Name: Aman Yadav Div: D15C Roll No: 60

# Expt No. 08 Advanced DevOps Lab

**Aim:** Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

**Theory:**

## What is SAST?

Static application security testing (SAST), or static analysis, is a testing methodology that analyzes source code to find security vulnerabilities that make your organization's applications susceptible to attack. SAST scans an application before the code is compiled. It's also known as white box testing.

## What problems does SAST solve?

SAST takes place very early in the software development life cycle (SDLC) as it does not require a working application and can take place without code being executed. It helps developers identify vulnerabilities in the initial stages of development and quickly resolve issues without breaking builds or passing on vulnerabilities to the final release of the application.

SAST tools give developers real-time feedback as they code, helping them fix issues before they pass the code to the next phase of the SDLC. This prevents security-related issues from being considered an afterthought. SAST tools also provide graphical representations of the issues found, from source to sink. These help you navigate the code easier. Some tools point out the exact location of vulnerabilities and highlight the risky code. Tools can also provide in-depth guidance on how to fix issues and the best place in the code to fix them, without requiring deep security domain expertise.

It's important to note that SAST tools must be run on the application on a regular basis, such as during daily/monthly builds, every time code is checked in, or during a code release.

### Why is SAST important?

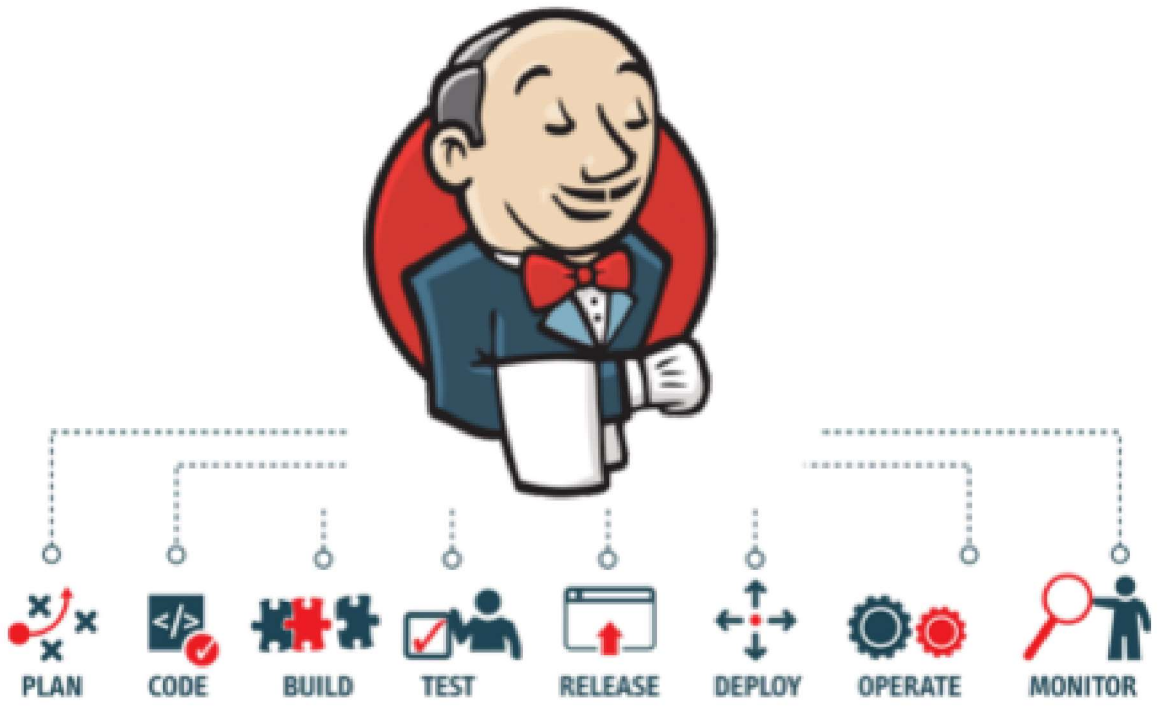
Developers dramatically outnumber security staff. It can be challenging for an organization to find the resources to perform code reviews on even a fraction of its applications. A key strength of SAST tools is the ability to analyze 100% of the codebase. Additionally, they are much faster than manual secure code reviews performed by humans. These tools can scan millions of lines of code in a matter of minutes. SAST tools automatically identify critical vulnerabilities—such as buffer overflows, SQL injection, cross-site scripting, and others—with high confidence.

## What is a CI/CD Pipeline?

Name: Aman Yadav Div: D15C Roll No: 60

CI/CD pipeline refers to the Continuous Integration/Continuous Delivery pipeline. Before we dive deep into this segment, let's first understand what is meant by the term ‘pipeline’?

A pipeline is a concept that introduces a series of events or tasks that are connected in a sequence to make quick software releases. For example, there is a task, that task has got five different stages, and each stage has got some steps. All the steps in phase one have to be completed, to mark the latter stage to be complete.



Now, consider the CI/CD pipeline as the backbone of the DevOps approach. This Pipeline is responsible for building codes, running tests, and deploying new software versions. The Pipeline executes the job in a defined manner by first coding it and then structuring it inside several blocks that may include several steps or tasks.

## What is SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of

code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.
It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

## Benefits of SonarQube

- **Sustainability -** Reduces complexity, possible vulnerabilities, and code duplications, optimizing the life of applications.
- **Increase productivity -** Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
- **Quality code -** Code quality control is an inseparable part of the process of software development.
- **Detect Errors -** Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
- **Increase consistency -** Determines where the code criteria are breached and enhances the quality
- **Business scaling -** No restriction on the number of projects to be evaluated
- **Enhance developer skills -** Regular feedback on quality problems helps developers to improve their coding skills

## Integrating Jenkins with SonarQube:

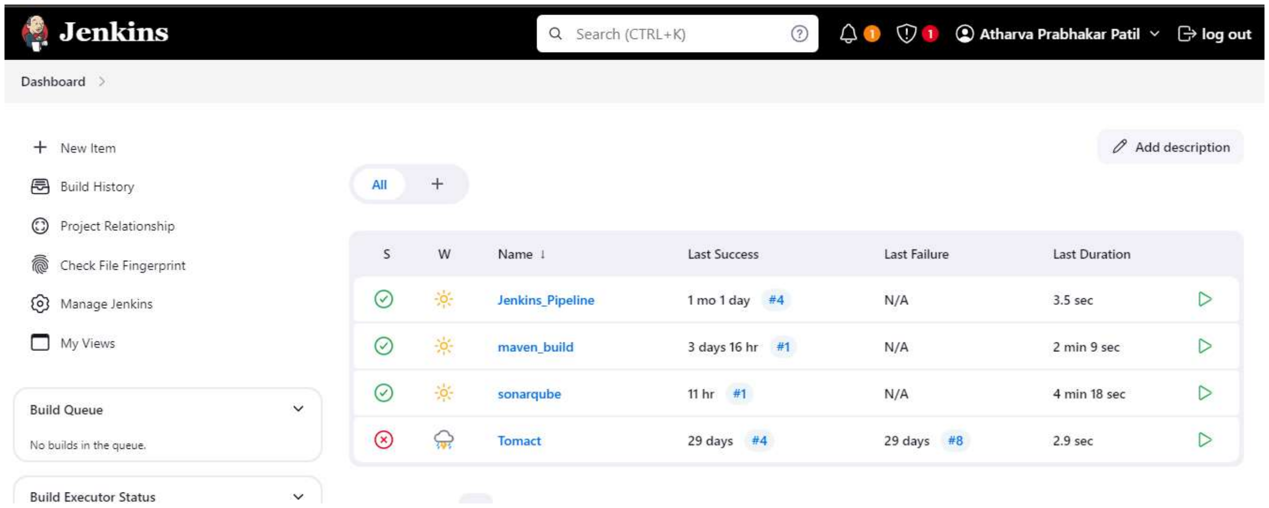
### Prerequisites:

