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)

```
describe("validateUsername()", () => {
  describe("Test 1: Username rõng", () => {
    test("nên trá vè không hợp lẹ khi username là chuõi rõng", () => {
      const result = validateUsername("");
      expect(result.isValid).toBe(false);
      expect(result.error).toBe("Username không ḍ ọc ḍé tróng");
    });

    .....

    test("nên trá vè không hợp lẹ khi username chí có khoáng tráng", () => {
      const result = validateUsername("   ");
      expect(result.isValid).toBe(false);
      expect(result.error).toBe("Username không ḍ ọc ḍé tróng");
    });
  });


  describe("Test 2: Username quá ngán/quá dài", () => {
    test("nên trá vè không hợp lẹ khi username quá ngán (ít h n 3 k ý tụ)", () => {
      const result = validateUsername("ab");
      expect(result.isValid).toBe(false);
      expect(result.error).toBe("Username phái ít nhát 3 ký tụ");
    });

    ........

    test("nên hợp lẹ vói ḍúng 50 ký tụ", () => {
      const result = validateUsername(
        "abcdefghijabcdefghijabcdefghijabcdefghijabcdefghij"
      );
      expect(result.isValid).toBe(true);
      expect(result.error).toBeNull();
    });
  });


  describe("Test 3: Ký tụ ḍạc biẹt không hợp lẹ", () => {
    test("nên trá vè không hợp lẹ khi username chúa khoáng tráng", () => {
      const result = validateUsername("user name");
      expect(result.isValid).toBe(false);
```

```
42        expect(result.error).toBe(
43          "Username chí chúa chũ, só, dáu chám (.), gạch d ói (_), và
      gạch ngang (-)"
44        );
45      });
46
47      .....
48
49      test("nên họp lẹ vói 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 họp lẹ", () => {
57      test("nên họp lẹ khi có chũ cái, gạch d ói, chám và gạch ngang
     ", () => {
58        const result = validateUsername("user_n.ame-");
59        expect(result.isValid).toBe(true);
60        expect(result.error).toBeNull();
61      });
62    });
63  });
64
```

Listing 1: validateUsername

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

```
1  describe("validatePassword()", () => {
2    describe("Test 1: Password rõng", () => {
3      test("nên trá vè không họp lẹ khi password là chuõi rõng", ()
     => {
4        const result = validatePassword("");
5        expect(result.isValid).toBe(false);
6        expect(result.error).toBe("Mạt kháu không d ọc dé tróng");
7      });
8
9      .....
10
11     test("nên trá vè không họp lẹ khi password là undefined", () =>
       {
12       const result = validatePassword(undefined);
13       expect(result.isValid).toBe(false);
14       expect(result.error).toBe("Mạt kháu không d ọc dé tróng");
15     });
16   });
17
18
19   describe("Test 2: Password quá ngán/quá dài", () => {
20     test("nên trá vè không họp lẹ khi password quá ngán (ít h n 6 k
      ý tụ)", () => {
21       const result = validatePassword("pass1");
22       expect(result.isValid).toBe(false);
23       expect(result.error).toBe("Mạt kháu phái ít nhát 6 ký tụ");
```

```
    });

    ....

    test("nên hợp lệ với đúng 50 ký tụ", () => {
      const password = "a".repeat(44) + "123456";
      const result = validatePassword(password);
      expect(result.isValid).toBe(true);
      expect(result.error).toBeNull();
    });
  });


  describe("Test 3: Password không có chũ hoạc só", () => {
    test("nên trá vè không hợp lẹ khi password không có chũ cái",
    () => {
      const result = validatePassword("123456");
      expect(result.isValid).toBe(false);
      expect(result.error).toBe("Mạt kháu phái chúa ít nhát mọt chũ
    cái");
    });

    .....

    test("nên trá vè không hợp lẹ khi password có chũ cái và ký tụ
    đạc biệt nh ng không có só", () => {
      const result = validatePassword("pass!@#$");
      expect(result.isValid).toBe(false);
      expect(result.error).toBe("Mạt kháu phái chúa ít nhát mọt só
    ");
    });
  });


  describe("Test 4: Password hợp lẹ", () => {
    test("nên hợp lẹ khi có chũ cái và só", () => {
      const result = validatePassword("password123");
      expect(result.isValid).toBe(true);
      expect(result.error).toBeNull();
    });

    ....

    test("nên hợp lẹ khi chí có mọt só và các chũ cái khác", () =>
    {
      const result = validatePassword("password1");
      expect(result.isValid).toBe(true);
      expect(result.error).toBeNull();
    });
  });
});

```

Listing 2: validatePassword

c) Coverage >= 90% cho validation module (1 điểm)

Assigment_template_SGU/ValidationLogin.jpeg

3.1.2 Backend Unit Tests - Login Service (5 điểm)

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

```java
@ExtendWith(MockitoExtension.class)
@DisplayName("AuthService Unit Test - Bean Validation")
public class AuthServiceTest {

    @Mock
    private UserRepository mockUserRepository;

    @Mock
    private PasswordEncoder mockPasswordEncoder;

    @Mock
    private JwtService mockJwtService;

    @Mock
    private Validator mockValidator;


    private Validator realValidator;
```

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

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

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

```
    ông ợ ọc ợé tróng"));
152         assertTrue(isHasPasswordError);
153     }
154
155 }
```

Listing 3: AuthServiceTest.java

b) Test validation methods riêng lẻ (1 điểm)

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

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

```
    realValidator.validate(req);

        //Assert
        assertFalse(violations.isEmpty(), "Phải có lỗi validate password");
        boolean isHasError = violations.stream()
                .anyMatch(v -> v.getMessage().contains("Password phải từ 6 đến 100 ký tự"));
        assertTrue(isHasError, "Phải có thông báo lỗi password");

    }

    @Test
    @DisplayName("TC9: Login thất bại với validation error - username có ký tự đạc biệt")
    void testLoginFailure_ValidationError_InvalidUsernameChars() {
        //Arrange
        LoginRequest req = new LoginRequest("user@#$#", "Test1234");

        //Act
        Set<ConstraintViolation<LoginRequest>> violations = realValidator.validate(req);

        //Assert
        assertFalse(violations.isEmpty(), "Phải có lỗi username có chúa ký tự đạc biệt");
        boolean isHasError = violations.stream()
                .anyMatch(v -> v.getMessage().contains("Username chỉ chúa chũ, só, và ký tụ (-, ., _)"));
        assertTrue(isHasError, "Phải có thông báo lỗi username có ký tụ đạc biệt");

    }

    @Test
    @DisplayName("TC10: Login thất bại với validation error - password thiếu chũ cái")
    void testLoginFailure_ValidationError_PasswordMissingLetter() {
        //Arrange
        LoginRequest req = new LoginRequest("testUser", "123456");

        //Act
        Set<ConstraintViolation<LoginRequest>> violations = realValidator.validate(req);

        //Assert
        assertFalse(violations.isEmpty(), "Phải có lỗi password thiếu chũ cái");
        boolean isHasError = violations.stream()
                .anyMatch(v -> v.getMessage().contains("Password phải chúa ít nhát 1 chũ cái"));
        assertTrue(isHasError, "Phải có thông báo lỗi password thiếu chũ cái");
    }
```

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

Listing 4: AuthServiceTest.java

c) Coverage >= 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):

```
describe('validateProduct - Kiém tra tên sán phám', () => {
  test('trá vè lõi khi tên rõng', () => {
    const product = { name: '', price: 100, quantity: 10,
   description: 'Valid description', category: 'Electronics' };
    const result = validateProduct(product);

    expect(result.isValid).toBe(false);
    expect(result.errors.name).toBe('Tên sán phám không ọ ọc ọé tró
   ng');
  });

    ....

  test('họp lẹ khi tên họp lẹ', () => {
    const product = { name: 'Valid Product Name', price: 100,
   quantity: 10, description: 'Valid description', category: '
   Electronics' };
```

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

```
60
61      expect(result.isValid).toBe(false);
62      expect(result.errors.quantity).toBe('Số l ợng không đ ợc đ é tró
        ng');
63    });
64      ......
65
66    test('họp lệ khi số l ợng là số nguyên họp lệ', () => {
67      const product = { name: 'Product', price: 100, quantity: 50,
        description: 'Valid description', category: 'Electronics' };
68      const result = validateProduct(product);
69
70      expect(result.errors.quantity).toBeUndefined();
71    });
72  });
73
74  describe('validateProduct - Kiém tra đ ộ dài mô tả', () => {
75    test('trả vè lõi khi mô tả rõng', () => {
76      const product = { name: 'Product', price: 100, quantity: 10,
        description: '', category: 'Electronics' };
77      const result = validateProduct(product);
78
79      expect(result.isValid).toBe(false);
80      expect(result.errors.description).toBe('Mô tả không đ ợc đ é tró
        ng');
81    });
82
83      .....
84
85    test('họp lệ khi mô tả họp 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 - Kiém tra danh mục', () => {
100   test('trả vè lõi khi danh mục rõng', () => {
101     const product = { name: 'Product', price: 100, quantity: 10,
        description: 'Valid description', category: '' };
102     const result = validateProduct(product);
103
104     expect(result.isValid).toBe(false);
105     expect(result.errors.category).toBe('Danh mục không đ ợc đ é tró
        ng');
106   });
107     ......
108
109   test('họp lệ khi danh mục là Other', () => {
```

```
    const product = { name: 'Product', price: 100, quantity: 10, description: 'Valid description', category: 'Other' };
    const result = validateProduct(product);

    expect(result.errors.category).toBeUndefined();
  });
});

describe('validateProduct - Kiểm tra tích hợp', () => {
  test('trả về isValid true khi tất cả trường hợp hợp lệ', () => {
    const product = {
      name: 'Valid Product',
      price: 100,
      quantity: 10,
      description: 'This is a valid description',
      category: 'Electronics'
    };
    const result = validateProduct(product);

    expect(result.isValid).toBe(true);
    expect(result.errors).toEqual({});
  });

  test('trả về nhiều lỗi khi nhiều trường không hợp lệ', () => {
    const product = {
      name: '',
      price: -100,
      quantity: 'abc',
      description: 'Short',
      category: ''
    };
    const result = validateProduct(product);

    expect(result.isValid).toBe(false);
    expect(result.errors.name).toBeDefined();
    expect(result.errors.price).toBeDefined();
    expect(result.errors.quantity).toBeDefined();
    expect(result.errors.description).toBeDefined();
    expect(result.errors.category).toBeDefined();
  });
});
```

