|  |
| --- |
| frontend  Version 0.0.1  Code analysis |

|  |
| --- |
| **By: default**  **2024-11-30** |

# Content

[Content 1](#_Toc97156258)

[Introduction 2](#_Toc97156259)

[Configuration 2](#_Toc97156260)

[Synthesis 3](#_Toc97156261)

[Analysis Status 3](#_Toc97156262)

[Quality gate status 3](#_Toc97156263)

[Metrics 3](#_Toc97156264)

[Tests 3](#_Toc97156265)

[Detailed technical debt 3](#_Toc97156266)

[Metrics Range 5](#_Toc97156267)

[Volume 5](#_Toc97156268)

[Issues 6](#_Toc97156269)

[Charts 6](#_Toc97156270)

[Issues count by severity and type 8](#_Toc97156271)

[Issues List 8](#_Toc97156272)

[Security Hotspots 9](#_Toc97156273)

[Security hotspots count by category and priority 9](#_Toc97156274)

[Security hotspots List 9](#_Toc97156275)

# Introduction

This document contains results of the code analysis of frontend.

Here I can add a description of my project

# Configuration

* Quality Profiles
  + Names: Sonar way [CSS]; Sonar way [JavaScript]; Sonar way [HTML];
  + Files: AZNrmB9KxUS2\_b0QqMUt.json; AZNrmCCExUS2\_b0QqMjl.json; AZNrmCOUxUS2\_b0QqNWX.json;
* Quality Gate
  + Name: Sonar way
  + File: Sonar way.xml

# Synthesis

## Analysis Status

|  |  |  |  |
| --- | --- | --- | --- |
| Reliability | Security | Security Review | Maintainability |
| A.png | **A.png** | **A.png** | **A.png** |

## Quality gate status

|  |  |
| --- | --- |
| Quality Gate Status | **OK.png** |

|  |  |
| --- | --- |
| Metric | Value |
| Reliability Rating on New Code | OK |
| Security Rating on New Code | OK |
| Maintainability Rating on New Code | OK |

## Metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coverage | Duplication | Comment  density | Median number of lines of code per file | Adherence to coding standard |
| 0.0 % | **0.0 %** | **3.9 %** | **44.5** | **99.6 %** |

## Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total | Success Rate | Skipped | Errors | Failures |
| 0 | **0 %** | **0** | **0** | **0** |

## Detailed technical debt

|  |  |  |  |
| --- | --- | --- | --- |
| Reliability | Security | Maintainability | Total |
| - | - | 0d 1h 40min | 0d 1h 40min |

## Metrics Range

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Cyclomatic  Complexity | Cognitive  Complexity | Lines of code per file | Comment  density (%) | Coverage | Duplication (%) |
| Min | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| Max | 243.0 | 192.0 | 2735.0 | 52.3 | 0.0 | 0.0 |

## Volume

|  |  |
| --- | --- |
| Language | Number |
| CSS | 477 |
| JavaScript | 2258 |
| HTML | 21 |
| Total | 2756 |

# Issues

## Charts

## Issues count by severity and type

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type / Severity | INFO | MINOR | MAJOR | CRITICAL | BLOCKER |
| BUG | 0 | 0 | 0 | 0 | 0 |
| VULNERABILITY | 0 | 0 | 0 | 0 | 0 |
| CODE\_SMELL | 0 | 4 | 7 | 3 | 0 |

