# Code smells & Maintainability

1. Unit test coverage
   1. 70-90% and higher.
2. Rate limiting
   1. Rate limiting middleware or other mechanisms.
   2. Prevent Denial of Service (DoS) attacks
3. Identify security vulnerabilities:
   1. Linter plugins, such as eslint-plugin-security,
   2. Scan potential issues in your Node.js code.
4. Code should generate with proper Indentation and spacing.
5. Naming Conventions should be fixed according to Java Standard:
   1. Class & interface name
   2. Package name
   3. Field Name & Variable name
   4. Method name
6. Remove commented lines.
7. Remove duplicate lines.
8. Ensure, we are hiding error details to client.
9. Loggin & Monitoring
   1. As of now there is no proper loggings.
10. Testing & Code Quality
    1. Setup proper DevOps pipeline when needed.
    2. Assess the performance of microservices under load (using tools like JMeter or Gatling).

# RESTful Services Practices

1. Choose meaningful and descriptive names for your resources.
2. Use nouns to represent resources instead of verbs.
3. Use lowercase letters and hyphens for multi-word resource names.

# Security

1. Authentication & Authorization:

Implement secure authentication mechanisms, such as OAuth or JWT, to protect your RESTful API.

1. SQL injection
   1. Avoid the risk of SQL injection attacks (use proper query parameterization and avoiding string concatenation)
2. Inspect for Outdated Packages and upgrade packages.
   1. Vulnerability Scanning (infrastructure, services, and dependencies)
   2. Nessus,
   3. OpenVAS
   4. Brup Suite
   5. Snyk
3. Rate Limiting and preventing DDOS attack
   1. Rate-limiter-flexible,
   2. Nginx,
   3. Application configuration
4. Prevent injection attacks by validating inputs:
   1. SQL
   2. NoSQL
   3. XSS
5. API Security
   1. Use HTTPS
   2. Secure endpoints and APIs using tools like OWASP ZAP or Burp Suite
   3. API rate limiting.
6. White-box (with access to code and architecture)
   1. Static Application Security Testing (SAST):
      1. Checkmarx
      2. Fortify
      3. Veracode
   2. Dependency Scanning
      1. OWASP Dependency-Check
      2. Snyk
   3. Code Quality and Security Analysis
      1. CodeClimate
      2. SonarQube
7. Black-box (without access) testing
   1. OWASP ZAP (Zed Attack Proxy) for Spring Boot backend APIs and Angular frontend
   2. Fuzz testing (injects invalid, malformed, or unexpected inputs into a system to reveal software defects and vulnerabilities
      1. Codenomicon’s product suite
      2. Peach Fuzzing Platform
      3. Beyond Security’s beSTORM product
      4. ForAllSecure Mayhem for Code
      5. CI Fuzz
      6. Fuzzbuzz
8. Pen testing
   1. Metasploit
   2. SQLMap
   3. Burp Suite
9. Security Misconfiguration Testing
   1. Databases
   2. Default settings
   3. Other application security
10. Dynamic Application Security Testing (DAST):
    1. Test the running application for vulnerabilities while it's in action, simulating attacks.
    2. Tools
       1. OWASP ZAP,
       2. Burp Suite,
       3. AppScan