Static testing tools and white box test design

Frontend

To do a test analysis of our frontend code we ran the jest testing tool. It showed us that 83% of our source code has been tested and what folders had most untested code. 18/27 folders where fully tested and only 3 of the folders had less than 50% code coverage.

During the process developing and testing the code as the application developed, jest testing tool helped to find the areas where we were lacking in tests. We then managed to get a high test coverage and let some part of our code have a low coverage.

All files

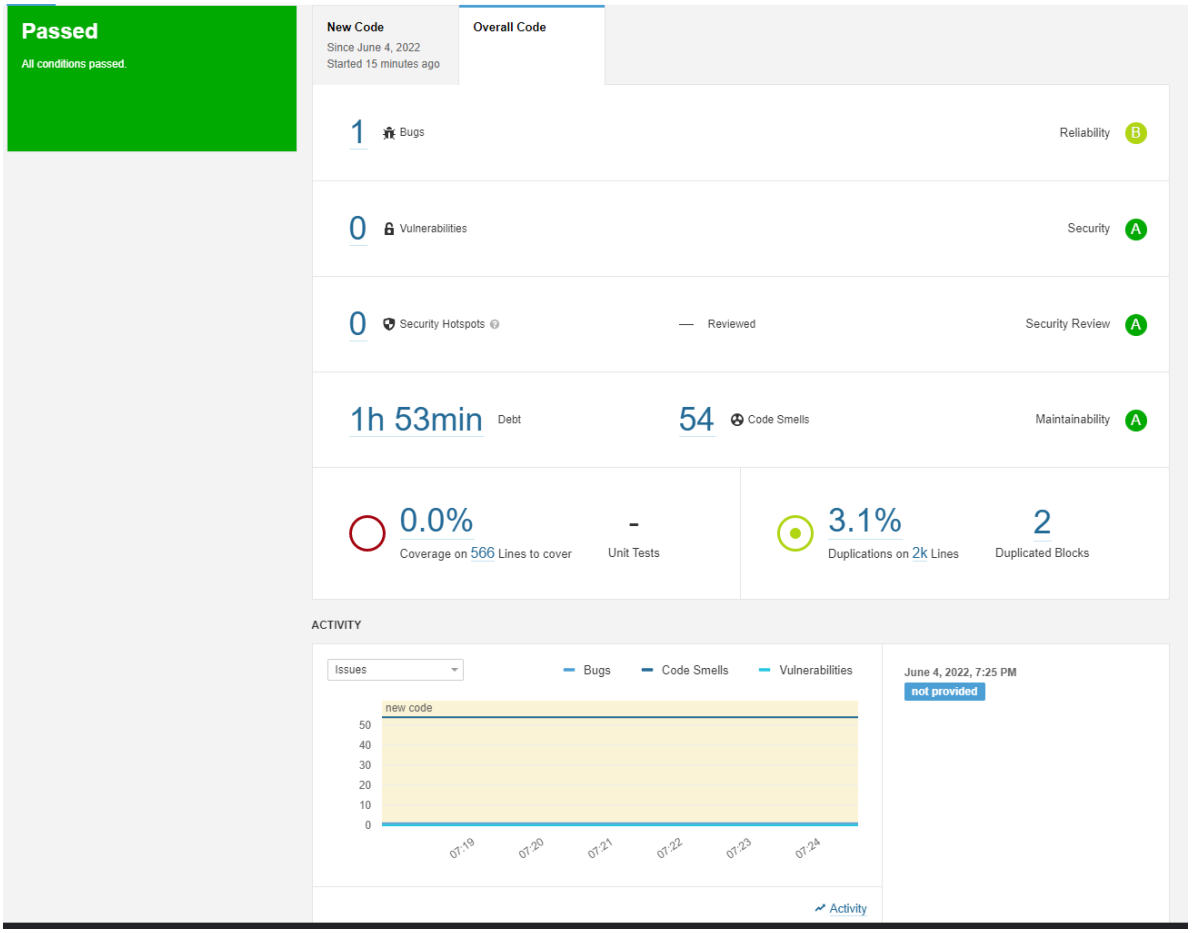
80.31% Statements 151/188 **64.58%** Branches 31/48 **81.6%** Functions 71/87 **79.67%** Lines 145/182

