pull out the text  
    // Get the text from the td element listing registered courses and turn it into an array  
    selectedCourses = document.getElementById("currentCoursesElementID");  
    selectedCoursesText = selectedCourses.innerHTML; // get the HTML text of the element  
    selectedCoursesText = selectedCoursesText.replace(/<td>/, ""); // remove outer td tags  
    selectedCoursesArray = selectedCoursesText.split("<br>"); // split elements on br tags  
    // Loop over the array to find out whether already registered  
    for (i = 0; i < selectedCoursesArray.length; i++) {  
        loopCourse = selectedCoursesArray[i];  
        if (selectedCourse == loopCourse) {  
            return "You are already registered for the " + selectedCourse;  
        }  
    }  
    return "";  
}
```

Insert here a screenshot(s) of the result of executing your JavaScript script client-side validation:

1. Take a screenshot of your browser window displaying the result of browsing to the URL of your Register-for-Courses.php.



Register for Next Term Courses Form

Select a course from these available courses for registration:

CMSC 105 Introduction to Problem Solving and Algorithm Design
CMSC 115 Introductory Programming
CMSC 215 Intermediate Programming
CMSC 310 Computer Systems and Architecture
CMSC 315 Data Structures and Analysis

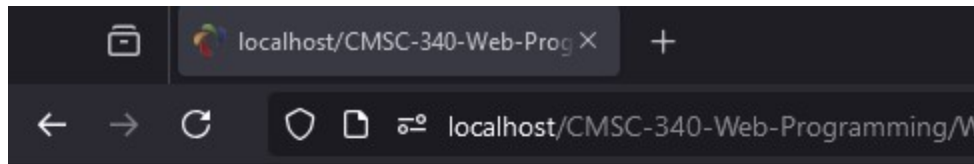
Register for the Selected Course

These are the course you registered for thus far:

Total credit hours thus far:

0

2. Take a screenshot of your browser window displaying the result of registering for only three (3) non-duplicate courses.



Register for Next Term Courses Form

Select a course from these available courses for registration:

CMSC 105 Introduction to Problem Solving and Algorithm Design
CMSC 115 Introductory Programming
