Static Code Analysis Exercise

For this exercise, you will run SonarQube to analyze your Java project code and SonarScanner to analyze your TypeScript code. There is an individual deliverable due soon which you should try to finish by end of class and another for Sprint 4.

With the generated reports, you will put together a plan to address issues that were flagged in your Java or TypeScript code.

Note: Make SonarQube is **running**, which you should have verified in the setup document.

Analyzing code with SonarQube

We will use SonarQube to analyze our Java code

1. Go to your project directory **where you would run maven** to build and test your code and enter the following placeholder incomplete command but **don't run it yet**:

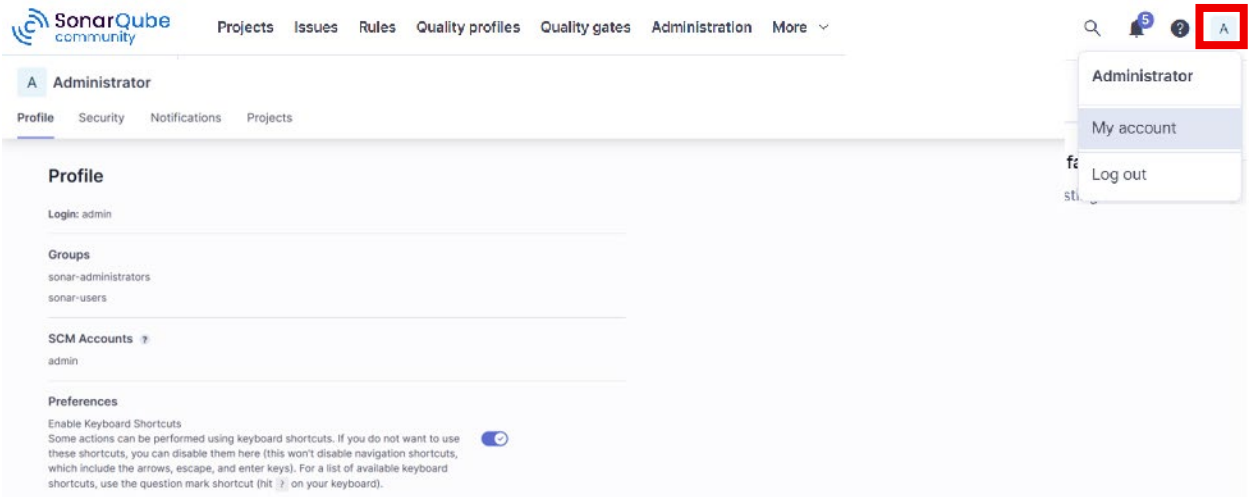
```
mvn clean test sonar:sonar -D sonar.host.url=http://localhost:9000 -D sonar.token=[token will go here]
```

NOTE: in some literature you may see the DEPRECATED DO NOT USE:

```
mvn clean test sonar:sonar -D sonar.login=admin -D sonar.password=admin password
```

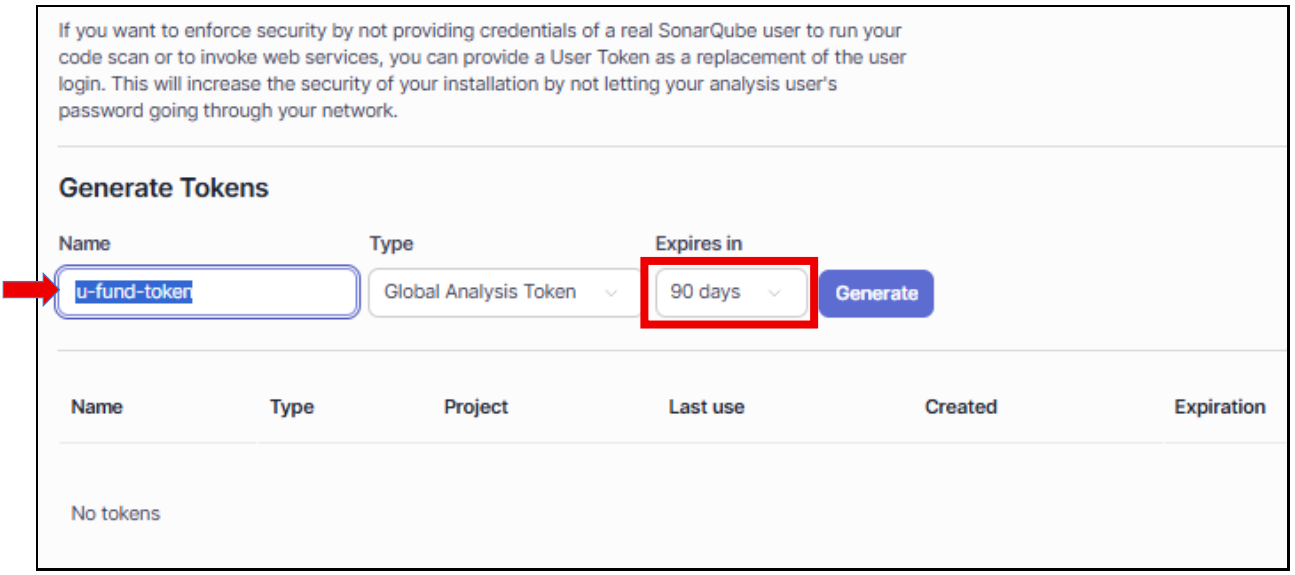
2. Before running you first need to MAKE a token by going to http://localhost:9000

