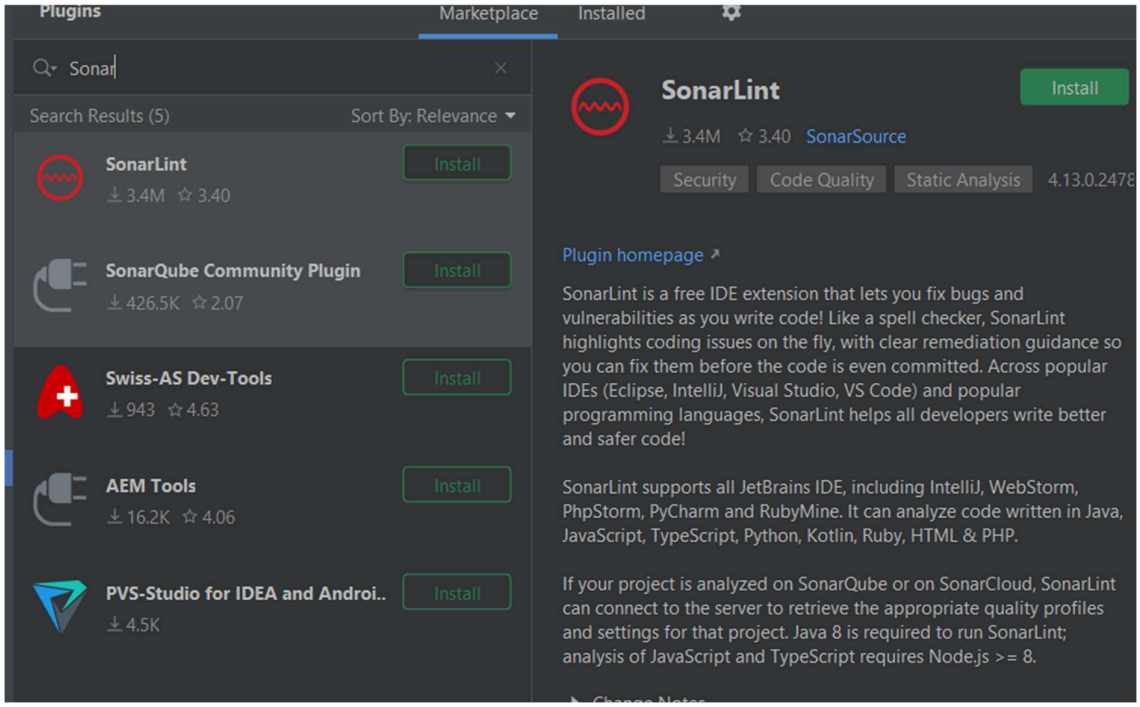
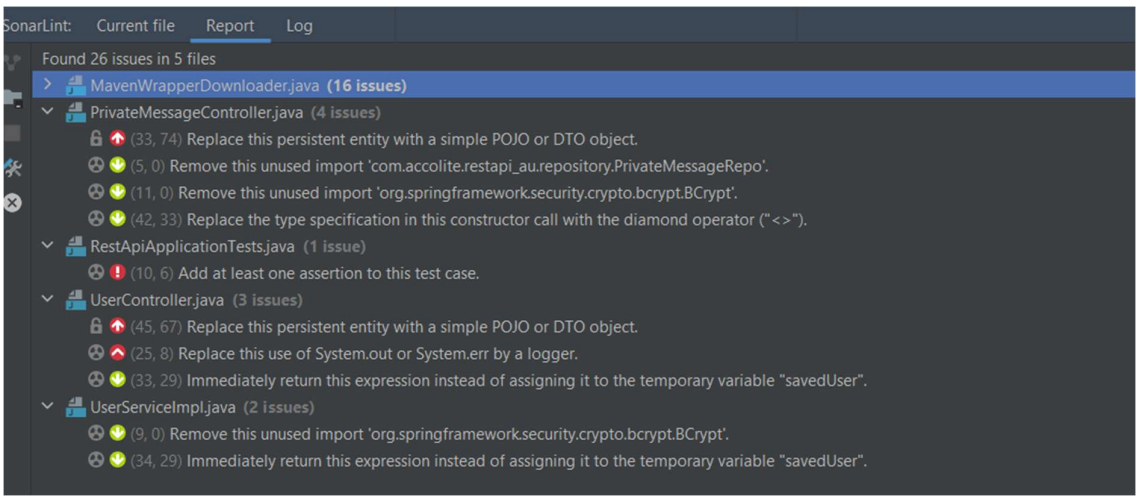
JAVA CODE QUALITY ASSIGNMENT

