

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 Lãng 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             const result = validateUsername("");
5             expect(result.isValid).toBe(false);
6             expect(result.error).toBe("Username không được để trống");
7         });
8         .....
9     });
10    test("nên trả về không hợp lệ khi username chỉ có khoảng trắng", () => {
11        const result = validateUsername("   ");
12        expect(result.isValid).toBe(false);
13        expect(result.error).toBe("Username không được để trống");
14    });
15 });
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 ký tự)", () => {
21         const result = validateUsername("ab");
22         expect(result.isValid).toBe(false);
23         expect(result.error).toBe("Username phải ít nhất 3 ký tự");
24     });
25     .....
26
27     test("nên hợp lệ với đúng 50 ký tự", () => {
28         const result = validateUsername(
29             "abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz"
30         );
31         expect(result.isValid).toBe(true);
32         expect(result.error).toBeNull();
33     });
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         const result = validateUsername("user name");
41         expect(result.isValid).toBe(false);
```



```
42     expect(result.error).toBe(
43       "Username chí chứa chữ, số, dấu chấm (.), gạch d ói (_), và
44       gạch ngang (-)"
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 ói, chấm và gạch ngang",
58     () => {
59       const result = validateUsername("user_n.ame-");
60       expect(result.isValid).toBe(true);
61       expect(result.error).toBeNull();
62     });
63 });
64 }
```

Listing 1: validateUsername

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

```
1 describe("validatePassword()", () => {
2   describe("Test 1: Password rỗng", () => {
3     test("nên trả về không hợp lệ khi password là chuỗi rỗng", () => {
4       const result = validatePassword("");
5       expect(result.isValid).toBe(false);
6       expect(result.error).toBe("Mật khẩu không đ ọc đ é trống");
7     });
8     ....
9
10    test("nên trả về không hợp lệ khi password là undefined", () => {
11      const result = validatePassword(undefined);
12      expect(result.isValid).toBe(false);
13      expect(result.error).toBe("Mật khẩu không đ ọc đ é trống");
14    });
15  });
16
17
18  describe("Test 2: Password quá ngắn/quá dài", () => {
19    test("nên trả về không hợp lệ khi password quá ngắn (ít h n 6 k
20         ý t u)", () => {
21      const result = validatePassword("pass1");
22      expect(result.isValid).toBe(false);
23      expect(result.error).toBe("Mật khẩu phải ít nhát 6 ký t u");
```



```
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 nhưng 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
58 describe("Test 4: Password hợp lệ", () => {
59     test("nên hợp lệ khi có chữ cái và số", () => {
60         const result = validatePassword("password123");
61         expect(result.isValid).toBe(true);
62         expect(result.error).toBeNull();
63     });
64
65     ....
66
67     test("nên hợp lệ khi chỉ có một số và các chữ cái khác", () =>
68     {
69         const result = validatePassword("password1");
70         expect(result.isValid).toBe(true);
71         expect(result.error).toBeNull();
72     });
73 });
74 });
75 }
```

Listing 2: validatePassword



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



### 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    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,
27             mockUserRepository, mockPasswordEncoder, mockValidator);
28
29         ValidatorFactory factory = Validation.
30         buildDefaultValidatorFactory();
31         realValidator = factory.getValidator();
32     }
33
34     // A) TEST METHOD authenticate()
35
36     @Test
37     @DisplayName("TC1: Login thành công với credential hợp lệ")
38     void testLoginSuccess() {
39         // Arrange
40         LoginRequest loginRequest = new LoginRequest("testUser", "Test123");
41
42         // Mock validator trả về không có lỗi
43         when(mockValidator.validate(loginRequest))
44             .thenReturn(Set.of());
45
46         // Mock các dependencies
47         when(mockUserRepository.findByUserName(loginRequest.
48             getUserName()))
49             .thenReturn(Optional.of(mockUser));
50         when(mockPasswordEncoder.matches("Test123", "hashedPassword123"))
51             .thenReturn(true);
52         when(mockJwtService.generateToken(mockUser))
53             .thenReturn("fake-jwt-token");
54
55         // Act
56         LoginResponse loginResponse = authService.authenticate(
57             loginRequest);
58
59         // Assert
60         assertTrue(loginResponse.isSuccess(), "Login phải thành công");
61         assertEquals("Login thành công", loginResponse.getMessage());
62         assertNotNull(loginResponse.getToken(), "Token không được null");
63         assertEquals("fake-jwt-token", loginResponse.getToken());
64
65         // Verify UserDto
66         assertNotNull(loginResponse.getUser(), "UserDto không được null");
67         assertEquals("testuser", loginResponse.getUser().getUsername());
```



```
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
72         testUser");
73         verify(mockPasswordEncoder, times(1)).matches("Test123", "74
75         hashedPassword123");
76         verify(mockJwtService, times(1)).generateToken(mockUser);
77     }
78
79     @Test
80     @DisplayName("TC2: Login thất bại khi username không tồn tại")
81     void testLoginFailure_UserNotFound() {
82         // Arrange
83         LoginRequest loginRequest = new LoginRequest("84
85         nonExistentUser", "Test123");
86
87         // Mock validator trả về không có lỗi
88         when(mockValidator.validate(loginRequest))
89             .thenReturn(Set.of());
90
91         // Mock repository trả về empty
92         when(mockUserRepository.findByUserName("nonExistentUser"))
93             .thenReturn(Optional.empty());
94
95         // Act
96         LoginResponse loginResponse = authService.authenticate(
97             loginRequest);
98
99         // Assert
100        assertFalse(loginResponse.isSuccess(), "Login phải thất bại");
101        assertEquals("Login thất bại với user name không tồn tại",
102            loginResponse.getMessage());
103        assertNull(loginResponse.getToken(), "Token phải là null");
104        assertNull(loginResponse.getUser(), "UserDto phải là null
105        khi thất bại");
106
107        // Verify interactions
108        verify(mockValidator, times(1)).validate(loginRequest);
109        verify(mockUserRepository, times(1)).findByUserName("110
111        nonExistentUser");
112        verify(mockPasswordEncoder, never()).matches(anyString(),
113            anyString());
114        verify(mockJwtService, never()).generateToken(any(User.
115            class));
116    }
117
118    @Test
119    @DisplayName("TC3: Login thất bại khi password sai")
120    void testLoginFailure_WrongPassword() {
121        // Arrange
122        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(
121         loginRequest);
122
123     // Assert
124     assertFalse(loginResponse.isSuccess(), "Login phái thát bại");
125     assertEquals("Login với password sai", loginResponse.getMessage());
126     assertNull(loginResponse.getToken(), "Token phái là null");
127     assertNull(loginResponse.getUser(), "UserDto phái là null
128     khi thát bại");
129
130     // Verify interactions
131     verify(mockValidator, times(1)).validate(loginRequest);
132     verify(mockUserRepository, times(1)).findByUserName("testUser");
133     verify(mockPasswordEncoder, times(1)).matches("WrongPassword123", "hashedPassword123");
134     verify(mockJwtService, never()).generateToken(any(User.class));
135 }
136
137 @Test
138     @DisplayName("TC4: Cá username và password đều null - Validator
139     phái trả về nhiều lỗi")
140     void validate_withBothNull_shouldReturnMultipleErrors() {
141         // Arrange
142         var request = new LoginRequest(null, null);
143
144         Set<ConstraintViolation<LoginRequest>> violations =
145         realValidator.validate(request);
146
147         // Act
148
149         // Assert
150         assertEquals(2, violations.size(), "Phái có đúng 2 lỗi");
151
152         boolean isHasUserNameError = violations.stream()
153             .anyMatch(v -> v.getMessage().contains("Username kh
154             ông đ ọc đé tróng"));
155         assertTrue(isHasUserNameError);
156
157         boolean isHasPasswordError = violations.stream()
158             .anyMatch(v -> v.getMessage().contains("Password kh
159             ông đ ọc đé tróng"));
```

```
    ông đ ọc d é tr óng"));  
    assertTrue(isHasPasswordError);  
}  
}  
}
```