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

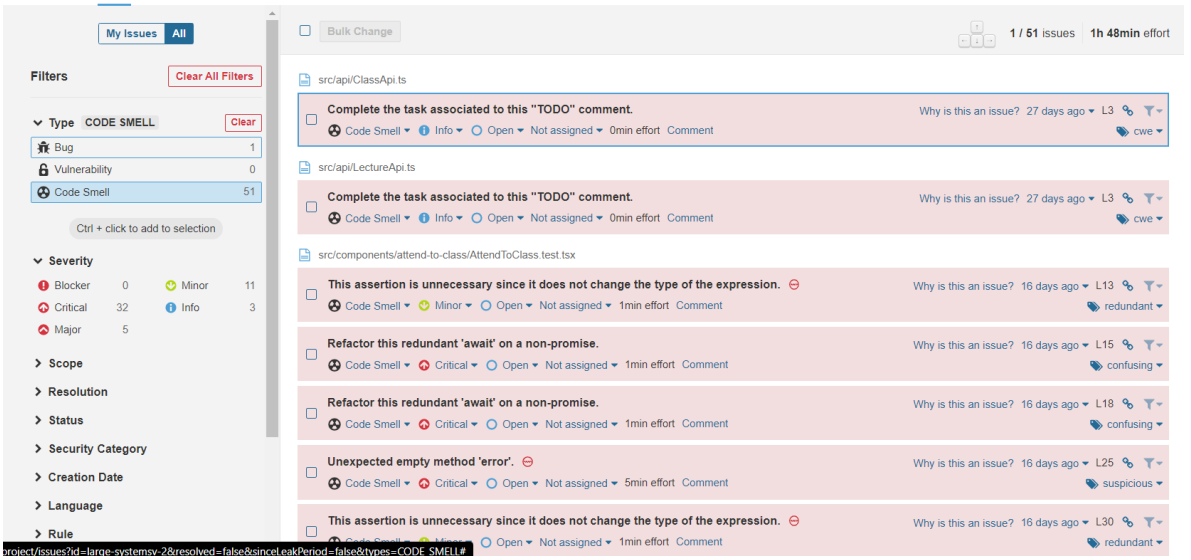
Filter:

File	Statements		Branches		Functions		Lines	
src	83.33%	25/30	20%	1/5	100%	10/10	83.33%	25/30
src/api	95.23%	40/42	50%	3/6	90.9%	20/22	94.59%	35/37
src/components/attend-to-class	100%	8/8	100%	0/0	100%	4/4	100%	8/8
src/components/attend-to-class/attend-to-class-form	100%	2/2	100%	0/0	100%	1/1	100%	2/2
src/components/loading-table	100%	4/4	100%	2/2	100%	3/3	100%	3/3
src/components/logout-button	100%	1/1	100%	0/0	100%	1/1	100%	1/1
src/components/main-card	100%	6/6	100%	2/2	100%	3/3	100%	6/6
src/components/primary-button	100%	1/1	100%	3/3	50%	1/2	100%	1/1
src/components/search-input	100%	1/1	100%	0/0	100%	1/1	100%	1/1
src/components/subject-table	83.33%	5/6	100%	6/6	75%	3/4	83.33%	5/6
src/components/table-row	100%	4/4	100%	1/1	75%	3/4	100%	4/4
src/hooks	0%	0/13	0%	0/4	0%	0/3	0%	0/13
src/layout/attendance-page	100%	3/3	100%	0/0	100%	1/1	100%	3/3
src/layout/attendance-page/attendanceView	60%	6/10	66.66%	4/6	33.33%	1/3	60%	6/10
src/layout/attendance-page/attendanceView/AttendanceCode	100%	2/2	100%	0/0	100%	1/1	100%	2/2
src/layout/central-layout	100%	1/1	100%	0/0	100%	1/1	100%	1/1
src/layout/lecture-code-accepted-modal	100%	1/1	100%	0/0	100%	1/1	100%	1/1
src/layout/login-page	50%	1/2	100%	0/0	50%	1/2	50%	1/2
src/layout/login-page/login-success	10%	1/10	0%	0/4	0%	0/2	10%	1/10
src/layout/navigation-bar/authorized-navigation-bar	100%	5/5	100%	3/3	100%	1/1	100%	5/5
src/layout/navigation-bar/default-navigation-bar	100%	1/1	100%	0/0	100%	1/1	100%	1/1
src/layout/navigation-bar/navigation-bar-link	100%	1/1	100%	0/0	100%	1/1	100%	1/1
src/layout/navigation-bar/roll-call-navigation-bar	100%	1/1	100%	0/0	100%	1/1	100%	1/1
src/layout/navigation-bar/student-navigation-bar	75%	3/4	100%	0/0	50%	1/2	75%	3/4
src/layout/navigation-bar/teacher-navigation-bar	75%	3/4	100%	0/0	50%	1/2	75%	3/4

To further analyze the code, we used Sonarqube. This provided us with information about the source codes code quality and helped us find vulnerabilities during development. As seen in the image bellow, we had 1 bug and 0 vulnerabilities last time we ran the code analysis.