By Raj Vignesh Karunakaran

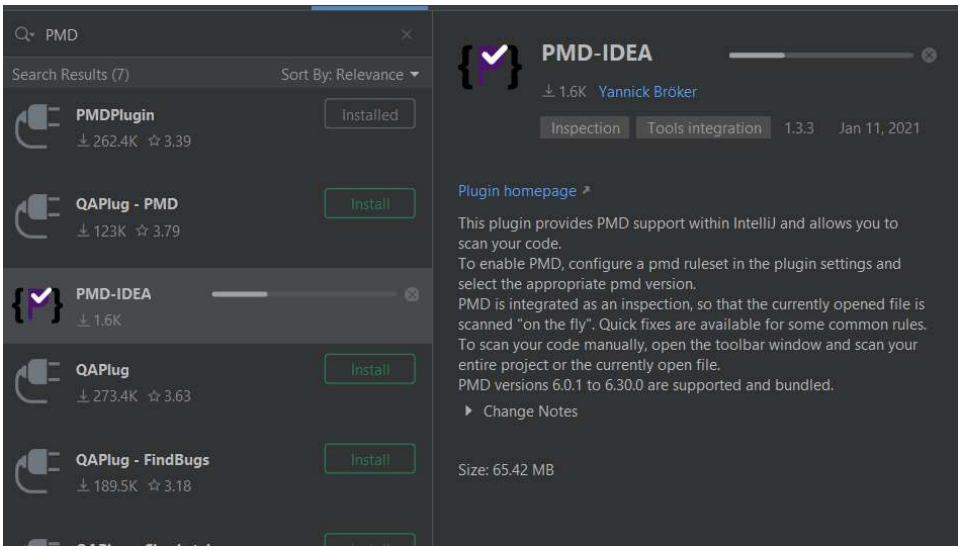
1. Installing Code Analyzers and Code Report:
   a. **SonarLint**



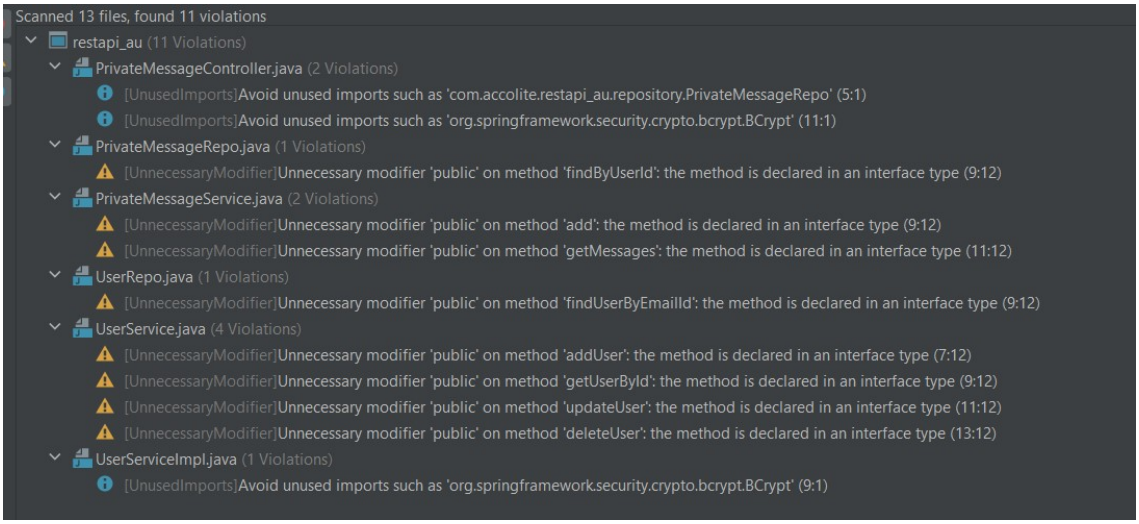
Report:



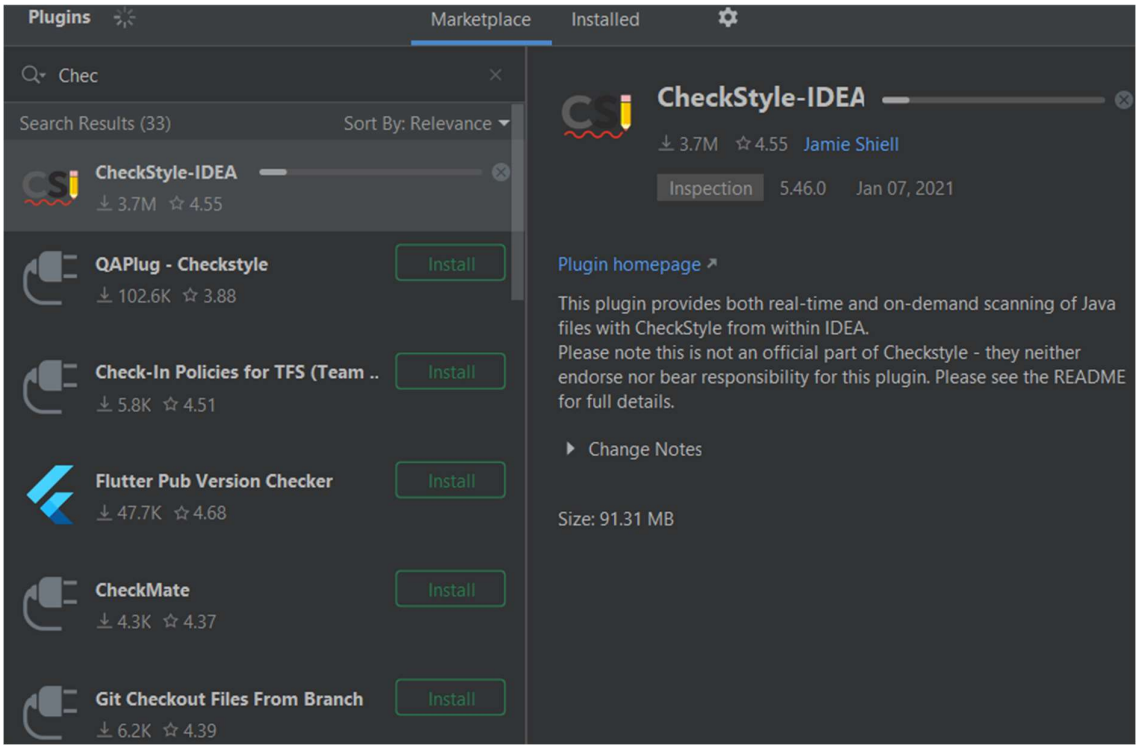
b. PMD


PMD
Search Results (7)
Sort By: Relevance
PMDPlugin
262.4K ☆3.39
Installed
QAPlug - PMD
123K ☆3.79
Install
PMD-IDEA
1.6K
QAPlug
273.4K ☆3.63
Install
QAPlug - FindBugs
189.5K ☆3.18
Install
PMD-IDEA
1.6K Yannick Bröker
Inspection
Tools integration
1.3.3
Jan 11, 2021
Plugin homepage
This plugin provides PMD support within IntelliJ and allows you to scan your code.
To enable PMD, configure a pmd ruleset in the plugin settings and select the appropriate pmd version.
PMD is integrated as an inspection, so that the currently opened file is scanned "on the fly". Quick fixes are available for some common rules.
To scan your code manually, open the toolbar window and scan your entire project or the currently open file.
PMD versions 6.0.1 to 6.30.0 are supported and bundled.
Change Notes
Size: 65.42 MB