Listing 5: validateProduct()

b) Viết tests cho Product form component (1 điểm)

c) Coverage >= 90 (1 điểm)

coverage_backend.jpg

3.2.2 Backend Unit Tests - Product Service (5 điểm)

a) Test CRUD operations (4 điểm):

```
@DisplayName("ProductService Unit Test")
@ExtendWith(MockitoExtension.class)
public class ProductServiceTest {

    @Mock
    private ProductRepository productRepository;

    @Mock
    private Validator mockValidator;

    private ProductService productService;

    @BeforeEach
    void setUp() {
        productService = new ProductService(productRepository,
    mockValidator);
    }
```

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

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

```
108     class DeleteProductTests {
109 
110         @Test
111         @DisplayName("TC20: Xóa sán phám 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(
    Optional.of(product));
115             doNothing().when(productRepository).deleteById(1L);
116 
117             productService.deleteProduct(1L);
118 
119             verify(productRepository, times(1)).findById(1L);
120             verify(productRepository, times(1)).deleteById(1L);
121         }
122 
123         @Test
124         @DisplayName("TC21: Xóa sán phám thát bại - ID không tòn tạ
    i")
125         void testDeleteProduct_NotFound() {
126             when(productRepository.findById(999L)).thenReturn(
    Optional.empty());
127 
128             NoSuchElementException exception = assertThrows(
129                     NoSuchElementException.class,
130                     () -> productService.deleteProduct(999L)
131             );
132 
133             assertTrue(exception.getMessage().contains("Product not
     found with id: 999"));
134         }
135     }
136 
137     // E) TEST GET ALL (PAGINATION)
138 
139         @Test
140         @DisplayName("TC22: Láy danh sách sán phám vói pagination")
141         void testGetAll_Success() {
142             Pageable pageable = PageRequest.of(0, 10);
143             List<Product> products = Arrays.asList(
144                     new Product(1L, "Electronics", "Product 1", 10,
     "Laptop", 15000.0),
145                     new Product(2L, "Books", "Product 2", 20, "Book
    ", 50.0)
146             );
147             Page<Product> productPage = new PageImpl<>(products,
    pageable, products.size());
148 
149             when(productRepository.findAll(pageable)).thenReturn(
    productPage);
150 
151             Page<ProductDto> result = productService.getAll(
    pageable);
152 
153             assertNotNull(result);
```

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

Listing 6: ProductServiceTest.java

b) Coverage >= 85 cho ProductService (1 điểm)

coverage_backend.jpg

4 Câu 3: Integration Testing

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

4.1.1 Frontend Component Integration (5 điểm)

a) Test rendering và user interactions (2 điểm)

```
  describe('Test 1: Rendering và User Interactions', () => {

    test('nên render login form component thành công', () => {
   render(
  <MemoryRouter>
    <LoginForm />
  </MemoryRouter>
);

      // Query cụ thé h n dé tránh matching multiple elements
      const heading = screen.getByRole('heading', { level: 1 });
      expect(heading).toHaveTextContent(/Đăng Nhạp/i);
      expect(screen.getByText(/Chào mùng bạn quay lại/i)).
   toBeInTheDocument();
    });

    test('nên có thé nhạp username vào input', async () => {
      const user = userEvent.setup();
   render(
  <MemoryRouter>
    <LoginForm />
  </MemoryRouter>
);

      const usernameInput = screen.getByPlaceholderText(/
   your_username/i);
      await user.type(usernameInput, 'testuser');

      expect(usernameInput.value).toBe('testuser');
    });

    test('nên có thé nhạp password vào input', async () => {
      const user = userEvent.setup();
   render(
  <MemoryRouter>
    <LoginForm />
  </MemoryRouter>
);

      const passwordInput = screen.getByPlaceholderText(/
                           /i);
      await user.type(passwordInput, 'password123');

      expect(passwordInput.value).toBe('password123');
    });

    test('nên disable button khi loading', async () => {
      const user = userEvent.setup();
```

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

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(/
     your_username/i);
18       const passwordInput = screen.getByPlaceholderText(/
                               /i);
19       const submitButton = screen.getByRole('button', { name: /Đăng
     Nhạp/i });
20
21       await user.type(usernameInput, 'admin');
22       await user.type(passwordInput, 'admin123');
23       await user.click(submitButton);
24
```

```
      // Verify API call
      await waitFor(() => {
        expect(authService.login).toHaveBeenCalledWith('admin', 'admin123');
      });
    });

    test('nên hiển thị success message khi API trả về success - testuser', async () => {
      const user = userEvent.setup();
      authService.login.mockResolvedValue({
        success: true,
        message: 'Login thành công',
        user: { userName: 'testuser', email: 'testuser@example.com' }
      });

    render(
  <MemoryRouter>
    <LoginForm />
  </MemoryRouter>
);

      const usernameInput = screen.getByPlaceholderText(/your_username/i);
      const passwordInput = screen.getByPlaceholderText(/i);
      const submitButton = screen.getByRole('button', { name: /Đăng Nhạp/i });

      await user.type(usernameInput, 'testuser');
      await user.type(passwordInput, 'test1234');
      await user.click(submitButton);

      // Verify API called
      await waitFor(() => {
        expect(authService.login).toHaveBeenCalledWith('testuser', 'test1234');
      });
    });
  });

```

Listing 8: form submission và API calls

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

```
  describe('Test 3: Error Handling và Success Messages', () => {

    test('nên hiển thị error khi username tróng', async () => {
      const user = userEvent.setup();
    render(
  <MemoryRouter>
    <LoginForm />
  </MemoryRouter>
);
```

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

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

Listing 9: error handling và success messages

4.1.2 Backend API Integration (5 điểm)

Test API endpoints của Login với `MockMvc`:

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

```java
@WebMvcTest(value = AuthController.class,
    excludeFilters = @ComponentScan.Filter(type = FilterType.ASSIGNABLE_TYPE,
        classes = {com.flogin.service.SecurityConfig.class, com.flogin.filter.JwtAuthenticationFilter.class}))
@AutoConfigureMockMvc(addFilters = false)
@DisplayName("Login API Integration Tests")
public class AuthControllerIntegrationTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    @MockitoBean
    private AuthService authService;

    @Nested
    @DisplayName("A) Test POST /api/auth/login endpoint")
    class LoginEndpointTests {

        @Test
        @DisplayName("1. Login thành công với credentials hợp lệ")
        void testLoginSuccess() throws Exception {
            // Arrange: Chuẩn bị dữ liệu test
            LoginRequest request = new LoginRequest("testuser", "Test123");
            UserDto userDto = new UserDto("testuser", "testuser@example.com");
            LoginResponse mockResponse = new LoginResponse(
                true,
                "Login thành công",
                "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.token123",
                userDto
            );

            // Mock service trả về response thành công
            when(authService.authenticate(any(LoginRequest.class)))
                .thenReturn(mockResponse);

            // Act & Assert: Thực hiện request và verify kết quả
            mockMvc.perform(post("/api/auth/login")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(objectMapper.writeValueAsString(request)))
                    .andExpect(status().isOk()) // Expect 200 OK
                    .andExpect(jsonPath("$.success").value(true))
                    .andExpect(jsonPath("$.message").value("Login thành công"))
                    .andExpect(jsonPath("$.token").exists())
                    .andExpect(jsonPath("$.token").value("eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.token123"))
                    .andExpect(jsonPath("$.user").exists())
                    .andExpect(jsonPath("$.user.userName").value("
```

```
    testuser"))
                    .andExpect(jsonPath("$.user.email").value("
    testuser@example.com"));
        }

        @Test
        @DisplayName("2. Login thất bại - Username không tồn tại")
        void testLoginFailure_UserNotFound() throws Exception {
            // Arrange: User không tồn tại trong database
            LoginRequest request = new LoginRequest("nonexistuser",
     "Test123");
            LoginResponse mockResponse = new LoginResponse(
                false,
                "Login thất bại với user name không tồn tại"
            );

            when(authService.authenticate(any(LoginRequest.class)))
                .thenReturn(mockResponse);

