

ASSIGNMENT 7

Aim :-

To perform static analysis on python programs using sonarQube SAST process

Theory :-

SonarQube is a universal tool for static code analysis that has become more or less the industry standard. Keeping code clean, simple, and easy to read is also a lot easier with SonarQube.

What is SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications. It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Benefits of SonarQube

Sustainability - Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications. Increase productivity - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code

Quality code - Code quality control is an inseparable part of the process of software development.

Detect Errors - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.

Why SonarQube?

Developers working with hard deadlines to deliver the required functionality to the customer. It is so important for developers that many times they compromise with the code quality, potential bugs, code duplications, and bad distribution of complexity. Additionally, they tend to leave unused variables, methods, etc. In this scenario, the code would work in the desired way.

To avoid these issues in code, developers should always follow the good coding practice, but sometimes it is not possible to follow the rules and maintain the good quality as there may be many reasons.

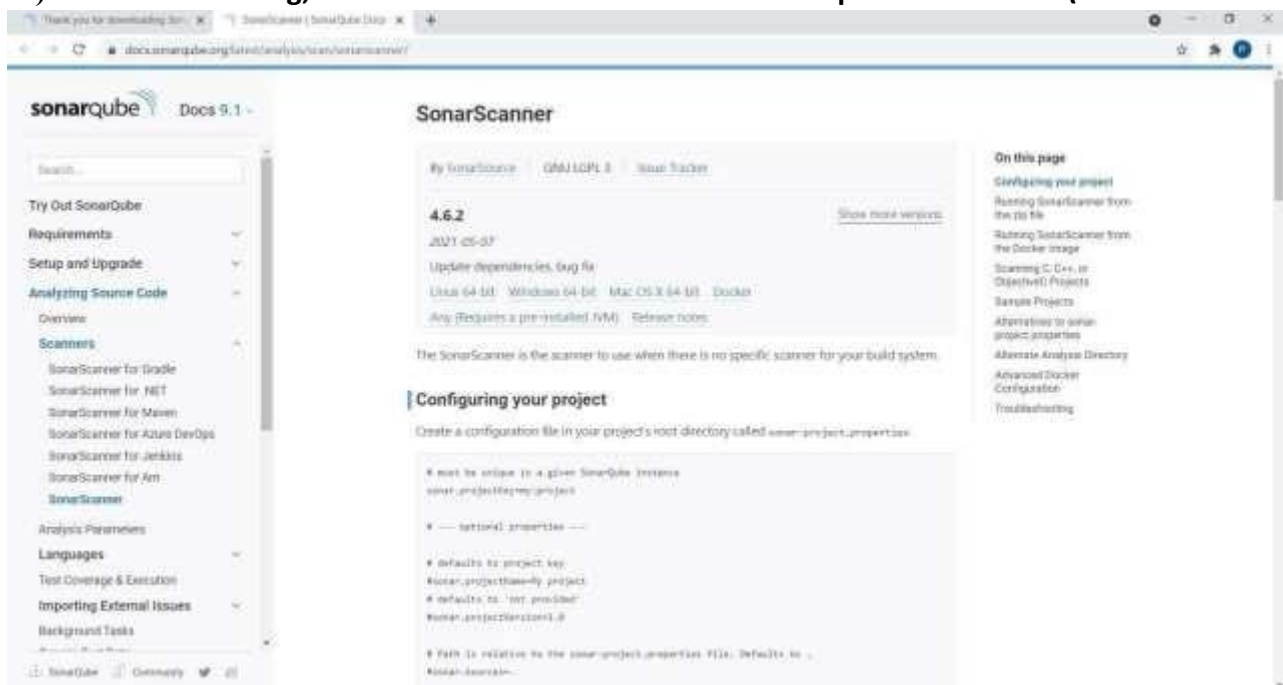
In order to achieve continuous code integration and deployment, developers need a tool that not only works once to check and tell them the problems in the code but also to track and control the code to check continuous code quality. To satisfy all these requirements, here comes SonarQube in the picture.

Steps :-

1)Download sonarqube



2) After downloading, set Environment Variables. Add “sonarqube-9.1.0.47736\bin” to Path.



3) Open command prompt. Run commands:

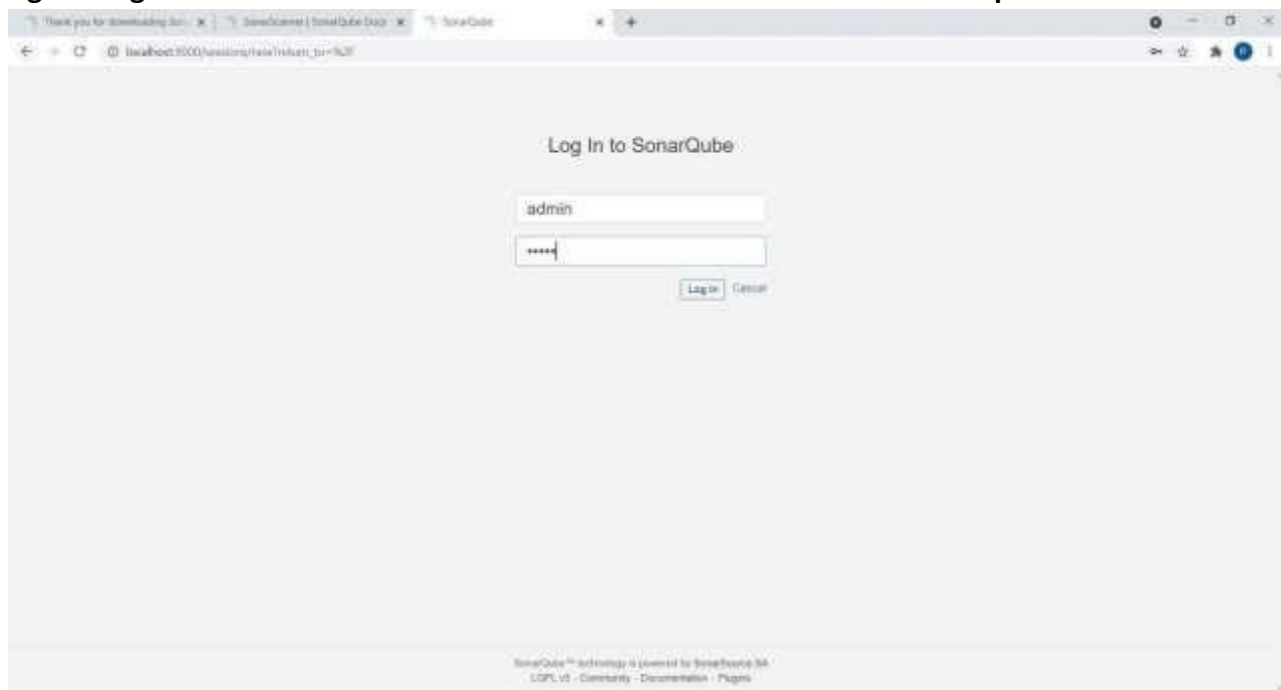
- **cd "sonarqube-9.1.0.47736\bin\windows-x86-64"**
- **StartSonar.bat**

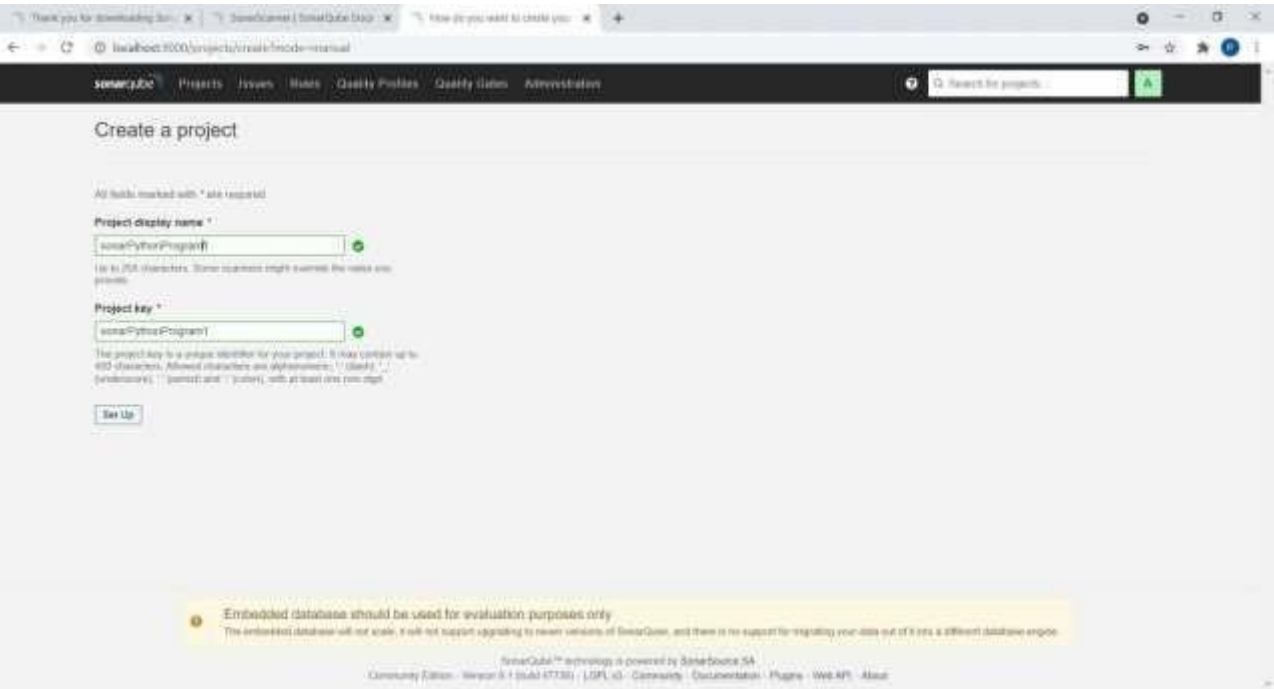
```
Command Prompt
C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin>sonar-scanner
INFO: Scanner configuration file: C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin\..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 4.6.2.2472
INFO: Java 11.0.11 AdoptOpenJDK (64-bit)
INFO: Windows 10 10.0 amd64
INFO: User cache: C:\Users\Priyansi\.sonar\cache
INFO: Scanner configuration file: C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin\..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: Analyzing on SonarQube server 9.1.0
INFO: Default locale: "en_IN", source code encoding: "windows-1252" (analysis is platform dependent)
INFO: Load global settings
INFO: -----
INFO: EXECUTION FAILURE
INFO: -----
INFO: Total time: 3.958s
INFO: Final Memory: 5M/20M
INFO: -----
ERROR: Error during SonarScanner execution
ERROR: Not authorized. Analyzing this project requires authentication. Please provide a user token in sonar.login or other credentials in sonar.login and sonar.password.
ERROR:
ERROR: Re-run SonarScanner using the -X switch to enable full debug logging.

C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin>
```

