**Step 14: Implement User Authentication**

In this step we will implement a user auth for both users and admins. Let’s go ahead and do that.

**Update The User.php Model**

***app/models/User.php***

<?php

namespace app\models;

// Include the bootstrap file for global setup

require\_once BASE\_DIR . '/bootstrap.php';

class User {

    private $db;

    // Constructor: Initializes database connection

    public function \_\_construct() {

        $this->db = new \Database; // Establishes database connection

    }

    // Fetches all users from the database

    public function getUsers() {

        $result = $this->db->conn->query('SELECT \* FROM users');

        return $result->fetchAll(\PDO::FETCH\_OBJ); // Returns users as objects

    }

    // Retrieves a single user by username

    public function getUserByUsername($username) {

        $stmt = $this->db->conn->prepare('SELECT \* FROM users WHERE username = ?');

        $stmt->execute([$username]); // Executes with provided username

        return $stmt->fetch(\PDO::FETCH\_OBJ); // Returns the user as an object

    }

    // Creates a new user record

    public function createUser($username, $password, $is\_admin = 0) {

        $stmt = $this->db->conn->prepare('INSERT INTO users (username, password, is\_admin) VALUES (?, ?, ?)');

        $stmt->execute([$username, $password, $is\_admin]); // Inserts new user

    }

    // Updates an existing user's username and password

    public function updateUser($id, $username, $password) {

        $stmt = $this->db->conn->prepare('UPDATE users SET username = ?, password = ? WHERE id = ?');

        $stmt->execute([$username, $password, $id]); // Updates user by ID

    }

    // Deletes a user by ID

    public function deleteUser($id) {

        $stmt = $this->db->conn->prepare('DELETE FROM users WHERE id = ?');

        $stmt->execute([$id]); // Deletes the specified user

    }

 // Check if username exists

  public function doesUsernameExist($username) {

        $stmt = $this->db->conn->prepare("SELECT COUNT(\*) FROM users WHERE username = ?");

        $stmt->execute([$username]);

        $count = $stmt->fetchColumn();

        return $count > 0;

    }

}

***app/controllers/UserController.php***

Let’s go ahead and Update the ***UserController.php*** we created earlier.

<?php

namespace app\controllers;

require\_once BASE\_DIR . '/app/models/User.php';

class UserController {

    private $model;

    // Constructor to initialize the model

    public function \_\_construct() {

        $this->model = new \app\models\User;

    }

    // Display the registration form

    public function register() {

        require \_\_DIR\_\_ . '/app/views/users/register.php';

    }

    // Handle the user registration request

    public function store($data) {

        // Check if username and password are provided

        if (empty($data['username']) || empty($data['password'])) {

            // Set an error message and redirect to the registration page

            $\_SESSION['message'] = 'Username and password are required';

            header('Location: /users/register');

            exit();

        }

    // Check if the username already exists

        if ($this->model->doesUsernameExist($data['username'])) {

            // Handle the error, e.g., set an error message in the session

            $\_SESSION['message'] = "Username already exists. Please choose another username.";

            header('Location: /users/register');

            exit;

        }

        // Hash the password for secure storage

        $hashedPassword = password\_hash($data['password'], PASSWORD\_DEFAULT);

        // Create a new user record in the database

        $this->model->createUser($data['username'], $hashedPassword);

        // Set a success message and redirect to the login page

        $\_SESSION['message'] = 'Registration successful. Please log in.';

        header('Location: /users/login');

        exit();

    }

    // Display the login form

    public function showLoginForm() {

        require BASE\_DIR . '/app/views/users/login.php';

    }

    // Authenticate the user

    public function authenticate($data) {

        // Validate the provided username and password

        if (empty($data['username']) || empty($data['password'])) {

            // Set an error message and redirect to the login page

            $\_SESSION['message'] = 'Username and password are required';

            header('Location: /users/login');

            exit();

        }

        // Retrieve the user from the database

        $user = $this->model->getUserByUsername($data['username']);

        // Check if the user exists and the password is correct

        // Access properties with -> since $user is now an object

        if ($user && password\_verify($data['password'], $user->password)) {

            // Initialize session with user info

            // Make sure to access properties as object properties, not array keys

            $this->initializeUserSession($user);