- Jenkins installed
- Docker Installed (for SonarQube)
- SonarQube Docker Image

## Steps to create a Jenkins CI/CD Pipeline and use SonarQube to perform SAST

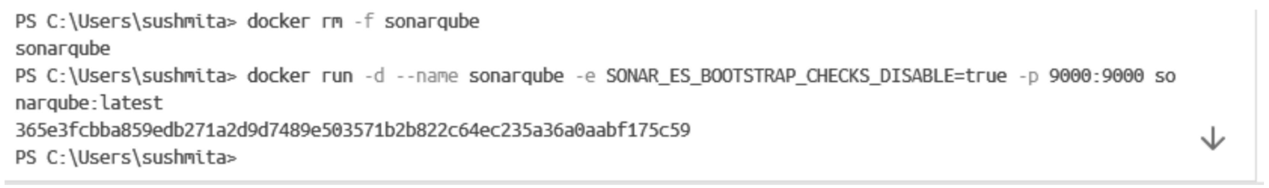
Name: Aman Yadav Div: D15C Roll No: 60

1. Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



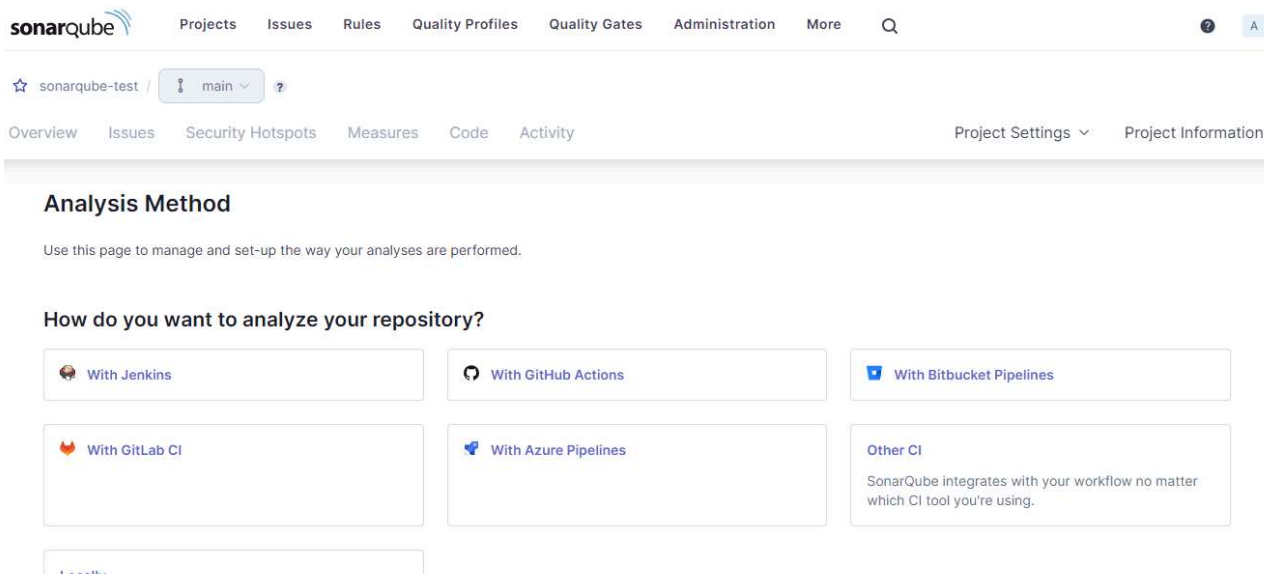
| S | W | Name ↓ | Last Success | Last Failure | Last Duration |
|---|---|---|---|---|---|
| ✓ | ☀ | Jenkins_Pipeline | 1 mo 1 day #4 | N/A | 3.5 sec |
| ✓ | ☀ | maven_build | 3 days 16 hr #1 | N/A | 2 min 9 sec |
| ✓ | ☀ | sonarqube | 11 hr #1 | N/A | 4 min 18 sec |
| ✗ | 🌧 | Tomact | 29 days #4 | 29 days #8 | 2.9 sec |

2. Run SonarQube in a Docker container using this command -



```
PS C:\Users\sushmita> docker rm -f sonarqube
sonarqube
PS C:\Users\sushmita> docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 so
narqube:latest
365e3fcbba859edb271a2d9d7489e503571b2b822c64ec235a36a0aabf175c59
PS C:\Users\sushmita>
```

3. Once the container is up and running, you can check the status of SonarQube at localhost port 9000.



Analysis Method

Use this page to manage and set-up the way your analyses are performed.

How do you want to analyze your repository?

With Jenkins

With GitHub Actions

With Bitbucket Pipelines

With GitLab CI

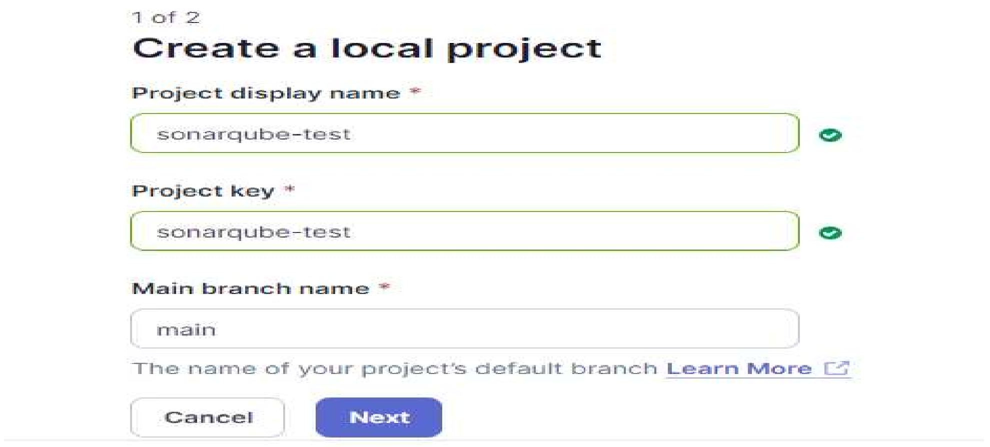
With Azure Pipelines

Other CI

SonarQube integrates with your workflow no matter which CI tool you're using.

