Web Systems Development

Quiz 2

Part 1:

* 1. User authentication is made easier with the use of server-side components like PHP. Authenticating via HTML is time-consuming, too simplistic, and not as secure as using PHP. With PHP, features such as hashes, encryptions, and salts are available to further secure user data and prevent things such as critical information from being exposed.

Data storage, retrieval, and modification is made possible with the use of server-side components like PHP. This allows for things such as front end code to be stored in databases and applied to multiple pages without having to retype the same content over and over again. Data can easily be modified and stored in databases without having to use complex JS.

PHP runs on its own memory making it process faster

* 1. Input validation on the front-end side prevents users from performing injections and XSS. By only allowing users to input certain characters, if used properly, this restricts the use of special characters that could be used for SQL and XSS.

Prepared statements can help prevent injections and XSS on the backend. For special characters that could expose key data to be used, they must be escaped. This makes prepared statements immune to SQL injections.

Part 2:

2.1) This code checks for a name set to ‘lname’ when a form was submitted and searches a database for that ‘lname’. If a user does not input any information, the page will return a message saying “lname not given, outputting entire file” and then output everything in that entire ‘customers’ table.

The first line checks if the ‘lname’ name submitted most likely from a form using the GET method is set as a variable

The second line checks if the set ‘lname’ does not contain an empty string.

For the third line, if the ‘lname’ variable is not empty, the variable $pstmt uses a prepared statement to query a results table from the ‘customers’ table that contains all values for the key lname that are equal to the :ln variable.

For the fourth line the user’s input for the ‘lname’ variable is binded to the :ln variable in the prepared statement using the bindParam function. When the prepared statement is executed, the contents of the ‘lname’ variable will take the place :ln.

For the fifth line, an else statement is started if the ‘if’ statement in the second line is False.

For the sixth line, PHP uses echo to output on the page “lname not given, outputting entire file”

For the seventh line, the variable $pstmt is assigned to a prepared statement that returns a result table for the entire ‘customers’ table in the database.

The eighth line ends the else statement

The ninth line executes the prepared statement in the $pstmt variable.

Lines ten through twelve format the output on the page based on the results table returned from the $pstmt executed prepared statement

2.2) When a specific element on a page is hovered over, data from a file is retrieved and displayed in an alert created by a browser

For line one, an element with the id ‘trigger’ runs a function when hovered over.

For line two, using jQuery, the function uses the getJSON method to retrieve the ‘people.json’ file.

For line three, the function parses through each object assigned to ‘people’ and produces a browser alert output with contents of the val object name, a comma, a space, and the val object profession.