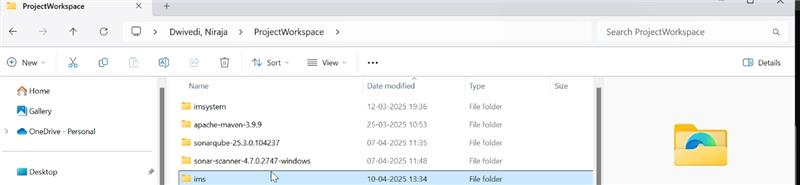
**Set Up SonarQube Locally for (IMS) Application**

1. Download the latest version **sonarqube-25.3.0.104237** from the SonarQube website and Unzip the downloaded file to a directory and also download **sonar-scanner-4.7.0.2747-windows** and unzip it

Sonar Qube : [SonarQube Download Success | Community Build | Sonar](https://www.sonarsource.com/products/sonarqube/downloads/success-download-community-edition/)

Sonar Scanner : [SonarScanner for Maven | SonarQube Server Documentation](https://docs.sonarsource.com/sonarqube-server/latest/analyzing-source-code/scanners/sonarscanner-for-maven/)



1. **From System properties ->Advance System Setting – Environment Variable – Path**

Add sonar qube and sonar scanner both in path variable of Environment Variable

C:\Users\nidwived\ProjectWorkspace\sonarqube-25.3.0.104237\bin

C:\Users\nidwived\ProjectWorkspace\sonar-scanner-4.7.0.2747-windows\bin

1. **Add Sonar Scanner Plugin to `pom.xml`\*\***

Include To Integrate SonarQube and Sonar Scanner Maven plugin with Spring Boot, include below plugin in `pom.xml` file. This plugin will allow you to run SonarQube analysis as part of your Maven build.

<build>

<pluginManagement>

<plugins>

<plugin>  
 <groupId>org.sonarsource.scanner.maven</groupId>  
 <artifactId>sonar-maven-plugin</artifactId>  
 <version>3.9.1.2184</version>  
 </plugin>

</plugins>

</pluginManagement>

</build>

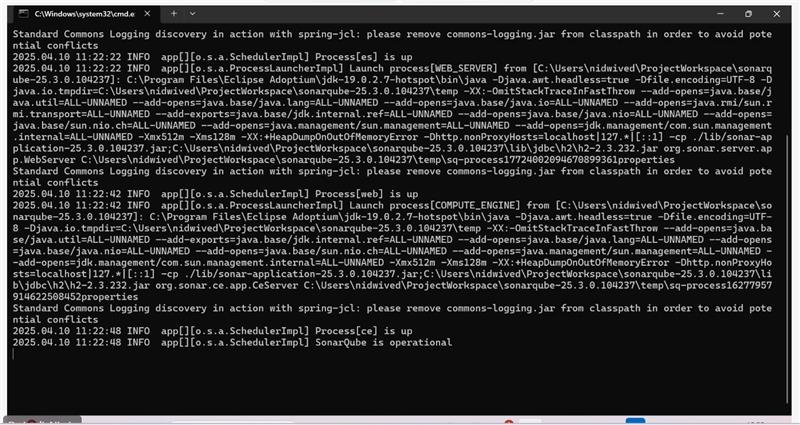
A computer screen shot of a computer screen

AI-generated content may be incorrect.

1. **Start Sonar Qube :**Navigate to the `bin` directory and run the appropriate script for your OS (e.g., `StartSonar.bat` for Windows or `sonar.sh` for Unix-based systems , if any issue will come we can check the log at path C:\Users\nidwived\Downloads\sonarqube-25.3.0.104237\sonarqube-25.3.0.104237\logs

Mostly port related issues will come so we can change the default port 9000 to 9001

We can navigate to path C:\Users\nidwived\ProjectWorkspace\sonarqube-25.3.0.104237\bin\windows-x86-64 and will start StartSonar.bat



1. To Generate Access Token Login to SonarQube and Open [http://localhost:9001](http://localhost:9001) and login with default credentials (`admin:admin`). Change the password when prompted. Go to your account settings, navigate to the Security tab, and generate a token for any provided string like IMS\_SPRING\_BOOT\_APP
   * Log in to your SonarQube instance at `http://localhost:9001`.
   * Navigate to \*\*My Account > Security\*\*.
   * Click on \*\*Generate Tokens\*\*. After entering name
   * It will generate token , copy that token

Token : squ\_21bae7aab99496f2697e0c4d09cea65940f39121

1. **Configure SonarQube Properties** we can specify SonarQube properties directly in `pom.xml` or pass them as command-line arguments.

