PHP & Laravel Backend Engineering

Week 3 – Sessions & Authentication

Program: PHP & Laravel Backend Engineering
Week: 3
Level: Beginner
Duration: 3 Days

Learning Objectives

By the end of this week, you will be able to:

- Understand how web sessions work
- Use cookies to remember users
- Create a secure login system
- Protect user passwords with hashing
- Build a safe authentication system

Day 1 – Understanding Sessions and Cookies

What are Sessions?

Think of a session like a shopping cart in a store. When you pick items, the store remembers what you have in your cart until you leave.

How sessions work:

1. User visits the website
2. Server creates a unique session ID
3. Session ID is stored in a cookie
4. Server remembers user's data using this ID

How to Use Sessions in PHP

```
// Start a session
session_start();

// Store data in session
$_SESSION['username'] = 'john_doe';
$_SESSION['last_login'] = time();

// Get data from session
$username = $_SESSION['username'];

// Remove data from session
unset($_SESSION['username']);

// Destroy the session
session_destroy();
```

What are Cookies?

Cookies are small text files stored in the user's browser. They help websites remember information about visitors.

```
// Set a cookie (lasts 30 days)
setcookie('username', 'john_doe', time() + (30 * 24 * 60 * 60), '/');

// Get a cookie
if(isset($_COOKIE['username'])) {
    $username = $_COOKIE['username'];
}

// Delete a cookie
setcookie('username', '', time() - 3600, '/');
```

Security Tips

- Always use `session_start()` at the beginning of your PHP files
- Never store sensitive data in cookies
- Use `session_regenerate_id()` after login to prevent session fixation
- Set `session.cookie_httponly` and `session.cookie_secure` in php.ini

Practice Exercises

1. Create a simple page counter using sessions
2. Make a "Remember Me" checkbox that keeps users logged in
3. Build a shopping cart that remembers items between page visits
4. Create a theme switcher (light/dark mode) using cookies

Day 2 – User Authentication

What is Authentication?

Authentication is like showing your ID card to prove who you are before entering a building.

How to Hash Passwords

Never store passwords in plain text! Always hash them.

```
// Hashing a password (when user registers)
$password = 'my_secure_password';
$hashed_password = password_hash($password, PASSWORD_DEFAULT);
// Store $hashed_password in database

// Verifying a password (when user logs in)
if (password_verify($password, $hashed_password_from_database)) {
    // Password is correct!
} else {
    // Wrong password
}
```

Login System Step by Step

1. **Create a users table in database**

   ```
   CREATE TABLE users (
       id INT AUTO_INCREMENT PRIMARY KEY,
       username VARCHAR(50) NOT NULL UNIQUE,
       email VARCHAR(100) NOT NULL UNIQUE,
       password VARCHAR(255) NOT NULL,
       created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
   );
   ```

2. **Registration Form (register.php)**

```php
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $username = trim($_POST['username']);
    $email = filter_var($_POST['email'], FILTER_SANITIZE_EMAIL);
    $password = $_POST['password'];

    // Validate input (add more validation as needed)
    if (empty($username) || empty($email) || empty($password)) {
        $error = "All fields are required";
    } else {
        // Hash the password
        $hashed_password = password_hash($password, PASSWORD_DEFAULT);

        // TODO: Save to database
        // $db->query("INSERT INTO users (username, email, password) VALUES (?, ?, ?)",
        //    [$username, $email, $hashed_password]);

        // Redirect to login page
        header('Location: login.php');
        exit;
    }
}
?>
<form method="POST">
    <input type="text" name="username" placeholder="Username" required>
    <input type="email" name="email" placeholder="Email" required>
    <input type="password" name="password" placeholder="Password" required>
    <button type="submit">Register</button>
</form>
```

3. **Login Form (login.php)**

```php
<?php
session_start();

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

    // TODO: Get user from database
    // $user = $db->query("SELECT * FROM users WHERE username = ?", [$username])->fetch();

    // For example:
    $user = [
        'id' => 1,
        'username' => 'john',
        'password' => '$2y$10$...' // hashed password
    ];

    if ($user && password_verify($password, $user['password'])) {
        // Password is correct, log the user in
        $_SESSION['user_id'] = $user['id'];
        $_SESSION['username'] = $user['username'];

        // Redirect to dashboard
        header('Location: dashboard.php');
        exit;
    } else {
        $error = "Invalid username or password";
    }
}
?>
<form method="POST">
    <?php if (isset($error)) echo "<p>$error</p>"; ?>
    <input type="text" name="username" placeholder="Username" required>
    <input type="password" name="password" placeholder="Password" required>
    <button type="submit">Login</button>
</form>
```

Practice Exercises

1. Create a complete registration and login system
2. Add a "Remember Me" feature using cookies
3. Build a password reset system
4. Create a user profile page that only logged-in users can see

Day 3 – Building a Secure Login System

Project: User Authentication System

What you'll build: A complete system where users can:

- Register a new account
- Log in with email/username and password
- Reset their password if forgotten
- Log out
- View their profile

Step-by-Step Instructions

1. **Set up the database**
    - Create a users table with fields: id, username, email, password, reset_token, reset_expires
2. **Create these files:**
    - `register.php` - Registration form
    - `login.php` - Login form
    - `logout.php` - Logout script
    - `reset-password.php` - Password reset form
    - `dashboard.php` - Protected page for logged-in users
    - `config.php` - Database connection and functions
3. **Add security features:**
    - Password hashing
    - CSRF protection
    - Session security
    - Input validation
    - Rate limiting for login attempts

Sample Project Structure

```
/auth-system/
├── config.php          # Database connection
├── register.php        # Registration page
├── login.php           # Login page
├── logout.php          # Logout script
├── dashboard.php       # Protected page
├── reset-password.php  # Password reset
└── includes/
    ├── header.php      # Common header
    └── footer.php      # Common footer
```

Practice Exercises

1. Add email verification when users register
2. Implement account lockout after 5 failed login attempts
3. Add a "Remember Me" checkbox that keeps users logged in for 30 days
4. Create an admin panel to manage users (for admin users only)

Resources

- PHP Sessions
- PHP Password Hashing
- OWASP Authentication Cheat Sheet
- PHP Security Guide

Tips for Success

- Test your code after each step
- Look for security vulnerabilities in your code
- Take notes on how sessions and authentication work
- Work with a classmate to test each other's login systems
- If something doesn't work, use `var_dump()` to check your variables)