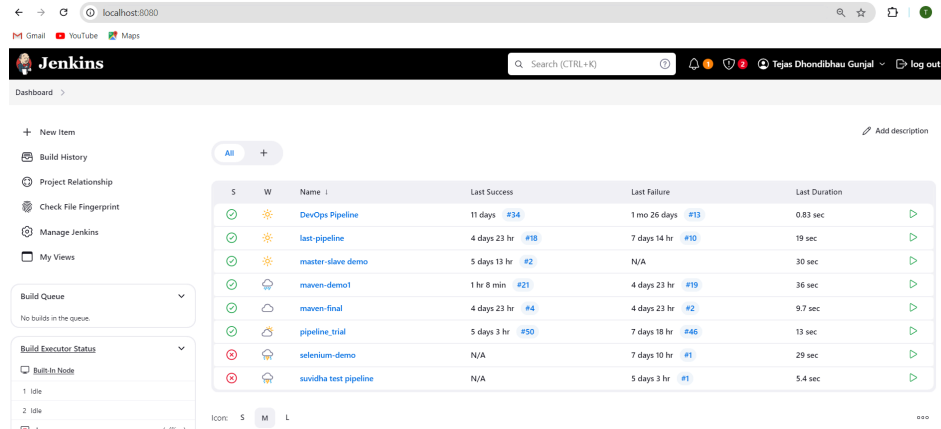


**EXPERIMENT NO: - 08**

**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.

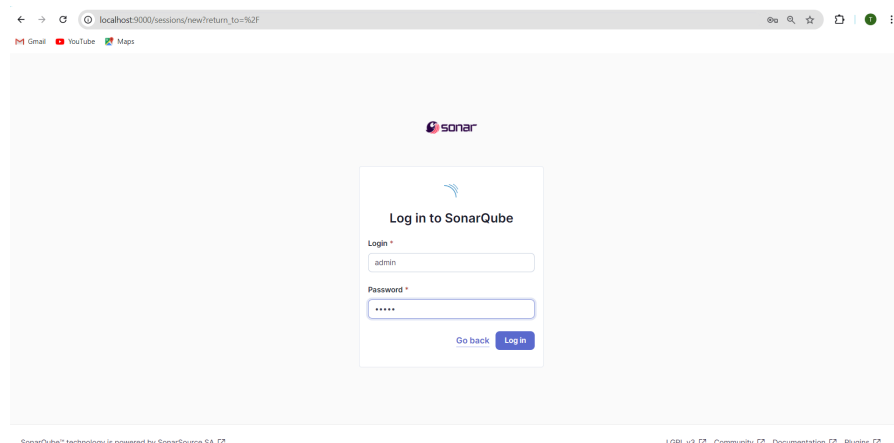
- Open up Jenkins Dashboard on localhost, port 8080 or whichever port it is at for you.



- 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`



- Once the container is up and running, you can check the status of SonarQube at localhost port 9000. Login to SonarQube using username admin and password admin.



- It also gives an option to update the password, for the security purposes

The image shows two screenshots of the SonarQube web interface. The top screenshot is the 'Update your password' page, which includes a warning that the default password should not be used, and fields for 'Old Password', 'New Password', and 'Confirm Password', all marked as required. The bottom screenshot is the 'How do you want to create your project?' page, which offers options to import from various DevOps platforms (Azure DevOps, Bitbucket Cloud, Bitbucket Server, GitHub, GitLab) or to create a local project. A footer note states that the embedded database is for evaluation purposes only.