            // Act & Assert: Expect 401 Unauthorized
            mockMvc.perform(post("/api/auth/login")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(objectMapper.writeValueAsString(
    request)))
                    .andExpect(status().isUnauthorized()) // Expect
     401
                    .andExpect(jsonPath("$.success").value(false))
                    .andExpect(jsonPath("$.message").value("Login
    thất bại với user name không tồn tại"))
                    .andExpect(jsonPath("$.token").doesNotExist())
                    .andExpect(jsonPath("$.user").doesNotExist());
        }


     }
```

Listing 10: AuthControllerIntegrationTest.java

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

```
@Nested
    @DisplayName("B) Test Response Structure và Status Codes")
    class ResponseStructureTests {

        @Test
        @DisplayName("1. Response structure có đầy đủ các field khi
     login thành công")
        void testSuccessResponseStructure() throws Exception {
            // Arrange
            LoginRequest request = new LoginRequest("testuser", "
    Test123");
            UserDto userDto = new UserDto("testuser", "
    testuser@example.com");
            LoginResponse mockResponse = new LoginResponse(
                true,
                "Login thành công",
```

```
                "token.jwt",
                userDto
            );

            when(authService.authenticate(any(LoginRequest.class)))
                .thenReturn(mockResponse);

            // Act & Assert: Verify response có đầy đủ field
            mockMvc.perform(post("/api/auth/login")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(objectMapper.writeValueAsString(request)))
                    .andExpect(status().isOk())
                    .andExpect(jsonPath("$.success").exists())
                    .andExpect(jsonPath("$.success").isBoolean())
                    .andExpect(jsonPath("$.message").exists())
                    .andExpect(jsonPath("$.message").isString())
                    .andExpect(jsonPath("$.token").exists())
                    .andExpect(jsonPath("$.token").isString())
                    .andExpect(jsonPath("$.user").exists())
                    .andExpect(jsonPath("$.user.userName").exists())
                    .andExpect(jsonPath("$.user.email").exists());
        }


        @Test
        @DisplayName("2. Response structure khi login thất bại (không có token)")
        void testFailureResponseStructure() throws Exception {
            // Arrange
            LoginRequest request = new LoginRequest("wronguser", "Test123");
            LoginResponse mockResponse = new LoginResponse(
                false,
                "Login thất bại với user name không tồn tại"
            );

            when(authService.authenticate(any(LoginRequest.class)))
                .thenReturn(mockResponse);

            // Act & Assert: Verify response không có token và user khi thất bại
            mockMvc.perform(post("/api/auth/login")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(objectMapper.writeValueAsString(request)))
                    .andExpect(status().isUnauthorized())
                    .andExpect(jsonPath("$.success").value(false))
                    .andExpect(jsonPath("$.message").exists())
                    .andExpect(jsonPath("$.token").doesNotExist()) // Token không tồn tại khi fail
                    .andExpect(jsonPath("$.user").doesNotExist()); // User cũng không tồn tại khi fail
        }

```

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

Listing 11: AuthControllerIntegrationTest.java

c) Test CORS và headers (1 điểm)

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

```
    class CorsAndHeadersTests {

        @Test
        @DisplayName("1. CORS - Access-Control-Allow-Origin header có tồn tại")
        void testCors_AllowOriginHeader() throws Exception {
            // Arrange
            LoginRequest request = new LoginRequest("testuser", "Test123");
            UserDto userDto = new UserDto("testuser", "testuser@example.com");
            LoginResponse mockResponse = new LoginResponse(true, "Login thành công", "token123", userDto);

            when(authService.authenticate(any(LoginRequest.class)))
                .thenReturn(mockResponse);

            // Act & Assert: Verify CORS header
            mockMvc.perform(post("/api/auth/login")
                    .contentType(MediaType.APPLICATION_JSON)
                    .header("Origin", "http://localhost:3000") // Giả lạp request từ frontend
                    .content(objectMapper.writeValueAsString(request)))
                    .andExpect(status().isOk())
                    .andExpect(header().exists("Access-Control-Allow-Origin"));
        }


        @Test
        @DisplayName("4. Request accept header - application/json")
        void testAcceptHeader_ApplicationJson() throws Exception {
            // Arrange
            LoginRequest request = new LoginRequest("testuser", "Test123");
            UserDto userDto = new UserDto("testuser", "testuser@example.com");
            LoginResponse mockResponse = new LoginResponse(true, "Login thành công", "token123", userDto);

            when(authService.authenticate(any(LoginRequest.class)))
                .thenReturn(mockResponse);

            // Act & Assert: Verify server accept JSON
            mockMvc.perform(post("/api/auth/login")
                    .contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON) // Client yêu cầu JSON response
                    .content(objectMapper.writeValueAsString(request)))
                    .andExpect(status().isOk())
                    .andExpect(content().contentType(MediaType.APPLICATION_JSON));
        }

```

```
46     }
```

Listing 12: AuthControllerIntegrationTest.java

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

4.2.1 Frontend Component Integration (5 điểm)

a) Test ProductList component với API (2 điểm)

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

Listing 13: ProductList component với API

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

```
describe("Test 2: ProductForm Component (create/edit) ", () => {

    test("nên validate product data tr óc submit", async () => {
      const user = userEvent.setup();
      render(<ProductManagement />);

      const addButton = screen
        .queryAllByRole("button")
        .find(
          (btn) =>
            btn.textContent?.includes("+") || btn.textContent?.
    includes("Thêm")
        );

      if (addButton) {
        await user.click(addButton);
        // Component renders successfully
        const inputs = screen.queryAllByRole("textbox");
        expect(inputs.length > 0).toBe(true);
      }
    });

    test("nên có category dropdown/select", async () => {
      const user = userEvent.setup();
      render(<ProductManagement />);

      // Verify select elements exist in page
      const selects = screen.queryAllByRole("combobox");
      expect(selects.length > 0).toBe(true);
    });

    test("nên có Submit/Save button", async () => {
      const user = userEvent.setup();
      render(<ProductManagement />);

      const addButton = screen
        .queryAllByRole("button")
        .find(
          (btn) =>
            btn.textContent?.includes("+") || btn.textContent?.
    includes("Thêm")
        );

      if (addButton) {
        await user.click(addButton);
        const buttons = screen.queryAllByRole("button");
        expect(buttons.length > 0).toBe(true);
      }
    });
  });

```

Listing 14: ProductForm component (create/edit)

c) Test ProductDetail component (1 điểm))

```
  describe("Test 3: ProductDetail", () => {
    test("nên display product info: name, price, quantity, category
   ", async () => {
      render(<ProductManagement />);

      // Check if table or list displays product info columns
      await waitFor(() => {
        const headers = screen.queryAllByText(
          /tên|name|giá|price|số l ọng|quantity|loại|category/i
        );
        expect(headers.length > 0).toBe(true);
      });
    });

    test("nên display product details khi click view/expand", async
    () => {
      const user = userEvent.setup();
      render(<ProductManagement />);

      // Look for view buttons or expandable rows
      const viewButtons = screen.queryAllByRole("button");
      expect(viewButtons.length > 0).toBe(true);
    });
  });

```

Listing 15: ProductDetail component

4.2.2 Backend API Integration (5 điểm)

Test các API endpoints của Product:

a) Test POST /api/products (Create) (1 điểm)

b) Test GET /api/products (Read all) (1 điểm)

c) Test GET /api/products/id (Read one) (1 điểm)

d) Test PUT /api/products/id (Update) (1 điểm)

e) Test DELETE /api/products/id (Delete) (1 điểm)

```
WebMvcTest(value = ProductController.class,
    excludeFilters = @ComponentScan.Filter(type = FilterType.
   ASSIGNABLE_TYPE,
        classes = {com.flogin.service.SecurityConfig.class, com.flogin.
   filter.JwtAuthenticationFilter.class}))
@AutoConfigureMockMvc(addFilters = false)
@DisplayName("Product API Integration Tests")
public class ProductControllerIntegrationTest {

    @Autowired
    private MockMvc mockMvc;

```

```
    @Autowired
    private ObjectMapper objectMapper;

    @MockitoBean
    private ProductService productService;


        @Test
        @DisplayName("1. Tạo product thành công với dữ liệu hợp lệ")
        void testCreateProduct_Success() throws Exception {
            // Arrange
            CreateProductRequest requestDto = new CreateProductRequest("Laptop", 15000000.0, "Gaming laptop", 10, "Electronics");
            ProductDto responseDto = new ProductDto(1L, "Electronics", 15000000.0, "Laptop", "Gaming laptop", 10);

            when(productService.createProduct(any(CreateProductRequest.class)))
                .thenReturn(responseDto);

            // Act & Assert
            mockMvc.perform(post("/api/products")
                    .contentType(MediaType.APPLICATION_JSON)
                    .content(objectMapper.writeValueAsString(requestDto))
                    .andExpect(status().isCreated()) // 201 Created
                    .andExpect(jsonPath("$.id").value(1))
                    .andExpect(jsonPath("$.productName").value("Laptop"))
                    .andExpect(jsonPath("$.price").value(15000000.0))
                    .andExpect(jsonPath("$.quantity").value(10))
                    .andExpect(jsonPath("$.category").value("Electronics"));

            verify(productService, times(1)).createProduct(any(CreateProductRequest.class));
        }