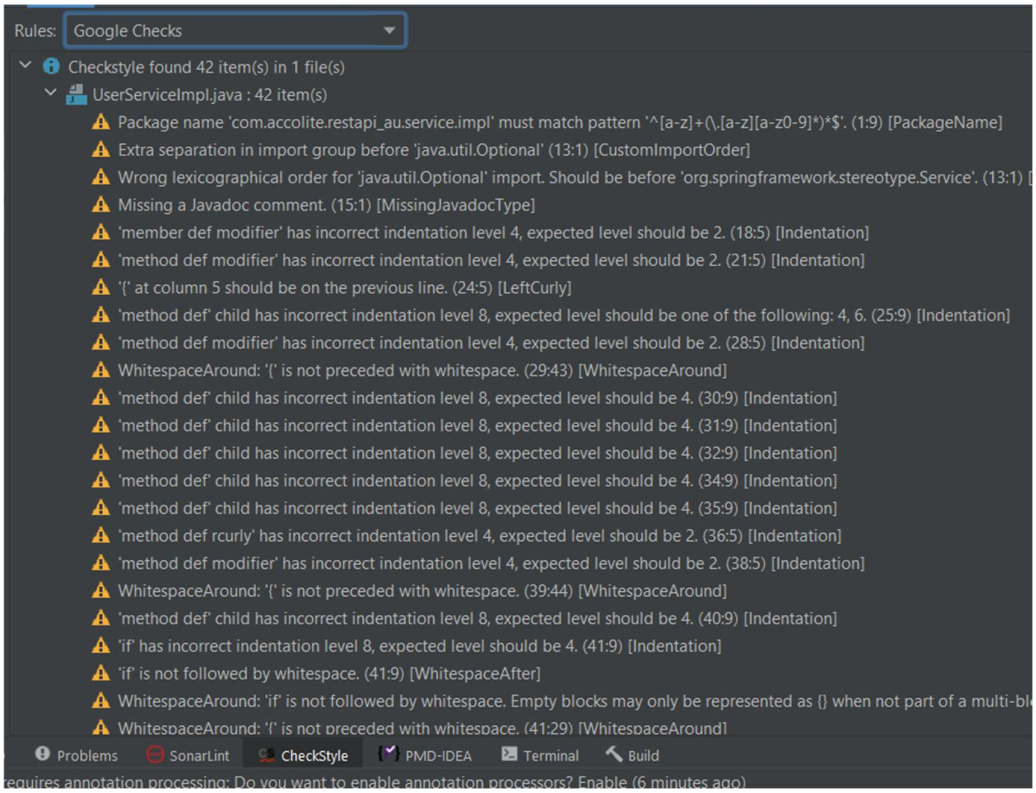
Report:


Scanned 13 files, found 11 violations
restapi_au (11 Violations)
PrivateMessageController.java (2 Violations)
[UnusedImports]Avoid unused imports such as 'com.accolite.restapi_au.repository.PrivateMessageRepo' (5:1)
[UnusedImports]Avoid unused imports such as 'org.springframework.security.crypto.bcrypt.BCrypt' (11:1)
PrivateMessageRepo.java (1 Violations)
[UnnecessaryModifier]Unnecessary modifier 'public' on method 'findByUserId': the method is declared in an interface type (9:12)
PrivateMessageService.java (2 Violations)
[UnnecessaryModifier]Unnecessary modifier 'public' on method 'add': the method is declared in an interface type (9:12)
[UnnecessaryModifier]Unnecessary modifier 'public' on method 'getMessages': the method is declared in an interface type (11:12)
UserRepo.java (1 Violations)
[UnnecessaryModifier]Unnecessary modifier 'public' on method 'findUserByEmailId': the method is declared in an interface type (9:12)
UserService.java (4 Violations)
[UnnecessaryModifier]Unnecessary modifier 'public' on method 'addUser': the method is declared in an interface type (7:12)
[UnnecessaryModifier]Unnecessary modifier 'public' on method 'getUserById': the method is declared in an interface type (9:12)
[UnnecessaryModifier]Unnecessary modifier 'public' on method 'updateUser': the method is declared in an interface type (11:12)
[UnnecessaryModifier]Unnecessary modifier 'public' on method 'deleteUser': the method is declared in an interface type (13:12)
UserServiceImpl.java (1 Violations)
[UnusedImports]Avoid unused imports such as 'org.springframework.security.crypto.bcrypt.BCrypt' (9:1)

c. CheckStyle

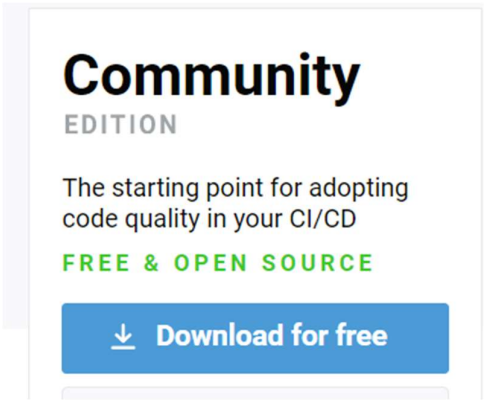


Report:

