PHP Form processing

Overview

This lab walks you through using PHP to create simple Web applications that use forms to gather information from a user. In addition, the important topic of sessions is demonstrated.

Learning Outcomes

At the completion of the lab you should be able to:

1. Create, and test PHP scripts to process HTML Forms
2. Compare and contrast session creating mechanisms in PHP

Lab Submission Requirements

After completing this lab, you will submit a word (or PDF) document that meets all of the requirements in the description at the end of this document. In addition, your PHP files should be submitted. You should submit multiple files in a zip file.

Virtual Machine Account Information

Your Virtual Machine has been preconfigured with all of the software you will need for this class. You have connected to this machine in the previous labs. Reconnect again using the Remote Desktop connection, your Administrator username and password.

Part 1 - Create and test PHP scripts to process HTML Forms

In this exercise we will create PHP web pages that include simple forms that use GET and POST methods for submission of data. The GET method is used to request data from a specified resource whereas the POST method is used to send data to a server. The first set of code below is the HTML file using three Text input fields and a PHP GET method.

1. Copy and paste the following code into a file named DemoGetForm.html in the SDEV300 folder on your AWS Cloud VM.

```
<html>
<head><title>Simple Form with Get Method </title>
</head>
<body>
   <h1> Please complete the Form </h1>
   <form action="getSubmit.php" method="get">
     First name: <input type="text" name="fname"><br>
     Last name: <input type="text" name="lname"><br>
    Password: <input type="password" name="mypass"><br>
     <input type="submit" value="Submit">
   </form>
</body>
</html>
```

Notice this is an HTML file with form fields similar to those created in a previous lab exercise. However; the action “getSubmit.php” and the method=”get” provide the uniqueness and real functionality of this form. When data is entered into the Text fields and a user clicks on the Submit button, the data is processed by the getSubmit.php using the HTTP GET method process.

Next, we will create the getSubmit.php file used to process the form.

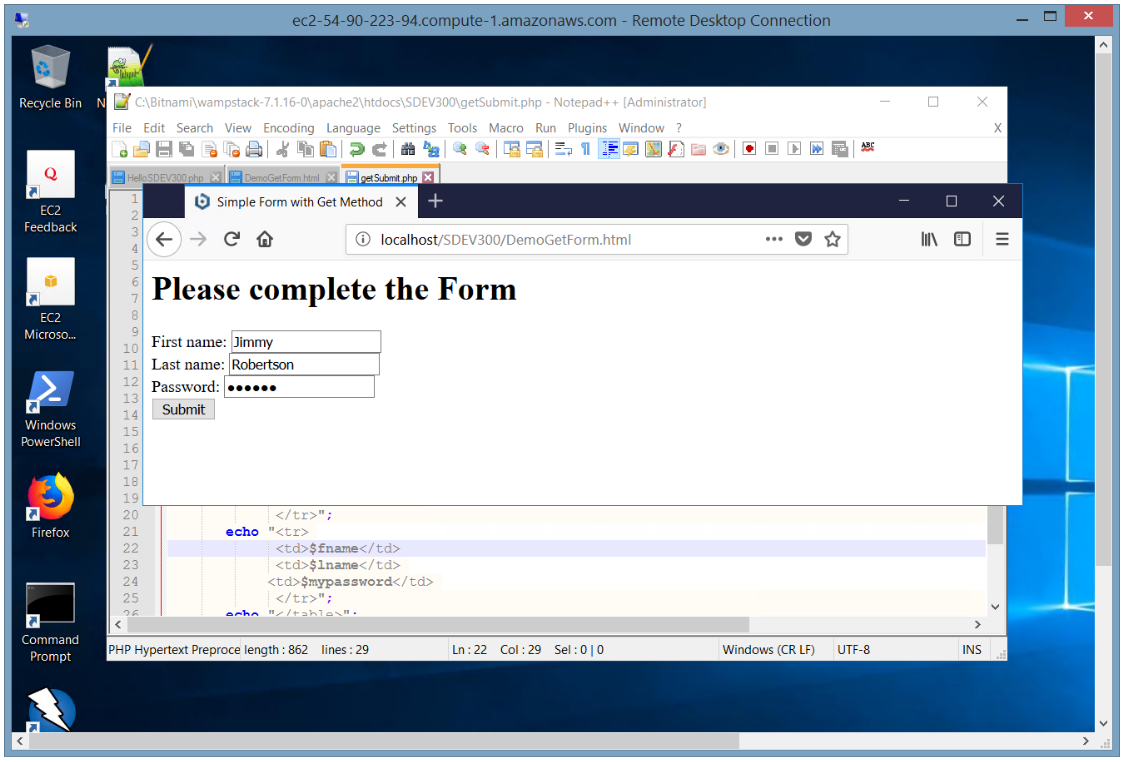
2. Create an additional file that will be used to process the HTML form that is submitted. The file should be named getSubmit.php and be placed in same location as the HTML file. The file should contain these contents:

```
<!-- HTML Forms with Get Submit
 Date: Jan 01, XXXX
 Author: Dr. Robertson
 Title: getSubmit.php
 description: Demo how to retrieve Form data
 -->
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>Get Form Echo</title>
</head>
<body>

<?php
   // Retrieve Data using GET method

   $fname = $_GET["fname"];
   $lname = $_GET["lname"];
        $mypassword = $_GET["mypass"];

        // Display in a table
       echo "<h3> Form Data </h3>";
       echo "<table border='1'>";
       echo "<tr>
             <th>Firstname</th>
             <th>Lastname</th>
             <th>Password</th>
             </tr>";
       echo "<tr>
             <td>$fname</td>
             <td>$lname</td>
            <td>$mypassword</td>
             </tr>";
       echo "</table>";

?>
</body>
</html>
```

Several key components are found in this PHP file that make the Web application dynamic. Note the following common functionality and elements in a PHP form processing file:

a. The names of the Text fields and other HTML form fields found in the HTML file are used to retrieve the data entered in the form by the user. Notice in this case, the text fields named "fname" and "lname" are the names associated the the $_GET method (e.g. `$_GET["fname"]).` Be sure to appropriately name those attributes when creating your PHP file.
b. You can use any acceptable and appropriate PHP variable to assign the values of the GET (or POST) methods. For example, the code assigned $fname to the GET["fname"] value.
c. Once you have the values from the form, you can process as needed. In this example, the results were just displayed in HTML table. However; you could store the data in database, send the data to another location or one of nearly infinite other tasks.
d. Use **echo** to encode HTML code. Notice echo starts the line and then quotes are used to enclose the HTML table code. (e.g. `echo "<table border='1'>"`;)
e. Note that quotes within double quotes should use single quotes for correctly processing the HTML attributes. `"<table border='1'>"`;)
f. Any valid HTML code can be embedded within the PHP code including style sheets and JavaScript code. However; be careful with interlacing too much as the code does become difficult to maintain and debug.

