

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

KIỂM THỬ PHẦN MỀM

Báo cáo Bài tập lớn

Ứng dụng FloginFE_BE (Đăng nhập & Quản lý Sản phẩm)

GVHD: Từ Lăng Phiêu
SV thực hiện: Nhóm ...
Nguyễn Văn A - MSSV...
Trần Văn B - MSSV...

TP. HỒ CHÍ MINH, THÁNG 11/2025

Mục lục

1 Giới thiệu về Dự án

1.1 Tổng quan

Dự án **FloginFE_BE** là một ứng dụng web hoàn chỉnh bao gồm:

- **Chức năng Login:** Hệ thống đăng nhập với validation đầy đủ.
- **Chức năng Product:** Quản lý sản phẩm (CRUD operations).
- **Frontend:** React 18+.
- **Backend:** Spring Boot 3.2+ (Java 17+).
- **Testing:** Phát triển theo phương pháp TDD, sử dụng JUnit 5, Mockito và Cypress.

1.2 Cấu trúc dự án

Dự án được tổ chức thành 2 module chính:

- **frontend/:** Ứng dụng React (Components, Services, Tests).
- **backend/:** Ứng dụng Spring Boot (Controller, Service, Repository, Entity).

2 Câu 1: Phân tích và Thiết kế Test Cases

2.1 Câu 1.1: Login - Phân tích và Test Scenarios (10 điểm)

2.1.1 Yêu cầu (5 điểm)

Dựa trên chức năng đăng nhập (Login), hãy thực hiện:

- a) Phân tích đầy đủ các yêu cầu chức năng của tính năng Login (2 điểm)

Test Case ID	Mô tả chi tiết
Validation Rules cho Username	Độ dài: Tối thiểu 3 ký tự, tối đa 50 ký tự. - Ký tự cho phép: Chỉ chứa chữ cái thường (a-z), chữ cái hoa (A-Z), chữ số (0-9), dấu gạch ngang (-), dấu chấm (.), và dấu gạch dưới .
Validation Rules cho Password	- Độ dài: Tối thiểu 6 ký tự, tối đa 100 ký tự. - Yêu cầu phức tạp: Phải chứa cả chữ cái và chữ số.
Authentication Flow (Luồng Xác thực)	1. Hệ thống nhận đầu vào: Username và Password. 2. Kiểm tra Validation: Kiểm tra Username và Password có tuân thủ các quy tắc Validation (độ dài, ký tự, phức tạp) không. Nếu không, hiển thị lỗi Validation. 3. Kiểm tra Tài khoản: Nếu hợp lệ, hệ thống gửi thông tin đến máy chủ để kiểm tra sự tồn tại và tính đúng đắn của cặp Username/Password trong cơ sở dữ liệu. 4. Xác thực thành công: Nếu cặp thông tin khớp, hệ thống tạo session/token và chuyển hướng người dùng đến trang chính/dashboard. 5. Xác thực thất bại: Nếu không khớp, hệ thống hiển thị thông báo lỗi chung về Username hoặc Password không hợp lệ.
Error Handling (Xử lý Lỗi)	- Lỗi Validation: Thông báo rõ ràng về lỗi format/độ dài. - Lỗi Xác thực: Thông báo lỗi chung, bảo mật. - Lỗi Hệ thống/Kết nối: Thông báo lỗi kỹ thuật. - Tài khoản bị khóa/vô hiệu hóa: Thông báo lỗi cụ thể.

Bảng 1: Bảng phân tích yêu cầu login

- b) Liệt kê và mô tả ít nhất 10 test scenarios cho Login bao gồm (2 điểm):

Loại Test	Tên Test Scenario	Mô tả chi tiết
-----------	-------------------	----------------

Happy Path	Đăng nhập Thành công (Positive)	Nhập Username và Password hợp lệ (đã đăng ký), nhấn nút Login. Mong đợi: Đăng nhập thành công, chuyển hướng đến trang Dashboard/Home.
Negative	Username Rỗng	Để trống trường Username, nhập Password hợp lệ, nhấn Login. Mong đợi: Hiển thị lỗi Validation cho trường Username (ví dụ: "Username không được để trống").
Negative	Password Rỗng	Nhập Username hợp lệ, để trống trường Password, nhấn Login. Mong đợi: Hiển thị lỗi Validation cho trường Password (ví dụ: "Password không được để trống").
Negative	Sai Mật khẩu	Nhập Username hợp lệ, nhập Password sai (không khớp với Username), nhấn Login. Mong đợi: Hiển thị thông báo lỗi xác thực chung ("Tên đăng nhập hoặc mật khẩu không đúng").
Negative	Username chưa đăng ký	Nhập Username chưa tồn tại trong hệ thống, nhập Password hợp lệ. Mong đợi: Hiển thị thông báo lỗi xác thực chung ("Tên đăng nhập hoặc mật khẩu không đúng").
Boundary (Min)	Username Dưới Min	Nhập Username 2 ký tự (dưới min 3), nhập Password hợp lệ. Mong đợi: Hiển thị lỗi Validation về độ dài Username.
Boundary (Max)	Username Trên Max	Nhập Username 51 ký tự (trên max 50), nhập Password hợp lệ. Mong đợi: Hiển thị lỗi Validation về độ dài Username.
Edge Case	Username Ký tự cấm	Nhập Username chứa ký tự đặc biệt không được phép (ví dụ: @, !,), nhập Password hợp lệ. Mong đợi: Hiển thị lỗi Validation về ký tự cho phép trong Username.
Edge Case	Password thiếu chữ cái	Nhập Username hợp lệ, nhập Password chỉ chứa số (hợp lệ về độ dài). Mong đợi: Hiển thị lỗi Validation yêu cầu Password phải có cả chữ và số.
Edge Case	Khoảng trắng ở đầu/cuối	Nhập Username hợp lệ nhưng có khoảng trắng thừa ở đầu hoặc cuối, nhập Password hợp lệ. Mong đợi: Kiểm tra xem hệ thống có tự động cắt khoảng trắng (trim) hay coi là không hợp lệ.

Bảng 2: Bảng test scenarios cho Login

b) Phân loại test scenarios theo mức độ ưu tiên (Critical, High, Medium, Low) và giải

thích (1 điểm):

2.1.2 Thiết kế Test Cases chi tiết (5 điểm)

Test Case ID	TC_LOGIN_001
Test Name	Đăng nhập thành công với credentials hợp lệ
Priority	Critical
Preconditions	- User account exists - Application is running
Test Steps	1. Navigate to login page 2. Enter valid username 3. Enter valid password 4. Click Login button
Test Data	Username: testuser Password: Test123
Expected Result	- Success message displayed - Token stored - Redirect to dashboard
Actual Result	(Để trống)
Status	Not Run

2.2 Câu 1.2: Product - Phân tích và Test Scenarios (10 điểm)

2.2.1 Yêu cầu (5 điểm)

a) Phân tích đầy đủ các yêu cầu chức năng của Product CRUD (2 điểm)

- **Create: Thêm sản phẩm mới**

Mô tả:

Chức năng cho phép người dùng có quyền (Admin, Manager) tạo một sản phẩm mới bằng cách nhập thông tin vào một biểu mẫu và lưu lại.

Yêu cầu chức năng:

- Hệ thống phải cung cấp một giao diện để nhập các thông tin của sản phẩm mới, bao gồm: Tên sản phẩm (Product Name), Giá (Price), Số lượng (Quantity), Mô tả (Description), và Danh mục (Category).
- Hệ thống phải thực hiện kiểm tra tính hợp lệ của dữ liệu đầu vào theo các quy tắc đã cho.
- Khi người dùng nhấn "Lưu", nếu dữ liệu hợp lệ, sản phẩm mới sẽ được tạo và lưu vào cơ sở dữ liệu.
- Hệ thống phải hiển thị một thông báo thành công sau khi tạo sản phẩm.
- Sản phẩm mới phải xuất hiện trong danh sách sản phẩm.
- Nếu dữ liệu không hợp lệ, hệ thống phải hiển thị thông báo lỗi tương ứng cho từng trường bị sai và không tạo sản phẩm mới.

- **Xem danh sách/chi tiết sản phẩm**

Mô tả:

Chức năng cho phép người dùng xem danh sách tất cả sản phẩm hoặc thông tin chi tiết của một sản phẩm cụ thể..

Yêu cầu chức năng:

- **Xem danh sách**

- Hệ thống phải hiển thị danh sách các sản phẩm dưới dạng bảng hoặc lưới.
- Các thông tin cơ bản như Tên, Giá, Số lượng cần được hiển thị.
- Hệ thống nên hỗ trợ phân trang (pagination) nếu danh sách sản phẩm quá dài.
- Chức năng tìm kiếm và lọc sản phẩm (theo tên, danh mục) nên được cung cấp.
- Nếu dữ liệu không hợp lệ, hệ thống phải hiển thị thông báo lỗi tương ứng cho từng trường bị sai và không tạo sản phẩm mới.

- **Xem chi tiết**

- Khi người dùng nhấp vào một sản phẩm trong danh sách, hệ thống phải hiển thị trang chi tiết chứa đầy đủ thông tin của sản phẩm đó.

b) Liệt kê và mô tả ít nhất 10 test scenarios cho Product bao gồm (2 điểm):

c) Phân loại test scenarios theo mức độ ưu tiên và giải thích (1 điểm):

2.2.2 Thiết kế Test Cases chi tiết (5 điểm)

3 Câu 2: Unit Testing và TDD

3.1 Câu 2.1: Login - Backend Unit Tests

Kiểm thử AuthService sử dụng JUnit 5 và Mockito.

3.1.1 Frontend Unit Tests - Validation Login (5 điểm)

a) Viết unit tests cho validateUsername() (2 điểm)

```
1 describe("validateUsername()", () => {
2     describe("Test 1: Username rỗng", () => {
3         test("nên trả về không hợp lệ khi username là chuỗi rỗng", ()
4             => {
5             const result = validateUsername("");
6             expect(result.isValid).toBe(false);
7             expect(result.error).toBe("Username không được để trống");
8         });
9         .....
10
11        test("nên trả về không hợp lệ khi username chỉ có khoảng trắng",
12            () => {
13            const result = validateUsername(" ");
14            expect(result.isValid).toBe(false);
15            expect(result.error).toBe("Username không được để trống");
16        });
17    });
18
19    describe("Test 2: Username quá ngắn/quá dài", () => {
20        test("nên trả về không hợp lệ khi username quá ngắn (ít hơn 3 k
21            ý tự)", () => {
22            const result = validateUsername("ab");
23            expect(result.isValid).toBe(false);
24            expect(result.error).toBe("Username phải ít nhất 3 ký tự");
25        });
26        .....
27
28        test("nên hợp lệ với đúng 50 ký tự", () => {
29            const result = validateUsername(
30                "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklj"
31            );
32            expect(result.isValid).toBe(true);
33            expect(result.error).toBeNull();
34        });
35    });
36
37
38    describe("Test 3: Ký tự đặc biệt không hợp lệ", () => {
39        test("nên trả về không hợp lệ khi username chứa khoảng trắng",
40            () => {
41            const result = validateUsername("user name");
42            expect(result.isValid).toBe(false);
43        });
44    });
45 }
```



```
42     expect(result.error).toBe(  
43         "Username chỉ chứa chữ, số, dấu chấm (.), gạch d ới (_), và  
         gạch ngang (-)"  
44     );  
45 });  
46  
47     .....  
48  
49     test("nên hợp lệ với gạch ngang (-)", () => {  
50         const result = validateUsername("user-name");  
51         expect(result.isValid).toBe(true);  
52         expect(result.error).toBeNull();  
53     });  
54 });  
55  
56     describe("Test 4: Username hợp lệ", () => {  
57         test("nên hợp lệ khi có chữ cái, gạch d ới, chấm và gạch ngang  
         ", () => {  
58             const result = validateUsername("user_n.ame-");  
59             expect(result.isValid).toBe(true);  
60             expect(result.error).toBeNull();  
61         });  
62     });  
63 });  
64
```

Listing 1: validateUsername

b) Viết unit tests cho validatePassword() (2 điểm)

```
1 describe("validatePassword()", () => {  
2     describe("Test 1: Password r  ̣ng", () => {  
3         test("nên trá v ề không hợp lệ khi password là chuỗi r  ̣ng", ()  
4             => {  
5                 const result = validatePassword("");  
6                 expect(result.isValid).toBe(false);  
7                 expect(result.error).toBe("Mật khẩu không đ  ợc để trống");  
8             });  
9  
10            .....  
11  
12            test("nên trá v ề không hợp lệ khi password là undefined", () =>  
13                {  
14                    const result = validatePassword(undefined);  
15                    expect(result.isValid).toBe(false);  
16                    expect(result.error).toBe("Mật khẩu không đ  ợc để trống");  
17                });  
18            });  
19  
20            describe("Test 2: Password quá ngắn/quá dài", () => {  
21                test("nên trá v ề không hợp lệ khi password quá ngắn (ít h  n 6 k  
22                    ý tự)", () => {  
23                        const result = validatePassword("pass1");  
24                        expect(result.isValid).toBe(false);  
25                        expect(result.error).toBe("Mật khẩu phải ít nhất 6 ký tự");  
26                    });  
27            });  
28        });  
29    });  
30
```

```
24     });
25
26     ....
27
28     test("nên hợp lệ với đúng 50 ký tự", () => {
29         const password = "a".repeat(44) + "123456";
30         const result = validatePassword(password);
31         expect(result.isValid).toBe(true);
32         expect(result.error).toBeNull();
33     });
34 });
35
36
37 describe("Test 3: Password không có chữ hoặc số", () => {
38     test("nên trả về không hợp lệ khi password không có chữ cái",
39     () => {
40         const result = validatePassword("123456");
41         expect(result.isValid).toBe(false);
42         expect(result.error).toBe("Mật khẩu phải chứa ít nhất một chữ
43         cái");
44     });
45
46     ....
47
48     test("nên trả về không hợp lệ khi password có chữ cái và ký tự
49     đặc biệt nh ng không có số", () => {
50         const result = validatePassword("pass!@#$");
51         expect(result.isValid).toBe(false);
52         expect(result.error).toBe("Mật khẩu phải chứa ít nhất một số
53         ");
54     });
55 });
56
57 describe("Test 4: Password hợp lệ", () => {
58     test("nên hợp lệ khi có chữ cái và số", () => {
59         const result = validatePassword("password123");
60         expect(result.isValid).toBe(true);
61         expect(result.error).toBeNull();
62     });
63
64     ....
65
66     test("nên hợp lệ khi chỉ có một số và các chữ cái khác", () =>
67     {
68         const result = validatePassword("password1");
69         expect(result.isValid).toBe(true);
70         expect(result.error).toBeNull();
71     });
72 });
73 });
74 });
```

Listing 2: validatePassword

c) Coverage $\geq 90\%$ cho validation module (1 điểm)

Assignment_template_SGU/ValidationLogin.jpeg

3.1.2 Backend Unit Tests - Login Service (5 điểm)

a) Test method authenticate() với các scenarios (3 điểm):

```
1 @ExtendWith(MockitoExtension.class)
2 @DisplayName("AuthService Unit Test - Bean Validation")
3 public class AuthServiceTest {
4
5     @Mock
6     private UserRepository mockUserRepository;
7
8     @Mock
9     private PasswordEncoder mockPasswordEncoder;
10
11     @Mock
12     private JwtService mockJwtService;
13
14     @Mock
15     private Validator mockValidator;
16
17
18     private Validator realValidator;
```

```
19
20     private AuthService authService;
21     private User mockUser;
22
23     @BeforeEach
24     void setUp() {
25         mockUser = new User(1L, "testuser", "hashedPassword123", "
testuser@example.com");
26         authService = new AuthService(mockJwtService,
mockUserRepository, mockPasswordEncoder, mockValidator);
27
28         ValidatorFactory factory = Validation.
buildDefaultValidatorFactory();
29         realValidator = factory.getValidator();
30     }
31
32     // A) TEST METHOD authenticate()
33
34     @Test
35     @DisplayName("TC1: Login thành công với credential hợp lệ")
36     void testLoginSuccess() {
37         // Arrange
38         LoginRequest loginRequest = new LoginRequest("testUser", "
Test123");
39
40         // Mock validator trả về không có lỗi
41         when(mockValidator.validate(loginRequest))
42             .thenReturn(Set.of());
43
44         // Mock các dependencies
45         when(mockUserRepository.findByUserName(loginRequest.
getUserName()))
46             .thenReturn(Optional.of(mockUser));
47         when(mockPasswordEncoder.matches("Test123", "
hashedPassword123"))
48             .thenReturn(true);
49         when(mockJwtService.generateToken(mockUser))
50             .thenReturn("fake-jwt-token");
51
52         // Act
53         LoginResponse loginResponse = authService.authenticate(
loginRequest);
54
55         // Assert
56         assertTrue(loginResponse.isSuccess(), "Login phải thành công");
57         assertEquals("Login thành công", loginResponse.getMessage());
58         assertNotNull(loginResponse.getToken(), "Token không được null");
59         assertEquals("fake-jwt-token", loginResponse.getToken());
60
61         // Verify UserDto
62         assertNotNull(loginResponse.getUser(), "UserDto không được null");
63         assertEquals("testuser", loginResponse.getUser().
```

```
64     getUsername();
65     assertEquals("testuser@example.com", loginResponse.getUser
66     ().getEmail());
67
68     // Verify interactions
69     verify(mockValidator, times(1)).validate(loginRequest);
70     verify(mockUserRepository, times(1)).findByUserName("
71     testUser");
72     verify(mockPasswordEncoder, times(1)).matches("Test123", "
73     hashedPassword123");
74     verify(mockJwtService, times(1)).generateToken(mockUser);
75 }
76
77 @Test
78 @DisplayName("TC2: Login thất bại khi username không tồn tại")
79 void testLoginFailure_UserNotFound() {
80     // Arrange
81     LoginRequest loginRequest = new LoginRequest("
82     nonExistentUser", "Test123");
83
84     // Mock validator trả về không có lỗi
85     when(mockValidator.validate(loginRequest))
86         .thenReturn(Set.of());
87
88     // Mock repository trả về empty
89     when(mockUserRepository.findByUserName("nonExistentUser"))
90         .thenReturn(Optional.empty());
91
92     // Act
93     LoginResponse loginResponse = authService.authenticate(
94     loginRequest);
95
96     // Assert
97     assertFalse(loginResponse.isSuccess(), "Login phải thất bại
98     ");
99     assertEquals("Login thất bại với user name không tồn tại",
100     loginResponse.getMessage());
101     assertNull(loginResponse.getToken(), "Token phải là null");
102     assertNull(loginResponse.getUser(), "UserDto phải là null
103     khi thất bại");
104
105     // Verify interactions
106     verify(mockValidator, times(1)).validate(loginRequest);
107     verify(mockUserRepository, times(1)).findByUserName("
108     nonExistentUser");
109     verify(mockPasswordEncoder, never()).matches(anyString(),
110     anyString());
111     verify(mockJwtService, never()).generateToken(any(User.
112     class));
113 }
114
115 @Test
116 @DisplayName("TC3: Login thất bại khi password sai")
117 void testLoginFailure_WrongPassword() {
118     // Arrange
119     LoginRequest loginRequest = new LoginRequest("testUser", "
```

```
WrongPassword123");
108
109 // Mock validator trả về không có lỗi
110 when(mockValidator.validate(loginRequest))
111     .thenReturn(Set.of());
112
113 // Mock dependencies
114 when(mockUserRepository.findByUserName("testUser"))
115     .thenReturn(Optional.of(mockUser));
116 when(mockPasswordEncoder.matches("WrongPassword123", "
hashedPassword123"))
117     .thenReturn(false);
118
119 // Act
120 LoginResponse loginResponse = authService.authenticate(
loginRequest);
121
122 // Assert
123 assertFalse(loginResponse.isSuccess(), "Login phải thất bại
");
124 assertEquals("Login với password sai", loginResponse.
getMessage());
125 assertNull(loginResponse.getToken(), "Token phải là null");
126 assertNull(loginResponse.getUser(), "UserDto phải là null
khi thất bại");
127
128 // Verify interactions
129 verify(mockValidator, times(1)).validate(loginRequest);
130 verify(mockUserRepository, times(1)).findByUserName("
testUser");
131 verify(mockPasswordEncoder, times(1)).matches("
WrongPassword123", "hashedPassword123");
132 verify(mockJwtService, never()).generateToken(any(User.
class));
133 }
134 @Test
135 @DisplayName("TC4: Cả username và password đều null - Validator
phải trả về nhiều lỗi")
136 void validate_withBothNull_shouldReturnMultipleErrors() {
137     // Arrange
138     var request = new LoginRequest(null, null);
139
140     Set<ConstraintViolation<LoginRequest>> violations =
realValidator.validate(request);
141     // Act
142
143     // Assert
144     assertEquals(2, violations.size(), "Phải có đúng 2 lỗi");
145
146     boolean isHasUserNameError = violations.stream()
147         .anyMatch(v -> v.getMessage().contains("Username kh
ông đ ợc để trống"));
148     assertTrue(isHasUserNameError);
149
150     boolean isHasPasswordError = violations.stream()
151         .anyMatch(v -> v.getMessage().contains("Password kh
```

```
152     ông ợ ọc ế trống"));
153     assertTrue(isHasPasswordError);
154 }
155 }
```