4. Login to SonarQube using username *admin* and password *admin*.

5. Create a local project in SonarQube with the name **sonarqube-test**



1 of 2

Create a local project

Project display name *: sonarqube-test

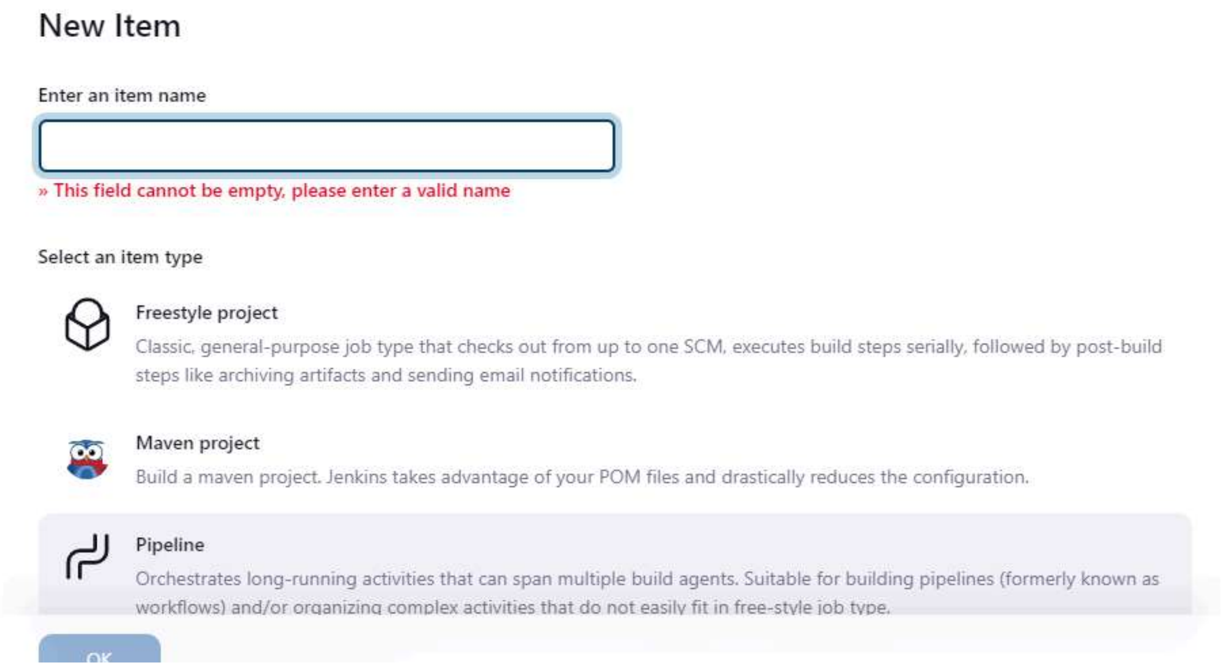
Project key *: sonarqube-test

Main branch name *: main

The name of your project's default branch Learn More

Setup the project and come back to Jenkins Dashboard.
Download the sonar scanner and place it in a directory using
mkdir Downloads\sonarqube
cd Downloads/sonarqube
Download by pasting the link in browser:
https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.2.0.1873-windows.zip
Paste the following command in Docker terminal
unzip sonar-scanner-cli-4.2.0.1873-windows.zip
Also download the latest java version

6. Create a New Item in Jenkins, choose **Pipeline**.



New Item

Enter an item name

» This field cannot be empty, please enter a valid name

Select an item type

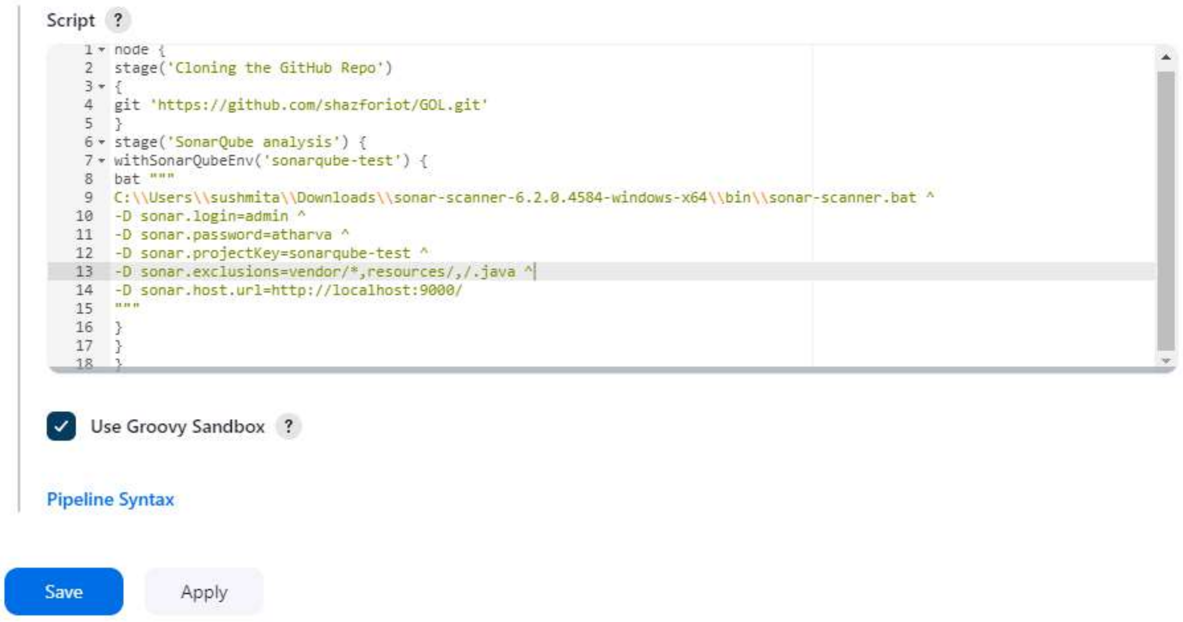
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

7. Under Pipeline Script, enter the following -

```
node {
stage('Cloning the GitHub Repo')
{
git 'https://github.com/shazforiot/GOL.git'
}
stage('SonarQube analysis') {
withSonarQubeEnv('sonarqube-test') {
bat """
C:\\Users\\sushmita\\Downloads\\sonar-scanner-6.2.0.4584-windows-x6
4\\bin\\sonar-scanner.bat ^
-D sonar.login=admin ^
-D sonar.password=atharva ^
-D sonar.projectKey=sonarqube-test ^
-D sonar.exclusions=vendor/*,resources/,/.java ^
-D sonar.host.url=http://localhost:9000/
"""
}
}
}
```

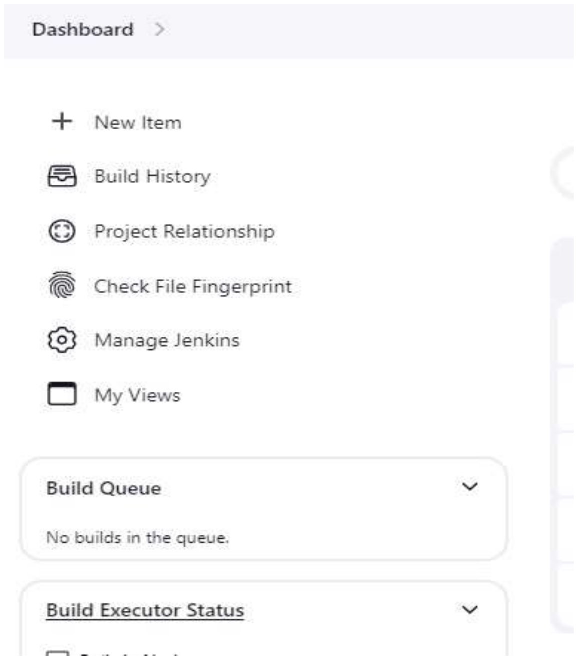

Script ?