     /**
      * B) Test GET /api/products - Read All (1 điểm)
      * Test láy danh sách tát cá products
      */
        @Test
        @DisplayName("2. Láy danh sách products thành công")
        void testGetAllProducts_Success() throws Exception {
            // Arrange
            List<ProductDto> products = Arrays.asList(
                new ProductDto(1L, "Electronics", 15000000.0, "Laptop", "Gaming laptop", 10),
                new ProductDto(2L, "Electronics", 200000.0, "Mouse", "Wireless mouse", 50)
            );
            Page<ProductDto> page = new PageImpl<>(products, PageRequest.of(0, 10), products.size());

            when(productService.getAll(any()))
```

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

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

Listing 16: ProductControllerIntegrationTest.java

5 Câu 4: Mock Testing (10 điểm)

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

5.1.1 Frontend Mocking (2.5 điểm)

Mock external dependencies cho Login component:

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

```

  describe('Test 1: Mock authService.login() ', () => {

    test('nên mock authService.login() function', () => {
      // Verify authService.login ḍ ọc mock
      expect(typeof authService.login).toBe('function');
      expect(jest.isMockFunction(authService.login)).toBe(true);
    });
  });

```

Listing 17: Mock authService.loginUser()

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

```
  describe('Test 2: Mocked Successful/Failed Responses (1 ḍiém)',
   () => {

    test('nên handle mocked successful response', async () => {
      const user = userEvent.setup();
      authService.login.mockResolvedValue({
        success: true,
        message: 'Login successful',
        user: {
          id: '123',
          username: 'testuser',
          name: 'Test User'
        }
      });

   render(
  <MemoryRouter>
    <LoginForm />
  </MemoryRouter>
);

      const usernameInput = screen.getByPlaceholderText(/
   your_username/i);
      const passwordInput = screen.getByPlaceholderText(/
                           /i);
      const submitButton = screen.getByRole('button', { name: /Dăng
    Nhạp/i });

      await user.type(usernameInput, 'testuser');
      await user.type(passwordInput, 'password123');
      await user.click(submitButton);
```

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

Listing 18: mocked successful/failed responses

c) Verify mock calls (0.5 điểm)

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

```
  </MemoryRouter>
);

      const usernameInput = screen.getByPlaceholderText(/your_username/i);
      const passwordInput = screen.getByPlaceholderText(/i);
      const submitButton = screen.getByRole('button', { name: /Đăng Nhạp/i });

      await user.type(usernameInput, 'user123');
      await user.type(passwordInput, 'pass123');
      await user.click(submitButton);

      await waitFor(() => {
        // Verify mock đ ọc gọi vói đúng parameters
        expect(authService.login).toHaveBeenCalledWith('user123', 'pass123');
      });
    });

    test('nên verify mock đ ọc gọi exactly 1 time', async () => {
      const user = userEvent.setup();
      authService.login.mockResolvedValue({ success: true });

   render(
  <MemoryRouter>
    <LoginForm />
  </MemoryRouter>
);

      const usernameInput = screen.getByPlaceholderText(/your_username/i);
      const passwordInput = screen.getByPlaceholderText(/i);
      const submitButton = screen.getByRole('button', { name: /Đăng Nhạp/i });

      await user.type(usernameInput, 'testuser');
      await user.type(passwordInput, 'password123');
      await user.click(submitButton);

      await waitFor(() => {
        // Verify called exactly once
        expect(authService.login).toHaveBeenCalledTimes(1);
      });
    });

    test('nên verify mock NOT đ ọc gọi khi invalid data', async () => {
      const user = userEvent.setup();
      authService.login.mockResolvedValue({ success: true });

   render(
  <MemoryRouter>
    <LoginForm />
```

```
  </MemoryRouter>
);

      const submitButton = screen.getByRole('button', { name: /Đăng Nhập/i });

      // Submit form trống
      await user.click(submitButton);

      // Mock không nên được gọi
      expect(authService.login).not.toHaveBeenCalled();
    });

  });

```

Listing 19: Verify mock calls

5.1.2 Backend Mocking (2.5 điểm)

Mock dependencies trong Backend tests:

a) Mock AuthService với @MockBean (1 điểm)

b) Test controller với mocked service (1 điểm)

c) Verify mock interactions (0.5 điểm)

```
@WebMvcTest(value = AuthController.class,
    excludeFilters = @ComponentScan.Filter(type = FilterType.ASSIGNABLE_TYPE,
        classes = {com.flogin.service.SecurityConfig.class, com.flogin.filter.JwtAuthenticationFilter.class}))
@AutoConfigureMockMvc(addFilters = false)
@DisplayName("Backend Mocking - AuthController Mock Tests")
public class AuthControllerMockTest {

    @Autowired
    private MockMvc mockMvc;

    /**
     * a) Mock AuthService với @MockitoBean (1 điểm)
     * @MockitoBean tạo mock object và inject vào Spring context
     */
    @MockitoBean
    private AuthService authService;

    @Autowired
    ObjectMapper objectMapper;

        @Test
        @DisplayName("1. Mock: Controller với mocked service - Login thành công")
        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 thà
    nh 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("$.user.userName").value("
    testUser"));
40
41              // Verify
42              verify(authService, times(1)).authenticate(any(LoginRequest.
    class));
43
44          }
45
46
47          @Test
48          @DisplayName("2. Mock: Login thát bại - Username không tòn tại")
49          void testLoginWithMockedService_UserNotFound() throws Exception
    {
50              // Arrange: Mock service trá vè failure response
51              LoginResponse mockResponse = new LoginResponse(
52                  false,
53                  "Login thát bại vói user name không tòn tại"
54              );
55
56              when(authService.authenticate(any(LoginRequest.class)))
57                  .thenReturn(mockResponse);
58
59              // Act & Assert
60              mockMvc.perform(post("/api/auth/login")
61                      .contentType(MediaType.APPLICATION_JSON)
62                      .content("{\"userName\":\"nonexistent\",\"password
    \":\"Pass123\"}"))
63                      .andExpect(status().isUnauthorized())
64                      .andExpect(jsonPath("$.success").value(false))
65                      .andExpect(jsonPath("$.message").value("Login thát b
    ại vói user name không tòn tại"))
66                      .andExpect(jsonPath("$.token").doesNotExist())
67                      .andExpect(jsonPath("$.user").doesNotExist());
68
69              // Verify interactions
70              verify(authService, times(1)).authenticate(any(LoginRequest.
    class));
```

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

```
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 ợc gọi khi validation fail
    ")
127         void testVerify_MockNotCalledWhenValidationFails() throws
    Exception {
128             // Arrange: Không càn mock vì validation sẽ fail tr óc khi g
    ọi service
129 
130             // Act: Gúi request vói username tróng (validation fail)
131             mockMvc.perform(post("/api/auth/login")
132                     .contentType(MediaType.APPLICATION_JSON)
133                     .content("{\"userName\":\"\",\"password\":\"Pass123
    \"}"))
134                     .andExpect(status().isBadRequest());
135 
136             // c) Verify: authService.authenticate() KHÔNG ợc gọi vì
    validation fail
137             verify(authService, never()).authenticate(any(LoginRequest.
    class));
138             verifyNoInteractions(authService);
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:

a) Mock CRUD operations (1.5 điểm)

b) Test success và failure scenarios (0.5 điểm)

c) 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,
```

```
      name: 'Laptop Dell',
      productName: 'Laptop Dell', // Thêm tr òng này cho chác chán
      title: 'Laptop Dell',
      price: 15000000,
      quantity: 10,
      category: 'Electronics',
      description: 'Mô tá sán phám mãu'
    }
  ];

  beforeEach(() => {
    jest.clearAllMocks();
    productService.getAllProducts.mockResolvedValue(mockProductList
    );
  });

  // Helper function: Súa lại ḍé xú lý tr òng họp có nhiều nút cùng
     tên
  const fillAndSubmitForm = async (user) => {
    // 1. Mó Modal
    await user.click(screen.getByText(/Thêm Sán Phám/i));

    // 2. Ḍiền Form
    await user.type(screen.getByLabelText(/tên sán phám/i), 'New
    Laptop');
    await user.type(screen.getByLabelText(/giá/i), '20000000');
    await user.type(screen.getByLabelText(/só l ọng/i), '5');
    await user.selectOptions(screen.getByLabelText(/danh mục/i), '
    Electronics');
    await user.type(screen.getByLabelText(/mô tá/i), 'Mô tá test');

    // 3. Submit
    // Láy tát cá các nút có tên L u/Thêm/Cạp Nhạt
    const submitBtns = screen.getAllByRole('button', { name: /L u|
    Thêm|Cạp Nhạt/i });
    // Chọn nút cuói cùng (th òng là nút nàm trong Modal vùa mó ra)
    const submitBtn = submitBtns[submitBtns.length - 1];
    await user.click(submitBtn);
  };

  test('Mock: Get products (READ)', async () => {
    productService.getAllProducts.mockResolvedValue(mockProductList
    );
    render(<ProductManagement />);

    await waitFor(() => {
      expect(screen.getByText('Laptop Dell')).toBeInTheDocument();
      expect(productService.getAllProducts).toHaveBeenCalledTimes
    (1);
    });
  });

  test('Mock: Create product thanh cong', async () => {
    const user = userEvent.setup();
    const newProductResponse = { id: 2, name: 'New Laptop', price:
```