3. Click on the little **A** in the top right, and click the "**My account**" option
4. That should bring you to this page:

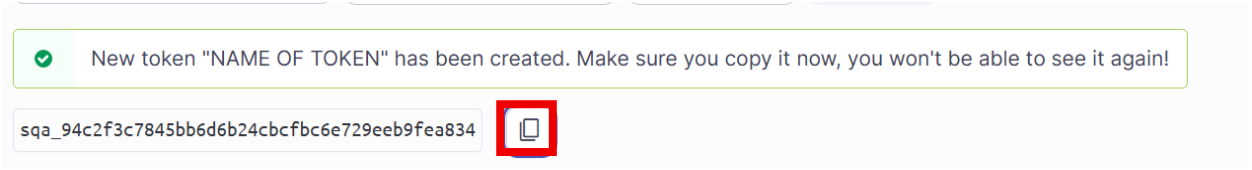


5. **NOTE**: If your browser window is small or your resolution is not high, the left menu might **only** be showing up as icons. Hover the left border to uncover the panel and then click on the "Security" option. The resulting page initially will likely state "No tokens" on the list

6. Fill in the token parameters in the Generate Tokens section. Select "Global" type of token. Then click on "Generate" as shown below:



7. This should produce a new token as such:



8. **COPY THE TOKEN** as you otherwise won't be able to recover it in the future
9. **Return to your terminal** and enter **YOUR TOKEN** at the end of the command as shown below and **run it!**

```
mvn clean test sonar:sonar -D sonar.host.url=http://localhost:9000 -D
sonar.token=sqa_014695a9e31bf45da13287fd5c10d73
```

10. This will attempt to execute all unit tests followed by SonarQube. If you see a **BUILD SUCCESS** message you can safely skip to the next step. Otherwise, you might see something like this **BUILD FAILURE** and need to fix the error(s):

