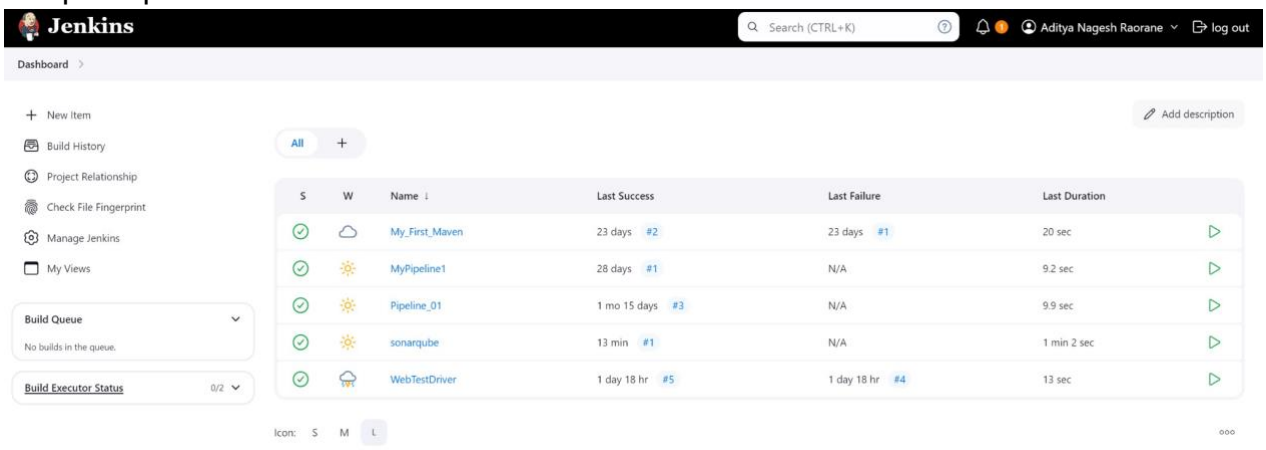
Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

1. Open up Jenkins Dashboard on localhost:8080.

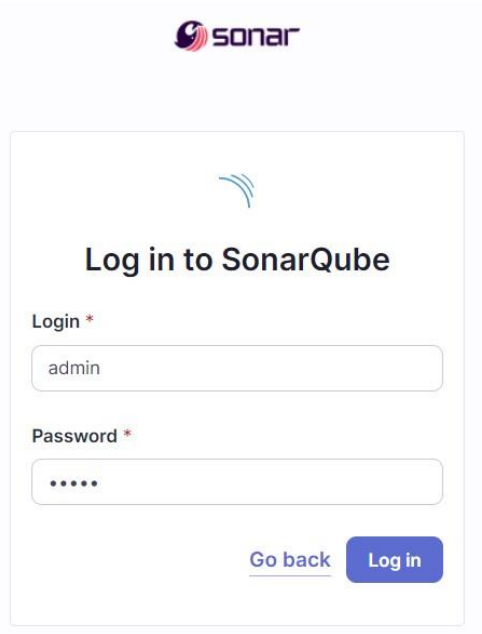


2. Run SonarQube in a Docker container using this command: a] docker -v b] docker pull sonarqube c] docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

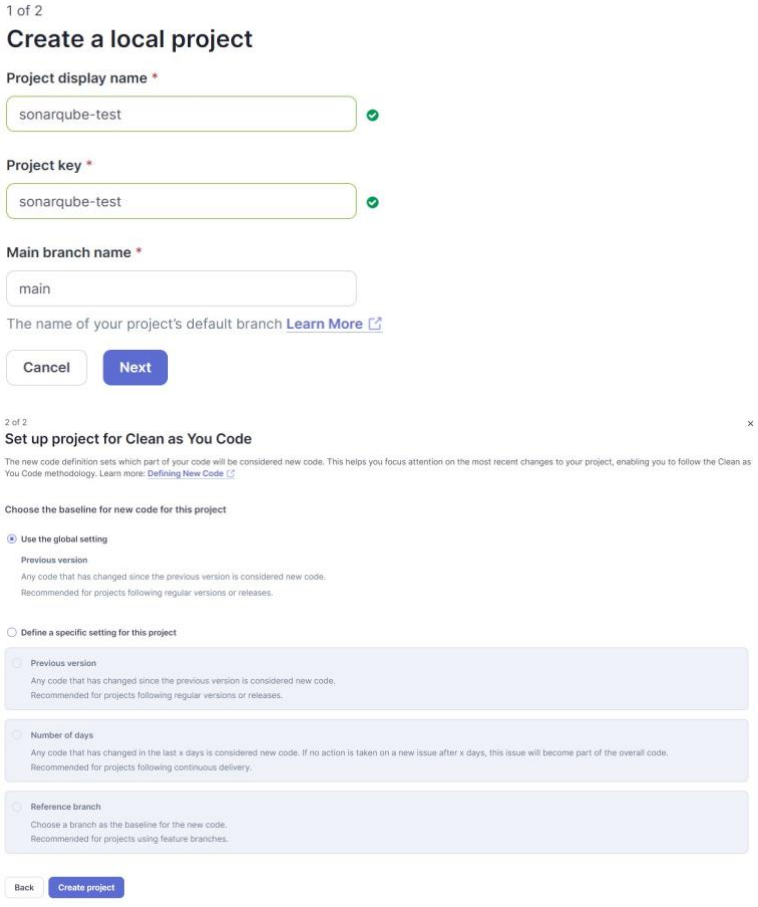
```
C:\Users\adity>docker -v
Docker version 27.0.3, build 7d4bcd8

C:\Users\adity>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
Unable to find image 'sonarqube:latest' locally
latest: Pulling from library/sonarqube
7478e0ac0f23: Pull complete
90a925ab929a: Pull complete
7d9a34308537: Pull complete
80338217a4ab: Pull complete
1a5fd5c7e184: Pull complete
7b87d6fa783d: Pull complete
bd819c9b5ead: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:72e9feec71242af83faf65f95a40d5e3bb2822a6c3b2cda8568790f3d31aecde
Status: Downloaded newer image for sonarqube:latest
4a6e73f4472de892b1ddead1abe77372a85a7b09408cce3a0abd37c5ab6b49a4
```

3. Once the container is up and running, you can check the status of SonarQube at **localhost port 9000**. The login id is “**admin**” and the password is “**aditya**”.

