User manual

Introduction

This section details how to correctly interact and use all functions of the code review tool.

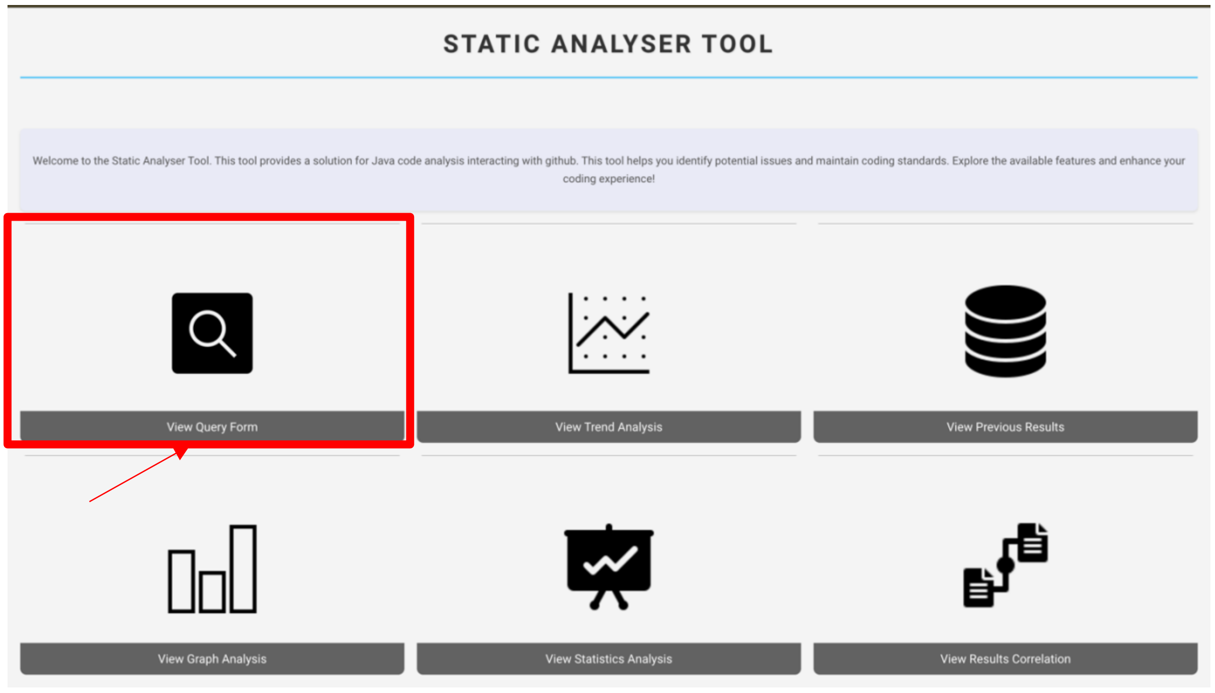
Accessing the tool

The code review tool is available as a Software as a Service Offering (SAAS), hosted on Microsoft Azure.

It can be accessed directly on the following URL: https://static-code-analyser.azurewebsites.net/

Dashboard

The dashboard serves as the central hub of the tool allowing access to that specific feature.



