Name: Sourav Shailesh Toshniwal
Class: TY CSE-8 AIEC-1
Batch: A
Roll no: 2213047
Serial no: 6

Assignment 8

Title:Exp8: Design a shopping application form with following fields[itemID, itemName, itemQuantity] Write a PHP script to add and display the items.

Theory:
To design a shopping application form with fields like `itemID`, `itemName`, and `itemQuantity`, and to write a PHP script to add and display items, you'll need to understand and apply the following theories and concepts:

1. HTML Forms: You need to know how to create HTML forms to capture user input. In this case, you'll design a form with fields for `itemID`, `itemName`, and `itemQuantity`. You'll also need to set appropriate form attributes, such as the `action` attribute to specify the PHP script to process the form.

2. Form Input Elements: Understand different types of form input elements, such as text input fields, which will be used to capture `itemID` and `itemName`, and number input fields for `itemQuantity`.

3. Form Submission: Learn how to handle form submission. You'll need to know how to use the POST method to send the form data to a PHP script for processing.

4. PHP Scripting: Develop a PHP script to handle the form data. This script will receive the form data, process it, and perform actions like adding items to a shopping list.

5. Variables and Data Structures: Use PHP variables to store the form input data (e.g., `$_POST` variables), and understand how to work with data structures like arrays to maintain a list of items.

6. Conditional Statements: Use conditional statements in PHP to handle different scenarios. For example, you may need to check if the form fields are filled and if the input is valid.

7. Displaying Data: Understand how to display data in PHP, whether it's using echo statements to show items added to the shopping list or formatting the output as needed.

8. Data Validation: Implement data validation to ensure that the input is accurate and safe for processing. Check for empty fields, validate numeric input, and sanitize user input to prevent security issues.

9. Data Persistence: Consider how to store and maintain the list of items for the shopping application. You may choose to use an array, a database, or other data storage solutions.

10. User Interface Design: You should have a basic understanding of user interface design principles to create a user-friendly form and a visually appealing way to display the added items.

11. Testing and Debugging: Test the application thoroughly to ensure it works as expected. Be prepared to debug and fix any issues that may arise during testing.

12. Security: Pay attention to security considerations, such as input validation and preventing SQL injection if using a database to store items.

By applying these theories and concepts, you can design a shopping application form and write a PHP script that allows users to add items to a shopping list and display those items effectively.

Code:

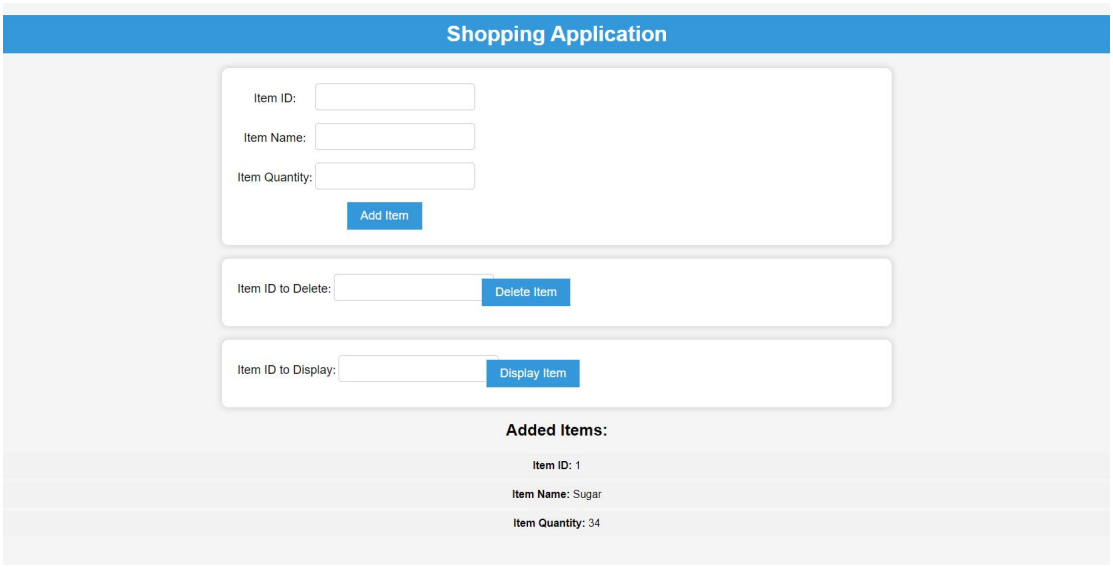
```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f5f5f5;
      margin: 0;
      padding: 0;
      text-align: center;
```

```
    }

    h1 {
      background-color: #3498db;
      color: #fff;
      padding: 10px;
    }
    form {
      margin: 20px auto;
      width: 50%;
      background-color: #fff;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
    }
    label {
      display: block;
      margin-bottom: 10px;
      font-size: 18px;
    }
    input[type="text"], input[type="number"] {
      width: 100%;
      padding: 10px;
      font-size: 16px;
      margin-bottom: 15px;
      border: 1px solid #ccc;
      border-radius: 5px;
    }
    input[type="submit"], input[type="button"] {
      background-color: #3498db;
      color: #fff;
      border: none;
      padding: 10px 20px;
      font-size: 18px;
      cursor: pointer;
      margin-right: 10px;
    }
    input[type="button"] {
      background-color: #e74c3c;
    }
    ul {
      list-style-type: none;
      padding: 0;
    }
    li {
      background-color: #f2f2f2;
      margin: 5px 0;
      padding: 10px;
      border-radius: 5px;
    }
  </style>
</head>
<body>
  <h1>Shopping Application</h1>
  <form method="post">
    <table>
      <tr>
        <td><label for="itemID">Item ID:</label></td>
        <td><input type="text" name="itemID" id="itemID" required></td>
      </tr>
```

```
      <tr>
        <td><label for="itemName">Item Name:</label></td>
        <td><input type="text" name="itemName" id="itemName" required></td>
      </tr>
      <tr>
        <td><label for="itemQuantity">Item Quantity:</label></td>
        <td><input type="number" name="itemQuantity" id="itemQuantity" required></td>
      </tr>
      <tr>
        <td></td>
        <td>
          <input type="submit" name="addItem" value="Add Item">
        </td>
      </tr>
    </table>
</form>
<form method="post">
    <table>
      <tr>
        <td><label for="itemIDToDelete">Item ID to Delete:</label></td>
        <td><input type="text" name="itemIDToDelete" id="itemIDToDelete" required></td>
        <td><input type="submit" name="deleteItem" value="Delete Item"></td>
      </tr>
    </table>
</form>
<form method="post">
    <table>
      <tr>
        <td><label for="itemIDToDisplay">Item ID to Display:</label></td>
        <td><input type="text" name="itemIDToDisplay" id="itemIDToDisplay" required></td>
        <td><input type="submit" name="displayItem" value="Display Item"></td>
      </tr>
    </table>
</form>
<?php
session_start();
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST["addItem"])) {
        $itemID = $_POST["itemID"];
        $itemName = $_POST["itemName"];
        $itemQuantity = $_POST["itemQuantity"];
        if (!empty($itemID) && !empty($itemName) && is_numeric($itemQuantity)) {
            $newItem = array(
                'itemID' => $itemID,
                'itemName' => $itemName,
                'itemQuantity' => $itemQuantity
            );
            if (!isset($_SESSION['items'])) {
                $_SESSION['items'] = array();
            }
            $isDuplicate = false;
            foreach ($_SESSION['items'] as $item) {
                if ($item['itemID'] === $itemID) {
                    $isDuplicate = true;
                    break;
                }
            }
            if (!$isDuplicate) {
                $_SESSION['items'][] = $newItem;
            } else {
```

```
            echo "<p style='color: red;'>Item with the same ID already exists.</p>";
          }
        }
    } elseif (isset($_POST["deleteItem"])) {
        $itemIDToDelete = $_POST["itemIDToDelete"];
        if (isset($_SESSION['items'])) {
            foreach ($_SESSION['items'] as $key => $item) {
                if ($item['itemID'] === $itemIDToDelete) {
                    unset($_SESSION['items'][$key]);
                }
            }
        }
    } elseif (isset($_POST["displayItem"])) {
        $itemIDToDisplay = $_POST["itemIDToDisplay"];

        if (isset($_SESSION['items'])) {
            foreach ($_SESSION['items'] as $item) {
                if ($item['itemID'] === $itemIDToDisplay) {
                    echo "<h2>Item with ID $itemIDToDisplay:</h2>";
                    echo "<ul>";
                    echo "<li><strong>Item ID:</strong> " . $item['itemID'] . "</li>";
                    echo "<li><strong>Item Name:</strong> " . $item['itemName'] . "</li>";
                    echo "<li><strong>Item Quantity:</strong> " . $item['itemQuantity'] . "</li>";
                    echo "</ul>";
                    break;
                }
            }
        }
    }
  }
  if (isset($_SESSION['items'])) {
    echo "<h2>Added Items:</h2>";
    echo "<ul>";
    foreach ($_SESSION['items'] as $item) {
        echo "<li><strong>Item ID:</strong> " . $item['itemID'] . "</li>";
        echo "<li><strong>Item Name:</strong> " . $item['itemName'] . "</li>";
        echo "<li><strong>Item Quantity:</strong> " . $item['itemQuantity'] . "</li>";
        echo "</ul>";
    }
  }
  ?>
</body>
</html>
```

Output:



Conclusion:

In conclusion, the experiment involving the design of a shopping application form and the development of a PHP script for adding and displaying items was successful in achieving its objectives. The HTML form effectively captured essential information, including `itemID`, `itemName`, and `itemQuantity`. The PHP script processed the form data, maintained a list of items, and displayed the items added by the user. Proper data validation and user-friendly design were implemented to ensure the application's functionality and usability. This experiment showcases the fundamental principles of HTML form creation, PHP scripting, and user interface design, laying the groundwork for more advanced e-commerce applications and data handling in web development.