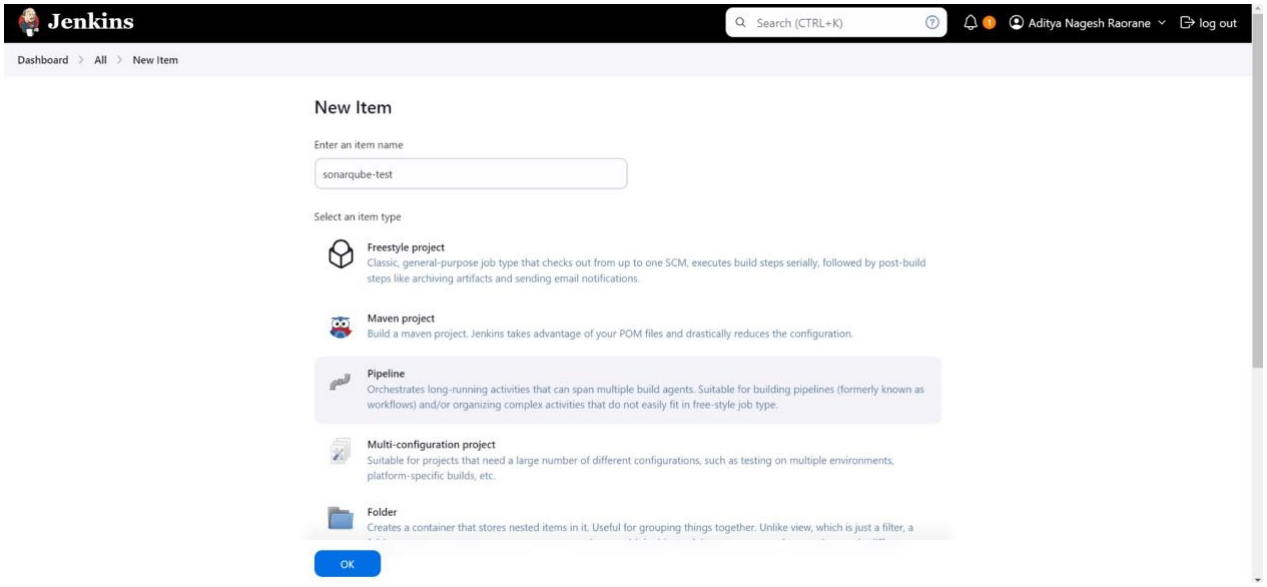


4. **Create a local project in SonarQube** with the name **sonarqube-test**.



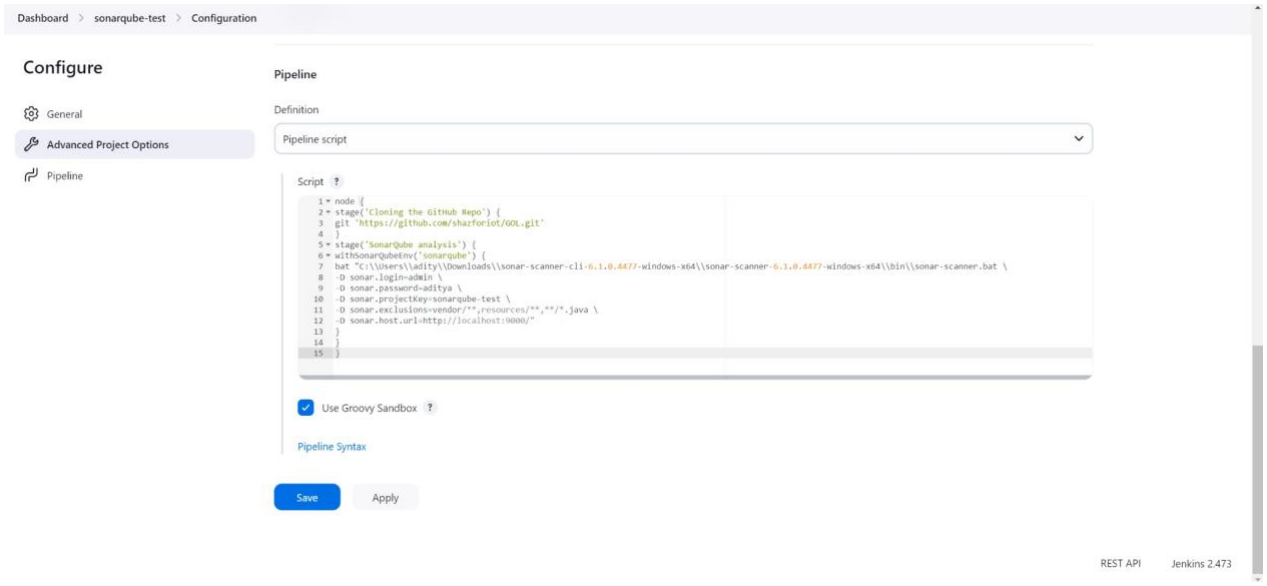
Setup the project and come back to Jenkins Dashboard.

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



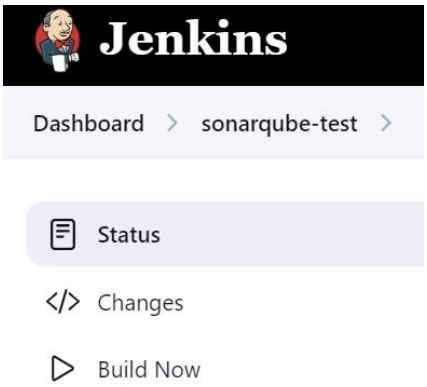
7. Under **Pipeline Script**, enter the following -

```
node { stage('Cloning the GitHub
        Repo')
        {
                git 'https://github.com/shazforiot/GOL.git'
        }
stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
        bat
        "C:\\Users\\adity\\Downloads\\sonar-scanner-cli-6.1.0.4477-windows-x64\\sonar-s
        canner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat \
                -D sonar.login=<YOUR ID> \
                -D sonar.password=<YOUR PASSWORD> \
                -D sonar.projectKey=<YOUR PROJECT KEY> \
                -D sonar.exclusions=vendor/**,resources/**,**/*.java \
                -D sonar.host.url=http://localhost:9000/"
                }
        }
}
```