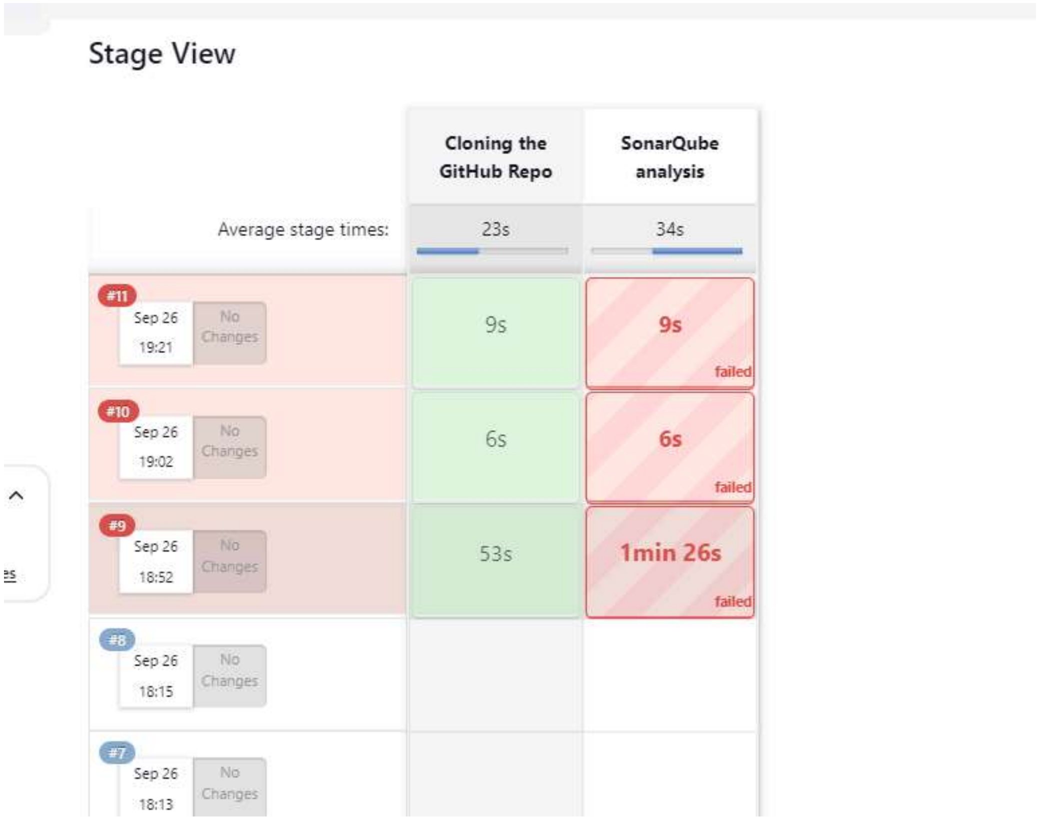
```
node {
stage('Cloning the GitHub Repo')
{
git 'https://github.com/shazforiot/GOL.git'
}
stage('SonarQube analysis') {
withSonarQubeEnv('sonarqube-test') {
bat """
C:\\Users\\sushmita\\Downloads\\sonar-scanner-6.2.0.4584-windows-x64\\bin\\sonar-scanner.bat ^
-D sonar.login=admin ^
-D sonar.password=atharva ^
-D sonar.projectKey=sonarqube-test ^
-D sonar.exclusions=vendor/*,resources/,/.java ^
-D sonar.host.url=http://localhost:9000/
"""
}
}
}
```

☑ Use Groovy Sandbox ?

It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

8. Run The Build.

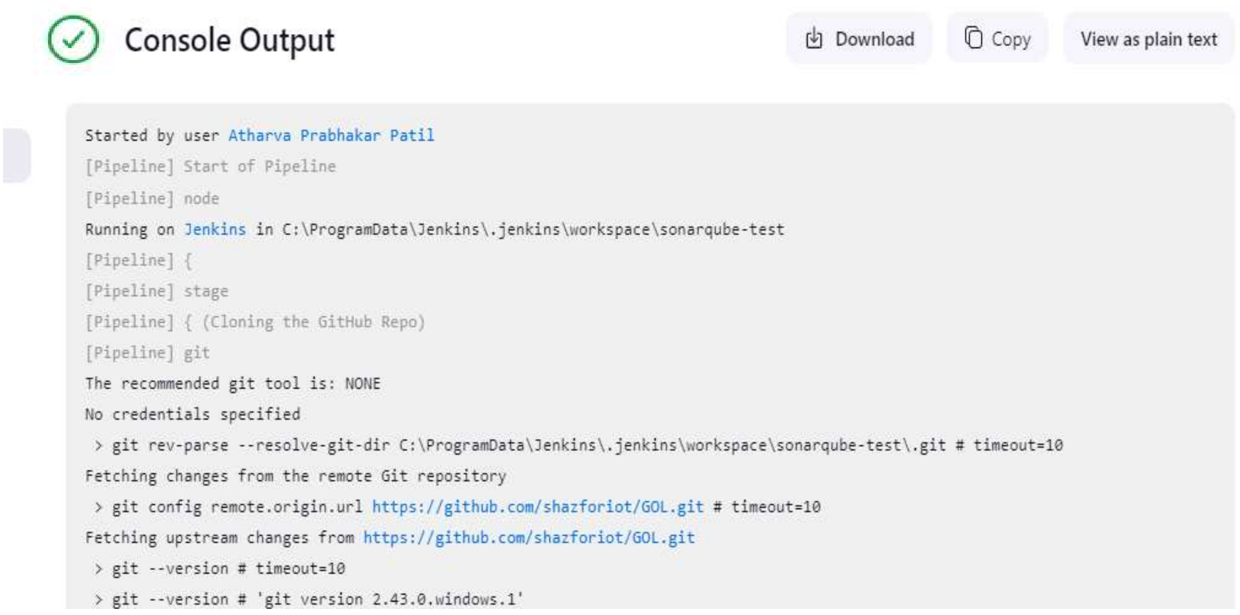



Stage View

| | Cloning the GitHub Repo | SonarQube analysis |
|---|---|---|
| Average stage times: | 23s | 34s |
| #11 Sep 26 19:21 No Changes | 9s | 9s failed |
| #10 Sep 26 19:02 No Changes | 6s | 6s failed |
| #9 Sep 26 18:52 No Changes | 53s | 1min 26s failed |
| #8 Sep 26 18:15 No Changes | | |
| #7 Sep 26 18:13 No Changes | | |