            // Redirect user to the appropriate page based on their role

            // Again, accessing is\_admin as an object property

            $this->redirectBasedOnRole($user->is\_admin);

        } else {

            // If authentication fails, set an error message and redirect

            $\_SESSION['message'] = 'Invalid username or password';

            header('Location: /users/login');

            exit();

        }

    }

    // Initialize the session with user data

    private function initializeUserSession($user) {

        // Set user ID and admin status in the session

        $\_SESSION['user\_id'] = $user->id;

        $\_SESSION['is\_admin'] = $user->is\_admin;

    }

    // Redirect user based on their role

    private function redirectBasedOnRole($isAdmin) {

        // If user is an admin, redirect to the admin posts page

        if ($isAdmin) {

            header('Location: /admin/posts');

        } else {

            // If user is not an admin, redirect to a general user page

            header('Location: /'); // Replace with the appropriate user page

            }

            exit();

            }

        // Handle the user logout process

        public function logout() {

            // Clear all session variables, and destroy the session

            session\_unset();

            session\_destroy();

            // Redirect to the login page after logout

            header('Location: /users/login');

            exit();

        }

    }

This controller will have different methods for storing user credentials upon registration. Another method for authenticating the user and a logout method.

**Create The View Files login and Register.php**

***views/users/register.php***

<div>

    <!--Display Message-->

    <?php if (isset($\_SESSION['message'])): ?>

        <div id="message" class="alert <?php echo strpos($\_SESSION['message'], 'error') !== false ? 'alert-danger' : 'alert-success' ?> text-center">

            <?php

                echo $\_SESSION['message'];

                unset($\_SESSION['message']);

                ?>

                </div>

    <?php endif; ?>

    <h4>Register</h4>

    <!-- Registration form -->

    <form method="POST" action="/users/store">

        <div>

            <label for="username">Username</label>

            <input type="text" id="username" name="username">

        </div>

        <div>

            <label for="password">Password</label>

            <input type="password" id="password" name="password">

        </div>

        <div>

            <button type="submit">Register</button>

        </div>

    </form>

    <div>

        <a href="/users/login">Already have an account? Login</a>

    </div>

</div>

***views/users/login.php***

<div>

    <!-- Display message -->

    <?php if (isset($\_SESSION['message'])): ?>

        <div id="message" class="alert <?php echo strpos($\_SESSION['message'], 'error') !== false ? 'alert-danger' : 'alert-success' ?> text-center">

            <?php

                echo $\_SESSION['message'];

                unset($\_SESSION['message']);

                ?>

                </div>

    <?php endif; ?>

    <h4>Login</h4>

    <!-- Login form -->

    <form method="POST" action="/users/authenticate">

        <div>

            <label for="username">Username</label>

            <input type="text" id="username" name="username">

        </div>

        <div>

            <label for="password">Password</label>

            <input type="password" id="password" name="password">

        </div>

        <div>

            <button type="submit">Login</button>

        </div>

    </form>

    <div>

        <a href="/users/register">Not a member? Create Account</a>

    </div>

</div>

**public/index.php**

Add the following routes to the ***public/index.php.***

$routes = [

    'GET' => [

// User Auth Routes

        'users/register' => ['controller' => '\app\controllers\UserController', 'method' => 'register'],

        'users/login' => ['controller' => '\app\controllers\UserController', 'method' => 'showLoginForm'],

   ],

 'POST' => [

  // User Auth Routes

         'users/store' => ['controller' => '\app\controllers\UserController', 'method' => 'store'],

         'users/authenticate' => ['controller' => '\app\controllers\UserController', 'method' => 'authenticate'],

    ]

];

**Add A Logout Link**

Here we are going to add a logout link into the ***admin/posts/index.php file.***

<div>

  <!-- Display message -->

    <?php if (isset($\_SESSION['message'])): ?>

        <div id="message" class="<?php echo strpos($\_SESSION['message'], 'error') !== false ? 'message-error' : 'message-success' ?> text-center">

            <?php

                echo $\_SESSION['message'];

                unset($\_SESSION['message']);

