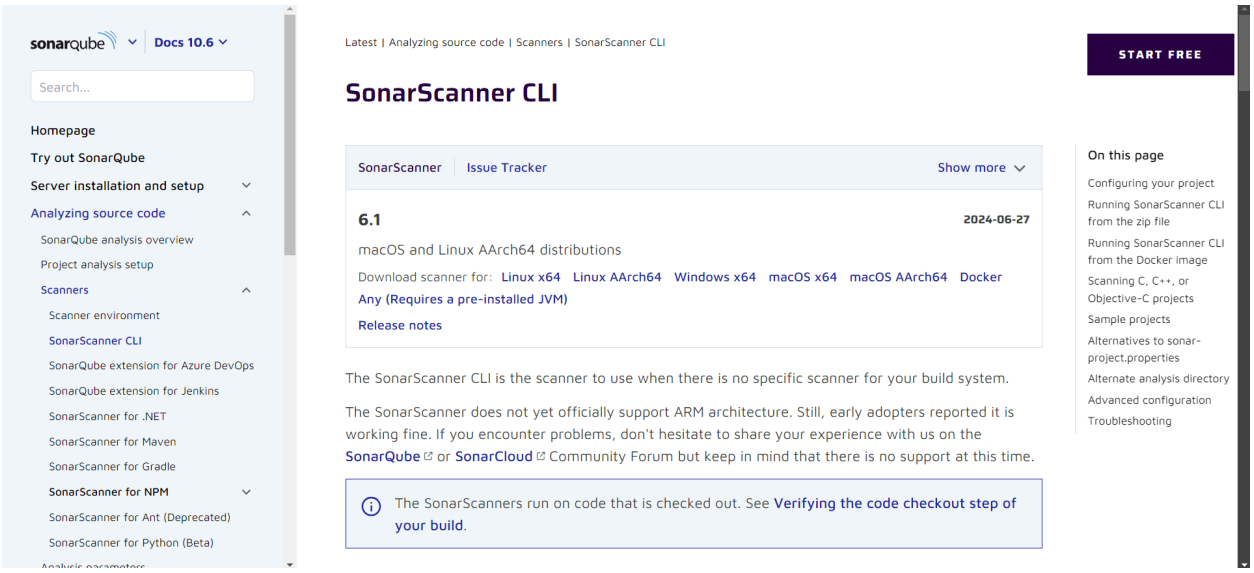
Name: Shivam.R.Prajapati **Div: D15C** **Roll No: 41** **Academic Year: 2024-25**

Experiment No: 8

AIM: Create a Jenkins CI/CD 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.

PREREQUISITES:

Step 1: Download sonar scanner https://docs.sonarsource.com/sonarqube/latest/analyzing-source-code/scanners/sonarscanner/ . Visit this link and download the sonarqube scanner CLI



Extract the downloaded zip file in a folder.

Step 2: Docker Run **docker -v command .**If docker is not installed so install it

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

Step 3: Install sonarqube image Command: **docker pull sonarqube**

```
C:\Users\Student>docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
762bedf4b1b7: Pull complete
95f9bd9906fa: Pull complete
a32d681e6b99: Pull complete
aabdd0a18314: Pull complete
5161e45ecd8d: Pull complete
aeb0020dfa06: Pull complete
01548d361aea: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:bb444c58c1e04d8a147a3bb12af941c57e0100a5b21d10e599384d59b
Status: Downloaded newer image for sonarqube:latest
docker.io/library/sonarqube:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker

C:\Users\Student>
C:\Users\Student>
```

Step 4: Keep **Jenkins** installed on your system.

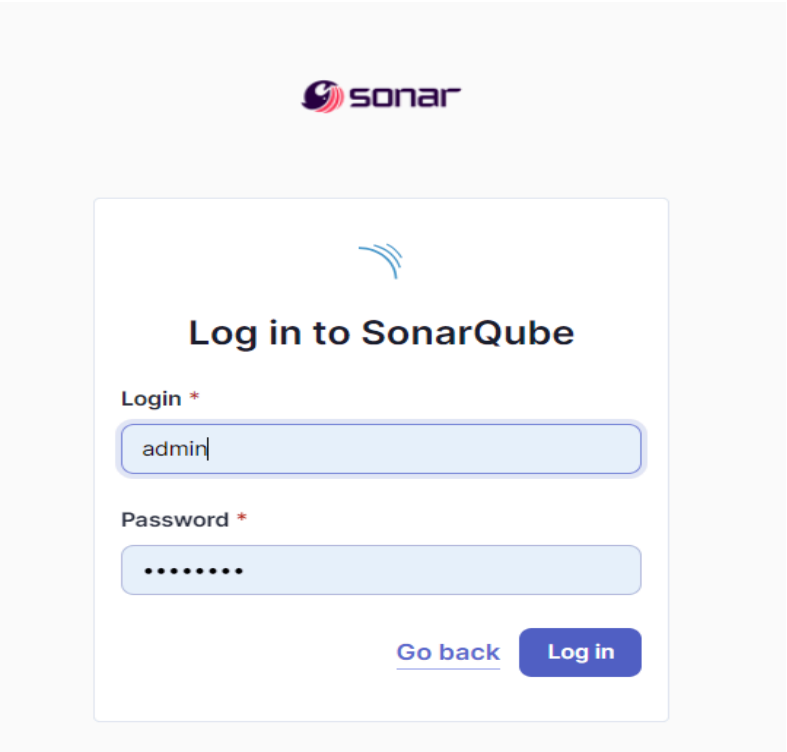
<u>EXPERIMENT STEPS:</u>

Step1: Run SonarQube image **docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest .**This command will run the SonarQube image that was just installed using docker.

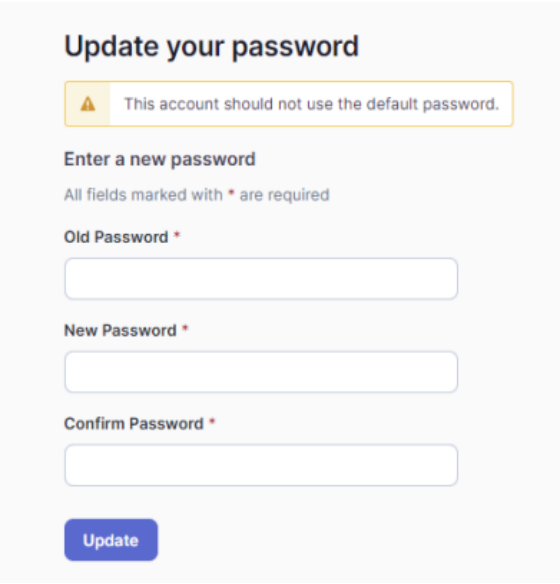
```
C:\Users\Student>docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
83330c33cd961d8d659f362c5f62c6cd1ff87f31ec99da134350b9b419370561

C:\Users\Student>
```

Step 2: Once the SonarQube image is started, you can go to **http://localhost:9000** to find the SonarQube that has started

