Lab 7: PHP Validation

Learning Objectives

After completion of this lab, **you should be able** to

- Work collaboratively as a **pair programming** team
- **Validate user input** with PHP and **regular expressions**
- Lab 7 Screen Snapshots
- Code ***all-in-one-page*** processing
- Use **Session Variables**
- Make previously entered user data re-display in form elements
- **To Receive Credit**

Work collaboratively as a pair programming team

CS 250 in-class labs will be done using **pair programming**. Your partner for today's lab is listed

in the table below:

Hebeler 204

Grader: John Wright II

Team 1 Abundiz, Sergio Burton, Henry	**Team 2** Bajwa, Deepinder Strom, Brandt	**Team 3** Burley, Jonathan Chandler, Alan	**Team 4** Byars, Frank Dickerson, Andrew	**Team 5** Carpenter, Daniel Juarez, Adrian
Team 6 Crockett, Jordan Plitkins, Kristofer	**Team 7** Hansen, Christopher Prescott, Brandon	**Team 8** Kinkade, Kyle Canada, Justin	**Team 9** Porter Jr, Anthony Ahmady, Temourshah	**Team 10** Rozelle, William Belfiglio, Alexander
Fill in: Taing, Pokuy				

You may wish to review basic pair programming guidelines before you begin.

- One team member (the **driver**) has control of the keyboard/mouse and actively implements the program
- The other team member (the **navigator**) continuously observes the work of the driver to identify tactical defects (such as syntactic and spelling errors, etc.) and also thinks strategically about the direction of the work

You should **change roles** about every ten minutes during lab.

Validate user input with PHP and regular expressions

Most often, web pages use HTML forms for user input. It is very important to validate user input before storing it in a database or other types of processing

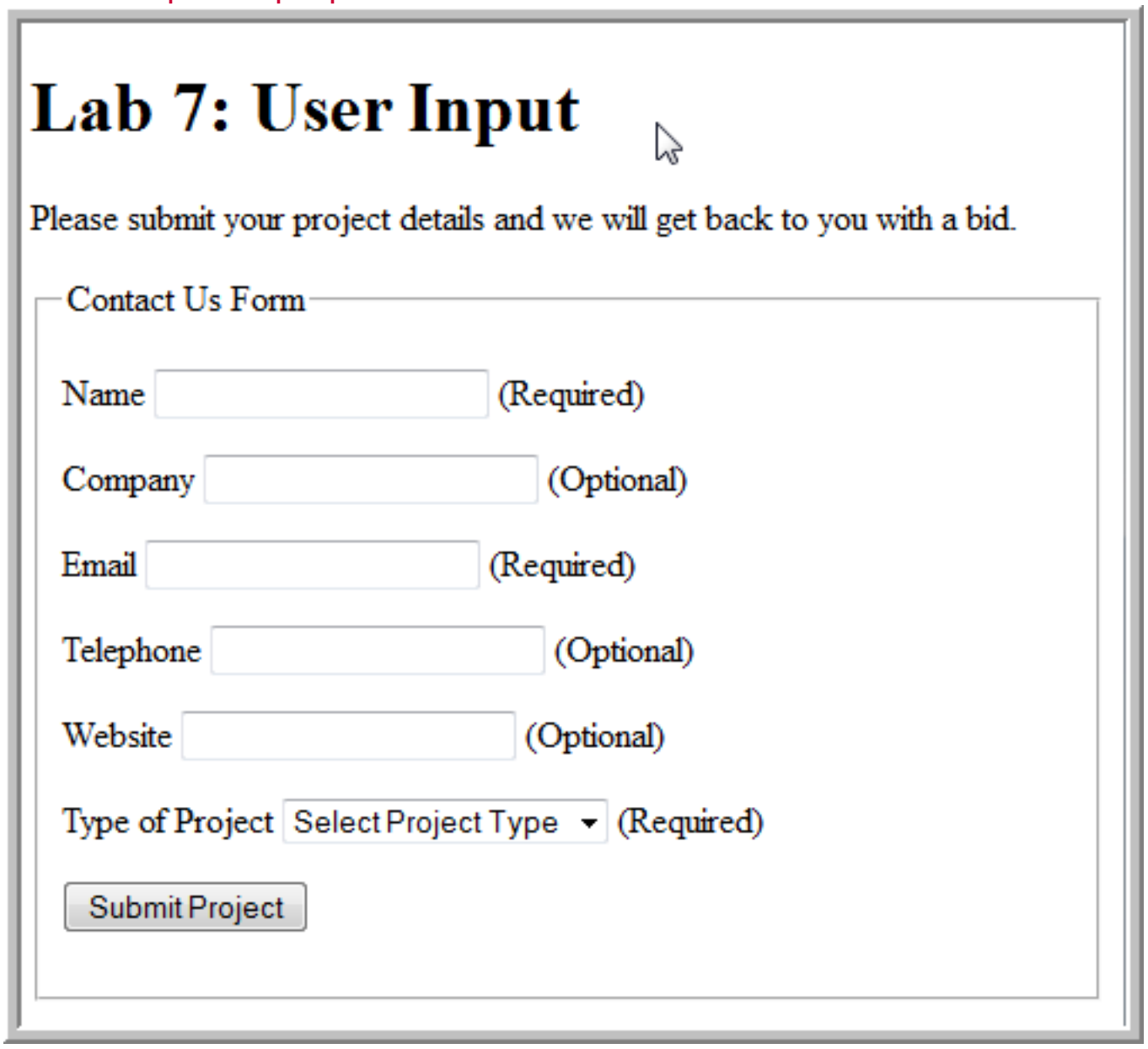
- On `lab7input.php` web page, use an **XHTML Form** with input widgets to allow the user to
  - Enter their **name** (Required. KISS: restrict input to letters or spaces, e.g. Ed Gellenbeck)
  - Enter their **company** (Optional, KISS: allow anything, e.g. W3C)
  - Enter their **email address** (Required. KISS: Validate email to match aa@bb.cc where a,b,c are any letters)
  - Enter their **telephone** (Optional. KISS: Validate to ddd-ddd-dddd or ddd-dddd if present
  - Enter their **website URL** (Optional. KISS: Validate to valid URLs
  - Select their **Project Type** (Required. Must select HTML, CSS, PHP, or other
  - **Note**: to speed up the lab, use the code on this solved **lab7input.php** page
    - This page contains no PHP yet, but save the HTML code as `lab7input.php`
- On `lab7validate.php` Web page, use PHP to validate the user input, displaying error message is there is a problem or displaying the user input if everything is correct
  - Note: to speed up the lab, download and use this skelton lab7validate.php code
  - Complete the PHP function `validate_input()` that **returns a string** containing any error messages or an empty string if the user input completely validates
  - A tool for testing PHP regular expressions is available online
  - 60 standard validation functions is also available online