Listing 3: AuthServiceTest.java

- b) Test validation methods riêng lẻ (1 điểm)



```
34     // Mock validator trả về violation
35     ConstraintViolation<LoginRequest> violation = mock(
36     ConstraintViolation.class);
37     when(violation.getMessage()).thenReturn("Password không đ ợc
38     c ốp 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 phái thất bại
48     do validation");
49     assertEquals("Password không đ ợc c ốp 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()).findByName(
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("tù 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(), "Phái 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, "Phái 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
94     //Act
95     Set<ConstraintViolation<LoginRequest>> violations =
realValidator.validate(req);
96
97     //Assert
98     assertFalse(violations.isEmpty(), "Phái có lỗi username có
chứa ký tự đặc biệt");
99     boolean isHasError = violations.stream()
100         .anyMatch(v -> v.getMessage().contains("Username ch
í chứa chữ, số, và ký tự (-, ., _)"));
101    assertTrue(isHasError, "Phái có thông báo lỗi username có k
ý tự đặc biệt");
102
103 }
104
105 @Test
106 @DisplayName("TC10: Login thất bại với validation error -
password thiếu chữ cái")
107 void testLoginFailure_ValidationError_PasswordMissingLetter() {
108     //Arrange
109     LoginRequest req = new LoginRequest("testUser", "123456");
110
111     //Act
112     Set<ConstraintViolation<LoginRequest>> violations =
realValidator.validate(req);
113
114     //Assert
115     assertFalse(violations.isEmpty(), "Phái có lỗi password thi
ếu chữ cái");
116     boolean isHasError = violations.stream()
117         .anyMatch(v -> v.getMessage().contains("Password ph
ái chứa ít nhất 1 chữ cái"));
118     assertTrue(isHasError, "Phái có thông báo lỗi password thié
u chữ cái");
}
```



```
119
120
121     @Test
122     @DisplayName("TC11: Login thất bại với validation error -"
123     "password thiếu só")
124     void testLoginFailure_ValidationError_PasswordMissingNumber() {
125         // Arrange
126         LoginRequest req = new LoginRequest("testUser", "
127         testpassword");
128
129         // Act
130         Set<ConstraintViolation<LoginRequest>> violations =
131         realValidator.validate(req);
132
133         // Assert
134         assertFalse(violations.isEmpty(), "Phải có lỗi password thi
135         éu só");
136         boolean isHasError = violations.stream()
137             .anyMatch(v -> v.getMessage().contains("Password ph
138             ái chứa ít nhất 1 chữ số"));
139         assertTrue(isHasError, "Phải có thông báo lỗi password thié
140         u só");
141     }
```

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 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 thử biên)', () => {
21   test('trả về lỗi khi giá là undefined', () => {
22     const product = { name: 'Product', price: undefined, quantity: 10, description: 'Valid description', category: 'Electronics' };
23     const result = validateProduct(product);
24
25     expect(result.isValid).toBe(false);
26     expect(result.errors.price).toBe('Giá sản phẩm không được để trống');
27   });
28
29   .....
30
31   test('hợp lệ khi giá hợp lệ (trong hợp điều kiện)', () => {
32     const product = { name: 'Product', price: 50000, quantity: 10, description: 'Valid description', category: 'Electronics' };
33     const result = validateProduct(product);
34
35     expect(result.errors.price).toBeUndefined();
36   });
37 });
38
39 describe('validateProduct - Kiểm tra số lượng', () => {
40   test('trả về lỗi khi số lượng là undefined', () => {
41     const product = { name: 'Product', price: 100, quantity: undefined, description: 'Valid description', category: 'Electronics' };
42     const result = validateProduct(product);
43
44     expect(result.isValid).toBe(false);
45     expect(result.errors.quantity).toBe('Số lượng không được để trống');
46   });
47
48   test('trả về lỗi khi số lượng là null', () => {
49     const product = { name: 'Product', price: 100, quantity: null, description: 'Valid description', category: 'Electronics' };
50     const result = validateProduct(product);
51
52     expect(result.isValid).toBe(false);
53     expect(result.errors.quantity).toBe('Số lượng không được để trống');
54   });
55
56   test('trả về lỗi khi số lượng là chuỗi rỗng', () => {
57     const product = { name: 'Product', price: 100, quantity: '', description: 'Valid description', category: 'Electronics' };
58     const result = validateProduct(product);
59   });

```



```
60
61     expect(result.isValid).toBe(false);
62     expect(result.errors.quantity).toBe('Số lượng không đọc được trong');
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 đọc được trong');
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 đọc được trong');
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
118 describe('validateProduct - Kiểm tra tích hợp', () => {
119     test('trả về isValid true khi tất cả trường hợp hợp lệ', () => {
120         const product = {
121             name: 'Valid Product',
122             price: 100,
123             quantity: 10,
124             description: 'This is a valid description',
125             category: 'Electronics'
126         };
127         const result = validateProduct(product);
128
129         expect(result.isValid).toBe(true);
130         expect(result.errors).toEqual({});
131     });
132
133     test('trả về nhiều lỗi khi nhiều trường không hợp lệ', () => {
134         const product = {
135             name: '',
136             price: -100,
137             quantity: 'abc',
138             description: 'Short',
139             category: ''
140         };
141         const result = validateProduct(product);
142
143         expect(result.isValid).toBe(false);
144         expect(result.errors.name).toBeDefined();
145         expect(result.errors.price).toBeDefined();
146         expect(result.errors.quantity).toBeDefined();
147         expect(result.errors.description).toBeDefined();
148         expect(result.errors.category).toBeDefined();
149     });
150 })
```

Listing 5: validateProduct()

- b) Viết tests cho Product form component (1 điểm)
- c) Coverage  $\geq 90$  (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);
16    }
```



```
17 // A) TEST CREATE - Bao gồm Boundary, Edge Cases và Negative
18 // 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", "Gaming
33             laptop", 10, "Laptop", 15000.0);
34
35             when(mockValidator.validate(request)).thenReturn(Set.of());
36             when(productRepository.save(any(Product.class))).thenReturn(savedProduct);
37
38             // Act
39             ProductDto result = productService.createProduct(
40             request);
41
42             // Assert
43             assertNotNull(result);
44             assertEquals("Laptop", result.getProductName());
45             assertEquals(15000.0, result.getPrice());
46             verify(mockValidator, times(1)).validate(request);
47             verify(productRepository, times(1)).save(any(Product.
48             class));
49         }
50     }
51
52     @Nested
53     @DisplayName("Test getProductId() method")
54     class GetProductTests {
55
56         @Test
57         @DisplayName("TC14: Lấy sản phẩm thành công theo ID")
58         void testGetProductById_Success() {
59             Product product = new Product(1L, "Electronics", "Gaming
60             laptop", 10, "Laptop", 15000.0);
61             when(productRepository.findById(1L)).thenReturn(
62             Optional.of(product));
63
64             ProductDto result = productService.getProductById(1L);
65
66             assertNotNull(result);
67             assertEquals(1L, result.getId());
68             assertEquals("Laptop", result.getProductName());
69             verify(productRepository, times(1)).findById(1L);
```



```
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(
70             Optional.empty());
71
72         NoSuchElementException exception = assertThrows(
73             NoSuchElementException.class,
74             () -> productService.getProductById(999L)
75         );
76
77         assertTrue(exception.getMessage().contains("Product not
78         found with id: 999"));
79         verify(productRepository, times(1)).findById(999L);
80     }
81 }
82
83 @Nested
84     @DisplayName("Test updateProduct() method")
85     class UpdateProductTests {
86
87         @Test
88         @DisplayName("TC16: Cập nhật sản phẩm thành công")
89         void testUpdateProduct_Success() {
90             UpdateProductRequest request = new UpdateProductRequest
91                 ("Updated Laptop", 20000.0, "Updated description", 15, "Electronics");
92             Product existingProduct = new Product(1L, "Electronics",
93                 "Old description", 10, "Laptop", 15000.0);
94             Product updatedProduct = new Product(1L, "Electronics",
95                 "Updated description", 15, "Updated Laptop", 20000.0);
96
97             when(mockValidator.validate(request)).thenReturn(Set.of());
98             when(productRepository.findById(1L)).thenReturn(
99                 Optional.of(existingProduct));
100            when(productRepository.save(any(Product.class))).thenReturn(updatedProduct);
101
102            ProductDto result = productService.updateProduct(1L,
103                request);
104
105            assertNotNull(result);
106            assertEquals("Updated Laptop", result.getProductName());
107        }
108    }
109
110 // D) TEST DELETE
111
112 @Nested
113     @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", "Gaming laptop", 10, "Laptop", 15000.0);
114             when(productRepository.findById(1L)).thenReturn(
115                 Optional.of(product));
116             doNothing().when(productRepository).deleteById(1L);
117
118             productService.deleteProduct(1L);
119
119             verify(productRepository, times(1)).findById(1L);
120             verify(productRepository, times(1)).deleteById(1L);
121         }
122
123         @Test
124         @DisplayName("TC21: Xóa sản phẩm thất bại - ID không tồn tại")
125         void testDeleteProduct_NotFound() {
126             when(productRepository.findById(999L)).thenReturn(
127                 Optional.empty());
128
129             NoSuchElementException exception = assertThrows(
130                 NoSuchElementException.class,
131                 () -> productService.deleteProduct(999L)
132             );
133
134             assertTrue(exception.getMessage().contains("Product not
135             found with id: 999"));
136         }
137
138 // E) TEST GET ALL (PAGINATION)
139
140         @Test
141         @DisplayName("TC22: Lấy danh sách sản phẩm với pagination")
142         void test GetAll_Success() {
143             Pageable pageable = PageRequest.of(0, 10);
144             List<Product> products = Arrays.asList(
145                 new Product(1L, "Electronics", "Product 1", 10,
146                 "Laptop", 15000.0),
147                 new Product(2L, "Books", "Product 2", 20, "Book",
148                 50.0));
149             Page<Product> productPage = new PageImpl<>(products,
150             pageable, products.size());
151
152             when(productRepository.findAll(pageable)).thenReturn(
153                 productPage);
154
155             Page<ProductDto> result = productService.getAll(
156             pageable);
157
158             assertNotNull(result);
159         }
160     }
161 }
```



