

# Code Quality & Design Principles Lab Assessment

Srienath Vaisakh N

Emp ID: 12215

SonarQube


Goto sonarqube file->bin->windowsx64 file->start sonar

Type localhost:9000 in web browser

## Create a project


All fields marked with \* are required

**Project display name \***



Up to 255 characters. Some scanners might override the value you provide.

**Project key \***



The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '\_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

**Main branch name \***

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


**Set Up**

## 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

Token name 

Expires in

30 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.

☐ Use existing token

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](#).

### 2 Run analysis on your project

What option best describes your build?

Maven

Gradle

.NET

Other (for JS, TS, Go, Python, PHP, ...)

What option best describes your build?

Maven

Gradle

.NET

Other (for JS, TS, Go, Python, PHP, ...)

#### Execute the Scanner for Maven

Running a SonarQube analysis with Maven is straightforward. You just need to run the following command in your project's folder.

```
mvn clean verify sonar:sonar \
-Dsonar.projectKey=CourseManage \
-Dsonar.host.url=http://localhost:9000 \
-Dsonar.login=sqp_d4ce83017a178a0973220edc10d79009fe2ed88d
```

Discover the [official documentation](#) of the Scanner for Maven for more details.

Copy and put this in the command prompt of the project you want to use.

```

[INFO] 13:32:44.874 More about the report processing at http://localhost:9000/api/ce/task?id=AZB3ngv
[INFO] 13:32:44.943 Analysis total time: 17.084 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 48.208 s
[INFO] Finished at: 2024-07-03T13:32:44+05:30
[INFO] -----

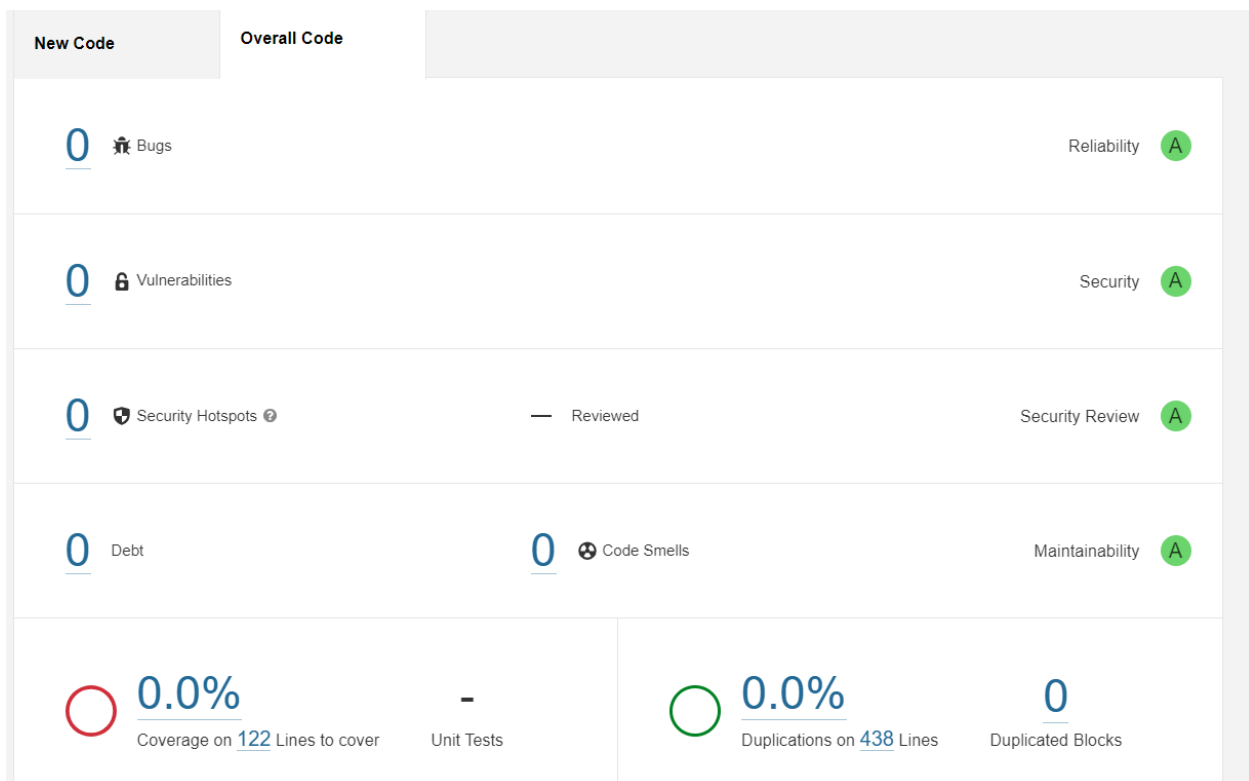
C:\Users\srienath.n\Documents\workspace-spring-tool-suite-4-4.21.1.RELEASE\CourseRestApi>

