CAFE-FINAL

Code analysis

By: SJIT - Team

2023-05-15

CONTENT

Content 1
Introduction 2
Configuration 2
Synthesis 3
  Analysis Status 3
  Quality gate status 3
  Metrics 3
  Tests 3
  Detailed technical debt 4
  Metrics Range 5
  Volume 5
Issues 6
  Charts 6
  Issues count by severity and type 8
  Issues List 8
Security Hotspots 10
  Security hotspots count by category and priority 10
  Security hotspots List 11

INTRODUCTION

This document contains results of the code analysis of cafe-final.

CONFIGURATION

- Quality Profiles
  - Names: Sonar way [CSS]; Sonar way [TypeScript]; Sonar way [HTML];
  - Files: AYgdr4A4uR6tzCGtCxJy.json; AYgdr43OuR6tzCGtCyo5.json; AYgdr4izuR6tzCGtCyLc.json;
- Quality Gate
  - Name: Sonar way
  - File: Sonar way.xml

SYNTHESIS

ANALYSIS STATUS

Reliability	Security	Security Review	Maintainability
A	A	E	A

QUALITY GATE STATUS

Quality Gate Status	Passed

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 %	2.2 %	24.0	99.6 %

TESTS

Total	Success Rate	Skipped	Errors	Failures
0	0 %	0	0	0

DETAILED TECHNICAL DEBT

Reliability	Security	Maintainability	Total
-	-	0d 0h 35min	**0d 0h 35min**

cafe-final

METRICS RANGE

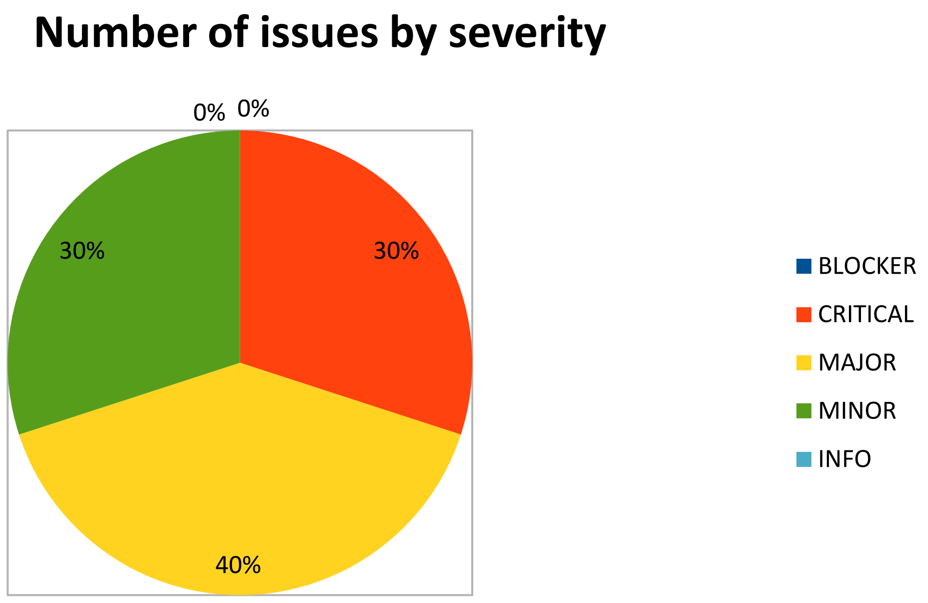
	Cyclomatic Complexity	Cognitive Complexity	Lines of code per file	Comment density (%)	Coverage	Duplication (%)
Min	0.0	0.0	0.0	0.0	0.0	0.0
Max	123.0	26.0	1545.0	100.0	0.0	0.0

