Reference Materials for Assignment 02

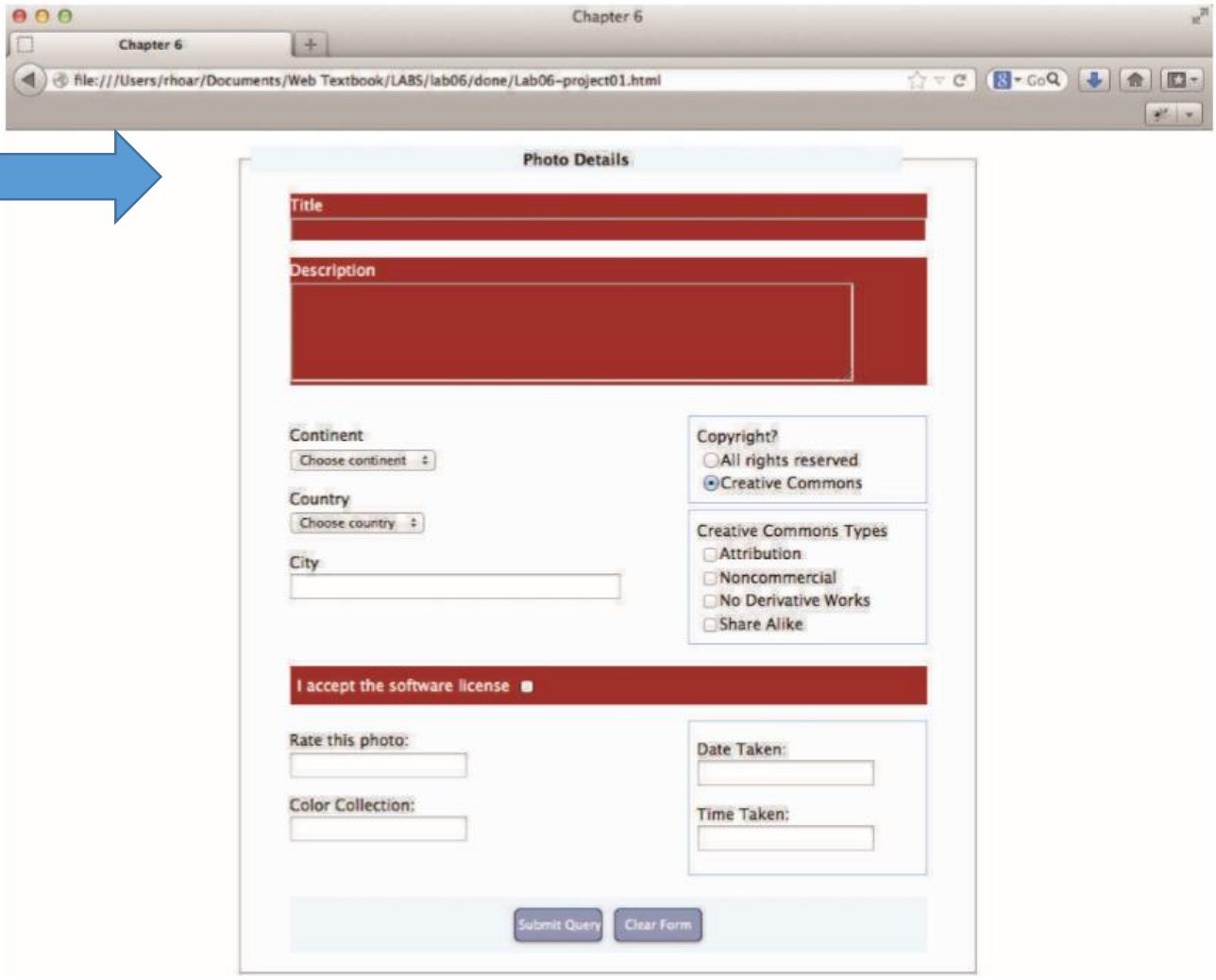
Modification Requested