Listing 3: AuthServiceTest.java

b) Test validation methods riêng lẻ (1 điểm)

```
1 @Test
2     @DisplayName("TC5: Login thất bại với validation error -
3     username null")
4     void testLoginFailure_ValidationError_NullUsername() {
5         // Arrange
6         LoginRequest loginRequest = new LoginRequest(null, "Test123
7         ");
8         // Mock validator trá về violation
9         ConstraintViolation<LoginRequest> violation = mock(
10        ConstraintViolation.class);
11        when(violation.getMessage()).thenReturn("Username không ợ
12        c ế trống");
13        when(mockValidator.validate(loginRequest))
14            .thenReturn(Set.of(violation));
15
16        // Act
17        LoginResponse loginResponse = authService.authenticate(
18        loginRequest);
19
20        // Assert
21        assertFalse(loginResponse.isSuccess(), "Login phải thất bại
22        do validation");
23        assertEquals("Username không ợ ọc ế trống", loginResponse.
24        getMessage());
25        assertNull(loginResponse.getToken());
26        assertNull(loginResponse.getUser(), "UserDto phải là null
27        khi validation fail");
28
29        // Verify không có interaction với dependencies khác
30        verify(mockValidator, times(1)).validate(loginRequest);
31        verify(mockUserRepository, never()).findByUserName(
32        anyString());
33        verify(mockPasswordEncoder, never()).matches(anyString(),
34        anyString());
35        verify(mockJwtService, never()).generateToken(any(User.
36        class));
37    }
38
39 @Test
40     @DisplayName("TC6: Login thất bại với validation error -
41     password null")
42     void testLoginFailure_ValidationError_NullPassword() {
43         // Arrange
44         LoginRequest loginRequest = new LoginRequest("testUser",
45         null);
46     }
```

```
34 // Mock validator trả về violation
35 ConstraintViolation<LoginRequest> violation = mock(
36     ConstraintViolation.class);
37 when(violation.getMessage()).thenReturn("Password không đ ợ
38 c để trống");
39 when(mockValidator.validate(loginRequest))
40     .thenReturn(Set.of(violation));
41
42 // Act
43 LoginResponse loginResponse = authService.authenticate(
44     loginRequest);
45
46 // Assert
47 assertFalse(loginResponse.isSuccess(), "Login phải thất bại
48 do validation");
49 assertEquals("Password không đ ợ c để trống", loginResponse.
50     getMessage());
51 assertNull(loginResponse.getToken());
52
53 // Verify không có interaction với dependencies
54 verify(mockValidator, times(1)).validate(loginRequest);
55 verify(mockUserRepository, never()).findByUserName(
56     anyString());
57 }
58
59 @Test
60 @DisplayName("TC7: Login thất bại với validation error -
61 username quá ngắn")
62 void testLoginFailure_ValidationError_ShortUsername() {
63     // Arrange
64     LoginRequest req = new LoginRequest("ab", "Test123");
65
66     //Act
67     Set<ConstraintViolation<LoginRequest>> violations =
68     realValidator.validate(req);
69
70     //Assert
71     assertFalse(violations.isEmpty(), "Phải có lỗi validate");
72     boolean hasSizeError = violations.stream()
73         .anyMatch(v -> v.getMessage().contains("từ 3 đến 50
74 ký tự"));
75     assertTrue(hasSizeError, "Phải chứa thông báo về độ dài
76 username");
77 }
78
79 @Test
80 @DisplayName("TC8: Login thất bại với validation error -
81 password quá ngắn")
82 void testLoginFailure_ValidationError_ShortPassword() {
83     //Arrange
84     LoginRequest req = new LoginRequest("testUser", "12345");
85
86     // Act
87     Set<ConstraintViolation<LoginRequest>> violations =
```



```
realValidator.validate(req);
78
79 //Assert
80 assertFalse(violations.isEmpty(), "Phải có lỗi validate
password");
81 boolean isHasError = violations.stream()
82     .anyMatch(v -> v.getMessage().contains("Password ph
ái từ 6 đến 100 ký tự"));
83 assertTrue(isHasError, "Phải có thông báo lỗi password");
84
85 }
86
87 @Test
88 @DisplayName("TC9: Login thất bại với validation error -
username có ký tự đặc biệt")
89 void testLoginFailure_ValidationError_InvalidUsernameChars() {
90     //Arrange
91     LoginRequest req = new LoginRequest("user@#$$", "Test1234")
;
92
93     //Act
94     Set<ConstraintViolation<LoginRequest>> violations =
realValidator.validate(req);
95
96     //Assert
97     assertFalse(violations.isEmpty(), "Phải có lỗi username có
chứa ký tự đặc biệt");
98     boolean isHasError = violations.stream()
99         .anyMatch(v -> v.getMessage().contains("Username ch
í chứa chữ, số, và ký tự (-, ., _)"));
100     assertTrue(isHasError, "Phải có thông báo lỗi username có k
ý tự đặc biệt");
101
102 }
103
104 @Test
105 @DisplayName("TC10: Login thất bại với validation error -
password thiếu chữ cái")
106 void testLoginFailure_ValidationError_PasswordMissingLetter() {
107     //Arrange
108     LoginRequest req = new LoginRequest("testUser", "123456");
109
110     //Act
111     Set<ConstraintViolation<LoginRequest>> violations =
realValidator.validate(req);
112
113     //Assert
114     assertFalse(violations.isEmpty(), "Phải có lỗi password thi
ếu chữ cái");
115     boolean isHasError = violations.stream()
116         .anyMatch(v -> v.getMessage().contains("Password ph
ái chứa ít nhất 1 chữ cái"));
117     assertTrue(isHasError, "Phải có thông báo lỗi password thié
u chữ cái");
118 }
```

```
119
120
121     @Test
122     @DisplayName("TC11: Login thất bại với validation error -
password thiếu số")
123     void testLoginFailure_ValidationError_PasswordMissingNumber() {
124         // Arrange
125         LoginRequest req = new LoginRequest("testUser", "
testpassword");
126
127         //Act
128         Set<ConstraintViolation<LoginRequest>> violations =
realValidator.validate(req);
129
130         //Assert
131         assertFalse(violations.isEmpty(), "Phải có lỗi password thi
éu số");
132         boolean isHasError = violations.stream()
133             .anyMatch(v -> v.getMessage().contains("Password ph
ái chứa ít nhất 1 chữ số"));
134         assertTrue(isHasError, "Phải có thông báo lỗi password thié
u số");
135     }
```

Listing 4: AuthServiceTest.java

c) Coverage $\geq 85\%$ cho AuthService (1 điểm)

coverage_backend.jpg

3.2 Câu 2.2: Product - Unit Tests Frontend và Backend (10 điểm)

3.2.1 Frontend Unit Tests - Product Validation (5 điểm)

a) Viết unit tests cho validateProduct() (3 điểm):

```
1 describe('validateProduct - Kiểm tra tên sản phẩm', () => {  
2   test('trả về lỗi khi tên rỗng', () => {  
3     const product = { name: '', price: 100, quantity: 10,  
4       description: 'Valid description', category: 'Electronics' };  
5     const result = validateProduct(product);  
6  
7     expect(result.isValid).toBe(false);  
8     expect(result.errors.name).toBe('Tên sản phẩm không được để trống');  
9   });  
10  
11   ....  
12  
13   test('hợp lệ khi tên hợp lệ', () => {  
14     const product = { name: 'Valid Product Name', price: 100,  
15       quantity: 10, description: 'Valid description', category: 'Electronics' };  
16   });  
17 }
```

```
14     const result = validateProduct(product);
15
16     expect(result.errors.name).toBeUndefined();
17   });
18 });
19
20 describe('validateProduct - Kiểm tra giá (kiểm thú biên)', () => {
21   test('trả về lỗi khi giá là undefined', () => {
22     const product = { name: 'Product', price: undefined, quantity:
23       10, description: 'Valid description', category: 'Electronics' };
24     const result = validateProduct(product);
25
26     expect(result.isValid).toBe(false);
27     expect(result.errors.price).toBe('Giá sản phẩm không đ ọc đ ể tr
28       óng');
29   });
30
31   .....
32
33   test('hợp lệ khi giá hợp lệ (tr òng hợp điển hình)', () => {
34     const product = { name: 'Product', price: 50000, quantity: 10,
35       description: 'Valid description', category: 'Electronics' };
36     const result = validateProduct(product);
37
38     expect(result.errors.price).toBeUndefined();
39   });
40 });
41
42 describe('validateProduct - Kiểm tra số l ợng', () => {
43   test('trả về lỗi khi số l ợng là undefined', () => {
44     const product = { name: 'Product', price: 100, quantity:
45       undefined, description: 'Valid description', category: '
46       Electronics' };
47     const result = validateProduct(product);
48
49     expect(result.isValid).toBe(false);
50     expect(result.errors.quantity).toBe('Số l ợng không đ ọc đ ể tró
51       ng');
52   });
53
54   test('trả về lỗi khi số l ợng là null', () => {
55     const product = { name: 'Product', price: 100, quantity: null,
56       description: 'Valid description', category: 'Electronics' };
57     const result = validateProduct(product);
58
59     expect(result.isValid).toBe(false);
60     expect(result.errors.quantity).toBe('Số l ợng không đ ọc đ ể tró
61       ng');
62   });
63
64   test('trả về lỗi khi số l ợng là chuỗi r ỗng', () => {
65     const product = { name: 'Product', price: 100, quantity: '',
66       description: 'Valid description', category: 'Electronics' };
67     const result = validateProduct(product);
```

```
60
61     expect(result.isValid).toBe(false);
62     expect(result.errors.quantity).toBe('Số lượng không được để trống');
63 });
64     .....
65
66 test('hợp lệ khi số lượng là số nguyên hợp lệ', () => {
67     const product = { name: 'Product', price: 100, quantity: 50,
68       description: 'Valid description', category: 'Electronics' };
69     const result = validateProduct(product);
70
71     expect(result.errors.quantity).toBeUndefined();
72 });
73
74 describe('validateProduct - Kiểm tra độ dài mô tả', () => {
75     test('trả về lỗi khi mô tả rỗng', () => {
76         const product = { name: 'Product', price: 100, quantity: 10,
77           description: '', category: 'Electronics' };
78         const result = validateProduct(product);
79
80         expect(result.isValid).toBe(false);
81         expect(result.errors.description).toBe('Mô tả không được để trống');
82     });
83     .....
84
85     test('hợp lệ khi mô tả hợp lệ', () => {
86         const product = {
87             name: 'Product',
88             price: 100,
89             quantity: 10,
90             description: 'This is a valid product description',
91             category: 'Electronics'
92         };
93         const result = validateProduct(product);
94
95         expect(result.errors.description).toBeUndefined();
96     });
97 });
98
99 describe('validateProduct - Kiểm tra danh mục', () => {
100     test('trả về lỗi khi danh mục rỗng', () => {
101         const product = { name: 'Product', price: 100, quantity: 10,
102           description: 'Valid description', category: '' };
103         const result = validateProduct(product);
104
105         expect(result.isValid).toBe(false);
106         expect(result.errors.category).toBe('Danh mục không được để trống');
107     });
108     .....
109     test('hợp lệ khi danh mục là Other', () => {
```

```
110     const product = { name: 'Product', price: 100, quantity: 10,  
111       description: 'Valid description', category: 'Other' };  
112     const result = validateProduct(product);  
113  
114     expect(result.errors.category).toBeUndefined();  
115   });  
116  
117   describe('validateProduct - Kiểm tra tích hợp', () => {  
118     test('trả về isValid true khi tất cả trường hợp hợp lệ', () => {  
119       const product = {  
120         name: 'Valid Product',  
121         price: 100,  
122         quantity: 10,  
123         description: 'This is a valid description',  
124         category: 'Electronics'  
125       };  
126       const result = validateProduct(product);  
127  
128       expect(result.isValid).toBe(true);  
129       expect(result.errors).toEqual({});  
130     });  
131  
132     test('trả về nhiều lỗi khi nhiều trường không hợp lệ', () => {  
133       const product = {  
134         name: '',  
135         price: -100,  
136         quantity: 'abc',  
137         description: 'Short',  
138         category: ''  
139       };  
140       const result = validateProduct(product);  
141  
142       expect(result.isValid).toBe(false);  
143       expect(result.errors.name).toBeDefined();  
144       expect(result.errors.price).toBeDefined();  
145       expect(result.errors.quantity).toBeDefined();  
146       expect(result.errors.description).toBeDefined();  
147       expect(result.errors.category).toBeDefined();  
148     });  
149   });
```

Listing 5: validateProduct()

- b) Viết tests cho Product form component (1 điểm)
- c) Coverage ≥ 90 (1 điểm)

coverage_backend.jpg

3.2.2 Backend Unit Tests - Product Service (5 điểm)

a) Test CRUD operations (4 điểm):

```
1 @DisplayName("ProductService Unit Test")
2 @ExtendWith(MockitoExtension.class)
3 public class ProductServiceTest {
4
5     @Mock
6     private ProductRepository productRepository;
7
8     @Mock
9     private Validator mockValidator;
10
11     private ProductService productService;
12
13     @BeforeEach
14     void setUp() {
15         productService = new ProductService(productRepository,
16         mockValidator);
17     }
```

```
17
18 // A) TEST CREATE - Bao gồm Boundary, Edge Cases và Negative
    Tests
19
20 @Nested
21 @DisplayName("Test createProduct() method - Simplified")
22 class CreateProductTests {
23
24     // ===== POSITIVE TESTS =====
25
26     @Test
27     @DisplayName("TC1: Tạo sản phẩm thành công - Happy path")
28     void testCreateProduct_Success() {
29         // Arrange
30         CreateProductRequest request = new CreateProductRequest
31         ("Laptop", 15000.0, "Gaming laptop", 10, "Electronics");
32         Product savedProduct = new Product(1L, "Electronics", "
33         Gaming laptop", 10, "Laptop", 15000.0);
34
35         when(mockValidator.validate(request)).thenReturn(Set.of
36         ());
37         when(productRepository.save(any(Product.class))).
38         thenReturn(savedProduct);
39
40         // Act
41         ProductDto result = productService.createProduct(
42         request);
43
44         // Assert
45         assertNotNull(result);
46         assertEquals("Laptop", result.getProductName());
47         assertEquals(15000.0, result.getPrice());
48         verify(mockValidator, times(1)).validate(request);
49         verify(productRepository, times(1)).save(any(Product.
50         class));
51     }
52 }
53
54 @Nested
55 @DisplayName("Test getProductById() method")
56 class GetProductTests {
57
58     @Test
59     @DisplayName("TC14: Lấy sản phẩm thành công theo ID")
60     void testGetProductById_Success() {
61         Product product = new Product(1L, "Electronics", "
62         Gaming laptop", 10, "Laptop", 15000.0);
63         when(productRepository.findById(1L)).thenReturn(
64         Optional.of(product));
65
66         ProductDto result = productService.getProductById(1L);
67
68         assertNotNull(result);
69         assertEquals(1L, result.getId());
70         assertEquals("Laptop", result.getProductName());
71         verify(productRepository, times(1)).findById(1L);
72     }
73 }
```



```
64     }
65
66     @Test
67     @DisplayName("TC15: Lấy sản phẩm thất bại - ID không tồn tại")
68     void testGetProductById_NotFound() {
69         when(productRepository.findById(999L)).thenReturn(Optional.empty());
70
71         NoSuchElementException exception = assertThrows(
72             NoSuchElementException.class,
73             () -> productService.getProductById(999L)
74         );
75
76         assertTrue(exception.getMessage().contains("Product not found with id: 999"));
77         verify(productRepository, times(1)).findById(999L);
78     }
79 }
80
81 @Nested
82 @DisplayName("Test updateProduct() method")
83 class UpdateProductTests {
84
85     @Test
86     @DisplayName("TC16: Cập nhật sản phẩm thành công")
87     void testUpdateProduct_Success() {
88         UpdateProductRequest request = new UpdateProductRequest("Updated Laptop", 20000.0, "Updated description", 15, "Electronics");
89         Product existingProduct = new Product(1L, "Electronics", "Old description", 10, "Laptop", 15000.0);
90         Product updatedProduct = new Product(1L, "Electronics", "Updated description", 15, "Updated Laptop", 20000.0);
91
92         when(mockValidator.validate(request)).thenReturn(Set.of());
93         when(productRepository.findById(1L)).thenReturn(Optional.of(existingProduct));
94         when(productRepository.save(any(Product.class))).thenReturn(updatedProduct);
95
96         ProductDto result = productService.updateProduct(1L, request);
97
98         assertNotNull(result);
99         assertEquals("Updated Laptop", result.getProductName());
100
101         assertEquals(20000.0, result.getPrice());
102     }
103
104     // D) TEST DELETE
105
106     @Nested
107     @DisplayName("Test deleteProduct() method")
```

```
108     class DeleteProductTests {
109
110         @Test
111         @DisplayName("TC20: Xóa sản phẩm thành công")
112         void testDeleteProduct_Success() {
113             Product product = new Product(1L, "Electronics", "
114 Gaming laptop", 10, "Laptop", 15000.0);
115             when(productRepository.findById(1L)).thenReturn(
116 Optional.of(product));
117             doNothing().when(productRepository).deleteById(1L);
118
119             productService.deleteProduct(1L);
120
121             verify(productRepository, times(1)).findById(1L);
122             verify(productRepository, times(1)).deleteById(1L);
123         }
124
125         @Test
126         @DisplayName("TC21: Xóa sản phẩm thất bại - ID không tồn t
127 a")
128         void testDeleteProduct_NotFound() {
129             when(productRepository.findById(999L)).thenReturn(
130 Optional.empty());
131
132             NoSuchElementException exception = assertThrows(
133                 NoSuchElementException.class,
134                 () -> productService.deleteProduct(999L)
135             );
136
137             assertTrue(exception.getMessage().contains("Product not
138 found with id: 999"));
139         }
140     }
141
142     // E) TEST GET ALL (PAGINATION)
143
144     @Test
145     @DisplayName("TC22: Lấy danh sách sản phẩm với pagination")
146     void testGetAll_Success() {
147         Pageable pageable = PageRequest.of(0, 10);
148         List<Product> products = Arrays.asList(
149             new Product(1L, "Electronics", "Product 1", 10,
150 "Laptop", 15000.0),
151             new Product(2L, "Books", "Product 2", 20, "Book
152 ", 50.0)
153         );
154         Page<Product> productPage = new PageImpl<>(products,
155             pageable, products.size());
156
157         when(productRepository.findAll(pageable)).thenReturn(
158             productPage);
159
160         Page<ProductDto> result = productService.getAll(
161             pageable);
162
163         assertNotNull(result);
164     }
```

```
154         assertEquals(2, result.getContent().size());  
155         assertEquals(0, result.getNumber());  
156     }  
157 }
```