We used another useful feature of Sonarqube which shows us our codebase's *smelly code*. Smelly code is another way of saying "code that needs refactoring", aka. needs optimizing. We found that we had 51 smelly code parts in our codebase, but most of them were found in the tests, making them lightly less critical as they don't pose an immediate problem to the system.



Backend

When testing our backend application we used jest's testing tool once again. On the backend our code coverage was higher with only one folder missing some testing. The folder missing test coverage is the authentication folder, as seen below. This was because our system uses Microsofts service to login, which is an external API. The authentication is tested through e2e testing in our system, though it doesn't show up here. In other words, everything is tested as it is supposed to.

All files

92.61% Statements 502/542 **50.63%** Branches 40/79 **92.62%** Functions 113/122 **92.04%** Lines 451/490

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
config	100%	4/4	100%	2/2	100%	0/0	100%	4/4
models	100%	71/71	100%	0/0	100%	15/15	100%	59/59
routes	100%	234/234	100%	3/3	100%	66/66	100%	204/204
services	100%	93/93	100%	20/20	100%	15/15	100%	92/92
utils	93.61%	88/94	93.75%	15/16	100%	16/16	93.25%	83/89
authentication	26.08%	12/46	0%	0/38	10%	1/10	21.42%	9/42

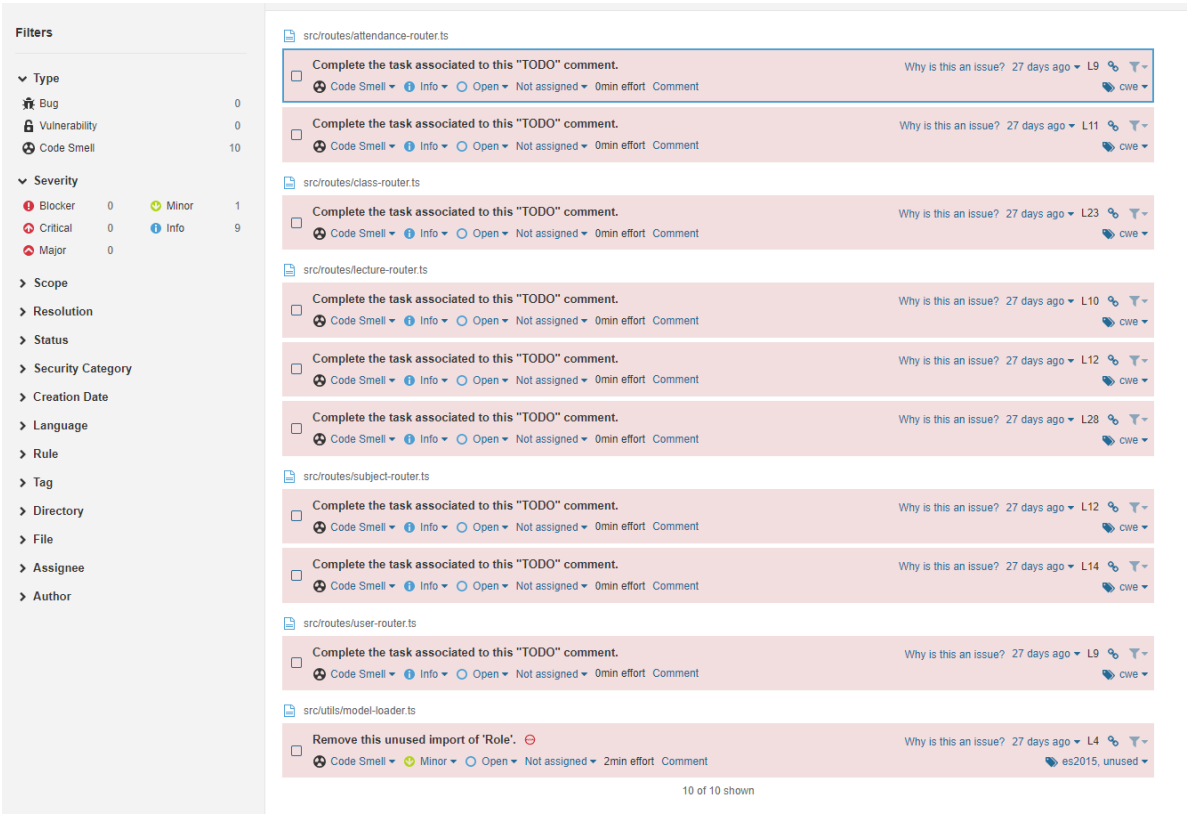
File	Statements		Branches		Functions		Lines	
generic-service-initializer.ts	100%	13/13	100%	0/0	100%	0/0	100%	13/13
input-validators.ts	100%	14/14	100%	8/8	100%	2/2	100%	13/13
model-loader.ts	89.65%	52/58	66.66%	2/3	100%	12/12	89.09%	49/55
response-handler.ts	100%	9/9	100%	5/5	100%	2/2	100%	8/8