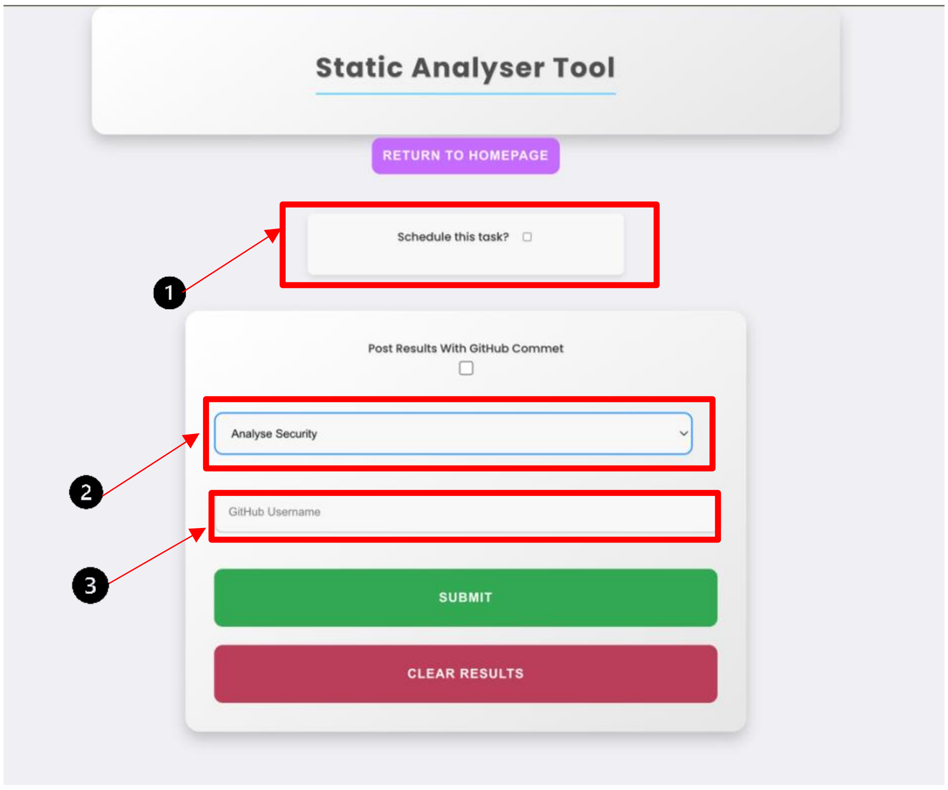
STATIC ANALYSER TOOL

Welcome to the Static Analyser Tool. This tool provides a solution for Java code analysis interacting with github. This tool helps you identify potential issues and maintain coding standards. Explore the available features and enhance your coding experience!

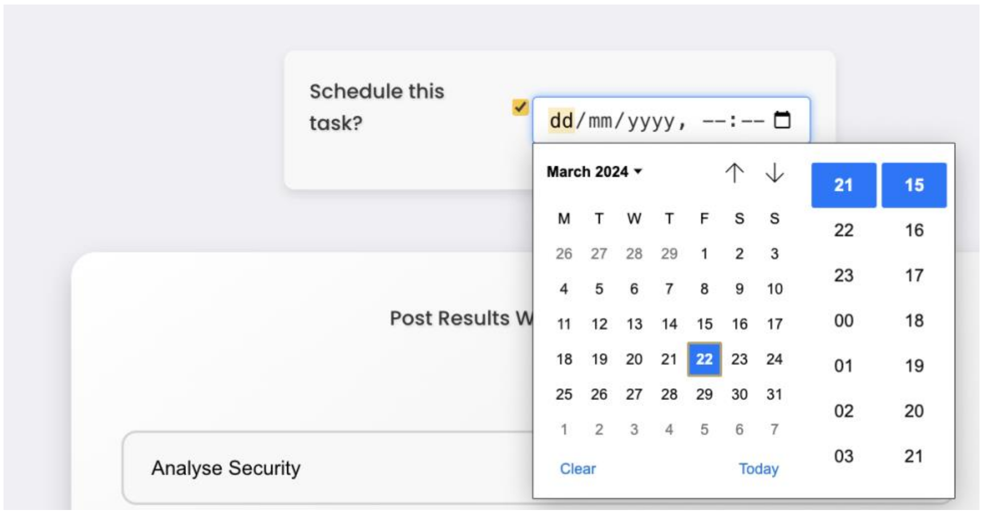
To interact with a specific feature, click on the tile representing the feature you are interested in interacting with. This will take you to the page responsible for that feature. For example, to interact with the query form click on the query form tile as shown above.

Query Form

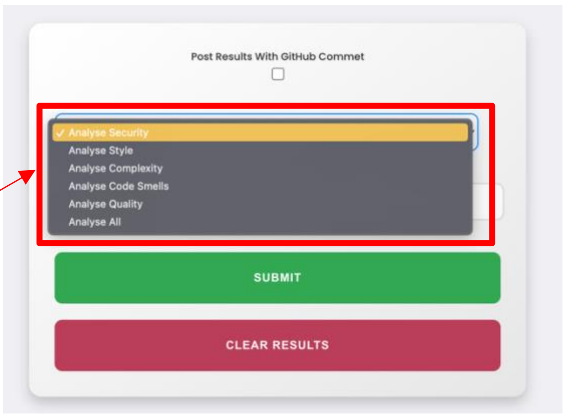
This page is responsible for allowing the running and displaying of the code review results of analysing the GitHub repository code. Upon initial loading of the query form page, you will be presenting with multiple radio buttons and query forms to allow for interaction.



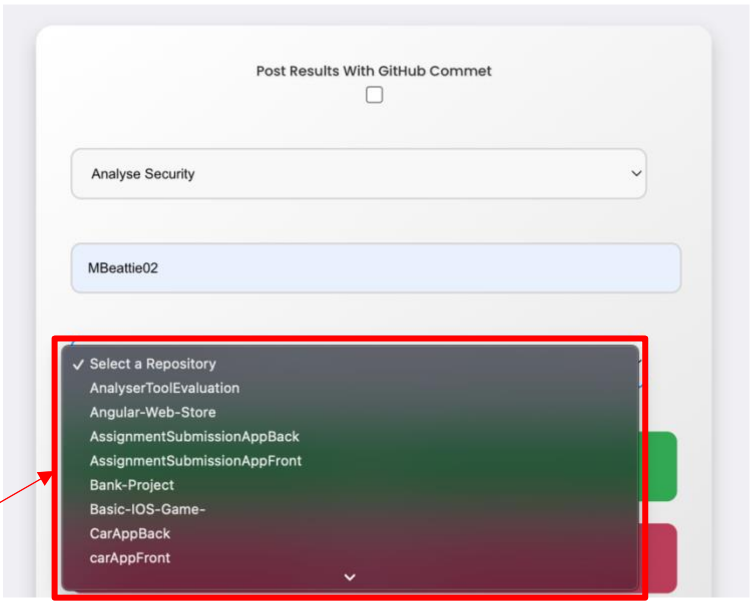
1. Firstly, you have the choice to either begin the code review immediately or to schedule it for a later date. If you select the radio button, you will be displayed a calendar which you can use to schedule the review run.