Listing 6: ProductServiceTest.java

b) Coverage ≥ 85 cho ProductService (1 điểm)

coverage_backend.jpg

4 Câu 3: Integration Testing

4.1 Câu 3.1: Login - Integration Testing (10 điểm)

4.1.1 Frontend Component Integration (5 điểm)

a) Test rendering và user interactions (2 điểm)

```
1 describe('Test 1: Rendering và User Interactions', () => {
2
3   test('nên render login form component thành công', () => {
4     render(
5       <MemoryRouter>
6         <LoginForm />
7       </MemoryRouter>
8     );
9
10    // Query cụ thể h n để tránh matching multiple elements
11    const heading = screen.getByRole('heading', { level: 1 });
12    expect(heading).toHaveTextContent(/Đăng Nhập/i);
13    expect(screen.getByText(/Chào mừng bạn quay lại/i)).
14      toBeInTheDocument();
15    });
16
17    test('nên có thể nhập username vào input', async () => {
18      const user = userEvent.setup();
19      render(
20        <MemoryRouter>
21          <LoginForm />
22        </MemoryRouter>
23      );
24
25      const usernameInput = screen.getByPlaceholderText(/
26        your_username/i);
27      await user.type(usernameInput, 'testuser');
28
29      expect(usernameInput.value).toBe('testuser');
30    });
31
32    test('nên có thể nhập password vào input', async () => {
33      const user = userEvent.setup();
34      render(
35        <MemoryRouter>
36          <LoginForm />
37        </MemoryRouter>
38      );
39
40      const passwordInput = screen.getByPlaceholderText(/
41        /i);
42      await user.type(passwordInput, 'password123');
43
44      expect(passwordInput.value).toBe('password123');
45    });
46
47    test('nên disable button khi loading', async () => {
48      const user = userEvent.setup();
```

```
46
47   render(
48     <MemoryRouter>
49       <LoginForm />
50     </MemoryRouter>
51   );
52
53   const usernameInput = screen.getByPlaceholderText(/
54   your_username/i);
55   const passwordInput = screen.getByPlaceholderText(/
56   /i);
57   const submitButton = screen.getByRole('button', { name: /Đăng
58   Nhập/i });
59
60   await user.type(usernameInput, 'testuser');
61   await user.type(passwordInput, 'password123');
62
63   // Button should exist and be clickable
64   expect(submitButton).toBeInTheDocument();
65   expect(submitButton).toBeEnabled();
66 });
67 });
68 });
```

Listing 7: rendering và user interaction

b) Test form submission và API calls (2 điểm)

```
1   describe('Test 2: Form Submission và API Calls', () => {
2
3     test('nên gọi login API khi form submit với dữ liệu hợp lệ -
4     admin', async () => {
5       const user = userEvent.setup();
6       authService.login.mockResolvedValue({
7         success: true,
8         message: 'Login thành công',
9         user: { userName: 'admin', email: 'admin@example.com' }
10      });
11
12     render(
13       <MemoryRouter>
14         <LoginForm />
15       </MemoryRouter>
16     );
17
18     const usernameInput = screen.getByPlaceholderText(/
19     your_username/i);
20     const passwordInput = screen.getByPlaceholderText(/
21     /i);
22     const submitButton = screen.getByRole('button', { name: /Đăng
23     Nhập/i });
24
25     await user.type(usernameInput, 'admin');
26     await user.type(passwordInput, 'admin123');
27     await user.click(submitButton);
28   });
29 }
```

```
25 // Verify API call
26 await waitFor(() => {
27   expect(authService.login).toHaveBeenCalledWith('admin', '
admin123');
28 });
29 });
30
31 test('nên hiển thị success message khi API trả về success -
testuser', async () => {
32   const user = userEvent.setup();
33   authService.login.mockResolvedValue({
34     success: true,
35     message: 'Login thành công',
36     user: { userName: 'testuser', email: 'testuser@example.com'
37   });
38
39   render(
40     <MemoryRouter>
41       <LoginForm />
42     </MemoryRouter>
43   );
44
45   const usernameInput = screen.getByPlaceholderText(/
your_username/i);
46   const passwordInput = screen.getByPlaceholderText(/
/i);
47   const submitButton = screen.getByRole('button', { name: /Đăng
Nhập/i });
48
49   await user.type(usernameInput, 'testuser');
50   await user.type(passwordInput, 'test1234');
51   await user.click(submitButton);
52
53   // Verify API called
54   await waitFor(() => {
55     expect(authService.login).toHaveBeenCalledWith('testuser',
'test1234');
56   });
57 });
58 });
59
```

Listing 8: form submission và API calls

c) Test error handling và success messages (1 điểm)

```
1 describe('Test 3: Error Handling và Success Messages', () => {
2
3   test('nên hiển thị error khi username trống', async () => {
4     const user = userEvent.setup();
5     render(
6       <MemoryRouter>
7         <LoginForm />
8       </MemoryRouter>
9     );
```

```
10
11     const submitButton = screen.getByRole('button', { name: /Đăng
    Nhập/i });
12     await user.click(submitButton);
13
14     await waitFor(() => {
15         expect(screen.getByText(/Username không đ ợc đ ể trống/i)).
    toBeInTheDocument();
16     });
17 });
18
19 test('nên hiển thị error khi username không hợp lệ', async ()
=> {
20     const user = userEvent.setup();
21     render(
22 <MemoryRouter>
23   <LoginForm />
24 </MemoryRouter>
25 );
26
27     const usernameInput = screen.getByPlaceholderText(/
    your_username/i);
28     const passwordInput = screen.getByPlaceholderText(/
        /i);
29     const submitButton = screen.getByRole('button', { name: /Đăng
    Nhập/i });
30
31     await user.type(usernameInput, 'ab');
32     await user.type(passwordInput, 'password123');
33     await user.click(submitButton);
34
35     await waitFor(() => {
36         const errorMessage = screen.getByText(/Username phải ít nhấ
    t 3 ký tự/i);
37         expect(errorMessage).toBeInTheDocument();
38     });
39 });
40
41 test('nên hiển thị error khi API trả về error', async () => {
42     const user = userEvent.setup();
43     authService.login.mockRejectedValue(new Error('Invalid
    credentials'));
44
45     render(
46 <MemoryRouter>
47   <LoginForm />
48 </MemoryRouter>
49 );
50
51     const usernameInput = screen.getByPlaceholderText(/
    your_username/i);
52     const passwordInput = screen.getByPlaceholderText(/
        /i);
53     const submitButton = screen.getByRole('button', { name: /Đăng
    Nhập/i });
54
```

```
55     await user.type(usernameInput, 'testuser');
56     await user.type(passwordInput, 'wrongpassword1');
57     await user.click(submitButton);
58
59     // Verify error message
60     await waitFor(() => {
61         const errorMessage = screen.queryByText(/Invalid
credentials/Error/i);
62         expect(errorMessage).toBeInTheDocument();
63     });
64 });
65
66 test('nên hiển thị success message khi login thành công', async
() => {
67     const user = userEvent.setup();
68     authService.login.mockResolvedValue({
69         success: true,
70         message: 'Login thành công',
71         user: { userName: 'admin', email: 'admin@example.com' }
72     });
73
74     render(
75     <MemoryRouter>
76     <LoginForm />
77     </MemoryRouter>
78 );
79
80     const usernameInput = screen.getByPlaceholderText(/
your_username/i);
81     const passwordInput = screen.getByPlaceholderText(/
/i);
82     const submitButton = screen.getByRole('button', { name: /Đăng
Nhập/i });
83
84     await user.type(usernameInput, 'admin');
85     await user.type(passwordInput, 'admin123');
86     await user.click(submitButton);
87
88     // Verify API called
89     await waitFor(() => {
90         expect(authService.login).toHaveBeenCalled();
91     });
92 });
93
94 });
95
```

Listing 9: error handling và success messages

4.1.2 Backend API Integration (5 điểm)

Test API endpoints của Login với MockMvc:

- a) Test POST /api/auth/login endpoint (3 điểm)


```
1 @WebMvcTest(value = AuthController.class,  
2     excludeFilters = @ComponentScan.Filter(type = FilterType.  
3     ASSIGNABLE_TYPE,  
4     classes = {com.flogin.service.SecurityConfig.class, com.  
5     flogin.filter.JwtAuthenticationFilter.class}))  
6 @AutoConfigureMockMvc(addFilters = false)  
7 @DisplayName("Login API Integration Tests")  
8 public class AuthControllerIntegrationTest {  
9  
10     @Autowired  
11     private MockMvc mockMvc;  
12  
13     @Autowired  
14     private ObjectMapper objectMapper;  
15  
16     @MockitoBean  
17     private AuthService authService;  
18  
19     @Nested  
20     @DisplayName("A) Test POST /api/auth/login endpoint")  
21     class LoginEndpointTests {  
22  
23         @Test  
24         @DisplayName("1. Login thành công với credentials hợp lệ")  
25         void testLoginSuccess() throws Exception {  
26             // Arrange: Chuẩn bị dữ liệu test  
27             LoginRequest request = new LoginRequest("testuser", "  
28             Test123");  
29             UserDto userDto = new UserDto("testuser", "  
30             testuser@example.com");  
31             LoginResponse mockResponse = new LoginResponse(  
32                 true,  
33                 "Login thành công",  
34                 "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.token123",  
35                 userDto  
36             );  
37  
38             // Mock service trả về response thành công  
39             when(authService.authenticate(any(LoginRequest.class)))  
40                 .thenReturn(mockResponse);  
41  
42             // Act & Assert: Thực hiện request và verify kết quả  
43             mockMvc.perform(post("/api/auth/login")  
44                 .contentType(MediaType.APPLICATION_JSON)  
45                 .content(objectMapper.writeValueAsString(  
46                 request)))  
47                 .andExpect(status().isOk()) // Expect 200 OK  
48                 .andExpect(jsonPath("$.success").value(true))  
49                 .andExpect(jsonPath("$.message").value("Login  
50                 thành công"))  
51                 .andExpect(jsonPath("$.token").exists())  
52                 .andExpect(jsonPath("$.token").value("  
53                 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.token123"))  
54                 .andExpect(jsonPath("$.user").exists())  
55                 .andExpect(jsonPath("$.user.userName").value(""
```

```
testuser"))
49         .andExpect(jsonPath("$.user.email").value("
testuser@example.com"));
50     }
51
52     @Test
53     @DisplayName("2. Login thất bại - Username không tồn tại")
54     void testLoginFailure_UserNotFound() throws Exception {
55         // Arrange: User không tồn tại trong database
56         LoginRequest request = new LoginRequest("nonexistuser",
"Test123");
57         LoginResponse mockResponse = new LoginResponse(
58             false,
59             "Login thất bại với user name không tồn tại"
60         );
61
62         when(authService.authenticate(any(LoginRequest.class)))
63             .thenReturn(mockResponse);
64
65         // Act & Assert: Expect 401 Unauthorized
66         mockMvc.perform(post("/api/auth/login")
67             .contentType(MediaType.APPLICATION_JSON)
68             .content(objectMapper.writeValueAsString(
request))))
69             .andExpect(status().isUnauthorized()) // Expect
401
70             .andExpect(jsonPath("$.success").value(false))
71             .andExpect(jsonPath("$.message").value("Login
thất bại với user name không tồn tại"))
72             .andExpect(jsonPath("$.token").doesNotExist())
73             .andExpect(jsonPath("$.user").doesNotExist());
74     }
75
76
77 }
```

Listing 10: AuthControllerIntegrationTest.java

b) Test response structure và status codes (1 điểm)

```
1 @Nested
2     @DisplayName("B) Test Response Structure và Status Codes")
3     class ResponseStructureTests {
4
5         @Test
6         @DisplayName("1. Response structure có đầy đủ các field khi
login thành công")
7         void testSuccessResponseStructure() throws Exception {
8             // Arrange
9             LoginRequest request = new LoginRequest("testuser", "
Test123");
10            UserDto userDto = new UserDto("testuser", "
testuser@example.com");
11            LoginResponse mockResponse = new LoginResponse(
12                true,
13                "Login thành công",
```

```
14         "token.jwt",
15         userDto
16     );
17
18     when(authService.authenticate(any(LoginRequest.class)))
19         .thenReturn(mockResponse);
20
21     // Act & Assert: Verify response có đầy đủ field
22     mockMvc.perform(post("/api/auth/login")
23         .contentType(MediaType.APPLICATION_JSON)
24         .content(objectMapper.writeValueAsString(
25             request)))
26         .andExpect(status().isOk())
27         .andExpect(jsonPath("$.success").exists())
28         .andExpect(jsonPath("$.success").isBoolean())
29         .andExpect(jsonPath("$.message").exists())
30         .andExpect(jsonPath("$.message").isString())
31         .andExpect(jsonPath("$.token").exists())
32         .andExpect(jsonPath("$.token").isString())
33         .andExpect(jsonPath("$.user").exists())
34         .andExpect(jsonPath("$.user.userName").exists())
35     )
36     .andExpect(jsonPath("$.user.email").exists());
37
38     @Test
39     @DisplayName("2. Response structure khi login thất bại (không có token)")
40     void testFailureResponseStructure() throws Exception {
41         // Arrange
42         LoginRequest request = new LoginRequest("wronguser", "Test123");
43         LoginResponse mockResponse = new LoginResponse(
44             false,
45             "Login thất bại với user name không tồn tại"
46         );
47
48         when(authService.authenticate(any(LoginRequest.class)))
49             .thenReturn(mockResponse);
50
51         // Act & Assert: Verify response không có token và user khi thất bại
52         mockMvc.perform(post("/api/auth/login")
53             .contentType(MediaType.APPLICATION_JSON)
54             .content(objectMapper.writeValueAsString(
55                 request)))
56             .andExpect(status().isUnauthorized())
57             .andExpect(jsonPath("$.success").value(false))
58             .andExpect(jsonPath("$.message").exists())
59             .andExpect(jsonPath("$.token").doesNotExist())
60             .andExpect(jsonPath("$.user").doesNotExist());
61         // Token không tồn tại khi fail
62         // User cũng không tồn tại khi fail
63     }
```

```
62     @Test
63     @DisplayName("3. Status code 200 OK khi login thành công")
64     void testStatusCode_200_OnSuccess() throws Exception {
65         // Arrange
66         LoginRequest request = new LoginRequest("testuser", "
Test123");
67         LoginResponse mockResponse = new LoginResponse(true, "
Login thành công", "token123");
68
69         when(authService.authenticate(any(LoginRequest.class)))
70             .thenReturn(mockResponse);
71
72         // Act & Assert: Verify status code = 200
73         mockMvc.perform(post("/api/auth/login")
74             .contentType(MediaType.APPLICATION_JSON)
75             .content(objectMapper.writeValueAsString(
request))))
76             .andExpect(status().isOk())
77             .andExpect(status().is(200));
78     }
79
80
81
82
83     @Test
84     @DisplayName("6. Response Content-Type là application/json"
)
85     void testResponseContentType() throws Exception {
86         // Arrange
87         LoginRequest request = new LoginRequest("testuser", "
Test123");
88         UserDto userDto = new UserDto("testuser", "
testuser@example.com");
89         LoginResponse mockResponse = new LoginResponse(true, "
Login thành công", "token123", userDto);
90
91         when(authService.authenticate(any(LoginRequest.class)))
92             .thenReturn(mockResponse);
93
94         // Act & Assert: Verify Content-Type header
95         mockMvc.perform(post("/api/auth/login")
96             .contentType(MediaType.APPLICATION_JSON)
97             .content(objectMapper.writeValueAsString(
request))))
98             .andExpect(status().isOk())
99             .andExpect(content().contentType(MediaType.
APPLICATION_JSON));
100     }
101 }
```

Listing 11: AuthControllerIntegrationTest.java

c) Test CORS và headers (1 điểm)

```
1 @ @Nested
2     @DisplayName("C) Test CORS và Headers ")
```

```
3      class CorsAndHeadersTests {
4
5          @Test
6          @DisplayName("1. CORS - Access-Control-Allow-Origin header  
có tồn tại")
7          void testCors_AllowOriginHeader() throws Exception {
8              // Arrange
9              LoginRequest request = new LoginRequest("testuser", "
Test123");
10             UserDto userDto = new UserDto("testuser", "
testuser@example.com");
11             LoginResponse mockResponse = new LoginResponse(true, "
Login thành công", "token123", userDto);
12
13             when(authService.authenticate(any(LoginRequest.class)))
14                 .thenReturn(mockResponse);
15
16             // Act & Assert: Verify CORS header
17             mockMvc.perform(post("/api/auth/login")
18                 .contentType(MediaType.APPLICATION_JSON)
19                 .header("Origin", "http://localhost:3000") //
Giả lập request từ frontend
20                 .content(objectMapper.writeValueAsString(
request)))
21                 .andExpect(status().isOk())
22                 .andExpect(header().exists("Access-Control-
Allow-Origin"));
23         }
24
25
26         @Test
27         @DisplayName("4. Request accept header - application/json")
28         void testAcceptHeader_ApplicationJson() throws Exception {
29             // Arrange
30             LoginRequest request = new LoginRequest("testuser", "
Test123");
31             UserDto userDto = new UserDto("testuser", "
testuser@example.com");
32             LoginResponse mockResponse = new LoginResponse(true, "
Login thành công", "token123", userDto);
33
34             when(authService.authenticate(any(LoginRequest.class)))
35                 .thenReturn(mockResponse);
36
37             // Act & Assert: Verify server accept JSON
38             mockMvc.perform(post("/api/auth/login")
39                 .contentType(MediaType.APPLICATION_JSON)
40                 .accept(MediaType.APPLICATION_JSON) // Client y
êu cầu JSON response
41                 .content(objectMapper.writeValueAsString(
request)))
42                 .andExpect(status().isOk())
43                 .andExpect(content().contentType(MediaType.
APPLICATION_JSON));
44         }
45     }
```

46 }
}

Listing 12: AuthControllerIntegrationTest.java

4.2 Câu 3.2: Product - Integration Testing (10 điểm)

4.2.1 Frontend Component Integration (5 điểm)

a) Test ProductList component với API (2 điểm)

```
1 describe("Test 1: ProductList Component (ProductManagement) với  
API", () => {  
2   test("nên render ProductManagement component thành công", () =>  
3     {  
4       render(<ProductManagement />);  
5  
6       // Verify component renders  
7       const heading = screen.getByText("Quản Lý Sản Phẩm");  
8       expect(heading).toBeInTheDocument();  
9     });  
10  
11   test("nên có category filter dropdown", () => {  
12     render(<ProductManagement />);  
13  
14     const filterElements =  
15       screen.queryAllByRole("button").length > 0 ||  
16       screen.queryAllByRole("combobox").length > 0 ||  
17       screen.queryByText(/Tất cả|All|category|Category/i);  
18     expect(filterElements).toBeTruthy();  
19   });  
20  
21   test("nên filter products theo category", async () => {  
22     const user = userEvent.setup();  
23     render(<ProductManagement />);  
24  
25     // Tìm filter button hoặc select  
26     const filterButtons = screen.queryAllByRole("button");  
27     expect(filterButtons.length > 0).toBe(true);  
28   });  
29  
30   test("nên có action buttons (Edit, Delete, View)", () => {  
31     render(<ProductManagement />);  
32  
33     // Verify edit/delete buttons exist or can be added  
34     const buttons = screen.queryAllByRole("button");  
35     expect(buttons.length > 0).toBe(true);  
36   });  
37 }  
38 }
```

Listing 13: ProductList component với API

b) Test ProductForm component (create/edit) (2 điểm)

```
1 describe("Test 2: ProductForm Component (create/edit) ", () => {
2
3     test("nên validate product data tr ớc submit", async () => {
4         const user = userEvent.setup();
5         render(<ProductManagement />);
6
7         const addButton = screen
8             .queryAllByRole("button")
9             .find(
10                 (btn) =>
11                     btn.textContent?.includes("+") || btn.textContent?.
12                     includes("Thêm")
13             );
14
15         if (addButton) {
16             await user.click(addButton);
17             // Component renders successfully
18             const inputs = screen.queryAllByRole("textbox");
19             expect(inputs.length > 0).toBe(true);
20         }
21     });
22
23     test("nên có category dropdown/select", async () => {
24         const user = userEvent.setup();
25         render(<ProductManagement />);
26
27         // Verify select elements exist in page
28         const selects = screen.queryAllByRole("combobox");
29         expect(selects.length > 0).toBe(true);
30     });
31
32     test("nên có Submit/Save button", async () => {
33         const user = userEvent.setup();
34         render(<ProductManagement />);
35
36         const addButton = screen
37             .queryAllByRole("button")
38             .find(
39                 (btn) =>
40                     btn.textContent?.includes("+") || btn.textContent?.
41                     includes("Thêm")
42             );
43
44         if (addButton) {
45             await user.click(addButton);
46             const buttons = screen.queryAllByRole("button");
47             expect(buttons.length > 0).toBe(true);
48         }
49     });
50 }
```