- Create a manual project in SonarQube with the name sonarqube-test

1 of 2

## Create a local project

Project display name \*



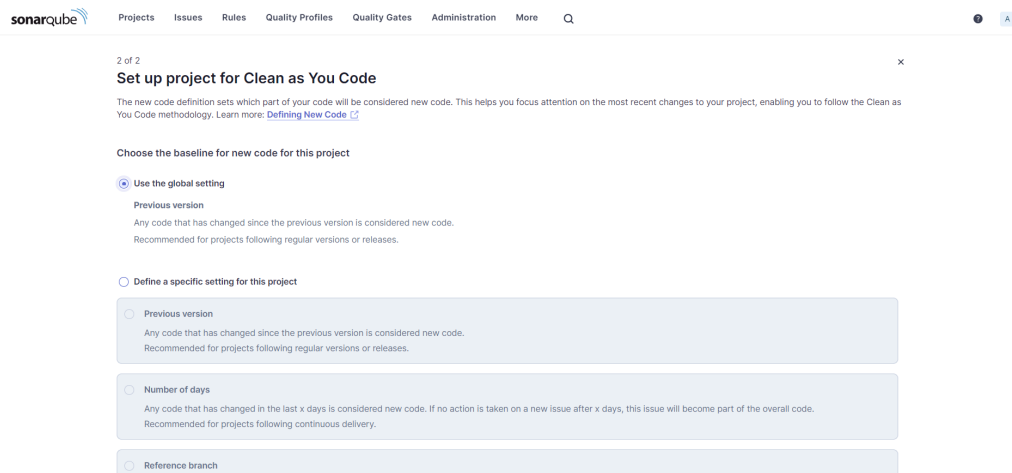
Project key \*



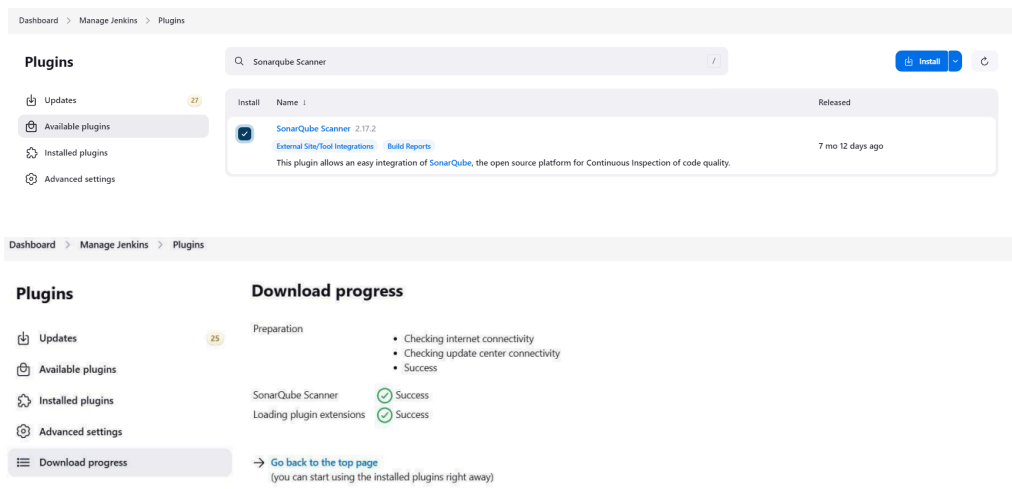
Main branch name \*

The name of your project's default branch [Learn More](#)

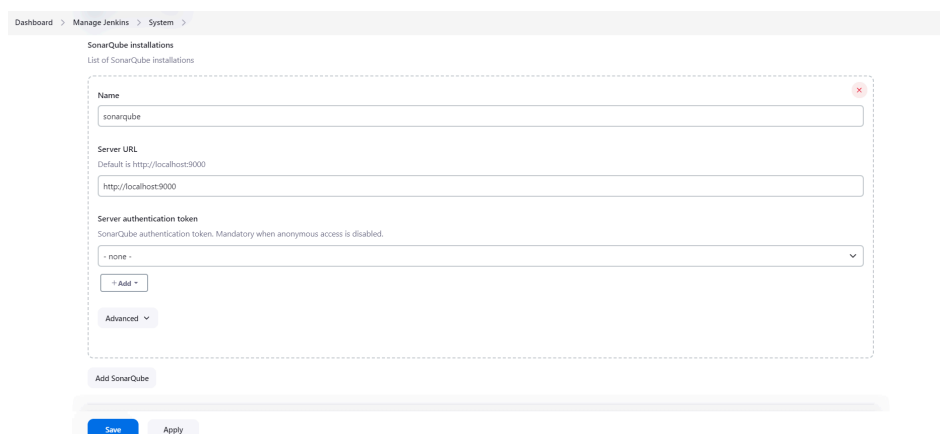
- ❑ Setup the project and come back to Jenkins Dashboard.



