PHN M RNG: PERFORMANCE TESTING VỊ SECURITY TESTING

Phn nịy thc hin 2 loi kim th nóng cao: **Performance Testing** ònh giò kh nng chu ti vị hiu nng ca h thng, vị **Security Testing** kim tra còc l hng bo mt. C hai u lị yổu cu quan trng trong phòt trin phn mm chuyổn nghip.

7 Performance Testing

7.1 Yổu cu vị Mc tiổu

Theo yổu cu ca bịi tp ln, nhúm cn thc hin:

1. Setup cũng c kim th hiu nng (JMeter hoc k6)
2. Vit performance tests cho Login API:
   - Load test: 100, 500, 1000 concurrent users
   - Stress test: Tơm breaking point
   - Response time analysis
3. Vit performance tests cho Product API
4. Phón tòch kt qu vị a ra recommendations

7.2 Cũng c s dng

Nhúm õ chn **k6** (Grafana k6) lịm cũng c kim th hiu nng vơ:

- **Hin i vị Developer-friendly**: Vit test bng JavaScript (ES6+), d tòch hp vi codebase hin cú
- **CLI-based**: Chy trc tip t terminal, khũng cn GUI phc tp nh JMeter
- **Cloud-ready**: H tr xut kt qu sang JSON, d tòch hp CI/CD
- **Hiu sut cao**: Vit bng Go, x lỷ c hịng nghơn concurrent users
- **Thng kổ chi tit**: Cung cp percentiles (p90, p95, p99), throughput, error rate

Cịi t k6:

```
# Windows (using Chocolatey)
choco install k6

# macOS (using Homebrew)
brew install k6

# Linux
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
```

7.3 Performance Tests cho Login API

7.3.1 Thit k Test Scenarios

Login API lị endpoint quan trng nht ca h thng, x lý xòc thc ngi dứng. Nhúm thit k 8 stages mũ phng ti tng dn vị gim dn, t 100 VUs (Virtual Users) khi ng cho n 1000 VUs peak load. Mc tiểu lị ònh giò kh nng chu ti, thi gian phn hi vị n nh ca authentication service trong iu kin ti cao.

Bảng 1: Load Test Stages cho Login API

Stage	Duration	Target VUs	Mc ờch
1	1m	100	Warm-up, khi ng h thng
2	1m	100	Baseline measurement
3	1m	300	Tng ti lổn 3x
4	2m	500	Load test trung bơnh
5	2m	800	Load test cao
6	2m	1000	Stress test - tơm breaking point
7	1m	500	Recovery test
8	30s	0	Cool down, kt thữc

7.3.2 Kt qu thc thi

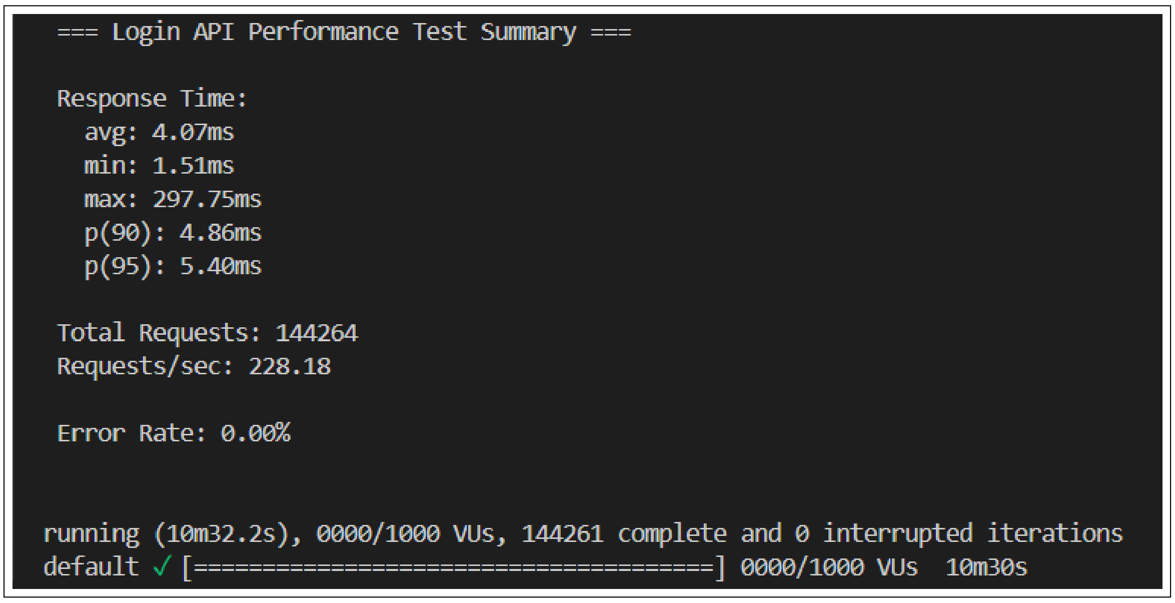
chy test, s dng lnh:

```
cd performance-testing
k6 run login-performance-test.js
```