File	Statements		Branches		Functions		Lines	
class-code-service.ts	100%	47/47	100%	13/13	100%	7/7	100%	46/46
model-service.ts	100%	37/37	100%	6/6	100%	6/6	100%	37/37
subject-service.ts	100%	9/9	100%	1/1	100%	2/2	100%	9/9

File	Statements		Branches		Functions		Lines	
attendance-router.ts	100%	37/37	100%	0/0	100%	11/11	100%	32/32
class-code-router.ts	100%	23/23	100%	2/2	100%	5/5	100%	21/21
class-router.ts	100%	37/37	100%	0/0	100%	11/11	100%	32/32
lecture-router.ts	100%	40/40	100%	1/1	100%	11/11	100%	35/35
role-router.ts	100%	16/16	100%	0/0	100%	4/4	100%	14/14
subject-router.ts	100%	44/44	100%	0/0	100%	13/13	100%	38/38
user-router.ts	100%	37/37	100%	0/0	100%	11/11	100%	32/32

File	Statements		Branches		Functions		Lines	
attendances.ts	100%	13/13	100%	0/0	100%	3/3	100%	11/11
classes.ts	100%	9/9	100%	0/0	100%	2/2	100%	7/7
lectures.ts	100%	14/14	100%	0/0	100%	4/4	100%	12/12
roles.ts	100%	9/9	100%	0/0	100%	2/2	100%	7/7
subjects.ts	100%	13/13	100%	0/0	100%	2/2	100%	11/11
users.ts	100%	13/13	100%	0/0	100%	2/2	100%	11/11

When running the sonarqube testing tool on our backend we managed to find and resolve the bugs and vulnerabilities since there were very few. At the end, we only had some *smelly code* which was mainly caused by some TODO's that we still had in the code. In other words, the test analysis of the backend went very smoothly with only few changes required based on the analyses.


Filters
Type
Bug 0
Vulnerability 0
Code Smell 10
Severity
Blocker 0
Minor 1
Critical 0
Info 9
Major 0
Scope
Resolution
Status
Security Category
Creation Date
Language
Rule
Tag
Directory
File
Assignee
Author
src/routes/attendance-router.ts
Complete the task associated to this "TODO" comment.
Why is this an issue? 27 days ago L9
Code Smell Info Open Not assigned 0min effort Comment
cwe
Complete the task associated to this "TODO" comment.
Why is this an issue? 27 days ago L11
Code Smell Info Open Not assigned 0min effort Comment
cwe
src/routes/class-router.ts
Complete the task associated to this "TODO" comment.
Why is this an issue? 27 days ago L23
Code Smell Info Open Not assigned 0min effort Comment
cwe
src/routes/lecture-router.ts
Complete the task associated to this "TODO" comment.
Why is this an issue? 27 days ago L10
Code Smell Info Open Not assigned 0min effort Comment
cwe
Complete the task associated to this "TODO" comment.
Why is this an issue? 27 days ago L12
Code Smell Info Open Not assigned 0min effort Comment
cwe
Complete the task associated to this "TODO" comment.
Why is this an issue? 27 days ago L28
Code Smell Info Open Not assigned 0min effort Comment
cwe
src/routes/subject-router.ts
Complete the task associated to this "TODO" comment.
Why is this an issue? 27 days ago L12
Code Smell Info Open Not assigned 0min effort Comment
cwe
Complete the task associated to this "TODO" comment.
Why is this an issue? 27 days ago L14
Code Smell Info Open Not assigned 0min effort Comment
cwe
src/routes/user-router.ts
Complete the task associated to this "TODO" comment.
Why is this an issue? 27 days ago L9
Code Smell Info Open Not assigned 0min effort Comment
cwe
src/utils/model-loader.ts
Remove this unused import of 'Role'.
Why is this an issue? 27 days ago L4
Code Smell Minor Open Not assigned 2min effort Comment
es2015, unused
10 of 10 shown