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

Listing 6: ProductServiceTest.java

- b) Coverage  $\geq 85$  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 thé nháp 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 thé nháp 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 Nhập/i });
58
59     await user.type(usernameInput, 'testuser');
60     await user.type(passwordInput, 'password123');
61
62     // Button should exist and be clickable
63     expect(submitButton).toBeInTheDocument();
64     expect(submitButton).toBeEnabled();
65   });
66 });
67
```

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 - admin', async () => {
4      const user = userEvent.setup();
5      authService.login.mockResolvedValue({
6        success: true,
7        message: 'Login thành công',
8        user: { userName: 'admin', email: 'admin@example.com' }
9      });
10
11    render(
12      <MemoryRouter>
13        <LoginForm />
14      </MemoryRouter>
15    );
16
17    const usernameInput = screen.getByPlaceholderText(
18      'your_username/i');
19    const passwordInput = screen.getByPlaceholderText(
20      '/i');
21    const submitButton = screen.getByRole('button', { name: /Đăng Nhập/i });
22
23    await user.type(usernameInput, 'admin');
24    await user.type(passwordInput, 'admin123');
25    await user.click(submitButton);
26  });
27
28  expect(authService.login).toHaveBeenCalledWith({
29    email: 'admin@example.com',
30    password: 'admin123'
31  });
32});
```



```
25     // Verify API call
26     await waitFor(() => {
27       expect(authService.login).toHaveBeenCalledWith('admin', ,
28         admin123);
29     });
30   );
31
32   test('nên hiện thị success message khi API trả về success -',
33     testuser, async () => {
34     const user = userEvent.setup();
35     authService.login.mockResolvedValue({
36       success: true,
37       message: 'Login thành công',
38       user: { userName: 'testuser', email: 'testuser@example.com' }
39     });
40
41     render(
42       <MemoryRouter>
43         <LoginForm />
44       </MemoryRouter>
45     );
46
47     const usernameInput = screen.getByPlaceholderText(/your_username/i);
48     const passwordInput = screen.getByPlaceholderText(/ /i);
49     const submitButton = screen.getByRole('button', { name: /Đăng Nhập/i });
50
51     await user.type(usernameInput, 'testuser');
52     await user.type(passwordInput, 'test1234');
53     await user.click(submitButton);
54
55     // Verify API called
56     await waitFor(() => {
57       expect(authService.login).toHaveBeenCalledWith('testuser',
58         'test1234');
59     });
60   );
61 );
```

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 hiện thị error khi username trống', async () => {
4      const user = userEvent.setup();
5      render(
6        <MemoryRouter>
7          <LoginForm />
8        </MemoryRouter>
9      );
10  );
```

```
10     const submitButton = screen.getByRole('button', { name: /Đăng Nhập/i });
11     await user.click(submitButton);
12
13     await waitFor(() => {
14       expect(screen.getByText(/Username không được để trống/i)).toBeInTheDocument();
15     });
16   });
17
18
19   test('nên hiện 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 nhất 3 ký tự/i);
37       expect(errorMessage).toBeInTheDocument();
38     });
39   });
40
41   test('nên hiện 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
62         credentials|Error/i);
63         expect(errorMessage).toBeInTheDocument();
64     });
65
66 test('nên hiện thị success message khi login thành công', async
67 () => {
68     const user = userEvent.setup();
69     authService.login.mockResolvedValue({
70         success: true,
71         message: 'Login thành công',
72         user: { userName: 'admin', email: 'admin@example.com' }
73     });
74
75     render(
76         <MemoryRouter>
77             <LoginForm />
78         </MemoryRouter>
79     );
80
81     const usernameInput = screen.getByPlaceholderText(/your_username/i);
82     const passwordInput = screen.getByPlaceholderText(/Nhập/i);
83     const submitButton = screen.getByRole('button', { name: /Đăng
84     nhập/i });
85
86     await user.type(usernameInput, 'admin');
87     await user.type(passwordInput, 'admin123');
88     await user.click(submitButton);
89
90     // Verify API called
91     await waitFor(() => {
92         expect(authService.login).toHaveBeenCalled();
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         ASSIGNEABLE_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", "Test123");
28            UserDto userDto = new UserDto("testuser", "testuser@example.com");
29            LoginResponse mockResponse = new LoginResponse(
30                true,
31                "Login thành công",
32                "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.token123",
33                userDto
34            );
35
36            // Mock service trả về response thành công
37            when(authService.authenticate(any(LoginRequest.class)))
38                .thenReturn(mockResponse);
39
40            // Act & Assert: Thực hiện request và verify kết quả
41            mockMvc.perform(post("/api/auth/login")
42                .contentType(MediaType.APPLICATION_JSON)
43                .content(objectMapper.writeValueAsString(
44                    request)))
45                .andExpect(status().isOk()) // Expect 200 OK
46                .andExpect(jsonPath("$.success").value(true))
47                .andExpect(jsonPath("$.message").value("Login
48                    thành công"))
49                .andExpect(jsonPath("$.token").exists())
50                .andExpect(jsonPath("$.token").value("eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.token123"))
51                .andExpect(jsonPath("$.user").exists())
52                .andExpect(jsonPath("$.user.userName").value("
```