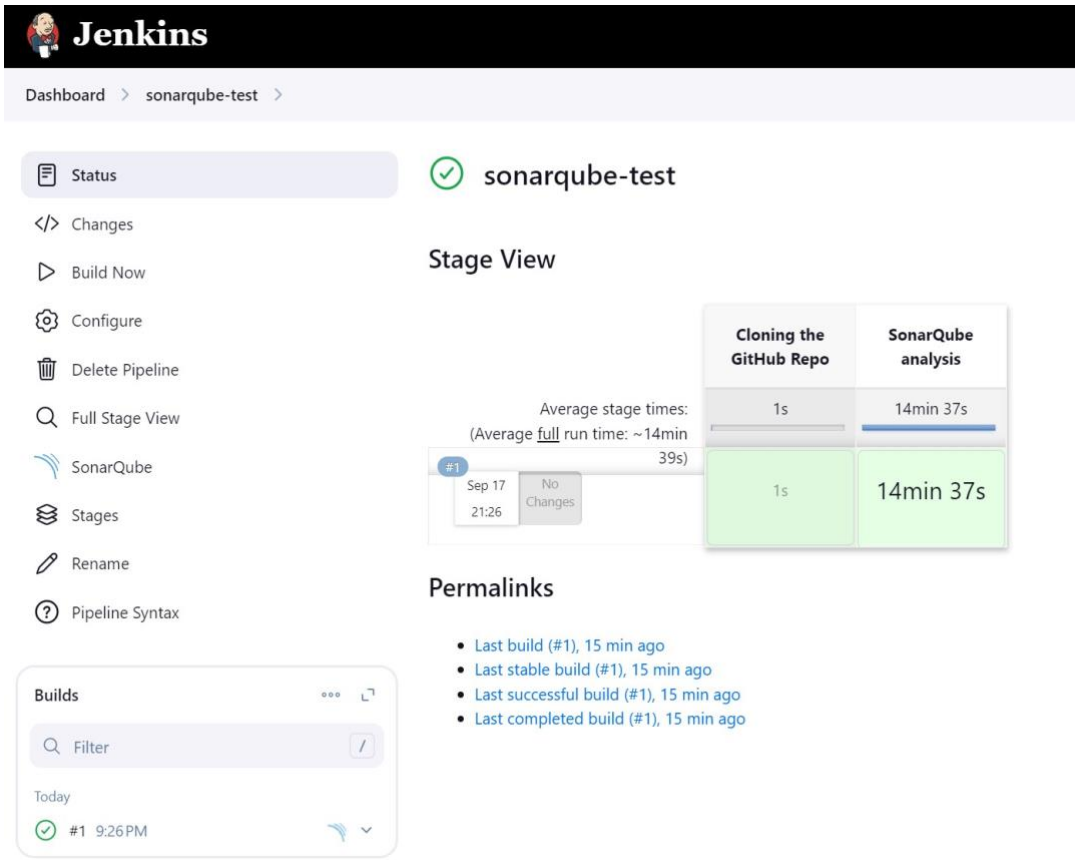


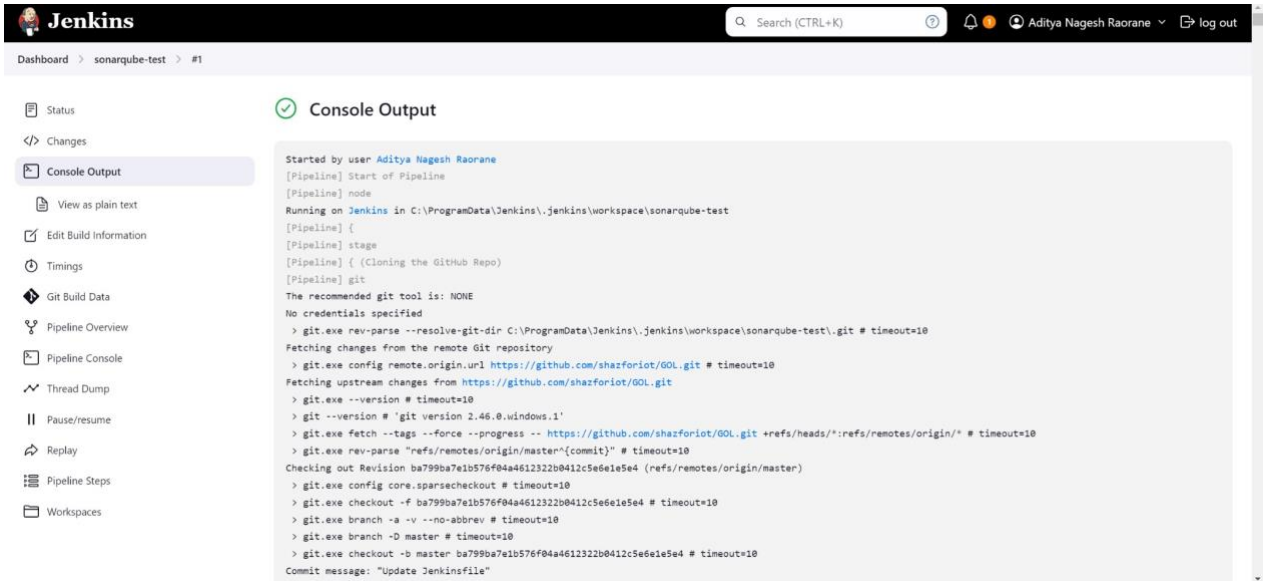
It is a java sample project which has a lot of repetitions and issues that will be detected by SonarQube.

