DFTI Log Book

Week 1

- Question 2b - The requested image appears.
- Question 3i – Content-type: image/png
- Question 3ii – Image data is sent as symbols "◊ ����\Yv ��s��^f�(Thγ�M���B"
- Question 4iii - By examining the content-type: line of the HTTP header
- Question 5a – It displays the http response content as is in the request. This is because the MIME type tells the browser to not interpret any and just to display the contents as is.
- Question 5bi – The browser does not interpret the code and displays it as plain text.
- Question 5bii – There was an error code saying that the package could not be interpreted and therefor the error was displayed in the browser.
- Question 6 – The HTTP request header is a lot smaller and there is just an error message contained in the HTTP request content.
- Question 7 – All work is now complete.

Week Summery

This week was a recap of HTML & CSS. I completed all tasks this week.

Week 2

- Question 1ii – The file did not work because the artist was sent with the name theArtist and we were looking for name. So, I changed the input name to name. The form also did not call the searchresults.php file as the action so, I updated the code to do this.
- Question 4 - My PHP did give me the expected result.

Week Summery

This week was and introduction to PHP and how it works I managed all tasks but the very last one which was to add a function to change the style sheet depending on the season. I completed all but the advanced tasks this week.

Week 3

There were no questions this week.

Week Summery

We looked at how to connect to a database using the PDO command. We also looked at how to run SELECT, INSERT, and UPDATE commands using PHP. I managed to complete all tasks in the list this week and feel that as I have done databases before I did not learn to much. I completed all tasks this week.

Week 4

There were no questions this week.

Week Summery

This week we looked at query strings and hidden fields. I completed all but the advanced tasks this week.

Week 5

There w ere no questions this w eek.

Week Summery

This w eek w e looked at session variable and how to create a login screen and stop people from accessing pages w ithout login in first. I completed all but the advanced tasks this w eek.

Week 6

There w ere no questions this w eek.

Cross Site Scripting (XSS) is fixed using HTML entities this means if a < or > symbol is encountered w hen taking an input from a query string that they are replaces w ith their HTML equivalent `&gt;` and `&lt;` this means that any tags such as `<script>` w ill not be fun but w ill instead be converted into text as they are read in. You should also validate any user inputs using a simple PHP validator function for example: `ctype_digit()` for numbers, `ctype_alpha()` for letters, and `ctype_alnum()` for alphanumeric inputs. If you are looking to validate a more complex input it is suggested that you use a regular expression function like this one. The below statement could be used to validate a university username for example 0dearm11.

```
preg_match("/^\d[a-z]{5}\d{2}$/")
```

The `/^` indicates the start of the expression. The `\d` represents a number. The `[a-z]{5}` indicates that you should have 5 letters from the alphabet. The `\d{2}` indicates that you are now expecting 2 numbers, and finally the `$/` indicates the end of the expression.

SQL Injection
To combat against SQL injection you should use SQL prepared statements w ith named variables. This should then be follow ed up w ith a check of the number of row s returned in the case of a login script. Also, in the case of the login script you should alw ays make the `$_SESSION["username"]` is set to the username entered by the user not the one from the database. This w ill stop the user from being able to gain information from your database. Below is an example of an SQL prepared statement. Replace all SQL database query's w ith prepared statements in your code.

```
$statement = $conn->prepare("SELECT * FROM students WHERE lastname=:thename AND course=:thecourse");
$statement->execute([":thename"=>$a, ":thecourse"=>$b]);
```

Cross Site Request Forgery
To combat this kind of attack you should use an auto generated key that you attach to any sensitive links on your w ebsite. This w ill allow you to run a test to make sure that the person clicking on the like is the logged in user and not a script running on another machine. Below is the example code for generating a random string and assigning it to a session variable.

```
S_SESSION["token"] = bin2hex(random_bytes(32));
```

Below is an example of a query string containing the token.

```
http://www.google.com/buy.php?id=3&token=R*>oN^zN66XDwTxTQVeb7N2*mF^7@^*
```

Below is an example of a token check if statement.

```
if($_SESSION["token"] == $token){

}
```

All examples show above can also be found on Nicks notes. If you w ould like to see the best login script done so far w ith as much safety as possible please check the one uploaded in w eek6.

Week Summery

This w eek w e looked at how to secure your w ebsite against SQL injection, Cross Site Scripting, and Cross Site Forgery attacks and examples can be seen above. I have completed all tasks including the advanced ones this w eek.

Week 7

Week Summery

Week 8

Week Summery

Week 9

Week Summery

Week 10

Week Summery

Week 11

Week Summery

Week 12

Week Summery