```
49     testuser"))
50             .andExpect(jsonPath("$.user.email").value("testuser@example.com"));
51         }
52
53     @Test
54     @DisplayName("2. Login thất bại - Username không tồn tại")
55     void testLoginFailure_UserNotFound() throws Exception {
56         // Arrange: User không tồn tại trong database
57         LoginRequest request = new LoginRequest("nonexistuser",
58                                         "Test123");
59         LoginResponse mockResponse = new LoginResponse(
60             false,
61             "Login thất bại với user name không tồn tại"
62         );
63
64         when(authService.authenticate(any(LoginRequest.class)))
65             .thenReturn(mockResponse);
66
67         // Act & Assert: Expect 401 Unauthorized
68         mockMvc.perform(post("/api/auth/login")
69                         .contentType(MediaType.APPLICATION_JSON)
70                         .content(objectMapper.writeValueAsString(
71                             request)))
72                         .andExpect(status().isUnauthorized()) // Expect
73                         401
74                         .andExpect(jsonPath("$.success").value(false))
75                         .andExpect(jsonPath("$.message").value("Login
76 thất bại với user name không tồn tại"))
77                         .andExpect(jsonPath("$.token").doesNotExist())
78                         .andExpect(jsonPath("$.user").doesNotExist());
79     }
80 }
```

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 dày dứ các field khi
7          login thành công")
8          void testSuccessResponseStructure() throws Exception {
9              // Arrange
10             LoginRequest request = new LoginRequest("testuser", "Test123");
11             UserDto userDto = new UserDto("testuser", "testuser@example.com");
12             LoginResponse mockResponse = new LoginResponse(
13                 true,
14                 "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
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
52         // khi thất bại
53         mockMvc.perform(post("/api/auth/login")
54                         .contentType(MediaType.APPLICATION_JSON)
55                         .content(objectMapper.writeValueAsString(
56                             request)))
57                         .andExpect(status().isUnauthorized())
58                         .andExpect(jsonPath("$.success").value(false))
59                         .andExpect(jsonPath("$.message").exists())
60                         .andExpect(jsonPath("$.token").doesNotExist());
61
62         // Token không tồn tại khi fail
63             .andExpect(jsonPath("$.user").doesNotExist());
64         // User cũng không tồn tại khi fail
65     }
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
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
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
637
638
639
639
640
641
642
643
644
645
645
646
647
647
648
649
649
650
651
652
653
654
655
655
656
657
657
658
659
659
660
661
662
663
664
664
665
666
666
667
667
668
668
669
669
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
154
```



```
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(
76                 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(
98                 request)))
98             .andExpect(status().isOk())
99             .andExpect(content().contentType(MediaType.
100 APPLICATION_JSON));
101     }
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") // Giá lặp 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
2    API", () => {
3      test("nên render ProductManagement component thành công", () =>
4        {
5          render(<ProductManagement />);
6
7          // Verify component renders
8          const heading = screen.getByText("Quán Lý Sản Phẩm");
9          expect(heading).toBeInTheDocument();
10     });
11
12     test("nên có category filter dropdown", () => {
13       render(<ProductManagement />);
14
15       const filterElements =
16         screen.queryAllByRole("button").length > 0 ||
17         screen.queryAllByRole("combobox").length > 0 ||
18         screen.queryByText(/Tất cả|All|category|Category/i);
19       expect(filterElements).toBeTruthy();
20     });
21
22     test("nên filter products theo category", async () => {
23       const user = userEvent.setup();
24       render(<ProductManagement />);
25
26       // Tìm filter button hoặc select
27       const filterButtons = screen.queryAllByRole("button");
28       expect(filterButtons.length > 0).toBe(true);
29     });
30
31     test("nên có action buttons (Edit, Delete, View)", () => {
32       render(<ProductManagement />);
33
34       // Verify edit/delete buttons exist or can be added
35       const buttons = screen.queryAllByRole("button");
36       expect(buttons.length > 0).toBe(true);
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 óc 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 });
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
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
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
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
756
756
757
758
758
759
759
760
761
762
763
764
765
765
766
767
767
768
768
769
769
770
771
772
773
774
775
775
776
777
777
778
778
779
779
780
781
782
783
784
784
785
785
786
786
787
787
788
788
789
789
790
791
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
801
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
159
```



```
1 describe("Test 3: ProductDetail", () => {
2     test("nên display product info: name, price, quantity, category",
3         async () => {
4             render(<ProductManagement />);
5
6             // Check if table or list displays product info columns
7             await waitFor(() => {
8                 const headers = screen.queryAllByText(
9                     /tên|name|giá|price|số l ượng|quantity|loại|category/i
10                );
11                expect(headers.length > 0).toBe(true);
12            });
13        });
14
15    test("nên display product details khi click view/expand", async
16        () => {
17        const user = userEvent.setup();
18        render(<ProductManagement />);
19
20        // Look for view buttons or expandable rows
21        const viewButtons = screen.queryAllByRole("button");
22        expect(viewButtons.length > 0).toBe(true);
23    });
24});
```

Listing 15: ProductDetail component

#### 4.2.2 Backend API Integration (5 điểm)

Test các API endpoints của Product:

- Test POST /api/products (Create) (1 điểm)
- Test GET /api/products (Read all) (1 điểm)
- Test GET /api/products/id (Read one) (1 điểm)
- Test PUT /api/products/id (Update) (1 điểm)
- 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("Laptop", 15000000.0, "Gaming laptop", 10, "Electronics");
23      ProductDto responseDto = new ProductDto(1L, "Electronics", 15000000.0, "Laptop", "Gaming laptop", 10);
24
25      when(productService.createProduct(any(CreateProductRequest.class)))
26          .thenReturn(responseDto);
27
28      // Act & Assert
29      mockMvc.perform(post("/api/products")
30                      .contentType(MediaType.APPLICATION_JSON)
31                      .content(objectMapper.writeValueAsString(requestDto))
32                  )
33                  .andExpect(status().isCreated()) // 201 Created
34                  .andExpect(jsonPath("$.id").value(1))
35                  .andExpect(jsonPath("$.productName").value("Laptop"))
36
37                  .andExpect(jsonPath("$.price").value(15000000.0))
38                  .andExpect(jsonPath("$.quantity").value(10))
39                  .andExpect(jsonPath("$.category").value("Electronics"));
40
41      verify(productService, times(1)).createProduct(any(
42 CreateProductRequest.class));
43  }
44
45 /**
46 * B) Test GET /api/products - Read All (1 điểm)
47 * Test lấy danh sách tất cả products
48 */
49 @Test
50 @DisplayName("2. Lấy danh sách products thành công")
51 void test GetAllProducts_Success() throws Exception {
52     // Arrange
53     List<ProductDto> products = Arrays.asList(
54         new ProductDto(1L, "Electronics", 15000000.0, "Laptop",
55         "Gaming laptop", 10),
56         new ProductDto(2L, "Electronics", 200000.0, "Mouse",
57         "Wireless mouse", 50)
58     );
59     Page<ProductDto> page = new PageImpl<>(products, PageRequest.of(0, 10), products.size());
60
61     when(productService.getAll(any()))
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
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
```

```
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").
64             value("Laptop"))
65         .andExpect(jsonPath("$.content[1].productName").
66             value("Mouse"))
67         .andExpect(jsonPath("$.totalElements").value(2));
68
69
70
71     @Test
72     @DisplayName("3. Lấy product theo ID thành công")
73     void testGetProductById_Success() throws Exception {
74         // Arrange
75         ProductDto product = new ProductDto(1L, "Electronics",
76             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",
100             16000000.0, "Laptop Updated", "New description", 15);
101
102         when(productService.updateProduct(eq(1L), any(
103             UpdateProductRequest.class)))
104             .thenReturn(responseDto);
105
106         // 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
112             Updated"))
113             .andExpect(jsonPath("$.price").value(16000000.0))
114             .andExpect(jsonPath("$.quantity").value(15));
115
116
117
118     @Test
119     @DisplayName("5. Xóa product thành công - 204 No Content")
120     void testDeleteProduct_Success() throws Exception {
121         // Arrange
122         doNothing().when(productService).deleteProduct(1L);
123
124         // Act & Assert
125         mockMvc.perform(delete("/api/products/1"))
126             .andExpect(status().isNoContent()); // 204 No
127             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:

- Mock authService.loginUser() (1 điểm)

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

Listing 17: Mock authService.loginUser()

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

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



```
28      // Verify success message appears
29      await waitFor(() => {
30          const successMessage = screen.queryByText(/Đăng nhập thành
31          công/i);
32          expect(successMessage).toBeInTheDocument();
33      }, { timeout: 2000 });
34  );
35
36  test('nên handle mocked failed response - Invalid credentials',
37  async () => {
38      const user = userEvent.setup();
39      authService.login.mockRejectedValue(new Error('Invalid
40      credentials'));
41
42      render(
43          <MemoryRouter>
44              <LoginForm />
45          </MemoryRouter>
46  );
47
48      const usernameInput = screen.getByPlaceholderText(/your_username/i);
49      const passwordInput = screen.getByPlaceholderText(/ /i);
50      const submitButton = screen.getByRole('button', { name: /Đăng Nhập/i });
51
52      await user.type(usernameInput, 'testuser');
53      await user.type(passwordInput, 'wrongpassword1');
54      await user.click(submitButton);
55
56      // Verify error message appears
57      await waitFor(() => {
58          const errorMessage = screen.getText(/Invalid credentials/i);
59          expect(errorMessage).toBeInTheDocument();
60      });
61  });
62
```

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          () => {
5              const user = userEvent.setup();
6              authService.login.mockResolvedValue({ success: true });
7
8              render(
9                  <MemoryRouter>
10                     <LoginForm />
```

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