```

Build has been successful

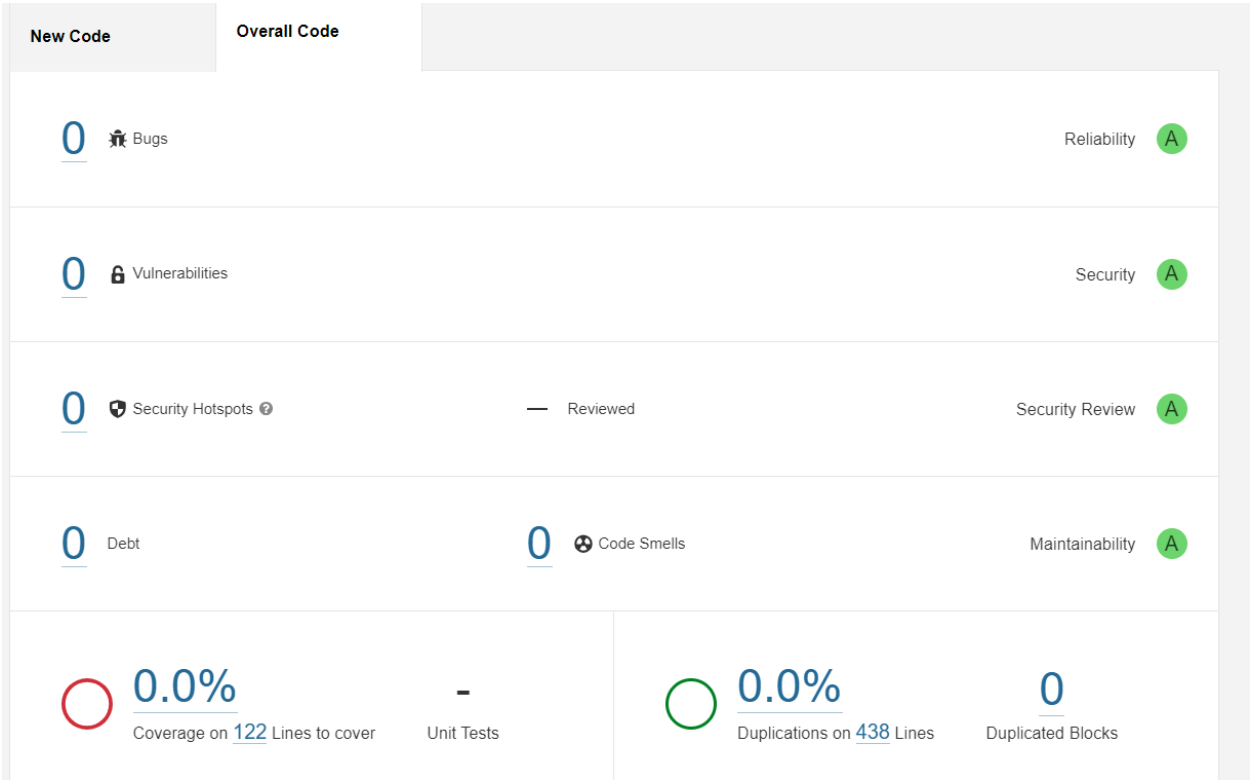
Now go to web browser

Before fixing errors & bugs



No errors needed to be fixed

After



Manual Review -> Accept Jeevarajans collaborator request for his project

-> Accept Pull Request made as reviewer for his project

-> Access his files and copy HTTPs clone link from Codes

-> Create a separate folder to access his git files and access git bash in it

```
srienath.n@RLP1430 MINGW64 /d/JeevaGitCheck
$ git clone https://github.com/Hardin-J/Online-Pet-shop-Management-System.git
Cloning into 'Online-Pet-shop-Management-System'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 29 (delta 4), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (29/29), 6.07 KiB | 1.21 MiB/s, done.
Resolving deltas: 100% (4/4), done.

srienath.n@RLP1430 MINGW64 /d/JeevaGitCheck
$ ls
Online-Pet-shop-Management-System/

srienath.n@RLP1430 MINGW64 /d/JeevaGitCheck
$ AC

srienath.n@RLP1430 MINGW64 /d/JeevaGitCheck
$ cd Online-Pet-shop-Management-System

srienath.n@RLP1430 MINGW64 /d/JeevaGitCheck/Online-Pet-shop-Management-System (main)
$ git branch -r
  origin/HEAD -> origin/main
  origin/FirstBranch
  origin/main

srienath.n@RLP1430 MINGW64 /d/JeevaGitCheck/Online-Pet-shop-Management-System (main)
$ git branch srienath

srienath.n@RLP1430 MINGW64 /d/JeevaGitCheck/Online-Pet-shop-Management-System (main)
$ git checkout srienath
Switched to branch 'srienath'

srienath.n@RLP1430 MINGW64 /d/JeevaGitCheck/Online-Pet-shop-Management-System (srienath)
$ git pull origin firstBranch
From https://github.com/Hardin-J/Online-Pet-shop-Management-System
 * branch            firstBranch -> FETCH_HEAD
Updating fbe6b9a..9e000c4
Fast-Forward
.../psms/PetShopManagementSystemApplication.java | 13 +++
.../com/mrj/psms/controller/PetShopController.java | 75 +++++
.../com/mrj/psms/controller/PetsController.java | 75 +++++
src/main/java/com/mrj/psms/model/PetShop.java | 98 +++++
src/main/java/com/mrj/psms/model/Pets.java | 84 +++++
src/main/java/com/mrj/psms/repository/PetRepo.java | 9 ++
.../java/com/mrj/psms/repository/PetShopRepo.java | 9 ++
src/main/java/com/mrj/psms/service/PetServ.java | 18 ++++
.../java/com/mrj/psms/service/PetShopServ.java | 18 ++++