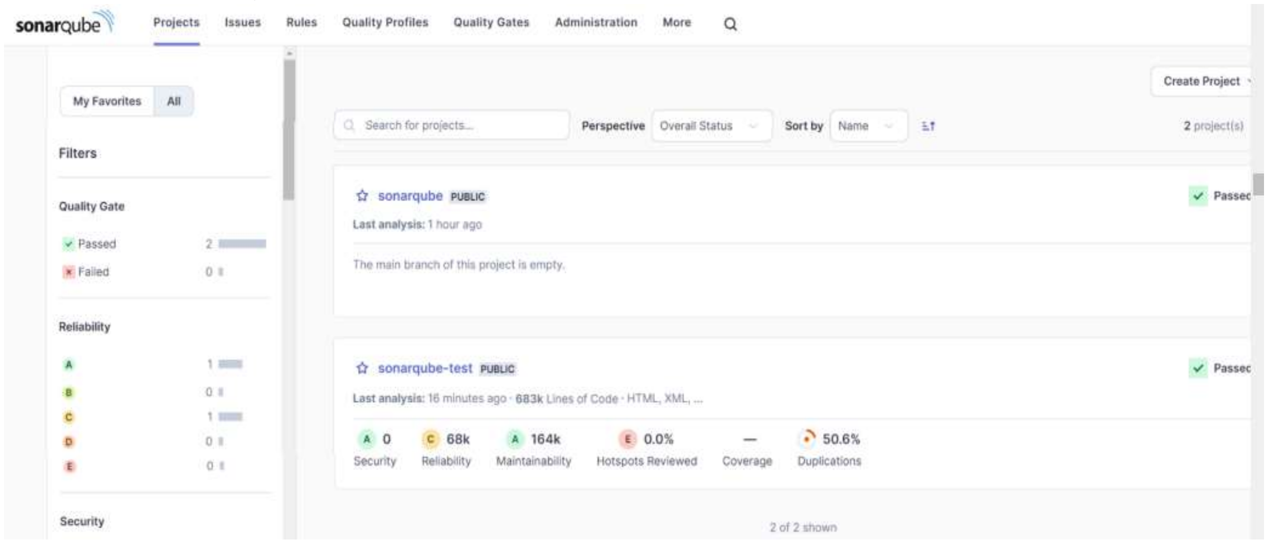
9. Check the console output once the build is complete.

Name: Aman Yadav Div: D15C Roll No: 60



```
Started by user Atharva Prabhakar Patil
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning the GitHub Repo)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test\.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
 > git --version # timeout=10
 > git --version # 'git version 2.43.0.windows.1'
```

10. After that, check the project in SonarQube.



sonarqube PUBLIC — Passed
Last analysis: 1 hour ago
The main branch of this project is empty.

sonarqube-test PUBLIC — Passed
Last analysis: 16 minutes ago · 683k Lines of Code · HTML, XML, ...

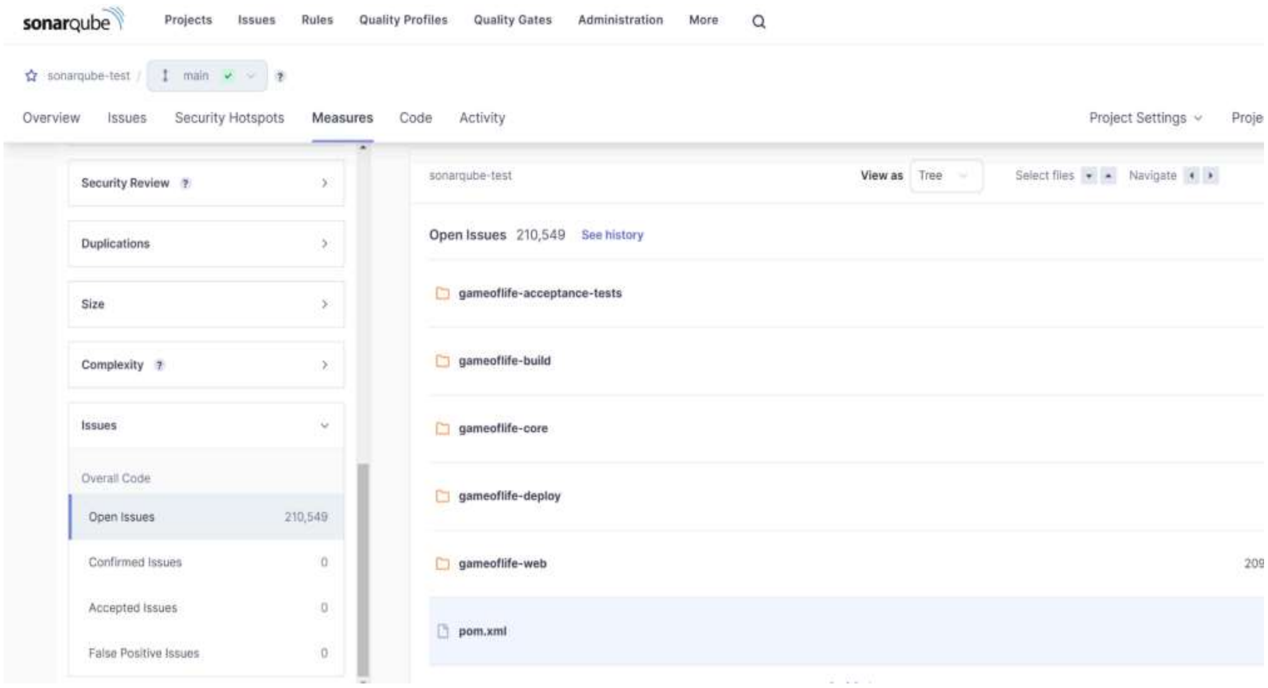
A 0 Security · C 68k Reliability · A 164k Maintainability · E 0.0% Hotspots Reviewed · — Coverage · 50.6% Duplications

2 of 2 shown

Name: Aman Yadav Div: D15C Roll No: 60

Under different tabs, check all different issues with the code.
11. Code Problems -

**Open Issues**



Overall Code

| | |
|---|---|
| Open Issues | 210,549 |
| Confirmed Issues | 0 |
| Accepted Issues | 0 |
| False Positive Issues | 0 |