Project 1

Project 2

Project 3

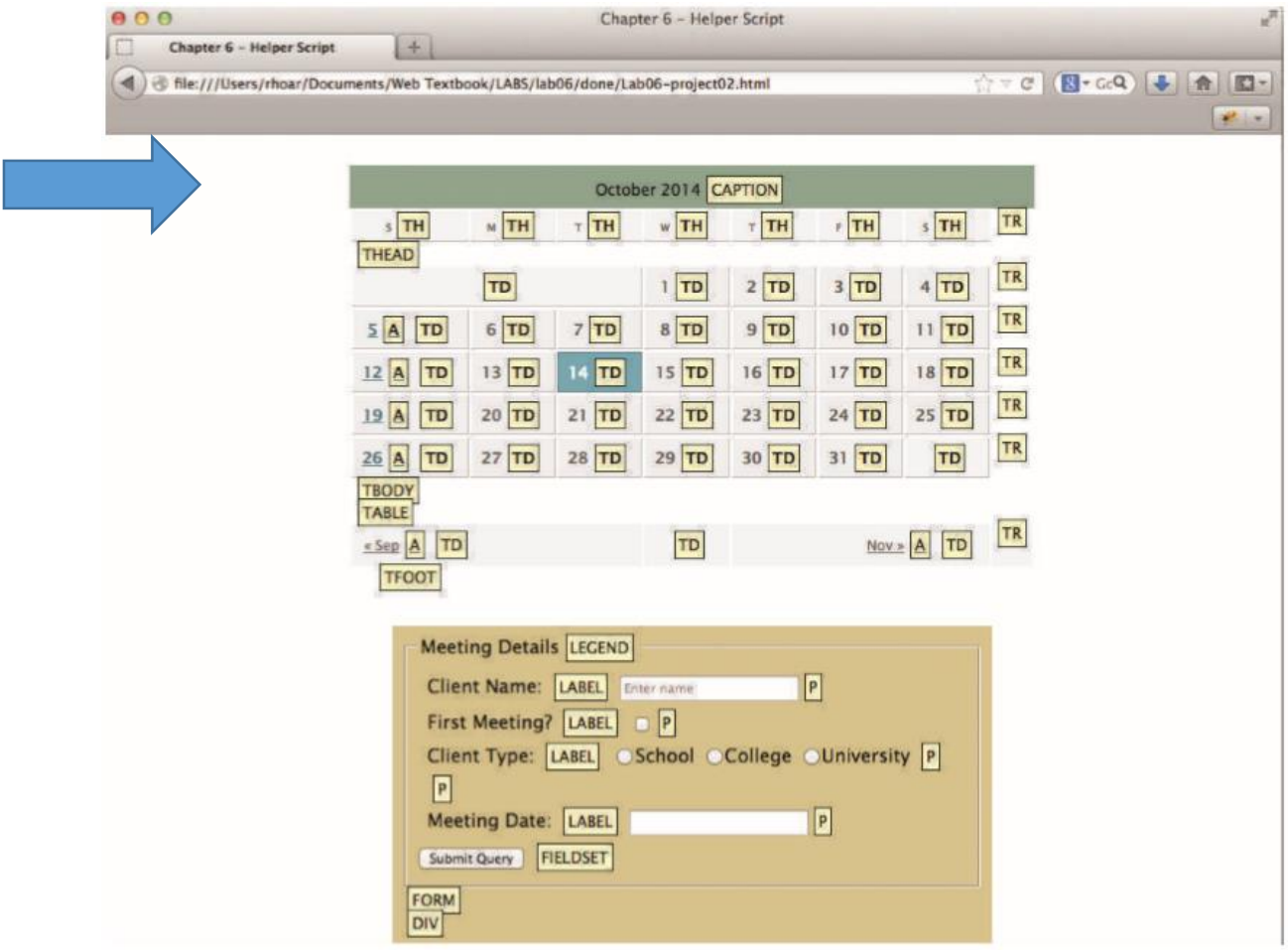
Project 1



Modification requested:

1. Replace title “Photo Details” with “your student ID number and your family name in English”

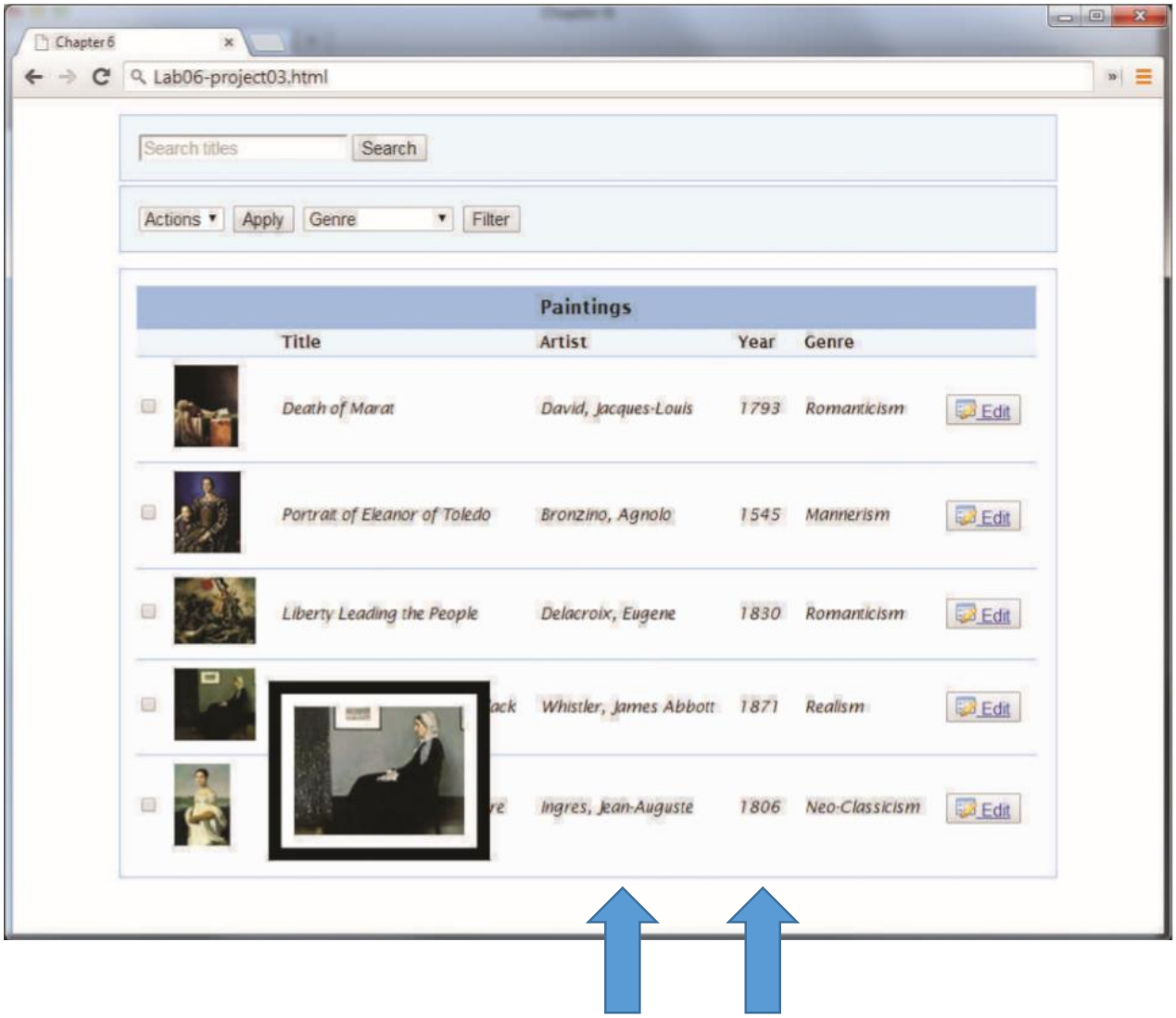
Project 2



Modification requested:

1. Replace “October 2014” with “your student ID number November 2022”.

Project 3



Modification requested:

1. Replace artist name with your name.
2. Replace year with 2022.

Chapter 6

Project 1

Project 2

Project 3

PROJECT 1: Simple Login Form Prevalidation

DIFFICULTY LEVEL: Beginner

Overview

You will create JavaScript prevalidation for the form in Chapter06-project01.html. This project builds on Chapter 4 Project 3 (the Photo sharing site upload form).

Instructions

1. You will need to link to an external JavaScript file in the head of the page so that you can write code in its own file.

FIGURE 6.21 Screenshot of the Photo form, being prevalidated to detect blank fields

2. You should define a CSS style to use when highlighting a blank field.
3. Set up a listener on the form's submit event so that the code prevents submission of the form (`preventDefault()`) if either the title or description field is left blank or the accept license box is not checked, but otherwise submits the form.
4. Enhance the JavaScript so that blank fields trigger a change in the appearance of the form (using the style defined earlier).
5. Add another listener to the fields so that when the user types into a field (changed event) JavaScript removes the red color you just added.