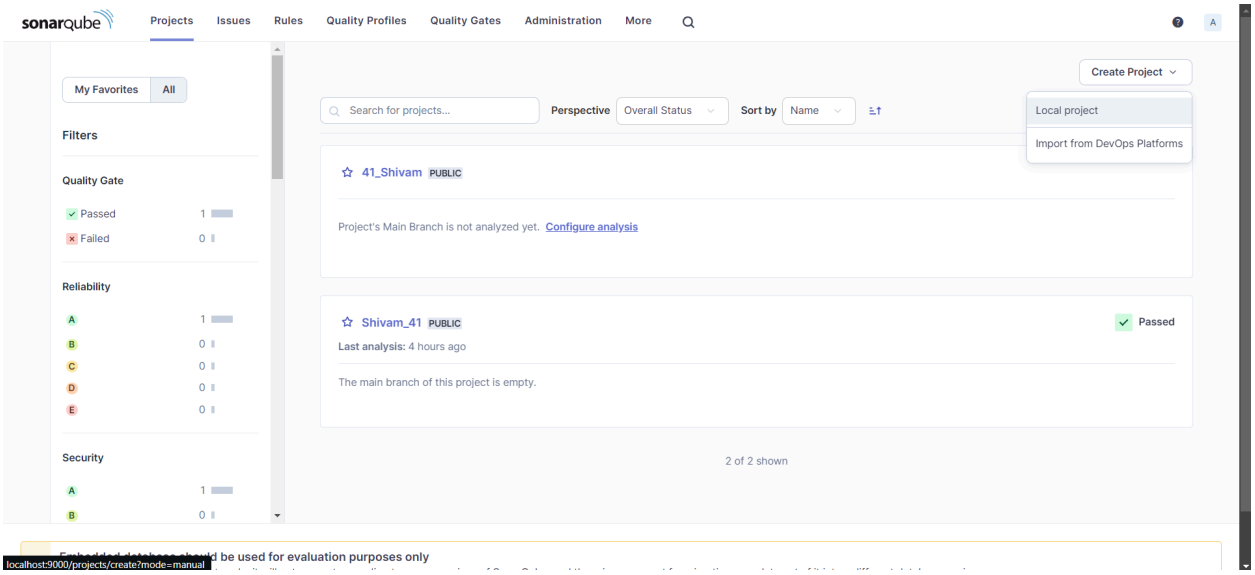


Step 3: On this interface, login with **username = 'admin'** and **password = 'admin**'. Once logged in successfully, SonarQube will ask you to reset this password. Reset it and remember this password.

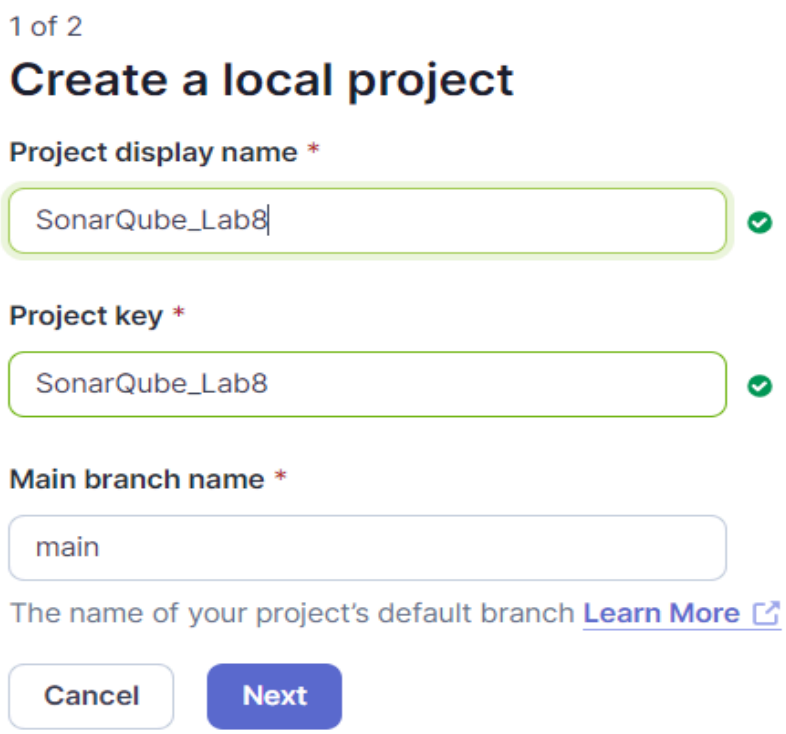


Step 4: After changing the password, you will be directed to this screen. Click on **Create a Local Project**. Give the project a display name and project key

Name: Shivam.R.Prajapati Div: D15C Roll No: 41 Academic Year: 2024-25

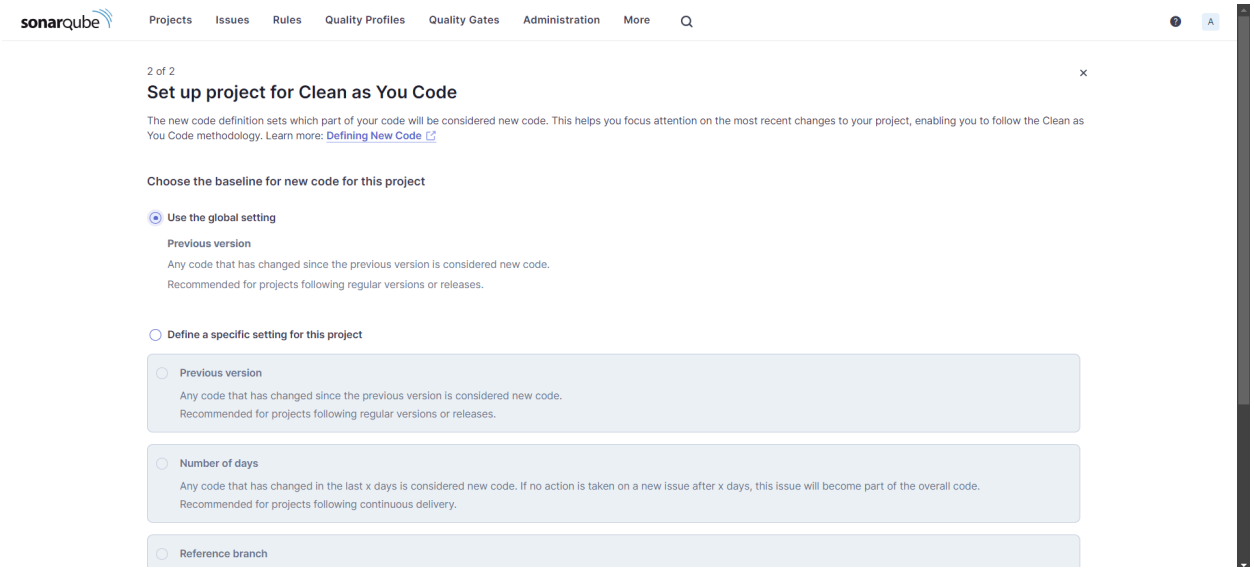


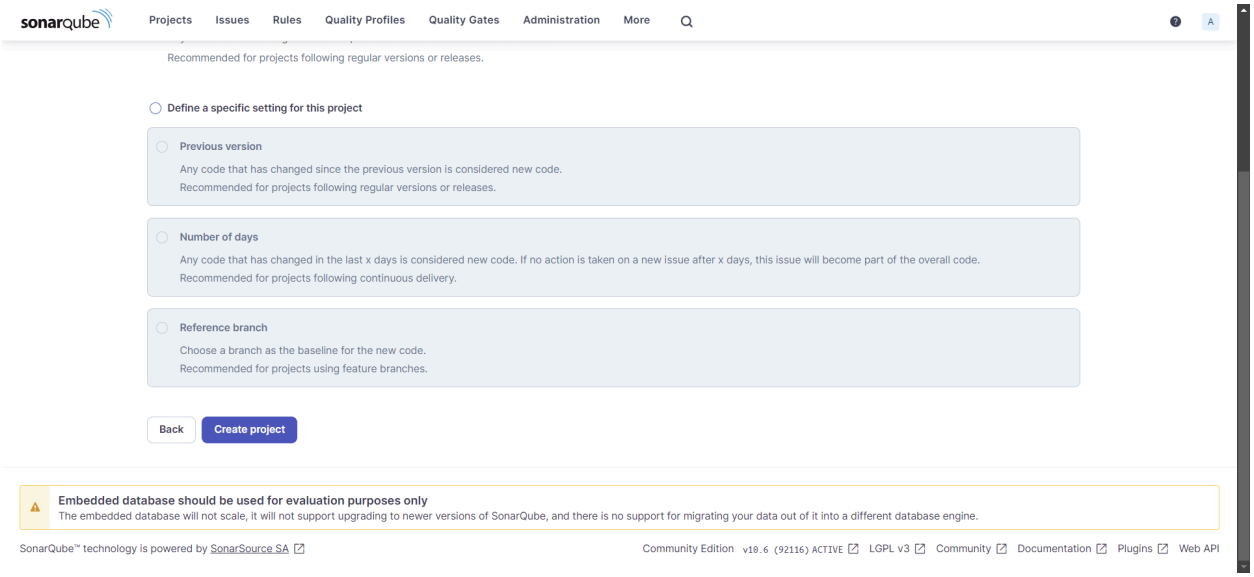
Click on Create Project



Set up the project as required and click on create.

