|  |
| --- |
| TWD-FRONT  Version not provided  Code analysis |

|  |
| --- |
| **By: Administrator**  **2024-12-12** |

# 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 TWD-FRONT.

# Configuration

* Quality Profiles
  + Names: Sonar way [CSS]; Sonar way [JavaScript]; Sonar way [Python]; Sonar way [HTML];
  + Files: AZLA0SfYUhnVQK0JYNUD.json; AZLA0SpzUhnVQK0JYNi7.json; AZLA0SudUhnVQK0JYNsX.json; AZLA0TDkUhnVQK0JYOVt.json;
* Quality Gate
  + Name: Sonar way
  + File: Sonar way.xml

# Synthesis

## Analysis Status

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

## Quality gate status

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



## Metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coverage | Duplication | Comment  density | Median number of lines of code per file | Adherence to coding standard |
| 0.0 % | **5.6 %** | **7.6 %** | **53.0** | **99.6 %** |

## Tests

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

## Detailed technical debt

|  |  |  |  |
| --- | --- | --- | --- |
| Reliability | Security | Maintainability | Total |
| - | - | 0d 1h 22min | 0d 1h 22min |

## 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 | 174.0 | 56.0 | 2754.0 | 58.3 | 0.0 | 90.3 |

## Volume

|  |  |
| --- | --- |
| Language | Number |
| CSS | 1265 |
| JavaScript | 1770 |
| Python | 166 |
| HTML | 13 |
| Total | 3214 |

# 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 | 3 | 7 | 3 | 0 |

## Issues List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Description | Type | Severity | Number |
| String literals should not be duplicated | Duplicated string literals make the process of refactoring error-prone, since you must be sure to update all occurrences. On the other hand, constants can be referenced from many places, but only need to be updated in a single place. Noncompliant Code Example With the default threshold of 3: def run(): prepare("this is a duplicate") # Noncompliant - "this is a duplicate" is duplicated 3 times execute("this is a duplicate") release("this is a duplicate") Compliant Solution ACTION\_1 = "action1" def run(): prepare(ACTION\_1) execute(ACTION\_1) release(ACTION\_1) Exceptions No issue will be raised on: duplicated string in decorators strings with less than 5 characters strings with only letters, numbers and underscores @app.route("/api/users/", methods=['GET', 'POST', 'PUT']) def users(): pass @app.route("/api/projects/", methods=['GET', 'POST', 'PUT']) # Compliant def projects(): pass | CODE\_SMELL | CRITICAL | 3 |
| 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 | 2 |
| No array index for keys in JSX list components |  | CODE\_SMELL | MAJOR | 3 |
| React Context Provider values should not have non-stable identities |  | CODE\_SMELL | MAJOR | 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 | 2 |
| 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 |

# 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 | 3 |
| 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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Name | Priority | Severity | Count |
| Authentication | Hard-coded credentials are security-sensitive | HIGH | BLOCKER | 3 |