Test

1. Test the form in the browser. Try submitting the form with either field blank. You should see the field highlighted and notice the page will not be refreshed as shown in Figure 6.21.
2. Type into one of the highlighted fields, and the error color should be immediately removed.

PROJECT 2: Write a Node Highlighting Helper Script

DIFFICULTY LEVEL: Intermediate

Overview

This exercise will be to write a helper script that could theoretically be used on any web page to help identify the elements, simply by including your JavaScript file! For the sake of illustration we will use Chapter 4 Project 1 as the basis for testing.

FIGURE 6.22 Screenshot of the helper script in action on the book database calendar page

Instructions

1. Define the script in a source file called highlightNodes.js.
2. This script should navigate every element in the DOM, and for each element in the body determine whether it is a textNode (type 3) or not.
3. Now add to your script code to create a new child node for every non-text node encountered. This new node should take on the class `"hoverNode"` and `innerHTML` equal to the parent tag name. Define appropriate styles for that CSS class.
4. Now add listeners so that when you click on the newly created nodes, they will alert you to information about the tag name, so that when a node is clicked a pop-up alerts us to the details about that node including its `ID` and `innerHTML`.

Test

1. By loading this script onto any page, all the tags should be identified and yellow boxes pop-up as shown in Figure 6.22.
2. Reflect on how you could enhance this script into a more useful tool to help with web development and debugging.

PROJECT 3: Progressive Enhancement Art Gallery Search

DIFFICULTY LEVEL: Intermediate

Overview

You will build upon your existing HTML page designed in Chapter 4 Project 2, but replace the existing content with an enhanced version (progressive enhancement) where a small piece of JavaScript is added so that every image can be hovered on to see a larger thumbnail as needed.

FIGURE 6.23 Screenshot of the progressive enhancement mouseover image close-up

Instructions

1. Like the previous two projects, begin by adding a link to a JavaScript file in the head of your page (or right before the `</body>` tag).
2. Slightly modify the HTML to add a class for each thumbnail image.
3. In your JavaScript file, write a loop to seek out all the img tags with the newly defined class (hint: `querySelectorAll()`).
4. For each image, attach a listener on the `mouseIn` event to create a new `<span>` with a larger image inside (based on the `src` attribute). Add another listener on the `mouseOut` event to hide the newly created `<span>`.

Test

1. Reload the page, and see that as you hover over images, larger quality thumbnails are fetched and seen in a `<div>` over the mouse location as shown in Figure 6.23.
2. As you move your mouse out the page should return to the way it was before you hovered.

Appendix

For your reference: Chap4 projects

4.7.3 Hands-On Practice

PROJECT 1: Book Rep Customer Relations Management

DIFFICULTY LEVEL: Beginner

Overview

Edit Chapter04-project01.html and Chapter04-project01.css so the page looks similar to that shown in Figure 4.30.

Instructions

1. You will need to create the calendar month using tables and provide the styling.
2. The month and date of the calendar should be within a `<caption>`.
3. Be sure to use the `<fieldset>` and `<legend>` elements for the form. As well, be sure to use the appropriate accessibility features in the form.
4. Set up the form's `method` attribute to `GET` and its `action` attribute to http://www.randyconnolly.com/tests/process.php.

Test

1. Test the form in the browser. Verify that the output from process.php matches that shown in Figure 4.30.
2. Change the form method to `POST` and retest.

PROJECT 2: Art Store

DIFFICULTY LEVEL: Intermediate

Overview

Edit Chapter04-project02.html and Chapter04-project02.css so the page looks similar to that shown in Figure 4.31.

Instructions

1. The form at the top of this page consists of a text box, and two drop-down lists. For the Genre list, make the other choices "Baroque," "Renaissance," and "Realism." The drop-down list items should have numeric values starting with O. Notice the placeholder text in the search box.
2. Create a table of paintings that looks similar to that shown in Figure 4.31. Be sure to make the table properly accessible.
3. The checkboxes in the table should be an array of elements, e.g., `<input type="checkbox" name "index[]" value="10" />`. The name and values are arbitrary, but each checkbox needs to have a unique value.
4. The button in each row is a `<button>` element with a dummy link.
5. Set the form's `method` attribute to `POST` and its `action` attribute to http://www.randyconnolly.com/tests/process.php.

