Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

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) is a methodology that analyzes source code to identify security vulnerabilities before compilation. It is often referred to as white box testing and helps developers detect issues early in the software development lifecycle (SDLC).

Problems SAST Solves

1. **Early Detection**: Identifies vulnerabilities in the initial development stages, reducing later risks.
2. **Real-Time Feedback**: Provides immediate insights, allowing developers to fix issues before moving forward.
3. **Code Navigation**: Offers visual representations of vulnerabilities for easier code understanding.
4. **Guidance on Fixes**: Suggests specific remediation steps without requiring deep security expertise.
5. **Comprehensive Coverage**: Analyzes the entire codebase quickly, outperforming manual reviews.
6. **Regular Scanning**: Ensures continuous security assessment through scheduled scans during builds or releases.

Importance of SAST

- **Resource Efficiency**: Automates code reviews, addressing the resource gap between developers and security staff.
- **Speed**: Processes millions of lines of code in minutes, identifying critical vulnerabilities.
- **Proactive Security**: Integrates security into the development process, preventing vulnerabilities from being overlooked.

What is a CI/CD Pipeline?

A **CI/CD Pipeline** refers to Continuous Integration and Continuous Delivery, automating software development tasks. It includes stages such as coding, building, testing, and deploying, ensuring each step is completed sequentially for efficient releases.

What is SonarQube?

SonarQube is an open-source platform for continuous code quality inspection. It performs static code analysis to generate reports on bugs, vulnerabilities, and code duplications across various programming languages.

Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Benefits of SonarQube

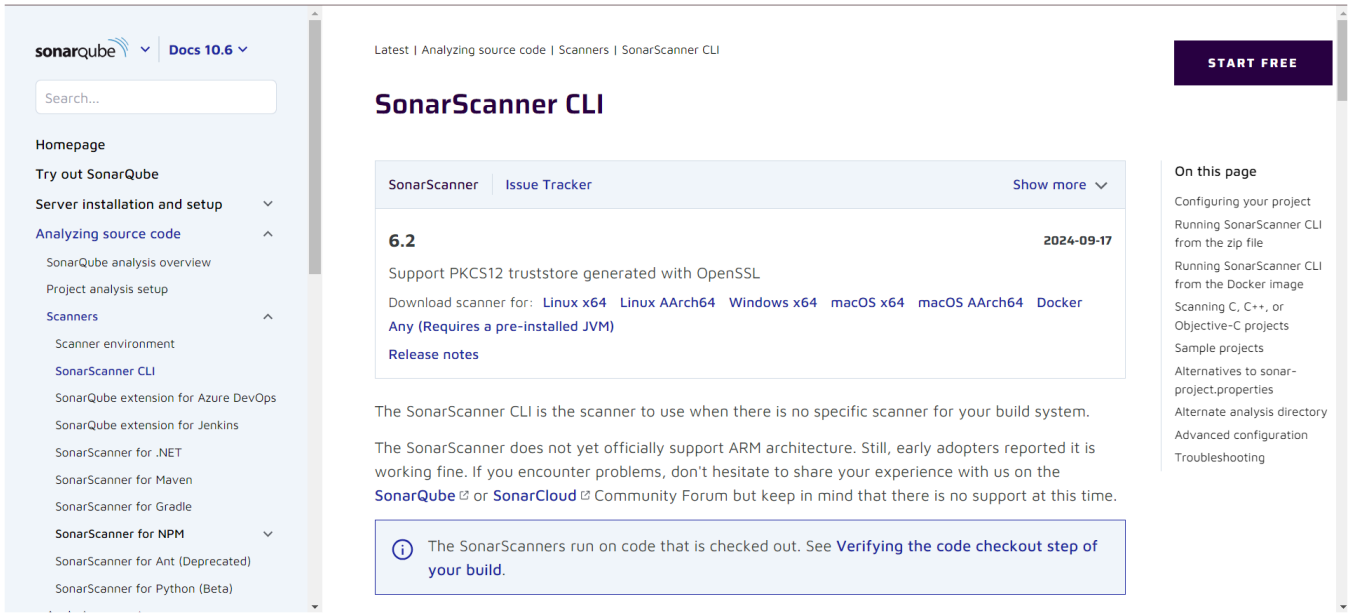
- **Sustainability**: Optimizes application lifecycle by reducing complexity and vulnerabilities.
- **Increased Productivity**: Minimizes maintenance efforts and costs.
- **Quality Control**: Integrates code quality checks into development.
- **Error Detection**: Alerts developers to fix issues before release.
- **Scalability**: Supports multiple projects without restrictions.
- **Skill Enhancement**: Provides regular feedback to improve developer skills.

Prerequisites:

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

Download The SonarQube CLI according to your system :

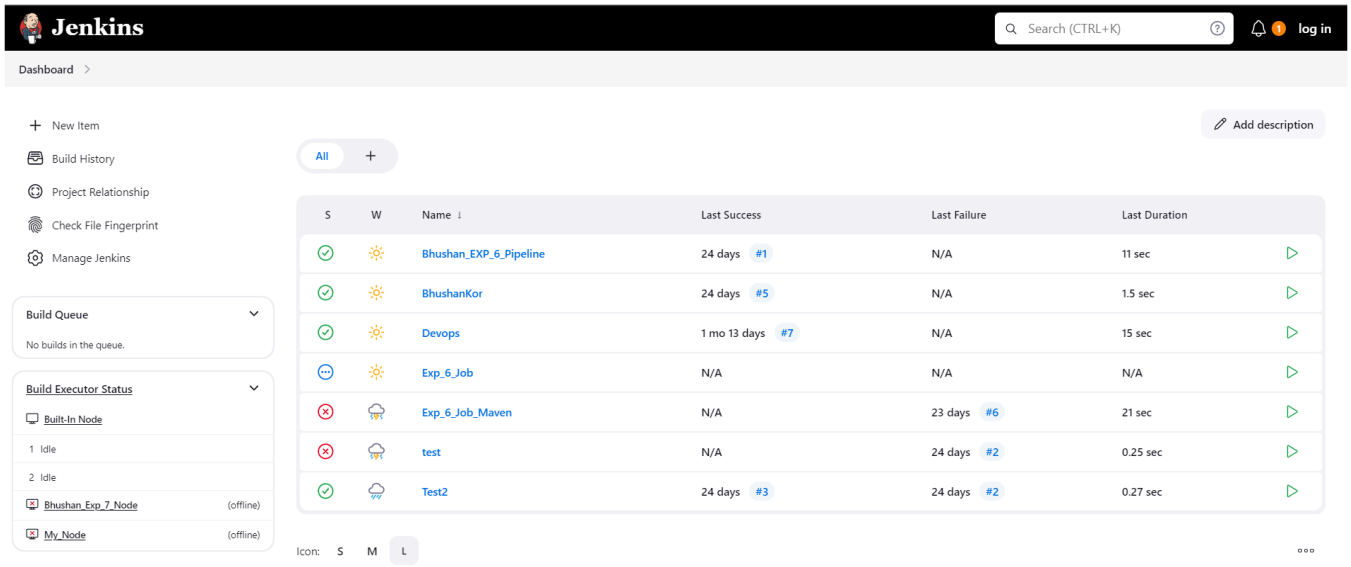
https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/



Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

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



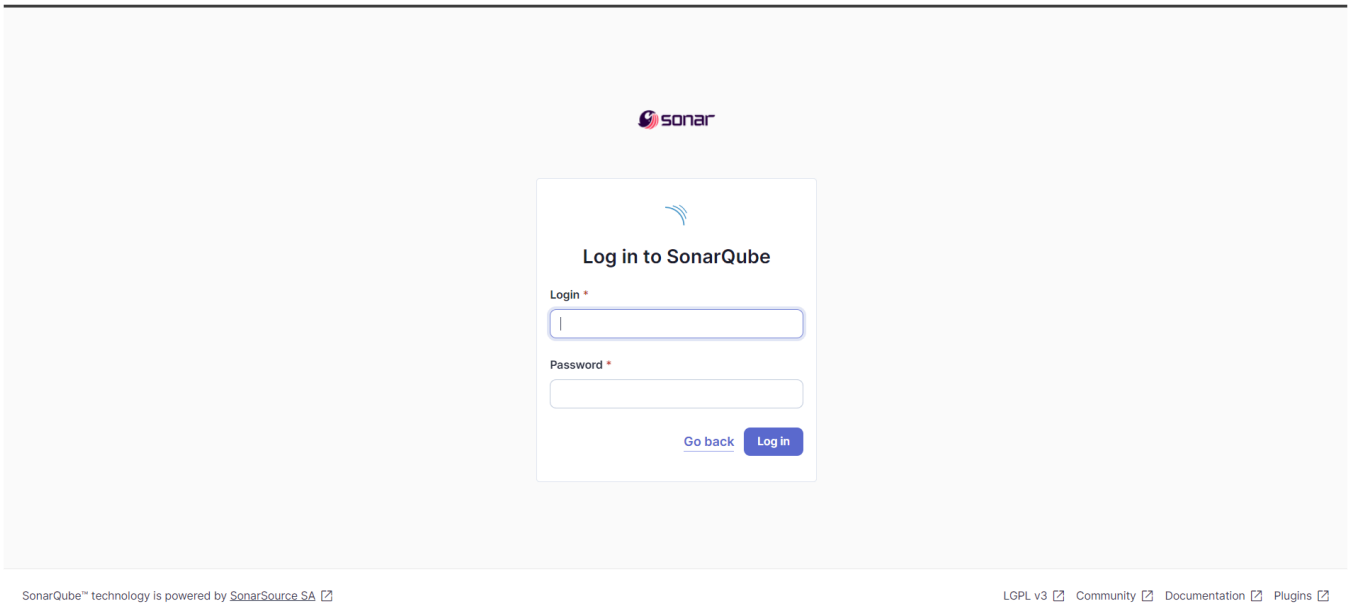
Step 2: Run SonarQube in a Docker container using this command
docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest

```
Microsoft Windows [Version 10.0.22621.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\INFT505-11>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:late
st
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
d72b183b1866cea7ecdb976a63dfe521172c307eb45eace7b769f726f0bbf989

C:\Users\INFT505-11>
```

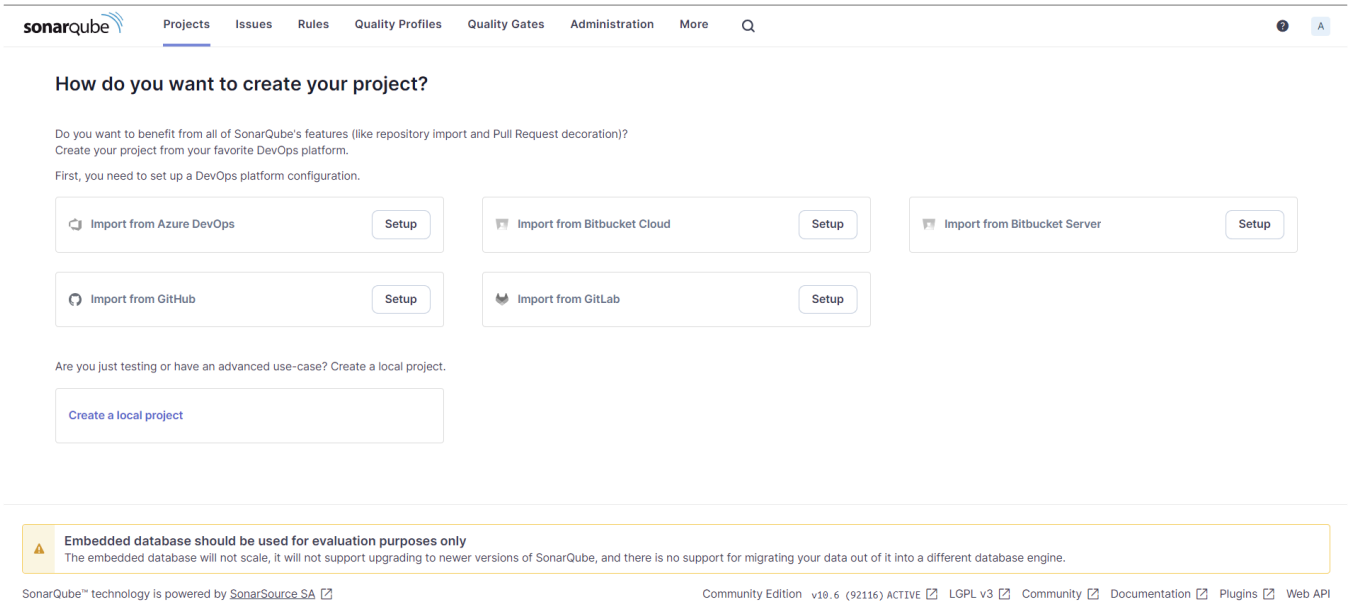
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

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



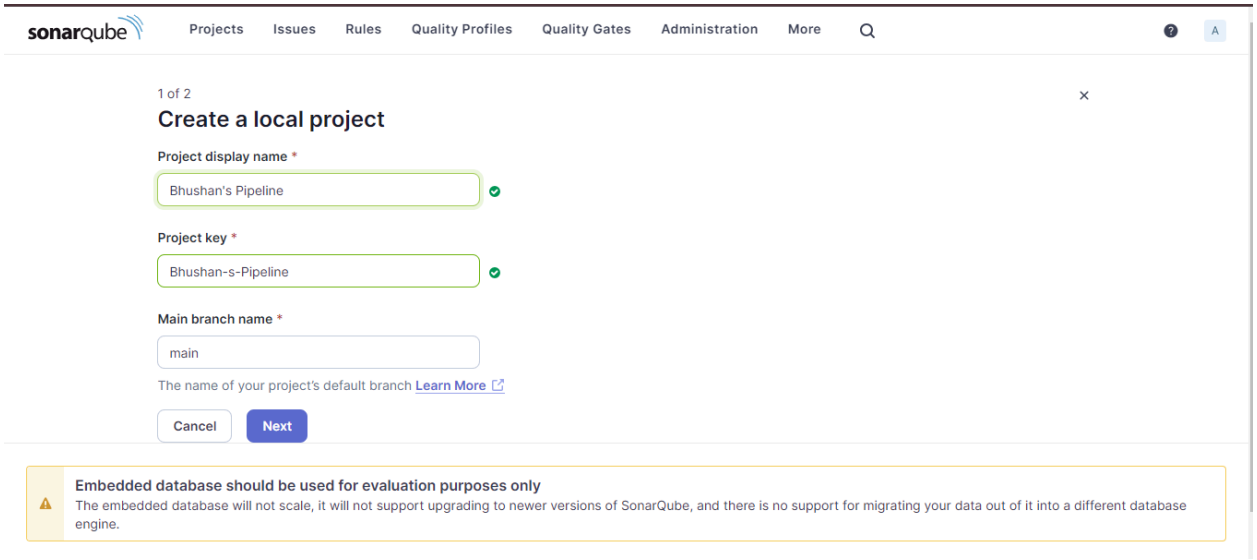
Start 4: Login to SonarQube using username admin and password admin.