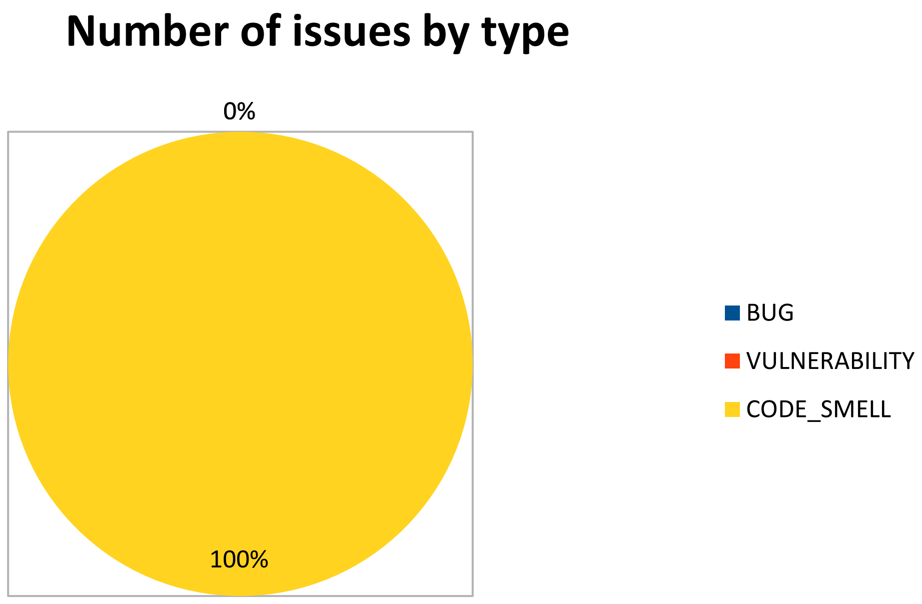
VOLUME

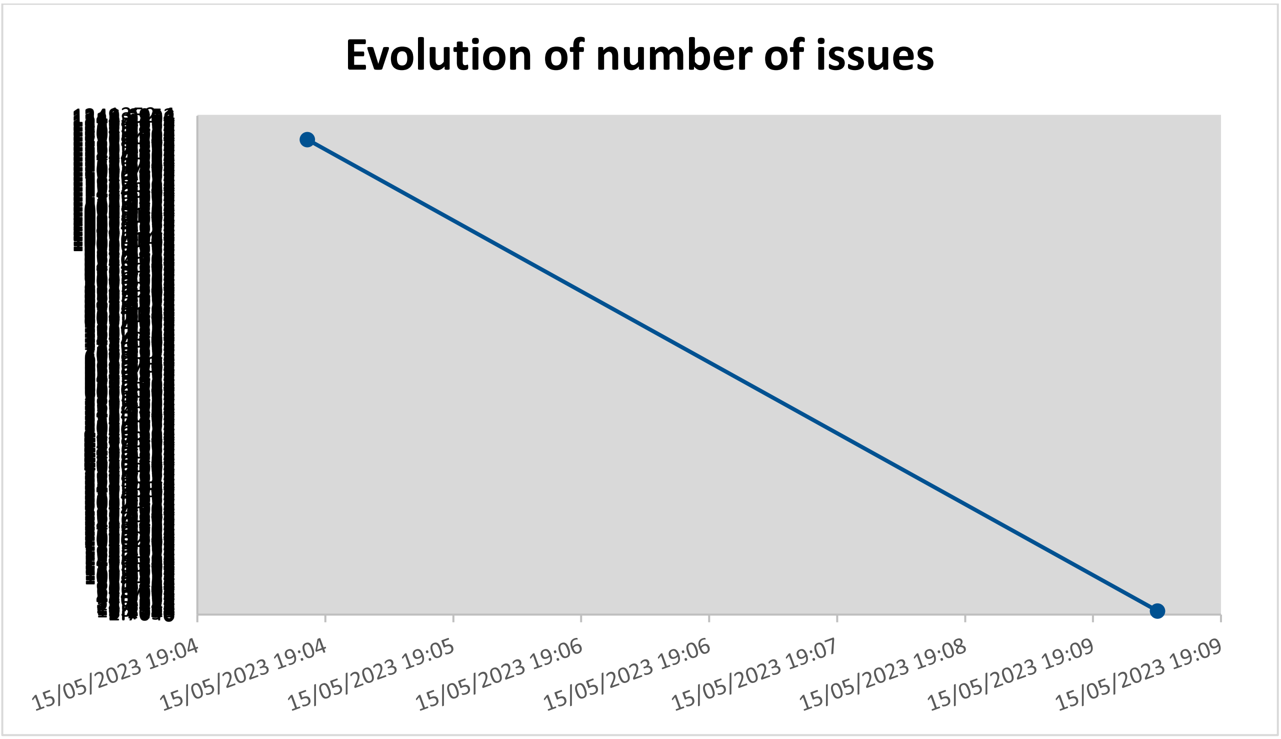
Language	Number
CSS	528
TypeScript	714
HTML	303
Total	1545