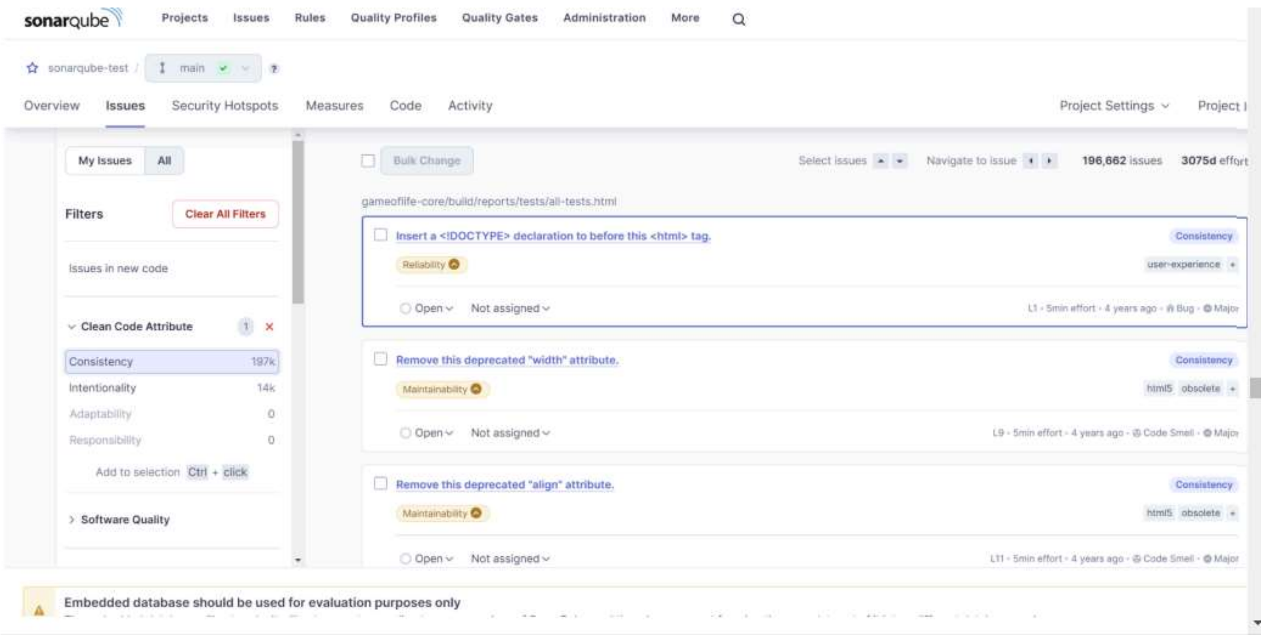
sonarqube-test

Open Issues 210,549 See history

gameoflife-acceptance-tests
gameoflife-build
gameoflife-core
gameoflife-deploy
gameoflife-web 209
pom.xml

**Consistency**

Name: Aman Yadav Div: D15C Roll No: 60



196,662 issues 3075d effort

gameoflife-core/build/reports/tests/all-tests.html

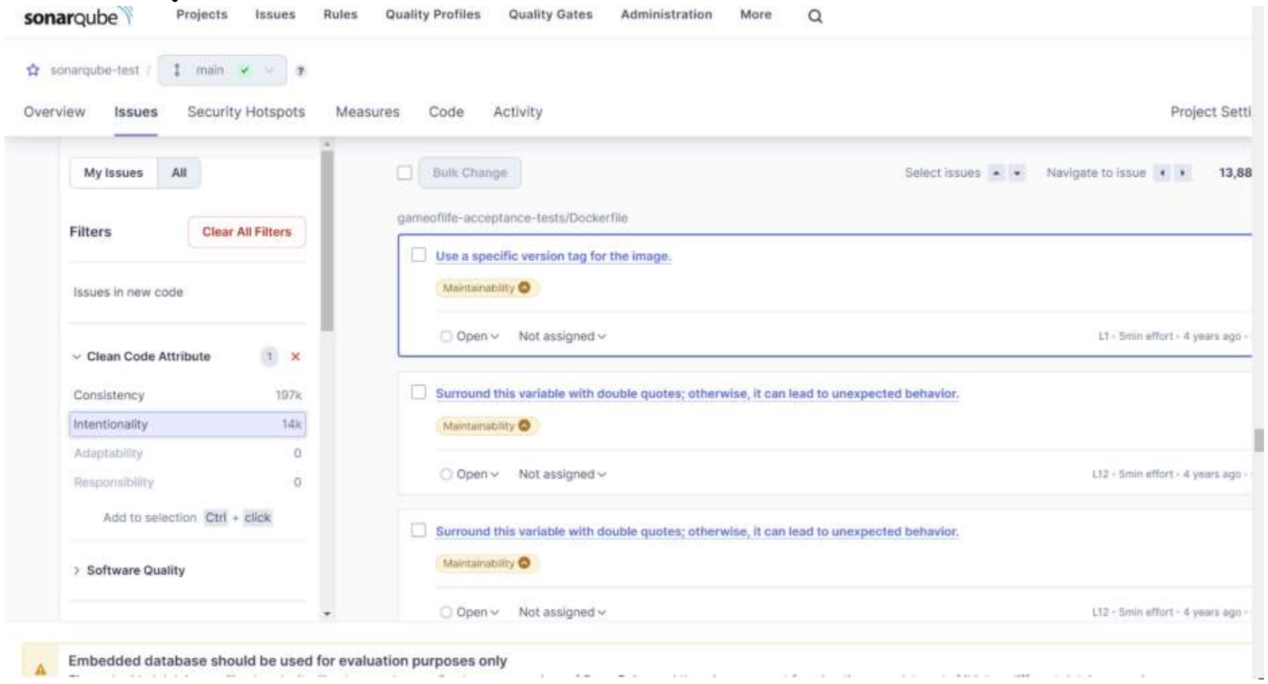
Insert a <!DOCTYPE> declaration to before this <html> tag. — Consistency — Reliability — user-experience
Open · Not assigned · L1 · 5min effort · 4 years ago · Bug · Major

Remove this deprecated "width" attribute. — Consistency — Maintainability — html5 obsolete
Open · Not assigned · L9 · 5min effort · 4 years ago · Code Smell · Major

Remove this deprecated "align" attribute. — Consistency — Maintainability — html5 obsolete
Open · Not assigned · L11 · 5min effort · 4 years ago · Code Smell · Major

Embedded database should be used for evaluation purposes only

## Intentionality



gameoflife-acceptance-tests/Dockerfile

Use a specific version tag for the image. — Maintainability
Open · Not assigned · L1 · 5min effort · 4 years ago