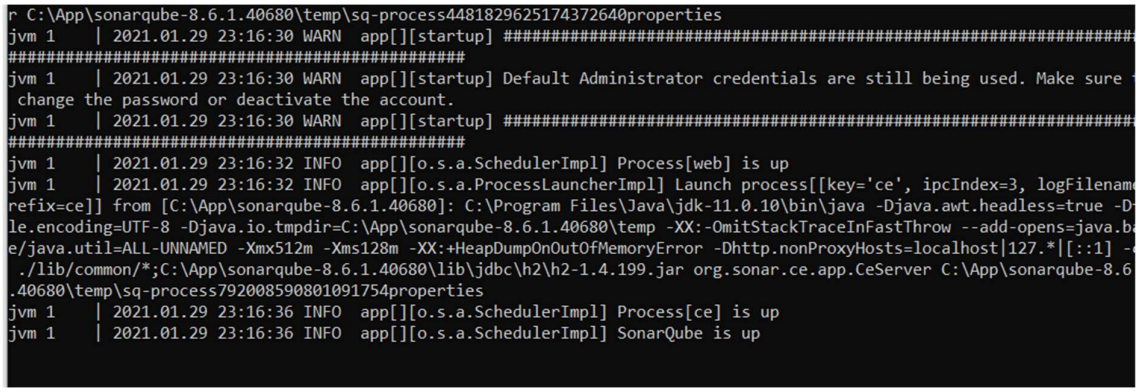


2. Installing SonarQube and Running analysis on an Existing Project

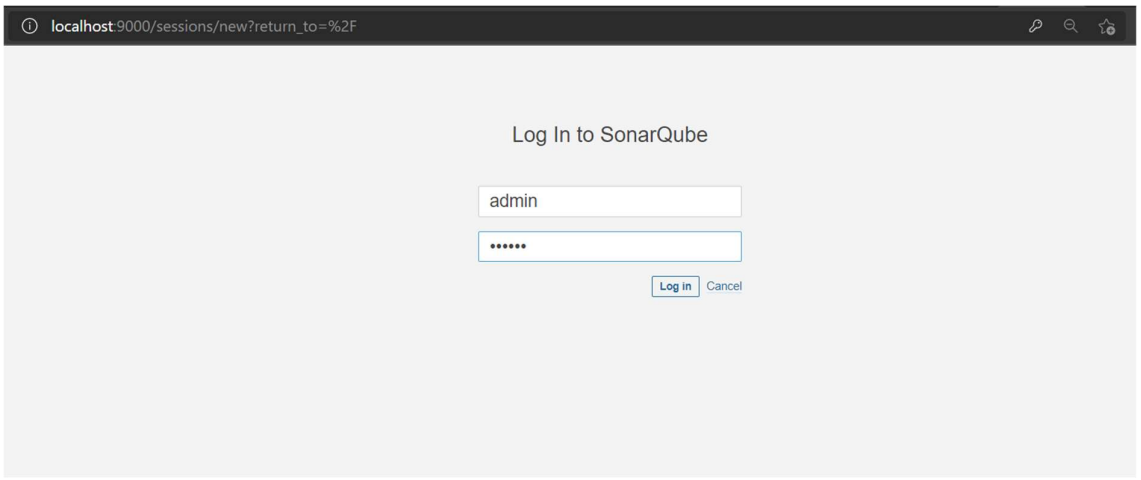
   a. Download SonarQube.zip from given link



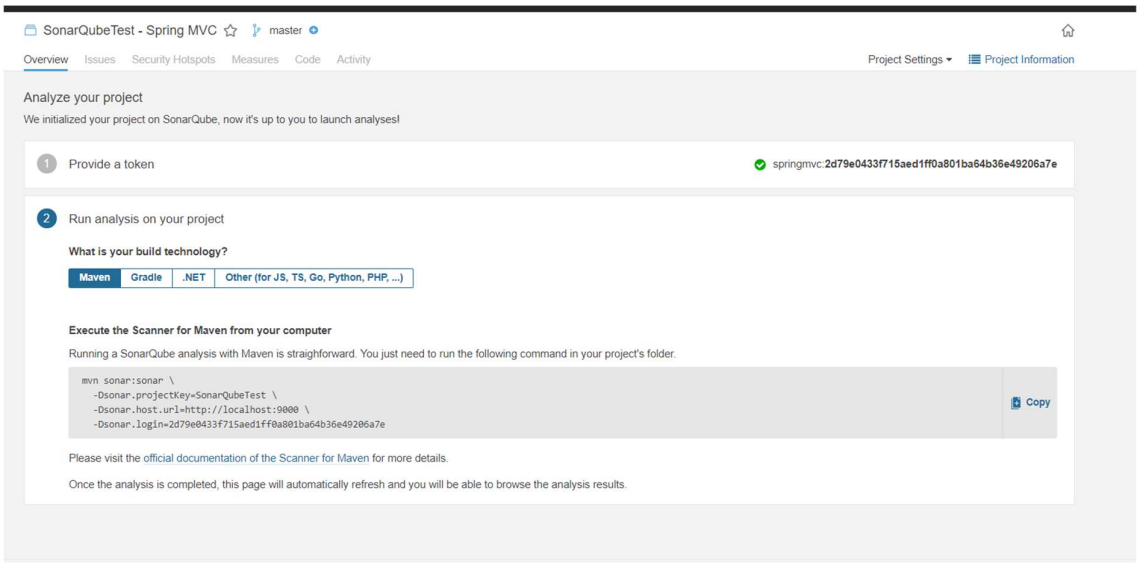
   b. Unzip the folder and Navigate to bin/windows_x86_64
   c. Run StartSonar.bat