There are various ways to configure it

* Add in pom.xml
* Add in command line argument while running command
* Add in setting.xml (Global /User setting)
* Add in sonar-project.properties

**POM** :Here’s how to add them to your `pom.xml`:

```xml

<properties>

<sonar.projectKey>PROJECT\_KEY</sonar.projectKey>

<sonar.host.url>http://localhost:9000</sonar.host.url>

<sonar.login>GENERATED\_TOKEN</sonar.login>

</properties>

A screenshot of a computer

AI-generated content may be incorrect.

**Command** : mvn clean verify sonar:sonar -Dsonar.projectKey=PROJECT\_KEY -Dsonar.host.url=http://localhost:9000 -Dsonar.login=GENERATED\_TOKEN

**Setting.xml :** The settings.xml file in Maven can be found in two main locations:

**Global Settings:** This is located in the Maven installation directory, typically at ${M2\_HOME}/conf/settings.xml. This file contains settings that apply to all users on a machine.

**User Settings:** This is located in the user's home directory, typically at ${user.home}/.m2/settings.xml. This file contains user-specific settings and takes precedence over the global setting

<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.2.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.2.0 https://maven.apache.org/xsd/settings-1.2.0.xsd">

<pluginGroups>

<pluginGroup>org.sonarsource.scanner.maven</pluginGroup>

</pluginGroups>

<profiles>

<profile>

<id>sonar</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<properties>

<!-- Optional URL to server. Default value is http://localhost:9000 -->

<sonar.projectKey>my-spring-boot-app</sonar.projectKey>

<sonar.host.url>http://localhost:9001</sonar.host.url>

<sonar.token>squ\_21bae7aab99496f2697e0c4d09cea65940f39121</sonar.token>

</properties>

</profile>

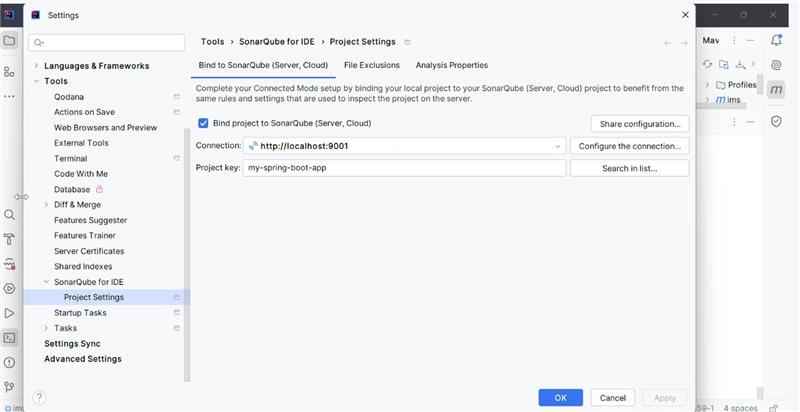
</profiles>

</settings>

**sonar-project.properties :**A screenshot of a computer

AI-generated content may be incorrect.

Once configured , restart sonar server , configured sonar qube setting will look like below



1. **Additional Configuration**

You can also configure additional properties such as exclusions, coverage reports, etc., in the `pom.xml`:

```xml

<properties>

<sonar.exclusions>\*\*/test/\*\*</sonar.exclusions>

<sonar.coverage.exclusions>\*\*/test/\*\*</sonar.coverage.exclusions>

</properties>

1. **Creating a local project or configure bitbucket/git labproject in SonarQube**

* Log in to SonarQube
* Navigate to Projects
* Create a New Project/ Local Project
* Enter Project Details

- \*\*Name\*\*: Enter a name for your project. This can be anything that helps you identify project.

- \*\*Key\*\*: Enter a unique project key. key will be used to identify your project in SonarQube.

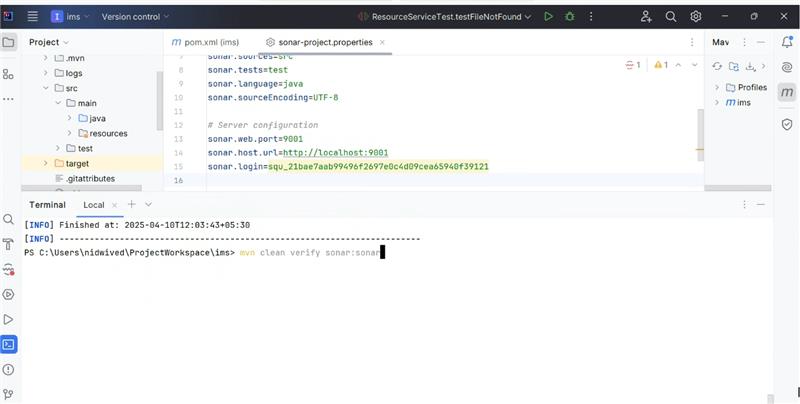
- \*\*Visibility\*\*: Choose the visibility of your project (public or private)