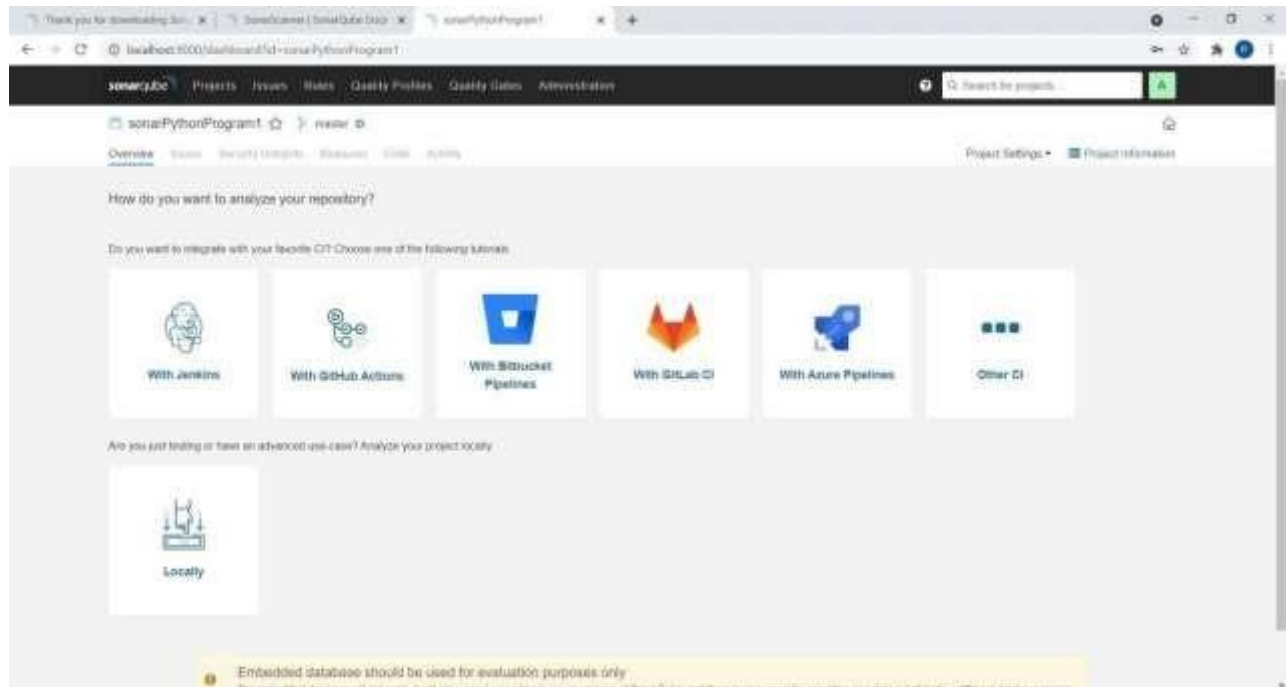
5) Server up and running on localhost:9000

Login using credentials as User: admin and Password: admin and Set a new password

