🧾 CIS 334 – Authentication in PHP (PHP 8.3)

🎯 Learning Objectives

By the end of this lesson, you should be able to:

- Explain what authentication is and how it differs from authorization
- Describe how sessions support authenticated state in PHP
- Use PHP's `password_hash()` and `password_verify()` functions securely
- Implement a simple session-based login and logout workflow
- Apply modern PHP 8.3 techniques to improve authentication security

---

🔍 Overview

Authentication verifies **who** a user is. **Authorization** determines **what** that user is allowed to do.

In PHP, authentication typically follows this pattern:

1. User submits a username and password.
2. PHP checks credentials against stored database data.
3. If valid, PHP starts a **session** and saves the user ID in `$_SESSION`.
4. Each protected page checks that session value to confirm authentication.

🔐 Password Hashing in PHP 8.3

Passwords are never stored directly. Instead, use PHP's one-way hashing API:

```php
// Creating a hash when registering a user
$hash = password_hash($password, PASSWORD_DEFAULT);

// Verifying the password at login
if (password_verify($password, $hash)) {
    echo "Valid login!";
}

// Updating to a stronger algorithm automatically
if (password_needs_rehash($hash, PASSWORD_DEFAULT)) {
    $hash = password_hash($password, PASSWORD_DEFAULT);
}
```

- `PASSWORD_DEFAULT` currently uses bcrypt and automatically updates when PHP introduces stronger algorithms.
- Hashes are long, random-looking strings — they cannot be reversed into the original password.

⚙️ Step-by-Step Example

1️⃣ Registration (`register.php`)

```php
<?php
require_once 'database.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $username = trim($_POST['username'] ?? '');
    $password = $_POST['password'] ?? '';

    if ($username && $password) {
        $pdo = Database::getConnection();
        $hash = password_hash($password, PASSWORD_DEFAULT);

        $stmt = $pdo->prepare(
            'INSERT INTO users (username, password_hash) VALUES (:username, :hash)'
        );
        $stmt->execute([':username' => $username, ':hash' => $hash]);

        echo "<p>Registration successful. <a href='login.php'>Log in</a></p>";
    }
}
?>
```

2️⃣ Login (`login.php`)

```php
<?php
session_start();
require_once 'database.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $username = trim($_POST['username'] ?? '');
    $password = $_POST['password'] ?? '';

    $pdo = Database::getConnection();
    $stmt = $pdo->prepare('SELECT id, password_hash FROM users WHERE username = :username');
    $stmt->execute([':username' => $username]);
    $user = $stmt->fetch();

    if ($user && password_verify($password, $user['password_hash'])) {
        $_SESSION['userId'] = (int)$user['id'];
        $_SESSION['username'] = $username;
        header('Location: dashboard.php');
        exit;
    } else {
        echo "<p>Invalid username or password.</p>";
    }
}
?>
```

---

3 Protected Page (`dashboard.php`)

```php
<?php
session_start();
if (!isset($_SESSION['userId'])) {
    header('Location: login.php');
    exit;
}
?>
<h2>Welcome, <?php echo htmlspecialchars($_SESSION['username']); ?>!</h2>
<p><a href="logout.php">Logout</a></p>
```

---

4 Logout (`logout.php`)

```php
<?php
session_start();
session_unset();
session_destroy();
header('Location: login.php');
exit;
```

Modern PHP 8.3 Security Enhancements

- **Strong defaults** – `password_hash()` automatically uses secure algorithms.
- **Session cookie controls** – add protections via `session_set_cookie_params()`:

  ```php
  session_set_cookie_params([
      'secure' => true,       // HTTPS only
      'httponly' => true,     // inaccessible to JavaScript
      'samesite' => 'Lax'
  ]);
  ```

- **Strict typing & exceptions** – reduce errors in authentication logic.

Quick Review

1. What does `password_hash()` do that makes storing passwords secure?

2. Why is `$_SESSION` used instead of cookies for authentication state?
3. What does `password_needs_rehash()` help you accomplish?
4. How can `session_set_cookie_params()` improve session security?
5. Describe what happens, step by step, when a user logs in successfully.

🎯 Practice Exercise

Modify your existing internship or guestbook project:

- Add a registration and login system using `password_hash()` and `password_verify()`.
- Protect at least one page by checking for `$_SESSION['userId']`.
- Add a logout link that destroys the session.
- Test your session cookie using Chrome DevTools → Application → Cookies.

✅ Key Takeaways

- Authentication is about **verifying identity**.
- Always store **hashed** passwords, never plain text.
- Maintain login state with **sessions**, not user-controlled cookies.
- PHP 8.3’s password API and session tools make this both simple and secure.