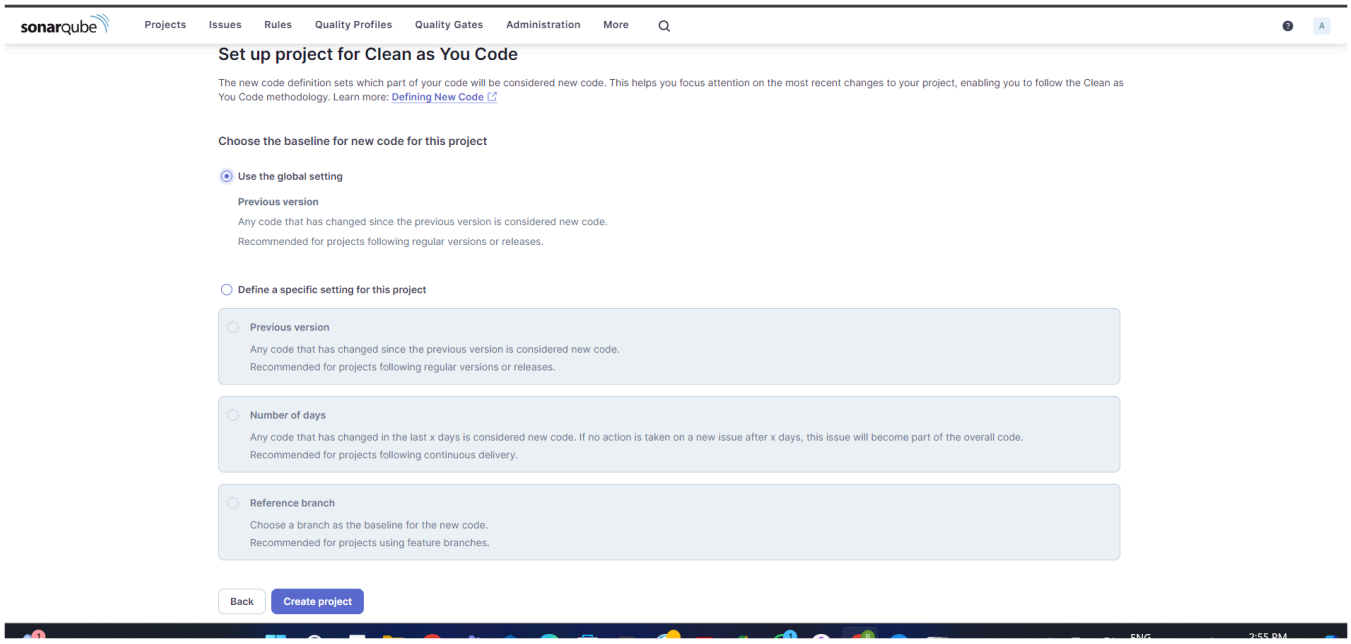


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Step 5: Create a manual project in SonarQube with any Name



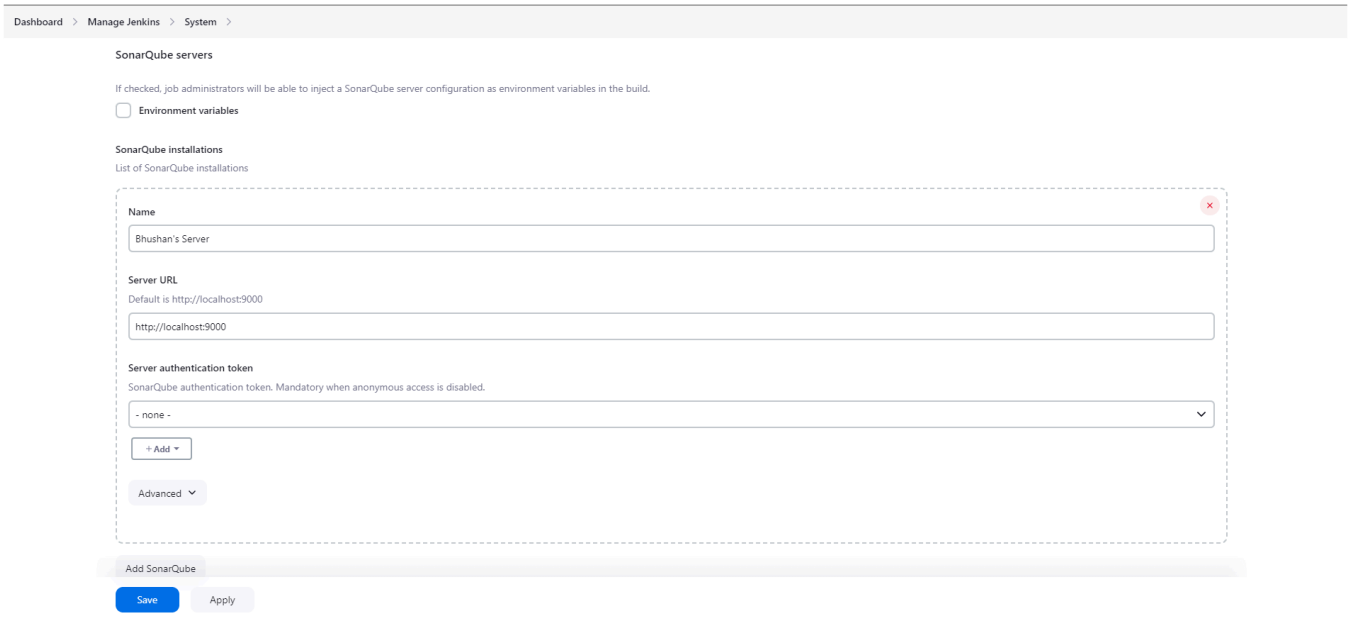


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

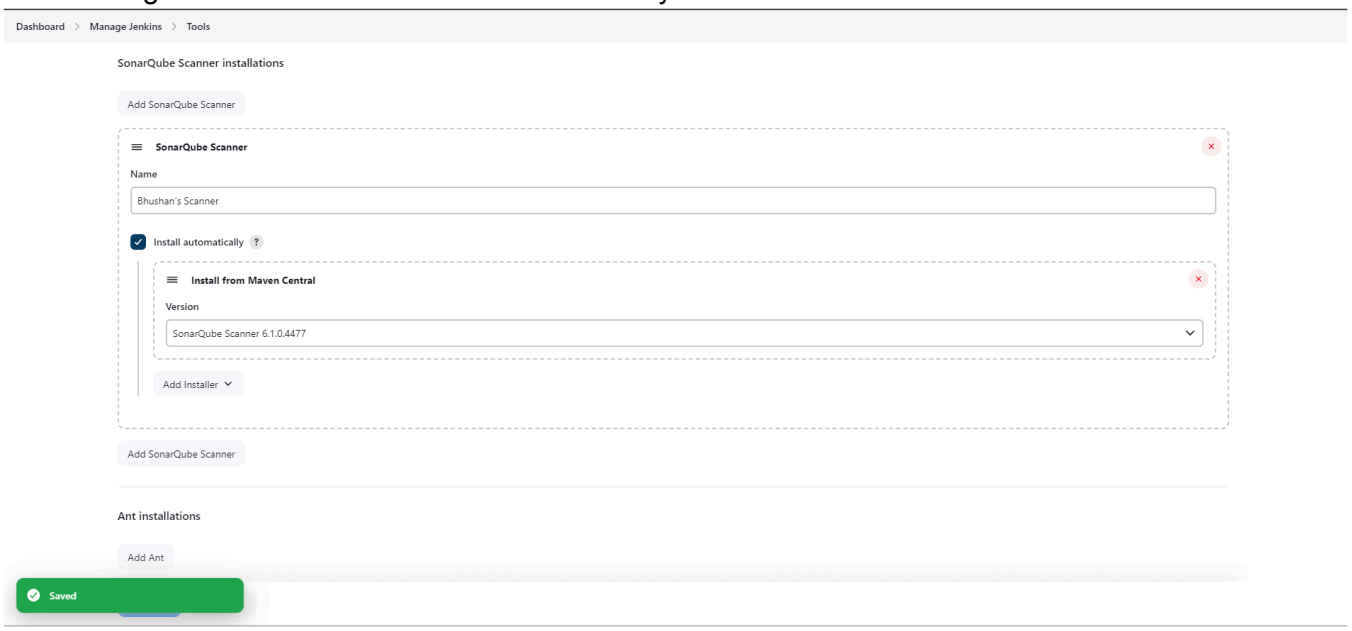
Division: D15C **Roll No: 28**

Step 6:Setup the project and come back to Jenkins Dashboard.
Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.

Step 7: Under Jenkins 'Configure System', look for SonarQube Servers and enter the details. Enter the Server Authentication token if needed.