```
    20000000 };

    productService.createProduct.mockResolvedValue(newProductResponse);

    render(<ProductManagement />);
    await fillAndSubmitForm(user);

    await waitFor(() => {
      expect(productService.createProduct).toHaveBeenCalledWith(expect.objectContaining({
        productName: 'New Laptop',
        price: '20000000'
      }));

      expect(screen.getByText(/thành công/i)).toBeInTheDocument();
    });
  });

  test('Mock: Create product that bai (Failure Scenario)', async () => {
    const user = userEvent.setup();
    productService.createProduct.mockRejectedValue(new Error('Server Error'));

    render(<ProductManagement />);
    await fillAndSubmitForm(user);

    await waitFor(() => {
      expect(productService.createProduct).toHaveBeenCalledTimes(1);
      expect(screen.getByText(/Server Error/i)).toBeInTheDocument();
    });
  });

  test('Mock: Delete product', async () => {
    const user = userEvent.setup();
    productService.deleteProduct.mockResolvedValue({ success: true });

    render(<ProductManagement />);
    await waitFor(() => screen.getByText('Laptop Dell'));

    const deleteBtns = screen.getAllByTitle(/Xóa/i);
    await user.click(deleteBtns[0]);

    const confirmBtn = await screen.findByText('Xóa', { selector: 'button' });
    await user.click(confirmBtn);

    await waitFor(() => {
      expect(productService.deleteProduct).toHaveBeenCalledWith(1);
    });
  });

```

```
  test('Mock: Update product', async () => {
    const user = userEvent.setup();
    productService.updateProduct.mockResolvedValue({ success: true
   });

    render(<ProductManagement />);
    await waitFor(() => screen.getByText('Laptop Dell'));

    const editBtns = screen.getAllByTitle(/Chính sửa/i);
    await user.click(editBtns[0]);

    // 2. Sửa giá
    const priceInput = screen.getByLabelText(/giá/i);
    await user.clear(priceInput);
    await user.type(priceInput, '18000000');

    const submitBtns = screen.getAllByRole('button', { name: /L u|C
   ập nhật/i });
    await user.click(submitBtns[submitBtns.length - 1]);

    await waitFor(() => {
      expect(productService.updateProduct).toHaveBeenCalledWith(1,
   expect.objectContaining({
        price: '18000000'
      }));
    });
  });
});

```

Listing 21: validateUsername

5.2.2 Backend Mocking (2.5 điểm)

Mock ProductRepository trong service tests:

a) Mock ProductRepository (1 điểm)

b) Test service layer với mocked repository (1 điểm)

c) Verify repository interactions (0.5 điểm)

```
@ExtendWith(MockitoExtension.class)
@DisplayName("Mock ProductRepository trong Service Tests (2.5 điém)")
public class ProductServiceMockTest {

    /**
     * a) Mock ProductRepository (1 điém)
     */
    @Mock
    private ProductRepository productRepository;

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

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

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

```
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

```
/**
 * Cypress Configuration File
 * @see https://docs.cypress.io/guides/references/configuration
 */
const { defineConfig } = require('cypress');

module.exports = defineConfig({
  /**
   * E2E Testing Configuration
   *
   * Configures end-to-end tests that run against the full application
   * in a real browser environment.
   */
  e2e: {
    /**
     * Base URL for all cy.visit() and cy.request() commands
     * Points to local development server
     * Change to production URL for production testing
     */
    baseUrl: 'http://localhost:5173',

    /**
     * Default viewport dimensions for test browser
     * Simulates desktop screen resolution
     */
    viewportWidth: 1280,
    viewportHeight: 720,

    /**
     * Time to wait for page to load before failing
     * Default: 60000ms (60 seconds)
     */
    pageLoadTimeout: 30000,

    /**
     * Time to wait for cy.request() to resolve
     * Used for API calls in tests
     */
    requestTimeout: 15000,

    /**
     * Time to wait for response in cy.wait()
     * Used for network request assertions
     */
    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     /**
```

```
     * Retry failed tests to reduce flakiness
     * - runMode: Retries in headless mode (CI/CD)
     * - openMode: Retries in interactive mode (development)
     */
    retries: {
      runMode: 1,    // Retry once in CI/CD pipeline
      openMode: 0    // No retries in interactive mode
    }
  },

  /**
   * Component Testing Configuration (Optional)
   *
   * Tests individual React components in isolation
   * using Vite dev server for fast feedback.
   */
  component: {
    /**
     * Dev server configuration for component testing
     * Uses Vite for fast component mounting and HMR
     */
    devServer: {
      framework: 'react',  // Framework being tested
      bundler: 'vite',     // Build tool (Vite for fast builds)
    },

    /**
     * Pattern to match component test files
     * Example: Button.cy.jsx, Form.cy.jsx
     */
    specPattern: 'src/tests/cypress/component/**/*.cy.jsx',

    /**
     * Support file for component tests
     * Contains component-specific setup and commands
     */
    supportFile: 'src/tests/cypress/support/component.js'
  }
});


```

Listing 23: test environment

- Setup Page Object Model

```
          class LoginPage {
  // All element getters return Cypress chainable elements

  /**
   * Get username input field
   * @returns {Cypress.Chainable} Username input element
   */
  get usernameInput() {
    return cy.get('input[type="text"]').first();
  }
```

```
11
12    get passwordInput() {
13      return cy.get('input[type="password"]');
14    }
15
16    get loginButton() {
17      return cy.get("button").contains(/login|đăng nhạp/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 nhạp/i);
50    }
51
52    get passwordVisibilityToggle() {
53      return cy.get('button[type="button"]').contains("    í    ");
54    }
55
56    // Methods
57    /**
58     * Navigate to login page
59     */
60    navigateToLoginPage() {
61      cy.visit("/");
62      this.pageTitle.should("be.visible");
63      return this;
64    }
65
```

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

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

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

```
    return this;
  }

  /**
   * Get username input value
   */
  getUsernameValue() {
    return this.usernameInput.invoke("val");
  }

  /**
   * Get password input type
   */
  getPasswordInputType() {
    return this.passwordInput.invoke("attr", "type");
  }

  /**
   * Verify username input is visible
   */
  verifyUsernameInputVisible() {
    this.usernameInput.should("be.visible");
    return this;
  }

  /**
   * Verify password input is visible
   */
  verifyPasswordInputVisible() {
    this.passwordInput.should("be.visible");
    return this;
  }

  /**
   * Verify all form elements are visible
   */
  verifyFormElementsVisible() {
    this.verifyPageTitle();
    this.usernameInput.should("be.visible");
    this.passwordInput.should("be.visible");
    this.loginButton.should("be.visible");
    return this;
  }

  /**
   * Verify username input is empty
   */
  verifyUsernameEmpty() {
    this.usernameInput.should("have.value", "");
    return this;
  }

  /**
   * Verify password input is empty
   */
  verifyPasswordEmpty() {
```

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

Listing 24: Setup Page Object Model

6.1.2 E2E Test Scenarios cho Login (2.5 điểm)

a) Test complete login flow (1 điểm)

b) Test validation messages (0.5 điểm)

c) Test success/error flows (0.5 điểm)

d) Test UI elements interactions (0.5 điểm)

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

```
17
18      it("SC1.2: Ng òi dùng có thé nhạp username hợp lẹ", () => {
19        const username = "test";
20        LoginPage.enterUsername(username);
21        LoginPage.verifyUsernameValue(username);
22      });
23
24      it("SC1.3: Ng òi dùng có thé nhạp password hợp lẹ", () => {
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 vói data hợp lẹ", ()
    => {
31        LoginPage.login("test", "Test123@");
32        // Form submitted - có thé verify bàng 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 hoạc spinner
46        cy.wait(500);
47      });
48
49      it("SC1.7: Ng òi dùng có thé clear username và nhạp 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 nhạp lại", () =>
    {
58        LoginPage.enterPassword("Test123@");
59        LoginPage.clearPassword();
60        LoginPage.passwordInput.should("have.value", "");
61        LoginPage.enterPassword("NewPass456@");
62        LoginPage.passwordInput.should("have.value", "NewPass456@");
63      });
64
65      it("SC1.9: Complete flow: Input -> Verify -> Clear -> Resubmit
    ", () => {
66        // First attempt
67        LoginPage.enterUsername("user1").enterPassword("Pass123@").
```

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

```
      // username input should still have invalid value
      LoginPage.usernameInput.should("have.value", "invalid-username");
    });

    it("SC2.5: Validation với password yếu", () => {
      LoginPage.enterUsername("test");
      LoginPage.enterPassword("123");
      LoginPage.clickLogin();
      cy.wait(300);
      // Short password validation
    });

    it("SC2.6: Validation messages appear immediately", () => {
      LoginPage.clickLogin();
      cy.wait(200);
      // Check for validation feedback
    });

    it("SC2.7: Can correct validation errors and resubmit", () => {
      // Submit with empty form
      LoginPage.clickLogin();
      cy.wait(300);

      // Now fill and submit
      LoginPage.enterUsername("test").enterPassword("Test123@").clickLogin();
    });

    it("SC2.8: Multiple validation errors show correctly", () => {
      LoginPage.clearAllInputs();
      LoginPage.clickLogin();
      cy.wait(300);
      // Both fields empty - verify validation
    });
  });