3. Launch the HTML file from your localhost/SDEV300 folder. Enter your firstname, lastname and some text for a password in the form as shown in Figure 1.


ec2-54-90-223-94.compute-1.amazonaws.com - Remote Desktop Connection
C:\Bitnami\wampstack-7.1.16-0\apache2\htdocs\SDEV300\getSubmit.php - Notepad++ [Administrator]
Simple Form with Get Method
localhost/SDEV300/DemoGetForm.html
Submit

Please complete the Form

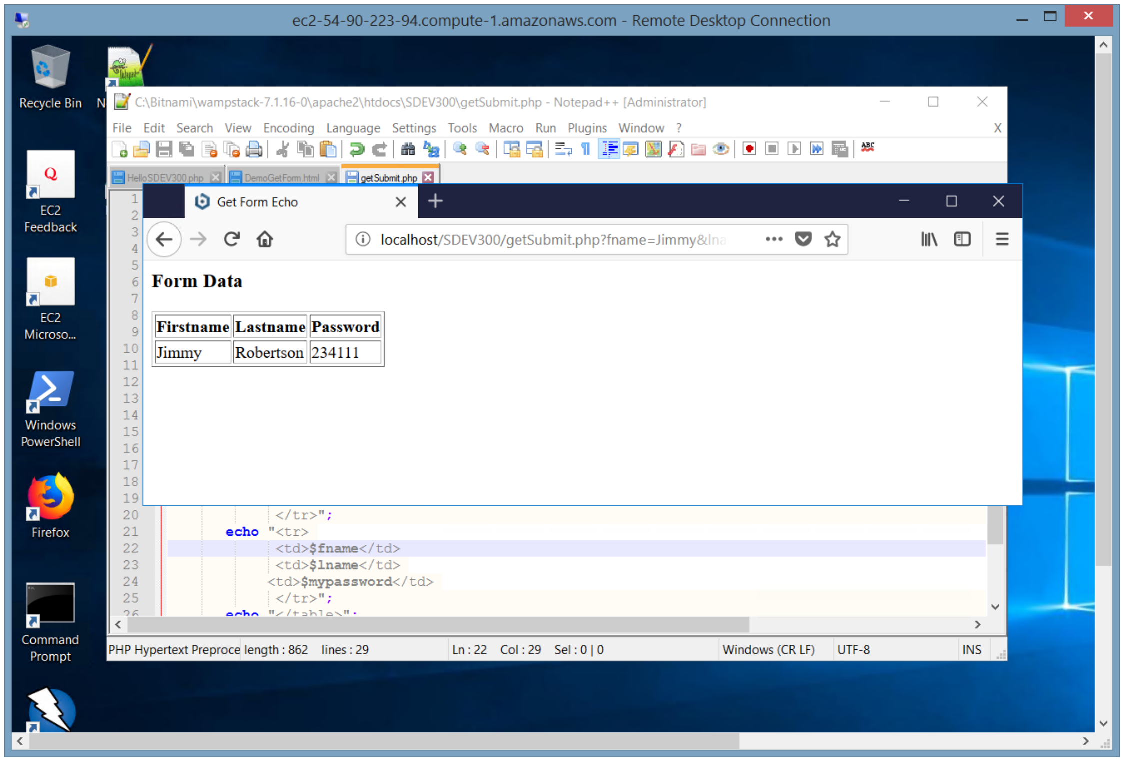
First name: Jimmy
Last name: Robertson
Password: ••••••

```
              </tr>";
        echo "<tr>
              <td>$fname</td>
              <td>$lname</td>
             <td>$mypassword</td>
              </tr>";
```

PHP Hypertext Preproce length : 862 lines : 29 Ln : 22 Col : 29 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

Figure 1 Entering Data in the Form

4. Press the Submit button and the form data you entered will be echoed to the display as shown in figure 2.



ec2-54-90-223-94.compute-1.amazonaws.com - Remote Desktop Connection

C:\Bitnami\wampstack-7.1.16-0\apache2\htdocs\SDEV300\getSubmit.php - Notepad++ [Administrator]

localhost/SDEV300/getSubmit.php?fname=Jimmy&lna

Form Data

Firstname	Lastname	Password
Jimmy	Robertson	234111

```
            </tr>";
    echo "<tr>
            <td>$fname</td>
            <td>$lname</td>
           <td>$mypassword</td>
            </tr>";
```

PHP Hypertext Preproce length : 862 lines : 29 Ln : 22 Col : 29 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

Figure 2 Get Submit Results

As you review the code and results, notice the query string sent at the URL provides all of the field parameters and their values. Also, notice this is in clear text and very insecure.

http://localhost/SDEV300/getSubmit.php?fname=Jimmy&lname=Robertson&mypass=234111

When you see this query string it is a clear sign, the GET method was used. Sending passwords or other sensitive information using the GET method is never recommended and a sure way to have your Web application hacked very quickly.

5. To compare with the Post method create two additional files using the following code:

HTML file: DemoPostForm.html

```
<html>
<head><title>Simple Form with Post Method </title>
</head>
<body>
<h1> Please complete the Form </h1>
<form action="postSubmit.php" method="post">
  First name: <input type="text" name="fname"><br>
  Last name: <input type="text" name="lname"><br>
  Password: <input type="password" name="mypass"><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

PHP file: postSubmit.php

```
<!-- HTML Forms with Post Submit
 Date: Jan 01, XXXX
 Author: Dr. Robertson
 Title: postSubmit.php
 description: Demo how to retrieve Form data
 -->
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <title>Get Form Echo</title>
</head>
<body>
<?php
      // Retrieve Data using Post method
      $fname = $_POST["fname"];
      $lname = $_POST["lname"];
        $mypassword = $_POST["mypass"];
        // Display in a table
       echo "<h3> Form Data </h3>";
       echo "<table border='1'>";
       echo "<tr>
             <th>Firstname</th>
             <th>Lastname</th>
             <th>Password</th>
             </tr>";
       echo "<tr>
             <td>$fname</td>
             <td>$lname</td>
            <td>$mypassword</td>
             </tr>";
       echo "</table>";