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.406 s
[INFO] Finished at: 2026-11-01T15:52:05-05:00
[INFO] ------------------------------------------------------------------------
[ERROR] No plugin found for prefix 'sonar' in the current project and in the plugin groups [org.springframework.boot, org.jacoco, org.cyclonedx, org.flywaydb, io.
github.git-commit-id, org.jooq, org.jetbrains.kotlin, org.liquibase, org.graalvm.buildtools, org.apache.maven.plugins, org.codehaus.mojo] available from the repos
itories [local (C:\Users\kbmvse\.m2\repository), central (https://repo.maven.apache.org/maven2)] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/NoPluginFoundForPrefixException
```

Make sure you are in the directory that contains your POM file. If the problem persists and coincides with the above error, it indicates that your project configuration (pom.xml) still requires additional modification to recognize 'sonar'. **NOTE: You need to coordinate with your teammates to ensure this change is ONLY DONE ONCE** to avoid conflicts. **Select a team representative and have them modify and push the change** so everyone can then **pull** before continuing.

To fix the error have the selected teammate:

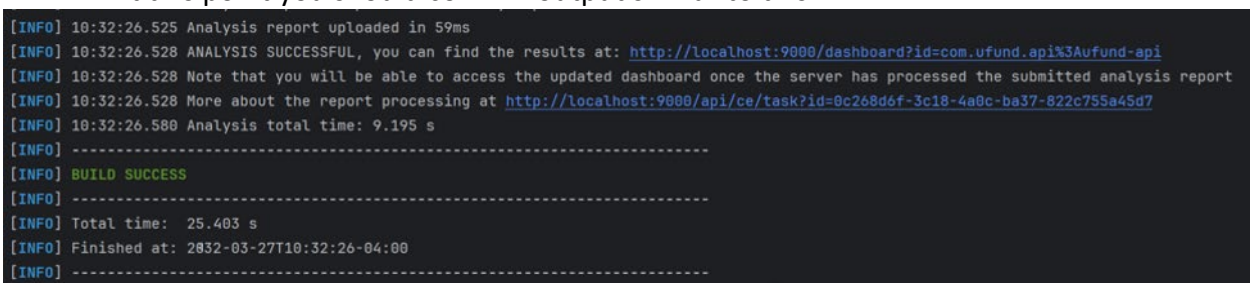
a. Open the pom.xml
b. Search for "plugin**s**" section

c. Add the following config, right below the **<plugins>** section starter marker:

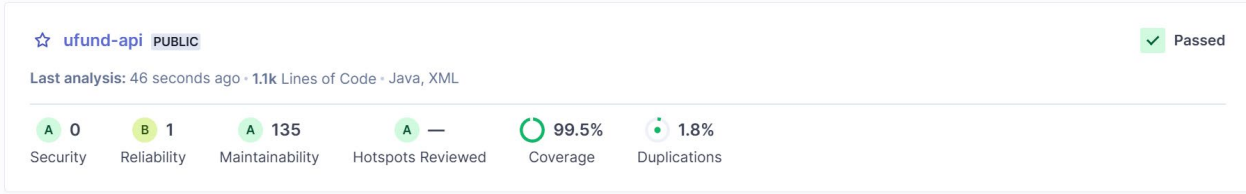
```
<plugin>
   <groupId>org.sonarsource.scanner.maven</groupId>
   <artifactId>sonar-maven-plugin</artifactId>
   <version>5.8.0.7211</version>
</plugin>
```

d. Re-run your command

