* Unit test Đăng nhập

use PHPUnit\Framework\TestCase;

class LoginTest extends TestCase

{

public function testLoginWithValidCredentials()

{

$auth = new Auth();

$result = $auth->login('user@example.com', 'password');

$this->assertEquals('Login successful!', $result);

}

public function testLoginWithInvalidCredentials()

{

$auth = new Auth();

$result = $auth->login('user@example.com', 'wrongpassword');

$this->assertEquals('Incorrect email or password.', $result);

}

public function testLoginWithoutEmail()

{

$auth = new Auth();

$result = $auth->login('', 'password');

$this->assertEquals('Email is required.', $result);

}

public function testLoginWithoutPassword()

{

$auth = new Auth();

$result = $auth->login('user@example.com', '');

$this->assertEquals('Password is required.', $result);

}

}

* Unit test Thêm sp

public function testCreateProduct()

{

// Tạo mock data cho sản phẩm mới

$productData = [

'name' => 'Test Product',

'price' => 100,

'description' => 'This is a test product.',

];

// Gọi hàm thêm sản phẩm mới

$response = $this->post('/products', $productData);

// Kiểm tra xem response status code có phải là 302 không

$response->assertStatus(302);

// Kiểm tra xem sản phẩm đã được thêm vào database chưa

$this->assertDatabaseHas('products', $productData);

}

* Xoá sp

public function testDeleteProduct()

{

// Tạo mock data cho sản phẩm cần xóa

$product = new Product([

'name' => 'Test Product',

'price' => 100,

'description' => 'This is a test product.',

]);

$product->save();

// Gọi hàm xóa sản phẩm

$response = $this->delete('/products/'.$product->id);

// Kiểm tra xem response status code có phải là 302 không

$response->assertStatus(302);

// Kiểm tra xem sản phẩm đã được xóa khỏi database chưa

$this->assertDatabaseMissing('products', ['id' => $product->id]);

}

* Sửa sp

public function testUpdateProduct()

{

// Tạo mock data cho sản phẩm cần sửa

$product = new Product([

'name' => 'Test Product',

'price' => 100,

'description' => 'This is a test product.',

]);

$product->save();

// Tạo mock data mới cho sản phẩm

$newProductData = [

'name' => 'New Test Product',

'price' => 200,

'description' => 'This is a new test product.',

];

// Gọi hàm sửa sản phẩm

$response = $this->put('/products/'.$product->id, $newProductData);

// Kiểm tra xem response status code có phải là 302 không

$response->assertStatus(302);

// Kiểm tra xem sản phẩm đã được sửa trong database chưa

$this->assertDatabaseHas('products', $newProductData);

}

* ***Test case***

1. Test case cho chức năng đăng nhập:

* Kiểm tra đăng nhập thành công với tài khoản đúng (username và password đúng)
* Kiểm tra đăng nhập thất bại với tài khoản sai (username hoặc password sai)
* Kiểm tra xem thông tin người dùng đã được lưu trong session hay không sau khi đăng nhập thành công

1. Test case cho chức năng đăng ký:

* Kiểm tra đăng ký thành công với thông tin hợp lệ (username và email chưa được sử dụng, password đủ mạnh)
* Kiểm tra đăng ký thất bại với thông tin không hợp lệ (username hoặc email đã được sử dụng, password yếu)
* Kiểm tra xem thông tin người dùng đã được lưu trong cơ sở dữ liệu hay không sau khi đăng ký thành công

1. Test case cho chức năng thêm sản phẩm:

* Kiểm tra thêm sản phẩm thành công với thông tin hợp lệ (tên sản phẩm, giá sản phẩm, mô tả sản phẩm)
* Kiểm tra thêm sản phẩm thất bại với thông tin không hợp lệ (thiếu thông tin bắt buộc)
* Kiểm tra xem sản phẩm đã được lưu trong cơ sở dữ liệu hay không sau khi thêm sản phẩm thành công

1. Test case cho chức năng sửa sản phẩm:

* Kiểm tra sửa sản phẩm thành công với thông tin hợp lệ (tên sản phẩm, giá sản phẩm, mô tả sản phẩm)
* Kiểm tra sửa sản phẩm thất bại với thông tin không hợp lệ (thiếu thông tin bắt buộc)
* Kiểm tra xem sản phẩm đã được lưu trong cơ sở dữ liệu hay không sau khi sửa sản phẩm thành công

1. Test case cho chức năng xóa sản phẩm:

* Kiểm tra xóa sản phẩm thành công khi sản phẩm tồn tại trong cơ sở dữ liệu
* Kiểm tra xóa sản phẩm thất bại khi sản phẩm không tồn tại trong cơ sở dữ liệu