- ❑ Go to Manage Jenkins and search for SonarQube Scanner for Jenkins and install it.



- ❑ Under Jenkins 'Manage Jenkins' then go to 'system', scroll and look for SonarQube Servers and enter the details. Enter the Server Authentication token if needed.



- ❑ Search for SonarQube Scanner under Global Tool Configuration. Choose the latest configuration and choose Install automatically. Dashboard > Manage Jenkins > Tools

Dashboard > Manage Jenkins > Tools

Gradle installations

Add Gradle

SonarScanner for MSBuild installations

Add SonarScanner for MSBuild

SonarQube Scanner installations

Add SonarQube Scanner

Ant installations

Add Ant

Maven installations

Save Apply

- Check the “Install automatically” option. → Under name any name as identifier → Check the “Install automatically” option.

Dashboard > Manage Jenkins > Tools

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name

sonarqube\_exp8

☒ Install automatically ?

Install from Maven Central

Version

SonarQube Scanner 6.2.0.4584

Add Installer

Add SonarQube Scanner

Save Apply

- Generating a token on sonarqubes for authentication [optional]

Analysis Method > Locally

## Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Generate a project token Use existing token

Token name ? Expires in

Analyze "sonarqube-test" 90 days Generate

Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your [user account](#). See the [documentation](#) for more information.

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

- Copy the token you get here and save it securely as we would need it in Jenkins.

Analysis Method > Locally

## Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Analyze "sonarqube-test": **sqp\_35944e83f22ae4380284e2ffe40e2ddf63a1706** 🗑

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

Continue

## □ Update the Token in Jenkins

Click on global under the domains part of Stores scoped to Jenkins section. Further click on add credentials. Proceed with the following details. Make sure to copy the token generated earlier in sonarqube and give any suitable name as the ID.

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

### New credentials

Kind: Secret text

Scope: Global (jenkins, nodes, items, all child items, etc)

Secret: .....

ID: sonarqube\_practical

Description:

Create

### Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
jenkins-github-cicd	jenkins-github-cicd	Secret text	
sonarqube_practical	sonarqube_practical	Secret text	

Icons: S M L

## □ Now go to Manage Jenkins—>System—>SonarQube servers and proceed with the following details. Reference the authentication token generated in the previous step.

### SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☒ Environment variables

#### SonarQube installations

List of SonarQube installations

Name: sonarqube

Server URL: Default is http://localhost:9000  
http://localhost:9000

Server authentication token: SonarQube authentication token. Mandatory when anonymous access is disabled.  
sonarqube\_practical

+ Add

Advanced

Save Apply

## □ After configuration, create a New Item → choose a pipeline project

Under Pipeline script, enter the following:

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

Dashboard > sonarqube\_practical > Configuration

### Configure

General

Advanced Project Options

Pipeline

#### Pipeline

Definition: Pipeline script

```

1 node {
2   stage('Cloning the Github Repo') {
3     git 'https://github.com/shazforintrol.git'
4   }
5   stage('SonarQube analysis') {
6     withSonarQubeEnv('sonarqube') {
7       sh 'mvn'
8       docker run --rm --
9         -e SONAR_HOST_URL=http://172.20.100.1:9000 --
10        -v $(pwd):/src --name sonarqube/sonar-scanner-${*} /usr/src
11        -Dsonar.projectKey=sonarqube-text
12        -Dsonar.sources=${*}_java,resources
13        -Dsonar.testResources=${*}_java,resources
14        -Dsonar.language=java
15        -Dsonar.includedResources=${*}_java,resources
16      }
17   }
18 }
19
20

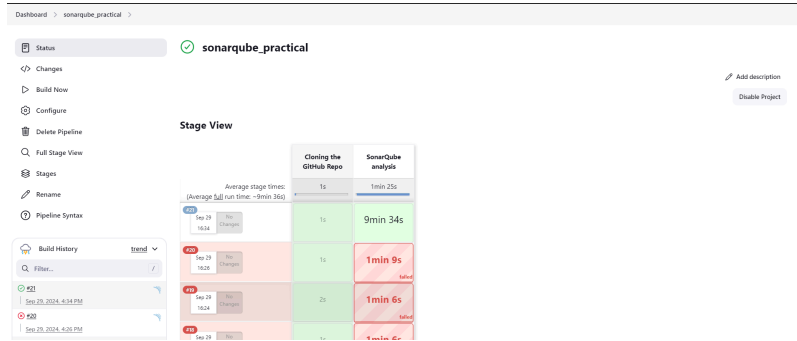
```