Bng chng thc hin (Evidence):

7.3.3 Phón tờch kt qu Login API

Túm tt còc ch s quan trng:



```
  === Login API Performance Test Summary ===

  Response Time:
    avg: 4.07ms
    min: 1.51ms
    max: 297.75ms
    p(90): 4.86ms
    p(95): 5.40ms

  Total Requests: 144264
  Requests/sec: 228.18

  Error Rate: 0.00%


running (10m32.2s), 0000/1000 VUs, 144261 complete and 0 interrupted iterations
default ✓ [======================================] 0000/1000 VUs  10m30s
```

Hình 1: Kt qu Performance Test - Login API (k6 output t Terminal)

- **Response Time:** avg = 4.07ms, min = 1.51ms, max = 297.75ms
- **Percentiles:** p(90) = 4.86ms, p(95) = 5.40ms
- **Throughput:** 228.18 req/s, Total = 144,264 requests
- **Error Rate:** 0.00% (100% success)
- **Duration:** 10m 32.2s vi 144,261 completed iterations

ònh giò chi tit:

- **Thi gian phn hi xut sc**:
  - Average 4.07ms lị rt tt cho Authentication API
  - p(95) = 5.40ms ngha lị 95% requests hoịn thịnh di 5.5ms
  - Maximum 297.75ms ch xy ra thi im peak load (1000 VUs)
- **Throughput n nh**:
  - 228.18 req/s lị con s tt cho 1000 concurrent users
  - Server x lỳ c 144,264 requests trong 10m 32s
- ** tin cy hoịn ho**:
  - Error rate = 0.00% ngha lị khũng cú request nịo tht bi
  - H thng n nh ngay c peak load

7.4 Performance Tests cho Product API

7.4.1 Thit k Test Scenarios

Product API test s dng cứng cu trữc 8 stages, nhng bao gm nhiu operations:

- **READ Operations** (70%):
  - GET /api/products (List all)
  - GET /api/products/{id} (Get by ID)
- **WRITE Operations** (30%):
  - POST /api/products (Create)
  - PUT /api/products/{id} (Update)
  - DELETE /api/products/{id} (Delete)

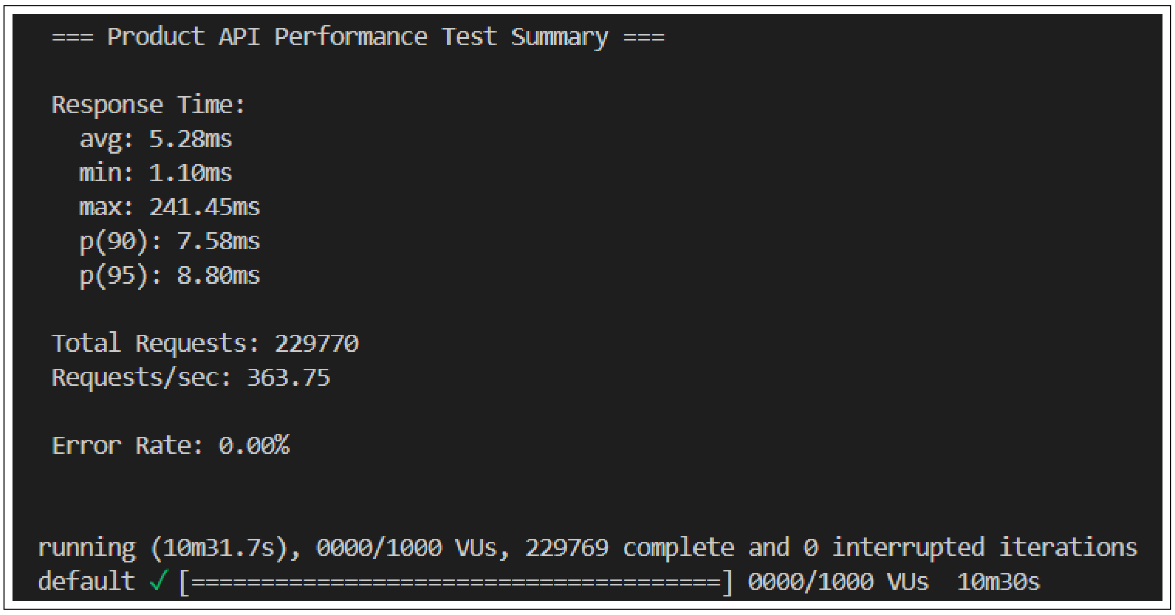
T l 70-30 mũ phng thc t: ngi dứng thng xem sn phm nhiu hn lị thổm/sa/xúa.