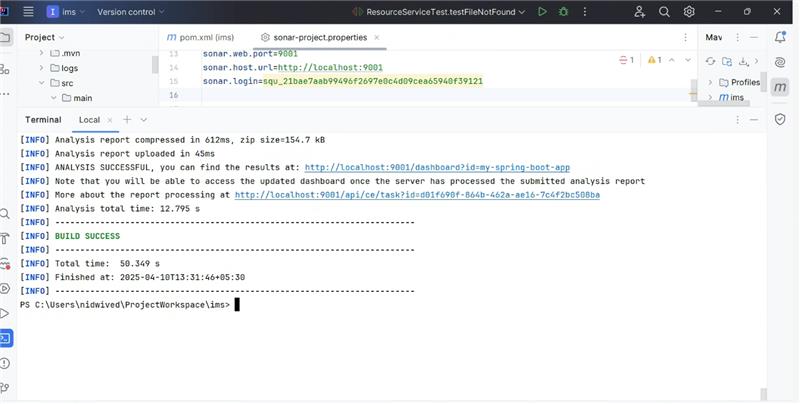
* Save the Project

- Click on \*\*Create\*\* to save your new project

1. **Running the First Analysis** Once the project is created, Run the analysis with the following

command **mvn clean verify sonar:sonar**

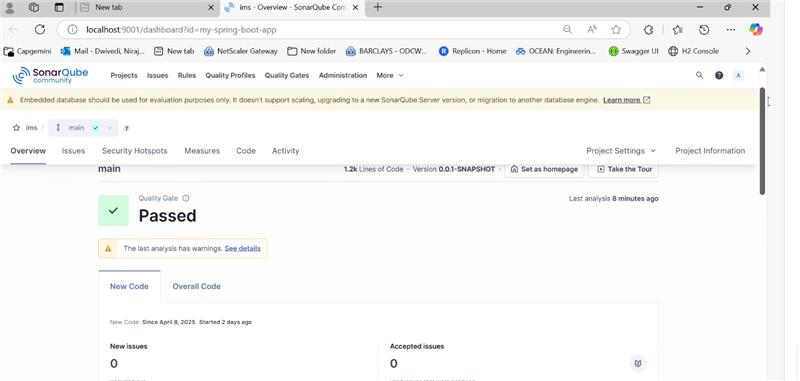




1. **Check the Analysis Report:**

Project dashboard provides an overview of the latest analysis, including key metrics like reliability, security, maintainability, coverage, and duplications.

- \*\*Issues\*\*: Navigate to the \*\*Issues\*\* tab to see a detailed list of issues detected in your code. You can filter these issues by severity, type, and status.



A screenshot of a computer

AI-generated content may be incorrect.

1. **Exclude** **from** **Sonar**:

We can exclude packages and files from SonarQube analysis in several ways. Below are the ways:

**1. Using `sonar-project.properties` File**

We can specify exclusions directly in the `sonar-project.properties` file:

```properties

sonar.exclusions=\*\*/com/example/package/\*\*

sonar.coverage.exclusions=\*\*/com/example/package/\*\*

**2**. **Using Maven Configuration/ Gradle Configuration**

If you are using Maven, you can add properties in your `pom.xml` file:

```xml

<properties> <sonar.exclusions>com/imsapp/ims/dto/claim/ClaimRequest.java,com/imsapp/ims/dto/policy/\*\*</sonar.exclusions>

<sonar.coverage.exclusions>\*\*/com/imsapp/ims/entity /\*\*</sonar.coverage.exclusions>

</properties>

**3**. **Using SonarQube UI**

We can also configure exclusions through the SonarQube web interface:

1. Navigate to \*\*Administration\*\* > \*\*General Settings\*\* > \*\*Analysis Scope\*\*.

2. Under \*\*Files\*\* > \*\*Source File Exclusions\*\*, add your exclusion patterns

**4. Using** **Command** **Line** **Parameters**

we can pass exclusion parameters directly when running the SonarQube scanner:

sonar-scanner -Dsonar.exclusions=\*\*/com/imsapp/ims/entity /\*\* -Dsonar.coverage.exclusions=\*\*/com/imsapp/ims/entity /\*\*

**Exclude** **from** **Jacoco**:

We can exclude packages and files from JaCoCo test coverage in several ways

**1. Using Maven Configuration**

You can specify exclusions in your `pom.xml` file:

xml

<plugin>

<groupId>org.jacoco</groupId>

<artifactId>jacoco-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>com/example/\*\*/ExcludedPOJO.class</exclude>

<exclude>com/example/\*\*/\*DTO.\*</exclude>

<exclude>\*\*/config/\*</exclude>

</excludes>

</configuration>

</plugin>

**2. Using Custom Annotations**

Starting from JaCoCo 0.8.2, you can exclude classes and methods by annotating them with a custom annotation that includes `Generated` in its name:

@Documented

@Retention(RUNTIME)

@Target({TYPE, METHOD, CONSTRUCTOR})

public @interface Generated {

}

```

**3. Using Command Line Parameters**

You can pass exclusion parameters directly when running the JaCoCo Maven plugin:

mvn jacoco:report -Djacoco.excludes=com/example/\*\*/ExcludedPOJO.class,com/example/\*\*/\*DTO.\*,\*\*/config/\*

**4. Using JaCoCo Agent Configuration**

You can configure exclusions in the JaCoCo agent settings, typically used for integration tests:

xml

<configuration>

<destFile>${project.build.directory}/jacoco.exec</destFile>

<excludes>

<exclude>com/example/\*\*/ExcludedPOJO.class</exclude>

<exclude>com/example/\*\*/\*DTO.\*</exclude>

<exclude>\*\*/config/\*</exclude>

</excludes>

</configuration>