11. At this point you should confirm output similar to this:



12. Go back to http://localhost:9000 and you should see your SonarQube report for project called **ufund-api**



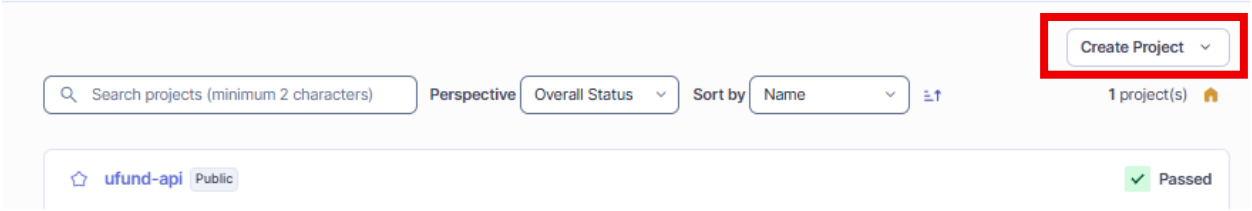
13. It might take a while to analyze it the **first time**, **so be patient** if docker doesn't show you the project immediately

Once successful with the above continue with the next steps

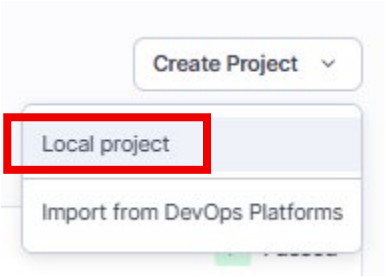
Running SonarScanner

Next, we will run SonarScanner to analyze our TypeScript code

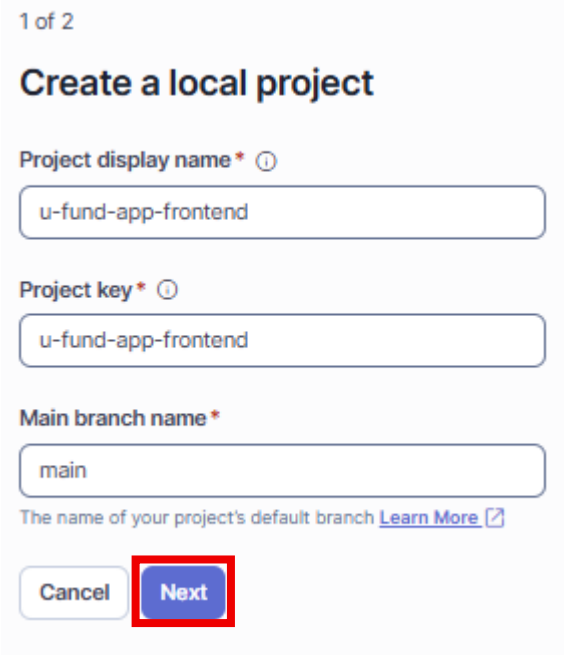
1. In SonarQube, click the “Create Project” button in the upper right corner



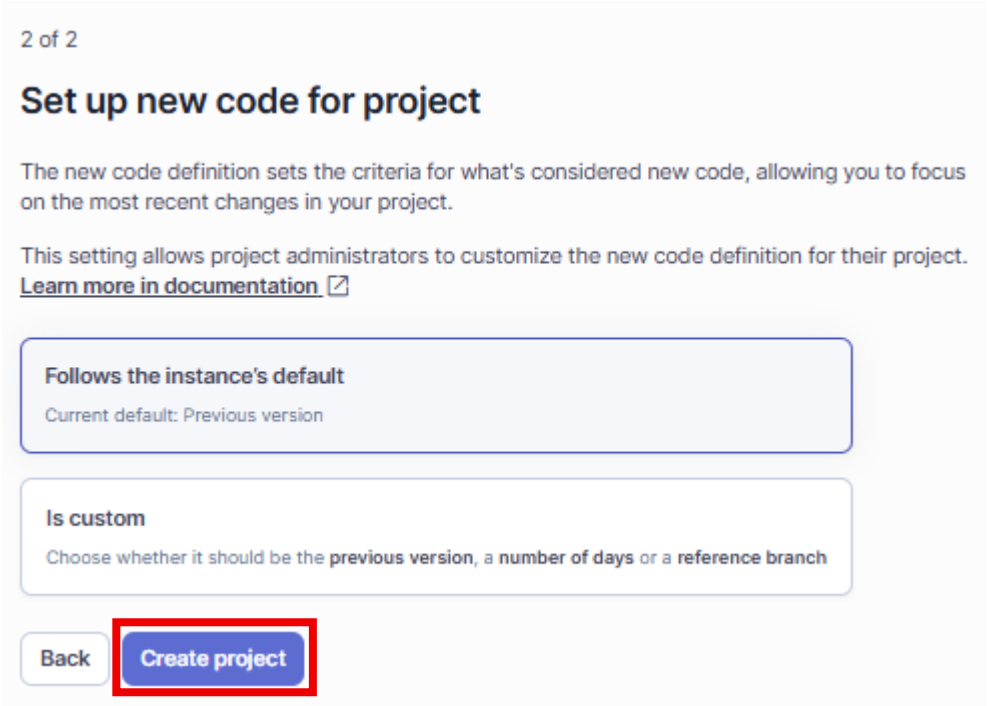
and select “Local project”



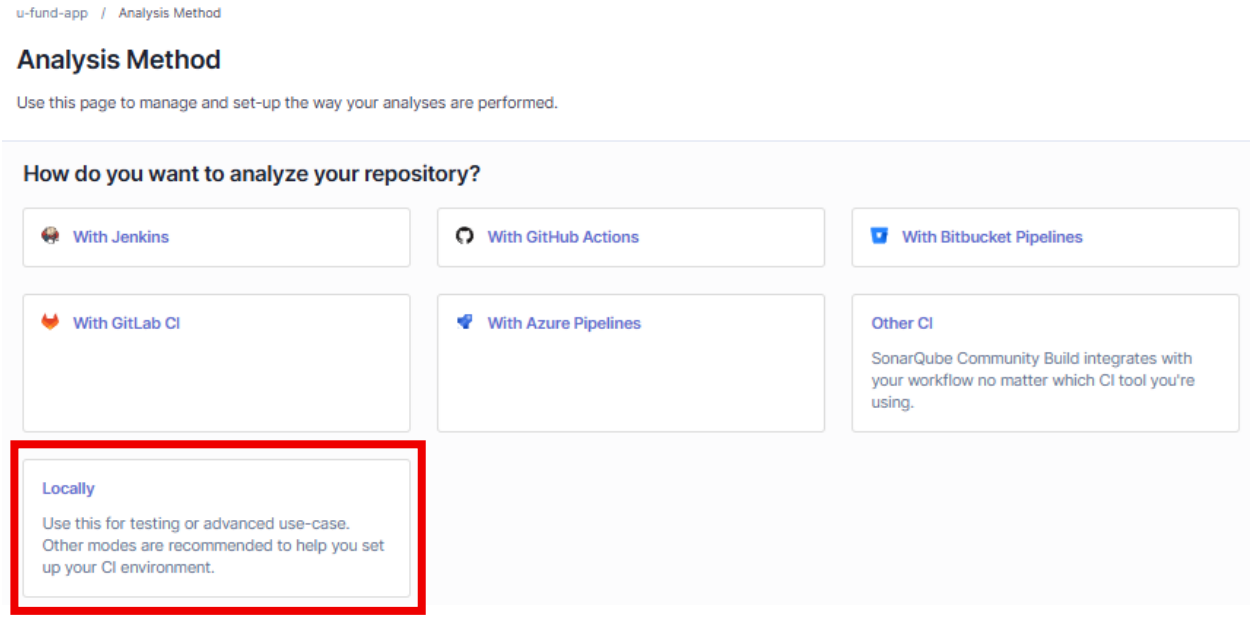
2. Enter a name for your project and click “Next”



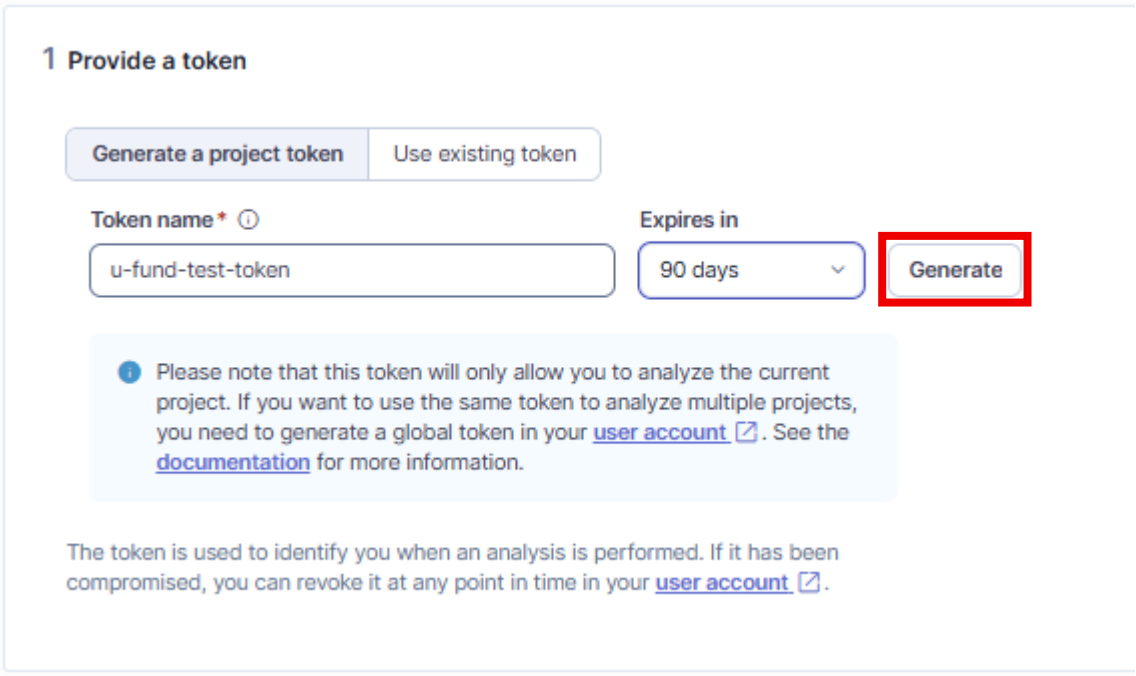
