Documentation for User Management System with MVC Architecture in PHP

System Overview

This documentation describes a user management system built with PHP using the Model-View-Controller (MVC) architecture. The system provides authentication, authorization, and user management functionalities for both administrators and regular users.

Table of Contents

1. [System Architecture](#system-architecture)

2. [Database Setup](#database-setup)

3. [Controllers](#controllers)

4. [Models](#models)

5. [Views](#views)

6. [Features](#features)

7. [Usage Guide](#usage-guide)

8. [Security Considerations](#security-considerations)

9. [Testing Checklist](#testing-checklist)

System Architecture

The application follows the MVC pattern:

```

project/

├── app/

│ ├── controllers/

│ │ ├── AuthController.php

│ │ ├── AdminController.php

│ │ └── UserController.php

│ ├── models/

│ │ └── UserModel.php

│ └── views/

│ ├── auth/

│ │ ├── login.php

│ │ ├── forgot\_password.php

│ │ ├── reset\_password.php

│ │ └── profile.php

│ └── user/

│ ├── index.php

│ ├── create.php

│ └── edit.php

├── public/

│ └── index.php

└── config/

└── database.php

```

Database Setup

Database Schema

```sql

CREATE DATABASE user\_management;

USE user\_management;

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,

role ENUM('admin', 'user') DEFAULT 'user',

status ENUM('active', 'inactive') DEFAULT 'active',

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

CREATE TABLE password\_resets (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

token VARCHAR(255) NOT NULL,

expires\_at DATETIME NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE

);

```

Initial Admin User

```sql

INSERT INTO users (username, email, password, role, status)

VALUES ('admin', 'admin@example.com', '$2y$10$yourhashedpasswordhere', 'admin', 'active');

```

Controllers

AuthController.php

Handles authentication-related operations:

- User login/logout

- Password reset functionality

- Profile management

Key Methods:

- `login()`: Authenticates users

- `logout()`: Ends user session

- `forgotPassword()`: Initiates password reset

- `resetPassword()`: Completes password reset

- `showProfile()`: Displays user profile

- `updateProfile()`: Updates profile information

AdminController.php

Handles admin-specific operations:

- User management

- Dashboard display

Key Methods:

- `dashboard()`: Shows admin dashboard with statistics

- `index()`: Lists all users

- `create()`: Shows user creation form

- `store()`: Stores new user

- `edit()`: Shows user edit form

- `update()`: Updates user information

- `toggleStatus()`: Activates/deactivates users

- `destroy()`: Deletes users

UserController.php

Handles regular user operations:

- User dashboard

- Session management

Key Methods:

- `dashboard()`: Shows user dashboard

- `updatePassword()`: Changes user password

Models

UserModel.php

Handles all database operations related to users:

Key Methods:

- `getAllUsers()`: Retrieves all users

- `getUserById($id)`: Gets single user by ID

- `createUser($data)`: Creates new user

- `updateUser($id, $data)`: Updates user

- `deleteUser($id)`: Deletes user

- `findByUsername($username)`: Finds user by username

- `updatePassword($id, $password)`: Updates password

- `createPasswordResetToken($userId, $token, $expiry)`: Creates reset token

- `findByResetToken($token)`: Finds user by reset token

- `deleteResetToken($token)`: Removes used token

Views

auth/

- `login.php`: Login form with username/email and password fields

- `forgot\_password.php`: Form to request password reset

- `reset\_password.php`: Form to set new password

- `profile.php`: User profile display and edit form

user/

- `index.php`: Lists all users (admin view)

- `create.php`: Form to create new user

- `edit.php`: Form to edit existing user

Features

Admin Functions

- Login to admin dashboard

- Create new users

- View all users

- Edit user details

- Toggle user status (active/inactive)

- Delete users

User Functions

- Login to user dashboard

- Update profile information

- Change password

- Logout

Security Features

- Password hashing (bcrypt)

- Role-based access control

- CSRF protection

- Input validation

- Secure session management

Usage Guide

Installation

1. Clone the repository

2. Set up database using the schema above

3. Configure database connection in `config/database.php`

4. Start from `public/index.php`

Testing Workflow

1. Authentication

- Access `/auth/login`

- Test with invalid credentials

- Login with admin: `admin/password`

- Verify redirection to `/users/index`

2. User Management

- Create new user at `/users/create`

- Edit user at `/users/edit/{id}`

- Toggle user status

- Delete user

3. Profile Management

- Login as regular user

- Update profile at `/auth/profile`

- Change password

4. Password Reset

- Request reset at `/auth/forgot-password`

- Complete reset at `/auth/reset-password/{token}`

5. Access Control

- Verify admin-only pages reject regular users

- Verify authenticated pages reject guests

- Test invalid reset tokens

Security Considerations

1. Password Security

- Always use `password\_hash()` with PASSWORD\_BCRYPT

- Never store plain text passwords

- Enforce strong password policies

2. Session Security

- Regenerate session ID after login

- Set proper session timeouts

- Store sessions securely

3. Input Validation

- Validate all user inputs

- Sanitize output to prevent XSS

- Use prepared statements for SQL queries

4. CSRF Protection

- Implement CSRF tokens for all forms

- Verify tokens on form submission

Testing Checklist

The system includes comprehensive test cases covering:

- User Creation

- Valid/invalid email formats

- Password strength requirements

- Duplicate username/email checks

- Required field validation

- Authentication

- Incorrect credentials

- Inactive user login attempts

- Invalid reset tokens

- Profile Updates

- Email format validation

- Password confirmation matching

- Current password verification for changes

- Access Control

- Admin page access by regular users

- Authenticated page access by guests

- Role-based permission enforcement

This documentation provides a complete reference for developers to implement, maintain, and extend the user management system. The MVC architecture ensures separation of concerns, making the system modular and easier to maintain.