   d. Go to localhost:9000 – Default Login username:password = admin:admin



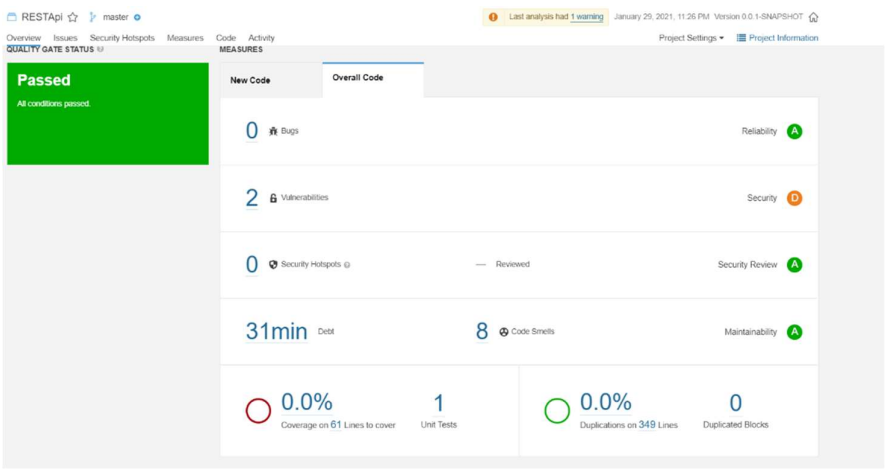
e. Create a new Project with your Required name. Run the following maven command in your project



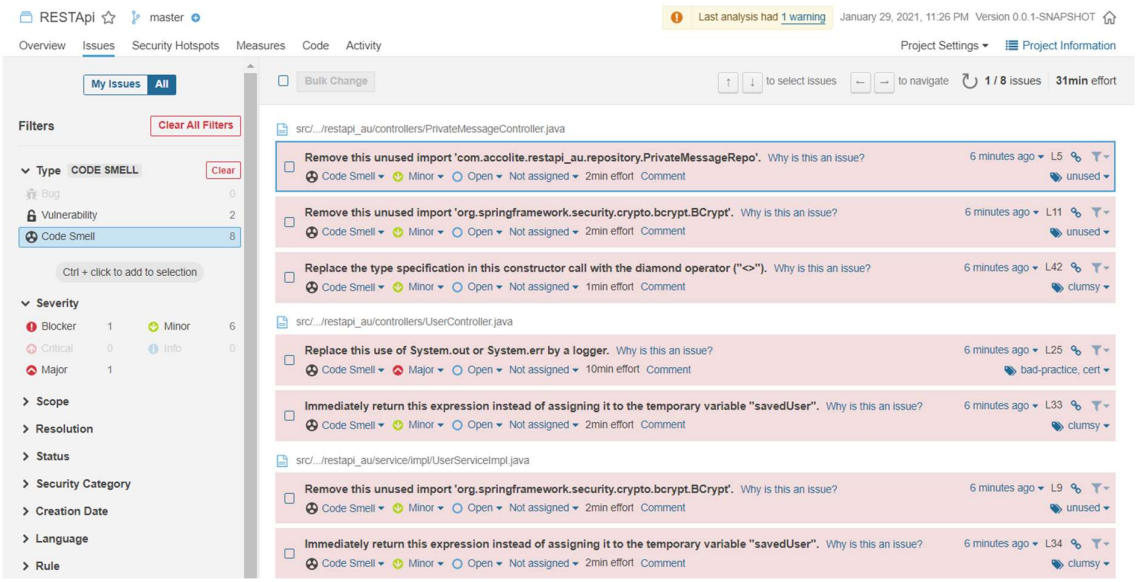

```
[INFO] Sensor Java CPD Block Indexer (done) | time=56ms
[INFO] SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO] CPD Executor 7 files had no CPD blocks
[INFO] CPD Executor Calculating CPD for 6 files
[INFO] CPD Executor CPD calculation finished (done) | time=28ms
[INFO] Analysis report generated in 144ms, dir size=117 KB
[INFO] Analysis report compressed in 902ms, zip size=36 KB
[INFO] Analysis report uploaded in 52ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=SonarQubeTest
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] More about the report processing at http://localhost:9000/api/ce/task?id=AXdPSkwWD2pDdw0ilZj4
[INFO] Analysis total time: 13.447 s
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  39.036 s
[INFO] Finished at: 2021-01-29T23:26:45+05:30
[INFO] ------------------------------------------------------------------------

C:\Users\rajvi\Desktop\Accolite\Day11\restapi_au>
```