CMSC 215 Intermediate Programming
CMSC 310 Computer Systems and Architecture
CMSC 315 Data Structures and Analysis

Register for the Selected Course

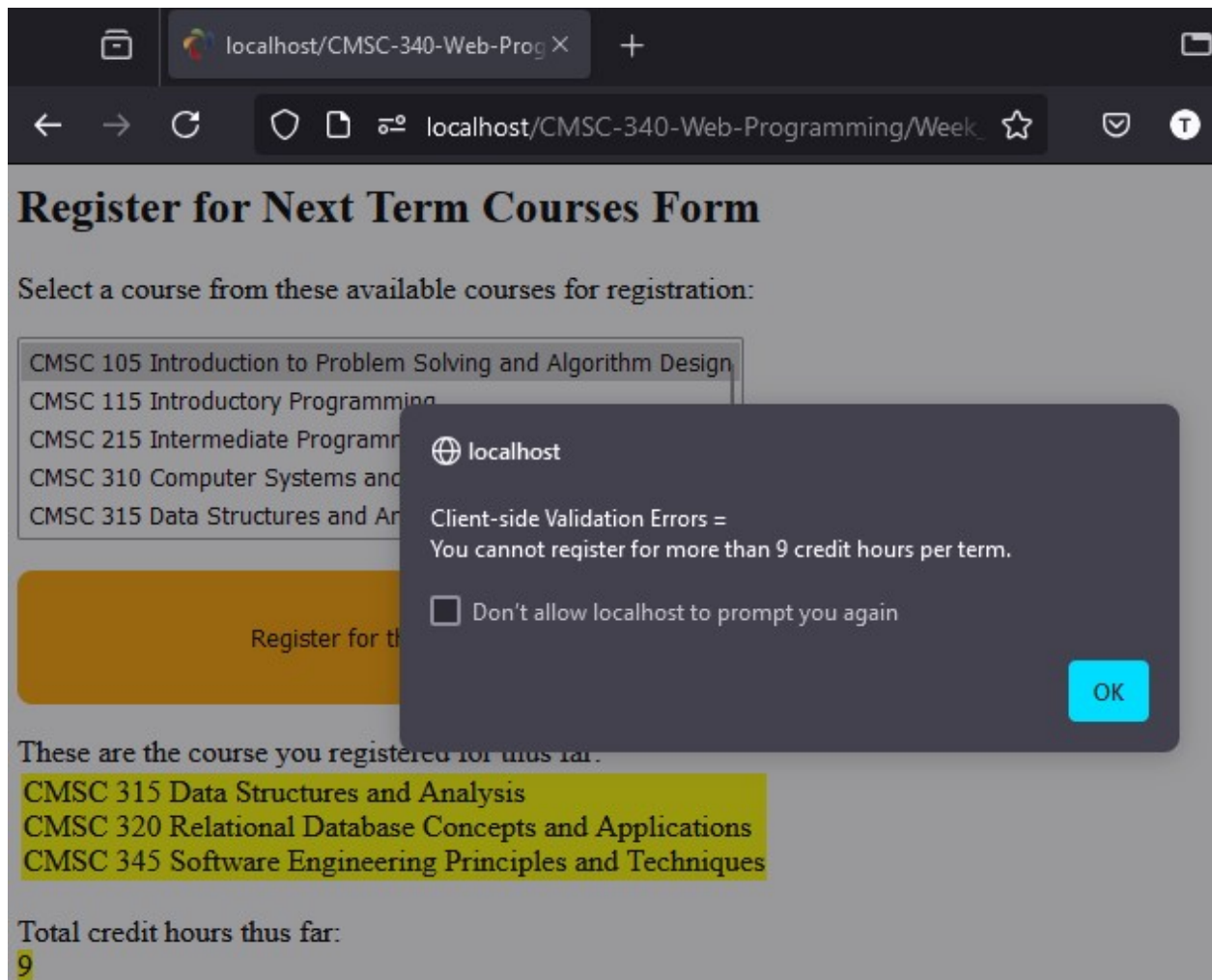
These are the course you registered for thus far:

CMSC 315 Data Structures and Analysis
CMSC 320 Relational Database Concepts and Applications
CMSC 345 Software Engineering Principles and Techniques

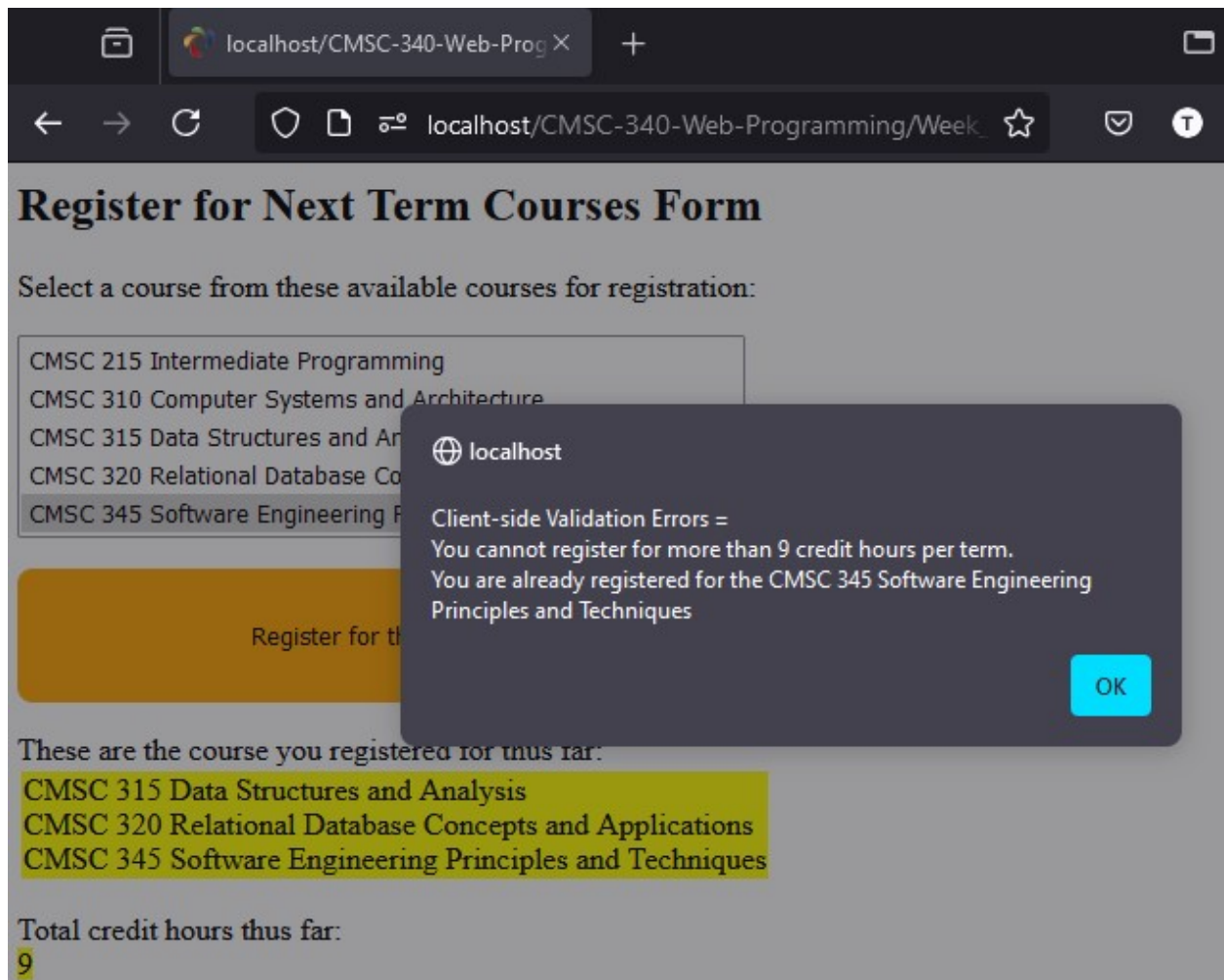
Total credit hours thus far:

9

3. Take a screenshot of your browser window displaying the result of registering for more than three (3) non-duplicate or duplicate courses.



4. Take a screenshot of your browser window displaying the result of registering for a duplicate course that you have previously registered for.



Rubric Criteria:

Number of relevant explanations out of three (3) explanations of approach, design, and steps of completing this part of the assignment (10%)

Your response here:

Approach

For the client-side validation, I focused on implementing real-time checks using JavaScript to enhance the user experience by preventing invalid registrations before the data was sent to the server. My approach was to enforce the same two registration rules as in the server-side validation: no more than 9 credit hours and no duplicate course registrations. I chose to tackle this after completing the server-side validation to ensure the logic was sound, and then shifted focus to improving the user interaction without having to disable JavaScript for testing later on.

Design

For the credit hour check, I designed the function to retrieve the current total credit hours from the HTML element displaying them and compare it to the maximum of 9. If exceeded, it triggers an alert with the appropriate error message. For the duplicate course validation, I used JavaScript