☒ Use Groovy Sandbox

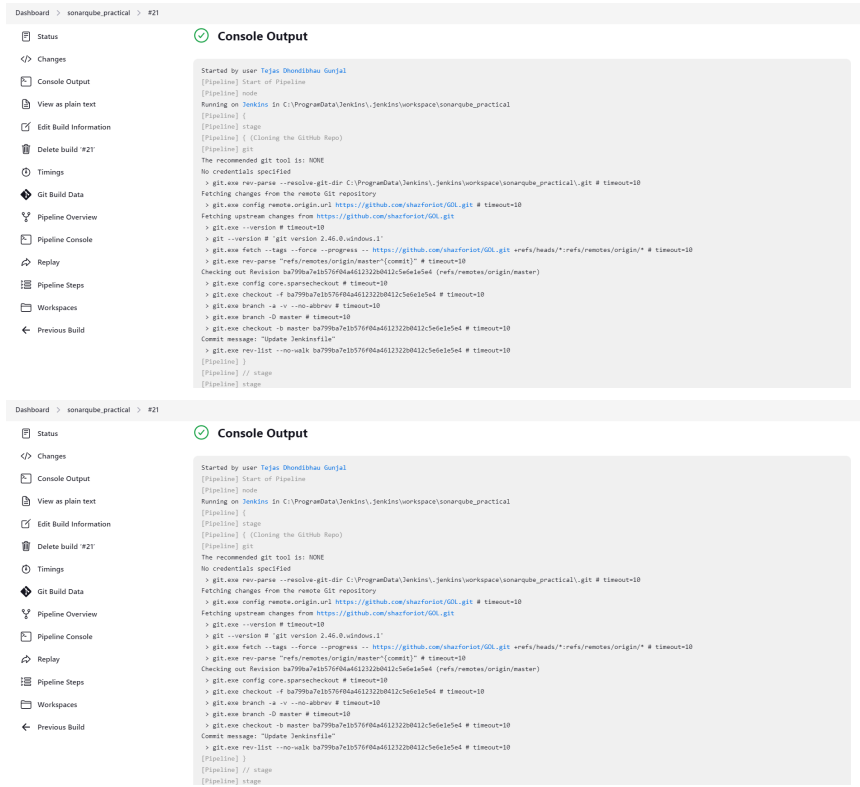
Pipeline Syntax

Save Apply

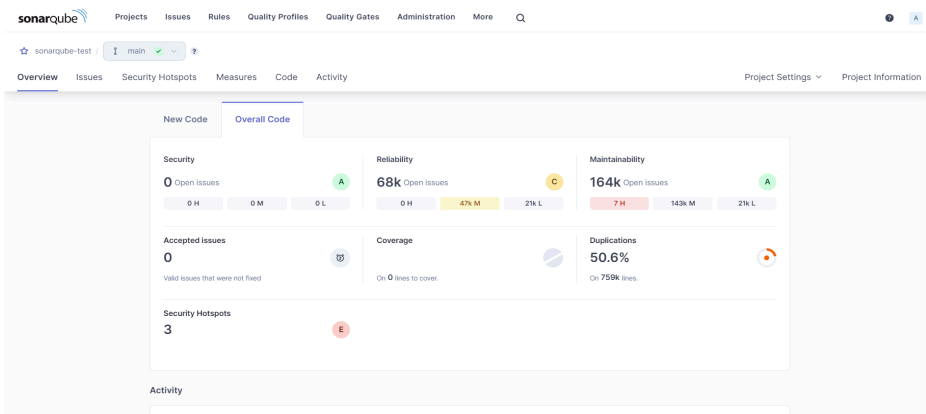
Build project



Check Console

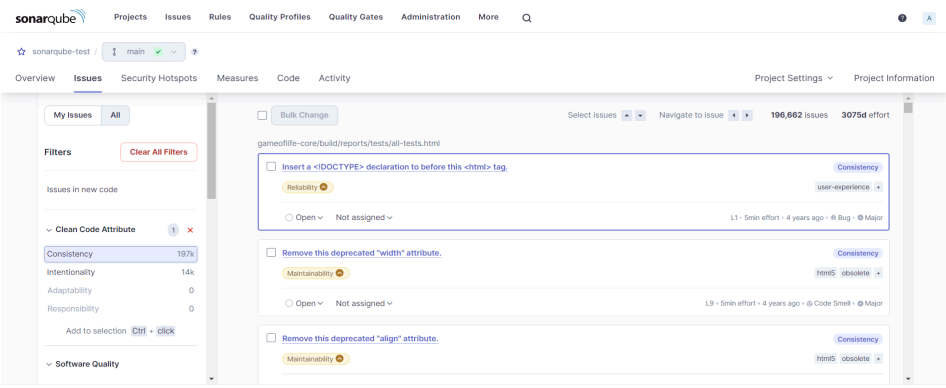