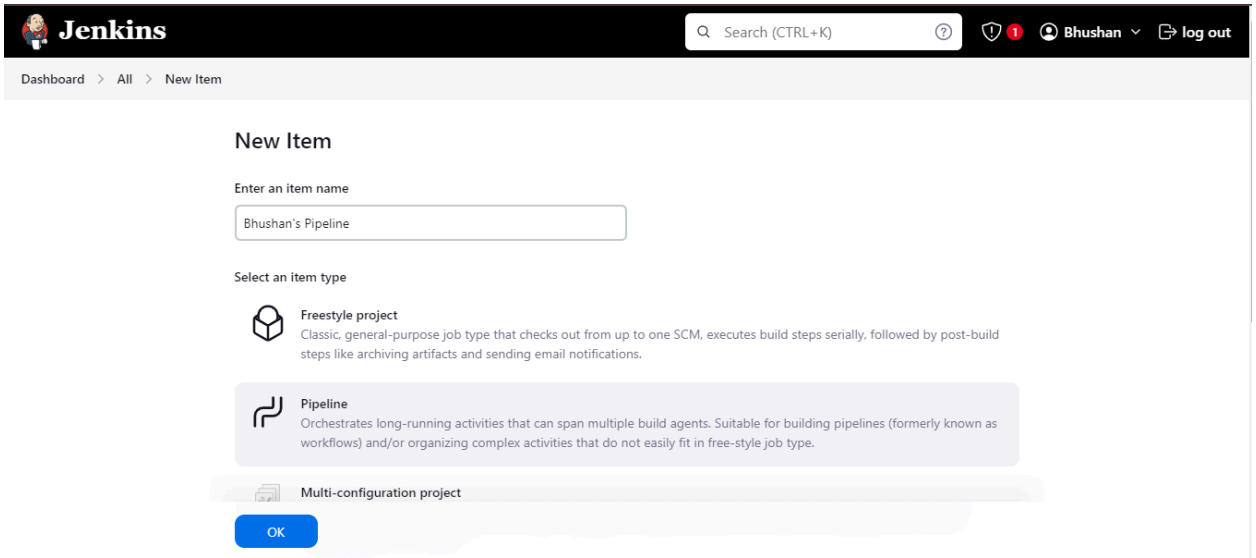
Step 8:Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically.



Name:Bhushan Mukund Kor **Academic Year:2024-2025**

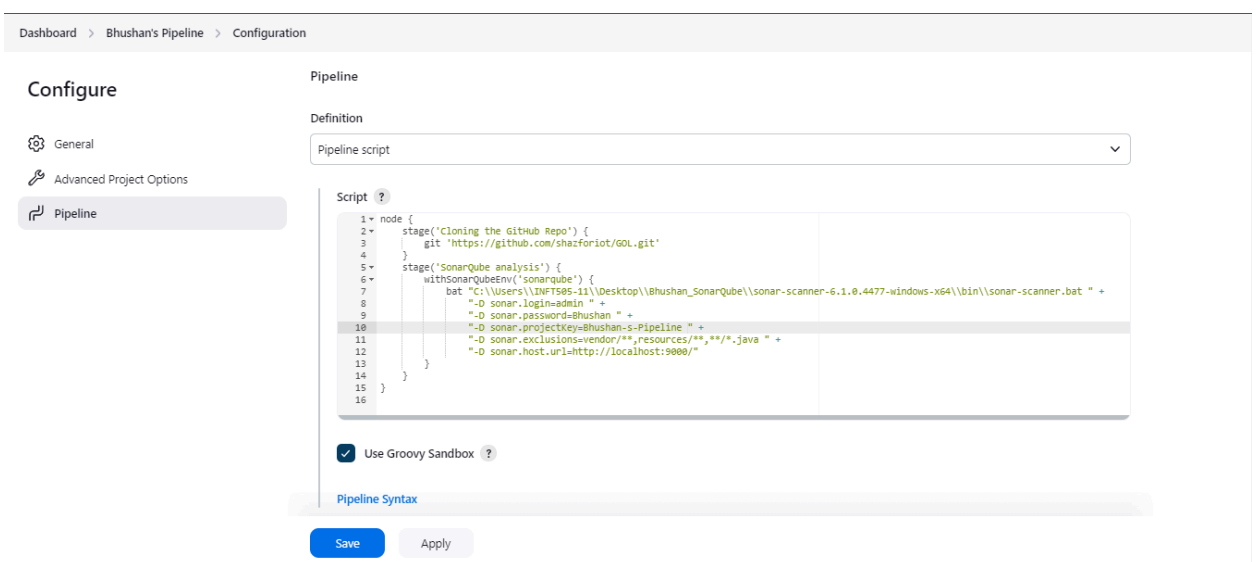
Division: D15C **Roll No: 28**

Step 9: Now click on the new item and select the pipeline project and give name.



Step 10: Under the scripts add the following script.**(Do not forget to replace details with your details.)**

```
node {
        stage('Cloning the GitHub Repo'){
                git 'https://github.com/shazforiot/GOL.git'
        }
        stage('SonarQube analysis') {
                withSonarQubeEnv('sonarqube') {
                        bat  ''<your bin file location of SonarQube CLI>"+
                                "-D sonar.login=<your user name>"+
                                "-D sonar.password=<your password>"+
                                "-D sonar.projectkey=<your projectkey>"+
                                "-D sonar.exclusions=vendor/**,resources/**,**/*.java "+
                                "-D sonar.host.url=http://localhost:9000/"+
                }
        }

}
```