Listing 14: ProductForm component (create/edit)

c) Test ProductDetail component (1 điểm)

```
1 describe("Test 3: ProductDetail", () => {
2   test("nên display product info: name, price, quantity, category", async () => {
3     render(<ProductManagement />);
4
5     // Check if table or list displays product info columns
6     await waitFor(() => {
7       const headers = screen.queryAllByText(
8         /tên|name|giá|price|số l ượng|quantity|loại|category/i
9       );
10      expect(headers.length > 0).toBe(true);
11    });
12  });
13
14  test("nên display product details khi click view/expand", async () => {
15    const user = userEvent.setup();
16    render(<ProductManagement />);
17
18    // Look for view buttons or expandable rows
19    const viewButtons = screen.queryAllByRole("button");
20    expect(viewButtons.length > 0).toBe(true);
21  });
22 });
23
```

Listing 15: ProductDetail component

4.2.2 Backend API Integration (5 điểm)

Test các API endpoints của Product:

- a) Test POST /api/products (Create) (1 điểm)
- b) Test GET /api/products (Read all) (1 điểm)
- c) Test GET /api/products/id (Read one) (1 điểm)
- d) Test PUT /api/products/id (Update) (1 điểm)
- e) Test DELETE /api/products/id (Delete) (1 điểm)

```
1 WebMvcTest(value = ProductController.class,
2   excludeFilters = @ComponentScan.Filter(type = FilterType.
3     ASSIGNABLE_TYPE,
4     classes = {com.flogin.service.SecurityConfig.class, com.flogin.
5       filter.JwtAuthenticationFilter.class}))
6 @AutoConfigureMockMvc(addFilters = false)
7 @DisplayName("Product API Integration Tests")
8 public class ProductControllerIntegrationTest {
9
10   @Autowired
11   private MockMvc mockMvc;
```



```
11  @Autowired
12  private ObjectMapper objectMapper;
13
14  @MockitoBean
15  private ProductService productService;
16
17
18  @Test
19  @DisplayName("1. Tạo product thành công với dữ liệu hợp lệ")
20  void testCreateProduct_Success() throws Exception {
21      // Arrange
22      CreateProductRequest requestDto = new CreateProductRequest("
23  Laptop", 15000000.0, "Gaming laptop", 10, "Electronics");
24      ProductDto responseDto = new ProductDto(1L, "Electronics",
25  15000000.0, "Laptop", "Gaming laptop", 10);
26
27      when(productService.createProduct(any(CreateProductRequest.
28  class)))
29          .thenReturn(responseDto);
30
31      // Act & Assert
32      mockMvc.perform(post("/api/products")
33          .contentType(MediaType.APPLICATION_JSON)
34          .content(objectMapper.writeValueAsString(requestDto))
35      )
36          .andExpect(status().isCreated()) // 201 Created
37          .andExpect(jsonPath("$.id").value(1))
38          .andExpect(jsonPath("$.productName").value("Laptop"))
39          .andExpect(jsonPath("$.price").value(15000000.0))
40          .andExpect(jsonPath("$.quantity").value(10))
41          .andExpect(jsonPath("$.category").value("Electronics
42  "));
43
44      verify(productService, times(1)).createProduct(any(
45  CreateProductRequest.class));
46  }
47
48  /**
49   * B) Test GET /api/products - Read All (1 điểm)
50   * Test lấy danh sách tất cả products
51   */
52  @Test
53  @DisplayName("2. Lấy danh sách products thành công")
54  void testGetAllProducts_Success() throws Exception {
55      // Arrange
56      List<ProductDto> products = Arrays.asList(
57          new ProductDto(1L, "Electronics", 15000000.0, "Laptop",
58  "Gaming laptop", 10),
59          new ProductDto(2L, "Electronics", 200000.0, "Mouse", "
60  Wireless mouse", 50)
61      );
62      Page<ProductDto> page = new PageImpl<>(products, PageRequest
63  .of(0, 10), products.size());
64
65      when(productService.getAll(any()))
```

```
57         .thenReturn(page);
58
59         // Act & Assert
60         mockMvc.perform(get("/api/products"))
61             .andExpect(status().isOk())
62             .andExpect(jsonPath("$.content", hasSize(2)))
63             .andExpect(jsonPath("$.content[0].productName").
value("Laptop"))
64             .andExpect(jsonPath("$.content[1].productName").
value("Mouse"))
65             .andExpect(jsonPath("$.totalElements").value(2));
66
67         verify(productService, times(1)).getAll(any());
68     }
69
70
71
72     @Test
73     @DisplayName("3. Lấy product theo ID thành công")
74     void testGetProductById_Success() throws Exception {
75         // Arrange
76         ProductDto product = new ProductDto(1L, "Electronics",
15000000.0, "Laptop", "Gaming laptop", 10);
77
78         when(productService.getProductById(1L))
79             .thenReturn(product);
80
81         // Act & Assert
82         mockMvc.perform(get("/api/products/1"))
83             .andExpect(status().isOk())
84             .andExpect(jsonPath("$.id").value(1))
85             .andExpect(jsonPath("$.productName").value("Laptop"))
86             .andExpect(jsonPath("$.price").value(15000000.0))
87             .andExpect(jsonPath("$.category").value("Electronics
"));
88
89         verify(productService, times(1)).getProductById(1L);
90     }
91
92
93
94     @Test
95     @DisplayName("4. Cập nhật product thành công")
96     void testUpdateProduct_Success() throws Exception {
97         // Arrange
98         UpdateProductRequest requestDto = new UpdateProductRequest("
Laptop Updated", 16000000.0, "New description", 15, "Electronics");
99         ProductDto responseDto = new ProductDto(1L, "Electronics",
16000000.0, "Laptop Updated", "New description", 15);
100
101         when(productService.updateProduct(eq(1L), any(
UpdateProductRequest.class)))
102             .thenReturn(responseDto);
103
104         // Act & Assert
```

```
105         mockMvc.perform(put("/api/products/1")
106             .contentType(MediaType.APPLICATION_JSON)
107             .content(objectMapper.writeValueAsString(requestDto)
108         ))
109             .andExpect(status().isOk())
110             .andExpect(jsonPath("$.id").value(1))
111             .andExpect(jsonPath("$.productName").value("Laptop
Updated"))
112             .andExpect(jsonPath("$.price").value(16000000.0))
113             .andExpect(jsonPath("$.quantity").value(15));
114
115         verify(productService, times(1)).updateProduct(eq(1L), any(
UpdateProductRequest.class));
116     }
117
118
119     @Test
120     @DisplayName("5. Xóa product thành công - 204 No Content")
121     void testDeleteProduct_Success() throws Exception {
122         // Arrange
123         doNothing().when(productService).deleteProduct(1L);
124
125         // Act & Assert
126         mockMvc.perform(delete("/api/products/1"))
127             .andExpect(status().isNoContent()); // 204 No
Content
128
129         verify(productService, times(1)).deleteProduct(1L);
130     }
```

Listing 16: ProductControllerIntegrationTest.java

5 Câu 4: Mock Testing (10 điểm)

5.1 Câu 4.1: Login - Mock Testing (5 điểm)

5.1.1 Frontend Mocking (2.5 điểm)

Mock external dependencies cho Login component:

a) Mock authService.loginUser() (1 điểm)

```
1 describe('Test 1: Mock authService.login() ', () => {
2
3
4     test('nên mock authService.login() function', () => {
5         // Verify authService.login đọc mock
6         expect(typeof authService.login).toBe('function');
7         expect(jest.isMockFunction(authService.login)).toBe(true);
8     });
9 });
10
```

Listing 17: Mock authService.loginUser()

b) Test với mocked successful/failed responses (1 điểm)

```
1 describe('Test 2: Mocked Successful/Failed Responses (1 điểm)',
2     () => {
3
4     test('nên handle mocked successful response', async () => {
5         const user = userEvent.setup();
6         authService.login.mockResolvedValue({
7             success: true,
8             message: 'Login successful',
9             user: {
10                 id: '123',
11                 username: 'testuser',
12                 name: 'Test User'
13             }
14         });
15
16         render(
17             <MemoryRouter>
18                 <LoginForm />
19             </MemoryRouter>
20         );
21
22         const usernameInput = screen.getByPlaceholderText(/
23             your_username/i);
24         const passwordInput = screen.getByPlaceholderText(/
25             /i);
26         const submitButton = screen.getByRole('button', { name: /Đăng
27             Nhập/i });
28
29         await user.type(usernameInput, 'testuser');
30         await user.type(passwordInput, 'password123');
31         await user.click(submitButton);
32     });
33 }
```

```
28
29 // Verify success message appears
30 await waitFor(() => {
31   const successMessage = screen.queryByText(/Đăng nhập thành
   công/i);
32   expect(successMessage).toBeInTheDocument();
33   }, { timeout: 2000 });
34 });
35
36 test('nên handle mocked failed response - Invalid credentials',
   async () => {
37   const user = userEvent.setup();
38   authService.login.mockRejectedValue(new Error('Invalid
   credentials'));
39
40   render(
41     <MemoryRouter>
42       <LoginForm />
43     </MemoryRouter>
44   );
45
46   const usernameInput = screen.getByPlaceholderText(/
   your_username/i);
47   const passwordInput = screen.getByPlaceholderText(/
   /i);
48   const submitButton = screen.getByRole('button', { name: /Đăng
   Nhập/i });
49
50   await user.type(usernameInput, 'testuser');
51   await user.type(passwordInput, 'wrongpassword1');
52   await user.click(submitButton);
53
54   // Verify error message appears
55   await waitFor(() => {
56     const errorMessage = screen.getByText(/Invalid credentials/
   i);
57     expect(errorMessage).toBeInTheDocument();
58   });
59 });
60 });
61
```

Listing 18: mocked successful/failed responses

c) Verify mock calls (0.5 điểm)

```
1 describe('Test 3: Verify Mock Calls (0.5 điểm)', () => {
2
3   test('nên verify mock đ ọc gọi với correct parameters', async
   () => {
4     const user = userEvent.setup();
5     authService.login.mockResolvedValue({ success: true });
6
7     render(
8       <MemoryRouter>
9         <LoginForm />
```

```
10 </MemoryRouter>
11 );
12
13     const usernameInput = screen.getByPlaceholderText(/
14     your_username/i);
15     const passwordInput = screen.getByPlaceholderText(/
16     /i);
17     const submitButton = screen.getByRole('button', { name: /Đăng
18     Nhập/i });
19
20     await user.type(usernameInput, 'user123');
21     await user.type(passwordInput, 'pass123');
22     await user.click(submitButton);
23
24     await waitFor(() => {
25         // Verify mock đ ọc gọi với đúng parameters
26         expect(authService.login).toHaveBeenCalledWith('user123', '
27         pass123');
28     });
29
30     test('nên verify mock đ ọc gọi exactly 1 time', async () => {
31         const user = userEvent.setup();
32         authService.login.mockResolvedValue({ success: true });
33
34         render(
35             <MemoryRouter>
36             <LoginForm />
37             </MemoryRouter>
38         );
39
40         const usernameInput = screen.getByPlaceholderText(/
41         your_username/i);
42         const passwordInput = screen.getByPlaceholderText(/
43         /i);
44         const submitButton = screen.getByRole('button', { name: /Đăng
45         Nhập/i });
46
47         await user.type(usernameInput, 'testuser');
48         await user.type(passwordInput, 'password123');
49         await user.click(submitButton);
50
51         await waitFor(() => {
52             // Verify called exactly once
53             expect(authService.login).toHaveBeenCalledTimes(1);
54         });
55
56         test('nên verify mock NOT đ ọc gọi khi invalid data', async ()
57         => {
58             const user = userEvent.setup();
59             authService.login.mockResolvedValue({ success: true });
60
61             render(
62                 <MemoryRouter>
63                 <LoginForm />
```

```
58     </MemoryRouter>
59 );
60
61     const submitButton = screen.getByRole('button', { name: /Đăng
        Nhập/i });
62
63     // Submit form trống
64     await user.click(submitButton);
65
66     // Mock không nên đ ọc gọi
67     expect(authService.login).not.toHaveBeenCalled();
68 });
69
70 });
71
```

Listing 19: Verify mock calls

5.1.2 Backend Mocking (2.5 điểm)

Mock dependencies trong Backend tests:

- Mock AuthService với @MockBean (1 điểm)
- Test controller với mocked service (1 điểm)
- Verify mock interactions (0.5 điểm)

```
1 @WebMvcTest(value = AuthController.class,
2     excludeFilters = @ComponentScan.Filter(type = FilterType.
3     ASSIGNABLE_TYPE,
4     classes = {com.flogin.service.SecurityConfig.class, com.flogin.
5     filter.JwtAuthenticationFilter.class}))
6 @AutoConfigureMockMvc(addFilters = false)
7 @DisplayName("Backend Mocking - AuthController Mock Tests")
8 public class AuthControllerMockTest {
9
10     @Autowired
11     private MockMvc mockMvc;
12
13     /**
14      * a) Mock AuthService với @MockitoBean (1 điểm)
15      * @MockitoBean tạo mock object và inject vào Spring context
16      */
17     @MockitoBean
18     private AuthService authService;
19
20     @Autowired
21     ObjectMapper objectMapper;
22
23     @Test
24     @DisplayName("1. Mock: Controller với mocked service - Login thà
25     nh công")
26     void testLoginWithMockedService_Success() throws Exception {
```

```
24         //Arrange
25         LoginRequest request = new LoginRequest("testUser", "
26         test12345");
27         UserDto userDto = new UserDto("testUser", "testUser@gmail.
28         com");
29         LoginResponse response = new LoginResponse(true, "Login thà
30         nh công", "jwt-fake-token", userDto);
31
32         when(authService.authenticate(any(LoginRequest.class)))
33             .thenReturn(response);
34
35         //Act & Assert
36         mockMvc.perform(post("/api/auth/login")
37             .contentType(MediaType.APPLICATION_JSON)
38             .content(objectMapper.writeValueAsString(request)))
39             .andExpect(status().isOk())
40             .andExpect(jsonPath("$.token").exists())
41             .andExpect(jsonPath("$.user").exists())
42             .andExpect(jsonPath("$.user.userName").value("
43         testUser")));
44
45         // Verify
46         verify(authService, times(1)).authenticate(any(LoginRequest.
47         class));
48     }
49
50     @Test
51     @DisplayName("2. Mock: Login thất bại - Username không tồn tại")
52     void testLoginWithMockedService_UserNotFound() throws Exception
53     {
54         // Arrange: Mock service trả về failure response
55         LoginResponse mockResponse = new LoginResponse(
56             false,
57             "Login thất bại với user name không tồn tại"
58         );
59
60         when(authService.authenticate(any(LoginRequest.class)))
61             .thenReturn(mockResponse);
62
63         // Act & Assert
64         mockMvc.perform(post("/api/auth/login")
65             .contentType(MediaType.APPLICATION_JSON)
66             .content("{\"userName\":\"nonexistent\",\"password
67         \":\"Pass123\"}"))
68             .andExpect(status().isUnauthorized())
69             .andExpect(jsonPath("$.success").value(false))
70             .andExpect(jsonPath("$.message").value("Login thất b
71         ại với user name không tồn tại"))
72             .andExpect(jsonPath("$.token").doesNotExist())
73             .andExpect(jsonPath("$.user").doesNotExist());
74
75         // Verify interactions
76         verify(authService, times(1)).authenticate(any(LoginRequest.
77         class));
```



```
71     }
72
73     @Test
74     @DisplayName("3. Mock: Login thất bại - Password sai")
75     void testLoginWithMockedService_WrongPassword() throws Exception
76     {
77         // Arrange: Mock service trả về password sai
78         LoginResponse mockResponse = new LoginResponse(false, "Login
79         với password sai");
80
81         when(authService.authenticate(any(LoginRequest.class)))
82             .thenReturn(mockResponse);
83
84         // Act & Assert
85         mockMvc.perform(post("/api/auth/login")
86             .contentType(MediaType.APPLICATION_JSON)
87             .content("{\"userName\":\"testuser\",\"password\":\"
88         WrongPass123\"}"))
89             .andExpect(status().isUnauthorized())
90             .andExpect(jsonPath("$.success").value(false))
91             .andExpect(jsonPath("$.message").value("Login với
92         password sai"));
93
94         // Verify interactions
95         verify(authService, times(1)).authenticate(any(LoginRequest.
96         class));
97     }
98
99     /**
100     * C) Verify Mock Interactions
101     */
102     @Nested
103     @DisplayName("C) Verify Mock Interactions - Chi tiết ")
104     class VerifyMockInteractionsTests {
105
106         @Test
107         @DisplayName("4. Verify: Mock đ ọc gọi đúng 1 lần với times(1)")
108         void testVerify_MockCalledExactlyOnce() throws Exception {
109             // Arrange
110             UserDto userDto = new UserDto("testuser", "test@example.com"
111         );
112             LoginResponse mockResponse = new LoginResponse(true, "
113         Success", "token", userDto);
114
115             when(authService.authenticate(any(LoginRequest.class)))
116                 .thenReturn(mockResponse);
117
118             // Act
119             mockMvc.perform(post("/api/auth/login")
120                 .contentType(MediaType.APPLICATION_JSON)
121                 .content("{\"userName\":\"testuser\",\"password\":\"
122         Pass123\"}"))
123                 .andExpect(status().isOk());
124
125             // c) Verify: Kiểm tra mock đ ọc gọi đúng 1 lần
```

```
119         verify(authService, times(1)).authenticate(any(LoginRequest.  
120             class));  
121         // Verify không có interaction nào khác  
122         verifyNoMoreInteractions(authService);  
123     }  
124  
125     @Test  
126     @DisplayName("5. Verify: Mock không đọc gọi khi validation fail  
127     ")  
128     void testVerify_MockNotCalledWhenValidationFails() throws  
129     Exception {  
130         // Arrange: Không cần mock vì validation sẽ fail tr ớc khi g  
131         // Act: Gửi request với username trống (validation fail)  
132         mockMvc.perform(post("/api/auth/login")  
133             .contentType(MediaType.APPLICATION_JSON)  
134             .content("{\"userName\":\"\", \"password\":\"Pass123  
135             \"}"))  
136             .andExpect(status().isBadRequest());  
137         // c) Verify: authService.authenticate() KHÔNG đọc gọi vì  
138         // validation fail  
139         verify(authService, never()).authenticate(any(LoginRequest.  
140             class));  
141         verifyNoInteractions(authService);  
142     }  
143 }
```

Listing 20: AuthControllerMockTest.java

5.2 Câu 4.2: Product - Mock Testing (5 điểm)

5.2.1 Frontend Mocking (2.5 điểm)

Mock ProductService trong component tests:

- Mock CRUD operations (1.5 điểm)
- Test success và failure scenarios (0.5 điểm)
- Verify all mock calls (0.5 điểm)

```
1 jest.mock('../services/productService');  
2  
3 describe('Product Mock Tests', () => {  
4  
5     // Mock Data  
6     const mockProductList = [  
7         {  
8             id: 1,
```

```
9      name: 'Laptop Dell',
10     productName: 'Laptop Dell', // Thêm tr òng này cho chắc chắn
11     title: 'Laptop Dell',
12     price: 15000000,
13     quantity: 10,
14     category: 'Electronics',
15     description: 'Mô tá sán phẩm mẫu'
16   }
17 ];
18
19 beforeEach(() => {
20   jest.clearAllMocks();
21   productService.getAllProducts.mockResolvedValue(mockProductList
22 );
23 });
24
25 // Helper function: Sửa lại để xử lý tr òng hợp có nhiều nút cùng
26 // tên
27 const fillAndSubmitForm = async (user) => {
28   // 1. Mở Modal
29   await user.click(screen.getByText(/Thêm Sản Phẩm/i));
30
31   // 2. Điền Form
32   await user.type(screen.getByLabelText(/tên sản phẩm/i), 'New
33 Laptop');
34   await user.type(screen.getByLabelText(/giá/i), '20000000');
35   await user.type(screen.getByLabelText(/số l ượng/i), '5');
36   await user.selectOptions(screen.getByLabelText(/danh mục/i), '
37 Electronics');
38   await user.type(screen.getByLabelText(/mô tá/i), 'Mô tá test');
39
40   // 3. Submit
41   // Lấy tất cả các nút có tên L u/Thêm/Cập Nhật
42   const submitBtns = screen.getAllByRole('button', { name: /L u|
43 Thêm|Cập Nhật/i });
44   // Chọn nút cuối cùng (th òng là nút nằm trong Modal vừa mở ra)
45   const submitBtn = submitBtns[submitBtns.length - 1];
46   await user.click(submitBtn);
47 };
48
49 test('Mock: Get products (READ)', async () => {
50   productService.getAllProducts.mockResolvedValue(mockProductList
51 );
52   render(<ProductManagement />);
53
54   await waitFor(() => {
55     expect(screen.getByText('Laptop Dell')).toBeInTheDocument();
56     expect(productService.getAllProducts).toHaveBeenCalledTimes
57 (1);
58   });
59 });
60
61 test('Mock: Create product thành công', async () => {
62   const user = userEvent.setup();
63   const newProductResponse = { id: 2, name: 'New Laptop', price:
```

```
20000000 };  
57  
58   productService.createProduct.mockResolvedValue(  
newProductResponse);  
59  
60   render(<ProductManagement />);  
61   await fillAndSubmitForm(user);  
62  
63   await waitFor(() => {  
64     expect(productService.createProduct).toHaveBeenCalledWith(  
expect.objectContaining({  
65       productName: 'New Laptop',  
66       price: '20000000'  
67     }));  
68  
69     expect(screen.getByText(/thành công/i)).toBeInTheDocument();  
70   });  
71 });  
72  
73 test('Mock: Create product that bai (Failure Scenario)', async ()  
=> {  
74   const user = userEvent.setup();  
75   productService.createProduct.mockRejectedValue(new Error(''  
Server Error''));  
76  
77   render(<ProductManagement />);  
78   await fillAndSubmitForm(user);  
79  
80   await waitFor(() => {  
81     expect(productService.createProduct).toHaveBeenCalledTimes(1)  
82     ;  
83     expect(screen.getByText(/Server Error/i)).toBeInTheDocument()  
84     ;  
85   });  
86 });  
87  
88 test('Mock: Delete product', async () => {  
89   const user = userEvent.setup();  
90   productService.deleteProduct.mockResolvedValue({ success: true  
91   });  
92  
93   render(<ProductManagement />);  
94   await waitFor(() => screen.getByText('Laptop Dell'));  
95  
96   const deleteBtns = screen.getAllByTitle(/Xóa/i);  
97   await user.click(deleteBtns[0]);  
98  
99   const confirmBtn = await screen.findByText('Xóa', { selector: '  
button' });  
100   await user.click(confirmBtn);  
101  
102   await waitFor(() => {  
103     expect(productService.deleteProduct).toHaveBeenCalledWith(1);  
104   });  
105 });  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197
```

```
104 test('Mock: Update product', async () => {
105     const user = userEvent.setup();
106     productService.updateProduct.mockResolvedValue({ success: true
107     });
108
109     render(<ProductManagement />);
110     await waitFor(() => screen.getByText('Laptop Dell'));
111
112     const editBtns = screen.getAllByTitle(/Chính sửa/i);
113     await user.click(editBtns[0]);
114
115     // 2. Sửa giá
116     const priceInput = screen.getByLabelText(/giá/i);
117     await user.clear(priceInput);
118     await user.type(priceInput, '18000000');
119
120     const submitBtns = screen.getAllByRole('button', { name: /L u|C
121     ập nhật/i });
122     await user.click(submitBtns[submitBtns.length - 1]);
123
124     await waitFor(() => {
125         expect(productService.updateProduct).toHaveBeenCalledWith(1,
126         expect.objectContaining({
127             price: '18000000'
128         }));
129     });
130 });
```

Listing 21: validateUsername

5.2.2 Backend Mocking (2.5 điểm)

Mock ProductRepository trong service tests:

- Mock ProductRepository (1 điểm)
- Test service layer với mocked repository (1 điểm)
- Verify repository interactions (0.5 điểm)

```
1 @ExtendWith(MockitoExtension.class)
2 @DisplayName("Mock ProductRepository trong Service Tests (2.5 điểm)")
3 public class ProductServiceMockTest {
4
5     /**
6      * a) Mock ProductRepository (1 điểm)
7      */
8     @Mock
9     private ProductRepository productRepository;
10
11     @Mock
12     private Validator validator;
```

```
13
14 @InjectMocks
15 private ProductService productService;
16
17 private Product mockProduct;
18 private CreateProductRequest mockCreateRequest;
19 private UpdateProductRequest mockUpdateRequest;
20
21 @BeforeEach
22 void setUp() {
23     mockProduct = new Product(1L, "Electronics", "Gaming laptop",
24     10, "Laptop", 15000000.0);
25     mockCreateRequest = new CreateProductRequest("Laptop",
26     15000000.0, "Gaming laptop", 10, "Electronics");
27     mockUpdateRequest = new UpdateProductRequest("Laptop",
28     15000000.0, "Gaming laptop", 10, "Electronics");
29 }
30
31 /**
32  * A) Mock ProductRepository
33  * Test các methods với mocked repository
34  */
35 @Nested
36 @DisplayName("A) Mock ProductRepository (1 điểm)")
37 class MockRepositoryTests {
38
39     @Test
40     @DisplayName("1. Mock findById - Tìm product thành công")
41     void testGetProductById_Success() {
42         // Arrange: Mock repository trả về product
43         when(productRepository.findById(1L))
44             .thenReturn(Optional.of(mockProduct));
45
46         // Act
47         ProductDto result = productService.getProductById(1L);
48
49         // Assert
50         assertNotNull(result);
51         assertEquals("Laptop", result.getProductName());
52         assertEquals(15000000.0, result.getPrice());
53         assertEquals("Electronics", result.getCategory());
54
55         // c) Verify repository interaction
56         verify(productRepository, times(1)).findById(1L);
57     }
58
59     @Test
60     @DisplayName("2. Mock save - Tạo product mới")
61     void testCreateProduct_Success() {
62         // Arrange
63         when(validator.validate(any(CreateProductRequest.class)))
64             .thenReturn(Set.of());
65         when(productRepository.save(any(Product.class)))
66             .thenReturn(mockProduct);
67     }
68 }
```

```
66         // Act
67         ProductDto result = productService.createProduct(
mockCreateRequest);
68
69         // Assert
70         assertNotNull(result);
71         assertEquals("Laptop", result.getProductName());
72
73         // c) Verify interactions
74         verify(validator, times(1)).validate(any(
CreateProductRequest.class));
75         verify(productRepository, times(1)).save(any(Product.class))
;
76     }
77 }
78
79 /**
80  * B) Test Service Layer với Mocked Repository
81  */
82 @Nested
83 @DisplayName("B) Test Service Layer với Mocked Repository (1 điểm)")
84 class ServiceLayerTests {
85
86     @Test
87     @DisplayName("3. Service update product với mock repository")
88     void testUpdateProduct_Success() {
89         // Arrange
90         when(validator.validate(any(UpdateProductRequest.class)))
91             .thenReturn(Set.of());
92         when(productRepository.findById(1L))
93             .thenReturn(Optional.of(mockProduct));
94         when(productRepository.save(any(Product.class)))
95             .thenReturn(mockProduct);
96
97         // Act
98         ProductDto result = productService.updateProduct(1L,
mockUpdateRequest);
99
100        // Assert
101        assertNotNull(result);
102        assertEquals(1L, result.getId());
103
104        // c) Verify repository gọi đúng thứ tự
105        verify(productRepository, times(1)).findById(1L);
106        verify(productRepository, times(1)).save(any(Product.class))
;
107    }
108
109     @Test
110     @DisplayName("4. Service delete product với mock repository")
111     void testDeleteProduct_Success() {
112         // Arrange
113         when(productRepository.findById(1L))
114             .thenReturn(Optional.of(mockProduct));
115         doNothing().when(productRepository).deleteById(1L);
116    }
```

```
117         // Act
118         productService.deleteProduct(1L);
119
120         // c) Verify repository interactions
121         verify(productRepository, times(1)).findById(1L);
122         verify(productRepository, times(1)).deleteById(1L);
123     }
124
125
126 }
127
128 /**
129  * C) Verify Repository Interactions
130  */
131 @Nested
132 @DisplayName("C) Verify Repository Interactions (0.5 điểm)")
133 class VerifyRepositoryInteractionsTests {
134
135     @Test
136     @DisplayName("5. Verify repository đọc gọi với argument cụ thể")
137     void testVerify_RepositoryCalledWithSpecificArgument() {
138         // Arrange
139         when(productRepository.findById(1L))
140             .thenReturn(Optional.of(mockProduct));
141
142         // Act
143         productService.getProductById(1L);
144
145         // c) Verify với argument cụ thể
146         verify(productRepository).findById(eq(1L));
147         verify(productRepository, times(1)).findById(1L);
148         verifyNoMoreInteractions(productRepository);
149     }
150
151     @Test
152     @DisplayName("6. Verify repository không đọc gọi khi validation fail")
153     void testVerify_RepositoryNotCalledWhenValidationFails() {
154         // Arrange: Mock validator trả về violation
155         var violation = mock(jakarta.validation.ConstraintViolation.
156         class);
157         when(violation.getMessage()).thenReturn("Price phải > 0");
158         when validator.validate(any(CreateProductRequest.class))
159             .thenReturn(Set.of(violation));
160
161         // Act & Assert
162         assertThrows(IllegalArgumentException.class, () -> {
163             productService.createProduct(mockCreateRequest);
164         });
165
166         // c) Verify repository.save() KHÔNG đọc gọi
167         verify(productRepository, never()).save(any(Product.class));
168         verifyNoInteractions(productRepository);
169     }
170 }
```




169
170
171

```
}  
}  
}
```

Listing 22: ProductServiceMockTest.java

6 Câu 5: Automation Testing và CI/CD (10 điểm)

6.1 Câu 5.1: Login - E2E Automation Testing (5 điểm)

6.1.1 Setup và Configuration (1 điểm)

- Cài đặt Cypress hoặc Selenium
- Cấu hình test environment

```
1 /**
2  * Cypress Configuration File
3  * @see https://docs.cypress.io/guides/references/configuration
4  */
5  const { defineConfig } = require('cypress');
6
7  module.exports = defineConfig({
8    /**
9     * E2E Testing Configuration
10    *
11    * Configures end-to-end tests that run against the full
12    application
13    * in a real browser environment.
14    */
15    e2e: {
16      /**
17       * Base URL for all cy.visit() and cy.request() commands
18       * Points to local development server
19       * Change to production URL for production testing
20       */
21      baseUrl: 'http://localhost:5173',
22
23      /**
24       * Default viewport dimensions for test browser
25       * Simulates desktop screen resolution
26       */
27      viewportWidth: 1280,
28      viewportHeight: 720,
29
30      /**
31       * Time to wait for page to load before failing
32       * Default: 60000ms (60 seconds)
33       */
34      pageLoadTimeout: 30000,
35
36      /**
37       * Time to wait for cy.request() to resolve
38       * Used for API calls in tests
39       */
40      requestTimeout: 15000,
41
42      /**
43       * Time to wait for response in cy.wait()
44       * Used for network request assertions
45       */
46      responseTimeout: 15000,
```

```
46
47  /**
48   * Default timeout for most Cypress commands
49   * Applies to cy.get(), cy.contains(), etc.
50   */
51  defaultCommandTimeout: 10000,
52
53  /**
54   * Disable video recording for all tests
55   * Enable for debugging: set to true
56   * Videos saved to: cypress/videos/
57   */
58  video: false,
59
60  /**
61   * Take screenshot when test fails
62   * Screenshots saved to: cypress/screenshots/
63   * Very useful for debugging failures in CI/CD
64   */
65  screenshotOnRunFailure: true,
66
67  /**
68   * Record video only when test fails
69   * Disabled here since video is disabled
70   */
71  videoOnFailure: false,
72
73  /**
74   * Pattern to match E2E test files
75   * Looks for *.cy.js files in e2e directory
76   * Example: login.cy.js, product-management.cy.js
77   */
78  specPattern: 'src/tests/cypress/e2e/**/*.cy.js',
79
80  /**
81   * Support file with custom commands and global config
82   * Loaded before every test file
83   * Contains custom commands like cy.login()
84   */
85  supportFile: 'src/tests/cypress/support/e2e.js',
86
87  /**
88   * Folder containing test data fixtures
89   * Access via: cy.fixture('users.json')
90   * Used for test data like users, products, etc.
91   */
92  fixturesFolder: 'src/tests/cypress/fixtures',
93
94  /**
95   * Disable Chrome web security
96   * Allows cross-origin requests in tests
97   * Useful for testing external APIs or CORS issues
98   */
99  chromeWebSecurity: false,
100
101  /**
```

```
102     * Retry failed tests to reduce flakiness
103     * - runMode: Retries in headless mode (CI/CD)
104     * - openMode: Retries in interactive mode (development)
105     */
106     retries: {
107         runMode: 1,      // Retry once in CI/CD pipeline
108         openMode: 0      // No retries in interactive mode
109     }
110 },
111
112 /**
113  * Component Testing Configuration (Optional)
114  *
115  * Tests individual React components in isolation
116  * using Vite dev server for fast feedback.
117  */
118 component: {
119     /**
120     * Dev server configuration for component testing
121     * Uses Vite for fast component mounting and HMR
122     */
123     devServer: {
124         framework: 'react', // Framework being tested
125         bundler: 'vite',    // Build tool (Vite for fast builds)
126     },
127
128     /**
129     * Pattern to match component test files
130     * Example: Button.cy.jsx, Form.cy.jsx
131     */
132     specPattern: 'src/tests/cypress/component/**/*.cy.jsx',
133
134     /**
135     * Support file for component tests
136     * Contains component-specific setup and commands
137     */
138     supportFile: 'src/tests/cypress/support/component.js'
139 }
140 });
141
142
```

Listing 23: test environment

- Setup Page Object Model

```
1         class LoginPage {
2             // All element getters return Cypress chainable elements
3
4             /**
5              * Get username input field
6              * @returns {Cypress.Chainable} Username input element
7              */
8             get usernameInput() {
9                 return cy.get('input[type="text"]').first();
10            }
11        }
```

```
11
12 get passwordInput() {
13     return cy.get('input[type="password"]');
14 }
15
16 get loginButton() {
17     return cy.get("button").contains(/login|đăng nhập/i);
18 }
19
20 get rememberMeCheckbox() {
21     return cy.get('input[type="checkbox"]');
22 }
23
24 get forgotPasswordLink() {
25     return cy.get("a").contains(/quên|forgot/i);
26 }
27
28 get emailInput() {
29     return cy.get('input[type="text"]').first();
30 }
31
32 get signupLink() {
33     return cy.get("a").contains(/đăng ký|sign up|register/i);
34 }
35
36 get errorMessage() {
37     return cy.get('[style*="color: rgb(220, 38, 38)"]').first();
38 }
39
40 get successMessage() {
41     return cy.get('[style*="background: rgb(34, 197, 94)"]').first();
42 }
43
44 get loadingSpinner() {
45     return cy.get('[class*="spinner"], [class*="loading"]');
46 }
47
48 get pageTitle() {
49     return cy.contains("h1", /login|đăng nhập/i);
50 }
51
52 get passwordVisibilityToggle() {
53     return cy.get('button[type="button"]').contains(" í ");
54 }
55
56 // Methods
57 /**
58  * Navigate to login page
59  */
60 navigateToLoginPage() {
61     cy.visit("/");
62     this.pageTitle.should("be.visible");
63     return this;
64 }
65
```

```
66  /**
67   * Enter username
68   */
69  enterUsername(username) {
70      this.usernameInput.clear().type(username, { delay: 100 });
71      return this;
72  }
73
74  /**
75   * Enter password
76   */
77  enterPassword(password) {
78      this.passwordInput.clear().type(password, { delay: 100 });
79      return this;
80  }
81
82  /**
83   * Click login button
84   */
85  clickLogin() {
86      this.loginButton.click();
87      return this;
88  }
89
90  /**
91   * Complete login flow
92   */
93  login(username, password) {
94      this.enterUsername(username);
95      this.enterPassword(password);
96      this.clickLogin();
97      return this;
98  }
99
100 /**
101  * Check remember me checkbox
102  */
103 checkRememberMe() {
104     this.rememberMeCheckbox.check();
105     return this;
106 }
107
108 /**
109  * Uncheck remember me checkbox
110  */
111 uncheckRememberMe() {
112     this.rememberMeCheckbox.uncheck();
113     return this;
114 }
115
116 /**
117  * Click forgot password link
118  */
119 clickForgotPassword() {
120     this.forgotPasswordLink.click();
121     return this;
```

```
122 }
123
124 /**
125  * Click signup link
126  */
127 clickSignup() {
128     this.signupLink.click();
129     return this;
130 }
131
132 /**
133  * Toggle password visibility
134  */
135 togglePasswordVisibility() {
136     this.passwordVisibilityToggle.click();
137     return this;
138 }
139
140 /**
141  * Verify error message exists
142  */
143 verifyErrorMessage() {
144     this.errorMessage.should("be.visible");
145     return this;
146 }
147
148 /**
149  * Verify error message with specific text
150  */
151 verifyErrorMessageText(text) {
152     this.errorMessage.should("contain", text);
153     return this;
154 }
155
156 /**
157  * Verify success message exists
158  */
159 verifySuccessMessage() {
160     this.successMessage.should("be.visible");
161     return this;
162 }
163
164 /**
165  * Verify login button is enabled
166  */
167 verifyLoginButtonEnabled() {
168     this.loginButton.should("not.be.disabled");
169     return this;
170 }
171
172 /**
173  * Verify login button is disabled
174  */
175 verifyLoginButtonDisabled() {
176     this.loginButton.should("be.disabled");
177     return this;
```

```
178 }
179
180 /**
181  * Verify loading spinner is visible
182  */
183 verifyLoadingSpinner() {
184     this.loadingSpinner.should("be.visible");
185     return this;
186 }
187
188 /**
189  * Verify username input has value
190  */
191 verifyUsernameValue(value) {
192     this.usernameInput.should("have.value", value);
193     return this;
194 }
195
196 /**
197  * Verify password input is empty
198  */
199 verifyPasswordEmpty() {
200     this.passwordInput.should("have.value", "");
201     return this;
202 }
203
204 /**
205  * Verify page title is visible
206  */
207 verifyPageTitle() {
208     this.pageTitle.should("be.visible");
209     return this;
210 }
211
212 /**
213  * Clear username input
214  */
215 clearUsername() {
216     this.usernameInput.clear();
217     return this;
218 }
219
220 /**
221  * Clear password input
222  */
223 clearPassword() {
224     this.passwordInput.clear();
225     return this;
226 }
227
228 /**
229  * Clear all inputs
230  */
231 clearAllInputs() {
232     this.clearUsername();
233     this.clearPassword();
```



```
234     return this;
235 }
236
237 /**
238  * Get username input value
239  */
240 getUsernameValue() {
241     return this.usernameInput.invoke("val");
242 }
243
244 /**
245  * Get password input type
246  */
247 getPasswordInputType() {
248     return this.passwordInput.invoke("attr", "type");
249 }
250
251 /**
252  * Verify username input is visible
253  */
254 verifyUsernameInputVisible() {
255     this.usernameInput.should("be.visible");
256     return this;
257 }
258
259 /**
260  * Verify password input is visible
261  */
262 verifyPasswordInputVisible() {
263     this.passwordInput.should("be.visible");
264     return this;
265 }
266
267 /**
268  * Verify all form elements are visible
269  */
270 verifyFormElementsVisible() {
271     this.verifyPageTitle();
272     this.usernameInput.should("be.visible");
273     this.passwordInput.should("be.visible");
274     this.loginButton.should("be.visible");
275     return this;
276 }
277
278 /**
279  * Verify username input is empty
280  */
281 verifyUsernameEmpty() {
282     this.usernameInput.should("have.value", "");
283     return this;
284 }
285
286 /**
287  * Verify password input is empty
288  */
289 verifyPasswordEmpty() {
```

```
290     this.passwordInput.should("have.value", "");
291     return this;
292 }
293
294 /**
295  * Wait for page to load
296  */
297 waitForPageLoad() {
298     cy.wait(1000);
299     this.pageTitle.should("be.visible");
300     return this;
301 }
302
303 /**
304  * Wait for loading to complete
305  */
306 waitForLoadingComplete() {
307     this.loadingSpinner.should("not.exist");
308     return this;
309 }
310 }
311
312 // Export page object
313 export default new LoginPage();
314
315
```