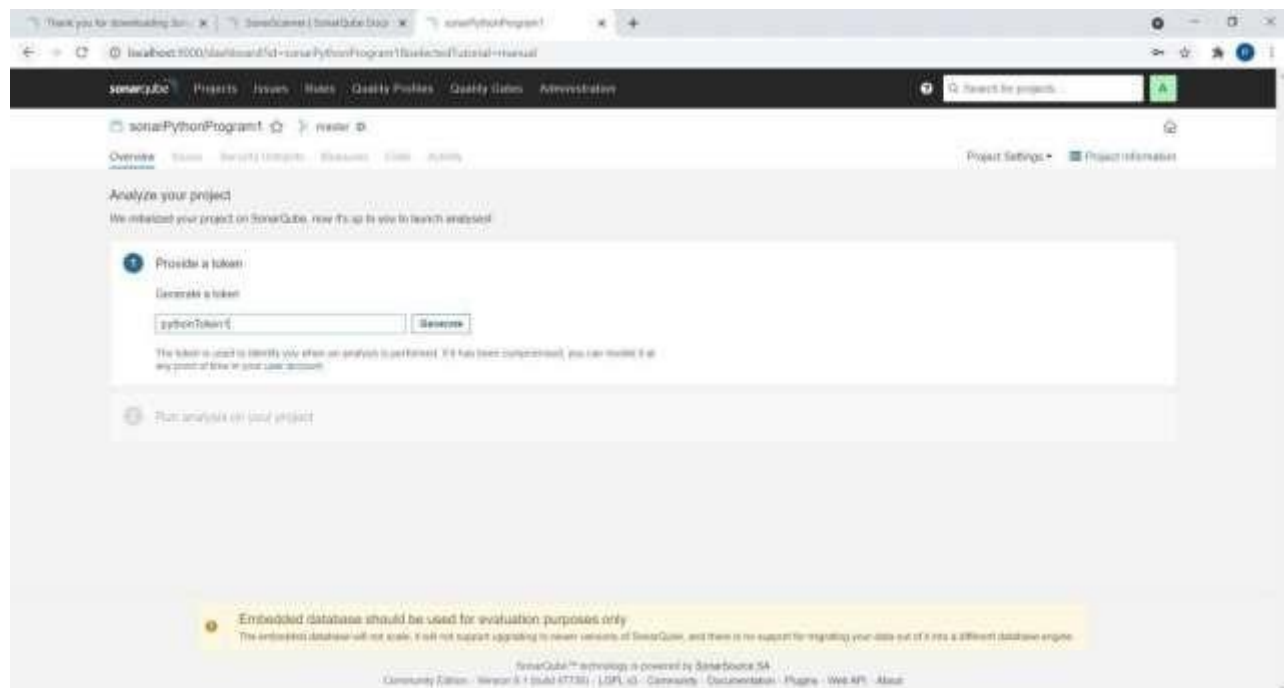




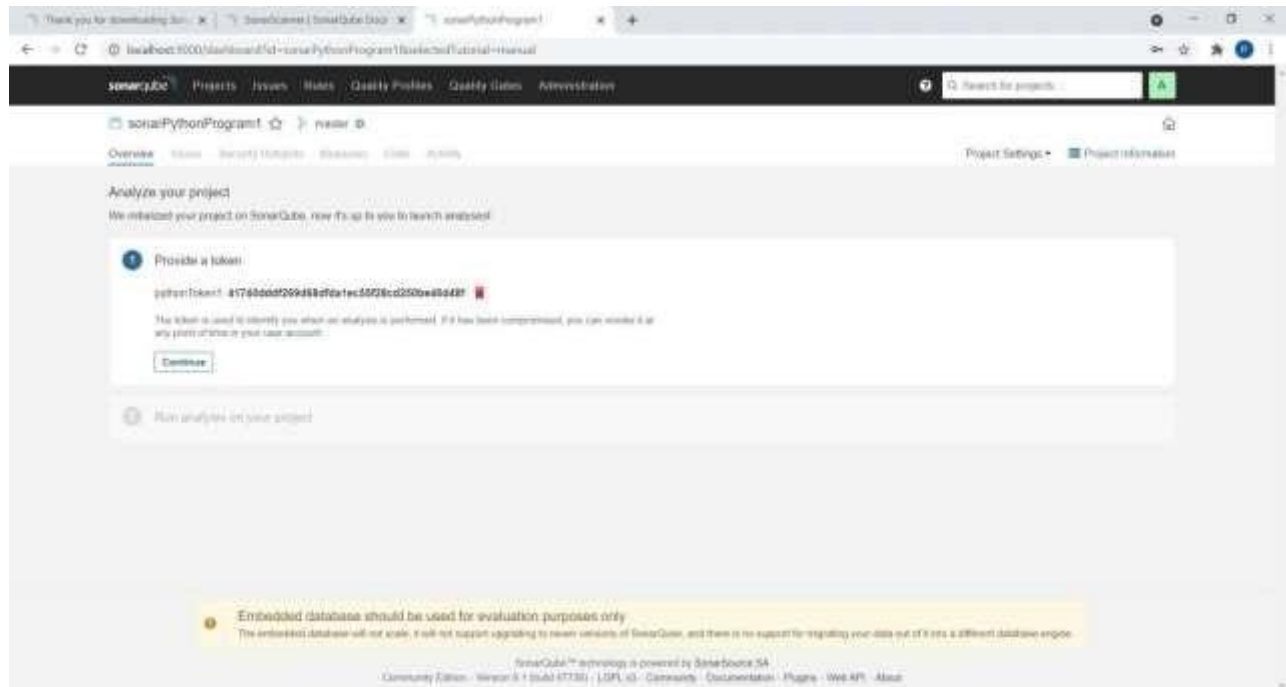
7) Give any Project display name.