Now, check the project in SonarQube

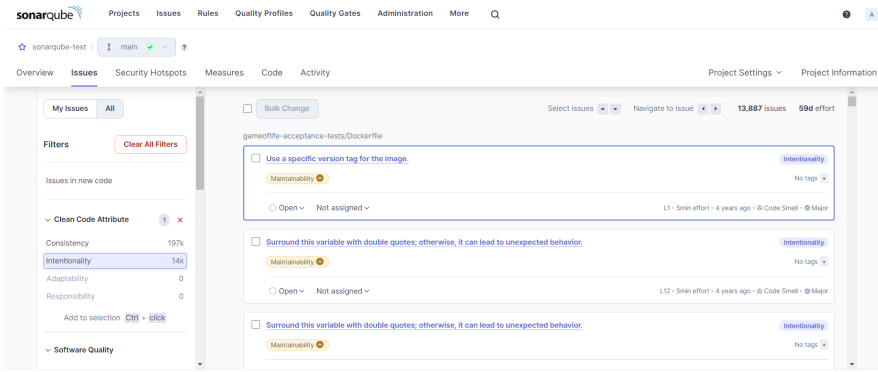


Code Problems

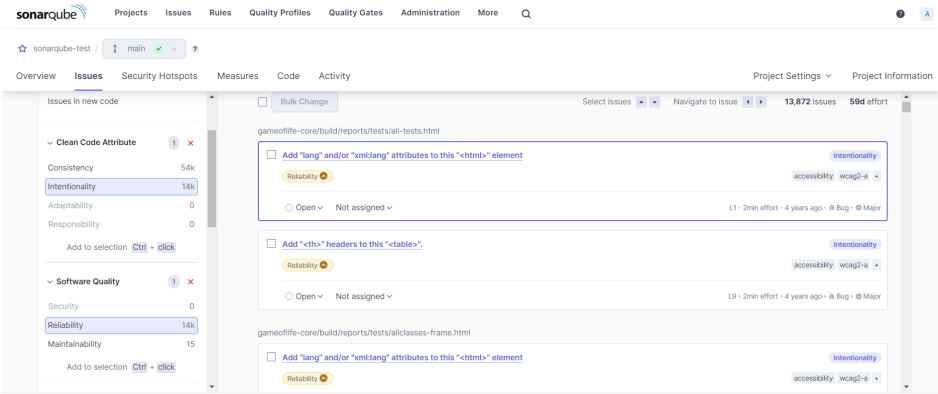
Consistency



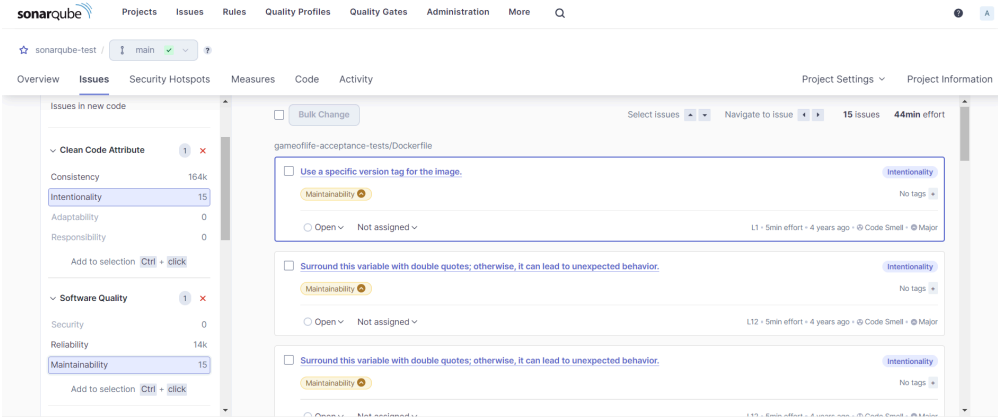