            ?>

        </div>

    <?php endif; ?>

    <div style="text-align: right;">

        <!-- Create Post Button -->

        <a href="/admin/posts/create">Create New Post</a>

        | <!-- Add separator for clarity -->

        <a href="/admin/categories/create">Create Categories</a>

        | <!-- Add separator for clarity -->

        <!-- User Login/Logout Links -->

        <?php if (isLoggedIn()): ?>

            <a href="/users/logout">Logout</a>

        <?php else: ?>

            <a href="/users/login">Login</a>

        <?php endif; ?>

    </div>

    <?php foreach ($posts as $post): ?>

        <div style="margin-bottom: 20px; margin-top: 20px;">

            <div>

                <h2><?php echo htmlspecialchars($post->title, ENT\_QUOTES); ?></h2>

                <p><?php echo htmlspecialchars($post->content, ENT\_QUOTES); ?></p>

                <a href="/admin/posts/show/<?php echo $post->id; ?>">View</a>

                <a href="/admin/posts/edit/<?php echo $post->id; ?>">Edit</a>

                <p><strong>Category:</strong> <?php echo $post->category\_name ? htmlspecialchars($post->category\_name, ENT\_QUOTES) : 'No category'; ?></p>

                <!-- Delete Post -->

                <form action="/admin/posts/delete/<?php echo $post->id; ?>" method="POST" style="display: inline;">

                    <input type="hidden" name="\_method" value="DELETE">

                    <button type="submit">Delete</button>

                </form>

            </div>

        </div>

    <?php endforeach; ?>

    <!-- Pagination Controls -->

    <nav aria-label="Page navigation">

        <ul style="list-style: none; padding: 0;">

            <!--we loop through each page number-->

            <?php for ($i = 1; $i <= $totalPages; $i++): ?>

                <!--Check if the loop's current page number ($i, e.g., 3) is the same as the current page ($currentPage, 3)-->

                <!--If yes, then this is the current page, so we could style it differently if needed-->

                <li style="<?php echo $i == $currentPage ? 'font-weight: bold;' : ''; ?>">

                    <!-- returns the page full URL e.g., http://admin/posts?page=2 -->

                    <a href="/admin/posts?page=<?php echo $i; ?>"><?php echo $i; ?></a>

                </li>

            <?php endfor; ?>

        </ul>

    </nav>

</div>

<style>

.message-success {

color: green;

/\* other styling \*/

}

.message-error {

    color: red;

    background-color: #ffd6d6;

    padding: 10px;

    border: 1px solid red;

    margin-bottom: 20px;

}

</style>

**Update the Routes File**

 $routes = [

    'GET' => [

   'users/logout' => ['controller' => '\app\controllers\UserController', 'method' => 'logout'],

]

**Step 15: Add Admin Checks**

Now we are going to add admin checks. These will ensure its only the admin who will access the backend part.

**Create a Base Controller (*app/controllers/AuthController.php)***

<?php

namespace app\controllers;

class AuthController {

    protected function ensureAdmin() {

        if (!isLoggedIn() || !isAdmin()) {

            header('Location: /users/login');

            exit;

        }

    }

}

This method ensure it’s the admin who can access the administrative part. If the person is not an admin he or she will be redirected to the login page.

**Extend the Base Controller**

Now here we will extend the base controller we’ve just created to all our controllers that require admin checks.

These controllers are ***AdminController.php, CategoryController.php*** and ***CommentController.php***

class AdminController extends AuthController {

**Call the Protected Method “ensureAdmin”**

In the controller methods call the ***ensureAdmin*** function on all the methods you want to protect.

 // Read: Display a list of posts

    public function index() {

        $this->ensureAdmin();

**Note:** Do the same for all the other controllers.

**Create a Utility File (utils.php)**

Lastly create file called ***utils.php.***

<?php

function isLoggedIn() {

    return isset($\_SESSION['user\_id']);

}

function isAdmin() {

    return isset($\_SESSION['is\_admin']) && $\_SESSION['is\_admin'] == 1;

}

?>

The **isLoggedIn()** and **isAdmin()** functions defined in **utils.php** are global utility functions. They are designed to be used anywhere in your application, regardless of the context. This means you can call these functions in any script where **utils.php** is included, whether it's within controllers, views, or other PHP files.

**Call The Utils File**

Call the utils file from the ***public/index.php.*** Remember this is the entry file for PHP MVC app.

require BASE\_DIR . '/utils.php';