## Issues List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Description | Type | Severity | Number |
| Cognitive Complexity of functions should not be too high | Cognitive Complexity is a measure of how hard the control flow of a function is to understand. Functions with high Cognitive Complexity will be difficult to maintain. See Cognitive Complexity | CODE\_SMELL | CRITICAL | 3 |
| Selectors should not be duplicated | Duplication of selectors might indicate a copy-paste mistake. The rule detects the following kinds of duplications: within a list of selectors in a single rule set for duplicated selectors in different rule sets within a single stylesheet. Noncompliant Code Example .foo, .bar, .foo { ... } /\* Noncompliant \*/ .class1 { ... } .class1 { ... } /\* Noncompliant \*/ Compliant Solution .foo, .bar { ... } .class1 { ... } .class2 { ... } | CODE\_SMELL | MAJOR | 1 |
| Sections of code should not be commented out | Programmers should not comment out code as it bloats programs and reduces readability. Unused code should be deleted and can be retrieved from source control history if required. | CODE\_SMELL | MAJOR | 1 |
| Unused assignments should be removed | A dead store happens when a local variable is assigned a value that is not read by any subsequent instruction. Calculating or retrieving a value only to then overwrite it or throw it away, could indicate a serious error in the code. Even if it’s not an error, it is at best a waste of resources. Therefore all calculated values should be used. Noncompliant Code Example i = a + b; // Noncompliant; calculation result not used before value is overwritten i = compute(); Compliant Solution i = a + b; i += compute(); Exceptions This rule ignores initializations to -1, 0, 1, undefined, [], {}, true, false and "". Variables that start with an underscore (e.g. '\_unused') are ignored. Assignment of null is ignored because it is sometimes used to help garbage collection Increment and decrement expressions are ignored because they are often used idiomatically instead of x+1 This rule also ignores variables declared with object destructuring using rest syntax (used to exclude some properties from object): let {a, b, ...rest} = obj; // 'a' and 'b' are ok doSomething(rest); let [x1, x2, x3] = arr; // but 'x1' is noncompliant, as omitting syntax can be used: "let [, x2, x3] = arr;" doSomething(x2, x3); See MITRE, CWE-563 - Assignment to Variable without Use ('Unused Variable') | CODE\_SMELL | MAJOR | 1 |
| Ternary operators should not be nested |  | CODE\_SMELL | MAJOR | 1 |
| No array index for keys in JSX list components |  | CODE\_SMELL | MAJOR | 3 |
| Unnecessary imports should be removed | There’s no reason to import modules you don’t use; and every reason not to: doing so needlessly increases the load. Noncompliant Code Example import A from 'a'; // Noncompliant, A isn't used import { B1 } from 'b'; console.log(B1); Compliant Solution import { B1 } from 'b'; console.log(B1); | CODE\_SMELL | MINOR | 1 |
| Unused local variables and functions should be removed | If a local variable or a local function is declared but not used, it is dead code and should be removed. Doing so will improve maintainability because developers will not wonder what the variable or function is used for. Noncompliant Code Example function numberOfMinutes(hours) { var seconds = 0; // seconds is never used return hours \* 60; } Compliant Solution function numberOfMinutes(hours) { return hours \* 60; } | CODE\_SMELL | MINOR | 1 |
| "for of" should be used with Iterables | If you have an iterable, such as an array, set, or list, your best option for looping through its values is the for of syntax. Use a counter, and …​ well you’ll get the right behavior, but your code just isn’t as clean or clear. In a browser environment, NodeList and other array-like collections should work by default. If you are using TypeScript and seeing a type error, make sure your configuration is correct. Noncompliant Code Example const arr = [4, 3, 2, 1]; for (let i = 0; i &lt; arr.length; i++) { // Noncompliant console.log(arr[i]); } Compliant Solution const arr = [4, 3, 2, 1]; for (let value of arr) { console.log(value); } | CODE\_SMELL | MINOR | 2 |

# Security Hotspots

## Security hotspots count by category and priority

|  |  |  |  |
| --- | --- | --- | --- |
| Category / Priority | LOW | MEDIUM | HIGH |
| LDAP Injection | 0 | 0 | 0 |
| Object Injection | 0 | 0 | 0 |
| Server-Side Request Forgery (SSRF) | 0 | 0 | 0 |
| XML External Entity (XXE) | 0 | 0 | 0 |
| Insecure Configuration | 0 | 0 | 0 |
| XPath Injection | 0 | 0 | 0 |
| Authentication | 0 | 0 | 0 |
| Weak Cryptography | 0 | 0 | 0 |
| Denial of Service (DoS) | 0 | 0 | 0 |
| Log Injection | 0 | 0 | 0 |
| Cross-Site Request Forgery (CSRF) | 0 | 0 | 0 |
| Open Redirect | 0 | 0 | 0 |
| Permission | 0 | 0 | 0 |
| SQL Injection | 0 | 0 | 0 |
| Encryption of Sensitive Data | 0 | 0 | 0 |
| Traceability | 0 | 0 | 0 |
| Buffer Overflow | 0 | 0 | 0 |
| File Manipulation | 0 | 0 | 0 |
| Code Injection (RCE) | 0 | 0 | 0 |
| Cross-Site Scripting (XSS) | 0 | 0 | 0 |
| Command Injection | 0 | 0 | 0 |
| Path Traversal Injection | 0 | 0 | 0 |
| HTTP Response Splitting | 0 | 0 | 0 |
| Others | 0 | 0 | 0 |

## Security hotspots List