­

Lab IV – Web Application Programming with PHP/MySQL

CPS 499-02/592-02

Software/Language Based Security

Fall 2020

Dr. Phu Phung

Evan Krimpenfort

# Task I: Mock-up login check with session in the index.php file

## Set a “logged” value

A picture containing calendar

Description automatically generated

**Figure 1: Setting the Logged Value**

When the user logs in, we want to be able to set a flag that states that a person with that session has some variable by the name of “logged” with the value of true. We can use this kind of variable to access certain things that a person with a different session of “logged” cannot access.

## Check if the “logged” value isn’t true

A picture containing text

Description automatically generated

**Figure 2: Checking if the Logged Value isn’t true**

Because we set that “logged” variable, we can now use it for page refreshes because when there is a refresh, the data in the text boxes goes away, but that variable doesn’t. It doesn’t go away because we tied that data to the session. If we already logged in and we decided to refresh the page, we check to see if that data is true so we can redirect that page now to the logged in page.

## c. Testing

### i. Get the Alert

Graphical user interface, application

Description automatically generated

**Figure 3: Alert from not logging in first. Logged = False**

Graphical user interface, text, application, email

Description automatically generated

**Figure 4: View-Source of the alert**

### Ii. Invalid Password

Graphical user interface, text, application, email

Description automatically generated

**Figure 5: Invalid Password Alert. “Logged” still false**

Graphical user interface, text, application, email

Description automatically generated

**Figure 6: View-Source of Invalid Password Alert**

### iii. Correct Password

Graphical user interface, text, application, email

Description automatically generated

**Figure 7: Correct Password Index page. “Logged” now true**

Graphical user interface, text, application, email

Description automatically generated

**Figure 8: View-Source of Correct Password index page**

### iv. Go back to the Login page

Graphical user interface, text, application

Description automatically generated

**Figure 9: Back to the Login Page**

This is because “logged” can’t be viewed from the login.php page pin figure 9. The “logged” value is only seen in the index.php page.

### v. Close and try index.php again

Graphical user interface, text, application, email

Description automatically generated

**Figure 10: Back to the index page**

After closing and opening my browser in figure 10, going to the index.php page allows me to log in right away because the session was never destroyed Thus, allowing that “logged” variable not to be turned false.

### vi. Logout and try index.php

Graphical user interface, application

Description automatically generated

**Figure 11: Logout and back to the index page**

Because we logged out and the logout page destroys the session, when we go to the index page, it sees “logged” as false and asks the user to login in again.

# Task II: Check if the session is logged in login.php page

## Write code

A picture containing calendar

Description automatically generated

**Figure 12: Adding “logged” arithmetic to login.php**

The implementation I did in figure 12 was getting the “logged” value and checking to see if the value was true (meaning the user as already logged in and has never logged out). After that the code pastes an alert saying you’ve been logged in and It also refreshes the page so that it’s now index.php.

## Test code

Graphical user interface, application

Description automatically generated

**Figure 13: Login page fixed Alert**

What happens now after you login in figure 13 and go back to the login page is that now login.php uses the “logged” variable to get the user back into the index page if they decide to close the page and get into it later.

# Task III: Database interaction

## Write code

A picture containing text

Description automatically generated

**Figure 14: Added code to checklogin(..)**

So what’s going on here is that $sql is being written as mysql code to retrieve if the data exists. If it does, the code returns that the username exists and the checklogin function returns true. If the username does not exist, the checklogin function returns false.

## Change the function call

Graphical user interface, text, application, email

Description automatically generated

**Figure 15: Test success with always true**

## Login to MySQL

A screenshot of a cell phone

Description automatically generated

**Figure 16: Email in the Database**

The data in the users query has two rows. One row for the admin username and password and one row for my email username and password. This allows the checklogin function to go through the database shown in figure 16 and find the correct username and corresponding password.

## Show new entry

Graphical user interface, text, application, email

Description automatically generated

**Figure 17: Email success in the index page**