  /**
   * SCENARIO 3: Success/Error Flows
   */
  describe("Scenario 3: Success/Error Flows (0.5 điểm)", () => {
    it("SC3.1: Successful login dengan valid credentials", () => {
      cy.fixture("users").then((users) => {
        LoginPage.login(users.validUser.username, users.validUser.password);
        cy.wait(500);
      });
    });

    it("SC3.2: Failed login dengan invalid credentials", () => {
      cy.fixture("users").then((users) => {
        LoginPage.login(users.invalidUser.username, users.invalidUser.password);
        cy.wait(500);
      });
    });
```

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

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

```
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 nhạp/i);
284       });
285     });
286
287     it("SC4.10: Form is responsive to interactions", () => {
288       // Click username
289       LoginPage.usernameInput.click();
290       LoginPage.usernameInput.should("be.focused");
291
292       // Type in username
293       LoginPage.usernameInput.type("test");
294       LoginPage.usernameInput.should("have.value", "test");
295
296       // Click password
297       LoginPage.passwordInput.click();
298       LoginPage.passwordInput.should("be.focused");
299
300       // Type in password
301       LoginPage.passwordInput.type("pass");
302       LoginPage.passwordInput.should("have.value", "pass");
303     });
304   });
305 });
306
307 export { LoginPage };
308
309
```

Listing 25: Test Scenarios cho Login

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

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

```yaml
jobs:
  test:
    runs-on: ubuntu-latest
    timeout-minutes: 15 # Tự hủy nếu treo quá 15p

    strategy:
      matrix:
        node-version: [20.x] # Bắt buộc dùng Node 20 cho Vite mới
      fail-fast: false

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Setup Node.js ${{ matrix.node-version }}
        uses: actions/setup-node@v4
        with:
          node-version: ${{ matrix.node-version }}
          # Tạm thời tắt cache để tránh xung đột file lock cũ
          # cache: "npm"
          # cache-dependency-path: "frontend/package-lock.json"

      - name: Install dependencies
        working-directory: ./frontend
        run: |
          # Xóa file lock cũ và node_modules để tránh xung đột
          rm -rf node_modules package-lock.json
          # Dùng npm install (thay vì npm ci) để nó tự tìm gói Linux
          npm install

      # --- JEST TEST ---
      - name: Run Integration Tests (Jest)
        id: run-jest  # Đã thêm ID để dùng cho báo cáo
        working-directory: ./frontend
        run: npm test -- --coverage --testPathPattern="Integration|Mock" --passWithNoTests

      - name: Upload Jest Coverage Reports
        uses: actions/upload-artifact@v4
        if: always()
        with:
          name: jest-coverage-${{ matrix.node-version }}
          path: frontend/coverage/
          retention-days: 15

      # --- BUILD & E2E ---
      - name: Build frontend
        working-directory: ./frontend
        run: npm run build

      - name: Start dev server & Wait
        working-directory: ./frontend
        run: |
          npm run dev &
          npx wait-on http://localhost:5173 --timeout 60000

```

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

Listing 26: CI/CD Integration cho Login Tests

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

6.2.1 Setup Page Object Model (1 điểm)

Implement POM cho Product pages:

```
    frontend/src/tests/cypress/pages/
              ProductPage.js

```

Listing 27: Implement POM cho Product page

6.2.2 E2E Test Scenarios cho Product (2.5 điểm)

Viết automated tests cho CRUD operations:

a) Test Create product flow (0.5 điểm)

b) Test Read/List products (0.5 điểm)

c) Test Update product (0.5 điểm)

d) Test Delete product (0.5 điểm)

e) Test Search/Filter functionality (0.5 điểm)

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

```
 *
 * Usage Example:
 * ```javascript
 * import ProductPage from './pages/ProductPage';
 *
 * describe('Product Management Tests', () => {
 *   it('should create new product', () => {
 *     ProductPage
 *       .navigateToProductPage()
 *       .createProduct({
 *         name: 'iPhone 15',
 *         price: '999.99',
 *         quantity: '50',
 *         category: 'Electronics',
 *         description: 'Latest iPhone model'
 *       })
 *       .verifySuccessNotification('Product created successfully')
 *       .verifyProductExists('iPhone 15');
 *   });
 *
 *   it('should update product', () => {
 *     ProductPage
 *       .updateProduct('iPhone 15', { price: '1099.99' })
 *       .verifySuccessNotification('Product updated');
 *   });
 *
 *   it('should delete product', () => {
 *     ProductPage
 *       .deleteProduct('iPhone 15')
 *       .verifyProductNotExists('iPhone 15');
 *   });
 * });
 * ```
 *
 * @class ProductPage
 * @author Software Testing Team
 * @version 1.0
 * @since 2025-11-26
 */
class ProductPage {
  // ============ HEADER & SEARCH SELECTORS ============
  // Elements in the page header and search section

  /**
   * Get page title element
   * @returns {Cypress.Chainable} Page title heading
   */
  get pageTitle() {
    return cy.contains('h1', /quán lý sán phám|product management/i
    );
  }

  get addProductButton() {
    return cy.contains('button', /thêm sán phám|add product/i);
  }

```

```
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   // ============ ACTION BUTTONS IN TABLE ============
110   getViewButton(productName) {
111     return this.getProductByName(productName).within(() => {
112       return cy.get('button.blue');
113     });
114   }
115
116   getEditButton(productName) {
117     return this.getProductByName(productName).within(() => {
118       return cy.get('button.green');
119     });
120   }
121
122   getDeleteButton(productName) {
123     return this.getProductByName(productName).within(() => {
124       return cy.get('button.red');
125     });
126   }
127
128   // ============ PAGINATION SELECTORS ============
129   get paginationPrevButton() {
130     return cy.get('.pagination-btn').first();
131   }
132
133   get paginationNextButton() {
134     return cy.get('.pagination-btn').last();
135   }
136
137   get paginationInfo() {
138     return cy.get('.pagination-info');
```

```
  }

  // ============ MODAL SELECTORS ============
  get modal() {
    return cy.get('.modal');
  }

  get modalTitle() {
    return cy.get('.modal-title');
  }

  get modalCloseButton() {
    return cy.get('.modal-close');
  }

  get modalOverlay() {
    return cy.get('.modal-overlay');
  }

  // ============ FORM INPUTS ============
  get productNameInput() {
    return cy.get('#name-input');
  }

  get priceInput() {
    return cy.get('#price-input');
  }

  get quantityInput() {
    return cy.get('#quantity-input');
  }

  get categorySelect() {
    return cy.get('#category-select');
  }

  get descriptionInput() {
    return cy.get('#description-textarea');
  }

  // ============ FORM ERROR MESSAGES ============
  get nameError() {
    return cy.contains('p.error-message', /tên sán phám|name/i);
  }

  get priceError() {
    return cy.contains('p.error-message', /giá|price/i);
  }

  get quantityError() {
    return cy.contains('p.error-message', /só l ọng|quantity/i);
  }

  get categoryError() {
    return cy.contains('p.error-message', /danh mục|category/i);
  }
```

```
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', /chính 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:
      100 });
277     }
278     if (productData.price) {
279       this.priceInput.clear().type(productData.price, { delay: 100
     });
280     }
281     if (productData.quantity) {
282       this.quantityInput.clear().type(productData.quantity, { delay
     : 100 });
283     }
284     if (productData.category) {
285       this.categorySelect.select(productData.category);
286     }
287     if (productData.description) {
288       this.descriptionInput.clear().type(productData.description, {
      delay: 100 });
289     }
290     return this;
291   }
292
293   /**
294    * Submit product form
295    */
296   submitForm() {
297     this.submitButton.click();
298     cy.wait(500);
299     return this;
300   }
301
```

```
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
      ('not.exist');
427     return this;
428   }
429
430   /**
431    * Verify success notification
432    */
433   verifySuccessNotification(message) {
434     cy.get('.notification.success', { timeout: 10000 }).should('be.
      visible');
435     if (message) {
436       if (message instanceof RegExp) {
437         cy.get('.notification').invoke('text').should('match',
      message);
438       } else {
439         cy.get('.notification').should('contain', message);
440       }
441     }
442     return this;
443   }
444
445   /**
446    * Verify error notification
447    */
448   verifyErrorNotification(message) {
449     cy.get('.notification.error', { timeout: 10000 }).should('be.
      visible');
450     if (message) {
451       if (message instanceof RegExp) {
452         cy.get('.notification').invoke('text').should('match',
      message);
453       } else {
454         cy.get('.notification').should('contain', message);
455       }
456     }
457     return this;
458   }
459
460   /**
461    * Verify error message for field
462    */
463   verifyFieldError(fieldName) {
464     let errorSelector;
```

```
    switch(fieldName.toLowerCase()) {
      case 'name':
      case 'tên':
        errorSelector = this.nameError;
        break;
      case 'price':
      case 'giá':
        errorSelector = this.priceError;
        break;
      case 'quantity':
      case 'số l ợng':
        errorSelector = this.quantityError;
        break;
      case 'category':
      case 'danh mục':
        errorSelector = this.categoryError;
        break;
      case 'description':
      case 'mô tả':
        errorSelector = this.descriptionError;
        break;
      default:
        throw new Error(`Unknown field: ${fieldName}`);
    }
    errorSelector.should('be.visible');
    return this;
  }

  /**
   * Get total product count
   */
  getTotalProductCount() {
    return this.tableRows.its('length');
  }

  /**
   * Verify pagination visible
   */
  verifyPaginationVisible() {
    this.paginationInfo.should('be.visible');
    return this;
  }