```
58     </MemoryRouter>
59 );
60
61     const submitButton = screen.getByRole('button', { name: /Đăng
62         Nhập/i });
63
64     // Submit form trống
65     await user.click(submitButton);
66
67     // Mock không nên đọc gọi
68     expect(authService.login).not.toHaveBeenCalled();
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
11   @Autowired
12   private MockMvc mockMvc;
13
14
15   /**
16    * a) Mock AuthService với @MockitoBean (1 điểm)
17    * @MockitoBean tạo mock object và inject vào Spring context
18    */
19   @MockitoBean
20   private AuthService authService;
21
22   @Autowired
23   ObjectMapper objectMapper;
24
25   @Test
26   @DisplayName("1. Mock: Controller với mocked service - Login thà
27   nh công")
28   void testLoginWithMockedService_Success() throws Exception {
```



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



```
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 g ọi đúng 1 l ần v ới 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, "Success",
113         "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 g ọi đúng 1 l ần
```



```
119         verify(authService, times(1)).authenticate(any(LoginRequest.  
class));  
120  
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     void testVerify_MockNotCalledWhenValidationFails() throws  
Exception {  
128         // Arrange: Không cần mock vì validation sẽ fail trước khi g  
ọi service  
129  
130         // Act: Gửi request với username trống (validation fail)  
mockMvc.perform(post("/api/auth/login")  
                .contentType(MediaType.APPLICATION_JSON)  
                .content("{\"userName\":\"\", \"password\":\"Pass123  
\"}")  
                .andExpect(status().isBadRequest());  
131  
132         // c) Verify: authService.authenticate() KHÔNG đọc gọi vì  
validation fail  
133         verify(authService, never()).authenticate(any(LoginRequest.  
class));  
134         verifyNoInteractions(authService);  
135     }  
136  
137 }  
138  
139 }  
140  
141 }  
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         },  
10        {  
11            id: 2,  
12            name: "Product 2",  
13            price: 1000000,  
14            quantity: 100,  
15            category: "Electronics",  
16            description: "A high-end electronic device.",  
17            image: "https://example.com/images/product2.jpg",  
18        },  
19    ]  
20  
21    // Test Case 1: Get All Products  
22    it('should return all products', () => {  
23        const response = {  
24            status: 200,  
25            data: mockProductList  
26        };  
27  
28        // Mock the productService.getAll() method to return the mock data  
29        productService.getAll = jest.fn(() => Promise.resolve(response));  
30  
31        // Call the component's method to get all products  
32        const result = component.getAllProducts();  
33  
34        // Expect the result to be equal to the mock data  
35        expect(result).toEqual(mockProductList);  
36    });  
37  
38    // Test Case 2: Get Single Product by ID  
39    it('should return product by ID', () => {  
40        const response = {  
41            status: 200,  
42            data: mockProductList[1]  
43        };  
44  
45        // Mock the productService.getProductById() method to return the mock data  
46        productService.getProductById = jest.fn(id => Promise.resolve(response));  
47  
48        // Call the component's method to get product by ID  
49        const result = component.getProductById(2);  
50  
51        // Expect the result to be equal to the mock data  
52        expect(result).toEqual(mockProductList[1]);  
53    });  
54  
55    // Test Case 3: Create New Product  
56    it('should create new product', () => {  
57        const newProduct = {  
58            name: "Product 3",  
59            price: 1500000,  
60            quantity: 50,  
61            category: "Electronics",  
62            description: "A new electronic device.",  
63            image: "https://example.com/images/product3.jpg",  
64        };  
65  
66        // Mock the productService.create() method to return the created product  
67        productService.create = jest.fn(product => Promise.resolve({  
68            ...product,  
69            id: 3  
70        }));  
71  
72        // Call the component's method to create a new product  
73        const result = component.createProduct(newProduct);  
74  
75        // Expect the result to be equal to the created product  
76        expect(result).toEqual({  
77            id: 3,  
78            name: "Product 3",  
79            price: 1500000,  
80            quantity: 50,  
81            category: "Electronics",  
82            description: "A new electronic device.",  
83            image: "https://example.com/images/product3.jpg",  
84        });  
85    });  
86  
87    // Test Case 4: Update Existing Product  
88    it('should update existing product', () => {  
89        const updatedProduct = {  
90            id: 1,  
91            name: "Updated Product 1",  
92            price: 1200000,  
93            quantity: 80,  
94            category: "Electronics",  
95            description: "An updated electronic device.",  
96            image: "https://example.com/images/product1.jpg",  
97        };  
98  
99        // Mock the productService.update() method to return the updated product  
100       productService.update = jest.fn(id, product => Promise.resolve({  
101           ...product,  
102           id: 1  
103       }));  
104  
105       // Call the component's method to update a product  
106       const result = component.updateProduct(updatedProduct);  
107  
108       // Expect the result to be equal to the updated product  
109       expect(result).toEqual({  
110           id: 1,  
111           name: "Updated Product 1",  
112           price: 1200000,  
113           quantity: 80,  
114           category: "Electronics",  
115           description: "An updated electronic device.",  
116           image: "https://example.com/images/product1.jpg",  
117       });  
118   });  
119  
120   // Test Case 5: Delete Product  
121   it('should delete product', () => {  
122       // Mock the productService.delete() method to return a success response  
123       productService.delete = jest.fn(id => Promise.resolve({  
124           status: 200,  
125           message: "Product deleted successfully."  
126       }));  
127  
128       // Call the component's method to delete a product  
129       const result = component.deleteProduct(1);  
130  
131       // Expect the result to be equal to the success response  
132       expect(result).toEqual({  
133           status: 200,  
134           message: "Product deleted successfully."  
135       });  
136   });  
137 }  
138  
139 }  
140  
141 }
```



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



```
20000000 };

57     productService.createProduct.mockResolvedValue(
58       newProductResponse);
59
60     render(<ProductManagement />);
61     await fillAndSubmitForm(user);
62
63     await waitFor(() => {
64       expect(productService.createProduct).toHaveBeenCalledWith(
65         expect.objectContaining({
66           productName: 'New Laptop',
67           price: '20000000'
68         }));
69       expect(screen.getByText(/thành công/i)).toBeInTheDocument();
70     });
71   });

72 test('Mock: Create product that bai (Failure Scenario)', async () => {
73   const user = userEvent.setup();
74   productService.createProduct.mockRejectedValue(new Error('
75     Server Error'));
76
77   render(<ProductManagement />);
78   await fillAndSubmitForm(user);
79
80   await waitFor(() => {
81     expect(productService.createProduct).toHaveBeenCalledTimes(1);
82     expect(screen.getByText(/Server Error/i)).toBeInTheDocument();
83   });
84 });
85
86 test('Mock: Delete product', async () => {
87   const user = userEvent.setup();
88   productService.deleteProduct.mockResolvedValue({ success: true });
89
90   render(<ProductManagement />);
91   await waitFor(() => screen.getByText('Laptop Dell'));
92
93   const deleteBtns = screen.getAllByTitle(/Xóa/i);
94   await user.click(deleteBtns[0]);
95
96   const confirmBtn = await screen.findByText('Xóa', { selector: 'button' });
97   await user.click(confirmBtn);
98
99   await waitFor(() => {
100     expect(productService.deleteProduct).toHaveBeenCalledWith(1);
101   });
102 });
103
```



```
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(/Chỉnh 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: /Lưu/ });
121   await user.click(submitBtNS[submitBtNS.length - 1]);
122
123   await waitFor(() => {
124     expect(productService.updateProduct).toHaveBeenCalledWith(1,
125     expect.objectContaining({
126       price: '18000000'
127     }));
128   });
129});
```

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);
```