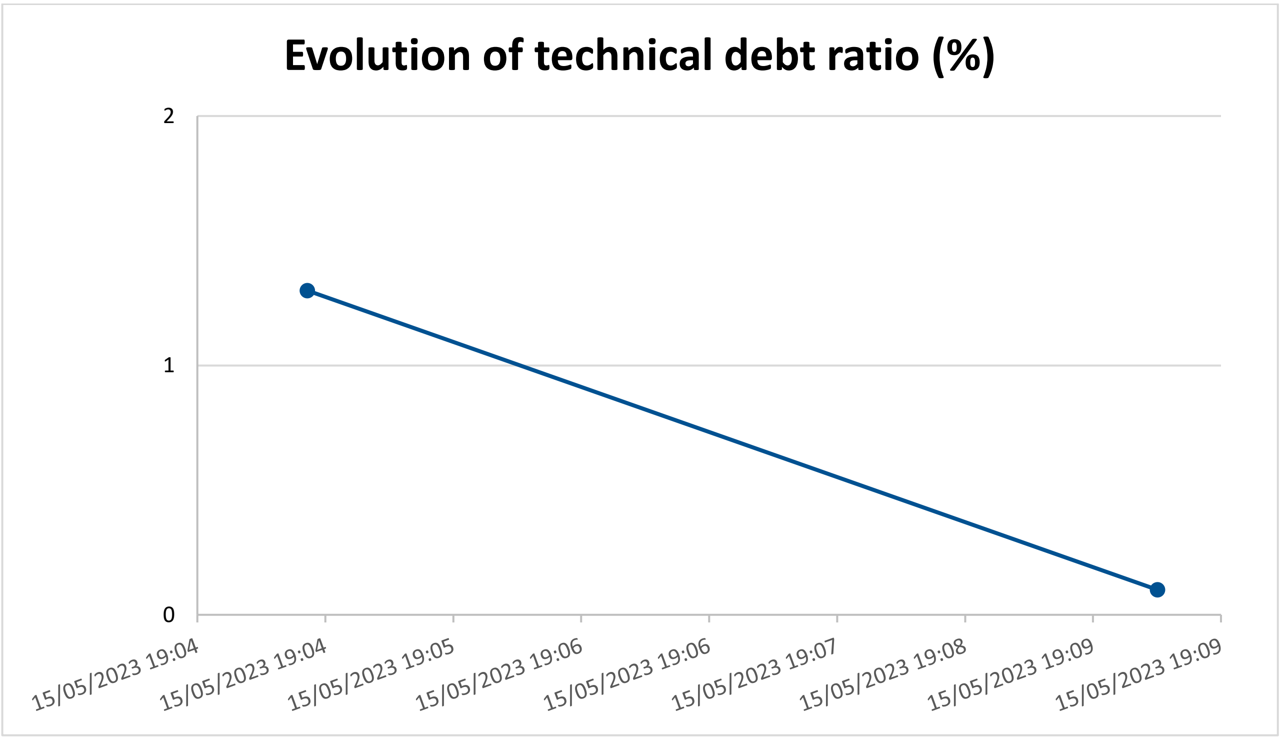
ISSUES

CHARTS






Evolution of number of issues
15/05/2023 19:04
15/05/2023 19:04
15/05/2023 19:05
15/05/2023 19:06
15/05/2023 19:06
15/05/2023 19:07
15/05/2023 19:08
15/05/2023 19:09
15/05/2023 19:09


Evolution of technical debt ratio (%)
2
1
0
15/05/2023 19:04
15/05/2023 19:04
15/05/2023 19:05
15/05/2023 19:06
15/05/2023 19:06
15/05/2023 19:07
15/05/2023 19:08
15/05/2023 19:09
15/05/2023 19:09

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	3	4	3	0

ISSUES LIST

Name	Description	Type	Severity	Number
Functions should not be empty	There are several reasons for a function not to have a function body: It is an unintentional omission, and should be fixed to prevent an unexpected behavior in production. It is not yet, or never will be, supported. In this case an exception should be thrown in languages where that mechanism is available. The method is an intentionally-blank override. In this case a nested comment should explain the reason for the blank override. Noncompliant Code Example function foo() { } var foo = () =&gt; {}; Compliant Solution function foo() { // This is intentional } var foo = () =&gt; { do_something(); }; Exceptions This rule does not apply to function expressions and arrow functions as they can denote default values. static defaultProps = { listStyle: () =&gt; {} }; The rule allows for empty functions with a name starting with the prefix on like onClick. function onClick() { }	CODE_SMELL	CRITICAL	3
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	2
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
CSS files should not be empty	This rule raises an issue when a CSS file is empty (ie: containing only spaces).	CODE_SMELL	MAJOR	1

Class names should comply with a naming convention	Shared coding conventions allow teams to collaborate effectively. This rule allows to check that all class names (and interfaces for TypeScript) match a provided regular expression. Noncompliant Code Example With default provided regular expression ^[A-Z][a-zA-Z0-9]*$: class my_class {...} Compliant Solution class MyClass {...}	CODE_SMELL	MINOR	1
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
Jump statements should not be redundant	Jump statements, such as return, break and continue let you change the default flow of program execution, but jump statements that direct the control flow to the original direction are just a waste of keystrokes. Noncompliant Code Example function redundantJump(x) { if (x == 1) { console.log("x == 1"); return; // Noncompliant } } Compliant Solution function redundantJump(x) { if (x == 1) { console.log("x == 1"); } } Exceptions break and return inside switch statement are ignored, because they are often used for consistency. continue with label is also ignored, because label is usually used for clarity. Also a jump statement being a single statement in a block is ignored.	CODE_SMELL	MINOR	1

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	2	0	0

SECURITY HOTSPOTS LIST

Category	Name	Priority	Severity	Count
Others	Disabling resource integrity features is security-sensitive	LOW	MINOR	2