  /**
   * Click next page
   */
  clickNextPage() {
    this.paginationNextButton.click();
    cy.wait(200);
    return this;
  }

  /**
   * Click previous page
   */
  clickPreviousPage() {
```

```
    this.paginationPrevButton.click();
    cy.wait(200);
    return this;
  }

  /**
   * Verify table empty
   */
  verifyTableEmpty() {
    // Check if empty state element exists or table shows empty
   message
    cy.get('.product-table tbody tr').then($rows => {
      expect($rows.text()).to.include('Không có sán phám');
    });
    return this;
  }

  /**
   * Verify table not empty
   */
  verifyTableNotEmpty() {
    cy.get('.product-table tbody tr').then($rows => {
      expect($rows.text()).not.to.include('Không có sán phám');
    });
    return this;
  }

  /**
   * Get product row data
   */
  getProductRowData(productName) {
    return this.getProductByName(productName).within(() => {
      return {
        name: cy.get('td').eq(0),
        price: cy.get('td').eq(1),
        quantity: cy.get('td').eq(2),
        category: cy.get('td').eq(3)
      };
    });
  }
}

// Export page object
export default new ProductPage();


```

Listing 28: Test Scenarios cho Product

6.2.3 CI/CD Integration (1.5 điểm)

Setup complete CI/CD pipeline:

```
name: FloginFE_BE CI/CD Pipeline

```

```yaml
on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]

env:
  JAVA_VERSION: "21"
  NODE_VERSION: "20"
  BACKEND_PORT: 6969
  SQL_SERVER_SA_PASSWORD: "YourStrong!Passw0rd@"

permissions:
  contents: read
  checks: write
  pull-requests: write
  statuses: write

jobs:
  # JOB 1: Backend Build & Test

  backend-tests:
    name: Backend - Build & Test
    runs-on: ubuntu-latest

    services:
      sqlserver:
        image: mcr.microsoft.com/mssql/server:2022-latest
        env:
          ACCEPT_EULA: Y
          SA_PASSWORD: ${{ env.SQL_SERVER_SA_PASSWORD }}
          MSSQL_PID: Developer
        ports:
          - 1433:1433
        options: >-
          --user 0:0
          --health-cmd "/opt/mssql-tools18/bin/sqlcmd -S localhost -U sa -P 'YourStrong!Passw0rd@' -C -Q 'SELECT 1' || exit 1"
          --health-interval 10s
          --health-timeout 5s
          --health-retries 10
          --health-start-period 10s

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name:     Setup Java ${{ env.JAVA_VERSION }}
        uses: actions/setup-java@v3
        with:
          java-version: ${{ env.JAVA_VERSION }}
          distribution: "temurin"
          cache: "maven"

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

```yaml

  frontend-tests:
    name: Frontend - Build & Test
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Setup Node.js ${{ env.NODE_VERSION }}
        uses: actions/setup-node@v3
        with:
          node-version: ${{ env.NODE_VERSION }}
          # Tạm thời tắt cache để tránh restore file lock cũ
          # cache: "npm"
          # cache-dependency-path: frontend/package-lock.json

      # FIX 2: Xóa package-lock.json để ép npm tìm gói Linux
      - name: Install Frontend Dependencies
        run: |
          cd frontend
          rm -rf node_modules package-lock.json
          npm install

      - name: Run ESLint
        run: |
          cd frontend
          npm run lint || true
        continue-on-error: true

      - name: Run Frontend Unit Tests
        run: |
          cd frontend
          npm test -- --coverage --watchAll=false
        env:
          CI: true

      - name: Upload Frontend Coverage to Codecov
        uses: codecov/codecov-action@v3
        with:
          token: ${{ secrets.CODECOV_TOKEN }}
          files: ./frontend/coverage/lcov.info
          flags: frontend
          name: frontend-coverage
          fail_ci_if_error: false

      - name: Build Frontend
        run: |
          cd frontend
          npm run build
        env:
          CI: false
          REACT_APP_API_URL: http://localhost:6969

      - name: Upload Frontend Build Artifacts
        uses: actions/upload-artifact@v4
```

```yaml
        with:
          name: frontend-build
          path: frontend/build/

      - name: Upload Frontend Test Results
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: frontend-test-results
          path: |
            frontend/coverage/


  # JOB 3: E2E Tests with SQL Server

  e2e-tests:
    name: E2E Tests
    runs-on: ubuntu-latest
    needs: [backend-tests, frontend-tests]

    services:
      sqlserver:
        image: mcr.microsoft.com/mssql/server:2022-latest
        env:
          ACCEPT_EULA: Y
          SA_PASSWORD: ${{ env.SQL_SERVER_SA_PASSWORD }}
          MSSQL_PID: Developer
        ports:
          - 1433:1433
        options: >-
          --user 0:0
          --health-cmd "/opt/mssql-tools18/bin/sqlcmd -S localhost -U sa -P 'YourStrong!Passw0rd@' -C -Q 'SELECT 1' || exit 1"
          --health-interval 10s
          --health-timeout 10s
          --health-retries 20
          --health-start-period 60s

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name:     Setup Java ${{ env.JAVA_VERSION }}
        uses: actions/setup-java@v3
        with:
          java-version: ${{ env.JAVA_VERSION }}
          distribution: "temurin"
          cache: "maven"

      - name: Setup Node.js ${{ env.NODE_VERSION }}
        uses: actions/setup-node@v3
        with:
          node-version: ${{ env.NODE_VERSION }}
          # cache: "npm"
          # cache-dependency-path: frontend/package-lock.json

```

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

```
          PORT: 5173
          CI: false

      - name:  Run E2E Tests (if exists)
        run: |
          cd frontend
          if [ -f "package.json" ] && grep -q "test:e2e" package.json; then
            npm run test:e2e || echo "E2E tests not configured yet"
          else
            echo "E2E tests not found, skipping..."
          fi
        continue-on-error: true

      - name:  Stop Servers
        if: always()
        run: |
          if [ -f backend/backend.pid ]; then
            kill $(cat backend/backend.pid) || true
          fi
          if [ -f frontend/frontend.pid ]; then
            kill $(cat frontend/frontend.pid) || true
          fi

      - name: Upload E2E Test Results
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: e2e-test-results
          path: |
            frontend/cypress/screenshots/
            frontend/cypress/videos/
            frontend/playwright-report/


  # JOB 4: Performance Tests (k6) with SQL Server

  performance-tests:
    name: Performance Tests (k6)
    runs-on: ubuntu-latest
    needs: [backend-tests]
    if: github.event_name == 'push' && github.ref == 'refs/heads/main'

    services:
      sqlserver:
        image: mcr.microsoft.com/mssql/server:2022-latest
        env:
          ACCEPT_EULA: Y
          SA_PASSWORD: ${{ env.SQL_SERVER_SA_PASSWORD }}
          MSSQL_PID: Developer
        ports:
          - 1433:1433
        options: >-
          --user 0:0
          --health-cmd "/opt/mssql-tools18/bin/sqlcmd -S localhost -U sa -P 'YourStrong!Passw0rd@' -C -Q 'SELECT 1' || exit 1"
```

```
          --health-interval 10s
          --health-timeout 10s
          --health-retries 20
          --health-start-period 60s

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name:  Setup Java ${{ env.JAVA_VERSION }}
        uses: actions/setup-java@v3
        with:
          java-version: ${{ env.JAVA_VERSION }}
          distribution: "temurin"
          cache: "maven"

      - name:  Create Performance Database
        run: |
          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 "CREATE DATABASE STDatabase_Perf;"

      - name:  Start Backend Server
        run: |
          cd backend
          chmod +x mvnw
          ./mvnw clean install -DskipTests
          nohup ./mvnw spring-boot:run \
            -Dspring-boot.run.arguments="--spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=STDatabase_Perf;trustServerCertificate=true --spring.datasource.username=sa --spring.datasource.password=${{ env.SQL_SERVER_SA_PASSWORD }}" &
          echo $! > backend.pid

          # Wait for backend
          for i in {1..60}; do
            if curl -s http://localhost:6969/actuator/health > /dev/null 2>&1; then
              echo "Backend is ready!"
              break
            fi
            sleep 5
          done

      - name: Install k6
        run: |
          sudo gpg -k
          sudo gpg --no-default-keyring --keyring /usr/share/keyrings/k6-archive-keyring.gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys C5AD17C747E3415A3642D57D77C6C491D6AC1D69
          echo "deb [signed-by=/usr/share/keyrings/k6-archive-keyring.gpg] https://dl.k6.io/deb stable main" | sudo tee /etc/apt/sources.list.d/k6.list
          sudo apt-get update
```

```
          sudo apt-get install k6

      - name: Run Performance Tests (100 users - 1 minute for CI)
        run: |
          cd performance-tests/scripts
          # Shortened test duration for CI (1 min instead of 5 min)
          k6 run login-test.js --vus 100 --duration 1m --out json=../results/ci-login-100-users.json
        continue-on-error: true

      - name: Analyze Performance Results
        run: |
          cd performance-tests
          if [ -f results/ci-login-100-users.json ]; then
            echo "Performance test completed. Results saved."
            # Extract basic metrics
            echo "=== Performance Summary ==="
            grep -o '"http_req_duration"[^}]*' results/ci-login-100-users.json | head -10 || true
          fi

      - name: Stop Backend
        if: always()
        run: |
          if [ -f backend/backend.pid ]; then
            kill $(cat backend/backend.pid) || true
          fi

      - name: Upload Performance Results
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: performance-test-results
          path: performance-tests/results/


  # JOB 6: Code Quality Analysis

  code-quality:
    name: Code Quality Analysis
    runs-on: ubuntu-latest
    needs: [backend-tests, frontend-tests]

    steps:
      - name: Checkout code
        uses: actions/checkout@v3
        with:
          fetch-depth: 0

      - name:     Setup Java ${{ env.JAVA_VERSION }}
        uses: actions/setup-java@v3
        with:
          java-version: ${{ env.JAVA_VERSION }}
          distribution: "temurin"

      - name: SonarCloud Scan (if configured)
```

```
        run: |
          echo "SonarCloud scan would run here if configured"
          echo "Requires SONAR_TOKEN secret"
        continue-on-error: true

      - name: Generate Coverage Report Summary
        run: |
          echo "## Test Coverage Summary" > coverage-summary.md
          echo "" >> coverage-summary.md
          echo "### Backend Coverage" >> coverage-summary.md
          echo "- Unit Tests:    " >> coverage-summary.md
          echo "- Integration Tests:    " >> coverage-summary.md
          echo "" >> coverage-summary.md
          echo "### Frontend Coverage" >> coverage-summary.md
          echo "- Component Tests:    " >> coverage-summary.md

      - name: Upload Coverage Summary
        uses: actions/upload-artifact@v4
        with:
          name: coverage-summary
          path: coverage-summary.md


  # JOB 7: Build Summary & Notification

  build-summary:
    name: Build Summary
    runs-on: ubuntu-latest
    needs:
      [
        backend-tests,
        frontend-tests,
        e2e-tests,
        performance-tests,
        code-quality,
      ]
    if: always()

    steps:
      - name: Generate Build Summary
        run: |
          echo "## CI/CD Pipeline Summary" > summary.md
          echo "" >> summary.md
          echo "### Jobs Status:" >> summary.md
          echo "- Backend Tests (SQL Server): ${{ needs.backend-tests.
    result }}" >> summary.md
          echo "- Frontend Tests: ${{ needs.frontend-tests.result }}" >>
     summary.md
          echo "- E2E Tests: ${{ needs.e2e-tests.result }}" >> summary.
    md
          echo "- Performance Tests: ${{ needs.performance-tests.result
    }}" >> summary.md
          echo "- Security Scan: ${{ needs.security-scan.result }}" >>
    summary.md
          echo "- Code Quality: ${{ needs.code-quality.result }}" >>
    summary.md
```