Test

1. Test the form in the browser. Verify that the output from process.php matches that shown in Figure 4.31.

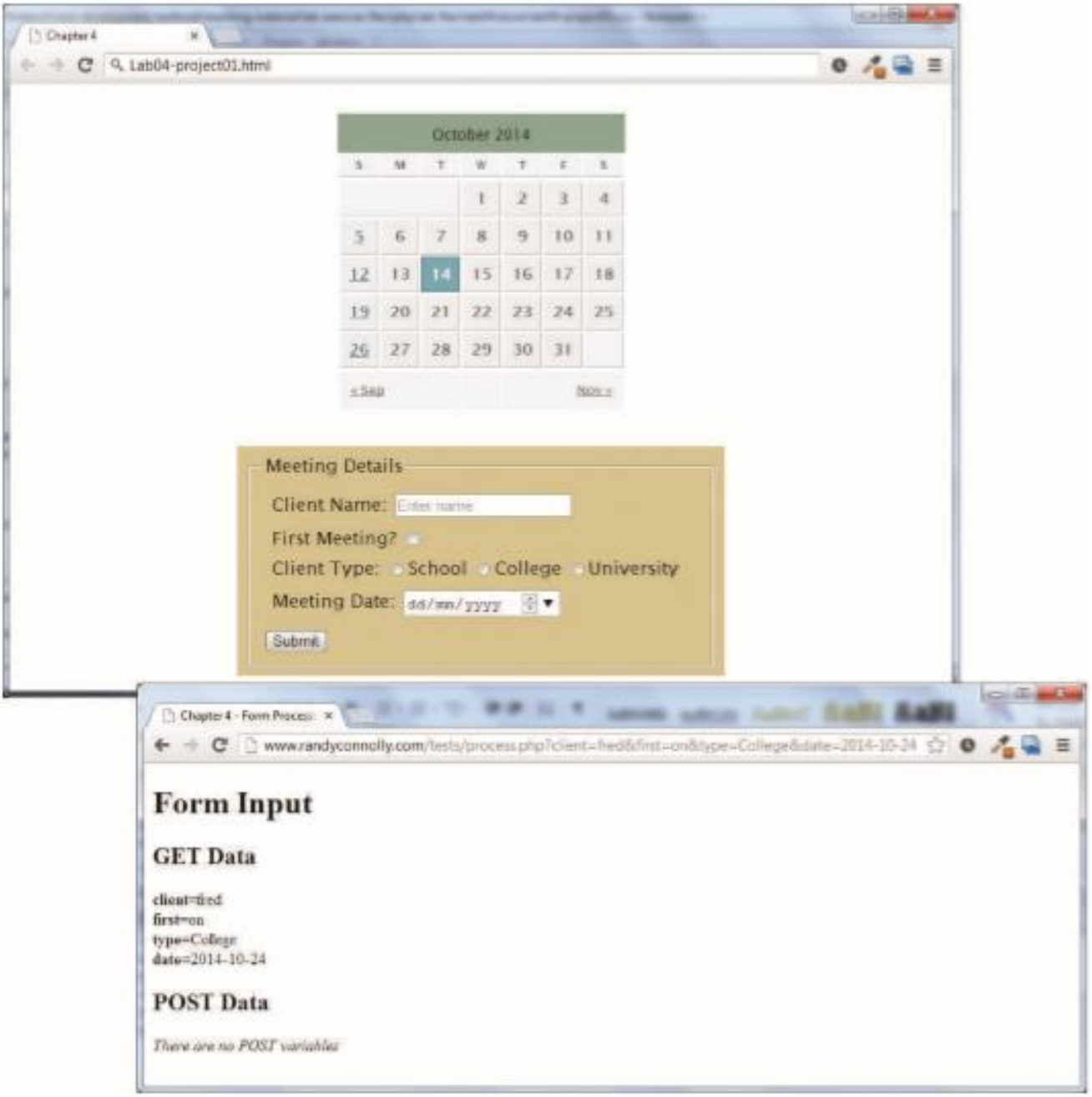


FIGURE 4.30 Completed Project 1

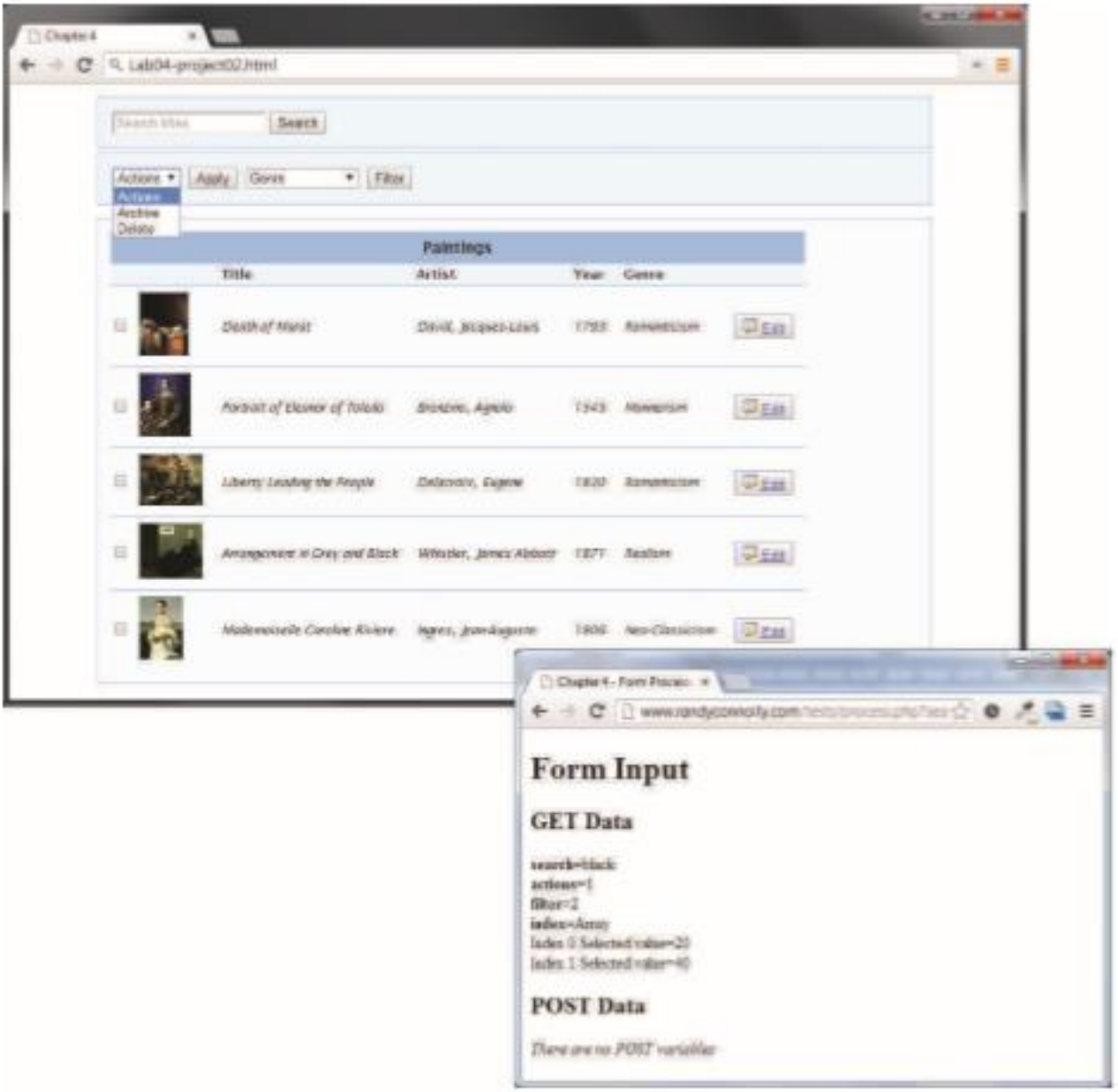


FIGURE 4.31 Completed Project 2

PROJECT 3: Share Your Travel Photos

DIFFICULTY LEVEL: Advanced

Overview

Edit Chapter04-project03.html and Chapter04-project03.css so the page looks similar to that shown in Figure 4.32.

1. Create the form and position the elements by placing them within a table. While we do not believe that this is best practice, legacy sites often use tables for layout so it may be sensible to get some experience with this approach. In the next chapter, you will learn how to use CSS for layout as a better alternative.