.../com/mrj/psms/service_impl/PetServImpl.java | 52 +++++
.../com/mrj/psms/service_impl/PetShopServImpl.java | 53 +++++
src/main/resources/application.properties | 10 +++
12 files changed, 514 insertions(+)
create mode 100644 src/main/java/com/mrj/psms/PetShopManagementSystemApplication.java
create mode 100644 src/main/java/com/mrj/psms/controller/PetShopController.java
create mode 100644 src/main/java/com/mrj/psms/controller/PetsController.java
create mode 100644 src/main/java/com/mrj/psms/model/PetShop.java
create mode 100644 src/main/java/com/mrj/psms/model/Pets.java
create mode 100644 src/main/java/com/mrj/psms/repository/PetRepo.java
create mode 100644 src/main/java/com/mrj/psms/repository/PetShopRepo.java
create mode 100644 src/main/java/com/mrj/psms/service/PetServ.java
create mode 100644 src/main/java/com/mrj/psms/service/PetShopServ.java
create mode 100644 src/main/java/com/mrj/psms/service_impl/PetServImpl.java
create mode 100644 src/main/java/com/mrj/psms/service_impl/PetShopServImpl.java
create mode 100644 src/main/resources/application.properties
```

Now all his files will be cloned into the folder we created locally access each backend file one by one and review it.

1) Package names are abbreviated very shortly, clear definition of package relating to project would make it easier to understand to readers.

```
com.mrj.psms.controller;  
com.mrj.psms.model;  
Com.mrj.psms.repository;  
com.mrj.psms.controller;
```

2) In Controller package, both in Petshop & Pets controller there seems to be no issue with code. No bug, no code smells, there is proper indentation used in each line, proper spacings, annotations are used properly.

3) Constructor injection is used instead of Auto wired in controller, service packages, Good.

4) In Model Pets & Petshop naming conventions maintained properly, getter setters and other constructors are implemented properly.

5) In Repo for both petshop and pet is good no repository annotation used.

6) In service package for both petshop and pets no unnecessary imports made, abstract methods called correctly.

7) ServiceImpl no methods used in service are left unused, implemented thoroughly

OverAll : Good Job in maintaining coding standard, quality. Correct the package names alone