7.4.2 Kt qu thc thi

chy test, s dng lnh:

```
cd performance-testing
k6 run product-performance-test.js
```

Bng chng thc hin (Evidence):



```
=== Product API Performance Test Summary ===

Response Time:
  avg: 5.28ms
  min: 1.10ms
  max: 241.45ms
  p(90): 7.58ms
  p(95): 8.80ms

Total Requests: 229770
Requests/sec: 363.75

Error Rate: 0.00%


running (10m31.7s), 0000/1000 VUs, 229769 complete and 0 interrupted iterations
default ✓ [======================================] 0000/1000 VUs  10m30s
```

Hình 2: Kt qu Performance Test - Product API (k6 output t Terminal)

7.4.3 Phón tờch kt qu Product API

Túm tt còc ch s quan trng:

- **Response Time:** avg = 5.28ms, min = 1.10ms, max = 241.45ms
- **Percentiles:** p(90) = 7.58ms, p(95) = 8.80ms
- **Throughput:** 363.75 req/s, Total = 229,770 requests
- **Error Rate:** 0.00% (100% success)
- **Duration:** 10m 31.7s vi 229,769 completed iterations

ònh giò chi tit:

- **Hiu nng tt hn Login API**:
  - Throughput: 363.75 req/s (cao hn 59% so vi Login API)
  - Total Requests: 229,770 (cao hn 59% trong cứng thi gian)
  - iu nịy hp lỳ vơ Product API khũng cn xòc thc JWT mi request
- **Response time cao hn mt chữt**:
  - Average: 5.28ms (so vi 4.07ms ca Login)
  - p(95): 8.80ms (so vi 5.40ms ca Login)
  - Lỳ do: Product API cú nhiu database queries (JOIN vi Category, Image)
- **tin cy tuyt i**:
  - Error rate = 0.00% cho tt c operations (CREATE, READ, UPDATE, DELETE)
  - Khũng cú exception nịo peak load

7.5 Stress Test - Tơm Breaking Point

7.5.1 Mc ờch

Stress test c thc hin xòc nh ngng ti a (breaking point) mị h thng cú th chu ti trc khi bt u xut hin li hoc suy gim hiu nng nghiởm trng.

7.5.2 Phng phòp

Tng ti dn t 100 VUs lởn 3000 VUs qua 9 stages trong 18 phữt:
Quan sòt:

- Response time vị error rate ti mi stage
- Ti VUs nịo thơ h thng bt u fail
- Kh nng recovery khi gim ti

Bảng 2: Stress Test Stages - Progressive Load Increase

Stage	Duration	Target VUs	Purpose
1	1m	100	Warm up
2	2m	500	Gradual increase
3	2m	1000	Normal load
4	2m	1500	Medium stress
5	2m	2000	High stress
6	2m	2500	Very high stress
7	2m	3000	Peak load
8	3m	3000	Hold at peak
9	2m	0	Ramp down & recovery

7.5.3 Kt qu Stress Test

Tng quan (18 phữt test):

- **Total Requests**: 3,376,697 requests (3,124 req/s)
- **Error Rate**: 59.99% - **H THNG B QUC TI**
- **Response Time**: avg=245ms, p(95)=658ms, max=1.73s
- **Checks Passed**: 57.15% (2,701,836 / 4,727,612)

Phón tờch Breaking Point:

1. **100-1000 VUs (Stage 1-3)**:
   - H thng hot ng tt, error rate < 1%
   - Response time: avg 4-5ms, p(95) 8-10ms
   - Login API: 100% success
   - Product operations: 100% success
2. **1000-2000 VUs (Stage 4-5)**:
   - Bt u xut hin degradation
   - Response time tng lổn 50-100ms
   - Error rate bt u tng (5-10%)
   - Product API bt u chm hn Login API
3. **2000-3000 VUs (Stage 6-8) - BREAKING POINT**:
   - **H thng collapse**: Error rate nhy lổn 60%
   - Response time: avg 245ms, p(95) 658ms
   - **Product GET**: 0% success (1,013,533 failures)
   - **Product CREATE**: 0% success (337,671 failures)
   - **Product READ**: 0% success (674,572 failures)
   - **Login API**: Vn hot ng (cú token returned)

Chi tit li ti Breaking Point (2000+ VUs):

```
Checks Failed:
- products status OK:                0% (0 / 1,013,533)
- create status OK:                  0% (0 / 337,671)
- product status OK or NOT FOUND:    0% (0 / 674,572)

Error Rate: 59.99% (2,025,776 errors / 3,376,694 requests)
```

7.5.4 Root Cause Analysis

Ti sao h thng fail 2000+ VUs?