to access the course selected in the <Select> element, compare it against the courses already registered, and if a match is found, display an alert indicating the user cannot register for the same course twice. These validations were integrated into the form submission process, ensuring that the form would only be submitted if no errors were found.

Steps

I started by running the SQL script again to load a fresh set of course data into the database. Then, I verified that the Register-for-Courses.php page displayed correctly by accessing it via my web browser and taking the required screenshots of the form with no registrations. Next, I modified the Validate.js file by first implementing the validate_NoMoreThan9CreditHours() function, which checked the total registered credit hours and ensured the user couldn't register for more than 9. After debugging issues where the changes weren't appearing due to the JavaScript file being cached, I resolved the issue and confirmed that the correct validation was applied. Then, I implemented the validate_NotPreviouslyRegistered() function, which checked whether the selected course had already been registered and prevented duplicates by comparing the selected course to a list of registered courses extracted from the page's HTML. Finally, I tested the form by registering multiple courses and verifying that both validation rules triggered the appropriate JavaScript alerts when violated. After ensuring everything worked correctly, I took the necessary screenshots.

Rubric Criteria:

Number of correctly implemented specifications of PHP server-side validation out of five (5) specifications (15%)

Embed here (or submit separately) a copy of your PHP script file Validate.php:

Submit separately

Copy and paste here the text of your two PHP validation functions:

```
<?php
```

```
function validate_NoMoreThan9CreditHours($avail_Reg_Courses)
```

```
{
```

```
    $total = 0;
```

```
    foreach ($avail_Reg_Courses as $i) {
```

```
        if ($i->getRegistered()) {
```

```
            $total += $i->getCreditHours();
```

```
        }
```

```
    }
```

```
    if ($total >= 9) {
```

```
        return "You cannot register for more than 9 credit hours per term.<br>";
```

```
    }
```

```
    return "";
```

```
}
```

```
function validate_NotPreviouslyRegistered($avail_Reg_Courses, $selectCourseElementName)
```

```
{
```

```
    foreach ($avail_Reg_Courses as $i) {
```

```
        if ($i->getRegistered() && $i->getCode() == $selectCourseElementName) {
```

```
            return "You are already registered for the $selectCourseElementName.<br>";
```

```
        }
```

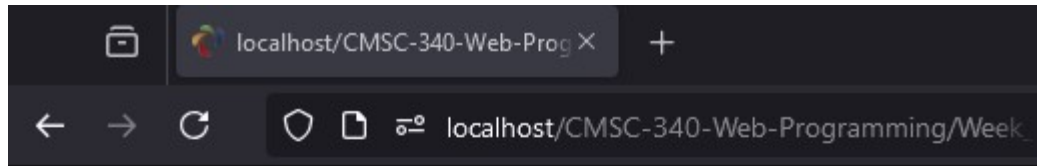
```
    }
```

```
    return "";
```

```
}
```

Insert here a screenshot(s) of the result of executing your PHP script server-side validation:

1. Take a screenshot of your browser window displaying the result of browsing to the URL of your Register-for-Courses.php.



Register for Next Term Courses Form

Select a course from these available courses for registration:

CMSC 105 Introduction to Problem Solving and Algorithm Design
CMSC 115 Introductory Programming
CMSC 215 Intermediate Programming
CMSC 310 Computer Systems and Architecture
CMSC 315 Data Structures and Analysis

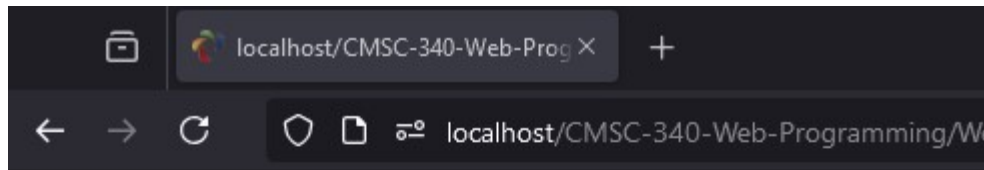
Register for the Selected Course

These are the course you registered for thus far:

Total credit hours thus far:

0

2. Take a screenshot of your browser window displaying the result of registering for only three (3) non-duplicate courses.



Register for Next Term Courses Form

Select a course from these available courses for registration:

CMSC 105 Introduction to Problem Solving and Algorithm Design
CMSC 115 Introductory Programming
CMSC 215 Intermediate Programming
CMSC 310 Computer Systems and Architecture
CMSC 315 Data Structures and Analysis

Register for the Selected Course

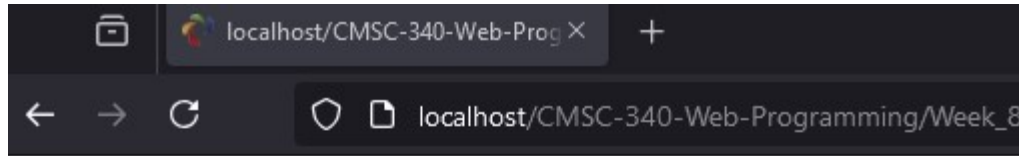
These are the course you registered for thus far:

CMSC 105 Introduction to Problem Solving and Algorithm Design
CMSC 315 Data Structures and Analysis
CMSC 345 Software Engineering Principles and Techniques

Total credit hours thus far:

9

3. Take a screenshot of your browser window displaying the result of registering for more than three (3) non-duplicate or duplicate courses.



Server-side Validation Errors =

You cannot register for more than 9 credit hours per term.

Register for Next Term Courses Form

Select a course from these available courses for registration:

CMSC 105 Introduction to Problem Solving and Algorithm Design
CMSC 115 Introductory Programming
CMSC 215 Intermediate Programming
CMSC 310 Computer Systems and Architecture
CMSC 315 Data Structures and Analysis

Register for the Selected Course

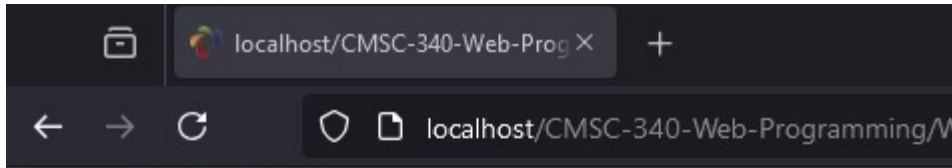
These are the course you registered for thus far:

CMSC 105 Introduction to Problem Solving and Algorithm Design
CMSC 315 Data Structures and Analysis
CMSC 345 Software Engineering Principles and Techniques

Total credit hours thus far:

9

4. Take a screenshot of your browser window displaying the result of registering for a duplicate course that you have previously registered for.



Server-side Validation Errors =
You are already registered for the CMSC 310.

Register for Next Term Courses Form

Select a course from these available courses for registration:

CMSC 105 Introduction to Problem Solving and Algorithm Design
CMSC 115 Introductory Programming
CMSC 215 Intermediate Programming
CMSC 310 Computer Systems and Architecture
CMSC 315 Data Structures and Analysis

Register for the Selected Course

These are the course you registered for thus far:

CMSC 105 Introduction to Problem Solving and Algorithm Design
CMSC 310 Computer Systems and Architecture

Total credit hours thus far:

6

Rubric Criteria:

Number of relevant explanations out of three (3) explanations of approach, design, and steps of completing this part of the assignment (10%)

Your response here:

Approach

I decided to implement server-side validation first to avoid conflicts with client-side validation. This allowed me to focus on ensuring the core logic for enforcing registration rules was securely handled on the server, independent of the client's JavaScript settings. By validating on the server side, I ensured that even if JavaScript is disabled or tampered with, the rules of not registering for more than 9 credit hours and preventing duplicate course registration are still enforced effectively. This approach also streamlined testing by avoiding the need to toggle JavaScript on and off later.

Design

