**Video Store Application**

**Objective:**  
In this assignment, you will create a secure video store application that allows users to log in and view a list of video games. The focus will be on proper form validation, using filter\_input to read form data, preventing Cross-Site Scripting (XSS) attacks by escaping user input, and effectively using include statements for partial files where appropriate. You will store data in CSV files and securely handle both login credentials and video game information.

**Part 1: CSV Files**

1. **Video Game CSV File**
   * Create a CSV file named games.csv with the following fields:
     + **Title**: The name of the video game.
     + **Genre**: The category of the game (e.g., Action, Adventure, RPG).
     + **Platform**: The system the game is available on (e.g., PlayStation, Xbox, PC).
     + **Image Path**: The path to the image file for each game (e.g., images/game1.jpg).

Ensure that the images are stored in the correct folder ahead of time. You will not be required to implement image uploads in this assignment.

1. **User CSV File**
   * Create a CSV file named users.csv that contains multiple username and password pairs. Each line should represent a unique combination of username and password.

**Part 2: Login Page with Form Validation**

1. **Login Form:**
   * Create a login form that allows users to input a username and password. The form should submit to the same page where it is displayed.
2. **Form Validation:**
   * Implement form validation that checks the following:
     + Both the username and password fields must not be empty.
     + Use filter\_input to retrieve the values of the username and password fields. Use the appropriate arguments for reading string data.
     + If either field is empty, display an error message (e.g., "Username is required").
     + If the entered username is incorrect, display an error message that includes the incorrect username. Ensure the username is properly escaped using htmlspecialchars() to avoid XSS vulnerabilities.

When redisplaying the form (e.g., after a failed login), ensure all user input securely escaped to prevent potential XSS attacks.

javascript test: <script>alert(“Aloha”)</script>

1. **Login Validation:**
   * After validating the form, compare the entered username and password with the values stored in users.csv. If the credentials are correct, use the header() method to redirect the user to the video game display page. If the credentials are incorrect, display an appropriate error message, ensuring that the error message is securely displayed using htmlspecialchars().
2. **Use of include Statements:**
   * Consider using include statements to separate different parts of the page (e.g., header, footer, or form) into partial files. This will allow for cleaner, more maintainable code.

**Part 3: Video Game Display with XSS Protection**

1. **Video Game Display:**
   * Once the user has successfully logged in, redirect them to a page that displays the video game list stored in games.csv. The page should display each game’s title, genre, platform, and corresponding image.
2. **XSS Protection:**
   * When displaying any user-inputted data, such as video game details, ensure that all output is safely escaped using htmlspecialchars() to prevent potential XSS attacks. This includes properly escaping fields like the title, genre, and platform before displaying them on the page.
3. **Use of include Statements:**
   * When displaying the video games, consider using include statements to structure your page (e.g., for the game list layout or repeated elements like headers). This will improve code reusability and clarity.

**Requirements:**

* Implement proper form validation for the login process, ensuring that empty fields are detected and error messages are displayed.
* Use filter\_input to retrieve the username and password fields from the form.
* Display the incorrect username (properly escaped) as part of the validation when the login fails.
* Use the header() method to redirect to the video game display page upon successful login.
* Ensure all user input is escaped when redisplaying form values and displaying video game details to prevent XSS attacks.
* Think about using include statements to separate parts of your code into partial files for better code organization and reusability.
* Correctly handle login attempts, showing errors for failed attempts and redirecting the user upon success.