14. Select “Follows the instance’s default” option and then “Create project”



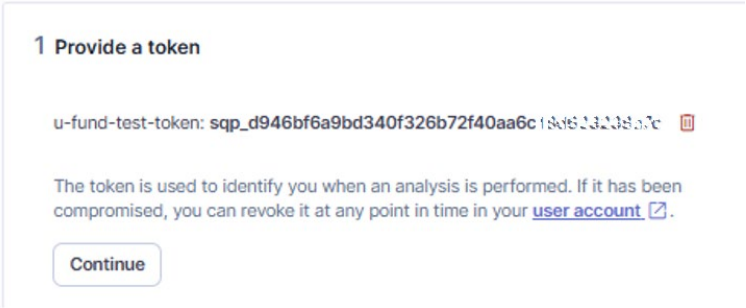
15. Select and click “Locally” as the option to analyze



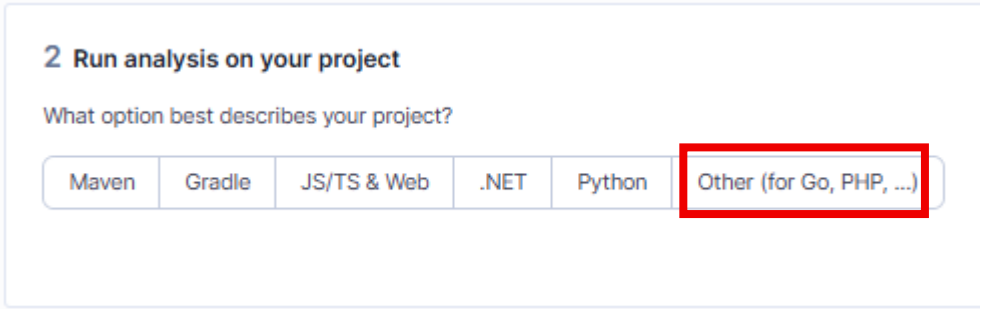
16. For **#1 Provide a Token**, Enter any name for your token and click “Generate”



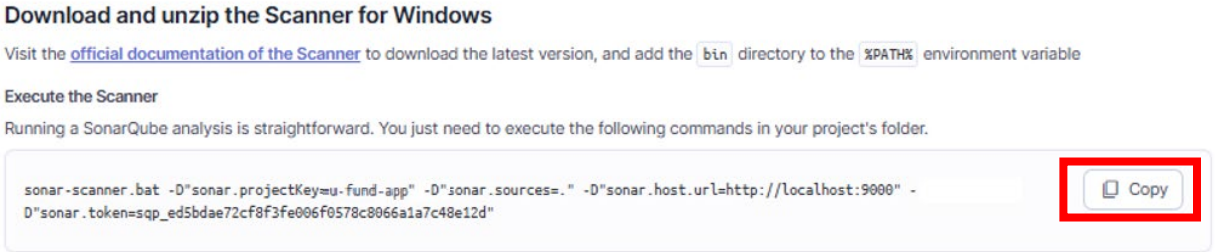
17. Your token will have been created. Click “Continue”



18. For **#2 Run analysis on your project**, select Other (for JS, TS, Go, Python, PHP, ...)



19. Select **your OS**
20. You should see something similar to below:



Or for Linux/macOS:

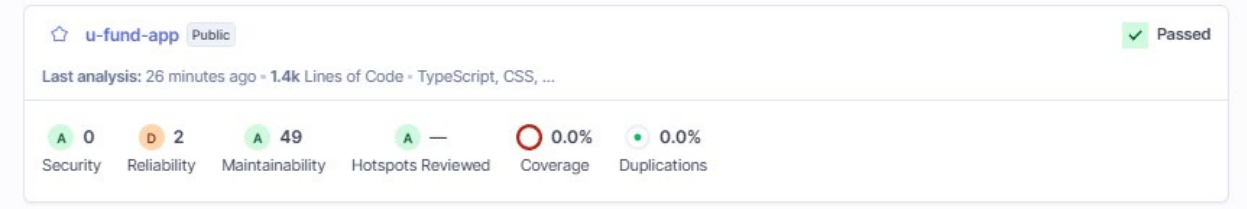


21. Copy the command and go to the directory where your TypeScript code resides (e.g. **u-fund-ui**) and paste into your shell and execute

22. You should see something similar to below:

```
20:51:37.160 INFO  Sensor Zero Coverage Sensor (done) | time=30ms
20:51:37.160 INFO  ------------ Gather SCA dependencies on project
20:51:37.164 INFO  Dependency analysis skipped
20:51:37.169 INFO  SCM Publisher SCM provider for this project is: git
20:51:37.170 INFO  SCM Publisher 44 source files to be analyzed
20:51:39.999 INFO  SCM Publisher 44/44 source files have been analyzed (done) | time=2829ms
20:51:40.006 INFO  CPD Executor 7 files had no CPD blocks
20:51:40.006 INFO  CPD Executor Calculating CPD for 29 files
20:51:40.036 INFO  CPD Executor CPD calculation finished (done) | time=29ms
20:51:40.048 INFO  SCM revision ID '2e18366d5caff904c0950d20ad941d0cab87c2be'
20:51:40.297 INFO  Analysis report generated in 207ms, dir size=415.3 kB
20:51:40.619 INFO  Analysis report compressed in 320ms, zip size=156.0 kB
20:51:40.740 INFO  Analysis report uploaded in 121ms
20:51:40.743 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=directly-with-new-token
20:51:40.743 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
20:51:40.744 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=db2a2751-1fe5-4877-bf0f-ad2ac7b95579
20:51:41.209 INFO  Analysis total time: 17.168 s
20:51:41.211 INFO  SonarScanner Engine completed successfully
20:51:41.810 INFO  EXECUTION SUCCESS
20:51:41.811 INFO  Total time: 26.507s
```

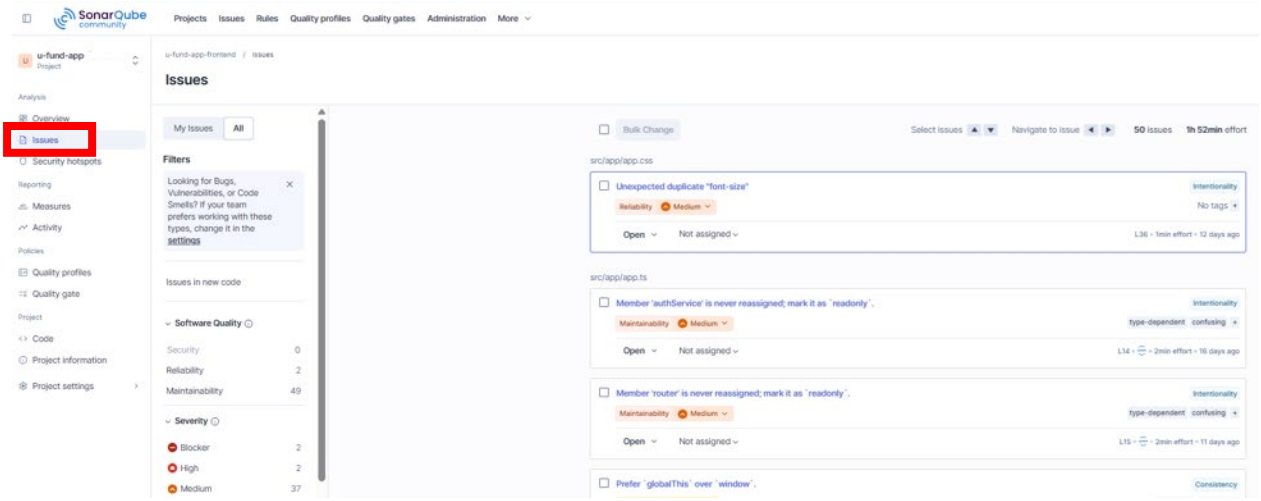
23. Go back to the SonarQube web page and you should see it listed with your previous project results.



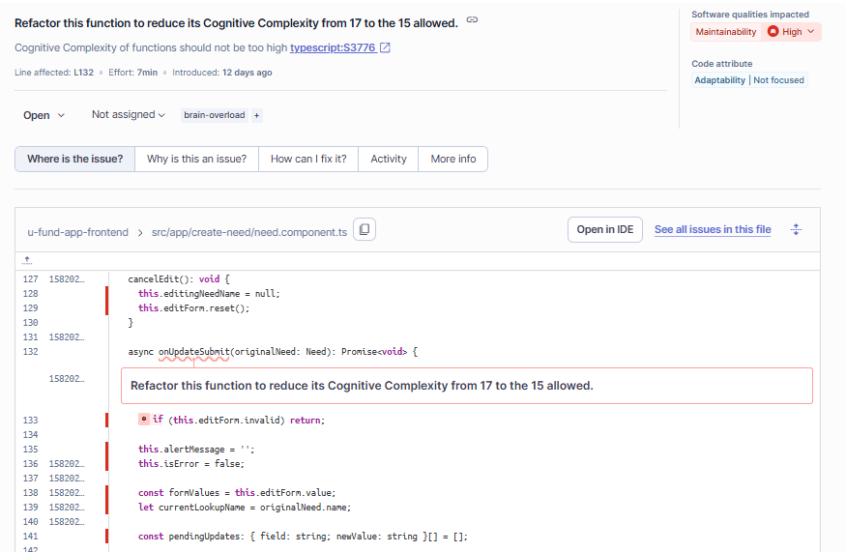
Take a screen shot of **all the projects** (similar to above) and deposit it in the ***Static Code Analysis - individual*** in the myCourses Assignments by the date shown on your section's schedule.