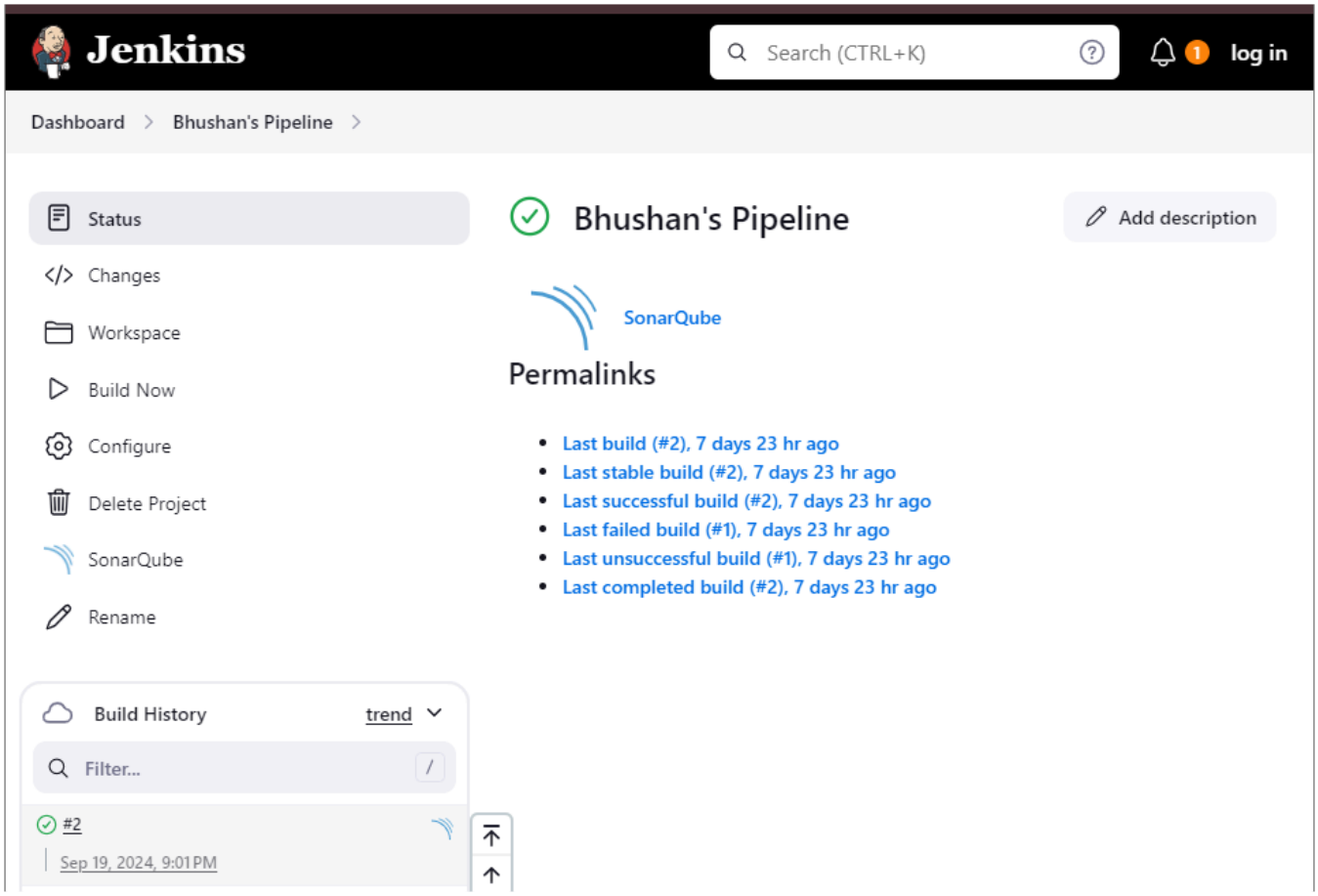


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

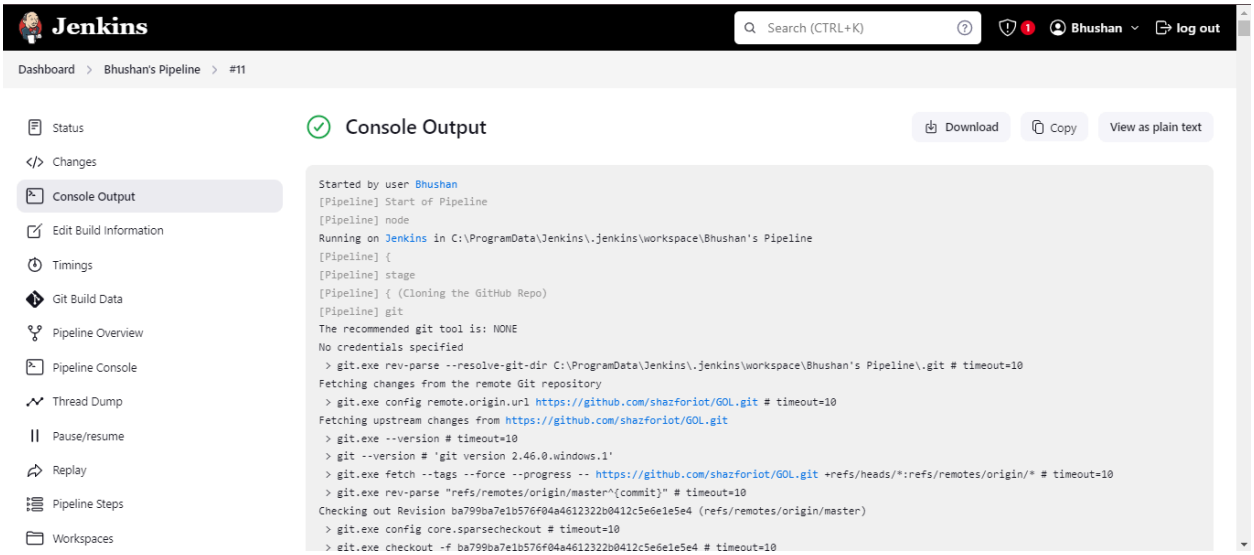
Division: D15C **Roll No: 28**

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

Step 11: Go back to jenkins. Go to the job you had just built and click on Build Now.



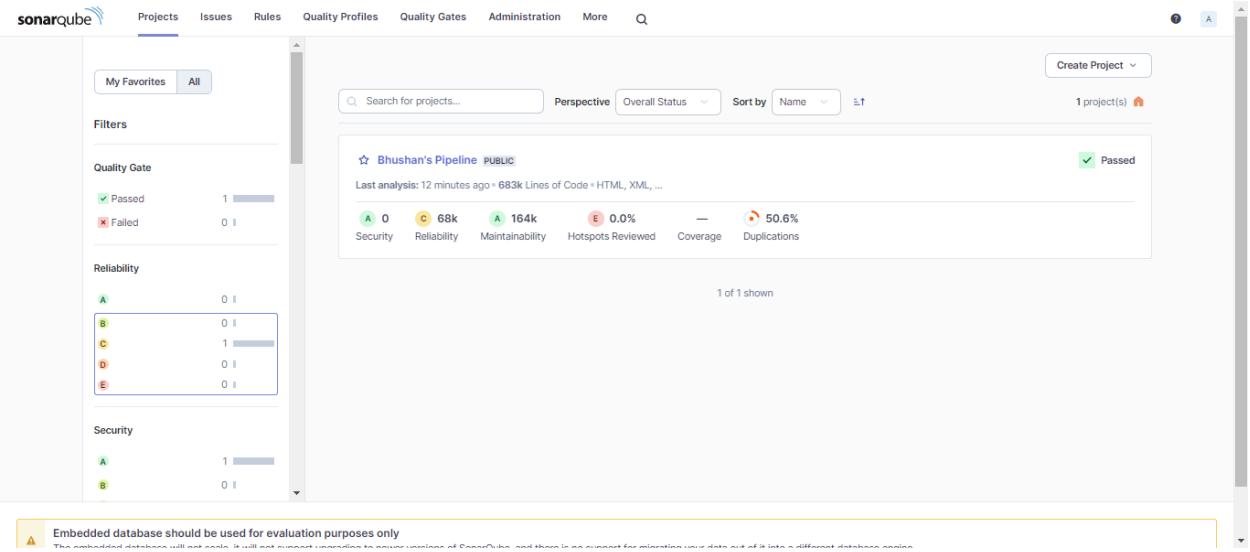
Step 12: Check the console output.

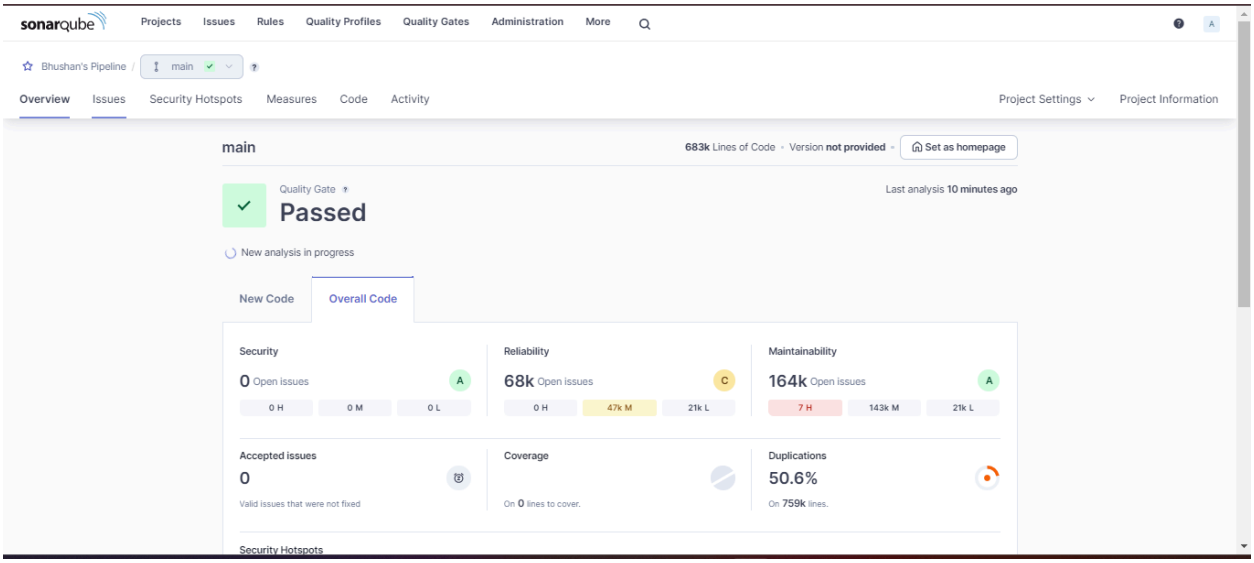


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Step 13: Once the build is complete, go back to SonarQube and check the project linked.