---

Lab 7 Screen Snapshots

Note: All HTML elements are laid out using the default browser styling. Inline elements are enclosed inside paragraphs `<p>`

`lab7input.php`



`lab7validate.php`

skelton

Code *all-in-one-page* processing

A better approach to handling PHP user input validation is to do it all-in-one-page: user input, validation, and processing.

The overall logic to implement all-in-one-page processing is

```
if this page has just called itself with the user input data
   validate the user input data
```

```
    if the data is valid
       process the data (save in db, etc.)
       go to a confirmation page

    else (the input data has problems)
       show the error messages

 Display the HTML form (re-display any previously entered user
 input data values so the user does not need to retype everything)
```

Modify `lab7input.php` to do ***all-in-one-page*** processing: prompt for input, validate, and process the data (minimally)

1. Copy and paste your function `validate_input()` into the start of `lab7input.php`
2. Change the `action` attribute of the PHP form to call itself (**action=""**)
3. Add the `if else` logic to take ***all-in-one-page*** approach to processing (Code given below)

```
<?php
  // if this page has just called itself with user input data
  //   (i.e. the user pressed the submit button)
  if (isset($_POST['submitButton'])) {

    // validate the user input data
```

```
    $error_messages = validate_input();

    // if user input is valid, process the data and go to
confirmation page
    if ($error_messages == "") {
        $_SESSION['name'] = $_POST['name']; // simplified
processing
        header('Location: thankyou.php');     // go to confirmation
page
        exit();
    }
    else {
      echo "<p>$error_messages</p>"; // show the error messages
    }
  }
?>
<!-- Display the HTML form -->
```

Use Session Variables

The simplified processing of the data above is to save the user's name as a session variable to

re-display on the confirmation page: `thankyou.php`

To use session variables on the lab machines, we need to reset the session save path to U:\htdocs\sessiondata (as well as creating the subfolder sessiondata)

1. Add calls to the session_save_path and session_start php functions to the beginning of `lab7input.php`

```php
<?php
session_save_path ($_SERVER['DOCUMENT_ROOT'] . "/sessiondata/");
session_start();
?>
```

2. Create a short `thankyou.php` page that displays the welcome message: **Thank you *name***, getting name from the SESSION variable set in lab7input.php

---

Make previously entered user data re-display in form elements

In order to re-show the form with user input still present, use the input element's **value** attribute set to the contents of the appropriate $_POST variable. For example,

```
<input id="name" name="name" type="text" value="<?php echo
```

```
    $_POST['name']; ?>" />;
```

To Receive Credit

Labs are graded by the student teaching assistant using the **lab's scoring rubric** [PDF].

Pair programming teams will receive the scoring rubric sheet at the start of lab. Write both names on the sheet to turn in the sheet in when you finish. Name 1 should be the student who saved the pair work in their CS 250 account.

- If you **finish during lab**, turn in to your grader the **Lab's scoring rubric sheet**.
  - If you have **not** yet satisfied all criteria to the level of 4, you need to continue working on the lab outside of class time
  - Your saved solution should be stored in the CS 250 account listed under Name 1
  - Your solution will be checked by the grader using the scoring rubric and both students will receive the same score for the lab assignment
- If you are **unable** to complete the solution during the lab period
  - Leave lab with both students having a copy of the partially completed solution
  - Agree to finish the lab together (establish a time) or independently
  - Turn in the **lab's scoring rubric** [PDF] at the **start of next Lab**
    - Use one rubric for teams finishing together or two rubrics for students finishing independently
    - Keep track of and include the **total completion time** (rounded to closest half-hour) it took to complete the lab assignment and include the time on the rubric

        - Write a score `0..4` in the rubric's **self assessment column** representing your completion status
    - Make sure your work is stored in your CS 250 account in folder `U:\htdocs\labs\lab7\`

Lab 7 is due at the start of lab 8 (in one week). **No** late lab assignments will be accepted without prior approval. Your lowest lab score will be dropped.