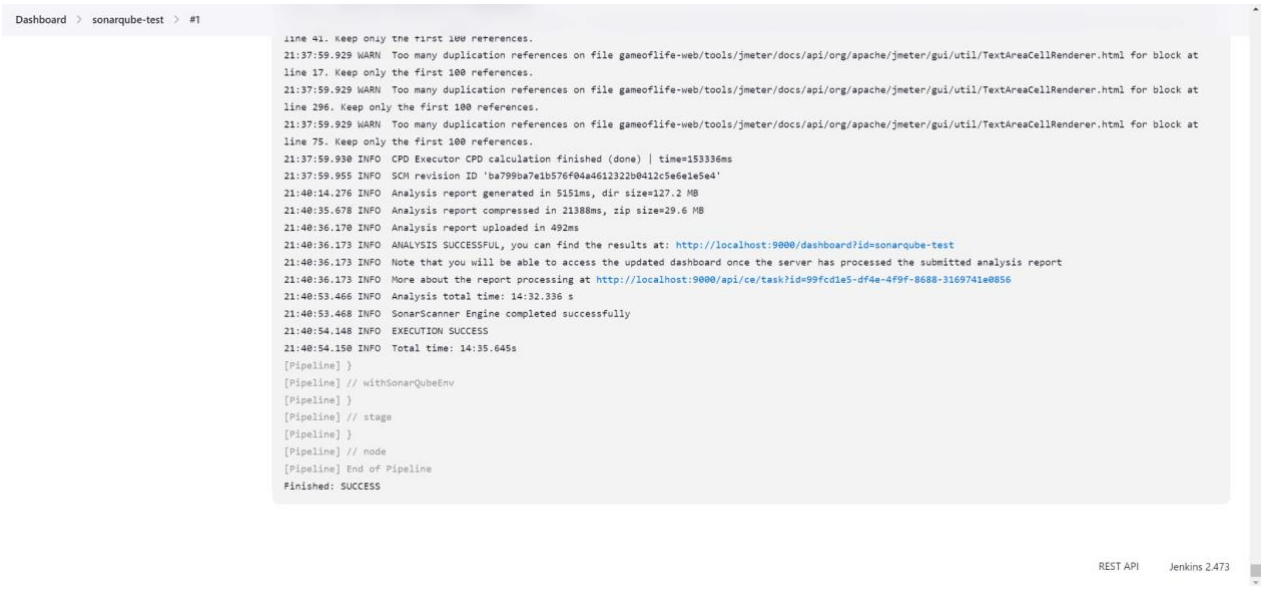
8. Run The Build.



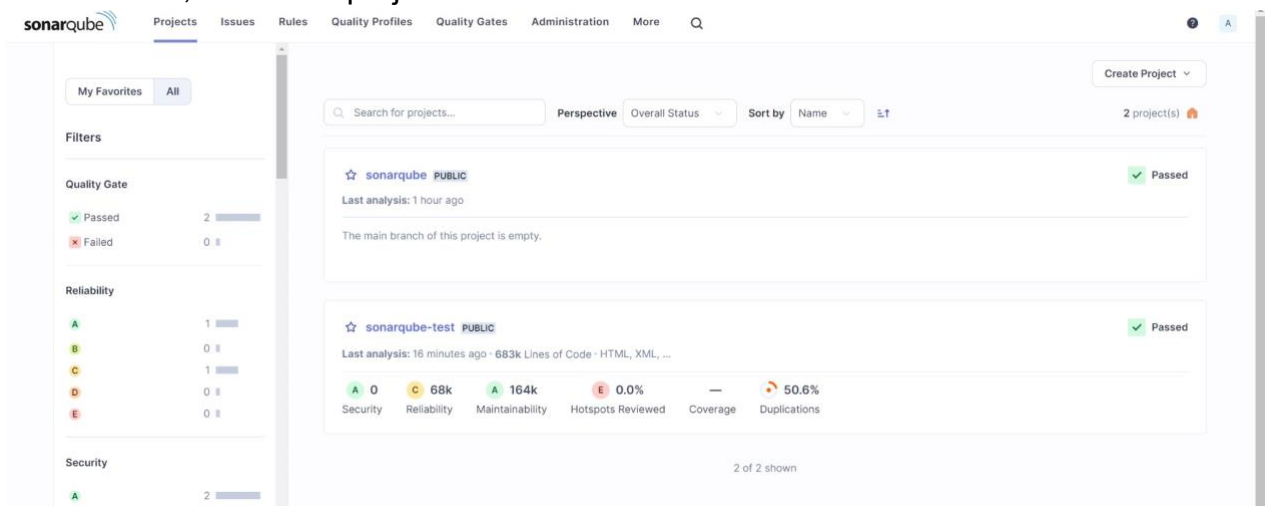
9. Check the console output once the build is complete.