?>
</body>
</html>
```

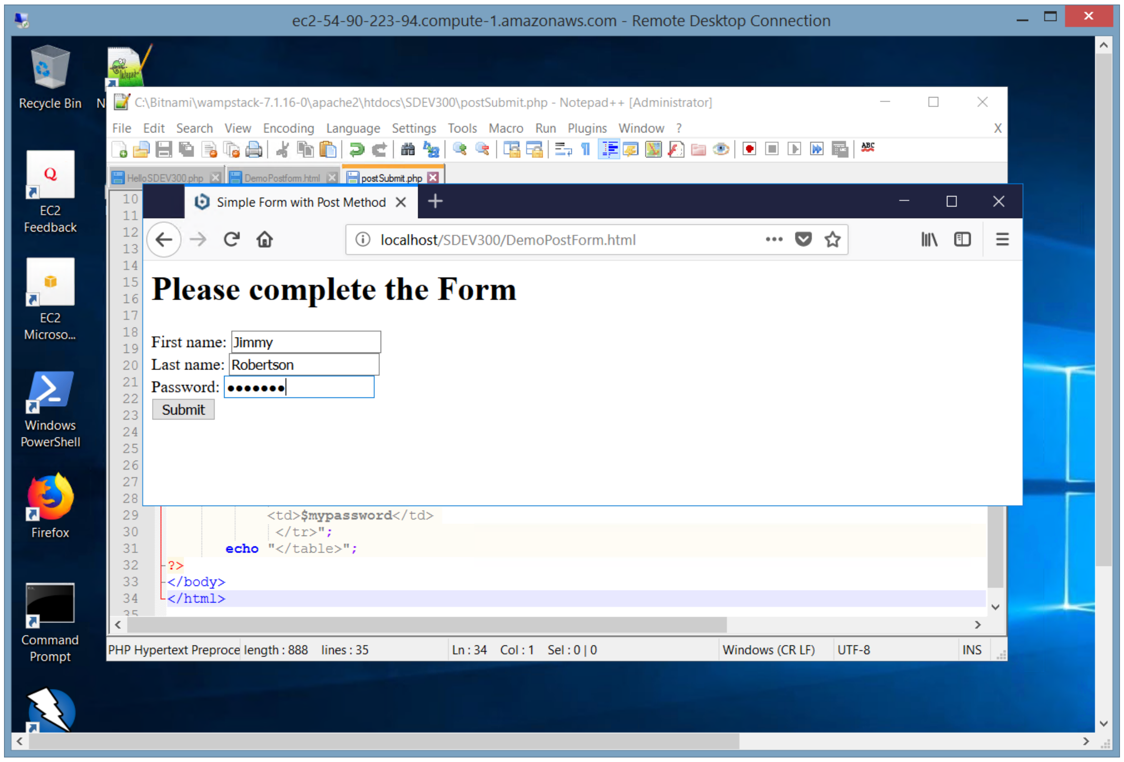
When comparing the POST method to the GET method files, notice the only real differences in code are the use of the "POST" as opposed to "GET". For example, for the postSubmit.php code, the data is retrieved using the following code for the $lname variable:

```
$lname = $_POST["lname"];
```

Also notice the form attributes in the DemoPostForm.html file contains the following line of code with the POST method:

```
<form action="postSubmit.php" method="post">
```

6. Launch and run the application and note the output display as shown in figures 3 and 4.


ec2-54-90-223-94.compute-1.amazonaws.com - Remote Desktop Connection

C:\Bitnami\wampstack-7.1.16-0\apache2\htdocs\SDEV300\postSubmit.php - Notepad++ [Administrator]

Simple Form with Post Method

localhost/SDEV300/DemoPostForm.html

Please complete the Form

First name: Jimmy
Last name: Robertson
Password: •••••••
Submit

```
                <td>$mypassword</td>
              </tr>";
        echo "</table>";
?>
</body>
</html>
```

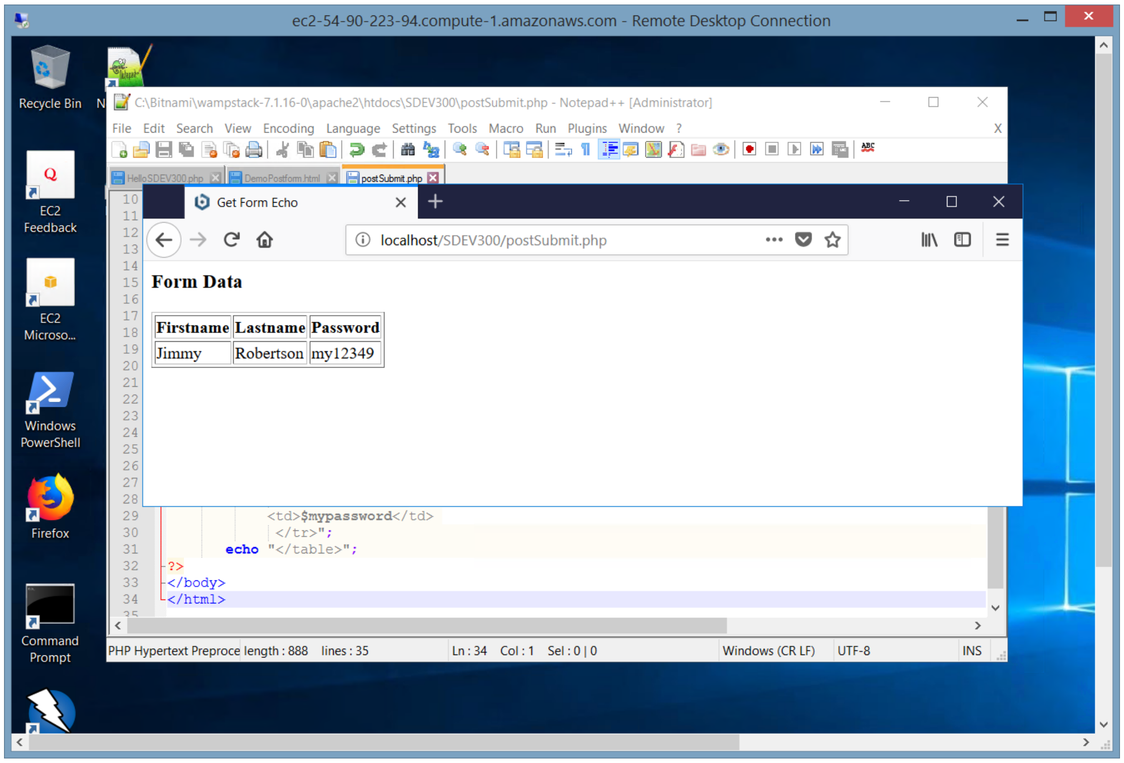
PHP Hypertext Preproce length : 888 lines : 35 Ln : 34 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

Figure 3 Running the HTTP Post

To send the data to for processing click on the “Submit” button.

As shown in figure 4, the end results are the same where the data entered in the form is echoed in the display. However; in this case the URL does not contain the clear text password:

```
http://localhost/SDEV300/postSubmit.php
```


ec2-54-90-223-94.compute-1.amazonaws.com - Remote Desktop Connection
Recycle Bin
EC2 Feedback
EC2 Microso...
Windows PowerShell
Firefox
Command Prompt
C:\Bitnami\wampstack-7.1.16-0\apache2\htdocs\SDEV300\postSubmit.php - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
HelloSDEV300.php
DemoPostform.html
postSubmit.php
Get Form Echo
localhost/SDEV300/postSubmit.php
PHP Hypertext Preproce length : 888 lines : 35
Ln : 34 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS

Form Data

Firstname	Lastname	Password
Jimmy	Robertson	my12349

```
                <td>$mypassword</td>
                 </tr>";
        echo "</table>";
?>
</body>
</html>
```

Figure 4 Post Results

As you will see in some of security analysis we perform with the OWASP ZAP tool later in this course, the post method isn’t necessarily more secure but at least the URL doesn’t display the sensitive data in a query string. This offers some advantages particularly in keeping your access logs more secure with less sensitive information. For example, compare the following two lines found in the access.log file in the logs area for the Apache server:

```
[18/May/2018:14:51:38 +0000] "GET
/SDEV300/getSubmit.php?fname=Jimmy&lname=Robertson&mypass=234111 HTTP/1.1"
200 332

[19/May/2018:13:57:00 +0000] "POST /SDEV300/postSubmit.php HTTP/1.1" 200 338
```

Clearly, the second line is more appropriate and more secure as the password is not available in the log files.

Part 2 - Compare and contrast session creating mechanisms in PHP

In this exercise we will create a simple PHP page that creates and reads available cookies as well as PHP session variables. For the first session example, we will create a cookie in PHP and use a form based submission to expire the cookie.

HTTP, by default is stateless. This means as a person navigates a Web page or even submits data via forms or other mechanism, once the user leaves that page, the state, and other information about what the use has done or requested is no longer available. Cookies and sessions provide a mechanism to add state to the interaction with a user to make the experience more personal.

The Mozilla site defines a cookie as "A small piece of data that a server sends to the user's web browser. The browser may store it and send it back with the next request to the same server". Ultimately, cookies help an application determine if requests come from the same browser.

Sessions are similar to cookies but often provide tokens or some specific identifier to preserve certain data across subsequent accesses. This allows the developer to build more customized applications enhancing the interactivity and personalization of the web site. PHP and most other languages has session methods that support this functionality.

As you can imagine, starting and correctly stopping sessions is critical in web application security. For example, imagine if a user logs into a web application on a public computer and forgets to terminate their session. Then, another use logs in and the application still believes the first user is still logged in and provides private or sensitive information. Clearly, much caution and meticulous testing must take place to ensure sessions are secure and not hijacked by someone else.

1. Copy and paste the following code into a file named DemoCookies.php in your AWS Cloud VM in your SDEV300 folder.

```
<!-- PHP and Cookies
 Date: Jan 01, XXXX
 Author: Dr. Robertson
 Title: DemoCookies.php
 description: Demo how to use Cookies with PHP
 -->
 <!DOCTYPE html>
<html>
<head>
  <title>Cookies Demo </title>
</head>
<body>
<h1>PHP Cookies Demo </h1>
<?php
$cookie_name = "UMUCGamer";
$cookie_value = "CMSC325";
// Set cookie for 7 days
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
// Check for cookie
if(!isset($_COOKIE[$cookie_name])) {
    echo $cookie_name . "," . $cookie_value . "' is not set!";
} else {
    echo "Welcome back" . $cookie_name . "-" . $_COOKIE[$cookie_name];

}
?>
</br>
<!-- Form to expire cookie -->
```

```
<form action="expireCookie.php" method="post">
    <input type="submit" value="Expire Cookie">
</form>

</body>
</html>
```

There are multiple code components related to managing cookies in this example worth mentioning:

a. The setcookie() method is used to set the name, value and expiration time. In this example, a cookie named UMUCGamer was set with a value of CMSC325 and an expiration time of 30 days from the current time. (86400 seconds = 1 day). The "/" denotes the path where the cookie is available. The "/" denotes the entire domain. (In this case, localhost)
b. The isset() method is used to determine if the cookie is set. The if logic is then used to take action. So in this case, messages are displayed based on if the cookie is set.
c. Finally, a form is provided with a single button that will expire the cookie. Keep in mind, in this case, the cookie, once set was enabled for 30 days. The expire cookie will remove it sooner if needed.

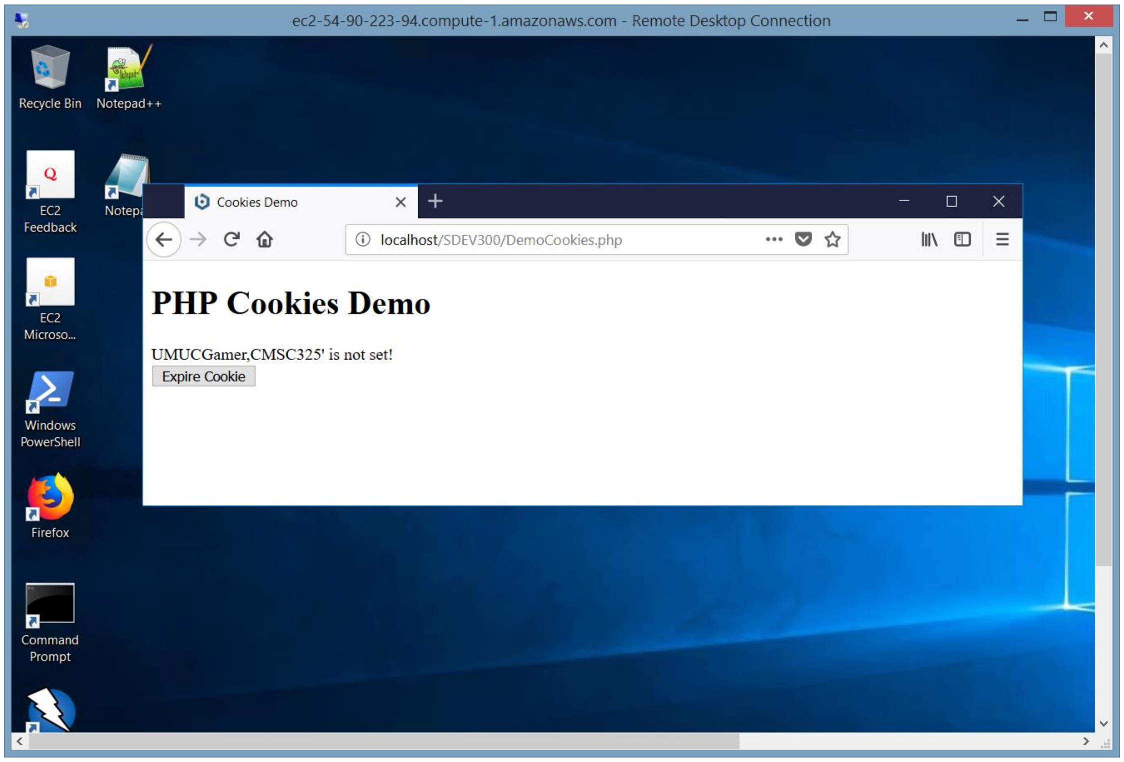
Cookies are quite handy in advertising or determining is a particular visits site quite regularly.

2. Create an additional file that will be respond when the Expire Cookie button is selected. The file should be named expireCookie.php and be placed in same location as the HTML file. The file should contain these contents:

```
<html>
<head><title>Expire the cookies </title>
</head>
<body>
<?php
// Expire the cookie
$cookie_name = "UMUCGamer";
if(isset($_COOKIE[$cookie_name])) {
 setcookie( $cookie_name, "", time() - 3600, "/" );
 echo "Expiring the cookie: " . $cookie_name;
}
else {
 echo "Cookie not found to expire ";
}
?>
<h2> Thanks for playing with PHP cookies </h2>
</body>
</html>
```

In the expire cookie PHP code, the isset() method is used to determine if a particular cookie is set. If it is the expiration time is expired by setting the expiration time to the current time less one hour (3600 seconds). If the cookie was already expired or not found, an appropriate message is displayed.

3. Launch the HTML file. Cookies can be challenging to work with, because they often linger beyond their actual expiration date. This does have security implications which we will touch on through this program. When launching the application for the first time, the cookie has not been set. (See figure 5).

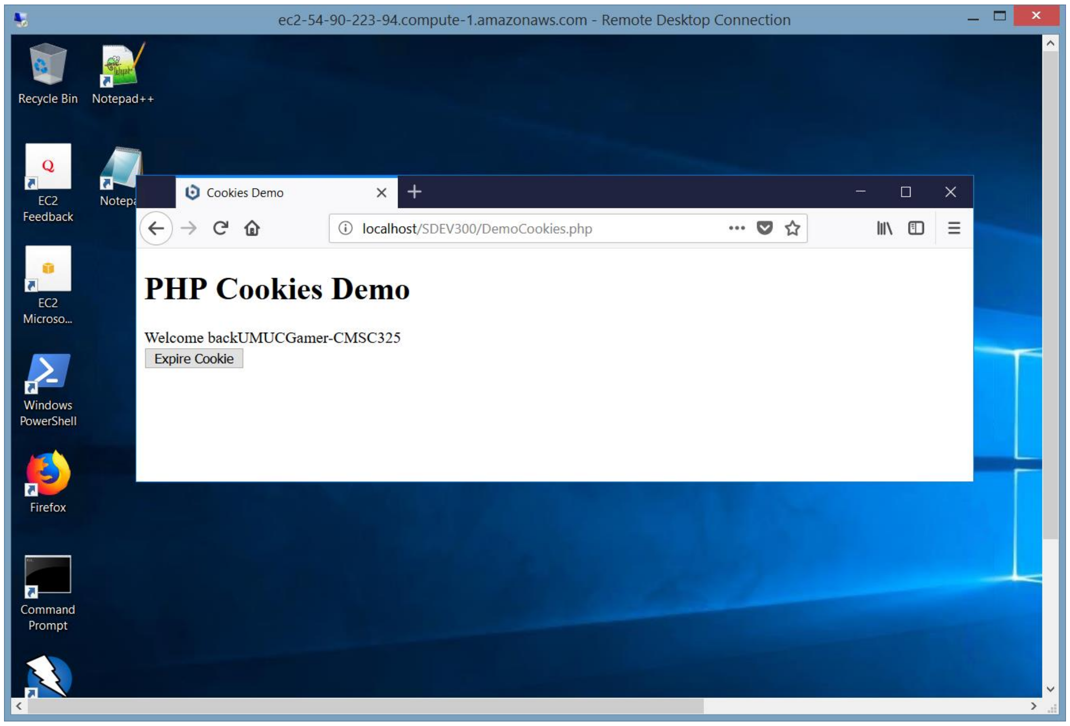


PHP Cookies Demo

UMUCGamer,CMSC325' is not set!

Figure 5 DemoCookies PHP Initial Run

As shown in figure 6, when you refresh the browser and open it again, the cookie will be present as indicated by the welcome back message.

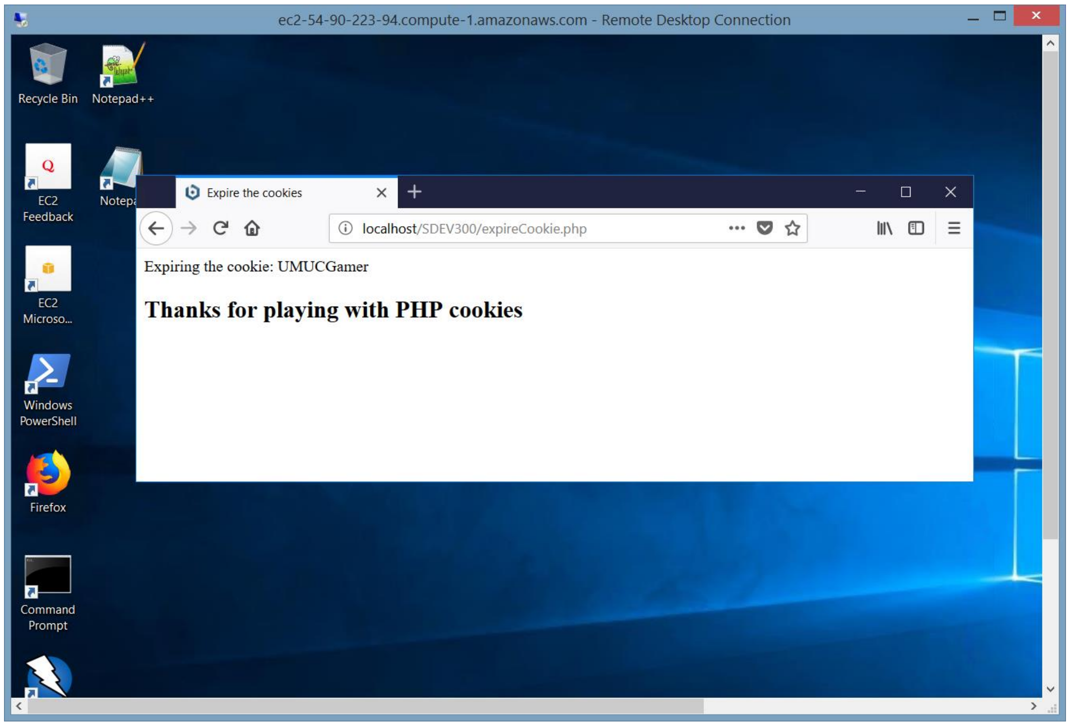

ec2-54-90-223-94.compute-1.amazonaws.com - Remote Desktop Connection
Recycle Bin
Notepad++
EC2 Feedback
EC2 Microso...
Windows PowerShell
Firefox
Command Prompt
Cookies Demo
localhost/SDEV300/DemoCookies.php

PHP Cookies Demo

Welcome backUMUCGamer-CMSC325

Figure 6 Refreshing or reloading the page shows the cookie was set

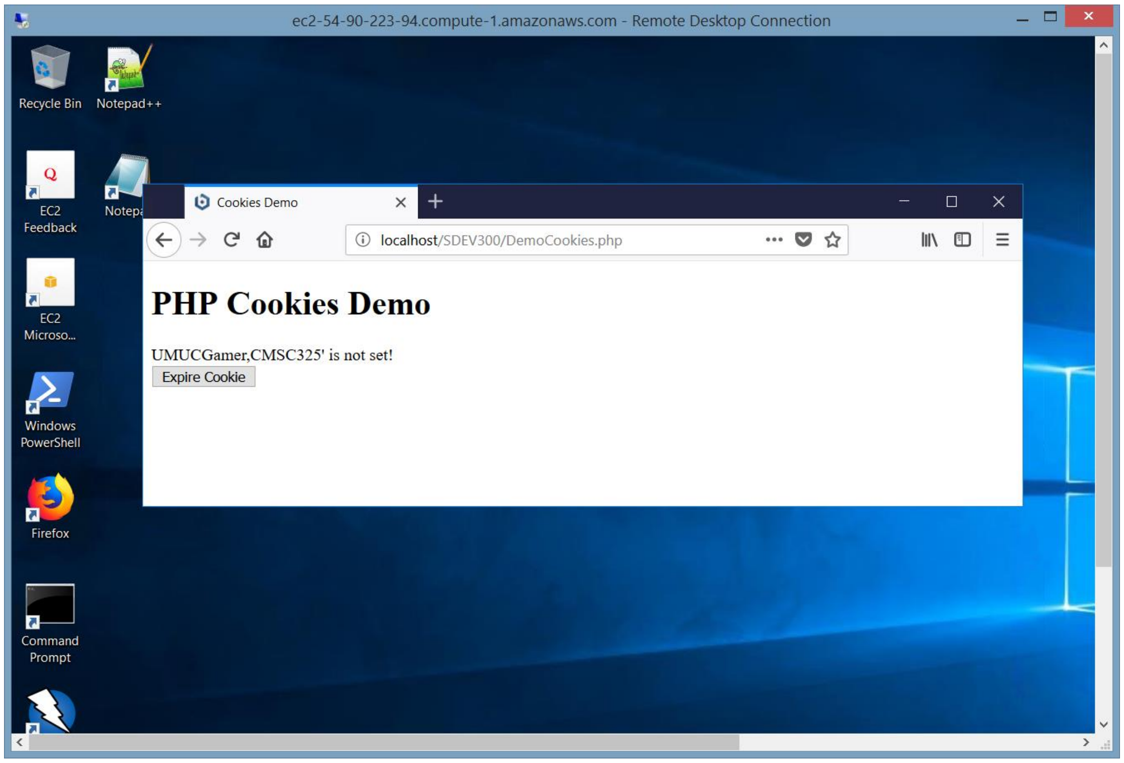
We would need to wait 30 days (with our current settings) to have the cookie expire. However; if you select the “Expire Cookie” button and refresh the browser, the cookie will no longer be present. (See figures 7 and 8)



Expiring the cookie: UMUCGamer

Thanks for playing with PHP cookies

Figure 7 Expiring the cookie


ec2-54-90-223-94.compute-1.amazonaws.com - Remote Desktop Connection
Recycle Bin
Notepad++
EC2 Feedback
EC2 Microso...
Windows PowerShell
Firefox
Command Prompt
Cookies Demo
localhost/SDEV300/DemoCookies.php

PHP Cookies Demo

UMUCGamer,CMSC325' is not set!

Figure 8 The cookie was expired

You should experiment with the cookie code by setting different expiration times to verify the performance and limitations of using cookies. If you visit a site, that uses cookies (which most do) you will often receive an informational note that the site uses cookies. This is required by developers to be in compliant with certain (and recent) privacy laws.

4. Next, we will look at the Sessions options in PHP. In this example, we will create 3 files. An html file will be used to input a username and email address using an html form. Upon submitting the form, a PHP file will start a session and store the username and email address in session variables. Finally, an option to logout and unset the stored session variables will be provided in a logout.php file.

Session variables are more commonly used than cookies for maintaining state in Web applications. However; they can be dangerous from a security perspective since often browsers do not properly remove these variables immediately. This is why you often see a note from the site, to completely close the browser after ending your session.

The files created for this demonstration of session variables will be named loginAuth.html, authcheck.php and logout.php; respectively.

File: loginAuth.html