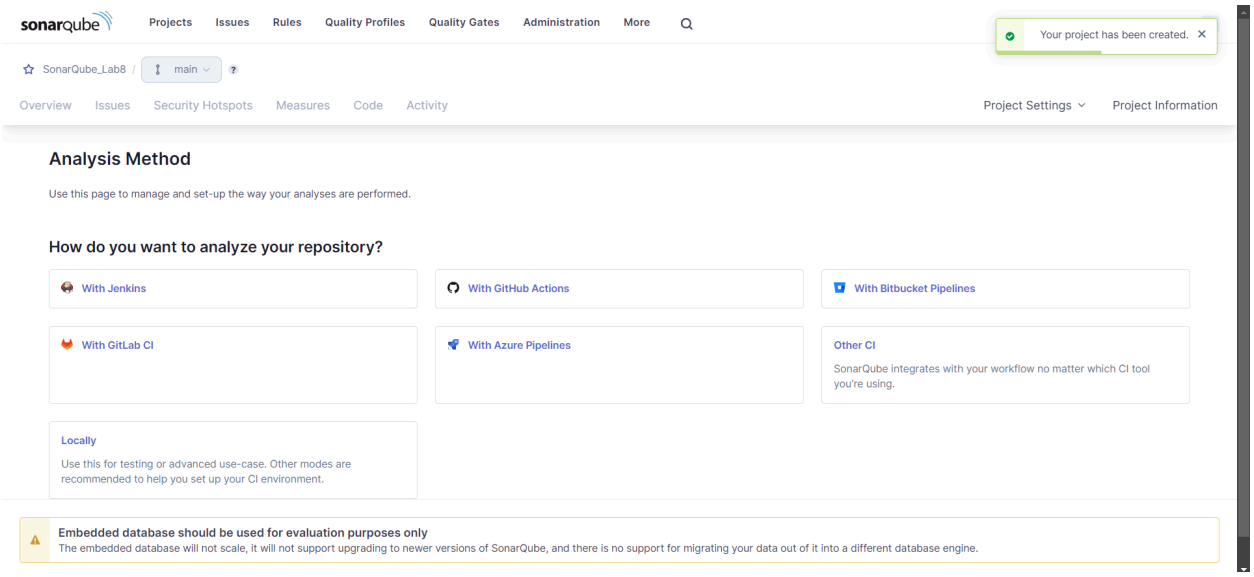
In the Step 2 while creating the project,Sonarqube ask you regarding which code should be considered as the new code for examining it .





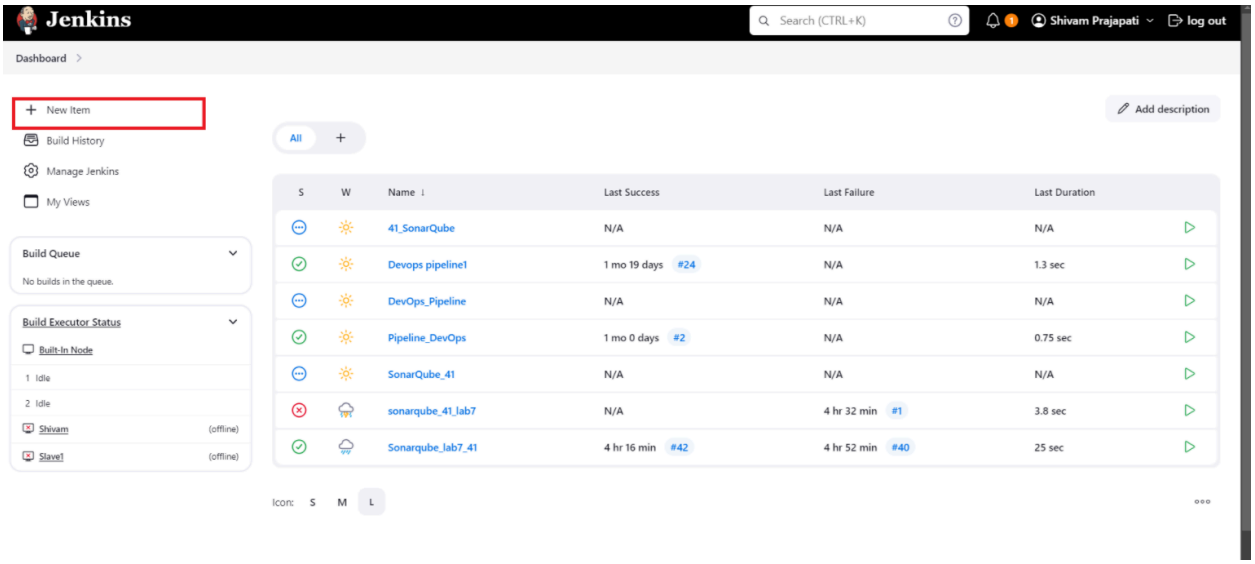
Click on Create

Name: Shivam.R.Prajapati Div: D15C Roll No: 41 Academic Year: 2024-25

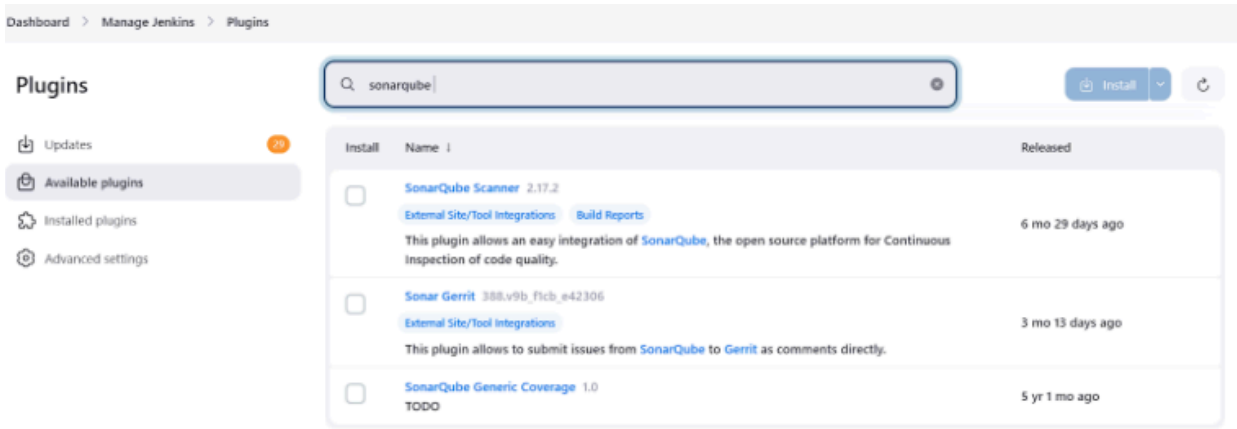


Project is created

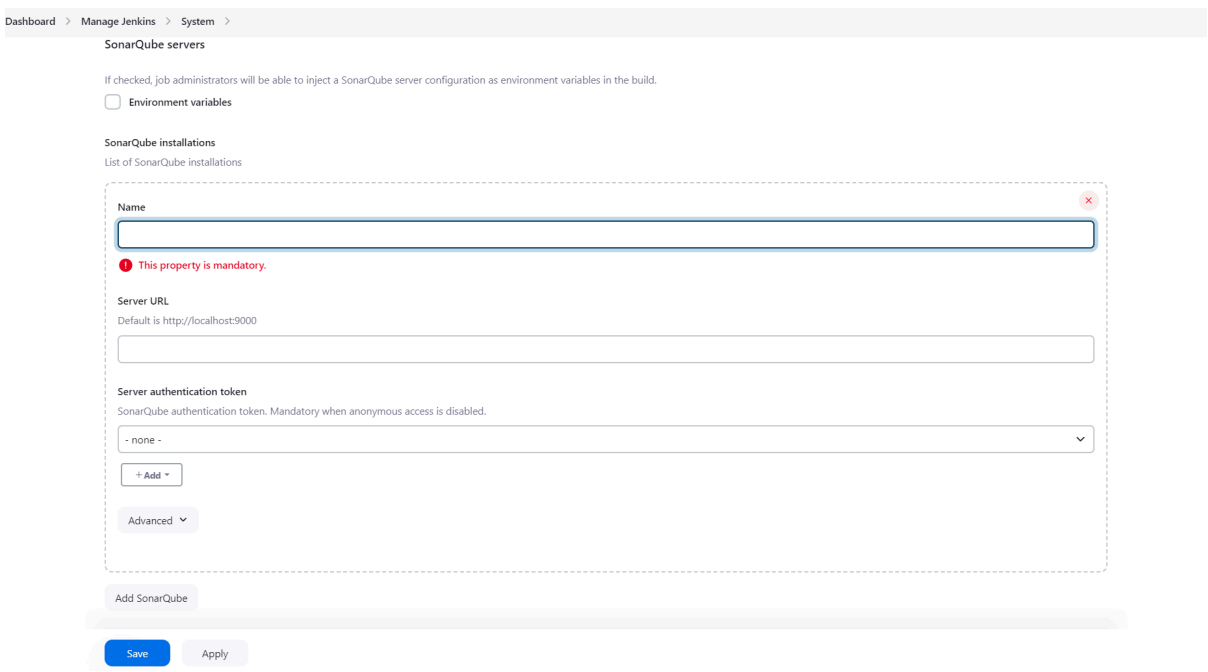
Step 5: Open **Jenkins** on whichever port it is installed. (http://loaclhost:). Go to the new item