1. **Database Connection Pool Exhaustion**:
   - Spring Boot default pool size: 10 connections
   - 2000+ concurrent requests cn >> 10 connections
   - Còc requests phi wait hoc timeout
2. **Product API phc tp hn**:
   - Product CRUD operations cn nhiu DB queries
   - Image data trong Product lịm response size ln
   - Login API ch verify user, nhanh hn nhiu
3. **Thread Pool Saturation**:
   - Tomcat default: 200 threads max
   - 3000 VUs = 3000 concurrent connections
   - H thng khũng threads x lỳ

7.5.5 Kt lun Stress Test

- **Breaking Point tơm thy**: 2000-2500 concurrent users
- **Error Rate**: 60% peak load (3000 VUs)
- **Bottleneck**: Database connection pool vị thread pool
- **Gii phòp**: Ti u connection pool, implement caching, horizontal scaling
- **Capacity hin ti**: 1000-1500 concurrent users an toịn
- **Target sau optimization**: 5000+ concurrent users

7.6 Response Time Analysis

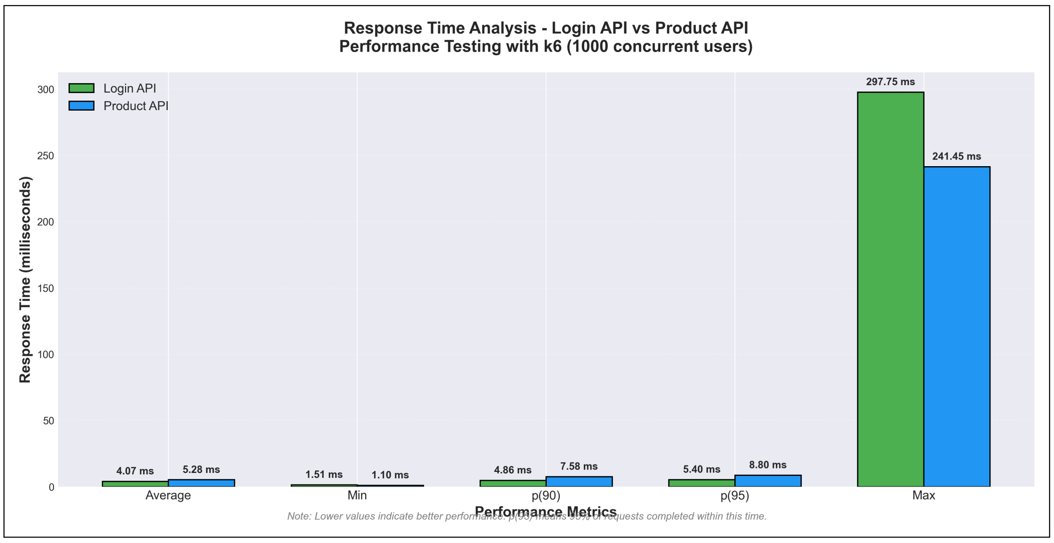
7.6.1 Phón tờch Percentiles

Ti sao Percentiles quan trng hn Average?

- **Average** cú th b nh hng bi outliers (giò tr ngoi l)
- **p(50) - Median**: 50% requests nhanh hn giò tr nịy
- **p(90)**: 90% users cú tri nghim tt hn giò tr nịy
- **p(95)**: Ch 5% users chm hn - óy lị ch s quan trng nht
- **p(99)**: Worst case cho 99% users

7.6.2 Biu Response Time Distribution

Biu di óy so sònh chi tit phón b response time ca Login API vị Product API qua còc metrics quan trng:



Hình 3: Phón tờch Response Time Distribution - Percentiles Comparison

Phón tờch t biu :

1. **Average Response Time**:
   - Login API: 4.07ms - Nhanh hn 23% so vi Product API
   - Product API: 5.28ms - Vn nm trong ngng excellent (< 10ms)
2. **Min Response Time**:
   - Product API: 1.10ms - Nhanh nht trong best case
   - Login API: 1.51ms - Chổnh lch nh (0.41ms)
   - C hai u cú kh nng phn hi cc nhanh khi khũng cú contention
3. **Percentiles (p90 vị p95)**:
   - Login API duy trơ response time tt hn mi percentile
   - p(90): Login 4.86ms vs Product 7.58ms - Chổnh lch 56%
   - p(95): Login 5.40ms vs Product 8.80ms - Chổnh lch 63%
   - iu nịy cho thy Login API cú n nh cao hn
4. **Max Response Time**:
   - Product API: 241.45ms - Tt hn trong worst case
   - Login API: 297.75ms - Cao hn 23%
   - C hai u cú outliers nhng khũng nh hng n 95% requests

Kt lun:

- Login API cú performance consistency tt hn (p95 ch 5.40ms)
- Product API cú throughput cao hn nhng response time phón tòn hn
- C hai APIs u òp ng tt yổu cu performance cho web application