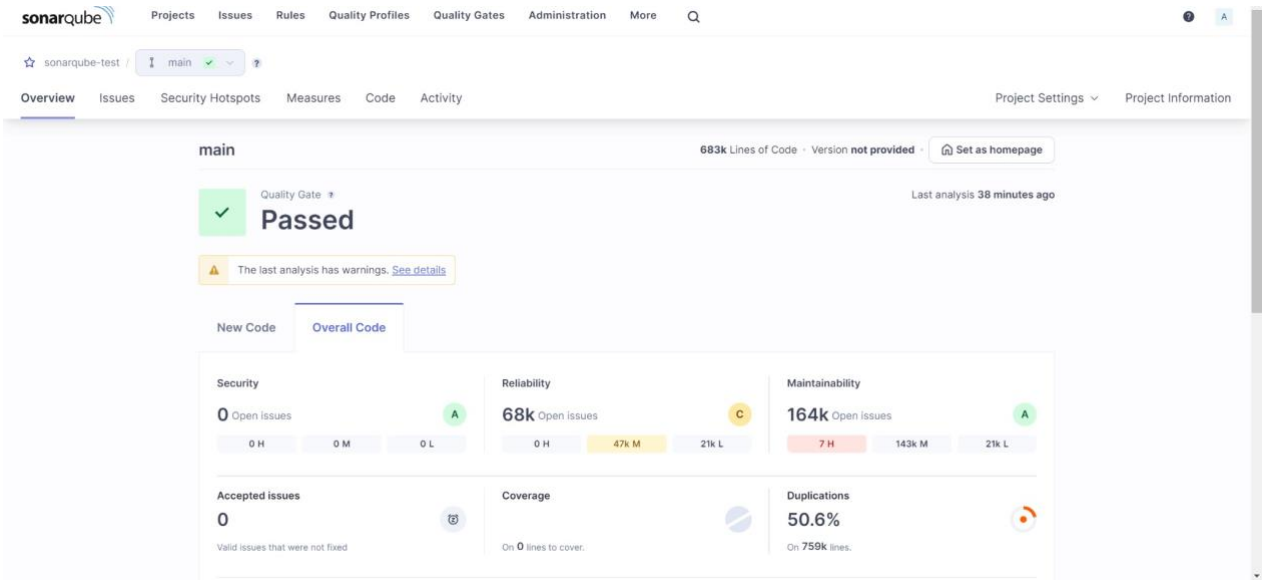

Jenkins
Dashboard > sonarqube-test >
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
SonarQube
Stages
Rename
Pipeline Syntax
Builds
Filter
Today
#1 9:26 PM
sonarqube-test
Stage View
Cloning the GitHub Repo
SonarQube analysis
Average stage times:
(Average full run time: ~14min 39s)
1s
14min 37s
#1
Sep 17 21:26
No Changes
1s
14min 37s
Permalinks
Last build (#1), 15 min ago
Last stable build (#1), 15 min ago
Last successful build (#1), 15 min ago
Last completed build (#1), 15 min ago


Jenkins
Search (CTRL+K)
Aditya Nagesh Raorane
log out
Dashboard > sonarqube-test > #1
Status
Changes
Console Output
View as plain text
Edit Build Information
Timings
Git Build Data
Pipeline Overview
Pipeline Console
Thread Dump
Pause/resume
Replay
Pipeline Steps
Workspaces
Console Output
Started by user Aditya Nagesh Raorane
[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.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\sonarqube-test\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/shazforiot/GOL.git # timeout=10
Fetching upstream changes from https://github.com/shazforiot/GOL.git
> git.exe --version # timeout=10
> git --version # 'git version 2.46.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/shazforiot/GOL.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
> git.exe branch -a -v --no-abbrev # timeout=10
> git.exe branch -D master # timeout=10
> git.exe checkout -b master ba799ba7e1b576f04a4612322b0412c5e6e1e5e4 # timeout=10
Commit message: "Update Jenkinsfile"



10. After that, check the project in SonarQube.

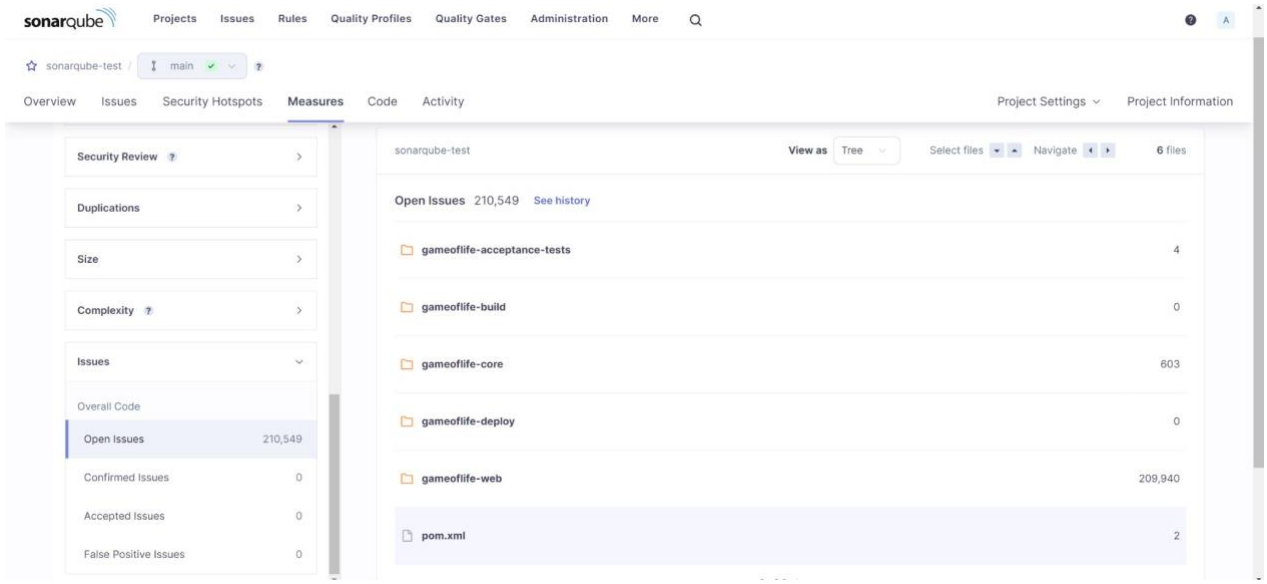




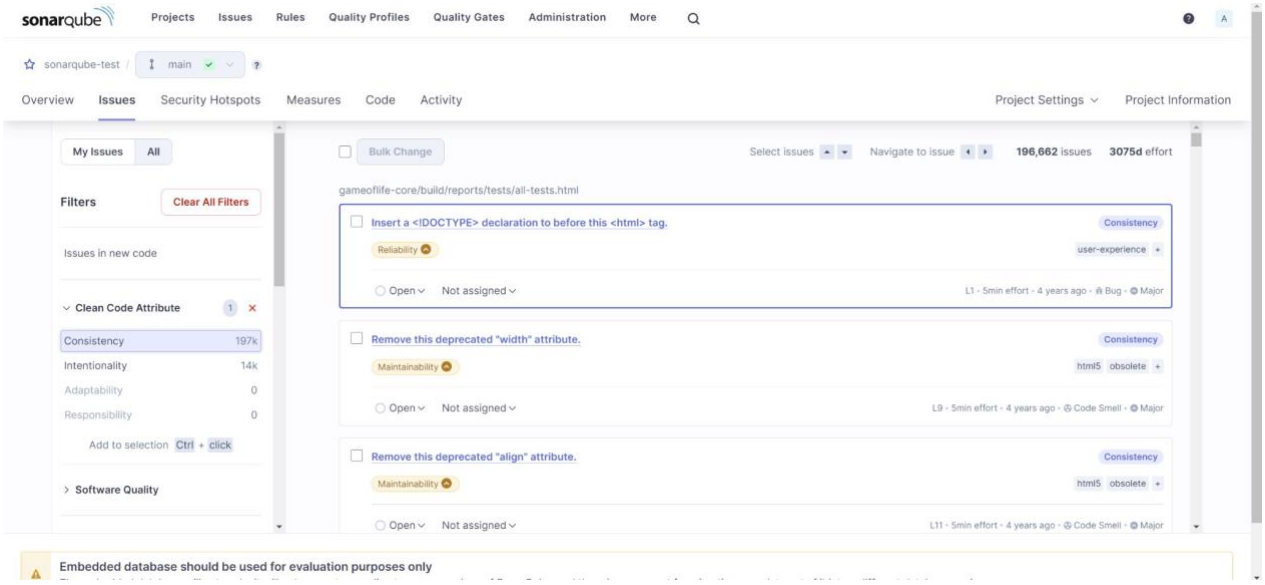
Under different tabs, check all different issues with the code.