Explore and Analyze Your Reports

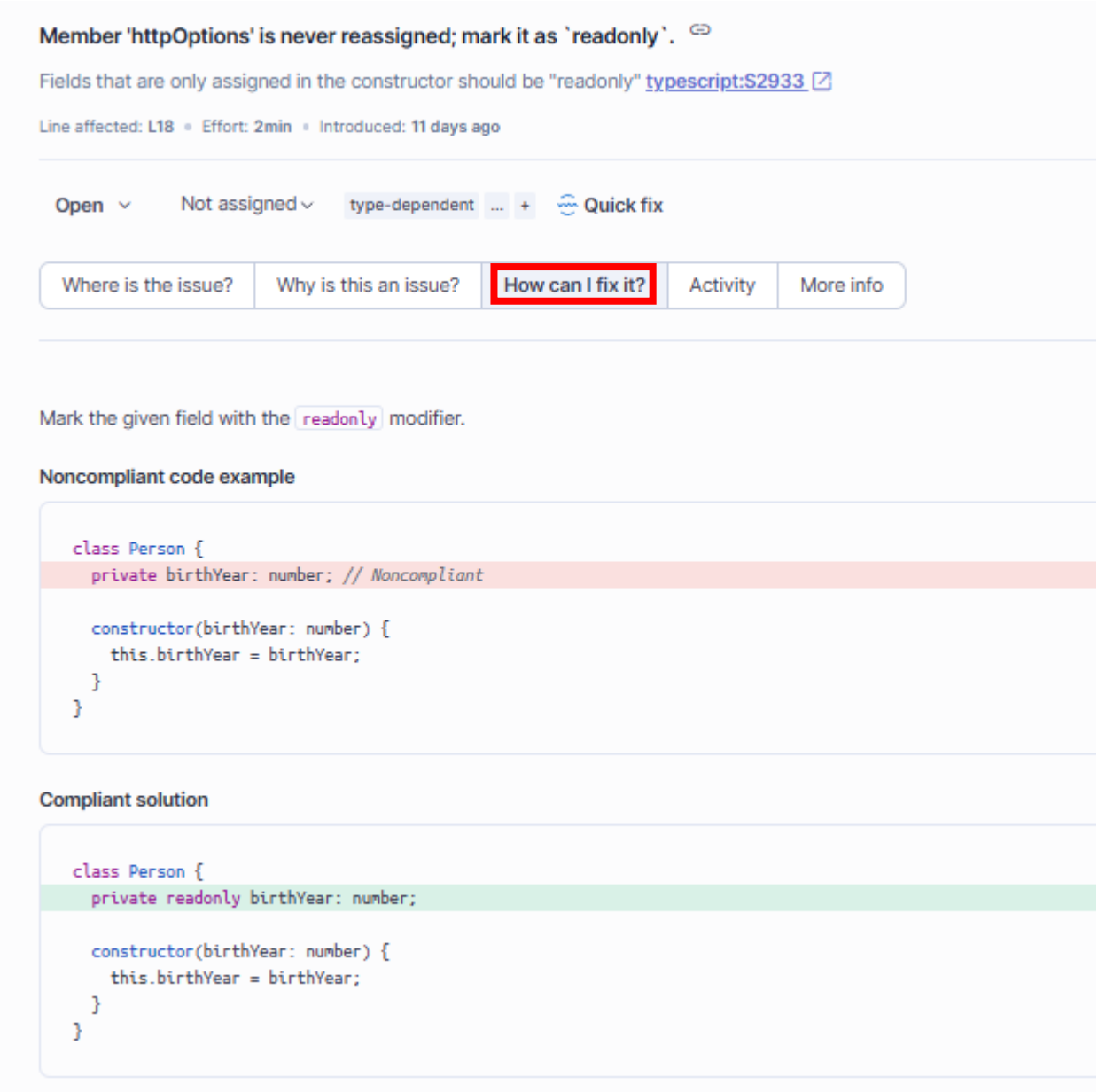
Every report will be different based on what the code analysis identified. Clicking on a project of interest and then select “Issues” on the left-hand menu. This will list all the issues on the selected project, which can then be categorized by severity (e.g. major, critical, blocker) and type (e.g. bug, vulnerability).



Depending on the complexity of your code, certain metrics like **Cognitive Complexity** might get triggered, requiring attention for potentially refactoring:



Other “code smells” may get flagged due to issues with readability or other factors. This may indicate a problem or possibly a false flag. If a particular issue is not clear, click the “Why is this an issue?”, which will provide a description of the issue or even “How can I fix it?” which can provide code examples of a non-compliant issue and a solution.



(plan for future) Project Sprint 4 Deliverable

Explore the various Issue types and identify 3-4 areas within your code that have been flagged by SonarQube and provide your analysis and recommendations. **Your instructor** may require you to deposit this as a preliminary work into your team's ***Static Code Analysis - team*** in the myCourses Assignments. Include any relevant screenshot(s) with each area. This will serve as a baseline and reminder of future work. Be sure to include at least one from both reports (Java and TypeScript). It will become part of your final design documentation in your **Sprint 4** submission.

Stopping SonarQube

Refer to the setup document to stop SonarQube.