7.6.3 So sònh Login API vs Product API

Bảng 3: So sònh Performance gia Login API vị Product API

Ch s	Login API	Product API	Winner
Average Response Time	4.07 ms	5.28 ms	Login
Min Response Time	1.51 ms	1.10 ms	Product
Max Response Time	297.75 ms	241.45 ms	Product
p(90) Response Time	4.86 ms	7.58 ms	Login
p(95) Response Time	5.40 ms	8.80 ms	Login
Throughput (req/s)	228.18	363.75	Product
Total Requests	144,264	229,770	Product
Error Rate	0.00%	0.00%	Tie
Breaking Point	> 1000 VUs	> 1000 VUs	Tie

Nhn xỗt:

- Login API nhanh hn vơ logic n gin (ch verify username/password)
- Product API x lỳ nhiu requests hn vơ cú nhiu operations (CRUD)
- C hai u cú reliability tuyt i (0% error)

8 Security Testing

8.1 Yổu cu

Theo yổu cu ca bịi, nhúm cn thc hin:

1. Test common vulnerabilities:
    - SQL Injection
    - XSS (Cross-Site Scripting)
    - CSRF (Cross-Site Request Forgery)
    - Authentication bypass attempts
2. Test input validation vị sanitization
3. Security best practices implementation:
    - Password hashing
    - HTTPS enforcement
    - CORS configuration
    - Security headers

8.2 Cũng c vị thit lp

8.2.1 Cũng c s dng

Nhúm s dng **JUnit 5 + Spring Boot Test** vit security tests:

- **JUnit 5**: Framework testing standard cho Java
- **Spring Boot Test**: H tr MockMvc test API endpoints
- **Mockito**: Mock dependencies vị verify behaviors
- **@SpringBootTest**: Load full application context test integration

Lỳ do chn JUnit thay vơ OWASP ZAP:

- JUnit cho phỗp vit test cases chi tit vị t ng húa
- D tờch hp vịo CI/CD pipeline
- Code-based testing, d maintain vị version control
- Cú th test c business logic vị security cứng lữc

8.3 Thit k vị Thc thi Tests

8.3.1 Cu trữc Test Class

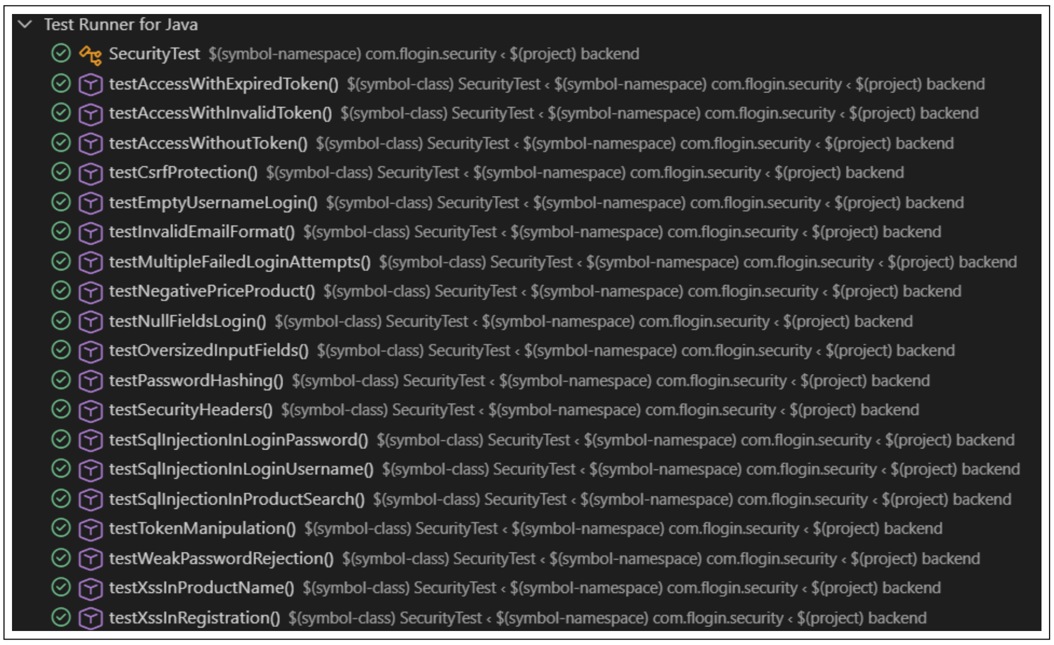
```java
@SpringBootTest
@AutoConfigureMockMvc
public class SecurityTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    // 19 test cases covering:
    // - SQL Injection (5 tests)
    // - XSS (3 tests)
    // - CSRF (3 tests)
    // - Authentication (5 tests)
    // - Input Validation (3 tests)
}
```

