Password Reset Implementation Documentation

Migration Directory Structure

The **database/migrations** folder should be at the root level of your project, parallel to app and public folders:

```
PHP8MVCBlog/
├── app/
├── public/
├── database/
│   ├── migrate.php
│   └── migrations/
│       ├── Migration.php
│       ├── 001_add_reset_password_fields.php
├── bootstrap.php
└── composer.json
```

```
mkdir database
mkdir database/migrations
```

1. Database Changes

Add Reset Password Fields

```php
<?php
// database/migrations/001_add_reset_password_fields.php

require_once __DIR__ . '/Migration.php';

class AddResetPasswordFields extends Migration {
    protected function up() {
        $query = "ALTER TABLE users
                  ADD COLUMN IF NOT EXISTS email VARCHAR(255) UNIQUE AFTER
username,
                  ADD COLUMN IF NOT EXISTS reset_token VARCHAR(64) DEFAULT
NULL,
                  ADD COLUMN IF NOT EXISTS reset_expires DATETIME DEFAULT
NULL";

        $this->db->conn->exec($query);
    }
}

// Execute migration
$migration = new AddResetPasswordFields();
```

```
$migration->execute();
```

Create Base Migration Class

Created Migration.php:

```
<?php
// database/migrations/Migration.php

class Migration {
    protected $db;

    public function __construct() {
        require_once __DIR__ . '/../../bootstrap.php';
        $this->db = new Database();
    }

    public function execute() {
        try {
            $this->up();
            echo "Migration executed successfully!\n";
        } catch (PDOException $e) {
            echo "Migration failed: " . $e->getMessage() . "\n";
        }
    }

    protected function up() {
        // This method should be overridden by child classes
    }
}
```

Create the migration runner script

database/migrate.php

```
<?php
// database/migrate.php

require_once __DIR__ . '/../bootstrap.php';

// Get all migration files
$migrations = glob(__DIR__ . '/migrations/[0-9]*_*.php');
sort($migrations); // Sort by filename to maintain order

foreach ($migrations as $migration) {
```

```
    echo "Running migration: " . basename($migration) . "\n";
    require_once $migration;
}
echo "All migrations completed!\n";
```

Create a Migration Controller

1. **Migration Route Configuration**

Add this route to index.php:

```
<?php
'GET' => [
    // ... other routes ...
    // Migrate Database
    'migrate' => ['controller' => '\app\controllers\MigrationController',
'method' => 'run'],
]
```

2. **Migration Controller**

Create MigrationController.php:

```
<?php
// app/controllers/MigrationController.php

namespace app\controllers;

class MigrationController {
    public function run() {
        // Check if the APP_ENV is not set to production
        if (getenv('APP_ENV') !== 'production') {
            require_once BASE_DIR . '/database/migrate.php';
        } else {
            die('Migrations disabled in production');
        }
    }
}
```

To run migrations, you have two options:

```
<?php
// Option 1: Access via URL (using MigrationController)
http://localhost/php8mvcblog/migrate

// Option 2: Run from command line
```

```
php database/migrate.php
```

2. Model Changes

Update User.php with new methods:

```
<?php
 // Retrieves a single user by email
    public function getUserByEmail($email) {
        $stmt = $this->db->conn->prepare('SELECT * FROM users WHERE email
= ?');
        $stmt->execute([$email]);
        return $stmt->fetch(\PDO::FETCH_OBJ);
    }

    // Checks if an email already exists in the database
    public function doesEmailExist($email) {
        $stmt = $this->db->conn->prepare("SELECT COUNT(*) FROM users WHERE
email = ?");
        $stmt->execute([$email]);
        $count = $stmt->fetchColumn();
        return $count > 0;
    }

// Authenticates a user by username and password
    public function createPasswordResetToken($email) {
        // Generate a secure random token
        $token = bin2hex(random_bytes(32));
        // Set expiration to 24 hours instead of 1 hour for testing
        $expires = date('Y-m-d H:i:s', strtotime('+24 hours'));

        try {
            $stmt = $this->db->conn->prepare('UPDATE users SET reset_token
= ?, reset_expires = ? WHERE email = ?');
            $success = $stmt->execute([$token, $expires, $email]);

            if (!$success) {
                error_log("Failed to update reset token for email:
$email");
                return false;
            }

            return $token;
        } catch (PDOException $e) {
```

```
            error_log("Database error while creating reset token: " . $e-
>getMessage());
            return false;
        }
    }

    // Verifies the password reset token
    public function verifyResetToken($token) {
    // Find a user with the given token and a valid expiration date
        $stmt = $this->db->conn->prepare('
            SELECT * FROM users
            WHERE reset_token = ?
            AND reset_expires > NOW()
            AND reset_token IS NOT NULL
        ');
        $stmt->execute([$token]);
        return $stmt->fetch(\PDO::FETCH_OBJ);
    }

 // Updates the user's password
    public function updatePassword($userId, $newPassword) {
        $hashedPassword = password_hash($newPassword, PASSWORD_DEFAULT);
        // Update the user's password and clear the reset token
        $stmt = $this->db->conn->prepare('UPDATE users SET password = ?,
reset_token = NULL, reset_expires = NULL WHERE id = ?');
        // Execute the query with the hashed password and user ID
        return $stmt->execute([$hashedPassword, $userId]);
    }
```

