**Step 1: Database Connection**

In this step, you establish a connection to your MySQL database using PDO (PHP Data Objects), which is a PHP extension to access databases.

$dbHost, $dbUsername, $dbPassword, and $dbName are variables that store your database connection details.

Inside the try block, you create a new PDO instance by passing the database connection parameters.

The setAttribute() method sets the error mode to ERRMODE\_EXCEPTION, which means PDO will throw exceptions if there are any errors.

**Step 2: Login Page**

This part involves creating a login form where users can enter their credentials (username and password). Upon form submission, the input data is validated, and if the user exists in the database and the provided password is correct, a session is started, and user data is stored in the session variables.

session\_start() initializes a new session or resumes an existing session.

The form data (username and password) is retrieved using $\_POST superglobal.

A prepared SQL statement is used to retrieve user data based on the provided username.

password\_verify() function is used to compare the hashed password stored in the database with the password entered by the user.

If the user authentication is successful, session variables ($\_SESSION['user\_id'] and $\_SESSION['username']) are set, and the user is redirected to the dashboard page.

If authentication fails, an error message is displayed.

**Step 3: Dashboard Page**

After successful login, users are redirected to the dashboard page. Here, we check if the user is logged in by verifying the presence of session data ($\_SESSION['user\_id']). If the user is not logged in, they are redirected back to the login page.

session\_start() is called to start the session.

The presence of $\_SESSION['user\_id'] is checked. If it doesn't exist, the user is redirected to the login page.

If the user is logged in, their user data is fetched from the database based on the user\_id stored in the session.

The dashboard page welcomes the user and provides a logout link.

**Step 4: Logout Page**

The logout script is responsible for destroying the session data when the user chooses to log out. This ensures that the user's session is terminated, and they are logged out of the system.

session\_start() initializes the session.

session\_destroy() function is called to destroy all session data.

The user is then redirected to the login page.

Additional Notes:

Security: It's essential to hash passwords using password\_hash() before storing them in the database to enhance security. Prepared statements are used to prevent SQL injection attacks.

Error Handling: Proper error handling should be implemented throughout the application to provide feedback to users and to handle unexpected situations gracefully.

User Experience: Customize the login and dashboard pages to provide a seamless user experience and match the overall design of your application.

By following these steps, you can create a secure and functional login/logout system with sessions using PDO in PHP.

The <form> element in HTML is used to create an HTML form for user input. The action attribute specifies the URL or file where the form data should be submitted for processing.

In the context of action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>":

$\_SERVER["PHP\_SELF"] is a PHP superglobal variable that contains the filename of the currently executing script. It represents the name of the script that is handling the form submission.

htmlspecialchars() is a PHP function used to convert special characters to HTML entities. This is important to prevent XSS (Cross-Site Scripting) attacks, where an attacker injects malicious HTML or JavaScript code into a web page.

<?php echo ... ?> is a PHP code block that evaluates the expression inside it and outputs the result.

So, action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" means that the form data will be submitted to the same script ($\_SERVER["PHP\_SELF"]) that is currently executing. This is a common practice in PHP to handle form submissions on the same page where the form is located.

Using htmlspecialchars() ensures that any special characters in the script filename are properly escaped, reducing the risk of XSS attacks.