Listing 24: Setup Page Object Model

6.1.2 E2E Test Scenarios cho Login (2.5 điểm)

- a) Test complete login flow (1 điểm)
- b) Test validation messages (0.5 điểm)
- c) Test success/error flows (0.5 điểm)
- d) Test UI elements interactions (0.5 điểm)

```
1     import LoginPage from "../pages/LoginPage.js";
2
3 describe("E2E Test Scenarios - Complete Login Flow", () => {
4     beforeEach(() => {
5         cy.fixture("users").as("users");
6         LoginPage.navigateToLoginPage();
7     });
8
9     /**
10    * SCENARIO 1: Complete Login Flow
11    */
12    describe("Scenario 1: Complete Login Flow (1 điểm)", () => {
13        it("SC1.1: Ng ời dùng có thể m ỏ trang login", () => {
14            LoginPage.verifyPageTitle();
15            LoginPage.verifyFormElementsVisible();
16        });
17    });
18 }
```

```
17
18   it("SC1.2: Ng ời dùng có thể nhập username hợp lệ", () => {
19     const username = "test";
20     LoginPage.enterUsername(username);
21     LoginPage.verifyUsernameValue(username);
22   });
23
24   it("SC1.3: Ng ời dùng có thể nhập password hợp lệ", () => {
25     const password = "Test123@";
26     LoginPage.enterPassword(password);
27     LoginPage.passwordInput.should("have.value", password);
28   });
29
30   it("SC1.4: Ng ời dùng có thể submit form với data hợp lệ", ()
=> {
31     LoginPage.login("test", "Test123@");
32     // Form submitted - có thể verify bằng network call
33   });
34
35   it("SC1.5: Login button enabled khi username & password filled
", () => {
36     LoginPage.enterUsername("test")
37       .enterPassword("Test123@")
38       .verifyLoginButtonEnabled();
39   });
40
41   it("SC1.6: Form hiển thị loading state sau khi submit", () => {
42     LoginPage.enterUsername("test");
43     LoginPage.enterPassword("Test123@");
44     LoginPage.clickLogin();
45     // Có thể verify button disabled hoặc spinner
46     cy.wait(500);
47   });
48
49   it("SC1.7: Ng ời dùng có thể clear username và nhập lại", () =>
{
50     LoginPage.enterUsername("test");
51     LoginPage.clearUsername();
52     LoginPage.usernameInput.should("have.value", "");
53     LoginPage.enterUsername("new");
54     LoginPage.verifyUsernameValue("new");
55   });
56
57   it("SC1.8: Ng ời dùng có thể clear password và nhập lại", () =>
{
58     LoginPage.enterPassword("Test123@");
59     LoginPage.clearPassword();
60     LoginPage.passwordInput.should("have.value", "");
61     LoginPage.enterPassword("NewPass456@");
62     LoginPage.passwordInput.should("have.value", "NewPass456@");
63   });
64
65   it("SC1.9: Complete flow: Input -> Verify -> Clear -> Resubmit
", () => {
66     // First attempt
67     LoginPage.enterUsername("user1").enterPassword("Pass123@").
```

```
clickLogin();
68
69     cy.wait(500);
70
71     // Verify login button still visible (for retry)
72     LoginPage.loginButton.should("be.visible");
73 });
74
75 it("SC1.10: Ng ời dùng có thể interact với form multiple times
", () => {
76     // Attempt 1
77     LoginPage.enterUsername("attempt1").enterPassword("Pass1@");
78     LoginPage.clearAllInputs();
79
80     // Attempt 2
81     LoginPage.enterUsername("attempt2")
82         .enterPassword("Pass2@")
83         .verifyFormElementsVisible();
84 });
85 });
86
87 /**
88  * SCENARIO 2: Validation Messages
89  */
90 describe("Scenario 2: Validation Messages (0.5  điểm)", () => {
91     it("SC2.1: Validation khi submit form trống", () => {
92         LoginPage.clickLogin();
93         cy.wait(300);
94         // Verify validation message hoặc button disabled
95         LoginPage.loginButton.should("be.visible");
96     });
97
98     it("SC2.2: Validation khi username trống, password filled", ()
=> {
99         LoginPage.enterPassword("Test123@");
100         LoginPage.clearUsername();
101         LoginPage.clickLogin();
102         cy.wait(300);
103         LoginPage.usernameInput.should("have.value", "");
104     });
105
106     it("SC2.3: Validation khi username filled, password trống", ()
=> {
107         LoginPage.enterUsername("test");
108         LoginPage.clearPassword();
109         LoginPage.clickLogin();
110         cy.wait(300);
111         LoginPage.passwordInput.should("have.value", "");
112     });
113
114     it("SC2.4: Validation với username format không hợp lệ", () =>
{
115         LoginPage.enterUsername("invalid-username");
116         LoginPage.enterPassword("Test123@");
117         LoginPage.clickLogin();
118         cy.wait(300);
```

```
119 // username input should still have invalid value
120 LoginPage.usernameInput.should("have.value", "invalid-
username");
121 });
122
123 it("SC2.5: Validation với password yếu", () => {
124     LoginPage.enterUsername("test");
125     LoginPage.enterPassword("123");
126     LoginPage.clickLogin();
127     cy.wait(300);
128     // Short password validation
129 });
130
131 it("SC2.6: Validation messages appear immediately", () => {
132     LoginPage.clickLogin();
133     cy.wait(200);
134     // Check for validation feedback
135 });
136
137 it("SC2.7: Can correct validation errors and resubmit", () => {
138     // Submit with empty form
139     LoginPage.clickLogin();
140     cy.wait(300);
141
142     // Now fill and submit
143     LoginPage.enterUsername("test").enterPassword("Test123@").
clickLogin();
144 });
145
146 it("SC2.8: Multiple validation errors show correctly", () => {
147     LoginPage.clearAllInputs();
148     LoginPage.clickLogin();
149     cy.wait(300);
150     // Both fields empty - verify validation
151 });
152 });
153
154 /**
155  * SCENARIO 3: Success/Error Flows
156  */
157 describe("Scenario 3: Success/Error Flows (0.5 điểm)", () => {
158     it("SC3.1: Successful login dengan valid credentials", () => {
159         cy.fixture("users").then((users) => {
160             LoginPage.login(users.validUser.username, users.validUser.
password);
161             cy.wait(500);
162         });
163     });
164
165     it("SC3.2: Failed login dengan invalid credentials", () => {
166         cy.fixture("users").then((users) => {
167             LoginPage.login(users.invalidUser.username, users.
invalidUser.password);
168             cy.wait(500);
169         });
170     });
171 });
```

```
171
172     it("SC3.3: Error message visible on failed login", () => {
173         cy.fixture("users").then((users) => {
174             LoginPage.login(users.invalidUser.username, users.
invalidUser.password);
175             cy.wait(500);
176             // Error message may appear
177         });
178     });
179
180     it("SC3.4: Success message visible on successful login", () =>
{
181         cy.fixture("users").then((users) => {
182             LoginPage.login(users.validUser.username, users.validUser.
password);
183             cy.wait(500);
184             // Success message may appear
185         });
186     });
187
188     it("SC3.5: User can retry after failed login", () => {
189         cy.fixture("users").then((users) => {
190             // First attempt - fail
191             LoginPage.login(users.invalidUser.username, users.
invalidUser.password);
192             cy.wait(500);
193
194             // Clear and retry
195             LoginPage.clearAllInputs();
196             LoginPage.login(users.validUser.username, users.validUser.
password);
197         });
198     });
199
200     it("SC3.6: Error handling for network issues", () => {
201         LoginPage.enterUsername("test").enterPassword("Test123@").
clickLogin();
202         cy.wait(300);
203         // Network error handling
204     });
205
206     it("SC3.7: Timeout handling during login", () => {
207         LoginPage.login("test", "Test123@");
208         cy.wait(1000);
209         // Verify form still responsive
210         LoginPage.loginButton.should("be.visible");
211     });
212
213     it("SC3.8: Recovery from error state", () => {
214         // Attempt that might fail
215         LoginPage.login("invalid@test.com", "invalid");
216         cy.wait(500);
217
218         // Verify can interact again
219         LoginPage.loginButton.should("be.visible");
220         LoginPage.clearAllInputs();
```

```
221     LoginPage.verifyUsernameEmpty().verifyPasswordEmpty();
222   });
223 });
224
225 /**
226  * SCENARIO 4: UI Elements Interactions
227  */
228 describe("Scenario 4: UI Elements Interactions (0.5 điểm)", () =>
229 {
230   it("SC4.1: Remember me checkbox can be checked", () => {
231     LoginPage.checkRememberMe();
232     LoginPage.rememberMeCheckbox.should("be.checked");
233   });
234
235   it("SC4.2: Remember me checkbox can be unchecked", () => {
236     LoginPage.checkRememberMe();
237     LoginPage.uncheckRememberMe();
238     LoginPage.rememberMeCheckbox.should("not.be.checked");
239   });
240
241   it("SC4.3: Forgot password link is clickable", () => {
242     LoginPage.forgotPasswordLink.should("be.visible");
243     LoginPage.forgotPasswordLink.should("have.attr", "href");
244   });
245
246   it("SC4.4: Sign up link is clickable", () => {
247     LoginPage.signupLink.should("be.visible");
248     LoginPage.signupLink.should("have.attr", "href");
249   });
250
251   it("SC4.5: Password visibility toggle works", () => {
252     LoginPage.enterPassword("Test123@");
253
254     // Get initial type
255     LoginPage.getPasswordInputType().then((type) => {
256       expect(type).to.equal("password");
257     });
258
259     // Toggle visibility
260     LoginPage.togglePasswordVisibility();
261
262     // After toggle, should be visible (type="text")
263     cy.wait(200);
264   });
265
266   it("SC4.6: All form labels are visible", () => {
267     cy.contains("label", /username|username/i).should("be.visible");
268     cy.contains("label", /password|mật khẩu>Password/i).should("be.visible");
269   });
270
271   it("SC4.7: Form placeholders are visible", () => {
272     LoginPage.usernameInput.should("have.attr", "placeholder");
273     LoginPage.passwordInput.should("have.attr", "placeholder");
274   });
275 }
```

```
274
275   it("SC4.8: Login button has proper styling", () => {
276     LoginPage.loginButton.should("be.visible");
277     LoginPage.loginButton.should("not.be.disabled");
278   });
279
280   it("SC4.9: Page title and branding visible", () => {
281     LoginPage.pageTitle.should("be.visible");
282     LoginPage.pageTitle.invoke("text").then((text) => {
283       expect(text).to.match(/login|đăng nhập/i);
284     });
285   });
286
287   it("SC4.10: Form is responsive to interactions", () => {
288     // Click username
289     LoginPage.usernameInput.click();
290     LoginPage.usernameInput.should("be.focused");
291
292     // Type in username
293     LoginPage.usernameInput.type("test");
294     LoginPage.usernameInput.should("have.value", "test");
295
296     // Click password
297     LoginPage.passwordInput.click();
298     LoginPage.passwordInput.should("be.focused");
299
300     // Type in password
301     LoginPage.passwordInput.type("pass");
302     LoginPage.passwordInput.should("have.value", "pass");
303   });
304 });
305 });
306
307 export { LoginPage };
308
309
```

Listing 25: Test Scenarios cho Login

6.1.3 CI/CD Integration cho Login Tests (1.5 điểm)

```
1 name: Automation Tests - Login E2E & Integration
2
3 on:
4   push:
5     branches: [main, develop]
6     paths:
7       - "frontend/**"
8       - ".github/workflows/**"
9   pull_request:
10    branches: [main, develop]
11    paths:
12      - "frontend/**"
13  workflow_dispatch:
14
```



```
15 jobs:
16   test:
17     runs-on: ubuntu-latest
18     timeout-minutes: 15 # Tự hủy nếu treo quá 15p
19
20     strategy:
21       matrix:
22         node-version: [20.x] # Bắt buộc dùng Node 20 cho Vite mới
23         fail-fast: false
24
25     steps:
26       - name: Checkout code
27         uses: actions/checkout@v4
28
29       - name: Setup Node.js ${{ matrix.node-version }}
30         uses: actions/setup-node@v4
31         with:
32           node-version: ${{ matrix.node-version }}
33           # Tạm thời tắt cache để tránh xung đột file lock cũ
34           # cache: "npm"
35           # cache-dependency-path: "frontend/package-lock.json"
36
37       - name: Install dependencies
38         working-directory: ./frontend
39         run: |
40           # Xóa file lock cũ và node_modules để tránh xung đột
41           rm -rf node_modules package-lock.json
42           # Dùng npm install (thay vì npm ci) để nó tự tìm gói Linux
43           npm install
44
45       # --- JEST TEST ---
46       - name: Run Integration Tests (Jest)
47         id: run-jest # Đã thêm ID để dùng cho báo cáo
48         working-directory: ./frontend
49         run: npm test -- --coverage --testPathPattern="Integration|Mock"
50         --passWithNoTests
51
52       - name: Upload Jest Coverage Reports
53         uses: actions/upload-artifact@v4
54         if: always()
55         with:
56           name: jest-coverage-${{ matrix.node-version }}
57           path: frontend/coverage/
58           retention-days: 15
59
60       # --- BUILD & E2E ---
61       - name: Build frontend
62         working-directory: ./frontend
63         run: npm run build
64
65       - name: Start dev server & Wait
66         working-directory: ./frontend
67         run: |
68           npm run dev &
69           npx wait-on http://localhost:5173 --timeout 60000
```

```
70 # --- CYPRESS TEST ---
71 - name: Run E2E Tests (Cypress)
72   working-directory: ./frontend
73   run: npx cypress run --spec "src/tests/cypress/e2e/login-
scenarios.cy.js" --record false
74   env:
75     CYPRESS_BASE_URL: http://localhost:5173
76
77 # --- UPLOAD ARTIFACTS (KHI LỖI) ---
78 - name: Upload Cypress Screenshots
79   uses: actions/upload-artifact@v4
80   if: failure()
81   with:
82     name: cypress-screenshots-${{ matrix.node-version }}
83     path: frontend/src/tests/cypress/screenshots/
84     retention-days: 15
85
86 - name: Upload Cypress Videos
87   uses: actions/upload-artifact@v4
88   if: failure()
89   with:
90     name: cypress-videos-${{ matrix.node-version }}
91     path: frontend/src/tests/cypress/videos/
92     retention-days: 15
93
94 # --- REPORTING ---
95 - name: Generate Test Report Summary
96   if: always()
97   run: |
98     echo "##      í      Automation Test Results" >>
$GITHUB_STEP_SUMMARY
99     echo "- **Node**: ${matrix.node-version}" >>
$GITHUB_STEP_SUMMARY
100    echo "- **Jest**: ${steps.run-jest.outcome}" >>
$GITHUB_STEP_SUMMARY
101    echo "- **Cypress**: Check logs above" >> $GITHUB_STEP_SUMMARY
102
103 - name: Comment PR with Test Results
104   if: github.event_name == 'pull_request' && always()
105   uses: actions/github-script@v7
106   with:
107     script: |
108       const status = `${job.status}`;
109       const icon = status === 'success' ? 'Dung' : 'Sai';
110       github.rest.issues.createComment({
111         issue_number: context.issue.number,
112         owner: context.repo.owner,
113         repo: context.repo.repo,
114         body: `## ${icon} Automation Tests (Node ${matrix.node-
version })\n\n- **Status**: ${status}\n- **Jest**: ${steps.run-
jest.outcome }\n- **Artifacts**: Check Summary for details.`
115       })
116
```

Listing 26: CI/CD Integration cho Login Tests

6.2 Câu 5.2: Product - E2E Automation Testing (5 điểm)

6.2.1 Setup Page Object Model (1 điểm)

Implement POM cho Product pages:

```
1 frontend/src/tests/cypress/pages/  
2   ProductPage.js  
3
```

Listing 27: Implement POM cho Product page

6.2.2 E2E Test Scenarios cho Product (2.5 điểm)

Viết automated tests cho CRUD operations:

- a) Test Create product flow (0.5 điểm)
- b) Test Read/List products (0.5 điểm)
- c) Test Update product (0.5 điểm)
- d) Test Delete product (0.5 điểm)
- e) Test Search/Filter functionality (0.5 điểm)

```
1 /**  
2  * ProductPage - Page Object Model for Product Management Page  
3  *  
4  * This class encapsulates all selectors and methods for Product  
5  * Management page  
6  * interactions following the Page Object Model (POM) design  
7  * pattern.  
8  *  
9  * Page Sections:  
10 * 1. Header & Search - Title, add button, search, filters  
11 * 2. Product Table - Table rows, product data display  
12 * 3. Action Buttons - View, Edit, Delete for each product  
13 * 4. Pagination - Next/Previous page navigation  
14 * 5. Modal Forms - Add/Edit product forms  
15 * 6. Form Inputs - Name, price, quantity, category, description  
16 * 7. Validation Messages - Error messages for each field  
17 * 8. Confirmation Dialogs - Delete confirmation  
18 * 9. Notifications - Success/Error notifications  
19 * 10. Detail View - Product detail display modal  
20 *  
21 * Features:  
22 * - Complete CRUD operations (Create, Read, Update, Delete)  
23 * - Search and filter products  
24 * - Form validation  
25 * - Pagination support  
26 * - Confirmation dialogs  
27 * - Success/Error notifications  
28 * - Product detail view  
29 * - Comprehensive element selectors  
30 * - Chainable methods for fluent API
```

```
29 *
30 * Usage Example:
31 * ```javascript
32 * import ProductPage from './pages/ProductPage';
33 *
34 * describe('Product Management Tests', () => {
35 *   it('should create new product', () => {
36 *     ProductPage
37 *       .navigateToProductPage()
38 *       .createProduct({
39 *         name: 'iPhone 15',
40 *         price: '999.99',
41 *         quantity: '50',
42 *         category: 'Electronics',
43 *         description: 'Latest iPhone model'
44 *       })
45 *       .verifySuccessNotification('Product created successfully')
46 *       .verifyProductExists('iPhone 15');
47 *   });
48 *
49 *   it('should update product', () => {
50 *     ProductPage
51 *       .updateProduct('iPhone 15', { price: '1099.99' })
52 *       .verifySuccessNotification('Product updated');
53 *   });
54 *
55 *   it('should delete product', () => {
56 *     ProductPage
57 *       .deleteProduct('iPhone 15')
58 *       .verifyProductNotExists('iPhone 15');
59 *   });
60 * });
61 * ```
62 *
63 * @class ProductPage
64 * @author Software Testing Team
65 * @version 1.0
66 * @since 2025-11-26
67 */
68 class ProductPage {
69   // ===== HEADER & SEARCH SELECTORS =====
70   // Elements in the page header and search section
71
72   /**
73    * Get page title element
74    * @returns {Cypress.Chainable} Page title heading
75    */
76   get pageTitle() {
77     return cy.contains('h1', /quản lý sản phẩm|product management/i);
78   }
79
80   get addProductButton() {
81     return cy.contains('button', /thêm sản phẩm|add product/i);
82   }
83 }
```

```
84 get searchInput() {
85     return cy.get('.search-input');
86 }
87
88 get categoryFilter() {
89     return cy.get('.filter-select');
90 }
91
92 // ===== TABLE SELECTORS =====
93 get productTable() {
94     return cy.get('table.product-table');
95 }
96
97 get tableRows() {
98     return cy.get('table.product-table tbody tr');
99 }
100
101 get firstProductRow() {
102     return cy.get('table.product-table tbody tr').first();
103 }
104
105 getProductByName(productName) {
106     return cy.contains('table.product-table tbody tr', productName)
107     ;
108 }
109
110 // ===== ACTION BUTTONS IN TABLE =====
111 getViewButton(productName) {
112     return this.getProductByName(productName).within(() => {
113         return cy.get('button.blue');
114     });
115 }
116
117 getEditButton(productName) {
118     return this.getProductByName(productName).within(() => {
119         return cy.get('button.green');
120     });
121 }
122
123 getDeleteButton(productName) {
124     return this.getProductByName(productName).within(() => {
125         return cy.get('button.red');
126     });
127 }
128
129 // ===== PAGINATION SELECTORS =====
130 get paginationPrevButton() {
131     return cy.get('.pagination-btn').first();
132 }
133
134 get paginationNextButton() {
135     return cy.get('.pagination-btn').last();
136 }
137
138 get paginationInfo() {
139     return cy.get('.pagination-info');
```

```
139 }
140
141 // ===== MODAL SELECTORS =====
142 get modal() {
143     return cy.get('.modal');
144 }
145
146 get modalTitle() {
147     return cy.get('.modal-title');
148 }
149
150 get modalCloseButton() {
151     return cy.get('.modal-close');
152 }
153
154 get modalOverlay() {
155     return cy.get('.modal-overlay');
156 }
157
158 // ===== FORM INPUTS =====
159 get productNameInput() {
160     return cy.get('#name-input');
161 }
162
163 get priceInput() {
164     return cy.get('#price-input');
165 }
166
167 get quantityInput() {
168     return cy.get('#quantity-input');
169 }
170
171 get categorySelect() {
172     return cy.get('#category-select');
173 }
174
175 get descriptionInput() {
176     return cy.get('#description-textarea');
177 }
178
179 // ===== FORM ERROR MESSAGES =====
180 get nameError() {
181     return cy.contains('p.error-message', /tên sản phẩm|name/i);
182 }
183
184 get priceError() {
185     return cy.contains('p.error-message', /giá|price/i);
186 }
187
188 get quantityError() {
189     return cy.contains('p.error-message', /số l ượng|quantity/i);
190 }
191
192 get categoryError() {
193     return cy.contains('p.error-message', /danh mục|category/i);
194 }
```

```
195
196 get descriptionError() {
197     return cy.contains('p.error-message', /mô tả|description/i);
198 }
199
200 // ===== ACTION BUTTONS IN MODAL =====
201 get submitButton() {
202     return cy.get('.form-actions').find('button.btn-primary, button
203     .btn-success, button.btn-danger').first();
204 }
205
206 get cancelButton() {
207     return cy.get('.form-actions').find('button.btn-secondary');
208 }
209
210 // ===== CONFIRMATION DIALOG =====
211 get deleteConfirmDialog() {
212     return cy.get('.modal-small');
213 }
214
215 get deleteConfirmButton() {
216     return cy.get('.modal-small').find('button.btn-danger');
217 }
218
219 get deleteConfirmCancelButton() {
220     return cy.get('.modal-small').find('button.btn-secondary');
221 }
222
223 // ===== NOTIFICATION =====
224 get notification() {
225     return cy.get('.notification');
226 }
227
228 get successNotification() {
229     return cy.get('.notification.success');
230 }
231
232 get errorNotification() {
233     return cy.get('.notification.error');
234 }
235
236 // ===== DETAIL VIEW SELECTORS =====
237 get detailValue() {
238     return cy.get('.detail-value');
239 }
240
241 get detailDescription() {
242     return cy.get('.detail-description');
243 }
244
245 get editButtonInDetail() {
246     return cy.contains('button', /chính sửa|edit/i);
247 }
248
249 get emptyState() {
250     return cy.get('.empty-state');
```

```
250 }
251
252 // ===== METHODS =====
253
254 /**
255  * Navigate to product management page
256  */
257 navigateToProductPage() {
258     cy.visit('/');
259     this.pageTitle.should('be.visible');
260     return this;
261 }
262
263 /**
264  * Click Add Product button
265  */
266 clickAddProduct() {
267     this.addProductButton.click();
268     return this;
269 }
270
271 /**
272  * Fill product form
273  */
274 fillProductForm(productData) {
275     if (productData.name) {
276         this.productNameInput.clear().type(productData.name, { delay:
277         100 });
278     }
279     if (productData.price) {
280         this.priceInput.clear().type(productData.price, { delay: 100
281         });
282     }
283     if (productData.quantity) {
284         this.quantityInput.clear().type(productData.quantity, { delay
285         : 100 });
286     }
287     if (productData.category) {
288         this.categorySelect.select(productData.category);
289     }
290     if (productData.description) {
291         this.descriptionInput.clear().type(productData.description, {
292         delay: 100 });
293     }
294     return this;
295 }
296
297 /**
298  * Submit product form
299  */
300 submitForm() {
301     this.submitButton.click();
302     cy.wait(500);
303     return this;
304 }
```



```
302  /**
303   * Create new product
304   */
305  createProduct(productData) {
306    this.clickAddProduct();
307    cy.wait(300);
308    this.fillProductForm(productData);
309    this.submitForm();
310    return this;
311  }
312
313  /**
314   * Search product by name
315   */
316  searchProduct(productName) {
317    this.searchInput.clear().type(productName, { delay: 100 });
318    cy.wait(300);
319    return this;
320  }
321
322  /**
323   * Filter by category
324   */
325  filterByCategory(category) {
326    this.categoryFilter.select(category);
327    cy.wait(300);
328    return this;
329  }
330
331  /**
332   * View product details
333   */
334  viewProduct(productName) {
335    this.getViewButton(productName).click();
336    cy.wait(300);
337    return this;
338  }
339
340  /**
341   * Edit product
342   */
343  editProduct(productName) {
344    this.getEditButton(productName).click();
345    cy.wait(300);
346    return this;
347  }
348
349  /**
350   * Update product
351   */
352  updateProduct(productName, updatedData) {
353    this.editProduct(productName);
354    this.fillProductForm(updatedData);
355    this.submitForm();
356    return this;
357  }
```

```
358
359 /**
360  * Delete product with confirmation
361  */
362 deleteProduct(productName) {
363     this.getDeleteButton(productName).click();
364     cy.wait(300);
365     this.deleteConfirmButton.click();
366     cy.wait(300);
367     return this;
368 }
369
370 /**
371  * Open delete confirmation and cancel
372  */
373 deleteProductCancel(productName) {
374     this.getDeleteButton(productName).click();
375     cy.wait(300);
376     this.deleteConfirmCancelButton.click();
377     cy.wait(300);
378     return this;
379 }
380
381 /**
382  * Clear all form inputs
383  */
384 clearForm() {
385     this.productNameInput.clear();
386     this.priceInput.clear();
387     this.quantityInput.clear();
388     this.categorySelect.select('');
389     this.descriptionInput.clear();
390     return this;
391 }
392
393 /**
394  * Close modal
395  */
396 closeModal() {
397     this.modalCloseButton.click();
398     cy.wait(200);
399     return this;
400 }
401
402 /**
403  * Verify product form is visible
404  */
405 verifyFormVisible() {
406     this.productNameInput.should('be.visible');
407     this.priceInput.should('be.visible');
408     this.quantityInput.should('be.visible');
409     this.categorySelect.should('be.visible');
410     this.descriptionInput.should('be.visible');
411     return this;
412 }
413
```

```
414  /**
415   * Verify product exists in table
416   */
417  verifyProductExists(productName) {
418    this.getProductByName(productName).should('be.visible');
419    return this;
420  }
421
422  /**
423   * Verify product not exists in table
424   */
425  verifyProductNotExists(productName) {
426    cy.contains('table.product-table tbody tr', productName).should(
427      ('not.exist');
428    return this;
429  }
430
431  /**
432   * Verify success notification
433   */
434  verifySuccessNotification(message) {
435    cy.get('.notification.success', { timeout: 10000 }).should('be.
436      visible');
437    if (message) {
438      if (message instanceof RegExp) {
439        cy.get('.notification').invoke('text').should('match',
440          message);
441      } else {
442        cy.get('.notification').should('contain', message);
443      }
444    }
445    return this;
446  }
447
448  /**
449   * Verify error notification
450   */
451  verifyErrorNotification(message) {
452    cy.get('.notification.error', { timeout: 10000 }).should('be.
453      visible');
454    if (message) {
455      if (message instanceof RegExp) {
456        cy.get('.notification').invoke('text').should('match',
457          message);
458      } else {
459        cy.get('.notification').should('contain', message);
460      }
461    }
462    return this;
463  }
464
465  /**
466   * Verify error message for field
467   */
468  verifyFieldError(fieldName) {
469    let errorSelector;
```

```
465     switch(fieldName.toLowerCase()) {
466         case 'name':
467             case 'tên':
468                 errorSelector = this.nameError;
469                 break;
470             case 'price':
471             case 'giá':
472                 errorSelector = this.priceError;
473                 break;
474             case 'quantity':
475             case 'số l  ợng':
476                 errorSelector = this.quantityError;
477                 break;
478             case 'category':
479             case 'danh mục':
480                 errorSelector = this.categoryError;
481                 break;
482             case 'description':
483             case 'mô tá':
484                 errorSelector = this.descriptionError;
485                 break;
486             default:
487                 throw new Error('Unknown field: ${fieldName}');
488         }
489         errorSelector.should('be.visible');
490         return this;
491     }
492
493     /**
494      * Get total product count
495      */
496     getTotalProductCount() {
497         return this.tableRows.its('length');
498     }
499
500     /**
501      * Verify pagination visible
502      */
503     verifyPaginationVisible() {
504         this.paginationInfo.should('be.visible');
505         return this;
506     }
507
508     /**
509      * Click next page
510      */
511     clickNextPage() {
512         this.paginationNextButton.click();
513         cy.wait(200);
514         return this;
515     }
516
517     /**
518      * Click previous page
519      */
520     clickPreviousPage() {
```

```
521     this.paginationPrevButton.click();
522     cy.wait(200);
523     return this;
524 }
525
526 /**
527  * Verify table empty
528  */
529 verifyTableEmpty() {
530     // Check if empty state element exists or table shows empty
    message
531     cy.get('.product-table tbody tr').then($rows => {
532         expect($rows.text()).to.include('Không có sản phẩm');
533     });
534     return this;
535 }
536
537 /**
538  * Verify table not empty
539  */
540 verifyTableNotEmpty() {
541     cy.get('.product-table tbody tr').then($rows => {
542         expect($rows.text()).not.to.include('Không có sản phẩm');
543     });
544     return this;
545 }
546
547 /**
548  * Get product row data
549  */
550 getProductRowData(productName) {
551     return this.getProductByName(productName).within(() => {
552         return {
553             name: cy.get('td').eq(0),
554             price: cy.get('td').eq(1),
555             quantity: cy.get('td').eq(2),
556             category: cy.get('td').eq(3)
557         };
558     });
559 }
560 }
561
562 // Export page object
563 export default new ProductPage();
564
565
```

Listing 28: Test Scenarios cho Product

6.2.3 CI/CD Integration (1.5 điểm)

Setup complete CI/CD pipeline:

```
1 name: FloginFE_BE CI/CD Pipeline
2
```

```
3 on:
4   push:
5     branches: [main, develop]
6   pull_request:
7     branches: [main]
8
9 env:
10  JAVA_VERSION: "21"
11  NODE_VERSION: "20"
12  BACKEND_PORT: 6969
13  SQL_SERVER_SA_PASSWORD: "YourStrong!Passw0rd@"
14
15 permissions:
16   contents: read
17   checks: write
18   pull-requests: write
19   statuses: write
20
21 jobs:
22   # JOB 1: Backend Build & Test
23
24   backend-tests:
25     name: Backend - Build & Test
26     runs-on: ubuntu-latest
27
28     services:
29       sqlserver:
30         image: mcr.microsoft.com/mssql/server:2022-latest
31         env:
32           ACCEPT_EULA: Y
33           SA_PASSWORD: ${ env.SQL_SERVER_SA_PASSWORD }
34           MSSQL_PID: Developer
35         ports:
36           - 1433:1433
37         options: >-
38           --user 0:0
39           --health-cmd "/opt/mssql-tools18/bin/sqlcmd -S localhost -U sa
-P 'YourStrong!Passw0rd@' -C -Q 'SELECT 1' || exit 1"
40           --health-interval 10s
41           --health-timeout 5s
42           --health-retries 10
43           --health-start-period 10s
44
45     steps:
46       - name: Checkout code
47         uses: actions/checkout@v3
48
49       - name: Setup Java ${ env.JAVA_VERSION }
50         uses: actions/setup-java@v3
51         with:
52           java-version: ${ env.JAVA_VERSION }
53           distribution: "temurin"
54           cache: "maven"
55
56       - name: Wait for SQL Server to be ready
57         run: |
```

```
58     echo "Waiting for SQL Server to start..."
59     for i in {1..30}; do
60         if docker exec $(docker ps -q -f ancestor=mcr.microsoft.com/
mssql/server:2022-latest) /opt/mssql-tools18/bin/sqlcmd -S localhost
-U sa -P 'YourStrong!PasswOrd@' -C -Q "SELECT 1" > /dev/null 2>&1;
then
61             echo "SQL Server is ready!"
62             break
63         fi
64         echo "Attempt $i: SQL Server not ready yet..."
65         sleep 5
66     done
67
68     - name: Create Test Database
69     run: |
70         docker exec $(docker ps -q -f ancestor=mcr.microsoft.com/mssql
/server:2022-latest) \
71             /opt/mssql-tools18/bin/sqlcmd \
72             -S localhost -U sa -P 'YourStrong!PasswOrd@' \
73             -C -Q "CREATE DATABASE STDatabase_Test;"
74         echo "Test database created successfully"
75
76     - name: Verify Maven wrapper
77     run: |
78         cd backend
79         chmod +x mvnw
80         ./mvnw --version
81
82     - name: Build Backend (Skip Tests)
83     run: |
84         cd backend
85         ./mvnw clean install -DskipTests
86     env:
87         MAVEN_OPTS: -Xmx3072m
88
89     - name: Run Unit Tests
90     run: |
91         cd backend
92         ./mvnw test -Dtest="!**/*IntegrationTest,!**/*E2ETest"
93     env:
94         SPRING_PROFILES_ACTIVE: test
95         SPRING_DATASOURCE_URL: jdbc:sqlserver://localhost:1433;
databaseName=STDatabase_Test;trustServerCertificate=true
96         SPRING_DATASOURCE_USERNAME: sa
97         SPRING_DATASOURCE_PASSWORD: ${ env.SQL_SERVER_SA_PASSWORD }}
98         SPRING_DATASOURCE_DRIVER_CLASS_NAME: com.microsoft.sqlserver.
jdbc.SQLServerDriver
99         SPRING_TEST_DATABASE_REPLACE: NONE
100        SPRING_JPA_DATABASE_PLATFORM: org.hibernate.dialect.
SQLServerDialect
101
102     - name: Run Security Tests
103     run: |
104         cd backend
105         ./mvnw test -Dtest="com.flogin.security.*Test"
106     env:
```

```
107     SPRING_PROFILES_ACTIVE: test
108     SPRING_DATASOURCE_URL: jdbc:sqlserver://localhost:1433;
databaseName=STDatabase_Test;trustServerCertificate=true
109     SPRING_DATASOURCE_USERNAME: sa
110     SPRING_DATASOURCE_PASSWORD: ${ env.SQL_SERVER_SA_PASSWORD }}
111     SPRING_DATASOURCE_DRIVER_CLASS_NAME: com.microsoft.sqlserver.
jdbc.SQLServerDriver
112     SPRING_TEST_DATABASE_REPLACE: NONE
113     SPRING_JPA_DATABASE_PLATFORM: org.hibernate.dialect.
SQLServerDialect
114
115 - name: Run Integration Tests
116   run: |
117     cd backend
118     ./mvnw test -Dtest="**/*IntegrationTest"
119   env:
120     SPRING_PROFILES_ACTIVE: test
121     SPRING_DATASOURCE_URL: jdbc:sqlserver://localhost:1433;
databaseName=STDatabase_Test;trustServerCertificate=true
122     SPRING_DATASOURCE_USERNAME: sa
123     SPRING_DATASOURCE_PASSWORD: ${ env.SQL_SERVER_SA_PASSWORD }}
124
125 - name: Generate JaCoCo Coverage Report
126   run: |
127     cd backend
128     ./mvnw jacoco:report
129
130 - name: Upload Backend Coverage to Codecov
131   uses: codecov/codecov-action@v3
132   with:
133     token: ${ secrets.CODECOV_TOKEN }}
134     files: ./backend/target/site/jacoco/jacoco.xml
135     flags: backend
136     name: backend-coverage
137     fail_ci_if_error: false
138
139 - name: Upload Backend Test Results
140   if: always()
141   uses: actions/upload-artifact@v4
142   with:
143     name: backend-test-results
144     path: |
145       backend/target/surefire-reports/
146       backend/target/site/jacoco/
147
148 - name: Publish Test Report
149   if: always()
150   uses: dorny/test-reporter@v1
151   with:
152     name: Backend Tests
153     path: backend/target/surefire-reports/*.xml
154     reporter: java-junit
155     fail-on-error: false
156
157
158 # JOB 2: Frontend Build & Test
```



```
159 frontend-tests:
160   name: Frontend - Build & Test
161   runs-on: ubuntu-latest
162
163   steps:
164     - name: Checkout code
165       uses: actions/checkout@v3
166
167     - name: Setup Node.js ${{ env.NODE_VERSION }}
168       uses: actions/setup-node@v3
169       with:
170         node-version: ${{ env.NODE_VERSION }}
171         # Tạm thời tắt cache để tránh restore file lock cũ
172         # cache: "npm"
173         # cache-dependency-path: frontend/package-lock.json
174
175     # FIX 2: Xóa package-lock.json để ép npm tìm gói Linux
176     - name: Install Frontend Dependencies
177       run: |
178         cd frontend
179         rm -rf node_modules package-lock.json
180         npm install
181
182     - name: Run ESLint
183       run: |
184         cd frontend
185         npm run lint || true
186         continue-on-error: true
187
188     - name: Run Frontend Unit Tests
189       run: |
190         cd frontend
191         npm test -- --coverage --watchAll=false
192       env:
193         CI: true
194
195     - name: Upload Frontend Coverage to Codecov
196       uses: codecov/codecov-action@v3
197       with:
198         token: ${{ secrets.CODECOV_TOKEN }}
199         files: ./frontend/coverage/lcov.info
200         flags: frontend
201         name: frontend-coverage
202         fail_ci_if_error: false
203
204     - name: Build Frontend
205       run: |
206         cd frontend
207         npm run build
208       env:
209         CI: false
210         REACT_APP_API_URL: http://localhost:6969
211
212     - name: Upload Frontend Build Artifacts
213       uses: actions/upload-artifact@v4
```

```
215     with:
216       name: frontend-build
217       path: frontend/build/
218
219   - name: Upload Frontend Test Results
220     if: always()
221     uses: actions/upload-artifact@v4
222     with:
223       name: frontend-test-results
224       path: |
225         frontend/coverage/
226
227
228 # JOB 3: E2E Tests with SQL Server
229
230 e2e-tests:
231   name: E2E Tests
232   runs-on: ubuntu-latest
233   needs: [backend-tests, frontend-tests]
234
235   services:
236     sqlserver:
237       image: mcr.microsoft.com/mssql/server:2022-latest
238       env:
239         ACCEPT_EULA: Y
240         SA_PASSWORD: ${ env.SQL_SERVER_SA_PASSWORD }
241         MSSQL_PID: Developer
242       ports:
243         - 1433:1433
244       options: >-
245         --user 0:0
246         --health-cmd "/opt/mssql-tools18/bin/sqlcmd -S localhost -U sa
-P 'YourStrong!Passw0rd@' -C -Q 'SELECT 1' || exit 1"
247         --health-interval 10s
248         --health-timeout 10s
249         --health-retries 20
250         --health-start-period 60s
251
252   steps:
253     - name: Checkout code
254       uses: actions/checkout@v3
255
256     - name: Setup Java ${ env.JAVA_VERSION }
257       uses: actions/setup-java@v3
258       with:
259         java-version: ${ env.JAVA_VERSION }
260         distribution: "temurin"
261         cache: "maven"
262
263     - name: Setup Node.js ${ env.NODE_VERSION }
264       uses: actions/setup-node@v3
265       with:
266         node-version: ${ env.NODE_VERSION }
267         # cache: "npm"
268         # cache-dependency-path: frontend/package-lock.json
269
```

```
270 - name: Create E2E Database
271 run: |
272     docker exec $(docker ps -q -f ancestor=mcr.microsoft.com/mssql
/server:2022-latest) \
273         /opt/mssql-tools18/bin/sqlcmd \
274         -S localhost -U sa -P "${{ env.SQL_SERVER_SA_PASSWORD }}" \
275         -C -Q "CREATE DATABASE STDatabase_E2E;"
276
277 - name: Start Backend Server
278 run: |
279     cd backend
280     chmod +x mvnw
281     ./mvnw clean install -DskipTests
282     nohup ./mvnw spring-boot:run \
283         -Dspring-boot.run.arguments="--spring.datasource.url=jdbc:
sqlserver://localhost:1433;databaseName=STDatabase_E2E;
trustServerCertificate=true --spring.datasource.username=sa --spring.
datasource.password=${{ env.SQL_SERVER_SA_PASSWORD }}" &
284     echo $! > backend.pid
285
286     # Wait for backend to be ready
287     echo "Waiting for backend to start..."
288     for i in {1..60}; do
289         if curl -s http://localhost:6969/actuator/health > /dev/null
2>&1; then
290             echo "Backend is ready!"
291             break
292         fi
293         echo "Attempt $i: Backend not ready yet..."
294         sleep 5
295     done
296
297 - name: Install Frontend Dependencies
298 run: |
299     cd frontend
300     rm -rf node_modules package-lock.json
301     npm install
302
303 - name: Start Frontend Server
304 run: |
305     cd frontend
306     nohup npm run dev &
307     echo $! > frontend.pid
308
309     # Wait for frontend to be ready
310     echo "Waiting for frontend to start..."
311     for i in {1..60}; do
312         if curl -s http://localhost:5173 > /dev/null 2>&1; then
313             echo "Frontend is ready!"
314             break
315         fi
316         echo "Attempt $i: Frontend not ready yet..."
317         sleep 5
318     done
319 env:
320     REACT_APP_API_URL: http://localhost:6969
```

```
321     PORT: 5173
322     CI: false
323
324     - name: Run E2E Tests (if exists)
325       run: |
326         cd frontend
327         if [ -f "package.json" ] && grep -q "test:e2e" package.json;
then
328           npm run test:e2e || echo "E2E tests not configured yet"
329         else
330           echo "E2E tests not found, skipping..."
331         fi
332     continue-on-error: true
333
334     - name: Stop Servers
335       if: always()
336       run: |
337         if [ -f backend/backend.pid ]; then
338           kill $(cat backend/backend.pid) || true
339         fi
340         if [ -f frontend/frontend.pid ]; then
341           kill $(cat frontend/frontend.pid) || true
342         fi
343
344     - name: Upload E2E Test Results
345       if: always()
346       uses: actions/upload-artifact@v4
347       with:
348         name: e2e-test-results
349         path: |
350           frontend/cypress/screenshots/
351           frontend/cypress/videos/
352           frontend/playwright-report/
353
354
355 # JOB 4: Performance Tests (k6) with SQL Server
356
357 performance-tests:
358   name: Performance Tests (k6)
359   runs-on: ubuntu-latest
360   needs: [backend-tests]
361   if: github.event_name == 'push' && github.ref == 'refs/heads/main'
362
363   services:
364     sqlserver:
365       image: mcr.microsoft.com/mssql/server:2022-latest
366       env:
367         ACCEPT_EULA: Y
368         SA_PASSWORD: ${ env.SQL_SERVER_SA_PASSWORD }}
369         MSSQL_PID: Developer
370       ports:
371         - 1433:1433
372       options: >-
373         --user 0:0
374         --health-cmd "/opt/mssql-tools18/bin/sqlcmd -S localhost -U sa
-P 'YourStrong!Passw0rd@' -C -Q 'SELECT 1' || exit 1"
```

```
375     --health-interval 10s
376     --health-timeout 10s
377     --health-retries 20
378     --health-start-period 60s
379
380 steps:
381   - name: Checkout code
382     uses: actions/checkout@v3
383
384   - name: Setup Java ${ env.JAVA_VERSION }
385     uses: actions/setup-java@v3
386     with:
387       java-version: ${ env.JAVA_VERSION }
388       distribution: "temurin"
389       cache: "maven"
390
391   - name: Create Performance Database
392     run: |
393       docker exec $(docker ps -q -f ancestor=mcr.microsoft.com/mssql
394 /server:2022-latest) \
395       /opt/mssql-tools18/bin/sqlcmd \
396       -S localhost -U sa -P 'YourStrong!PasswOrd@' \
397       -C -Q "CREATE DATABASE STDatabase_Perf;"
398
399   - name: Start Backend Server
400     run: |
401       cd backend
402       chmod +x mvnw
403       ./mvnw clean install -DskipTests
404       nohup ./mvnw spring-boot:run \
405         -Dspring-boot.run.arguments="--spring.datasource.url=jdbc:
406 sqlserver://localhost:1433;databaseName=STDatabase_Perf;
407 trustServerCertificate=true --spring.datasource.username=sa --spring.
408 datasource.password=${ env.SQL_SERVER_SA_PASSWORD }}" &
409       echo $! > backend.pid
410
411       # Wait for backend
412       for i in {1..60}; do
413         if curl -s http://localhost:6969/actuator/health > /dev/null
414         2>&1; then
415           echo "Backend is ready!"
416           break
417         fi
418         sleep 5
419       done
420
421   - name: Install k6
422     run: |
423       sudo gpg -k
424       sudo gpg --no-default-keyring --keyring /usr/share/keyrings/k6
425 -archive-keyring.gpg --keyserver hkps://keyserver.ubuntu.com:80 --recv
426 -keys C5AD17C747E3415A3642D57D77C6C491D6AC1D69
427       echo "deb [signed-by=/usr/share/keyrings/k6-archive-keyring.
428 gpg] https://dl.k6.io/deb stable main" | sudo tee /etc/apt/sources.
429 list.d/k6.list
430       sudo apt-get update
```

```
422     sudo apt-get install k6
423
424     - name: Run Performance Tests (100 users - 1 minute for CI)
425       run: |
426         cd performance-tests/scripts
427         # Shortened test duration for CI (1 min instead of 5 min)
428         k6 run login-test.js --vus 100 --duration 1m --out json=../
429         results/ci-login-100-users.json
430         continue-on-error: true
431
432     - name: Analyze Performance Results
433       run: |
434         cd performance-tests
435         if [ -f results/ci-login-100-users.json ]; then
436           echo "Performance test completed. Results saved."
437           # Extract basic metrics
438           echo "=== Performance Summary ==="
439           grep -o '"http_req_duration"[^"]*' results/ci-login-100-
440           users.json | head -10 || true
441         fi
442
443     - name: Stop Backend
444       if: always()
445       run: |
446         if [ -f backend/backend.pid ]; then
447           kill $(cat backend/backend.pid) || true
448         fi
449
450     - name: Upload Performance Results
451       if: always()
452       uses: actions/upload-artifact@v4
453       with:
454         name: performance-test-results
455         path: performance-tests/results/
456
457 # JOB 6: Code Quality Analysis
458
459 code-quality:
460   name: Code Quality Analysis
461   runs-on: ubuntu-latest
462   needs: [backend-tests, frontend-tests]
463
464   steps:
465     - name: Checkout code
466       uses: actions/checkout@v3
467       with:
468         fetch-depth: 0
469
470     - name: Setup Java ${ env.JAVA_VERSION }
471       uses: actions/setup-java@v3
472       with:
473         java-version: ${ env.JAVA_VERSION }
474         distribution: "temurin"
475
476     - name: SonarCloud Scan (if configured)
```

```
476     run: |
477         echo "SonarCloud scan would run here if configured"
478         echo "Requires SONAR_TOKEN secret"
479     continue-on-error: true
480
481 - name: Generate Coverage Report Summary
482   run: |
483     echo "## Test Coverage Summary" > coverage-summary.md
484     echo "" >> coverage-summary.md
485     echo "### Backend Coverage" >> coverage-summary.md
486     echo "- Unit Tests:      " >> coverage-summary.md
487     echo "- Integration Tests:  " >> coverage-summary.md
488     echo "" >> coverage-summary.md
489     echo "### Frontend Coverage" >> coverage-summary.md
490     echo "- Component Tests:    " >> coverage-summary.md
491
492 - name: Upload Coverage Summary
493   uses: actions/upload-artifact@v4
494   with:
495     name: coverage-summary
496     path: coverage-summary.md
497
498
499 # JOB 7: Build Summary & Notification
500
501 build-summary:
502   name: Build Summary
503   runs-on: ubuntu-latest
504   needs:
505     [
506       backend-tests,
507       frontend-tests,
508       e2e-tests,
509       performance-tests,
510       code-quality,
511     ]
512   if: always()
513
514   steps:
515     - name: Generate Build Summary
516       run: |
517         echo "## CI/CD Pipeline Summary" > summary.md
518         echo "" >> summary.md
519         echo "### Jobs Status:" >> summary.md
520         echo "- Backend Tests (SQL Server): ${ needs.backend-tests.
result }}" >> summary.md
521         echo "- Frontend Tests: ${ needs.frontend-tests.result }}" >>
summary.md
522         echo "- E2E Tests: ${ needs.e2e-tests.result }}" >> summary.
md
523         echo "- Performance Tests: ${ needs.performance-tests.result
}}" >> summary.md
524         echo "- Security Scan: ${ needs.security-scan.result }}" >>
summary.md
525         echo "- Code Quality: ${ needs.code-quality.result }}" >>
summary.md
```

```
526     echo "" >> summary.md
527     echo "### Build Information:" >> summary.md
528     echo "- Branch: ${github.ref_name}" >> summary.md
529     echo "- Commit: ${github.sha}" >> summary.md
530     echo "- Author: ${github.actor}" >> summary.md
531     echo "- Database: SQL Server 2022" >> summary.md
532     cat summary.md
533
534 - name: Comment PR (if applicable)
535   if: github.event_name == 'pull_request'
536   uses: actions/github-script@v6
537   with:
538     github-token: ${secrets.GITHUB_TOKEN}
539     script: |
540       github.rest.issues.createComment({
541         issue_number: context.issue.number,
542         owner: context.repo.owner,
543         repo: context.repo.repo,
544         body: '## CI/CD Pipeline Completed\n\n All tests executed
with **SQL Server 2022**\n\nCheck the workflow run for detailed
results.'
545       })
546
547 - name: Pipeline Complete
548   run: |
549     echo "====="
550     echo "  CI/CD Pipeline Completed!"
551     echo "  Database: SQL Server 2022"
552     echo "====="
```

