**Unit Testing the AuthManager Controller**

While full unit testing of controllers can be challenging due to dependencies, here's how we can test specific functionalities of the AuthManager controller:

**1. Testing Login Functionality:**

We can mock the Auth facade and test successful and unsuccessful login attempts.

|  |
| --- |
| public function testSuccessfulLogin() |
| { |
| $auth = Mockery::mock(Illuminate\Support\Facades\Auth::class); |
| $auth->shouldReceive('check')->andReturn(false); |
| $auth->shouldReceive('attempt')->with(['username' => 'test', 'password' => 'secret'])->andReturn(true); |
|  |
| $request = new Request(['username' => 'test', 'password' => 'secret']); |
|  |
| $controller = new AuthManager(); |
| $response = $controller->loginPost($request); |
|  |
| $this->assertEquals(302, $response->getStatusCode()); |
| $this->assertEquals(route('welcome'), $response->headers->get('Location')); |
| $this->assertEquals('success', $response->session()->get('success')); |
| } |
|  |
| public function testFailedLogin() |
| { |
| $auth = Mockery::mock(Illuminate\Support\Facades\Auth::class); |
| $auth->shouldReceive('check')->andReturn(false); |
| $auth->shouldReceive('attempt')->with(['username' => 'test', 'password' => 'wrong'])->andReturn(false); |
|  |
| $request = new Request(['username' => 'test', 'password' => 'wrong']); |
|  |
| $controller = new AuthManager(); |
| $response = $controller->loginPost($request); |
|  |
| $this->assertEquals(302, $response->getStatusCode()); |
| $this->assertEquals(route('login'), $response->headers->get('Location')); |
| $this->assertEquals('error', $response->session()->get('error')); |
| } |

**2. Testing Registration Functionality:**

We can mock the User model and test successful and unsuccessful registration.

|  |
| --- |
| public function testSuccessfulRegistration() |
| { |
| $user = Mockery::mock(App\Models\User::class); |
| User::shouldReceive('create')->andReturn($user); |
|  |
| $request = new Request([ |
| 'username' => 'test', |
| 'email' => 'test@example.com', |
| 'password' => 'secret', |
| 'permission' => 'user' |
| ]); |
|  |
| $controller = new AuthManager(); |
| $response = $controller->registrationPost($request); |
|  |
| $this->assertEquals(302, $response->getStatusCode()); |
| $this->assertEquals(route('login'), $response->headers->get('Location')); |
| $this->assertEquals('success', $response->session()->get('success')); |
| } |
|  |
| public function testFailedRegistration() |
| { |
| $user = Mockery::mock(App\Models\User::class); |
| User::shouldReceive('create')->andReturn(false); |
|  |
| $request = new Request([ |
| 'username' => 'test', |
| 'email' => 'test@example.com', |
| 'password' => 'secret', |
| 'permission' => 'user' |
| ]); |
|  |
| $controller = new AuthManager(); |
| $response = $controller->registrationPost($request); |
|  |
| $this->assertEquals(302, $response->getStatusCode()); |
| $this->assertEquals(route('registration'), $response->headers->get('Location')); |
| $this->assertEquals('error', $response->session()->get('error')); |
| } |

**3. Testing Validation:**

We can directly test the validation rules applied to requests in methods like loginPost and registrationPost.

|  |
| --- |
| public function testLoginValidation() |
| { |
| $request = new Request([]); |
|  |
| $validator = app('validator')->make($request->all(), [ |
| 'username' => 'required', |
| 'password' => 'required' |
| ]); |
|  |
| $this->assertFalse($validator->passes()); |
| } |
|  |
| public function testRegistrationValidation() |
| { |
| $request = new Request([]); |
|  |
| $validator = app('validator')->make($request->all(), [ |
| 'username' => ['required', Rule::unique('users')], |
| 'email' => 'required', |
| 'password' => 'required', |
| 'permission' => 'required' |
| ]); |
|  |
| $this->assertFalse($validator->passes()); |
| } |