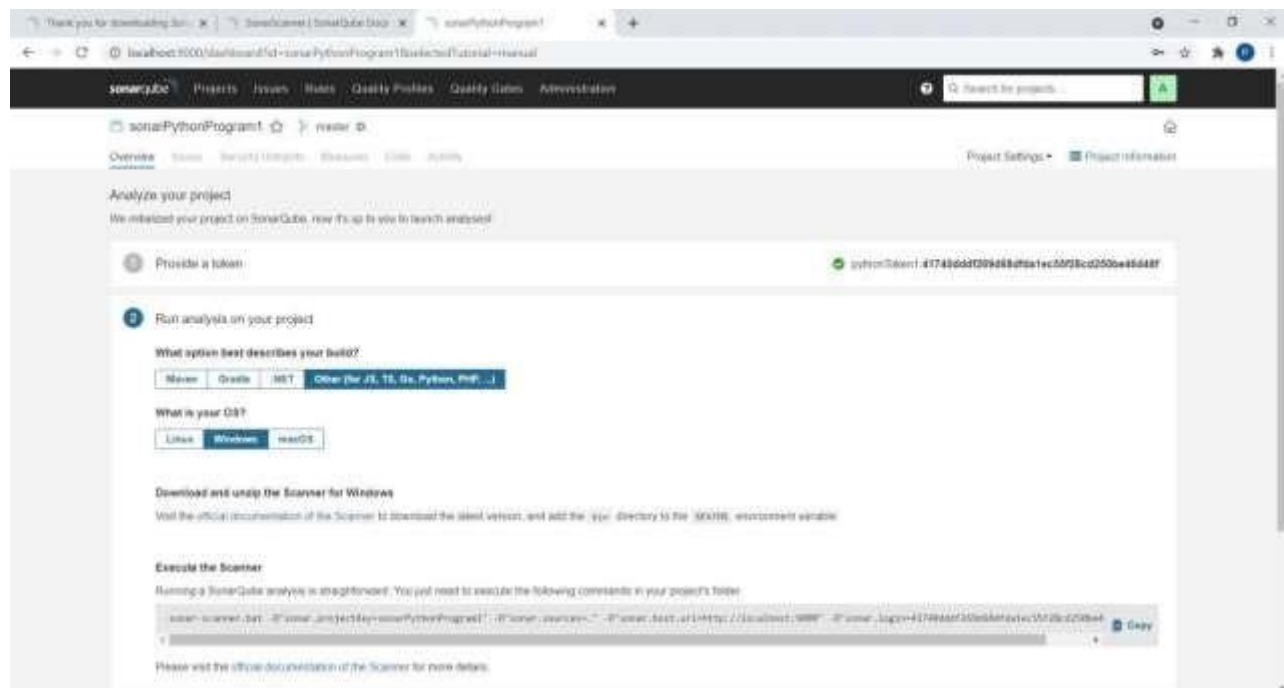
Click on Locally.



9) Give any name to token and click on Generate.



Click on Continue.



10) Save a Python program in a folder. class Solution(object):
 def romanToInt(self, s):
 roman =

```
{'I':1,'V':5,'X':10,'L':50,'C':100,'D':500,'M':1000,'IV':4,'IX':9,'XL':40,'XC':90,'CD':400,'CM':900}
```

```
i = 0
```

```
num = " " while
```

```
i < len(s):
```

```
    if i+1<len(s) and s[i:i+2] in roman:
```

```
        num+=roman[s[i:i+2]] i+=2
```

```
    else:
```

```
        #print(i)
```

```
        num+=roman[s[
```

```
        i]] i+=1
```

```
    return num
```

```
ob1 = Solution()
```

```
print(ob1.romanToInt("III")) print(ob1.romanToInt("CDXLIII"))
```