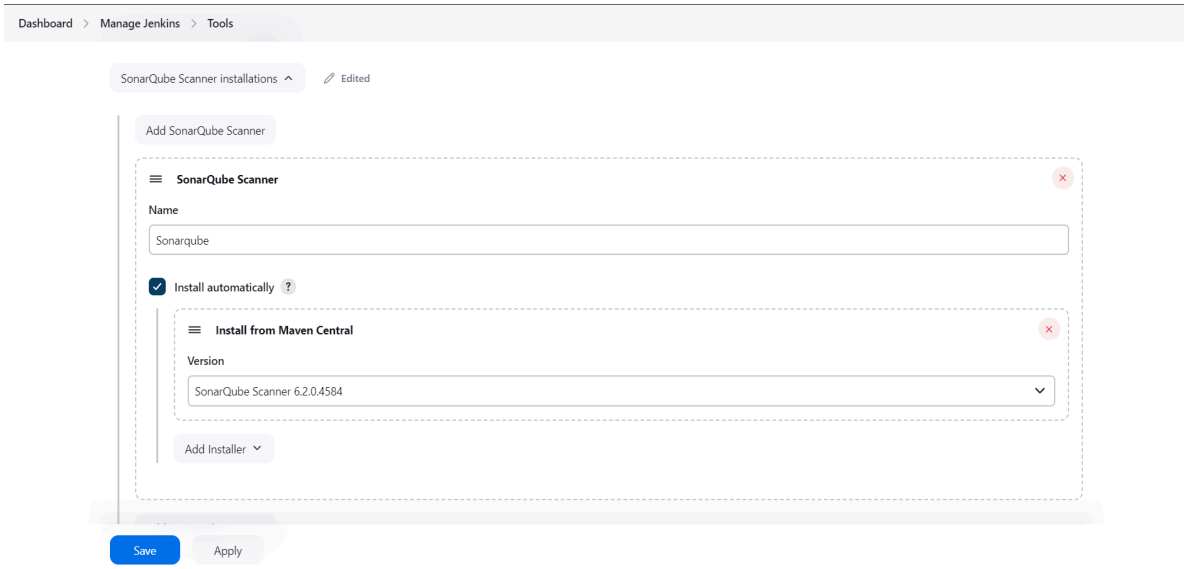
Step 6: Go to manage jenkins →available plugins then Search for **Sonarqube Scanner** for Jenkins and install it



Step 7: Now, go to Manage Jenkins → System. Under Sonarqube servers, add a server. Add server authentication token if needed.

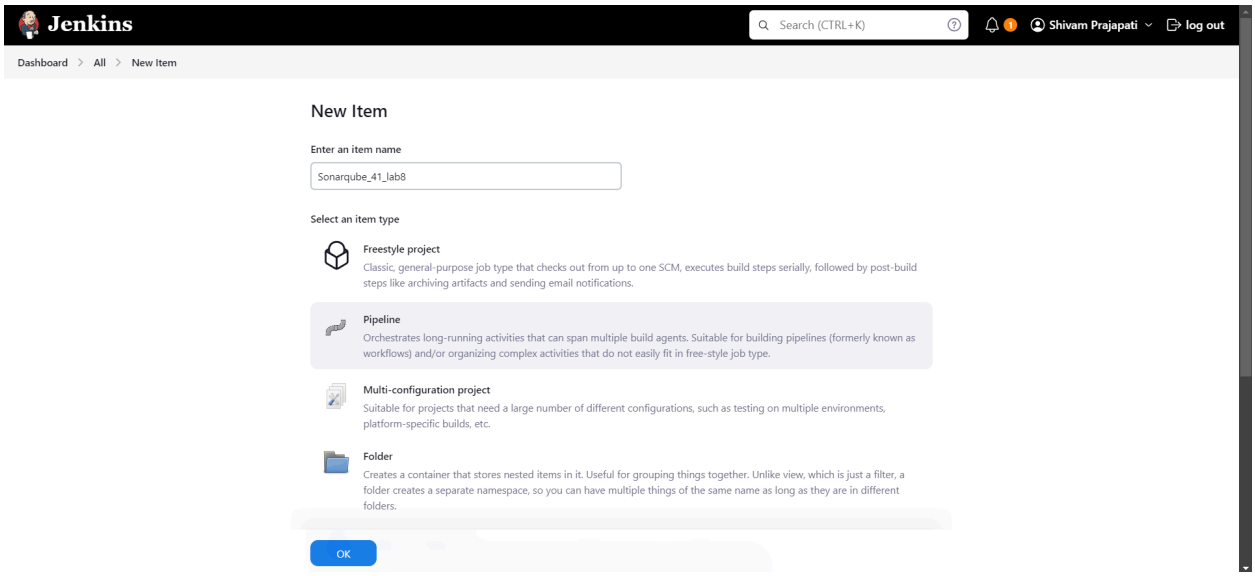


Step 8: Go to Manage Jenkins → Tools. Go to SonarQube scanner, choose the latest configuration and choose to install automatically.



Step 9: After configuring, click on **New Item** and select **Pipeline Project**

.

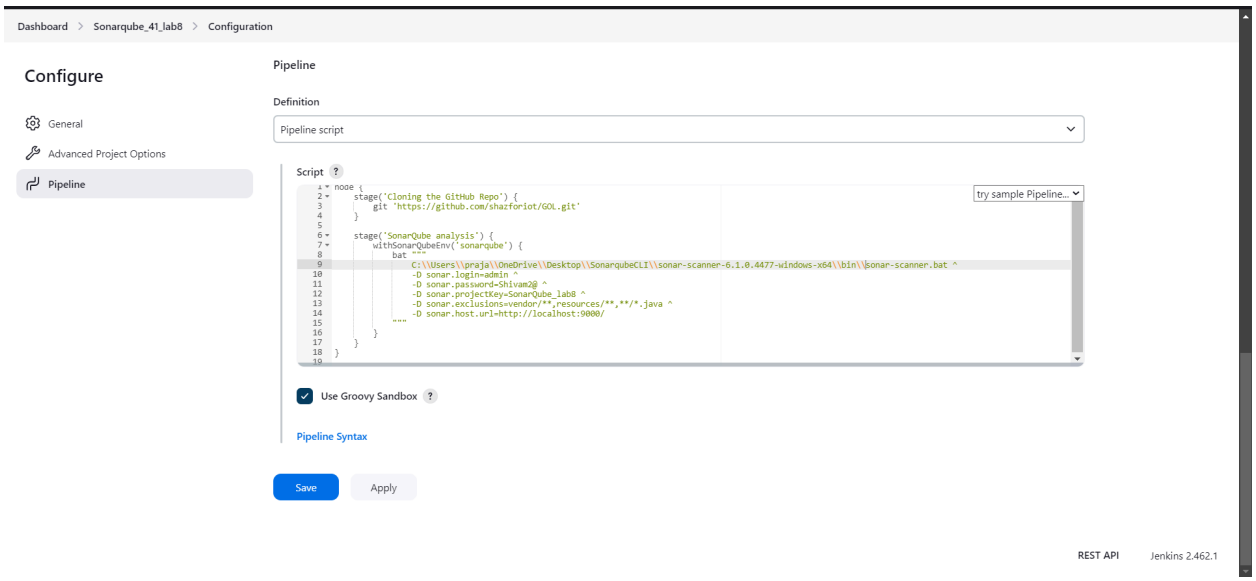


Step 10: Under Pipeline script, enter the following:

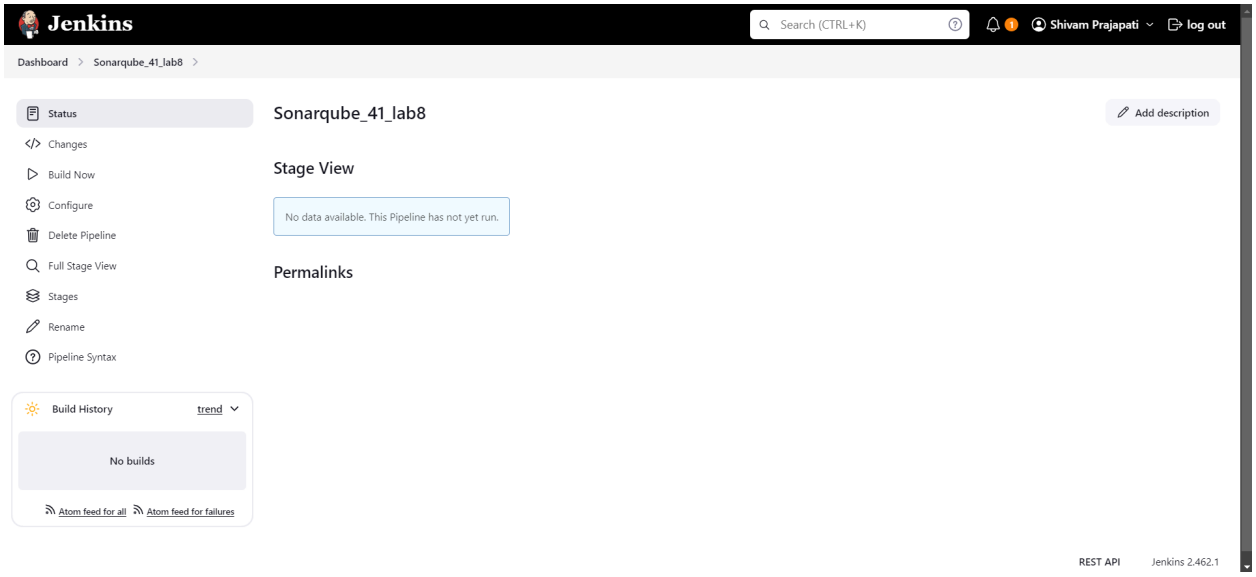
```
node {
   stage('Cloning the GitHub Repo') {
      git 'https://github.com/shazforiot/GOL.git'
```

```
    }
    stage('SonarQube analysis') {
        withSonarQubeEnv('sonarqube') {
            bat """

C:\\Users\\praja\\Downloads\\SonarqubeCLI\\sonar-scanner-6.1.0.4477-windows-x64\\bin\\sonar-scanner.bat ^
            -D sonar.login=admin ^
            -D sonar.password=Shivam2@ ^
            -D sonar.projectKey=SonarQube_Lab8 ^
            -D sonar.exclusions=vendor/**,resources/**,**/*.java ^
            -D sonar.host.url=http://localhost:9000/
            """
        }
    }
}
```

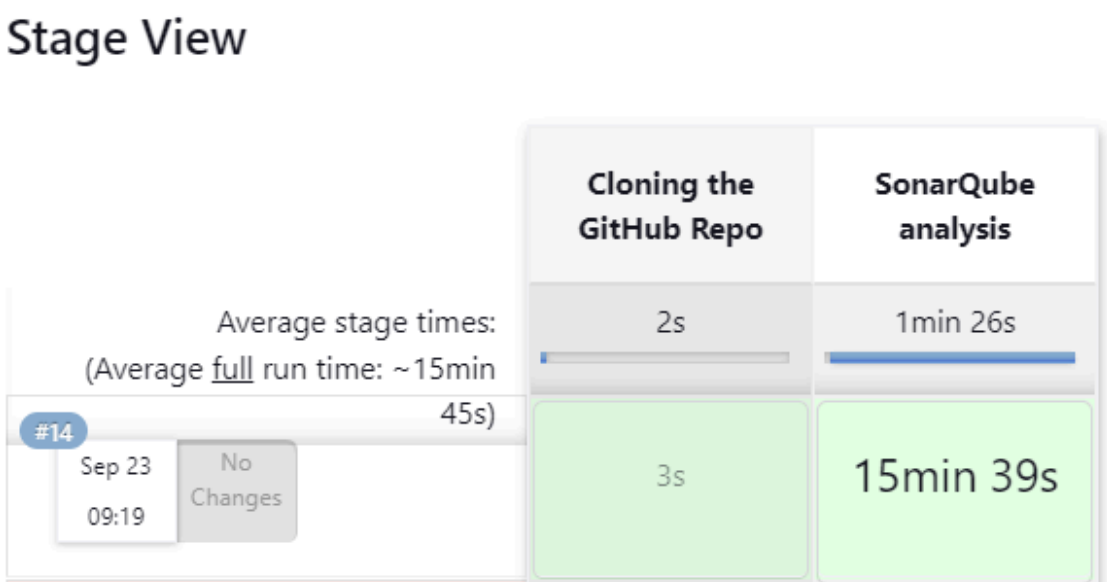


Click on save.



This is a Java sample project with many repetitive sections and coding issues that SonarQube will be able to detect during analysis.

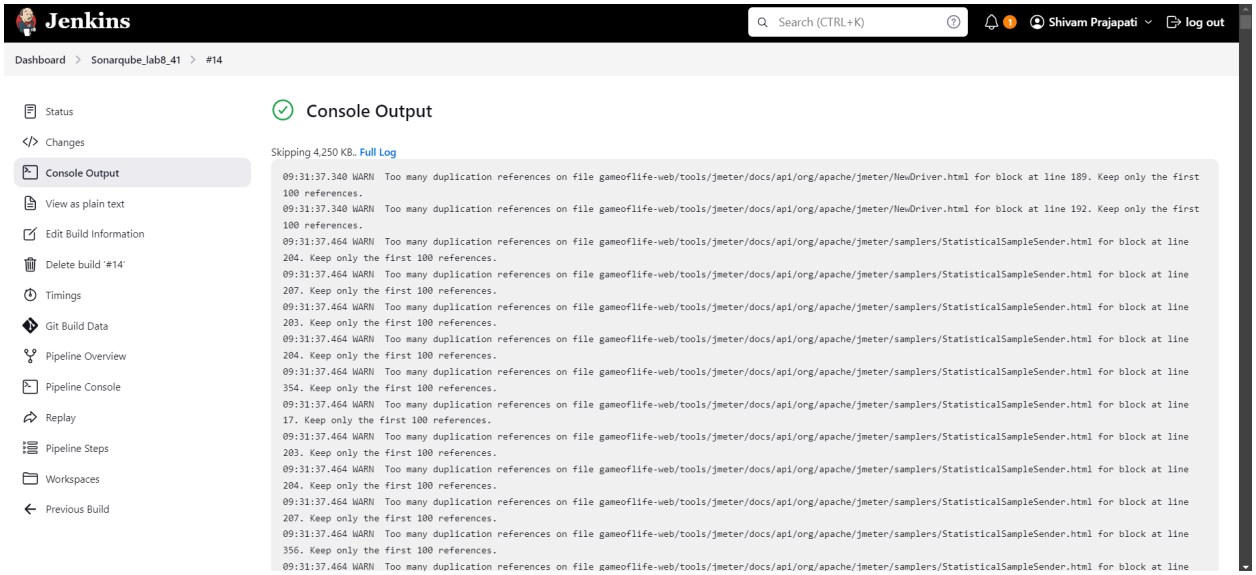
Step 11: Go back to jenkins. Go to the job you had just built and click on Build Now.