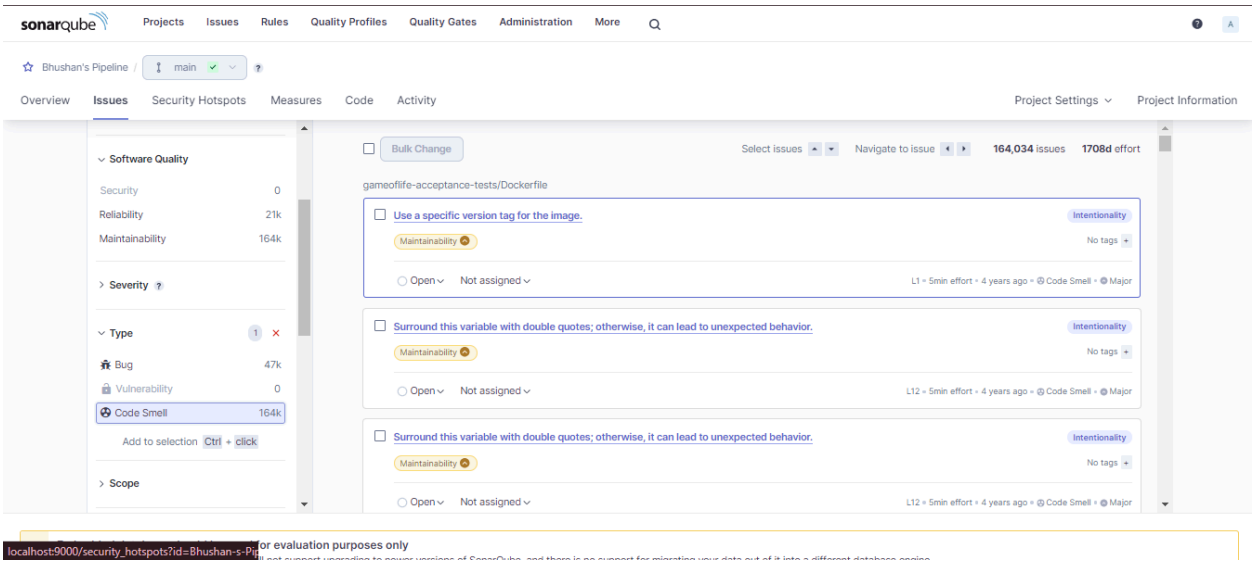


Name:Bhushan Mukund Kor **Academic Year:2024-2025**

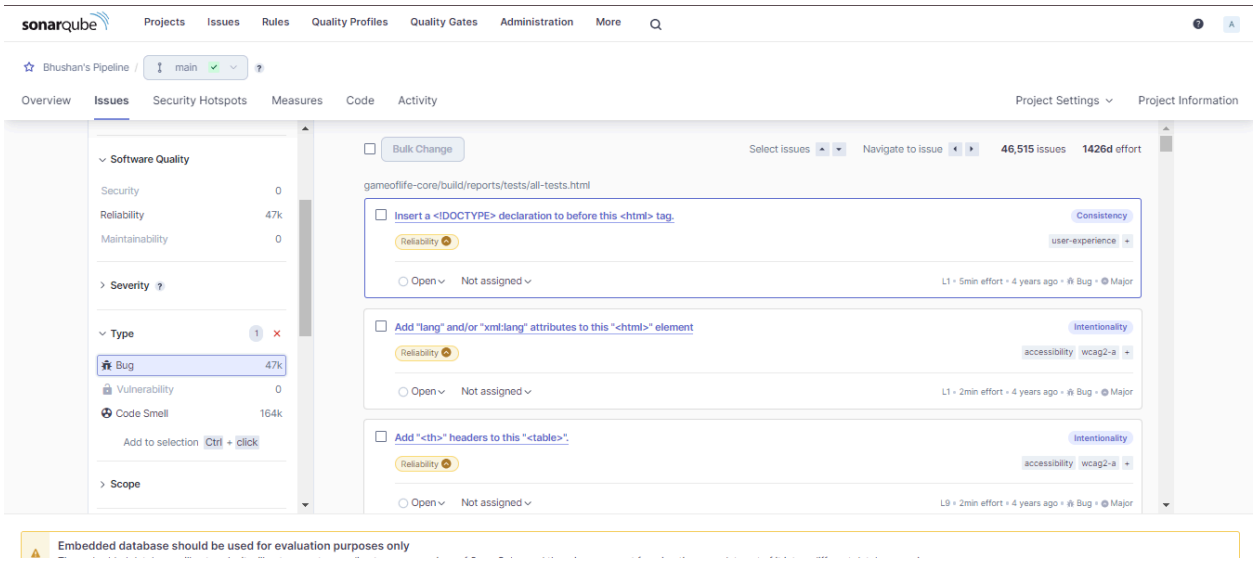
Division: D15C **Roll No: 28**

Click on Issues and see the different Issues.

Codesmell:



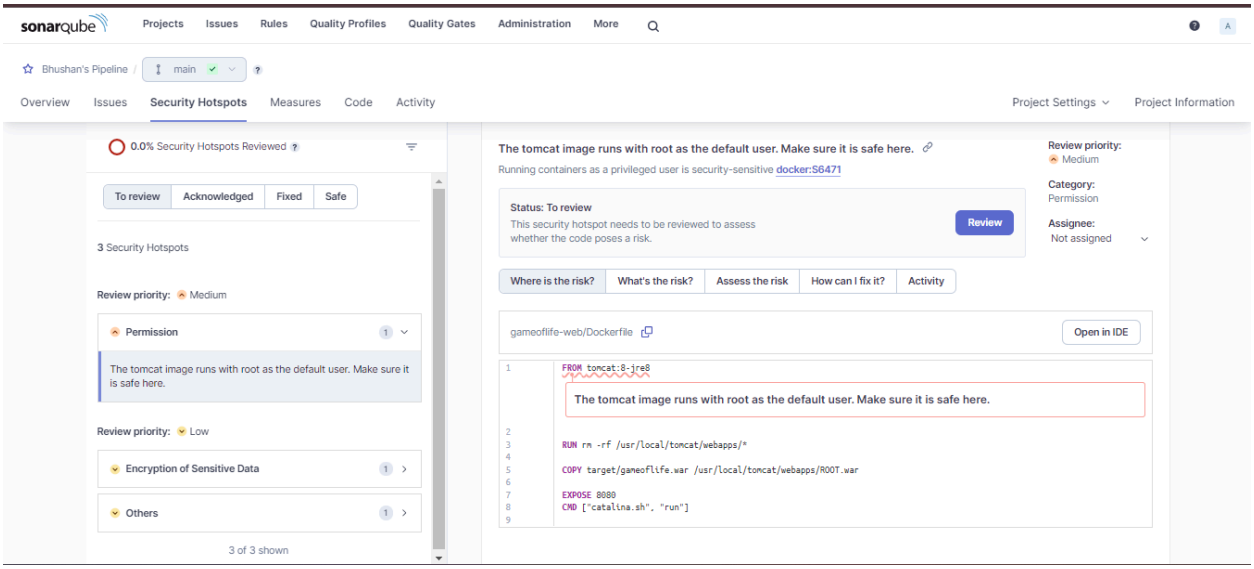
Bugs:



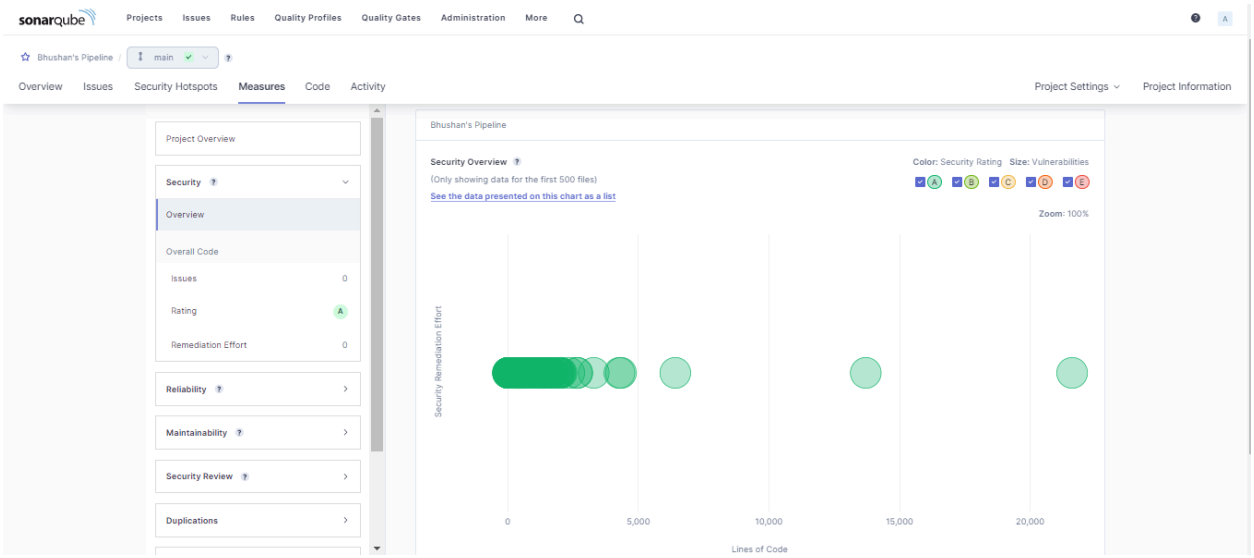
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

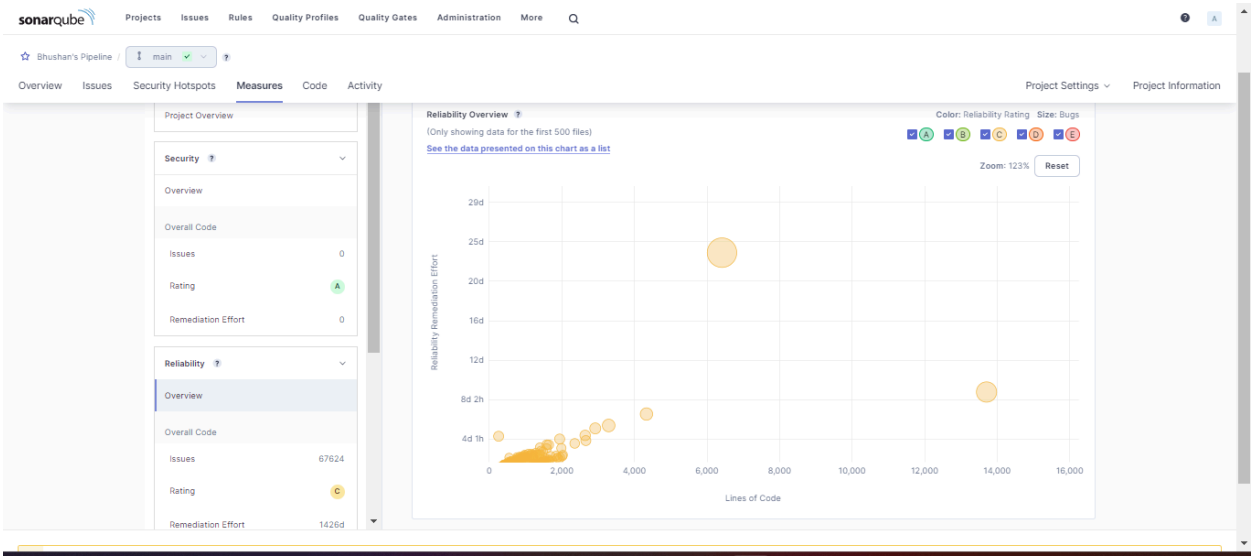
Click on Security hotspots and see the different Security hotspots.



Click on Measures and see the Measures in the form of Graphs.
Security:



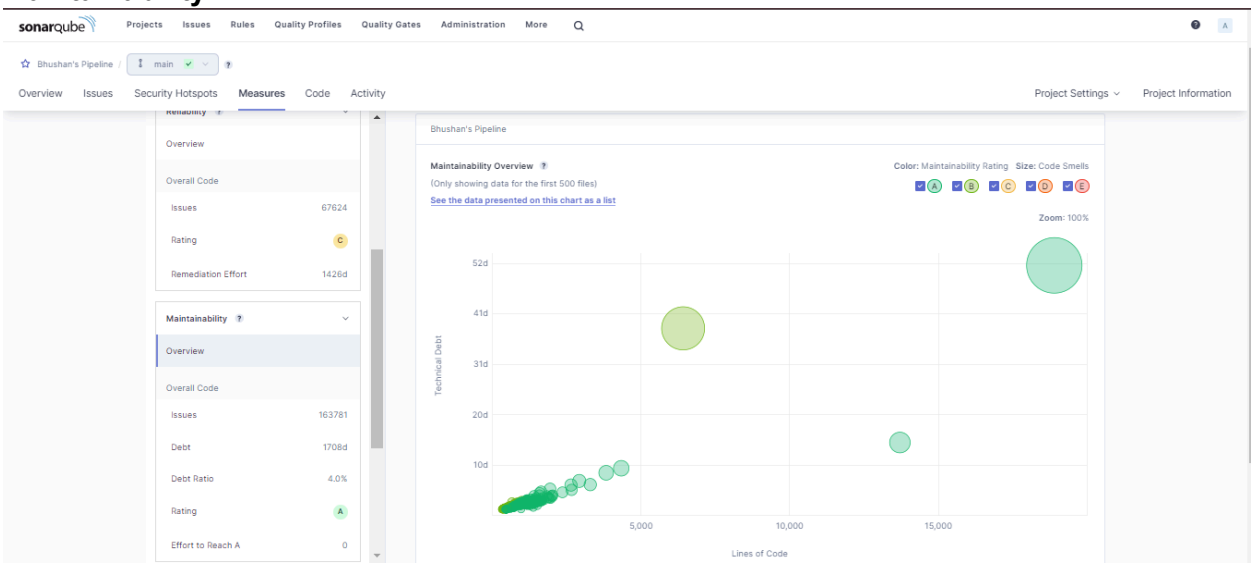
Reliability:



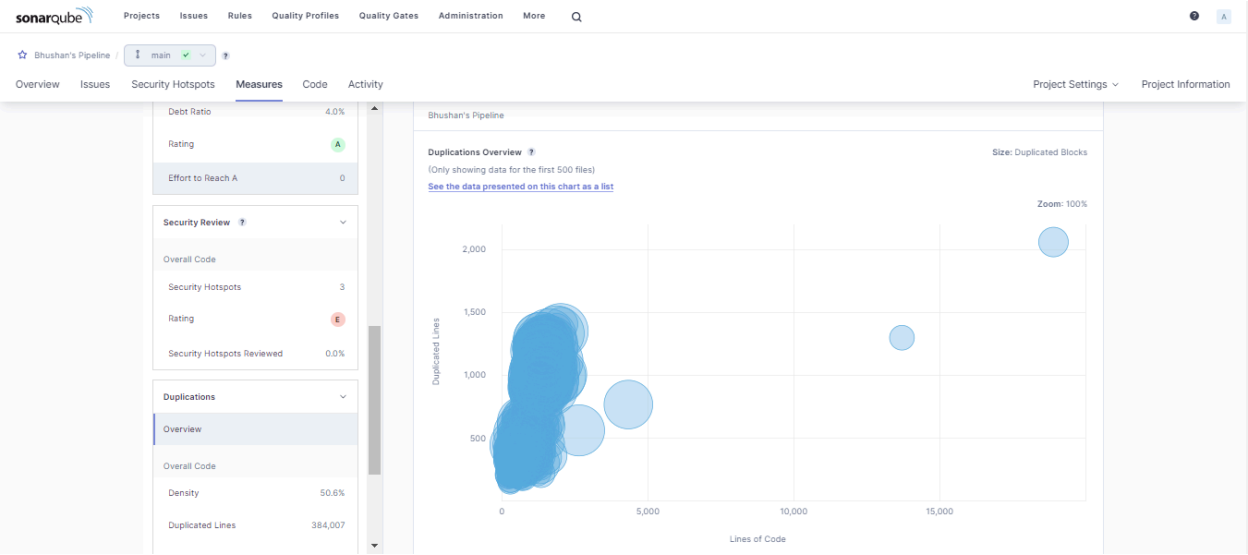
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