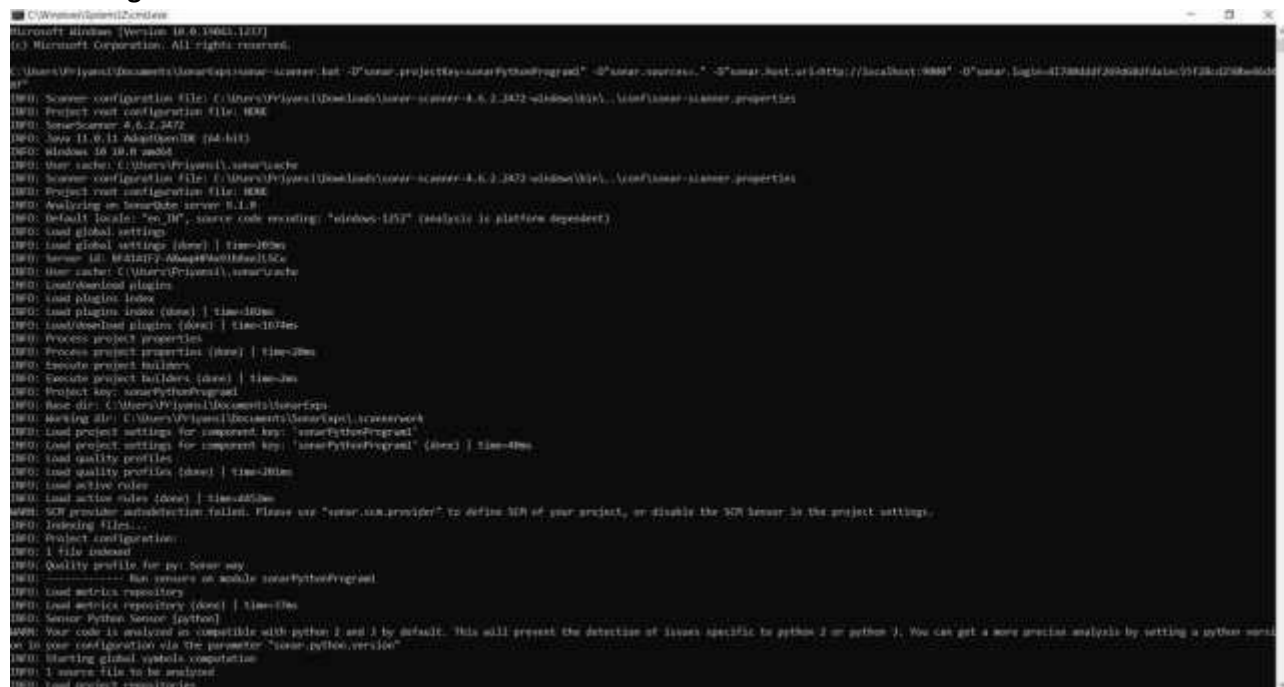
11) Open command prompt in this folder and Run program using copied command.

```
"sonar-scanner.bat -D"sonar.projectKey=sonarPythonProgram1"
```

```
D"sonar.sources=" -
```

```
D"sonar.host.url=http://localhost:9000" -
```

```
D"sonar.login=41740dddf269d68dfda1ec55f28cd250be46d48f"
```



```
C:\ProgramData\Sonar\bin>sonar-scanner.bat -D"sonar.projectKey=sonarPythonProgram1" -D"sonar.sources=" -D"sonar.host.url=http://localhost:9000" -D"sonar.login=41740dddf269d68dfda1ec55f28cd250be46d48f"
INFO: Scanner configuration file: C:\Users\Prjyansh\Downloads\sonar-scanner-4.6.2.3472-windows\bin\..\conf\sonar-scanner.properties
INFO: Project root configuration file: none
INFO: SonarScanner 4.6.2.3472
INFO: Java 11.0.11 AdoptOpenJDK (64-bit)
INFO: Windows 10 10.0 build
INFO: User cache: C:\Users\Prjyansh\AppData\Local
INFO: Scanner configuration file: C:\Users\Prjyansh\Downloads\sonar-scanner-4.6.2.3472-windows\bin\..\conf\sonar-scanner.properties
INFO: Project root configuration file: none
INFO: Analyzing on SonarQube server 6.11.0
INFO: default locale: "en_US", source code encoding: "windows-1252" (analysis is platform dependent)
INFO: Load global settings
INFO: Load global settings (done) | time=10ms
INFO: Sonar ID: 8421277-46a4a44a0bba152c
INFO: User cache: C:\Users\Prjyansh\AppData\Local
INFO: Load/Download plugins
INFO: Load plugins index
INFO: Load plugins index (done) | time=10ms
INFO: Load/Download plugins (done) | time=107ms
INFO: Process project properties
INFO: Process project properties (done) | time=2ms
INFO: Exclude project builders (done) | time=2ms
INFO: Project key: sonarPythonProgram1
INFO: Base dir: C:\Users\Prjyansh\Documents\sonarExps
INFO: Working dir: C:\Users\Prjyansh\Documents\sonarExps\scannerwork
INFO: Load project settings for component key: 'sonarPythonProgram1'
INFO: Load project settings for component key: 'sonarPythonProgram1' (done) | time=4ms
INFO: Load quality profiles
INFO: Load quality profiles (done) | time=10ms
INFO: Load active rules
INFO: Load active rules (done) | time=45ms
WARN: XN provider initialization failed. Please use "sonar.xn.provider" to define XN of your project, or disable the XN sensor in the project settings.
INFO: Loading files...
INFO: Project configuration:
INFO: 1 file indexed
INFO: Quality profile for py: Sonar way.
INFO: ----- Run sensors on module sonarPythonProgram1
INFO: Load metrics repository
INFO: Load metrics repository (done) | time=0ms
INFO: Sonar Python Sensor (python)
WARN: Your code is analyzed as compatible with python 2 and 3 by default. This will prevent the detection of issues specific to python 2 or python 3. You can get a more precise analysis by setting a python version in your configuration via the parameter "sonar.python.version"
INFO: Starting global symbols computation
INFO: 1 source file to be analyzed
INFO: Load project repositories
```