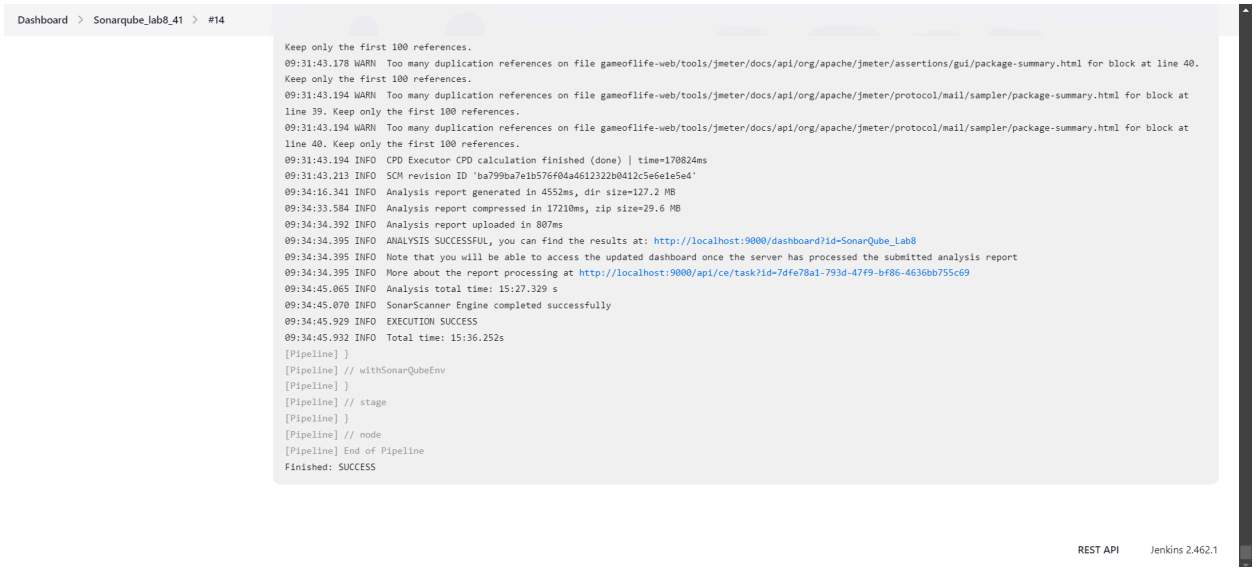


The problem was C:\windows\system32 was not there so we need to add in our environment variable .

Name: Shivam.R.Prajapati Div: D15C Roll No: 41 Academic Year: 2024-25

Now Check the console output once

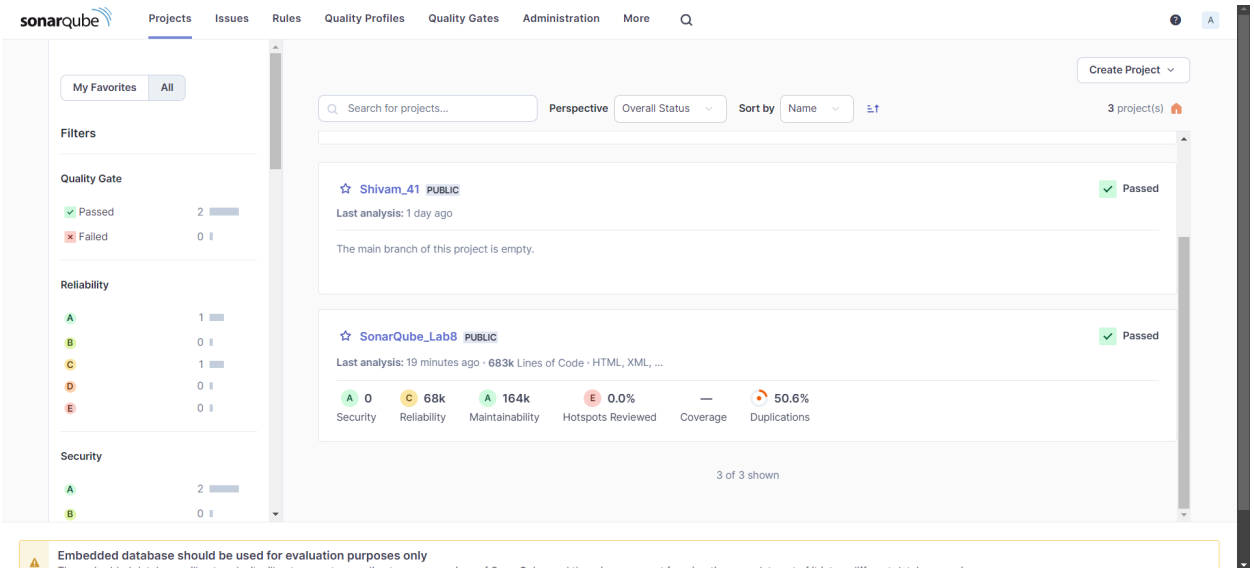


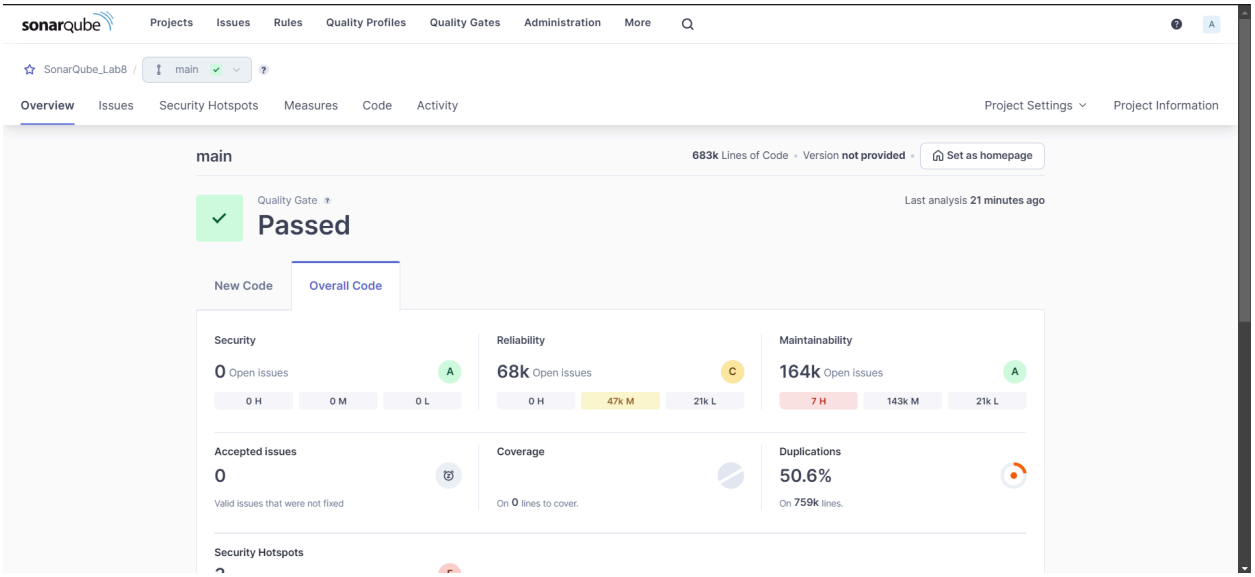


Successfully BUILD

Name: Shivam.R.Prajapati Div: D15C Roll No: 41 Academic Year: 2024-25

Step 12: After the build is finished, return to SonarQube and review the linked project in detail.