```
<html>
<head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
   <title>Form Login</title>
</head>

<body OnLoad="document.main.username.focus();">


<table >
      <tr>
            <td colspan="2">
<h4>Enter your Username and Email Address to continue</h4>
</td>
</tr>
<!-- create the main form with an input text boxes -->
<form name="main" method="post" action="authcheck.php">
<tr>
<td>username:</td>
<td><input name="username" type="text" size="50"></td>
</tr>
<tr>
<td>Email Address:</td>
<td><input name="emailadd" type="text" size="50"></td>
</tr>
<tr>
<td colspan="2" align="center"><input name="btnsubmit" type="submit"
value="Submit"></td>
</tr>
</table>
</form>

</body>
</html>
```

In the loginAuth.html file, a simple login form requesting the use to enter the username and email address. Notice the form uses a HTTP POST method with the resulting action of “authcheck.php”.

One additional note is JavaScript was used to focus the user on the username field of the main form. We haven’t covered JavaScript but you will see quite a bit of JavaScript code intermingled in HTML (and PHP) code. It is fairly easy to use as is it is object-based and has a good reference manual at the Mozilla web site. In this case, the object-based notation of onLoad=”document.main.username.focus();” is just instructing that when the page is loaded, focus on the main.username field of the form. The document object always refers to the entire page. Note JavaScript is not Java code, although some syntax is similar.

If you are interested in learning more about Javascript, check out this developer reference:

https://developer.mozilla.org/en-US/docs/Web/JavaScript

File: authcheck.php

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>User Authenticate </title>
```

```
</head>
<body>

<?php
      // Retrieve Post Data
      $username = $_POST["username"];
      $email = $_POST["emailadd"];

             // Set the session information
             session_start();
             $_SESSION['appusername'] = $username;
             $_SESSION['appemail'] = $email;

 // Display the Session information
 echo "<h3> Session Data  </h3>";
echo "<table border='1'>";
echo "<tr>
        <td>Username </td>
        <td> Email </td>
      </tr>";
echo "<tr>
        <td>" . $_SESSION['appusername'] . "</td>";
echo  "<td>" . $_SESSION['appemail']. "</td>";
echo   "</tr>";
 echo "</table>";

// Provide a button to logout

echo "<form name='logout' method='post' action='logout.php'>
<input name='btnsubmit' type='submit' value='Logout'>
</form>";
?>
</body>
</html>
```

The authcheck.php file takes the two POST variables from the HTML submit and sets two session variables in PHP. A couple of comments about this code:

1. The session_start(); call must be used for all PHP files that require session use. Even if a previous file used session_start(); subsequest PHP files that request session values must use that call as well.
2. Use $_SESSION[]to set a session value where the name of the session variable is specifically listed in the []. For example, `$_SESSION['appusername'] = $username;` In this case, the value in the $username is assigned to the Session variable named appusername.
3. At the end of the code, a form (that calls logout.php) is used to unset the session variables. It is critical all session based web applications have a logout option.

By default, the PHP session timeout is 30 minutes.