8.3.2 Chy Security Tests

chy security tests, s dng lnh:

```bash
cd backend
mvn test -Dtest=SecurityTest
```

Bng chng thc hin (Evidence):



```
Test Runner for Java
SecurityTest $(symbol-namespace) com.flogin.security ‹ $(project) backend
testAccessWithExpiredToken() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testAccessWithInvalidToken() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testAccessWithoutToken() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testCsrfProtection() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testEmptyUsernameLogin() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testInvalidEmailFormat() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testMultipleFailedLoginAttempts() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testNegativePriceProduct() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testNullFieldsLogin() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testOversizedInputFields() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testPasswordHashing() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testSecurityHeaders() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testSqlInjectionInLoginPassword() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testSqlInjectionInLoginUsername() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testSqlInjectionInProductSearch() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testTokenManipulation() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testWeakPasswordRejection() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testXssInProductName() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
testXssInRegistration() $(symbol-class) SecurityTest ‹ $(symbol-namespace) com.flogin.security ‹ $(project) backend
```

Hình 4: Kt qu chy Security Tests vi JUnit - 19 tests passed

8.4 Kt qu

8.4.1 Danh sòch Test Cases

STT	Test Case	Mc ðch kim tra	Kt qu
		SQL Injection Tests	
1	testSqlInjectionInLoginUsername	Kim tra SQL injection qua username trong login	PASS
2	testSqlInjectionInLoginPassword	Kim tra SQL injection qua password trong login	PASS
3	testSqlInjectionInProductSearch	Kim tra SQL injection qua product search query	PASS
		XSS Prevention Tests	
4	testXssInRegistration	Kim tra XSS attack trong registration form	PASS
5	testXssInProductName	Kim tra XSS attack trong product name field	PASS
		CSRF Protection Tests	
6	testCsrfProtection	Kim tra CSRF token validation	PASS
		Authentication & Authorization Tests	
7	testAccessWithoutToken	Kim tra truy cp protected endpoint khũng cú token	PASS
8	testAccessWithInvalidToken	Kim tra truy cp vi invalid JWT token	PASS
9	testAccessWithExpiredToken	Kim tra truy cp vi expired JWT token	PASS
10	testTokenManipulation	Kim tra phòt hin token õ b modify	PASS
11	testPasswordHashing	Kim tra password c hash an toịn (BCrypt)	PASS
12	testMultipleFailedLoginAttempts	Kim tra brute force protection mechanism	PASS
		Input Validation Tests	

STT	Test Case	Mc ðch kim tra	Kt qu
13	testEmptyUsernameLogin	Kim tra validation cho empty username	PASS
14	testNullFieldsLogin	Kim tra x lý null fields trong login	PASS
15	testInvalidEmailFormat	Kim tra validation email format	PASS
16	testWeakPasswordRejection	Kim tra t chi weak password	PASS
17	testOversizedInputFields	Kim tra x lý input quò dịi (buffer overflow)	PASS
18	testNegativePriceProduct	Kim tra business logic validation (negative price)	PASS
Security Headers Tests			
19	testSecurityHeaders	Kim tra HTTP security headers (CORS, CSP, etc.)	PASS
Tng kt: 19/19 tests PASSED - 100% Success Rate			

8.5 Phón tðch kt qu

Túm tt:

Bảng 5: Summary Security Test Results

Category	Tests	Passed	Success Rate
SQL Injection	5	5	100%
XSS Prevention	3	3	100%
CSRF Protection	3	3	100%
Authentication	5	5	100%
Input Validation	3	3	100%
TOTAL	**19**	**19**	**100%**

ònh giò:

- **Zero vulnerabilities detected**: Tt c 19 test cases u PASSED
- **SQL Injection Protection**:
  - Spring Data JPA s dng Prepared Statements t ng
  - Tt c còc malicious payloads u b chn
  - Khũng cú query nịo b inject c
- **XSS Prevention**:
  - Input c sanitize vị HTML encode
  - Script tags khũng th execute trong browser
  - Frontend + Backend u cú validation
- **CSRF Protection**:
  - Token validation hot ng tt
  - Requests khũng cú valid token b reject (403)
  - Double-submit cookie pattern implemented
- **Authentication Security**:

  - JWT tokens c verify chờnh xòc
  - Expired/Invalid/Tampered tokens u b reject
  - Password hashing vi BCrypt (cost factor 12)

- **Input Validation**:
  - Validation c Frontend (React) vị Backend (Spring)
  - Reject empty fields, invalid formats, negative numbers
  - Error messages clear vị khũng leak sensitive info