2. For the drop-down lists, add a few sensible items to each list. For the checkbox list, they should be an array of elements. Notice also that this form makes use of a number of HTML5 form elements.

Test

1. Test the form in the browser. Verify that the output from process.php matches that shown in Figure 4.32. Because this form uses HTML5 input elements that are not supported by all browsers, be sure to test in more than one browser.

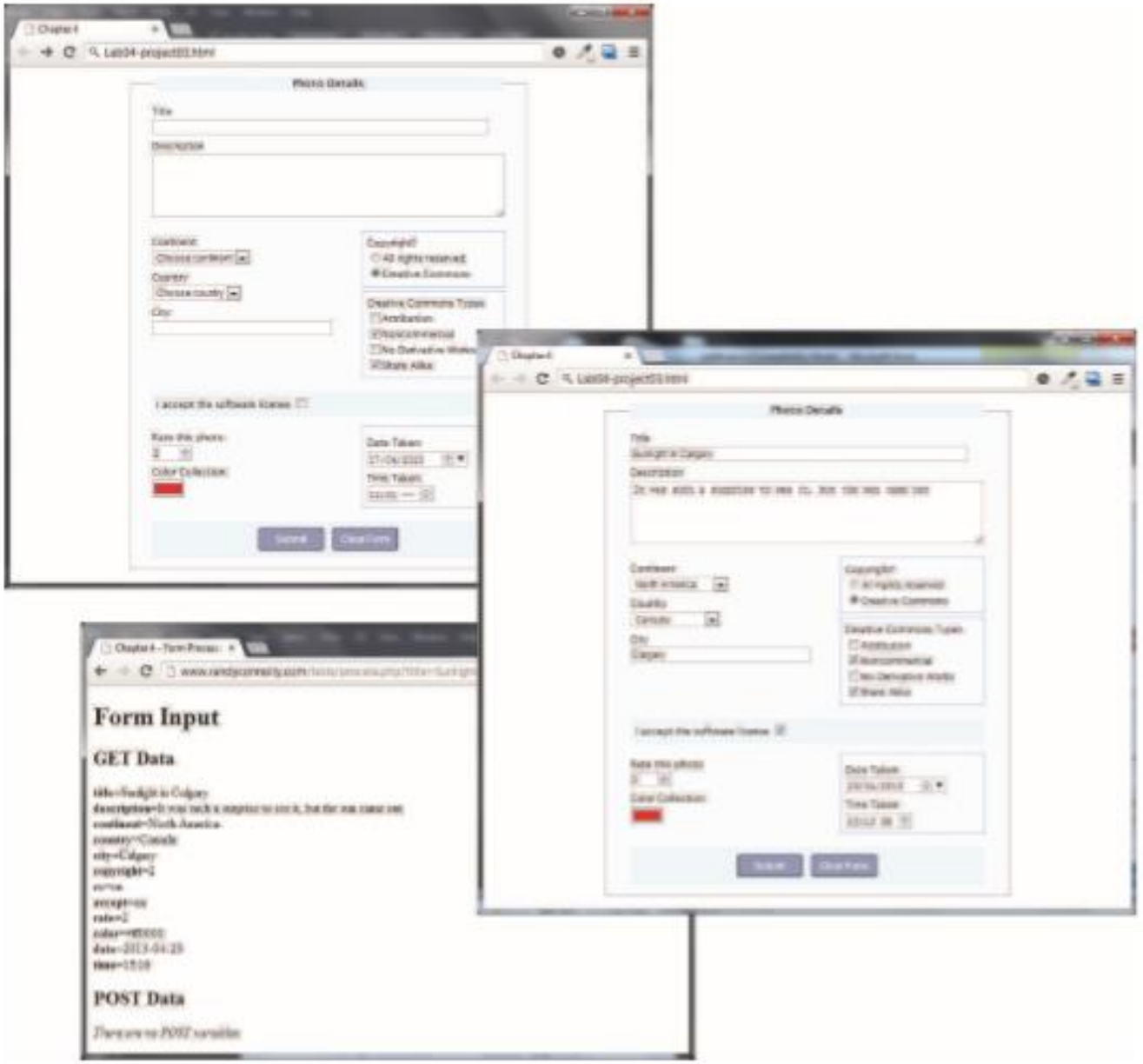


FIGURE 4.32 Completed Project 3