11. **Code Problems**

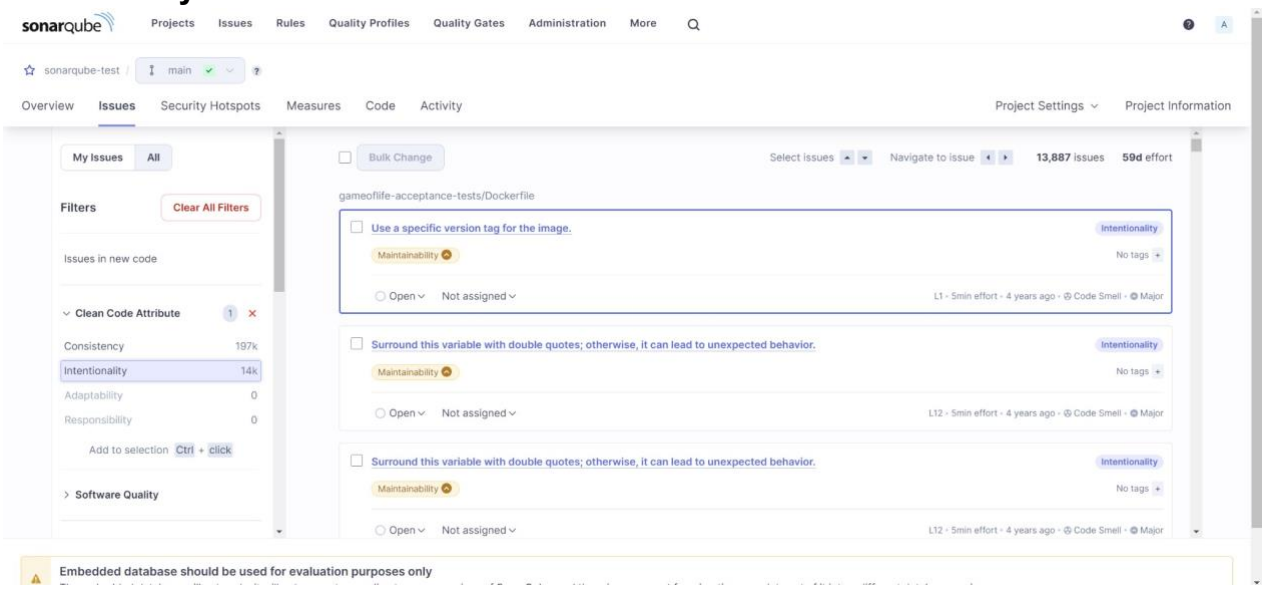
Open Issues



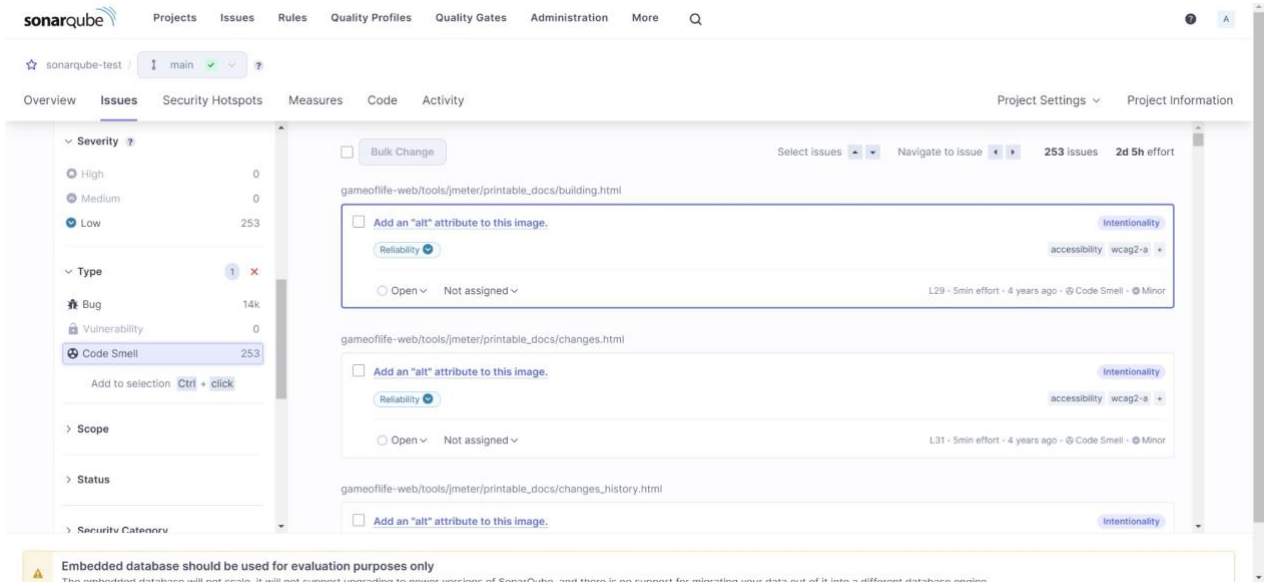
Consistency



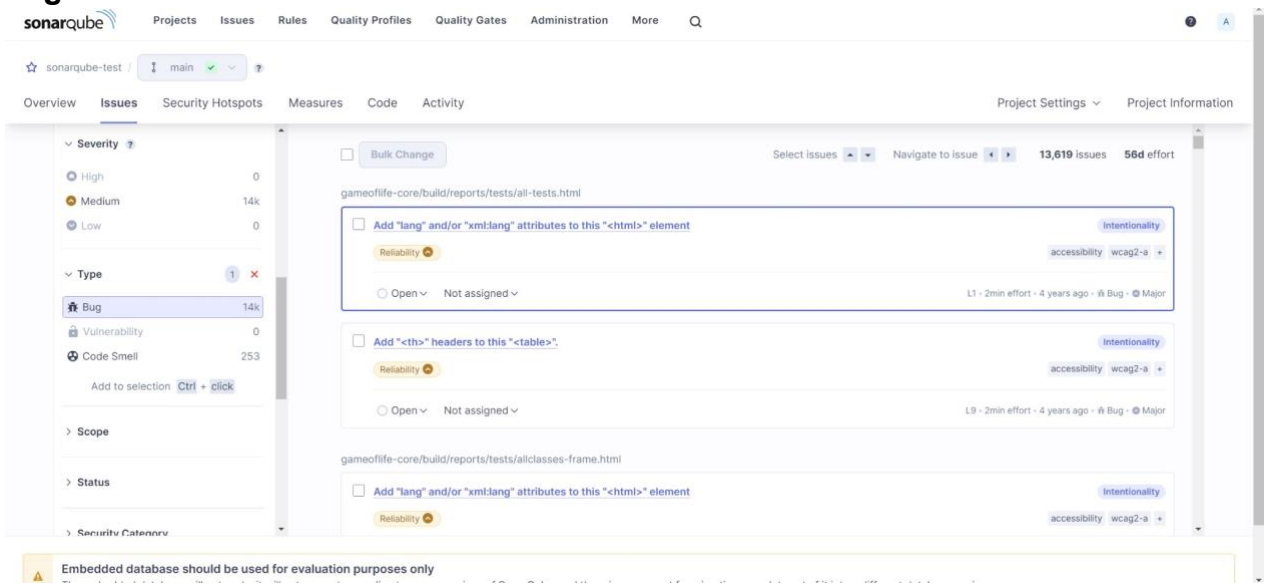