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

Listing 29: .github/workflows/cicd.yml

7 Phần Mở Rộng (Bonus 20 điểm)

7.1 Performance Testing (10 điểm)

7.1.1 Yêu cầu

a) Setup JMeter hoặc k6 cho performance testing (2 điểm)

b) Viết performance tests cho Login API (3 điểm):

c) Viết performance tests cho Product API (3 điểm)

d) Phân tích kết quả và đưa ra recommendations (2 điểm)

7.2 Security Testing (10 điểm)

7.2.1 Yêu cầu

a) Test common vulnerabilities (5 điểm):

- SQL Injection

```
        @Test
    @DisplayName("TC1.1: SQL Injection - Login bypass với ' OR '1'='1")
    void testSqlInjection_LoginBypass_Classic() throws Exception {
        LoginRequest maliciousRequest = new LoginRequest("admin' OR '1'='1", "anything");

        mockMvc.perform(post("/api/auth/login")
                .contentType(MediaType.APPLICATION_JSON)
                .content(objectMapper.writeValueAsString(maliciousRequest)))
                .andExpect(status().isBadRequest())
                .andExpect(jsonPath("$.success").value(false));
    }

    @Test
    @DisplayName("TC1.2: SQL Injection - Login bypass với ' OR 1=1--")
    void testSqlInjection_LoginBypass_Comment() throws Exception {
        LoginRequest maliciousRequest = new LoginRequest("admin' OR 1=1--", "anything");

        mockMvc.perform(post("/api/auth/login")
                .contentType(MediaType.APPLICATION_JSON)
                .content(objectMapper.writeValueAsString(maliciousRequest)))
                .andExpect(status().isBadRequest())
                .andExpect(jsonPath("$.success").value(false));
    }

    @Test
```

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

Listing 30: SqlInjectionTest.java

- Cross-Site Scripting (XSS)

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

```
                .content(objectMapper.writeValueAsString(xssRequest)))
                .andExpect(status().isBadRequest())
                .andExpect(jsonPath("$.errors.productName").exists());
    }


```

Listing 31: XssTest.java

- Authentication bypass attempts

```
        @Test
    @DisplayName("TC1.3: Auth bypass - Empty credentials")
    void testAuthBypass_EmptyCredentials() throws Exception {
        LoginRequest loginRequest = new LoginRequest("", "");

        mockMvc.perform(post("/api/auth/login")
                .contentType(MediaType.APPLICATION_JSON)
                .content(objectMapper.writeValueAsString(loginRequest)))
                .andExpect(status().isBadRequest());
    }

    @Test
    @DisplayName("TC2.1: Auth bypass - Wrong password")
    void testAuthBypass_WrongPassword() throws Exception {
        LoginRequest loginRequest = new LoginRequest("admin", "wrongpassword");

        mockMvc.perform(post("/api/auth/login")
                .contentType(MediaType.APPLICATION_JSON)
                .content(objectMapper.writeValueAsString(loginRequest)))
                .andExpect(status().isBadRequest()) // Password validation fails (no digit)
                .andExpect(jsonPath("$.success").value(false));
    }

    @Test
    @DisplayName("TC4.1: Auth bypass - Invalid JWT token")
    void testAuthBypass_InvalidToken() throws Exception {
        mockMvc.perform(get("/api/products")
                .header("Authorization", "Bearer invalid.token.here"))
                .andExpect(status().isForbidden());
    }


```

Listing 32: AuthBypassTest.java

b) Test input validation và sanitization (3 điểm)

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

```
    @Test
     @DisplayName("TC1.1: Validation - Product name quá ngán (< 3 chars)")
     void testValidation_ProductName_TooShort() throws Exception {
         CreateProductRequest request = new CreateProductRequest("AB", 100.0, "Test", 10, "Electronics");

         mockMvc.perform(post("/api/products")
                 .header("Authorization", authToken)
                 .contentType(MediaType.APPLICATION_JSON)
                 .content(objectMapper.writeValueAsString(request)))
                 .andExpect(status().isBadRequest())
                 .andExpect(jsonPath("$.errors.productName").exists());
     }


     @Test
     @DisplayName("TC6.1: Sanitization - Special characters trong product name")
     void testSanitization_SpecialCharacters_ProductName() throws Exception {
         CreateProductRequest request = new CreateProductRequest("Product <>&\"'", 100.0, "Test", 10, "Electronics");

         mockMvc.perform(post("/api/products")
                 .header("Authorization", authToken)
                 .contentType(MediaType.APPLICATION_JSON)
                 .content(objectMapper.writeValueAsString(request)))
                 .andExpect(status().isBadRequest())
                 .andExpect(jsonPath("$.errors.productName").exists());

     }


```

Listing 33: InputValidationTest.java

c) Security best practices implementation (2 điểm):

```
      @SpringBootTest
@AutoConfigureMockMvc
@DisplayName("Security Best Practices Tests")
public class SecurityBestPracticesTest {
    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private UserRepository userRepository;

    @Test
```

```
    @DisplayName("1. Password Hashing - Mật khẩu được hash bằng BCrypt")
    void testPasswordHashing() {
        // Arrange
        String rawPassword = "mySecurePassword123";

        // Act - Hash password
        String hashedPassword = passwordEncoder.encode(rawPassword);

        // Assert
        assertNotNull(hashedPassword, "Hashed password không được null");
        assertNotEquals(rawPassword, hashedPassword, "Password không được lưu dạng plain text");
        assertTrue(hashedPassword.startsWith("$2a$") || hashedPassword.startsWith("$2b$"),
                "Password phải được hash bằng BCrypt");
        assertTrue(hashedPassword.length() >= 60, "BCrypt hash phải có độ dài >= 60 ký tự");

        // Verify password matching
        boolean matches = passwordEncoder.matches(rawPassword, hashedPassword);
        assertTrue(matches, "Password encoder phải verify được password đúng");

        // Verify wrong password doesn't match
        boolean wrongMatch = passwordEncoder.matches("wrongPassword", hashedPassword);
        assertFalse(wrongMatch, "Password encoder phải reject password sai");

    }


    // 3. CORS CONFIGURATION TEST

    @Test
    @DisplayName("3. CORS Configuration - CORS headers được cấu hình đúng")
    void testCorsConfiguration() throws Exception {
        // Arrange
        String origin = "http://localhost:5173";

        // Act - Preflight request (OPTIONS)
        mockMvc.perform(options("/api/products")
                        .header("Origin", origin)
                        .header("Access-Control-Request-Method", "POST")
                        .header("Access-Control-Request-Headers", "Content-Type,Authorization"))

                // Assert
                .andExpect(status().isOk())
                .andExpect(header().exists("Access-Control-Allow-
```

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

Listing 34: SecurityBestPracticesTest.java