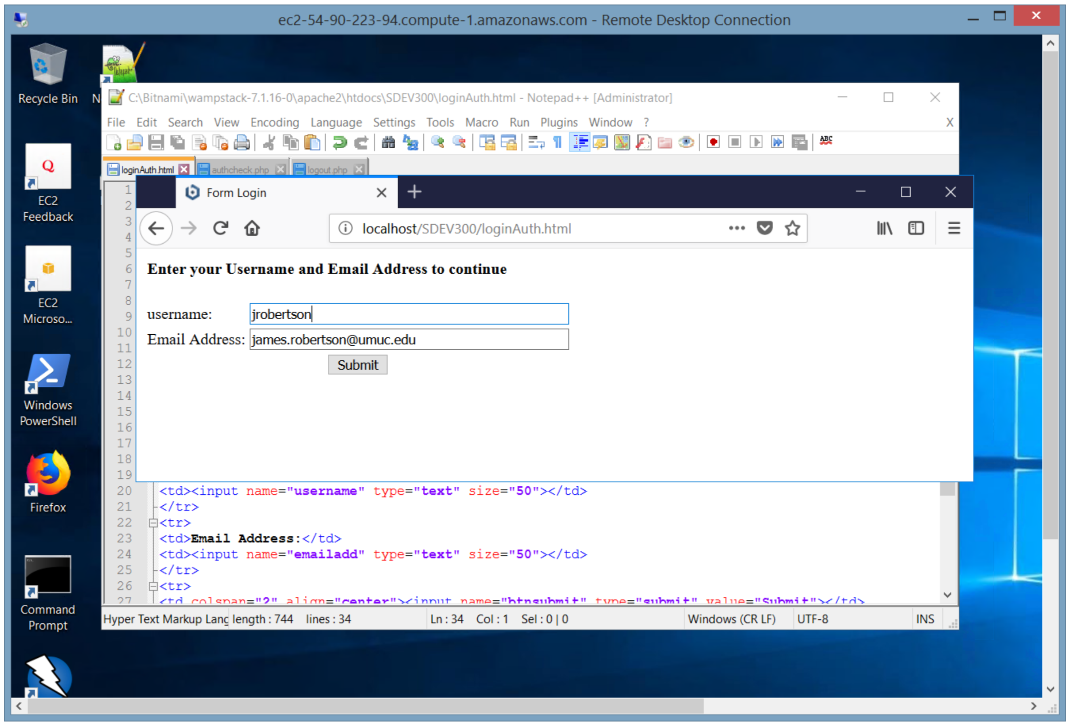
File: logout.php

```
<html>
<head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
   <title>Form Login</title>
</head>

<?php
session_start();
unset($_SESSION['appusername']);
unset($_SESSION['appemail']);


// Display the Session information
echo "<h3> Session Data after Logout  </h3>
<table border='1'>
<tr>
        <td>Username </td>
        <td> Email </td>
      </tr>
<tr>
        <td>" . $_SESSION['appusername'] . "</td>" .
        "<td>" . $_SESSION['appemail'] . "</td>
      </tr>
    </table>";
 ?>

</body>
</html>
```

The logout.php file uses the unset() method to unset the appusername, and appemail session variables. Then the values are displayed (which should now be null) in an HTML table. Notice, the session_start() was called prior to unsetting the session variables.

5. Run loginAuth.html from your localhost/SDEV300 folder in your AWS Cloud VM Browser as shown in figure 9.



ec2-54-90-223-94.compute-1.amazonaws.com - Remote Desktop Connection

C:\Bitnami\wampstack-7.1.16-0\apache2\htdocs\SDEV300\loginAuth.html - Notepad++ [Administrator]

localhost/SDEV300/loginAuth.html

Enter your Username and Email Address to continue

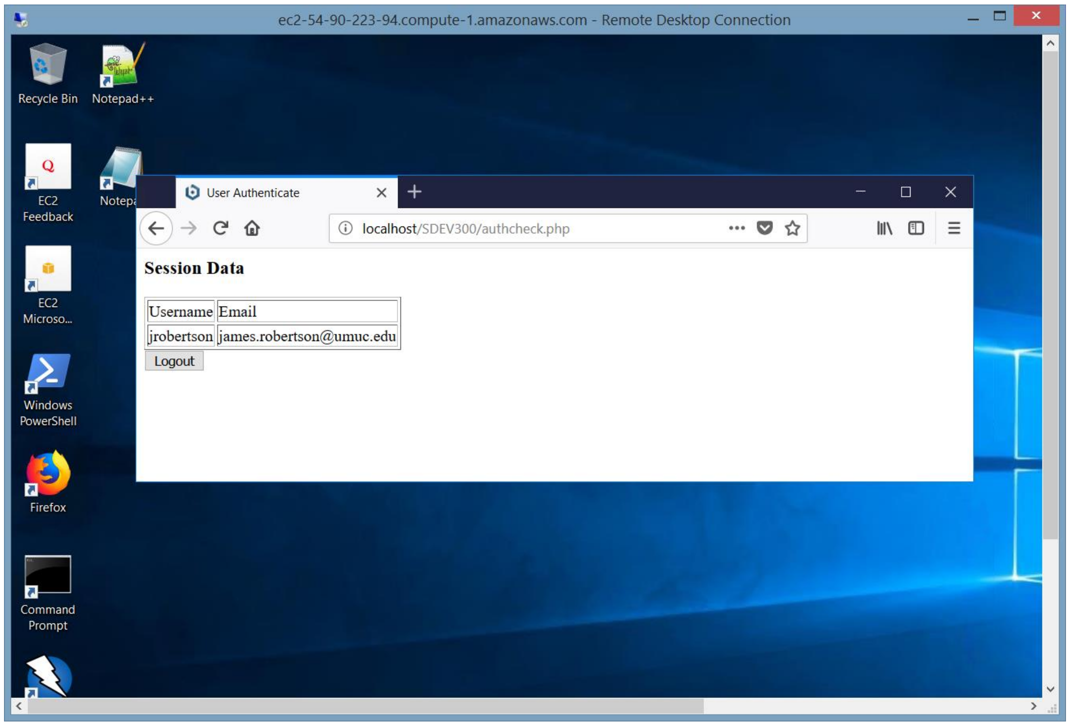
username:	jrobertson
Email Address:	james.robertson@umuc.edu

```
<td><input name="username" type="text" size="50"></td>
</tr>
<tr>
<td>Email Address:</td>
<td><input name="emailadd" type="text" size="50"></td>
</tr>
<tr>
<td colspan="2" align="center"><input name="btnsubmit" type="submit" value="Submit"></td>
```

Form Login
Submit
Hyper Text Markup Lang length : 744 lines : 34 Ln : 34 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

Figure 9 Enter Username and Email

After typing in your username and email address click on “Submit”. As shown in figure 10, the check will set the session variables and show their current values.