I designed the `validate_NoMoreThan9CreditHours()` function to calculate the total number of credit hours by iterating over the `$avail_Reg_Courses` array and checking the `getRegistered()` method for each course. This decision ensures that only courses the student has already registered for contribute to the total. If the total credit hours reach or exceed 9, the function returns a validation error message, preventing further registration. For the `validate_NotPreviouslyRegistered()` function, I used a similar structure of iterating through the `$avail_Reg_Courses` array. The function checks each course's registration status and compares the course code with the selected course name to ensure the student is not registering for the same course twice. If a match is found, the function returns a specific error message indicating the student is already registered for that course.

Steps

I started by re-running the SQL script to load a fresh set of course data into the MySQL database. Then, I modified the `Validate.php` file. For the `validate_NoMoreThan9CreditHours()` function, I added code to check if the total registered credit hours exceeded 9 and, if so, returned the error message. Next, I attempted to register for more than 9 credit hours and confirmed the PHP validation error displayed properly and took a screenshot. Finally, I modified the `validate_NotPreviouslyRegistered()` function to check if the user was trying to register for a course they had already signed up for. If a duplicate course was detected, it returns an error message. I tested this functionality by attempting to register for the same course twice and confirmed the error message appeared. I took a final screenshot to document the validation.

Rubric Criteria:

Number of objective self-evaluations out of three (3) self-evaluations of strengths, weaknesses, and areas of improvement of self-performance on the assignment (10%)

Your response here:

Strengths

My decision to implement server-side validation before client-side validation in my Web Development project reflects strategic thinking, as it ensured that I could thoroughly test server-side functionality without interference from client-side code. I also consistently use debugging techniques, like printing elements to the console, to resolve issues. These habits of planning, thoughtful sequencing of tasks, and thorough testing demonstrate my problem-solving abilities.

Weaknesses

One area where I tend to struggle is when unexpected technical issues arise, such as the JavaScript caching problem during the client side validation. Although I was eventually able to resolve the issue, the time spent troubleshooting slowed down my progress and caused some frustration. Technical glitches like this can sometimes leave me feeling stuck, especially when they aren't immediately obvious.

Improvement

I could improve by developing faster troubleshooting techniques, particularly when dealing with unexpected technical issues. Learning more about browser development tools or caching mechanisms would help reduce the time I spend resolving these kinds of problems, as was the case with the cached JavaScript file. Gaining more experience with advanced debugging and web development environments would increase my overall efficiency in handling complex projects.

Rubric Criteria:

**Number of relevant reflections on the learning experience out of three (3) reflections of setbacks, triumphs, and lessons learned
10%**

Your Response here:

Setbacks

My big setback was the caching issue with my JavaScript file. The problem delayed my progress, as the changes I made to the JavaScript code were not being reflected on the web page, leading to unnecessary frustration. It was a technical issue I hadn't anticipated, and it forced me to slow down and investigate the problem before I could proceed. This setback taught me the importance of understanding the intricacies of the development environment and being prepared for unforeseen obstacles, where browser behavior can significantly impact the outcome of a project.

Triumphs

One major success was successfully implementing server-side validation first, which streamlined the overall process and made the client-side validation easier to manage afterward. This saved me from the potential frustration of needing to disable JavaScript to test server-side functionality. Similarly, after troubleshooting and solving the caching issue, I was able to quickly get back on track, debugging and implementing the required functionality in my JavaScript code.

Lessons Learned

One key lesson I learned is the importance of understanding and managing the development environment in web programming. The issue with my JavaScript file being cached taught me the value of double-checking for technical factors beyond the code itself. This experience underscored the need to be adaptable and persistent, while also reinforcing the importance of having a solid understanding of the tools and platforms I'm working with.

Additionally, I learned that thoughtful planning can significantly impact the efficiency of a project. My decision to handle server-side validation first, before implementing client-side validation, allowed me to maintain focus and avoid potential roadblocks later on. This lesson in strategic task management has strengthened my belief in the value of planning and sequencing tasks logically, whether in coding or in other areas of my work.