Intentionality



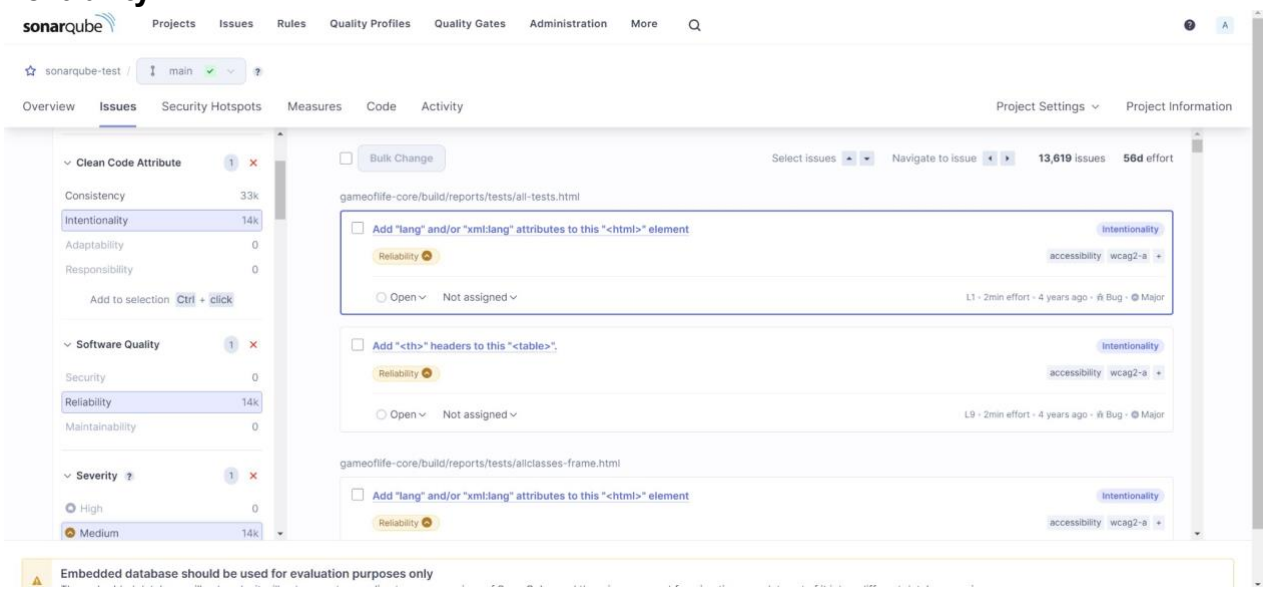
Code Smells



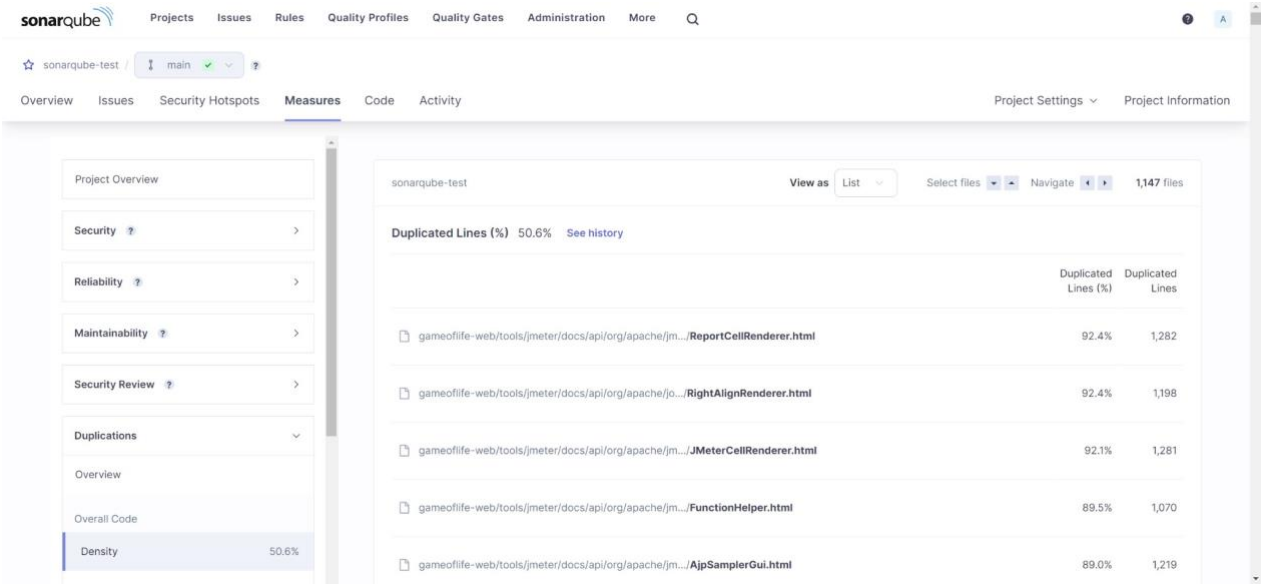
Bugs



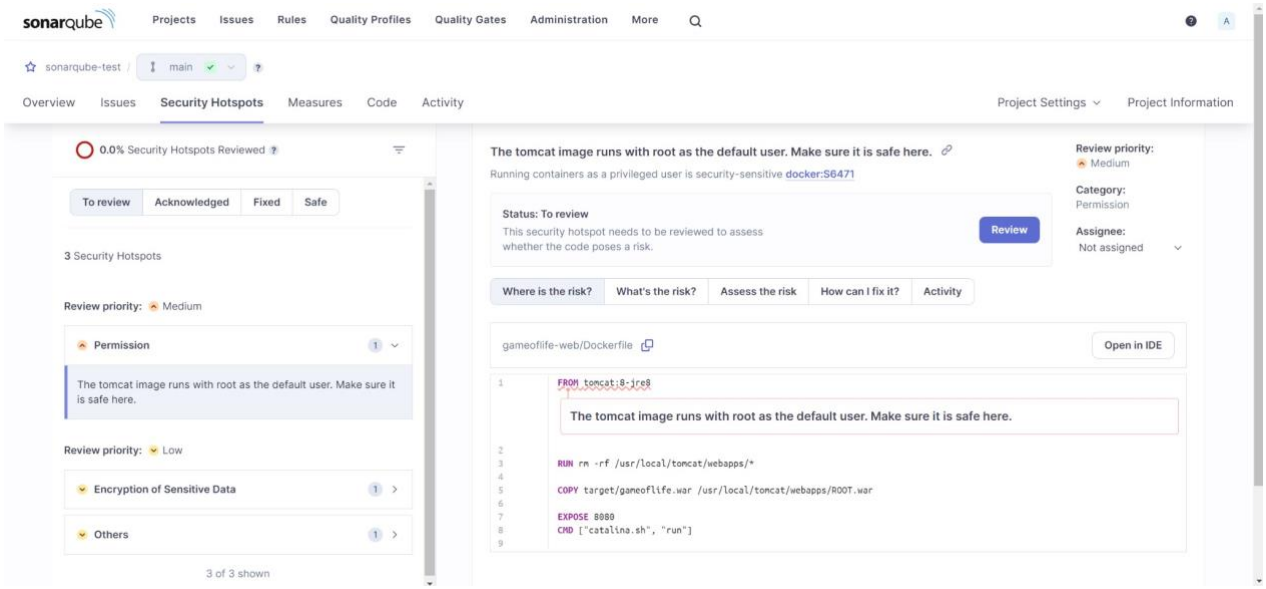