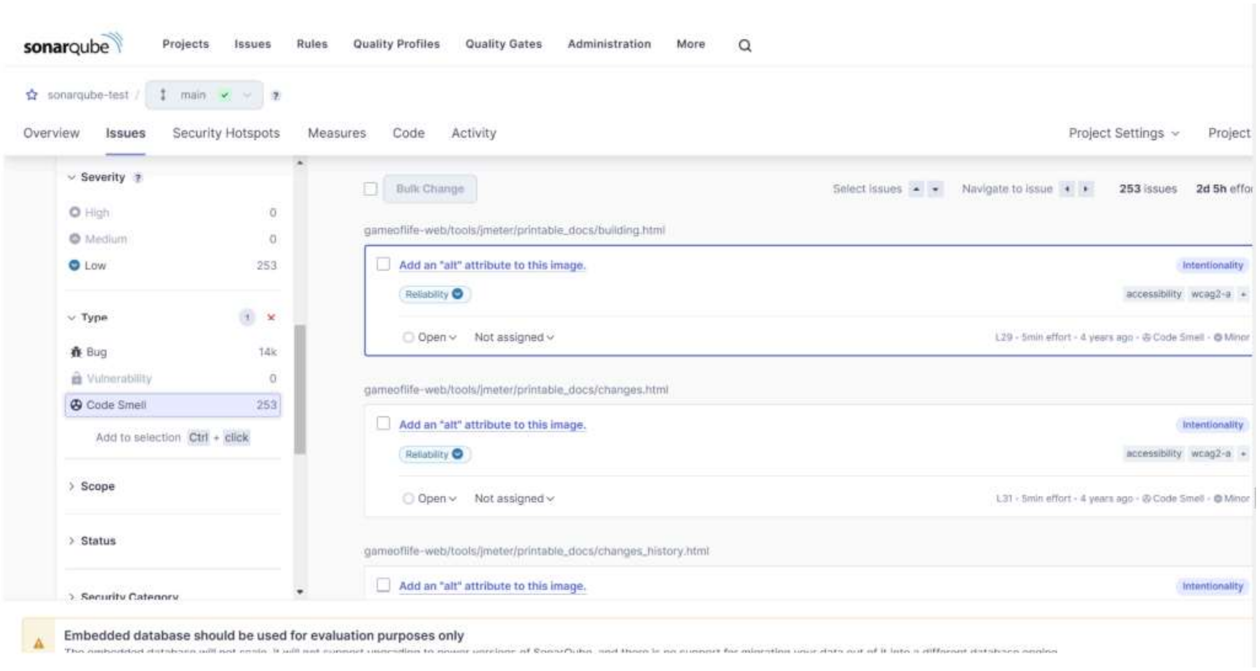
Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. — Maintainability
Open · Not assigned · L12 · 5min effort · 4 years ago

Surround this variable with double quotes; otherwise, it can lead to unexpected behavior. — Maintainability
Open · Not assigned · L12 · 5min effort · 4 years ago

Embedded database should be used for evaluation purposes only

Name: Aman Yadav Div: D15C Roll No: 60

## Bugs and Code Smells



Severity

| | |
|---|---|
| High | 0 |
| Medium | 0 |
| Low | 253 |

Type

| | |
|---|---|
| Bug | 14k |
| Vulnerability | 0 |
| Code Smell | 253 |

253 issues 2d 5h effo

gameoflife-web/tools/jmeter/printable_docs/building.html

Add an "alt" attribute to this image. Intentionality
Reliability accessibility wcag2-a
Open Not assigned L29 · 5min effort · 4 years ago · Code Smell · Minor

gameoflife-web/tools/jmeter/printable_docs/changes.html

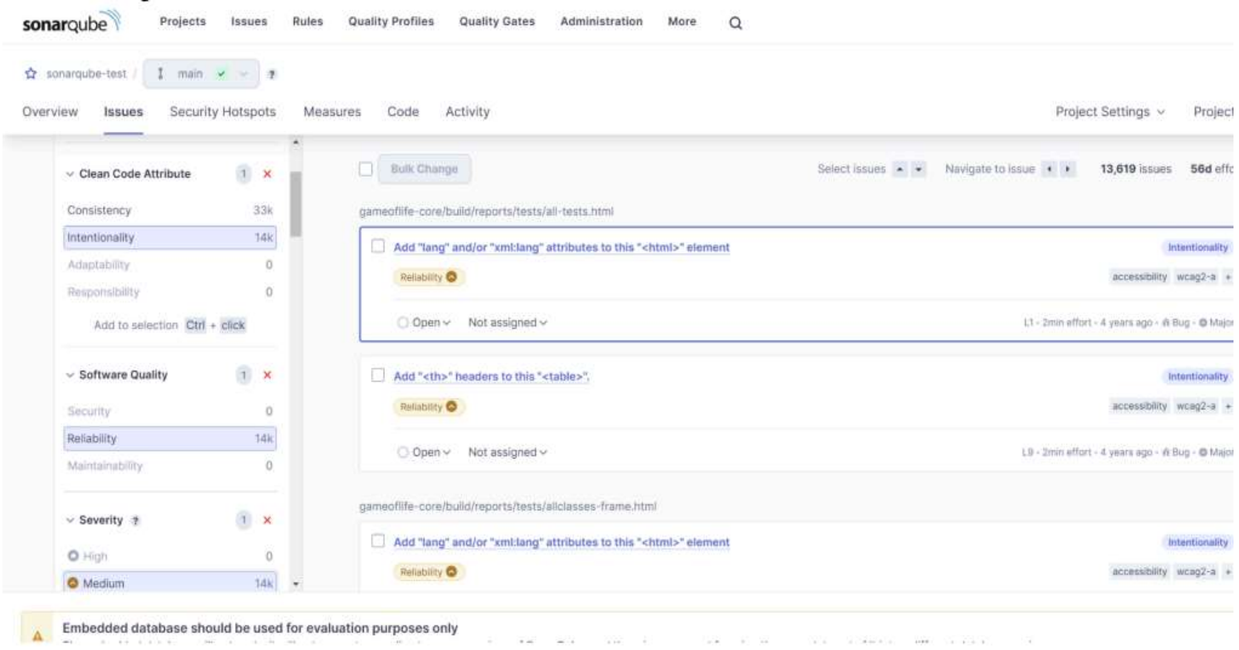
Add an "alt" attribute to this image. Intentionality
Reliability accessibility wcag2-a
Open Not assigned L31 · 5min effort · 4 years ago · Code Smell · Minor

gameoflife-web/tools/jmeter/printable_docs/changes_history.html

Add an "alt" attribute to this image. Intentionality

Embedded database should be used for evaluation purposes only

## Reliability



Clean Code Attribute

| | |
|---|---|
| Consistency | 33k |
| Intentionality | 14k |
| Adaptability | 0 |
| Responsibility | 0 |

Software Quality

| | |
|---|---|
| Security | 0 |
| Reliability | 14k |
| Maintainability | 0 |

Severity

| | |
|---|---|
| High | 0 |
| Medium | 14k |

13,619 issues 56d effo

gameoflife-core/build/reports/tests/all-tests.html

Add "lang" and/or "xml:lang" attributes to this "<html>" element Intentionality
Reliability accessibility wcag2-a
Open Not assigned L1 · 2min effort · 4 years ago · Bug · Major

Add "<th>" headers to this "<table>". Intentionality
Reliability accessibility wcag2-a
Open Not assigned L9 · 2min effort · 4 years ago · Bug · Major