3. Controller Changes

Add new methods to UserController.php:

Let's start with `**email**` changes first.

```
<?php
public function store($data) {
    // Validate required fields
    if (empty($data['username']) || empty($data['email']) ||
empty($data['password'])) {
        $_SESSION['message'] = 'All fields are required';
        header('Location: /users/register');
        exit;
    }
```

```
    // Validate email format
    if (!filter_var($data['email'], FILTER_VALIDATE_EMAIL)) {
        $_SESSION['message'] = 'Invalid email format';
        header('Location: /users/register');
        exit;
    }

    // Check if email exists
    if ($this->model->doesEmailExist($data['email'])) {
        $_SESSION['message'] = 'Email already exists';
        header('Location: /users/register');
        exit;
    }

    // Create user with email
    $hashedPassword = password_hash($data['password'], PASSWORD_DEFAULT);
    if ($this->model->createUser($data['username'], $data['email'],
$hashedPassword)) {
        $_SESSION['message'] = 'Registration successful. Please log in.';
        header('Location: /users/login');
    } else {
        $_SESSION['message'] = 'Registration failed. Please try again.';
        header('Location: /users/register');
    }
    exit;
}

public function authenticate($data) {
    if (empty($data['email']) || empty($data['password'])) {
        $_SESSION['message'] = 'Email and password are required';
        header('Location: /users/login');
        exit();
    }

    // Changed from username to email authentication
    $user = $this->model->getUserByEmail($data['email']);

    if ($user && password_verify($data['password'], $user->password)) {
        $this->initializeUserSession($user);
        $this->redirectBasedOnRole($user->is_admin);
    } else {
        $_SESSION['message'] = 'Invalid email or password';
        header('Location: /users/login');
        exit();
    }
```

```
}
```

Other methods in `**UserController.php**`

```
// Display the forgot password form
        public function showForgotPasswordForm() {
            require BASE_DIR . '/app/views/users/forgot-password.php';
        }

        public function resetPasswordRequest() {
            $email = $_POST['email'] ?? '';

            if (empty($email)) {
                $_SESSION['message'] = 'Email is required';
                header('Location: /users/forgot-password');
                exit();
            }

            $user = $this->model->getUserByEmail($email);

            if ($user) {
                try {
                    // Generate and save token
                    $token = $this->model-
>createPasswordResetToken($email);

                    // Configure PHPMailer
                    $mail = new \PHPMailer\PHPMailer\PHPMailer(true);
                    $mail->isSMTP();
                    $mail->Host = 'smtp.gmail.com';
                    $mail->SMTPAuth = true;
                    $mail->Username = 'mwengibrian@gmail.com';
                    $mail->Password = 'pmmp stmc nner gsdq';
                    $mail->SMTPSecure =
\PHPMailer\PHPMailer\PHPMailer::ENCRYPTION_STARTTLS;
                    $mail->Port = 587;

                    // Set SSL options
                    $mail->SMTPOptions = [
                        'ssl' => [
                            'verify_peer' => false,
                            'verify_peer_name' => false,
                            'allow_self_signed' => true
                        ]
                    ];
```

```
                    // Build email content
                    $resetLink = "http://" . $_SERVER['HTTP_HOST'] .
"/users/reset-password/" . $token;

                    $mail->setFrom('mwengibrian@gmail.com', 'PHP MVC
Blog');
                    $mail->addAddress($email);
                    $mail->isHTML(true);
                    $mail->Subject = 'Password Reset Request';
                    $mail->Body = "Click the following link to reset your
password: <a href='{$resetLink}'>{$resetLink}</a>";
                    $mail->AltBody = "Click the following link to reset
your password: {$resetLink}";

                    $mail->send();

                    // Log success
                    error_log("Reset email sent to: $email with token:
$token");
                    $_SESSION['message'] = 'Password reset instructions
have been sent to your email';
                    header('Location: /users/login');
                } catch (Exception $e) {
                    error_log("Email sending failed: " . $e-
>getMessage());
                    $_SESSION['message'] = 'Error sending email. Please
try again later.';
                    header('Location: /users/forgot-password');
                }
            } else {
                $_SESSION['message'] = 'Email not found';
                header('Location: /users/forgot-password');
            }
            exit();
        }

        // Display the reset password form
        public function showResetPasswordForm($token) {
            $user = $this->model->verifyResetToken($token);

            if (!$user) {
                // Add debug information
                error_log("Token validation failed. Token: " . $token);
                $_SESSION['message'] = 'Invalid or expired reset token';
```

```
                header('Location: /users/login');
                exit();
            }

            require BASE_DIR . '/app/views/users/reset-password.php';
        }

        // Update the user password
        public function updatePassword() {
            $token = $_POST['token'] ?? '';
            $password = $_POST['password'] ?? '';
            $confirmPassword = $_POST['confirm_password'] ?? '';

            if ($password !== $confirmPassword) {
                $_SESSION['message'] = 'Passwords do not match';
                header("Location: /users/reset-password/{$token}");
                exit();
            }

            $user = $this->model->verifyResetToken($token);

            if ($user && $this->model->updatePassword($user->id,
$password)) {
                $_SESSION['message'] = 'Password has been updated
successfully';
                header('Location: /users/login');
            } else {
                $_SESSION['message'] = 'Error updating password';
                header('Location: /users/forgot-password');
            }
            exit();
        }
    }
```