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



```
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 * C) Verify Repository Interactions
129 */
130 @Nested
131 @DisplayName("C) Verify Repository Interactions (0.5 điểm)")
132 class VerifyRepositoryInteractionsTests {
133
134     @Test
135     @DisplayName("5. Verify repository đ ọc gọi với argument cụ thể")
136     void testVerify_RepositoryCalledWithSpecificArgument() {
137         // Arrange
138         when(productRepository.findById(1L))
139             .thenReturn(Optional.of(mockProduct));
140
141         // Act
142         productService.getProductById(1L);
143
144         // c) Verify với argument cụ thể
145         verify(productRepository).findById(eq(1L));
146         verify(productRepository, times(1)).findById(1L);
147         verifyNoMoreInteractions(productRepository);
148     }
149
150
151     @Test
152     @DisplayName("6. Verify repository không đ ọc gọi khi validation
153 fail")
154     void testVerify_RepositoryNotCalledWhenValidationFails() {
155         // Arrange: Mock validator trả về violation
156         var violation = mock(jakarta.validation.ConstraintViolation.
157 class);
157         when(violation.getMessage()).thenReturn("Price phải > 0");
158         when(validation.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 đ ọc 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  * Component Testing Configuration (Optional)
113  *
114  * Tests individual React components in isolation
115  * using Vite dev server for fast feedback.
116  */
117 component: {
118   /**
119    * Dev server configuration for component testing
120    * Uses Vite for fast component mounting and HMR
121    */
122   devServer: {
123     framework: 'react', // Framework being tested
124     bundler: 'vite',    // Build tool (Vite for fast builds)
125   },
126   /**
127    * Pattern to match component test files
128    * Example: Button.cy.jsx, Form.cy.jsx
129    */
130   specPattern: 'src/tests/cypress/component/**/*.cy.jsx',
131   /**
132    * Support file for component tests
133    * Contains component-specific setup and commands
134    */
135   supportFile: 'src/tests/cypress/support/component.js'
136 },
137 }
138 );
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
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 setUsername(username) {
70     this.usernameInput.clear().type(username, { delay: 100 });
71     return this;
72 }

73 /**
74 * Enter password
75 */
76 setPassword(password) {
77     this.passwordInput.clear().type(password, { delay: 100 });
78     return this;
79 }

80 /**
81 * Click login button
82 */
83 clickLogin() {
84     this.loginButton.click();
85     return this;
86 }

87 /**
88 * Complete login flow
89 */
90 login(username, password) {
91     this.setUsername(username);
92     this.setPassword(password);
93     this.clickLogin();
94     return this;
95 }

96 /**
97 * Check remember me checkbox
98 */
99 checkRememberMe() {
100    this.rememberMeCheckbox.check();
101    return this;
102 }

103 /**
104 * Uncheck remember me checkbox
105 */
106 uncheckRememberMe() {
107    this.rememberMeCheckbox.uncheck();
108    return this;
109 }

110 /**
111 * Click forgot password link
112 */
113 clickForgotPassword() {
114    this.forgotPasswordLink.click();
115    return this;
116 }

117 /**
118 * Click forgot password link
119 */
120 clickForgotPassword() {
121    this.forgotPasswordLink.click();
122    return this;
123 }
```



```
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: Ngồi dùng có thẻ mờ trang login", () => {
14             LoginPage.verifyPageTitle();
15             LoginPage.verifyFormElementsVisible();
16         });
17     });
18
19     /**
20      * SCENARIO 2: Validation messages
21     */
22     describe("Scenario 2: Validation messages (0.5 điểm)", () => {
23         it("SC2.1: Nhập tên và mật khẩu không hợp lệ", () => {
24             LoginPage.fillLoginForm("invalid", "invalid");
25             LoginPage.verifyErrorMessages("Invalid email or password");
26         });
27     });
28
29     /**
30      * SCENARIO 3: Success/Error flows
31     */
32     describe("Scenario 3: Success/Error flows (0.5 điểm)", () => {
33         it("SC3.1: Đăng nhập thành công", () => {
34             LoginPage.fillLoginForm("valid", "valid");
35             LoginPage.verifySuccessMessage("Login successful!");
36         });
37         it("SC3.2: Đăng nhập thất bại", () => {
38             LoginPage.fillLoginForm("invalid", "invalid");
39             LoginPage.verifyErrorMessages("Invalid email or password");
40         });
41     });
42
43     /**
44      * SCENARIO 4: UI Element Interactions
45     */
46     describe("Scenario 4: UI Element Interactions (0.5 điểm)", () => {
47         it("SC4.1: Click forgot password link", () => {
48             LoginPage.clickForgotPasswordLink();
49             LoginPage.verifyForgotPasswordPageTitle();
50         });
51     });
52 });
53
54 // Export test object
55 export default new E2ETest();
```



```
17     it("SC1.2: Ng òi dùng có thé nhập username hợp lệ", () => {
18         const username = "test";
19         LoginPage.enterUsername(username);
20         LoginPage.verifyUsernameValue(username);
21     });
22
23
24     it("SC1.3: Ng òi dùng có thé 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 òi dùng có thé submit form với data hợp lệ", () => {
31         LoginPage.login("test", "Test123@");
32         // Form submitted - có thé 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 hién thị loading state sau khi submit", () => {
42         LoginPage.enterUsername("test");
43         LoginPage.enterPassword("Test123@");
44         LoginPage.clickLogin();
45         // Có thé verify button disabled hoặc spinner
46         cy.wait(500);
47     });
48
49     it("SC1.7: Ng òi dùng có thé clear username và nhap 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 òi dùng có thé clear password và nhap 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@").
```





```
119     // username input should still have invalid value
120     LoginPage.usernameInput.should("have.value", "invalid-
121     username");
122   });
123
124   it("SC2.5: Validation với password yếu", () => {
125     LoginPage.enterUsername("test");
126     LoginPage.enterPassword("123");
127     LoginPage.clickLogin();
128     cy.wait(300);
129     // Short password validation
130   });
131
132   it("SC2.6: Validation messages appear immediately", () => {
133     LoginPage.clickLogin();
134     cy.wait(200);
135     // Check for validation feedback
136   });
137
138   it("SC2.7: Can correct validation errors and resubmit", () => {
139     // Submit with empty form
140     LoginPage.clickLogin();
141     cy.wait(300);
142
143     // Now fill and submit
144     LoginPage.enterUsername("test").enterPassword("Test123@").
145     clickLogin();
146
147   it("SC2.8: Multiple validation errors show correctly", () => {
148     LoginPage.clearAllInputs();
149     LoginPage.clickLogin();
150     cy.wait(300);
151     // Both fields empty - verify validation
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.
161       password);
162       cy.wait(500);
163     });
164   });
165
166   it("SC3.2: Failed login dengan invalid credentials", () => {
167     cy.fixture("users").then((users) => {
168       LoginPage.login(users.invalidUser.username, users.
169       invalidUser.password);
170       cy.wait(500);
171     });
172   });
173 });
```



```
171
172     it("SC3.3: Error message visible on failed login", () => {
173         cy.fixture("users").then((users) => {
174             LoginPage.login(users.invalidUser.username, users.
175             invalidUser.password);
176             cy.wait(500);
177             // Error message may appear
178         });
179     });
180
181     it("SC3.4: Success message visible on successful login", () =>
182     {
183         cy.fixture("users").then((users) => {
184             LoginPage.login(users.validUser.username, users.validUser.
185             password);
186             cy.wait(500);
187             // Success message may appear
188         });
189     });
190
191     it("SC3.5: User can retry after failed login", () => {
192         cy.fixture("users").then((users) => {
193             // First attempt - fail
194             LoginPage.login(users.invalidUser.username, users.
195             invalidUser.password);
196             cy.wait(500);
197
198             // Clear and retry
199             LoginPage.clearAllInputs();
200             LoginPage.login(users.validUser.username, users.validUser.
201             password);
202             });
203     });
204
205     it("SC3.6: Error handling for network issues", () => {
206         LoginPage.enterUsername("test").enterPassword("Test123@").
207         clickLogin();
208         cy.wait(300);
209         // Network error handling
210     });
211
212     it("SC3.7: Timeout handling during login", () => {
213         LoginPage.login("test", "Test123@");
214         cy.wait(1000);
215         // Verify form still responsive
216         LoginPage.loginButton.should("be.visible");
217     });
218
219     it("SC3.8: Recovery from error state", () => {
220         // Attempt that might fail
221         LoginPage.login("invalid@test.com", "invalid");
222         cy.wait(500);
223
224         // Verify can interact again
225         LoginPage.loginButton.should("be.visible");
226         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:
```



```
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 LỎI) ---
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 "##     i     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:

- Test Create product flow (0.5 điểm)
- Test Read/List products (0.5 điểm)
- Test Update product (0.5 điểm)
- Test Delete product (0.5 điểm)
- 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', /quán 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 lượng|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.btn-success, button.btn-danger').first();
203     }
204
205     get cancelButton() {
206         return cy.get('.form-actions').find('button.btn-secondary');
207     }
208
209 // ===== CONFIRMATION DIALOG =====
210     get deleteConfirmDialog() {
211         return cy.get('.modal-small');
212     }
213
214     get deleteConfirmButton() {
215         return cy.get('.modal-small').find('button.btn-danger');
216     }
217
218     get deleteConfirmCancelButton() {
219         return cy.get('.modal-small').find('button.btn-secondary');
220     }
221
222 // ===== NOTIFICATION =====
223     get notification() {
224         return cy.get('.notification');
225     }
226
227     get successNotification() {
228         return cy.get('.notification.success');
229     }
230
231     get errorNotification() {
232         return cy.get('.notification.error');
233     }
234
235 // ===== DETAIL VIEW SELECTORS =====
236     get detailValue() {
237         return cy.get('.detail-value');
238     }
239
240     get detailDescription() {
241         return cy.get('.detail-description');
242     }
243
244     get editButtonInDetail() {
245         return cy.contains('button', /chỉnh sửa|edit/i);
246     }
247
248     get emptyState() {
249         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, { delay:
292             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             errorSelector = this.nameError;
468             break;
469         case 'price':
470             errorSelector = this.priceError;
471             break;
472         case 'quantity':
473             errorSelector = this.quantityError;
474             break;
475         case 'số l ợng':
476             errorSelector = this.quantityError;
477             break;
478         case 'category':
479             errorSelector = this.categoryError;
480             break;
481         case 'danh mục':
482             errorSelector = this.categoryError;
483             break;
484         case 'description':
485             errorSelector = this.descriptionError;
486             break;
487         default:
488             throw new Error(`Unknown field: ${fieldName}`);
489     }
490     errorSelector.should('be.visible');
491     return this;
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
531     // message
532     cy.get('.product-table tbody tr').then($rows => {
533         expect($rows.text()).to.include('Không có sản phẩm');
534     });
535     return this;
536 }
537
538 /**
539 * Verify table not empty
540 */
541 verifyTableNotEmpty() {
542     cy.get('.product-table tbody tr').then($rows => {
543         expect($rows.text()).not.to.include('Không có sản phẩm');
544     });
545     return this;
546 }
547
548 /**
549 * Get product row data
550 */
551 getProductName(productName) {
552     return this.getProductByName(productName).within(() => {
553         return {
554             name: cy.get('td').eq(0),
555             price: cy.get('td').eq(1),
556             quantity: cy.get('td').eq(2),
557             category: cy.get('td').eq(3)
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!Passw0rd@' -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!Passw0rd@' \
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;
109     databaseName=STDatabase_Test;trustServerCertificate=true
110     SPRING_DATASOURCE_USERNAME: sa
111     SPRING_DATASOURCE_PASSWORD: ${{ env.SQL_SERVER_SA_PASSWORD }}
112     SPRING_DATASOURCE_DRIVER_CLASS_NAME: com.microsoft.sqlserver.
113         jdbc.SQLServerDriver
114     SPRING_TEST_DATABASE_REPLACE: NONE
115     SPRING_JPA_DATABASE_PLATFORM: org.hibernate.dialect.
116         SQLServerDialect
117
118     - name: Run Integration Tests
119         run: |
120             cd backend
121             ./mvnw test -Dtest="**/*IntegrationTest"
122         env:
123             SPRING_PROFILES_ACTIVE: test
124             SPRING_DATASOURCE_URL: jdbc:sqlserver://localhost:1433;
125             databaseName=STDatabase_Test;trustServerCertificate=true
126             SPRING_DATASOURCE_USERNAME: sa
127             SPRING_DATASOURCE_PASSWORD: ${{ env.SQL_SERVER_SA_PASSWORD }}
128
129     - name: Generate JaCoCo Coverage Report
130         run: |
131             cd backend
132             ./mvnw jacoco:report
133
134     - name: Upload Backend Coverage to codecov
135         uses: codecov/codecov-action@v3
136         with:
137             token: ${{ secrets.CODECOV_TOKEN }}
138             files: ./backend/target/site/jacoco/jacoco.xml
139             flags: backend
140             name: backend-coverage
141             fail_ci_if_error: false
142
143     - name: Upload Backend Test Results
144         if: always()
145         uses: actions/upload-artifact@v4
146         with:
147             name: backend-test-results
148             path: |
149                 backend/target/surefire-reports/
150                 backend/target/site/jacoco/
151
152     - name: Publish Test Report
153         if: always()
154         uses: dornny/test-reporter@v1
155         with:
156             name: Backend Tests
157             path: backend/target/surefire-reports/*.xml
158             reporter: java-junit
159             fail-on-error: false
160
161
162 # JOB 2: Frontend Build & Test
```



```
159
160 frontend-tests:
161   name: Frontend - Build & Test
162   runs-on: ubuntu-latest
163
164 steps:
165   - name: Checkout code
166     uses: actions/checkout@v3
167
168   - name: Setup Node.js ${{ env.NODE_VERSION }}
169     uses: actions/setup-node@v3
170     with:
171       node-version: ${{ env.NODE_VERSION }}
172       # Tạm thời tắt cache để tránh restore file lock cũ
173       # cache: "npm"
174       # cache-dependency-path: frontend/package-lock.json
175
176   # FIX 2: Xóa package-lock.json để ép npm tìm gói Linux
177   - name: Install Frontend Dependencies
178     run: |
179       cd frontend
180       rm -rf node_modules package-lock.json
181       npm install
182
183   - name: Run ESLint
184     run: |
185       cd frontend
186       npm run lint || true
187       continue-on-error: true
188
189   - name: Run Frontend Unit Tests
190     run: |
191       cd frontend
192       npm test -- --coverage --watchAll=false
193     env:
194       CI: true
195
196   - name: Upload Frontend Coverage toCodecov
197     uses: codecov/codecov-action@v3
198     with:
199       token: ${{ secrets.CODECOV_TOKEN }}
200       files: ./frontend/coverage/lcov.info
201       flags: frontend
202       name: frontend-coverage
203       fail_ci_if_error: false
204
205   - name: Build Frontend
206     run: |
207       cd frontend
208       npm run build
209     env:
210       CI: false
211       REACT_APP_API_URL: http://localhost:6969
212
213   - name: Upload Frontend Build Artifacts
214     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 $(docker ps -q -f ancestor=mcr.microsoft.com/mssql-
273 /server:2022-latest) \
274                  /opt/mssql-tools18/bin/sqlcmd \
275                      -S localhost -U sa -P "${{ env.SQL_SERVER_SA_PASSWORD }}" \
276                      -C -Q "CREATE DATABASE STDatabase_E2E;""
277
278      - name: Start Backend Server
279          run: |
280              cd backend
281              chmod +x mvnw
282              ./mvnw clean install -DskipTests
283              nohup ./mvnw spring-boot:run \
284                  -Dspring-boot.run.arguments="--spring.datasource.url=jdbc:
285 sqlserver://localhost:1433;databaseName=STDatabase_E2E;
286 trustServerCertificate=true --spring.datasource.username=sa --spring.
287 datasource.password=${{ env.SQL_SERVER_SA_PASSWORD }}" &
288                  echo $! > backend.pid
289
290
291      # Wait for backend to be ready
292      echo "Waiting for backend to start..."
293      for i in {1..60}; do
294          if curl -s http://localhost:6969/actuator/health > /dev/null
295 2>&1; then
296              echo "Backend is ready!"
297              break
298          fi
299          echo "Attempt $i: Backend not ready yet..."
300          sleep 5
301      done
302
303      - name: Install Frontend Dependencies
304          run: |
305              cd frontend
306              rm -rf node_modules package-lock.json
307              npm install
308
309      - name: Start Frontend Server
310          run: |
311              cd frontend
312              nohup npm run dev &
313              echo $! > frontend.pid
314
315
316      # Wait for frontend to be ready
317      echo "Waiting for frontend to start..."
318      for i in {1..60}; do
319          if curl -s http://localhost:5173 > /dev/null 2>&1; then
320              echo "Frontend is ready!"
321              break
322          fi
323          echo "Attempt $i: Frontend not ready yet..."
324          sleep 5
325      done
326
327      env:
328          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;
328      then
329          npm run test:e2e || echo "E2E tests not configured yet"
330      else
331          echo "E2E tests not found, skipping..."
332      fi
333      continue-on-error: true
334
335      - name: Stop Servers
336          if: always()
337          run: |
338              if [ -f backend/backend.pid ]; then
339                  kill $(cat backend/backend.pid) || true
340              fi
341              if [ -f frontend/frontend.pid ]; then
342                  kill $(cat frontend/frontend.pid) || true
343              fi
344
345      - name: Upload E2E Test Results
346          if: always()
347          uses: actions/upload-artifact@v4
348          with:
349              name: e2e-test-results
350              path: |
351                  frontend/cypress/screenshots/
352                  frontend/cypress/videos/
353                  frontend/playwright-report/
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 $(docker ps -q -f ancestor=mcr.microsoft.com/mssql
394        /server:2022-latest) \
395          /opt/mssql-tools18/bin/sqlcmd \
396            -S localhost -U sa -P 'YourStrong!Passw0rd@' \
397            -C -Q "CREATE DATABASE STDatabase_Perf;""
398
399    - name: Start Backend Server
400      run: |
401        cd backend
402        chmod +x mvnw
403        ./mvnw clean install -DskipTests
404        nohup ./mvnw spring-boot:run \
405          -Dspring-boot.run.arguments="--spring.datasource.url=jdbc:
406          sqlserver://localhost:1433;databaseName=STDatabase_Perf;
407          trustServerCertificate=true --spring.datasource.username=sa --spring.
408          datasource.password=${{ 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 hkp://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 Phân 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
3     '1='1")
4     void testSqlInjection_LoginBypass_Classic() throws
5     Exception {
6         LoginRequest maliciousRequest = new LoginRequest("admin
7         ' OR '1='1", "anything");
8
9         mockMvc.perform(post("/api/auth/login")
10             .contentType(MediaType.APPLICATION_JSON)
11             .content(objectMapper.writeValueAsString(
12                 maliciousRequest)))
13             .andExpect(status().isBadRequest())
14             .andExpect(jsonPath("$.success").value(false));
15     }
16
17     @Test
18     @DisplayName("TC1.2: SQL Injection - Login bypass với ' OR
19     1=1--")
20     void testSqlInjection_LoginBypass_Comment() throws
21     Exception {
22         LoginRequest maliciousRequest = new LoginRequest("admin
23         ' OR 1=1--", "anything");
24
25         mockMvc.perform(post("/api/auth/login")
26             .contentType(MediaType.APPLICATION_JSON)
27             .content(objectMapper.writeValueAsString(
28                 maliciousRequest)))
29             .andExpect(status().isBadRequest())
30             .andExpect(jsonPath("$.success").value(false));
31     }
32
33     @Test
```



```
26     @DisplayName("TC1.3: SQL Injection - Login bypass với ';
27     DROP TABLE users--")
28     void testSqlInjection_LoginBypass_DropTable() throws
29     Exception {
30
31         LoginRequest maliciousRequest = new LoginRequest("admin
32         '; DROP TABLE users--", "password");
33
34
35         mockMvc.perform(post("/api/auth/login")
36                         .contentType(MediaType.APPLICATION_JSON)
37                         .content(objectMapper.writeValueAsString(
38                     maliciousRequest)))
39                         .andExpect(status().isBadRequest());
40
41
42         // Verify table vẫn tồn tại
43         long userCount = userRepository.count();
44         assert userCount > 0 : "Users table should still exist";
45     }
46 }
```