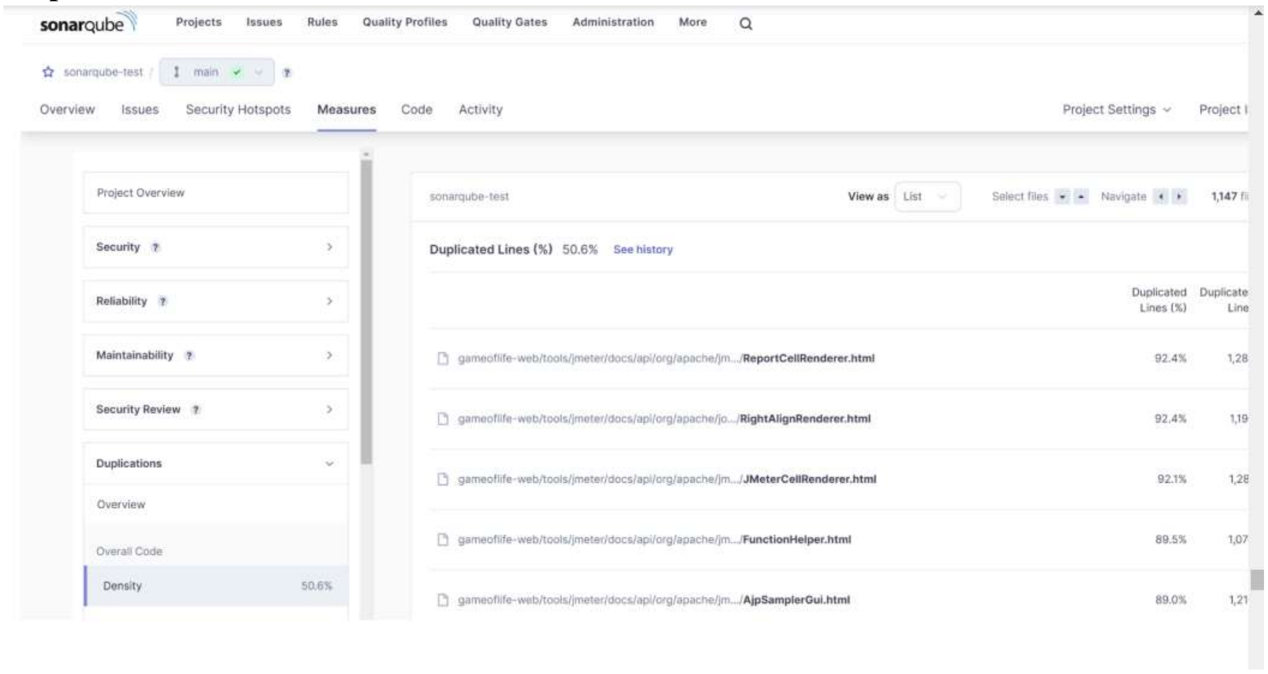
gameoflife-core/build/reports/tests/allclasses-frame.html

Add "lang" and/or "xml:lang" attributes to this "<html>" element Intentionality
Reliability accessibility wcag2-a

Embedded database should be used for evaluation purposes only

Name: Aman Yadav Div: D15C Roll No: 60

## Duplicates



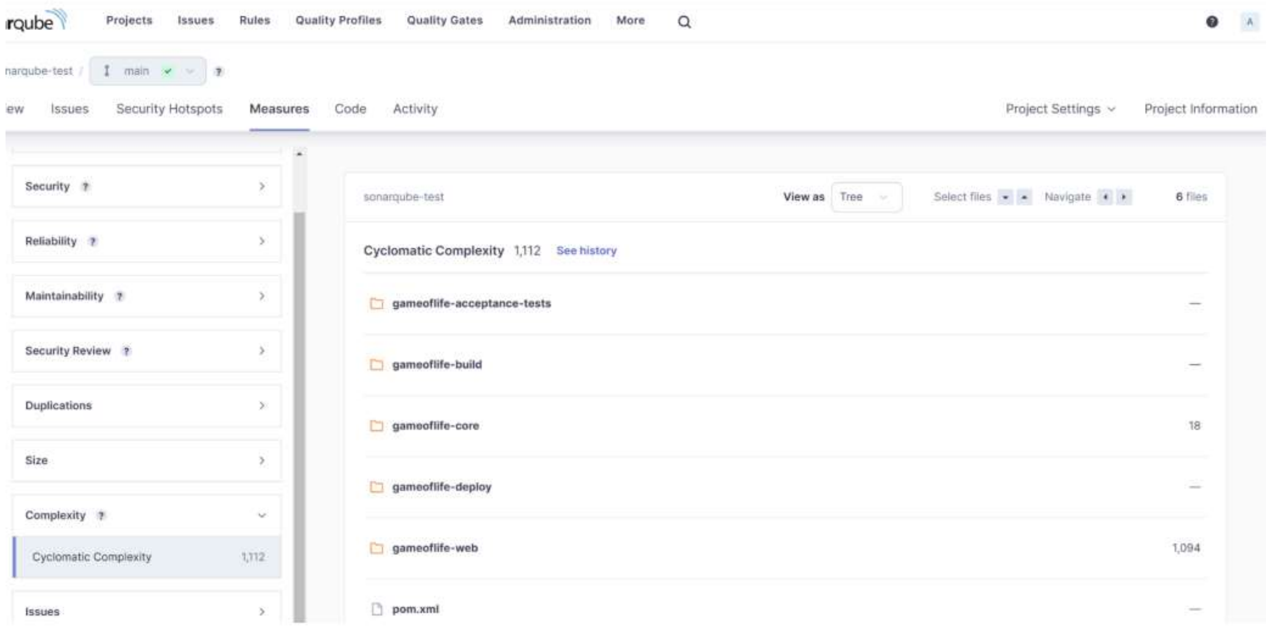
Project Overview | Security | Reliability | Maintainability | Security Review | Duplications | Overview | Overall Code | Density 50.6%

sonarqube-test — View as List — 1,147 fi

Duplicated Lines (%) 50.6% See history

| | Duplicated Lines (%) | Duplicate Line |
|---|---|---|
| gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../ReportCellRenderer.html | 92.4% | 1,28 |
| gameoflife-web/tools/jmeter/docs/api/org/apache/jo.../RightAlignRenderer.html | 92.4% | 1,19 |
| gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../JMeterCellRenderer.html | 92.1% | 1,28 |
| gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../FunctionHelper.html | 89.5% | 1,07 |
| gameoflife-web/tools/jmeter/docs/api/org/apache/jm.../AjpSamplerGui.html | 89.0% | 1,21 |

## Cyclomatic Complexities



Security | Reliability | Maintainability | Security Review | Duplications | Size | Complexity | Cyclomatic Complexity 1,112 | Issues

sonarqube-test — View as Tree — 6 files

Cyclomatic Complexity 1,112 See history

| | |
|---|---|
| gameoflife-acceptance-tests | — |
| gameoflife-build | — |
| gameoflife-core | 18 |
| gameoflife-deploy | — |
| gameoflife-web | 1,094 |
| pom.xml | — |

In this way, we have created a CI/CD Pipeline with Jenkins and integrated it with SonarQube to find issues in the code like bugs, code smells, duplicates, cyclomatic complexities, etc.

**Conclusion:**

In this experiment, we performed a static analysis of the code to detect bugs, code smells, and security vulnerabilities on our sample Java application.