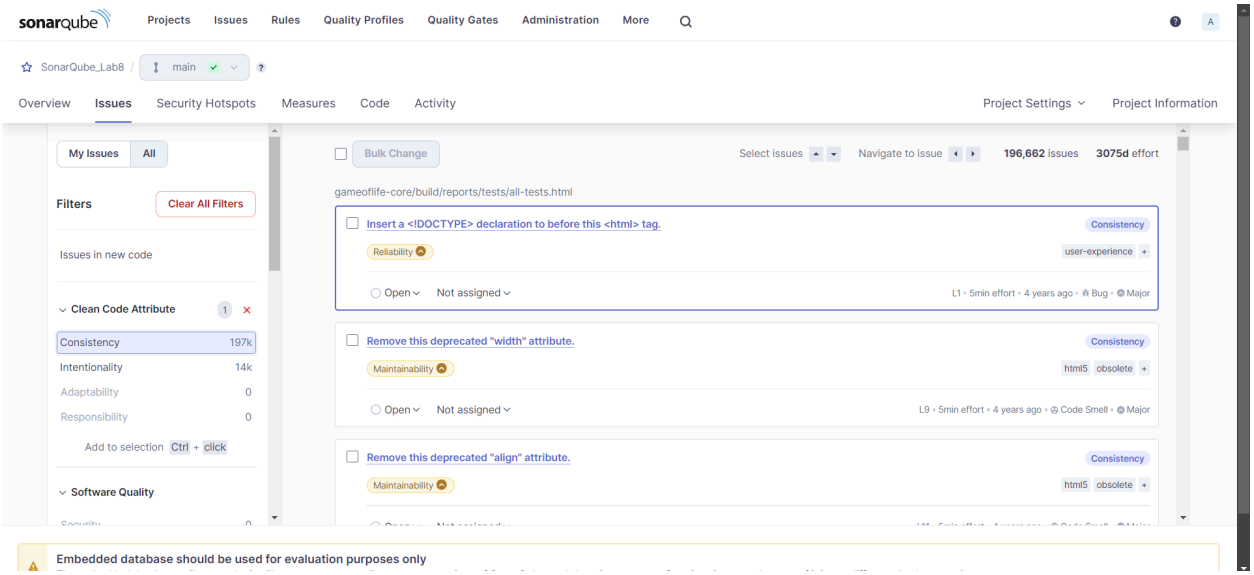


Under different options on the navbar , we can check all the issues with the code.

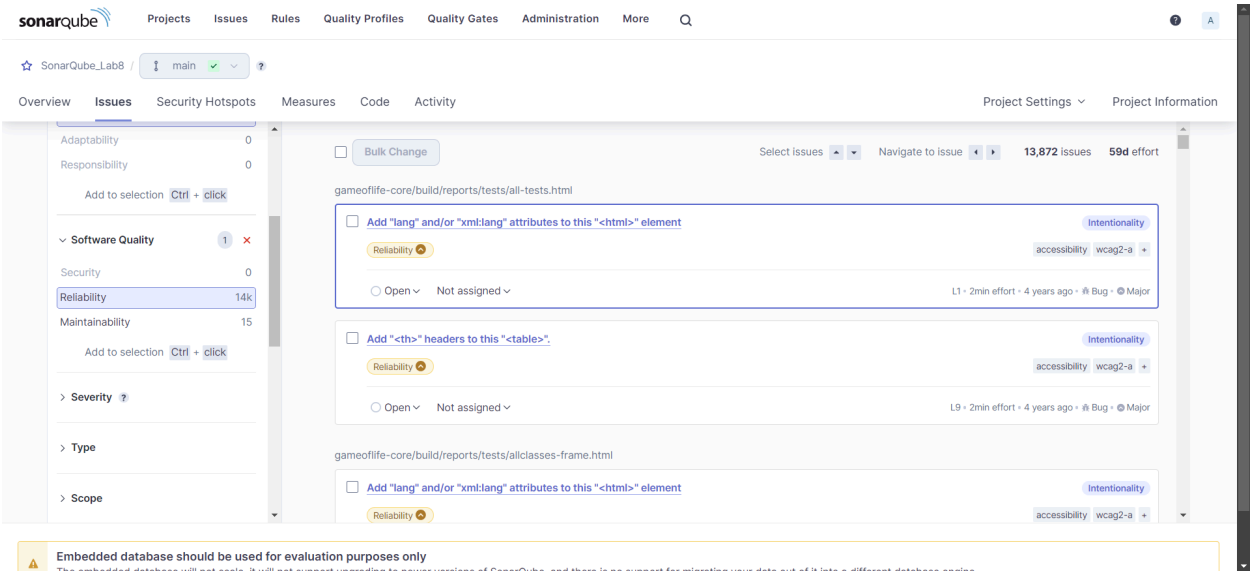
Name: Shivam.R.Prajapati Div: D15C Roll No: 41 Academic Year: 2024-25

UNDER ISSUES:

1) Consistency

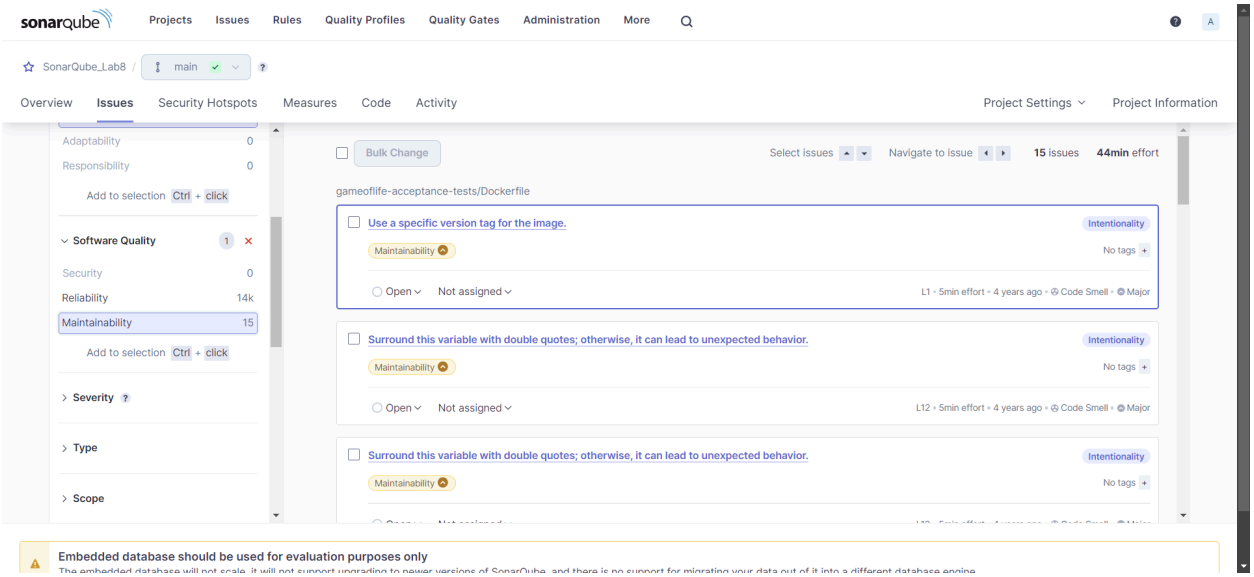


2) Reliability

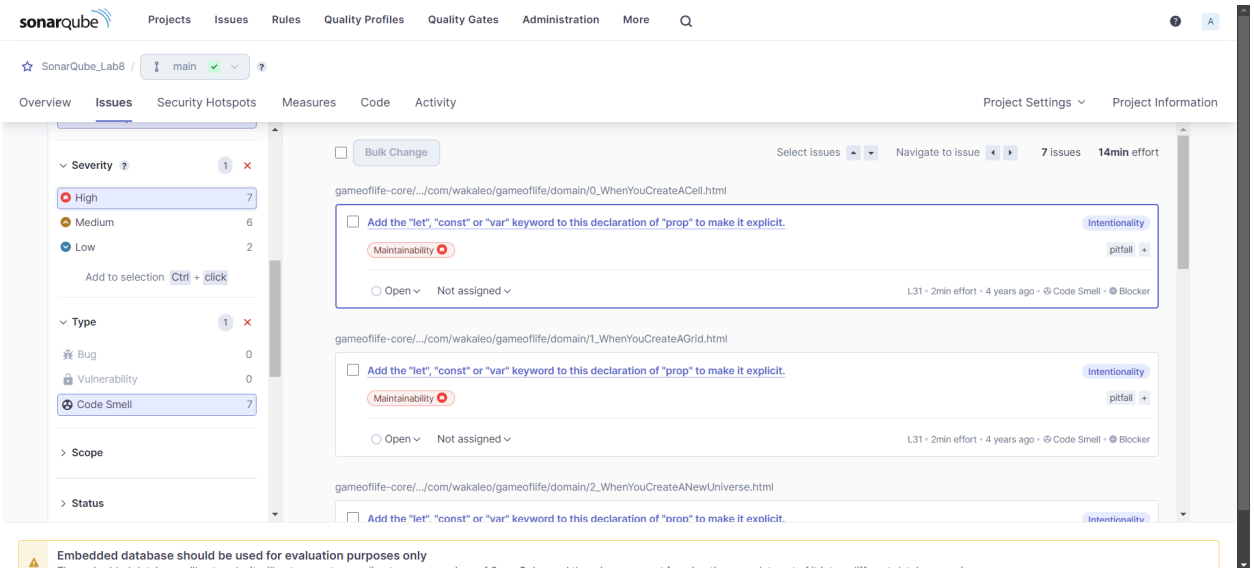


Name: Shivam.R.Prajapati Div: D15C Roll No: 41 Academic Year: 2024-25

3) Maintainability

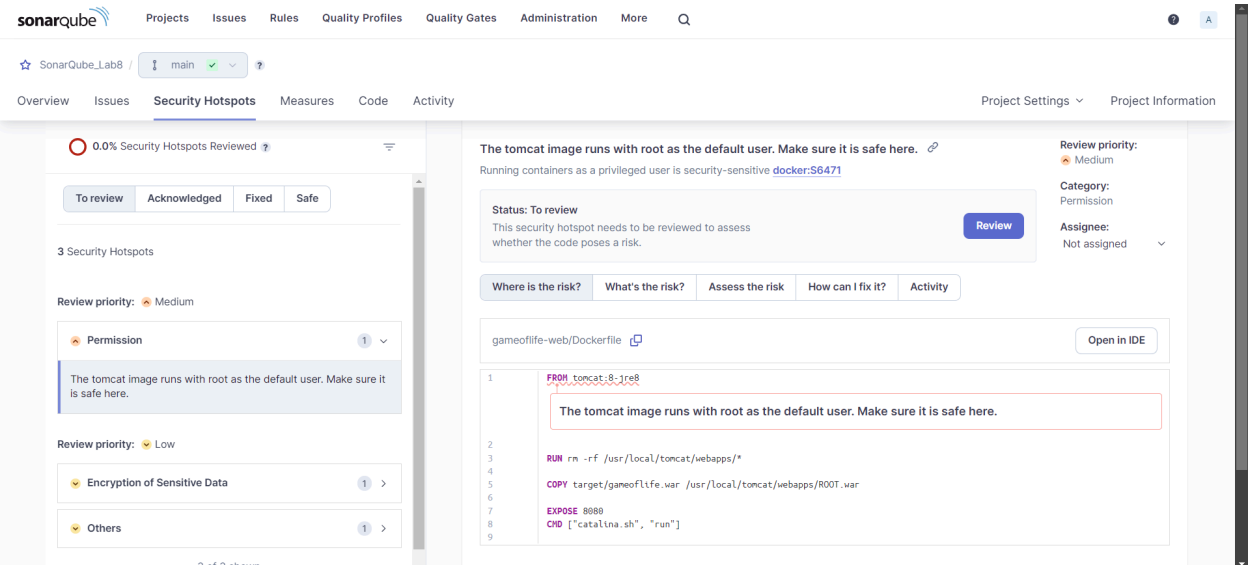


4) Severity

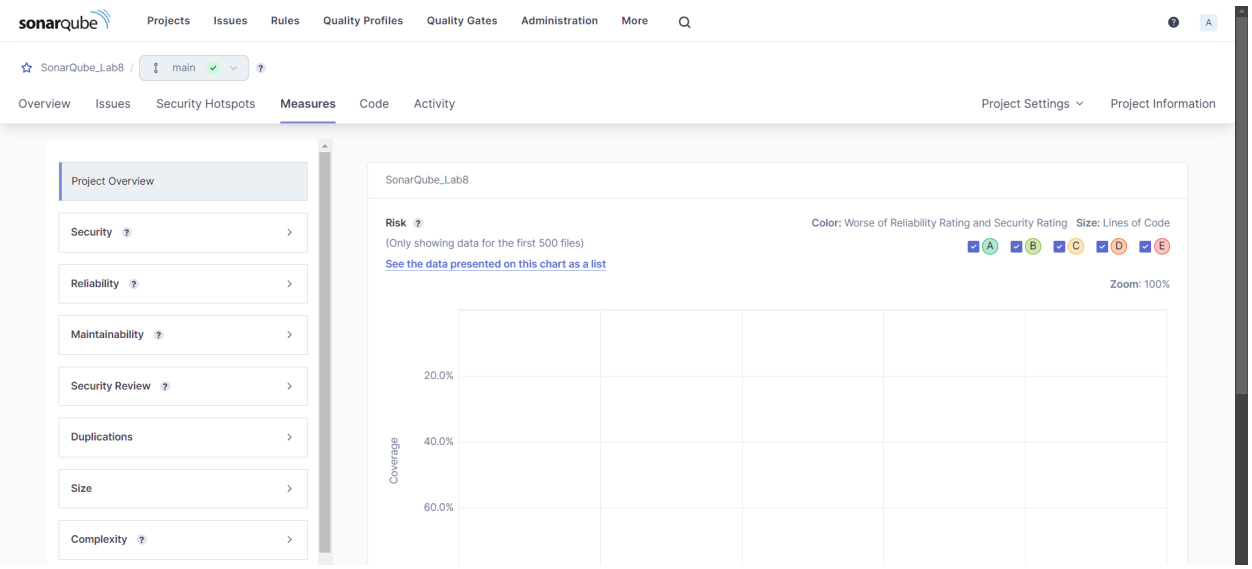


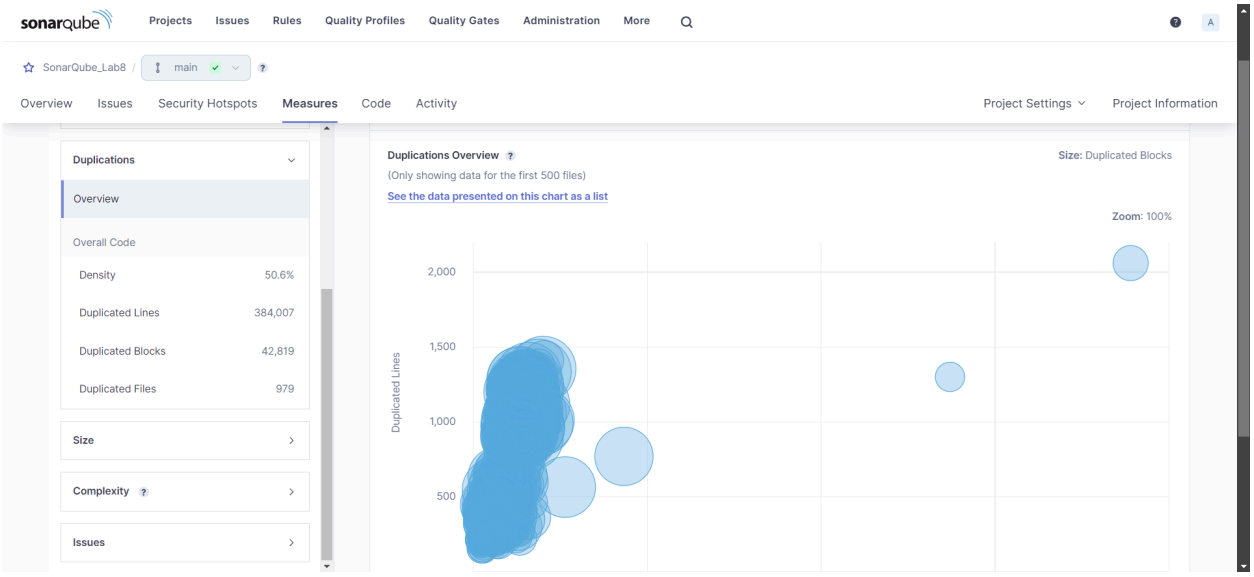
Name: Shivam.R.Prajapati Div: D15C Roll No: 41 Academic Year: 2024-25

UNDER SECURITY HOTSPOT:



UNDER MEASURES:





CONCLUSION:

In this experiment, we demonstrated how to perform static code analysis using Jenkins CI/CD Pipeline with SonarQube integration. We created a pipeline project with a specific script that contains all the instructions necessary to run SonarQube analysis. After configuring Jenkins appropriately, we built the project. The analyzed code in this experiment had several issues, such as errors, bugs, and duplications, all of which were detected and displayed in the linked SonarQube project.