4. View Changes

Create Forgot Password View

Create forgot-password.php:

```
<!-- app/views/users/forgot-password.php -->
<?php ob_start(); ?>
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <div class="card">
```

```
                <div class="card-body">
                    <h3 class="card-title text-center">Reset Password</h3>
                    <!-- Display any form validation errors here -->
                    <?php if (isset($_SESSION['message'])): ?>
                        <div id="message" class="alert alert-info">
                            <?php echo $_SESSION['message'];
unset($_SESSION['message']); ?>
                        </div>
                    <?php endif; ?>
                    <form action="/users/reset-password-request"
method="POST">
                        <div class="mb-3">
                            <label for="email" class="form-
label">Email</label>
                            <input type="email" id="email" name="email"
class="form-control" required>
                        </div>
                        <div class="d-grid gap-2">
                            <button type="submit" class="btn btn-
primary">Reset Password</button>
                        </div>
                    </form>
                    <div class="text-center mt-3">
                        <a href="/users/login">Back to Login</a>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<?php
$content = ob_get_clean();
include BASE_DIR . '/public/Layouts/layout.php';
?>
```

Create Reset Password View

Create reset-password.php:

```
<!-- app/views/users/reset-password.php -->
<?php ob_start(); ?>
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <div class="card">
```

```
                <div class="card-body">
                    <h3 class="card-title text-center">Set New
Password</h3>
                    <!-- Display any form validation errors here -->
                    <?php if (isset($_SESSION['message'])): ?>
                        <div id="message" class="alert alert-info">
                            <?php echo $_SESSION['message'];
unset($_SESSION['message']); ?>
                        </div>
                    <?php endif; ?>
                    <form action="/users/update-password" method="POST">
                        <input type="hidden" name="token" value="<?php
echo htmlspecialchars($token); ?>">
                        <div class="mb-3">
                            <label for="password" class="form-label">New
Password</label>
                            <input type="password" id="password"
name="password" class="form-control" required>
                        </div>
                        <div class="mb-3">
                            <label for="confirm_password" class="form-
label">Confirm Password</label>
                            <input type="password" id="confirm_password"
name="confirm_password" class="form-control" required>
                        </div>
                        <div class="d-grid gap-2">
                            <button type="submit" class="btn btn-
primary">Update Password</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
<?php
$content = ob_get_clean();
include BASE_DIR . '/public/Layouts/layout.php';
?>
```

5. Route Changes

Added new routes in index.php:

```
<?php
```

```
'GET' => [
    // ... existing routes ...
   // Forgot Password Routes
 'users/forgot-password' => ['controller' =>
'\app\controllers\UserController', 'method' => 'showForgotPasswordForm'],

        'users/reset-password/([A-Za-z0-9]+)' => ['controller' =>
'\app\controllers\UserController', 'method' => 'showResetPasswordForm'],
],
'POST' => [
    // ... existing routes ...
 // Forgot Password Routes
         'users/reset-password-request' => ['controller' =>
'\app\controllers\UserController', 'method' => 'resetPasswordRequest'],

         'users/update-password' => ['controller' =>
'\app\controllers\UserController', 'method' => 'updatePassword']
]
```

6. Dependencies

```
composer require phpmailer/phpmailer
```

or…

Add PHPMailer dependency in composer.json:

```
{
    "require": {
        "phpmailer/phpmailer": "^6.9"
    }
}
```

Run:

```
composer update
```