f. Report:

Overview



CodeSmells



3. Secure Coding Standards:

Secure coding standards are practices that are implemented to prevent the introduction of security vulnerabilities, such as bugs and logic laws. By following secure coding standards, companies can significantly reduce vulnerabilities before deployment.

a. **CWE**:

Common Weakness Enumeration (CWE) list identifies software security weaknesses in software and hardware. This includes C, C++, and Java. The list is compiled by feedback from the CWE Community.

Sponsored by the MITRE Corporation, the community is made up of representatives from major operating systems vendors, commercial information security tool vendors, academia, government agencies, and research institutions.

The full list is regularly updated every few months with the latest version released in August 2020. The security weakness list includes over 600 categories, which include:

- Buffer overflow
- Cross-site scripting
- Insecure random numbers

b. OWASP Top 10

OWASP is the Open Web Application Security Project. It's an international nonprofit organization that educates software development teams on how to conceive, develop, acquire, operate, and maintain secure applications.
OWASP Top 10 is the most well-known resource that the organization produces. Each year, a team of security experts from across the globe updates the report. This report features the 10 most critical web application and API security risks.

The current list includes:

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components With Known Vulnerabilities
10. Insufficient Logging and Monitoring

c. CERT

CERT is a secure coding standard maintained by the Software Engineering Institute at Carnegie Mellon University. It supports commonly used programming languages such as C, C++, and Java. The standards are developed through a broad-based community effort by members of the software development and software security communities. The rules and recommendations target insecure coding practices and undefined behaviors that lead to security risks.

The latest rules and recommendations are available on the secure coding standard's website and are also periodically published: C and C++ in 2016 and Java in 2011.

d. SANS 25

The SANS Institute is a cooperative research and education organization.
The SANS Top 25 Most Dangerous Software Errors is a list of the most widespread and critical errors that can lead to serious vulnerabilities in software (please note: not all vulnerability types apply to all programming languages). The vulnerabilities include insecure interaction between components, risky resource management, and porous defenses.

RIPS is able to detect 24 out of the SANS Top 25 Most Dangerous Software Errors that can be detected by static analysis software, helps you quickly locate them in your application, and provides detailed information on how to fix the errors.

TOP 25 SANS Software Error:

1 PImproper Restriction of Operations within the Bounds of a Memory Buffer
2 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
3 Improper Input Validation
4 Information Exposure
5 Out-of-bounds Read
6 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
7 Use After Free
8 Integer Overflow or Wraparound
9 Cross-Site Request Forgery (CSRF)
10 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
11 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
12 Out-of-bounds Write
13 Improper Authentication
14 NULL Pointer Dereference
15 Incorrect Permission Assignment for Critical Resource
16 Unrestricted Upload of File with Dangerous Type
17 Improper Restriction of XML External Entity Reference
18 Improper Control of Generation of Code ('Code Injection')
19 Use of Hard-coded Credentials
20 Uncontrolled Resource Consumption
21 Missing Release of Resource after Effective Lifetime
22 Untrusted Search Path
23 Deserialization of Untrusted Data
24 Improper Privilege Management
25 Improper Certificate Validation