Intentionality



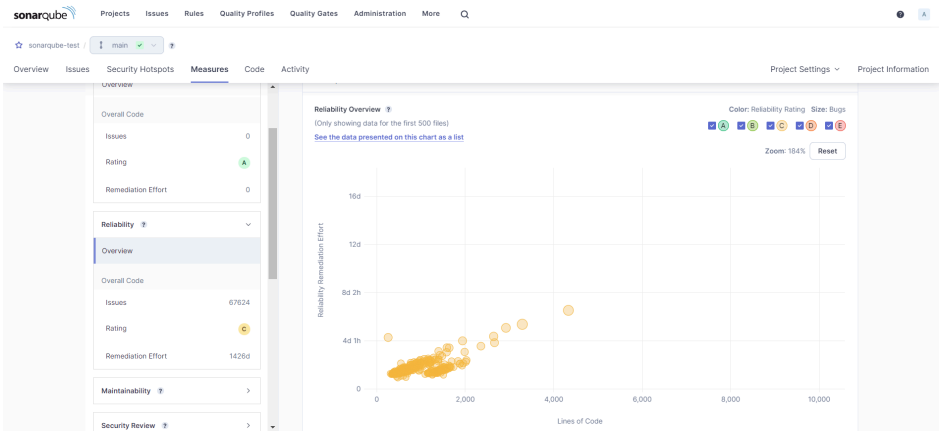
Bugs



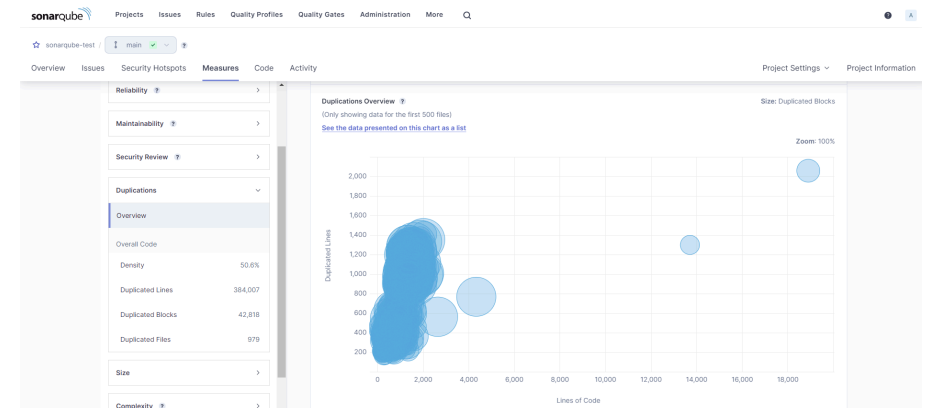
Code Smells



Reliability



Duplications



Cyclomatic Complexities