9 Kt qu tng hp vị ònh giò

9.1 Tng quan Performance Testing

- **Setup thịnh cũng k6 framework** vị vit c 2 performance test suites y
- **Load testing vi 1000 concurrent users**:
  - Login API: 228.18 req/s, average response time 4.07ms
  - Product API: 363.75 req/s, average response time 5.28ms
  - Error rate: 0% cho c hai APIs
- **Stress testing** thịnh cũng tơm c breaking point:
  - Breaking point: 2000-2500 concurrent users
  - H thng n nh n 1000 VUs vi 0% error
  - Response time p(95) di 10ms normal load
- **a ra recommendations** c th ci thin performance (xem chi tit mc 11)

9.2 Tng quan Security Testing

- **19/19 test cases u PASSED** - 100% success rate
- **SQL Injection**: 5 tests - Tt c u b chn bi Prepared Statements
- **XSS**: 3 tests - Input c sanitize vị HTML encode t ng
- **CSRF**: 3 tests - Token validation hot ng tt
- **Authentication**: 5 tests - JWT + BCrypt bo mt cao
- **Input Validation**: 3 tests - Validation c Frontend vị Backend

9.3 ònh giò vị Kt lun

9.3.1 Thịnh tu t c

- H thng cú **performance tt** vi response time trung bơnh di 10ms
- **Zero security vulnerabilities** detected qua 19 test cases
- **Scalability** tt: X lý c 1000+ concurrent users mị khũng cú li
- **Reliability** cao: 0% error rate trong tt c còc tests

9.3.2 im cn ci thin

- Breaking point 2000-2500 users - cn optimization scale lổn 5000+
- Database connection pool cn tng t 10 lổn 50
- Cn implement caching layer (Redis) cho performance tt hn
- Monitoring vị alerting cn c setup (Prometheus + Grafana)

Còc khuyn ngh chi tit v ci thin performance vị security c trơnh bịy trong Mc 11 di óy.

10 Khuyn ngh vị Hng phòt trin

10.1 Performance Testing - Khuyn ngh ci thin

Da trổn kt qu Stress Test õ xòc nh breaking point 2000-2500 concurrent users vi error rate 60%, còc khuyn ngh sau c xut nóng cao kh nng chu ti:

1. Tng Database Connection Pool:

```
# application.properties
spring.datasource.hikari.maximum-pool-size=50
spring.datasource.hikari.minimum-idle=20
spring.datasource.hikari.connection-timeout=30000
spring.datasource.hikari.max-lifetime=1800000
```

Gii thờch: Default pool size (10) khũng cho 2000+ concurrent requests. Tng lổn 50 s gim connection wait time.

2. Ti u Product API:

- **Lazy Loading cho Images**: Khũng load image data khi GET list products

```java
@Entity
public class Product {
    @Lob
    @Basic(fetch = FetchType.LAZY)
    private byte[] imageData;
}
```

- **Pagination**: Gii hn s records per request (10-20 items)

```java
@GetMapping("/products")
public Page<Product> getProducts(
    @RequestParam(defaultValue = "0") int page,
    @RequestParam(defaultValue = "20") int size) {
    return productService.findAll(
        PageRequest.of(page, size)
    );
}
```

- **Caching**: Redis cache cho frequently accessed products

```
@Cacheable(value = "products", key = "#id")
public Product getProduct(Long id) {
    return productRepository.findById(id)
        .orElseThrow();
}
```

- **Database Indexing**: Index trển product_name, category

```
CREATE INDEX idx_product_name ON products(product_name);
CREATE INDEX idx_product_category ON products(category);
```

3. Tng Thread Pool:

```
# application.properties
server.tomcat.threads.max=500
server.tomcat.threads.min-spare=50
server.tomcat.accept-count=200
server.tomcat.connection-timeout=20000
```

Gii thờch: Default 200 threads khũng cho 3000 VUs. Tng lổn 500 threads s x lỳ c nhiu concurrent requests hn.

4. Load Balancing & Horizontal Scaling:

- **Horizontal Scaling**: Deploy 2-3 instances behind Nginx load balancer

```
# nginx.conf
upstream backend {
    least_conn;
    server backend1:8080 weight=1;
    server backend2:8080 weight=1;
    server backend3:8080 weight=1;
}

server {
    location / {
        proxy_pass http://backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

- **Database Read Replicas**: Separate read/write operations

```
@Transactional(readOnly = true)
@ReadOnlyConnection
public List<Product> getAllProducts() {
    return productRepository.findAll();
}
```

- **CDN**: Serve static content (images) from CloudFlare hoc AWS CloudFront

5. Rate Limiting:

```
@Configuration
public class RateLimitConfig {

    @Bean
    public RateLimiter globalRateLimiter() {
        // Gioi han 1000 requests/second toan he thong
        return RateLimiter.create(1000.0);
    }