Listing 29: .github/workflows/cicd.yml

7 Phần Mở Rộng (Bonus 20 điểm)

7.1 Performance Testing (10 điểm)

7.1.1 Yêu cầu

- Setup JMeter hoặc k6 cho performance testing (2 điểm)
- Viết performance tests cho Login API (3 điểm):
- Viết performance tests cho Product API (3 điểm)
- Phân tích kết quả và đưa ra recommendations (2 điểm)

7.2 Security Testing (10 điểm)

7.2.1 Yêu cầu

- Test common vulnerabilities (5 điểm):

- SQL Injection

```
1      @Test
2      @DisplayName("TC1.1: SQL Injection - Login bypass với ' OR '1'='1'")
3      void testSqlInjection_LoginBypass_Classic() throws
4      Exception {
5          LoginRequest maliciousRequest = new LoginRequest("admin
6          ' OR '1'='1", "anything");
7
8          mockMvc.perform(post("/api/auth/login")
9                      .contentType(MediaType.APPLICATION_JSON)
10                     .content(objectMapper.writeValueAsString(
11                     maliciousRequest)))
12                     .andExpect(status().isBadRequest())
13                     .andExpect(jsonPath("$.success").value(false));
14      }
15
16      @Test
17      @DisplayName("TC1.2: SQL Injection - Login bypass với ' OR 1=1--")
18      void testSqlInjection_LoginBypass_Comment() throws
19      Exception {
20          LoginRequest maliciousRequest = new LoginRequest("admin
21          ' OR 1=1--", "anything");
22
23          mockMvc.perform(post("/api/auth/login")
24                      .contentType(MediaType.APPLICATION_JSON)
25                     .content(objectMapper.writeValueAsString(
26                     maliciousRequest)))
27                     .andExpect(status().isBadRequest())
28                     .andExpect(jsonPath("$.success").value(false));
29      }
30
31      @Test
```

```
26     @DisplayName("TC1.3: SQL Injection - Login bypass với '  
DROP TABLE users--")  
27     void testSqlInjection_LoginBypass_DropTable() throws  
Exception {  
28         LoginRequest maliciousRequest = new LoginRequest("admin  
'; DROP TABLE users--", "password");  
29  
30  
31         mockMvc.perform(post("/api/auth/login")  
32             .contentType(MediaType.APPLICATION_JSON)  
33             .content(objectMapper.writeValueAsString(  
maliciousRequest)))  
34             .andExpect(status().isBadRequest());  
35  
36         // Verify table vẫn tồn tại  
37         long userCount = userRepository.count();  
38         assert userCount > 0 : "Users table should still exist"  
39     ;  
40     }  
41
```

Listing 30: SqlInjectionTest.java

- Cross-Site Scripting (XSS)

```
1         @Test  
2         @DisplayName("TC2.1: XSS - onerror event handler")  
3         void testXss_EventHandler_OnError() throws Exception {  
4             CreateProductRequest xssRequest = new  
CreateProductRequest("<img src=x onerror=alert('XSS')>",  
100.0, "Test", 10, "Electronics");  
5  
6             mockMvc.perform(post("/api/products")  
7                 .header("Authorization", authToken)  
8                 .contentType(MediaType.APPLICATION_JSON)  
9                 .content(objectMapper.writeValueAsString(  
xssRequest)))  
10                .andExpect(status().isBadRequest())  
11                .andExpect(jsonPath("$.errors.productName").  
exists());  
12        }  
13  
14  
15        @Test  
16        @DisplayName("TC10.1: XSS - innerHTML manipulation attempt")  
17        void testXss_DomBased_InnerHtml() throws Exception {  
18            CreateProductRequest xssRequest = new  
CreateProductRequest("Product<img src=x onerror='document.  
body.innerHTML=\"<h1>Hacked</h1>\">", 100.0, "Test", 10, "  
Electronics");  
19  
20            mockMvc.perform(post("/api/products")  
21                .header("Authorization", authToken)  
22                .contentType(MediaType.APPLICATION_JSON)
```

```
23         .content(objectMapper.writeValueAsString(  
xssRequest)))  
24         .andExpect(status().isBadRequest())  
25         .andExpect(jsonPath("$.errors.productName").  
exists());  
26     }  
27  
28
```

Listing 31: XssTest.java

- Authentication bypass attempts

```
1         @Test  
2         @DisplayName("TC1.3: Auth bypass - Empty credentials")  
3         void testAuthBypass_EmptyCredentials() throws Exception {  
4             LoginRequest loginRequest = new LoginRequest("", "");  
5  
6             mockMvc.perform(post("/api/auth/login")  
7                 .contentType(MediaType.APPLICATION_JSON)  
8                 .content(objectMapper.writeValueAsString(  
loginRequest)))  
9                 .andExpect(status().isBadRequest());  
10        }  
11  
12        @Test  
13        @DisplayName("TC2.1: Auth bypass - Wrong password")  
14        void testAuthBypass_WrongPassword() throws Exception {  
15            LoginRequest loginRequest = new LoginRequest("admin", "  
wrongpassword");  
16  
17            mockMvc.perform(post("/api/auth/login")  
18                .contentType(MediaType.APPLICATION_JSON)  
19                .content(objectMapper.writeValueAsString(  
loginRequest)))  
20                .andExpect(status().isBadRequest()) // Password  
validation fails (no digit)  
21                .andExpect(jsonPath("$.success").value(false));  
22        }  
23  
24        @Test  
25        @DisplayName("TC4.1: Auth bypass - Invalid JWT token")  
26        void testAuthBypass_InvalidToken() throws Exception {  
27            mockMvc.perform(get("/api/products")  
28                .header("Authorization", "Bearer invalid.token.  
here"))  
29                .andExpect(status().isForbidden());  
30        }  
31  
32
```

Listing 32: AuthBypassTest.java

b) Test input validation và sanitization (3 điểm)

```
1 //=====input validation=====
```

```
2  @Test
3  @DisplayName("TC1.1: Validation - Product name quá ngắn (< 3
4  chars)")
5  void testValidation_ProductName_TooShort() throws Exception {
6      CreateProductRequest request = new CreateProductRequest("AB
7      ", 100.0, "Test", 10, "Electronics");
8
9      mockMvc.perform(post("/api/products")
10         .header("Authorization", authToken)
11         .contentType(MediaType.APPLICATION_JSON)
12         .content(objectMapper.writeValueAsString(request)))
13         .andExpect(status().isBadRequest())
14         .andExpect(jsonPath("$.errors.productName").exists
15         ());
16 }
17
18 @Test
19 @DisplayName("TC6.1: Sanitization - Special characters trong
20 product name")
21 void testSanitization_SpecialCharacters_ProductName() throws
22 Exception {
23     CreateProductRequest request = new CreateProductRequest("
24     Product <>&\"' ", 100.0, "Test", 10, "Electronics");
25
26     mockMvc.perform(post("/api/products")
27         .header("Authorization", authToken)
28         .contentType(MediaType.APPLICATION_JSON)
29         .content(objectMapper.writeValueAsString(request)))
30         .andExpect(status().isBadRequest())
31         .andExpect(jsonPath("$.errors.productName").exists
32         ());
33 }
```

Listing 33: InputValidationTest.java

c) Security best practices implementation (2 điểm):

```
1  @SpringBootTest
2  @AutoConfigureMockMvc
3  @DisplayName("Security Best Practices Tests")
4  public class SecurityBestPracticesTest {
5      @Autowired
6      private MockMvc mockMvc;
7
8      @Autowired
9      private PasswordEncoder passwordEncoder;
10
11      @Autowired
12      private UserRepository userRepository;
13
14      @Test
```

```
15     @DisplayName("1. Password Hashing - Mật khẩu đ ọc hash bằng  
BCrypt")  
16     void testPasswordHashing() {  
17         // Arrange  
18         String rawPassword = "mySecurePassword123";  
19  
20         // Act - Hash password  
21         String hashedPassword = passwordEncoder.encode(rawPassword)  
22         ;  
23  
24         // Assert  
25         assertNotNull(hashedPassword, "Hashed password không đ ọc  
null");  
26         assertEquals(rawPassword, hashedPassword, "Password khô  
ng đ ọc l u dạng plain text");  
27         assertTrue(hashedPassword.startsWith("$2a$") ||  
hashedPassword.startsWith("$2b$"),  
28             "Password phải đ ọc hash bằng BCrypt");  
29         assertTrue(hashedPassword.length() >= 60, "BCrypt hash phải  
có độ dài >= 60 ký tự");  
30  
31         // Verify password matching  
32         boolean matches = passwordEncoder.matches(rawPassword,  
hashedPassword);  
33         assertTrue(matches, "Password encoder phải verify đ ọc  
password đúng");  
34  
35         // Verify wrong password doesn't match  
36         boolean wrongMatch = passwordEncoder.matches("wrongPassword  
", hashedPassword);  
37         assertFalse(wrongMatch, "Password encoder phải reject  
password sai");  
38     }  
39  
40  
41     // 3. CORS CONFIGURATION TEST  
42  
43     @Test  
44     @DisplayName("3. CORS Configuration - CORS headers đ ọc cấu hì  
nh đúng")  
45     void testCorsConfiguration() throws Exception {  
46         // Arrange  
47         String origin = "http://localhost:5173";  
48  
49         // Act - Preflight request (OPTIONS)  
50         mockMvc.perform(options("/api/products")  
51             .header("Origin", origin)  
52             .header("Access-Control-Request-Method", "  
POST")  
53             .header("Access-Control-Request-Headers", "  
Content-Type, Authorization"))  
54  
55         // Assert  
56         .andExpect(status().isOk())  
57         .andExpect(header().exists("Access-Control-Allow-
```

```
Origin"))
58         .andExpect(header().string("Access-Control-Allow-
Methods",
59             org.hamcrest.Matchers.containsString("POST"
)))
60         .andExpect(header().exists("Access-Control-Allow-
Headers"));
61     }
62
63     // 4. SECURITY HEADERS TEST
64
65     @Test
66     @DisplayName("4. Security Headers - Response có các security
headers cần thiết")
67     void testSecurityHeaders() throws Exception {
68         // Arrange & Act
69         MvcResult result = mockMvc.perform(get("/api/auth/login")
70             .contentType(MediaType.APPLICATION_JSON))
71             .andReturn();
72
73         // Assert - Kiểm tra các security headers quan trọng
74         String xContentTypeOptions = result.getResponse().getHeader
75         ("X-Content-Type-Options");
76         String xFrameOptions = result.getResponse().getHeader("X-
Frame-Options");
77         String xXssProtection = result.getResponse().getHeader("X-
XSS-Protection");
78
79         // Spring Security tự động thêm các headers này
80         assertNotNull(xContentTypeOptions, "X-Content-Type-Options
header phải có");
81         assertEquals("nosniff", xContentTypeOptions,
82             "X-Content-Type-Options phải là 'nosniff'");
83
84         assertNotNull(xFrameOptions, "X-Frame-Options header phải c
85         ó");
86         assertTrue(xFrameOptions.equals("DENY") || xFrameOptions.
equals("SAMEORIGIN"),
87             "X-Frame-Options phải là DENY hoặc SAMEORIGIN");
88     }
89 }
```

Listing 34: SecurityBestPracticesTest.java