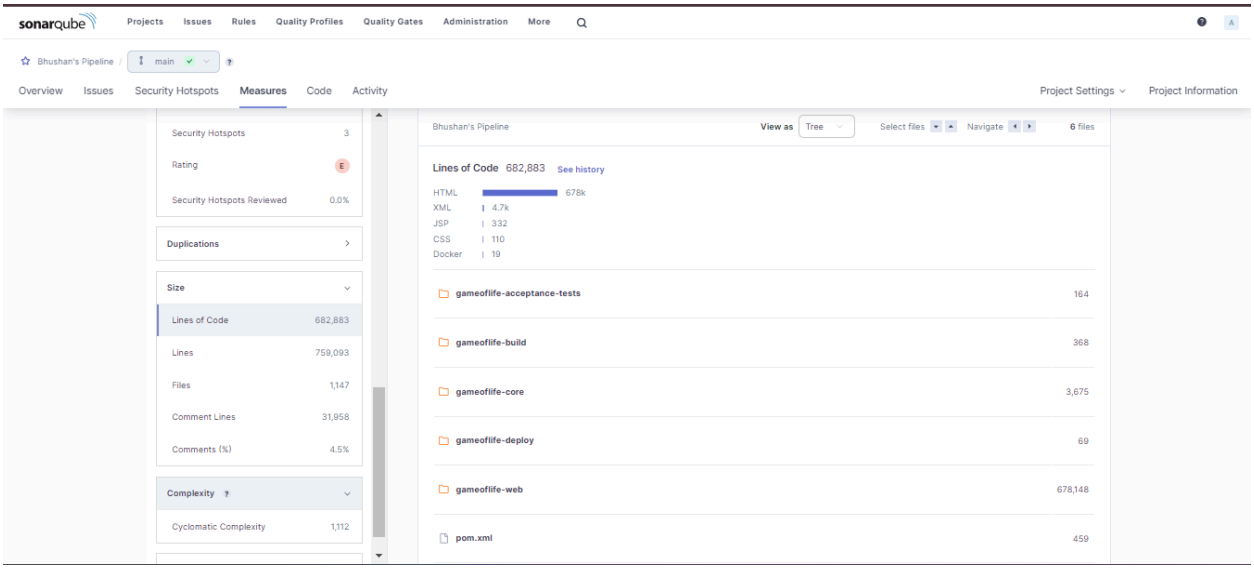
Maintainability:



Duplication:



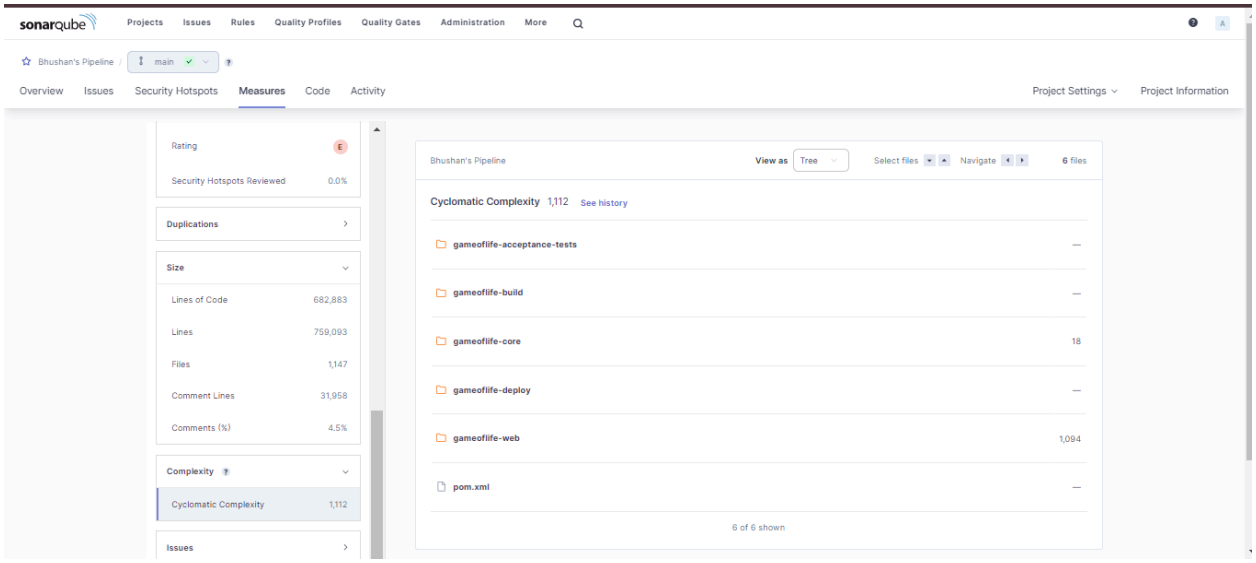
Sizes:



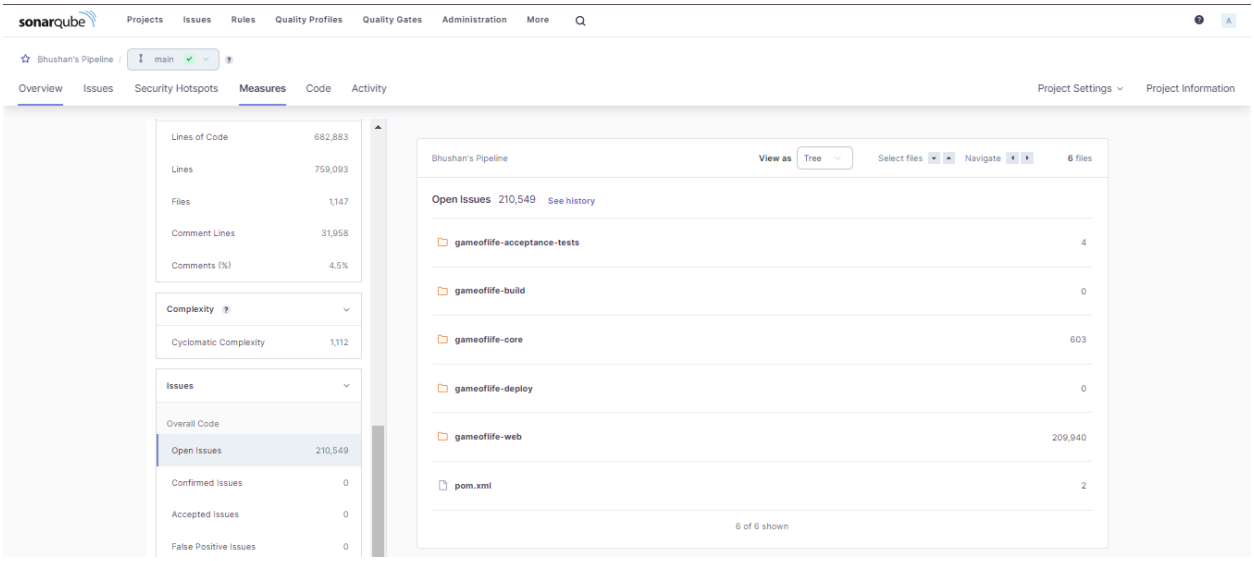
Name:Bhushan Mukund Kor **Academic Year:2024-2025**

Division: D15C **Roll No: 28**

Complexity:



Issues:



Conclusion: In this experiment, we have learned how to perform static analysis of a code using Jenkins CI/CD Pipeline with SonarQune analysis. A pipeline project is to be created which is given a pipeline script. This script contains all the information needed for the project to run the SonarQube analysis. After the necessary configurations are made on Jenkins, the Jenkins project is built. The code provided in this experiment contains lots of errors, bugs, duplications which can be checked on the SonarQube project linked with this build.