ec2-54-90-223-94.compute-1.amazonaws.com - Remote Desktop Connection

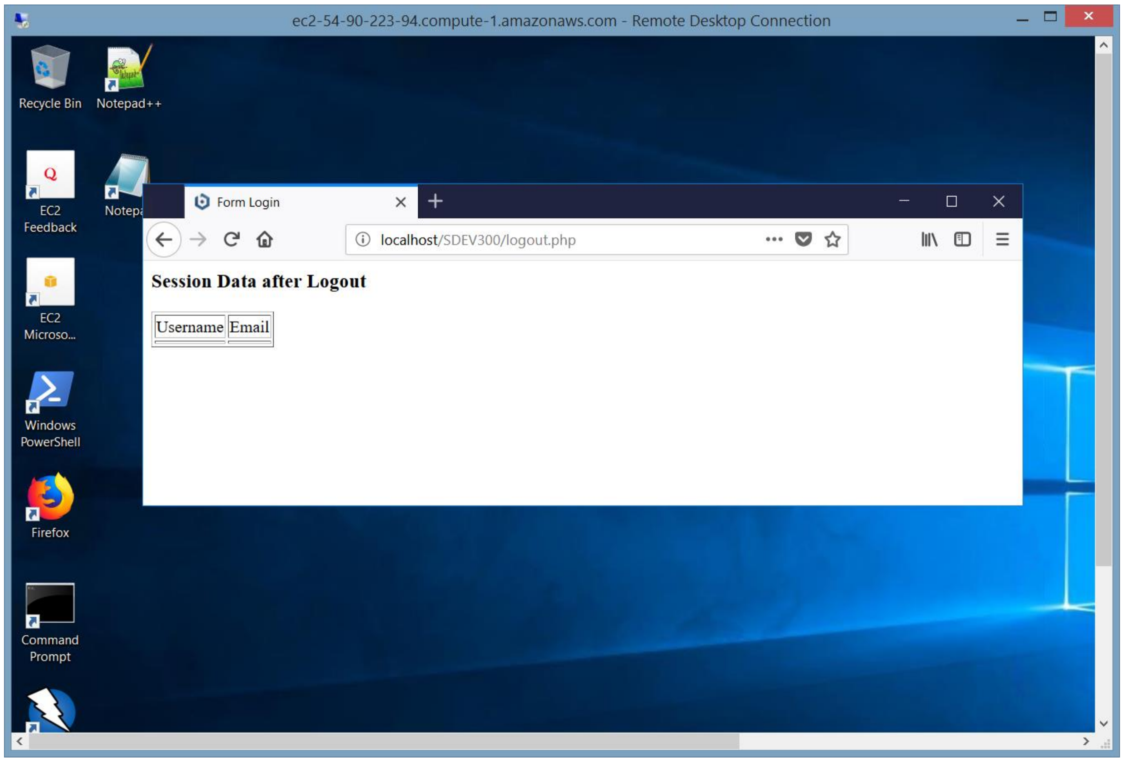
localhost/SDEV300/authcheck.php

Session Data

Username	Email
jrobertson	james.robertson@umuc.edu

Figure 10 Display the Session Variables

6. Click on “logout” to unset the session variables. (See figure 11).


ec2-54-90-223-94.compute-1.amazonaws.com - Remote Desktop Connection
Recycle Bin
Notepad++
EC2 Feedback
EC2 Microso...
Windows PowerShell
Firefox
Command Prompt
Form Login
localhost/SDEV300/logout.php

Session Data after Logout

Username	Email

Figure 11 Unset the Session Variables

You should experiment with the code to learn more about session variables. Be sure to add more session variables and test with additional PHP code that uses the isset() method to check for the presence of session variables.

The isset() check is critical in checking for session variable for displaying a PHP page. For example, you don’t want a customer to directly jump to a Products pages without logging in. So, you must only show the Products page if first check to see if the PHP session variables are set. If they are not set, you must redirect them to the login page.

The following PHP provide similar functionality to what would be needed:

```
session_start();

// check for valid login
if (!isset($_SESSION['username'])
{ // Send to login
    include('login.php');
}
else   // Session is Okay. Let ‘em in
{
    // Code here to display the Products Page
}
```

Lab submission details:

As part of the submission for this Lab, you will create your own unique PHP Web application to store and use session variables in a simple project management application. **You must use the AWS Cloud VM for this exercise.**

Specifically, you will create a **unique** Project Management application using PHP and HTML that allows a user to login to a website, check the progress of a specific project and provide feedback about the progress.

The following guidelines should be used in your design and development:

1. The Login form should consist of fields for username, email address and a password.
2. After login, a personal welcome message should appear along with a list of several projects (at least 10) your company is working on and the current percent completion of each project.
3. The display should include the ability for the customer that logged in to provide comments about one or more of the projects and submit those comments.
4. A customer can remain on the site for up to 30 minutes, without activity, before the session will expire.
5. Once all comments are provided, the customer should be able to submit their comments and a summary page showing all of the project information plus the customer comments will result. The customer should be able to add as many comments as they like for each project. (Hint: use a textarea)
6. The summary page should include every project available regardless of whether comments were provided or not.
7. The summary page should include a "Thank you" message at the top as well along with the project data and customer comments and date and time the comments were submitted.
8. Each PHP page should provide the ability for the customer to logout at any time.

The following is a possible mock-up of what a page might look like for listing projects and providing customer feedback.

Login	Logout

Welcome John. Please add comments in the last column of the table below and press Submit when ready to review your comments.

Project	% complete	Customer Comments:
SpaceX	35%	
Mission to Mars	15%	
PHP Into	65%	
…		

Submit

When the customer selects "Submit" they should see the personal thank you and the project data and any comments they added for that submission. Note, you do not store the comments in session variables. Assume a refresh occurs for each submission. However; you should store appropriate session variable for the login and make sure the customer has appropriately logged in before displaying the projects for comment.

Feel free to add additional HTML and PHP elements to enhance your web application. Create screen captures showing the successful running of your application. Describe each screen capture to explain the results of your application.

For your deliverables, you should submit a zip file containing your word document (or PDF file) with screen shots of the application running successfully along with your PHP web application file and the access.log file.

You should include the Apache log file. **Submissions without access.log files will not be accepted.**

Include your full name, class number, professor name, and section and date in the document.

Grading Rubric:

Attribute	**Meets**
PHP App	**75 points** The Login form includes fields for username, email address and a password. (5 points) After login, a personal welcome message should appear along with a list of several projects (at least 10) your company is working on and the current percent completion of each project. (10 points) The display should include the ability for the customer that logged in to provide comments about one or more of the projects and submit those comments. (10 points) Stores appropriate session variable for the login and makes sure the customer has appropriately logged in before displaying the projects for comment. (10 points) A customer can remain on the site for up to 30 minutes, without activity, before the session will expire. (10 points) Once all comments are provided, the customer should be able to submit their comments and a summary page showing all of the project information plus the customer comments will result. The customer should be able to add as many comments as they like for each project. (Hint: use a textarea) (10 points) The summary page should include every project available regardless of whether comments were provided or not. (5 points)

	The summary page should include a “Thank you” message at the top as well along with the project data and customer comments and date and time the comments were submitted. (5 points) Each PHP page should provide the ability for the customer to logout at any time. (10 points) **Does not include access.log (-100)** **Does not use the SDEV AMI. (-100)**
Documentation and submission	**25 points** Submits a winzip file containing your word document with screen captures and access.log, html files, image files, PHP files and other required Web application files. (10 points) Includes labeled, screen captures of the AWS Cloud VM running the Web/PHP page. Screen captures are described with VM IP address clearly visible. (10 points) Title page includes your full name, class number and section, date and the professor’s name. Document is neat, well-organized and free from spelling and grammar errors. (5 points)