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')>",
5          100.0, "Test", 10, "Electronics");
6
7          mockMvc.perform(post("/api/products")
8                          .header("Authorization", authToken)
9                          .contentType(MediaType.APPLICATION_JSON)
10                         .content(objectMapper.writeValueAsString(
11                     xssRequest)))
12                         .andExpect(status().isBadRequest())
13                         .andExpect(jsonPath("$.errors.productName").
14                         exists());
15
16
17      @Test
18      @DisplayName("TC10.1: XSS - innerHTML manipulation attempt"
19      )
20      void testXss_DomBased_InnerHtml() throws Exception {
21          CreateProductRequest xssRequest = new
CreateProductRequest("Product<img src=x onerror='document.
22 body.innerHTML=\\"<h1>Hacked</h1>\\"'>", 100.0, "Test", 10, "Electronics");
23
24          mockMvc.perform(post("/api/products")
25                          .header("Authorization", authToken)
26                          .contentType(MediaType.APPLICATION_JSON)
```



```
23             .content(objectMapper.writeValueAsString(
24                 XSSRequest)))
25                 .andExpect(status().isBadRequest())
26                 .andExpect(jsonPath("$.errors.productName").
27             exists());
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(
9                             loginRequest)))
10                    .andExpect(status().isBadRequest());
11    }
12
13    @Test
14    @DisplayName("TC2.1: Auth bypass - Wrong password")
15    void testAuthBypass_WrongPassword() throws Exception {
16        LoginRequest loginRequest = new LoginRequest("admin", "wrongpassword");
17
18        mockMvc.perform(post("/api/auth/login")
19                         .contentType(MediaType.APPLICATION_JSON)
20                         .content(objectMapper.writeValueAsString(
21                             loginRequest)))
22                    .andExpect(status().isBadRequest()) // Password
23                    .andExpect(jsonPath("$.success").value(false));
24
25    @Test
26    @DisplayName("TC4.1: Auth bypass - Invalid JWT token")
27    void testAuthBypass_InvalidToken() throws Exception {
28        mockMvc.perform(get("/api/products")
29                         .header("Authorization", "Bearer invalid.token.
30 here"))
31                         .andExpect(status().isForbidden());
32    }
33
34
```

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("$.errors.productName").exists
15     );
16
17  @Test
18      @DisplayName("TC6.1: Sanitization - Special characters trong
19          product name")
20      void testSanitization_SpecialCharacters_ProductName() throws
Exception {
21          CreateProductRequest request = new CreateProductRequest("Product <>&\\"", 100.0, "Test", 10, "Electronics");
22
23          mockMvc.perform(post("/api/products")
24             .header("Authorization", authToken)
25             .contentType(MediaType.APPLICATION_JSON)
26             .content(objectMapper.writeValueAsString(request)))
27             .andExpect(status().isBadRequest())
28             .andExpect(jsonPath("$.errors.productName").exists
29     );
30 }
```

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
16     BCrypt")
17     void testPasswordHashing() {
18         // Arrange
19         String rawPassword = "mySecurePassword123";
20
21         // Act - Hash password
22         String hashedPassword = passwordEncoder.encode(rawPassword)
23     ;
24
25         // Assert
26         assertNotNull(hashedPassword, "Hashed password không đ ọc
27         null");
28         assertEquals(rawPassword, hashedPassword, "Password không
29         đ ọc l u dạng plain text");
30         assertTrue(hashedPassword.startsWith("$2a$") ||
31         hashedPassword.startsWith("$2b$"),
32             "Password phải đ ọc hash bằng BCrypt");
33         assertTrue(hashedPassword.length() >= 60, "BCrypt hash phải
34         có độ dài >= 60 ký tự");
35
36         // Verify password matching
37         boolean matches = passwordEncoder.matches(rawPassword,
38         hashedPassword);
39         assertTrue(matches, "Password encoder phải verify đ ọc
40         password đúng");
41
42         // Verify wrong password doesn't match
43         boolean wrongMatch = passwordEncoder.matches("wrongPassword",
44         hashedPassword);
45         assertFalse(wrongMatch, "Password encoder phải reject
46         password sai");
47     }
48
49
50 // 3. CORS CONFIGURATION TEST
51
52
53     @Test
54     @DisplayName("3. CORS Configuration - CORS headers đ ọc câu hì
55     nh đúng")
56     void testCorsConfiguration() throws Exception {
57         // Arrange
58         String origin = "http://localhost:5173";
59
60         // Act - Preflight request (OPTIONS)
61         mockMvc.perform(options("/api/products")
62             .header("Origin", origin)
63             .header("Access-Control-Request-Method", "POST")
64             .header("Access-Control-Request-Headers", "Content-Type, Authorization"))
65
66             // Assert
67             .andExpect(status().isOk())
68             .andExpect(header().exists("Access-Control-Allow-  
Content-Type"))
```



```
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
ó");
85     assertTrue(xFrameOptions.equals("DENY") || xFrameOptions.
equals("SAMEORIGIN"),
86                 "X-Frame-Options phải là DENY hoặc SAMEORIGIN");
87 }
88 }
```

Listing 34: SecurityBestPracticesTest.java