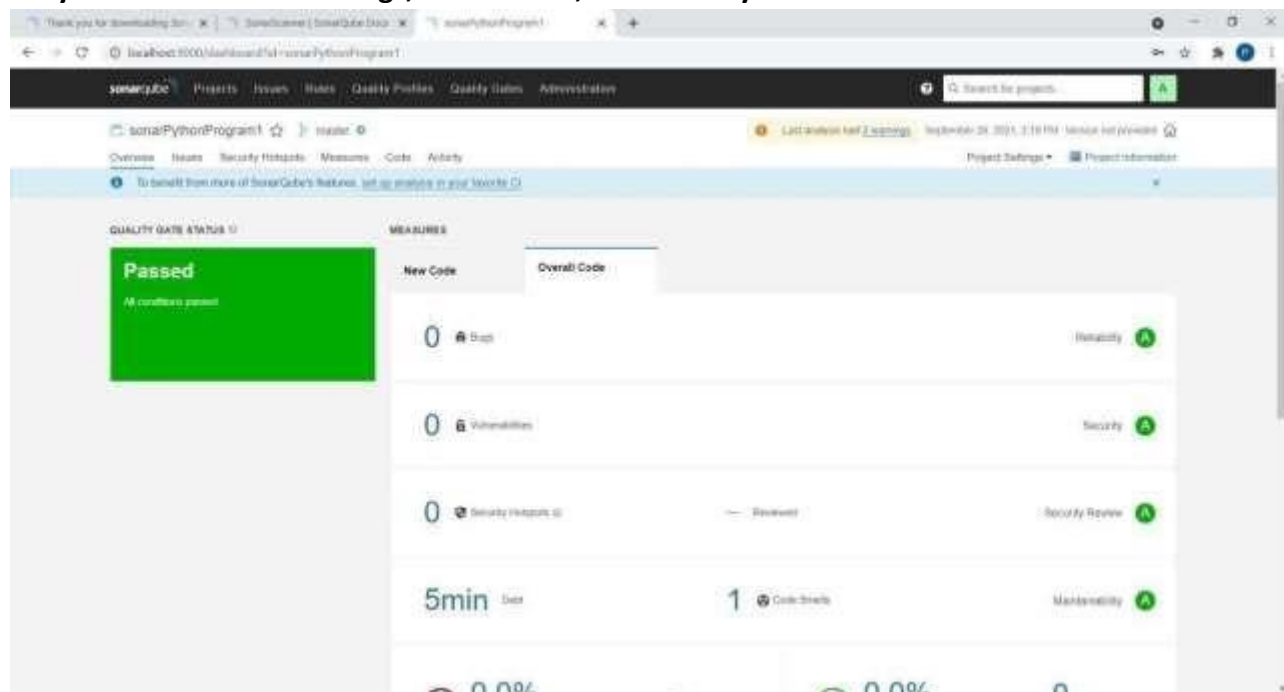


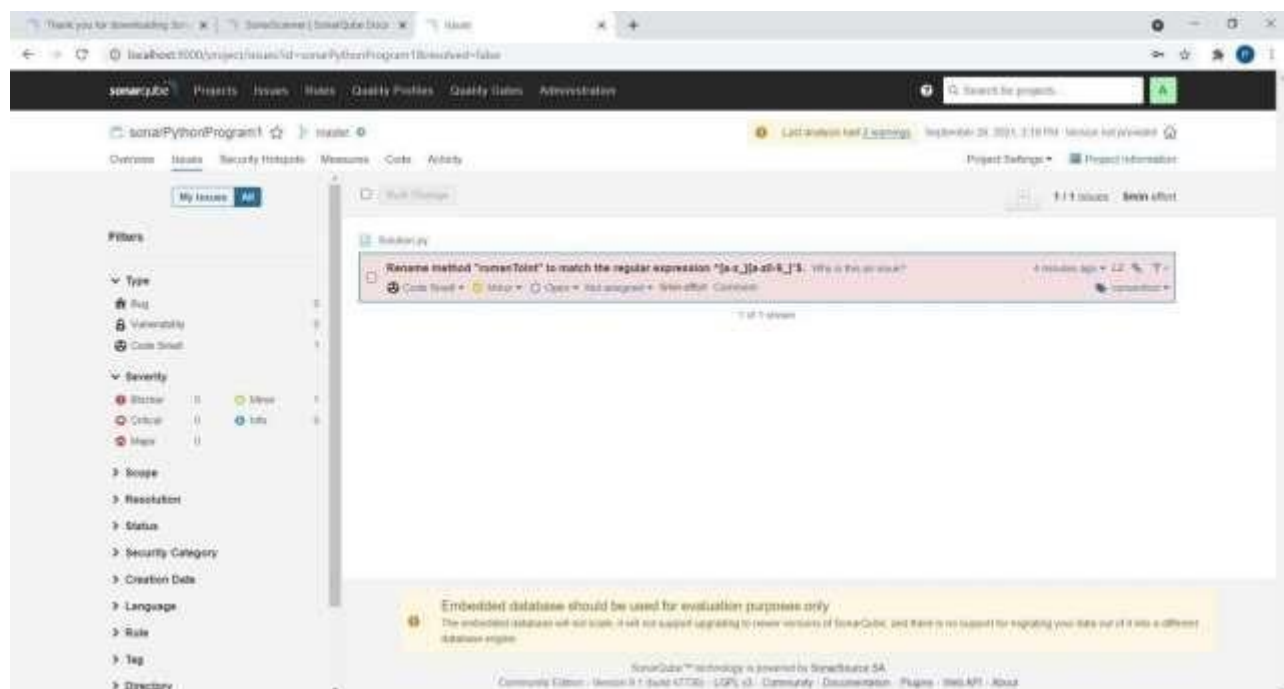
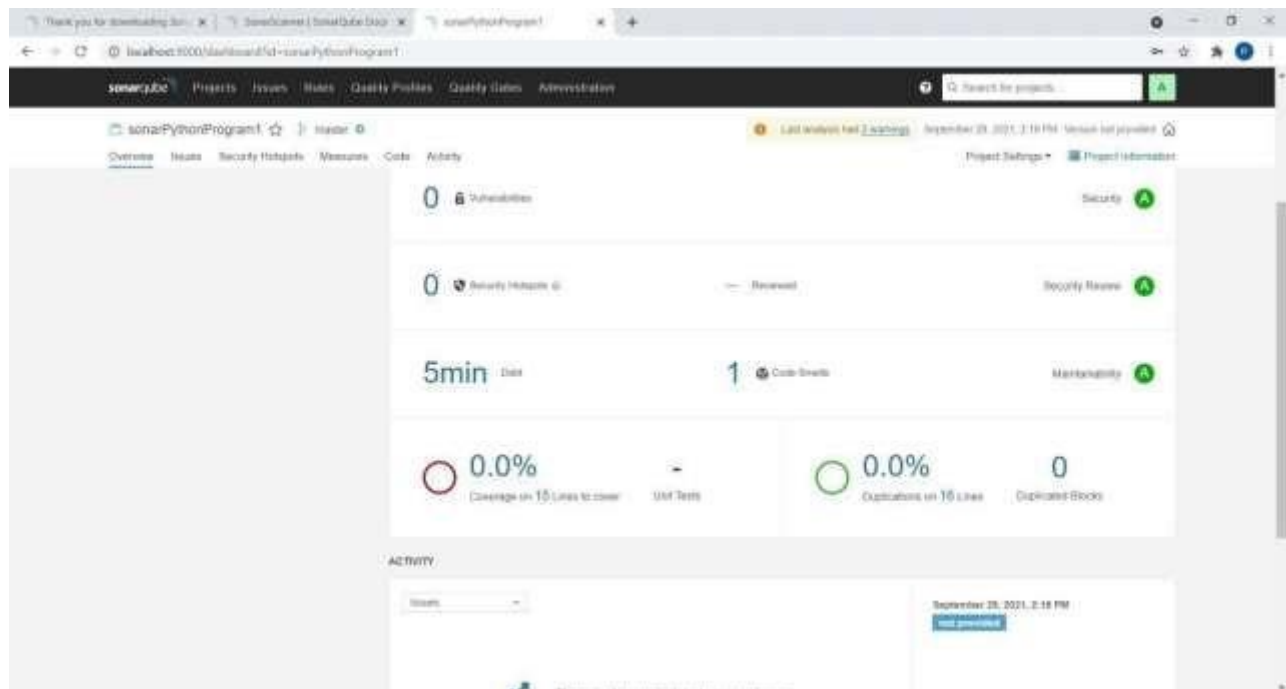
```

C:\Windows\system32\cmd.exe
INFO: Sensor DBE [web] (done) | time=2m
INFO: Sensor VB.NET Project Type Information [web] (done) | time=1m
INFO: Sensor VB.NET Project Type Information [web] (done) | time=1m
INFO: Sensor VB.NET Analysis Log [web] (done) | time=12m
INFO: Sensor VB.NET Properties [web]
INFO: Sensor VB.NET Properties [web] (done) | time=4m
INFO: ..... Run sensors on project:
INFO: Sensor Data Coverage Sensor ..... | time=12m
INFO: Sensor Zero Coverage Sensor [DBE] | time=12m
INFO: SOT Publisher: No SOT system was detected. You can use the 'solar.sot.provider' property to explicitly specify it.
INFO: QD Executor Calculating QD for 1 file
INFO: QD Executor QD calculation finished (done) | time=12m
INFO: Analysis report provided in 35m, file size=103.5 KB
INFO: Analysis report compressed in 15m, file size=14.7 KB
INFO: Analysis report uploaded in 7m
INFO: SUCCESS! SUCCESS! you can browse http://localhost:8080/dashboard?id=solarPythonProgram1
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: Here about the report processing at http://localhost:8080/api/en/task?id=66b6b73fa7125ac21301
INFO: Analysis total time: 7.502 s
INFO: .....
INFO: EXECUTION SUCCESS
INFO: .....
INFO: Total time: 18.687s
INFO: Final memory: 70/500
INFO: .....
C:\Users\T23\Documents\SolarExplo

```

13) Given below is the inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.





Press “Ctrl + C” to stop the server.

[illegible]

Conclusion :-

The SonarQube SAST process is a powerful tool for performing static analysis on Python programs. It can help you to identify and fix security vulnerabilities and code quality issues, improving the overall security and quality of your software.