Reliability



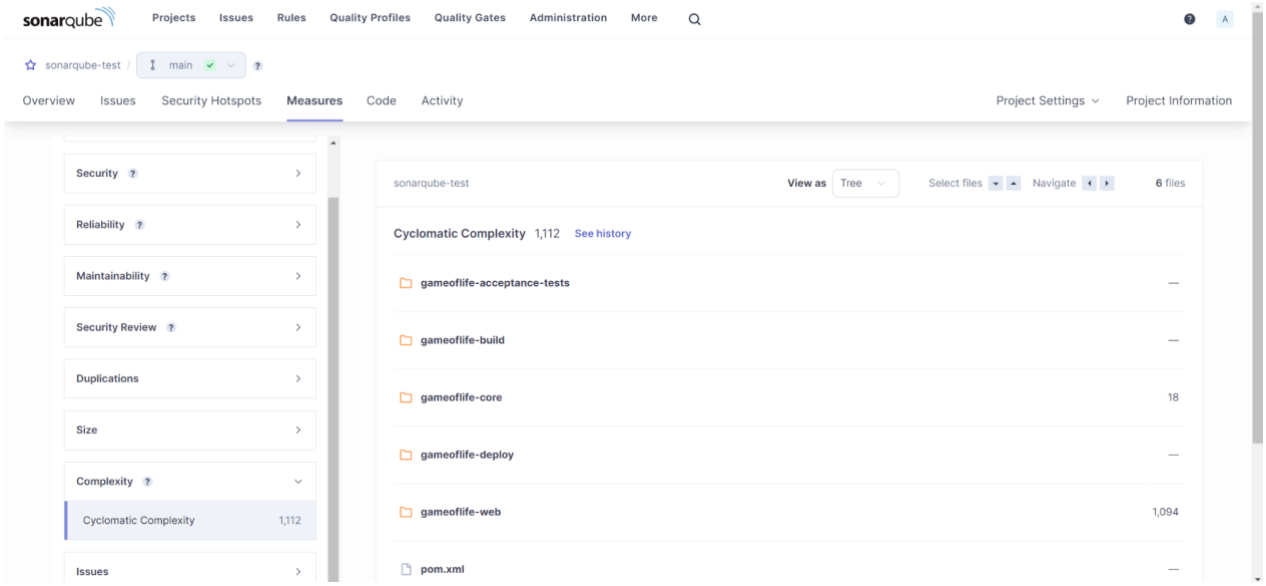
Duplicates



Security Hotspot



Cyclomatic Complexity



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.