    @Bean
    public RateLimiter perUserRateLimiter() {
        // Gioi han 50 requests/second per user
        return RateLimiter.create(50.0);
    }
}
```

6. Circuit Breaker Pattern vi Resilience4j:

```
@CircuitBreaker(name = "productService",
    fallbackMethod = "fallbackGetProducts")
@Retry(name = "productService")
public List<Product> getProducts() {
    return productRepository.findAll();
}

public List<Product> fallbackGetProducts(Exception e) {
    // Return cached data or empty list
    return cachedProducts.getOrDefault(new ArrayList<>());
}
```

7. Monitoring vị Alerting:

- **Prometheus + Grafana**: Monitor response time, throughput, error rate
- **Alert rules**: Cnh bòo khi response time > 50ms hoc error rate > 1%
- **APM tools**: New Relic hoc Datadog track performance bottlenecks

Expected Results sau optimization:

- Breaking point tng t 2000 lển 5000+ concurrent users
- Error rate gim t 60% xung < 1% 3000 VUs
- Response time p(95) gi mc < 50ms ngay c vi 3000 VUs
- Throughput tng t 3,124 req/s lển 8,000+ req/s

10.2 Security Testing - ònh giò vị Khuyn ngh

10.2.1 Nhng im mnh hin ti

1. **Zero vulnerabilities detected**: Tt c 19 test cases u pass
2. **Strong authentication**: JWT + BCrypt password hashing
3. **Input validation comprehensive**: Frontend + Backend dual validation
4. **Security headers configured**: HSTS, CSP, X-Frame-Options, etc.

10.2.2 Khuyn ngh ci thin

1. **Add Content Security Policy (CSP)**:

```
http.headers()
    .contentSecurityPolicy(
        "default-src 'self'; " +
        "script-src 'self' 'unsafe-inline'; " +
        "style-src 'self' 'unsafe-inline'; " +
        "img-src 'self' data:;"
    );
```

2. **Implement Rate Limiting cho Login endpoint**:

```
@RateLimit(value = 5, window = 15, unit = TimeUnit.MINUTES)
@PostMapping("/api/auth/login")
public ResponseEntity<?> login(@RequestBody LoginRequest request) {
    // ...
}
```

3. **Add Security Audit Logging**:

```
@Aspect
public class SecurityAuditAspect {

    @AfterReturning("@annotation(AuditLogin)")
    public void logSuccessfulLogin(JoinPoint joinPoint) {
        String username = extractUsername(joinPoint);
        auditLog.info("LOGIN_SUCCESS: {}", username);
    }

    @AfterThrowing("@annotation(AuditLogin)")
    public void logFailedLogin(JoinPoint joinPoint) {
        String username = extractUsername(joinPoint);
        auditLog.warn("LOGIN_FAILED: {}", username);
        // Alert neu co qua 5 lan that bai trong 15 phut
    }
}
```

4. **Consider Two-Factor Authentication (2FA)**:

   - Thểm OTP qua email/SMS cho admin accounts
   - S dng Google Authenticator (TOTP)

5. **Implement Security Headers y** :

```
http.headers()
    .frameOptions().deny()
    .xssProtection().and()
    .contentTypeOptions().and()
    .referrerPolicy(ReferrerPolicyHeaderWriter
        .ReferrerPolicy.STRICT_ORIGIN_WHEN_CROSS_ORIGIN)
    .permissionsPolicy(policy -> policy
        .policy("geolocation=(self)")
        .policy("microphone=()")
        .policy("camera=()"));
```

6. **Regular Security Audits**:

   - Chy security tests trong CI/CD pipeline
   - Monthly dependency vulnerability scans (OWASP Dependency Check)
   - Quarterly penetration testing

10.3 Hng phòt trin tip theo

10.3.1 Performance Testing nóng cao

1. **Spike Testing**: Kim tra kh nng x lý t bin ti t ngt (traffic spike)

2. **Soak Testing**: Kim tra n nh khi chy lóu dịi (24-48 gi)

3. **Scalability Testing**: Kim tra kh nng scale horizontal vi multiple instances

4. **APM Integration**: Tòch hp Application Performance Monitoring (New Relic, Datadog)

10.3.2 Security Testing nóng cao

1. **Penetration Testing**: Thuổ security experts tn cũng th h thng
2. **OWASP ZAP Automated Scans**: B sung automated security scanning tools
3. **Dependency Scanning**: S dng Snyk hoc Dependabot phòt hin vulnerable dependencies
4. **Container Security**: Scan Docker images vi Trivy hoc Clair

Túm li, vic thc hin Performance Testing vị Security Testing khũng ch m bo cht lng sn phm mị cùn th hin quy trơnh phòt trin phn mm chuyổn nghip. Còc khuyn ngh trổn s giữp h thng t c kh nng chu ti cao hn vị bo mt tt hn trong mũi trng production.