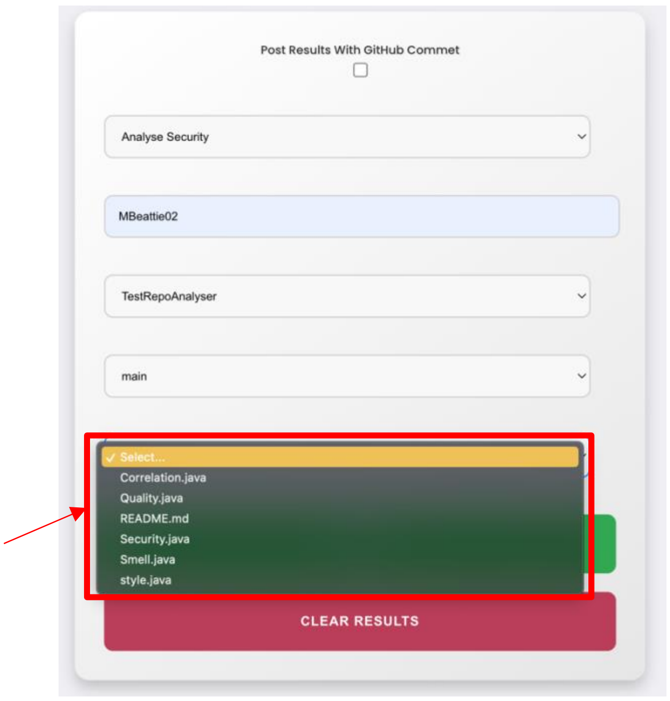
2. You are then required to enter the type of code review you would like to perform. This should be chosen from the drop-down options in the query box.



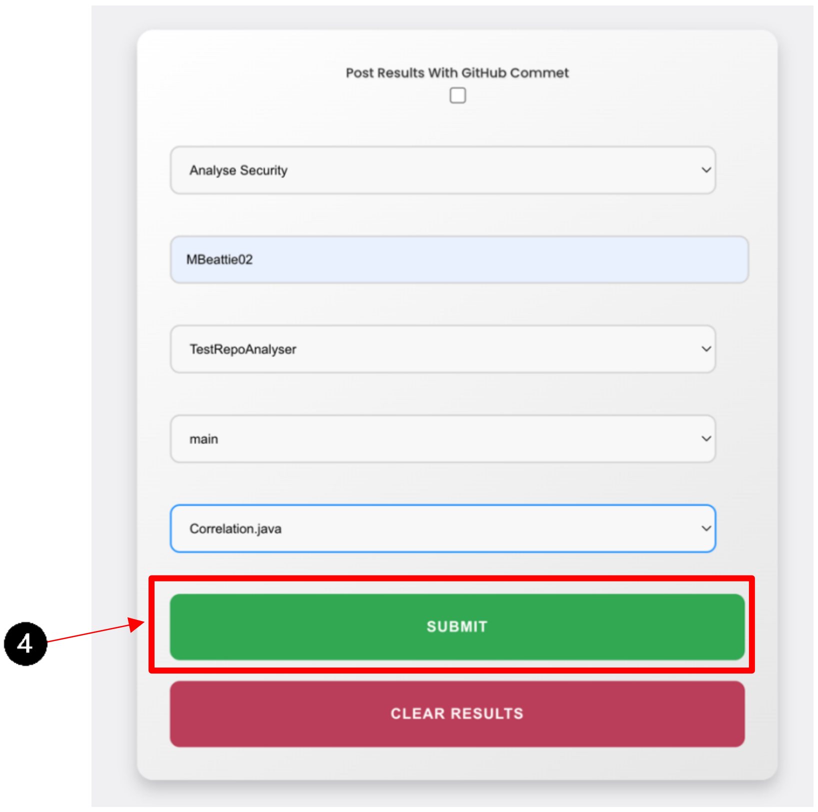
3. Next, you are required to enter the GitHub Username of the account which holds the file you would like to perform the code rewiew on. Once this is completed a drop down will be displayed of all repositories that are publicly available to choose from.



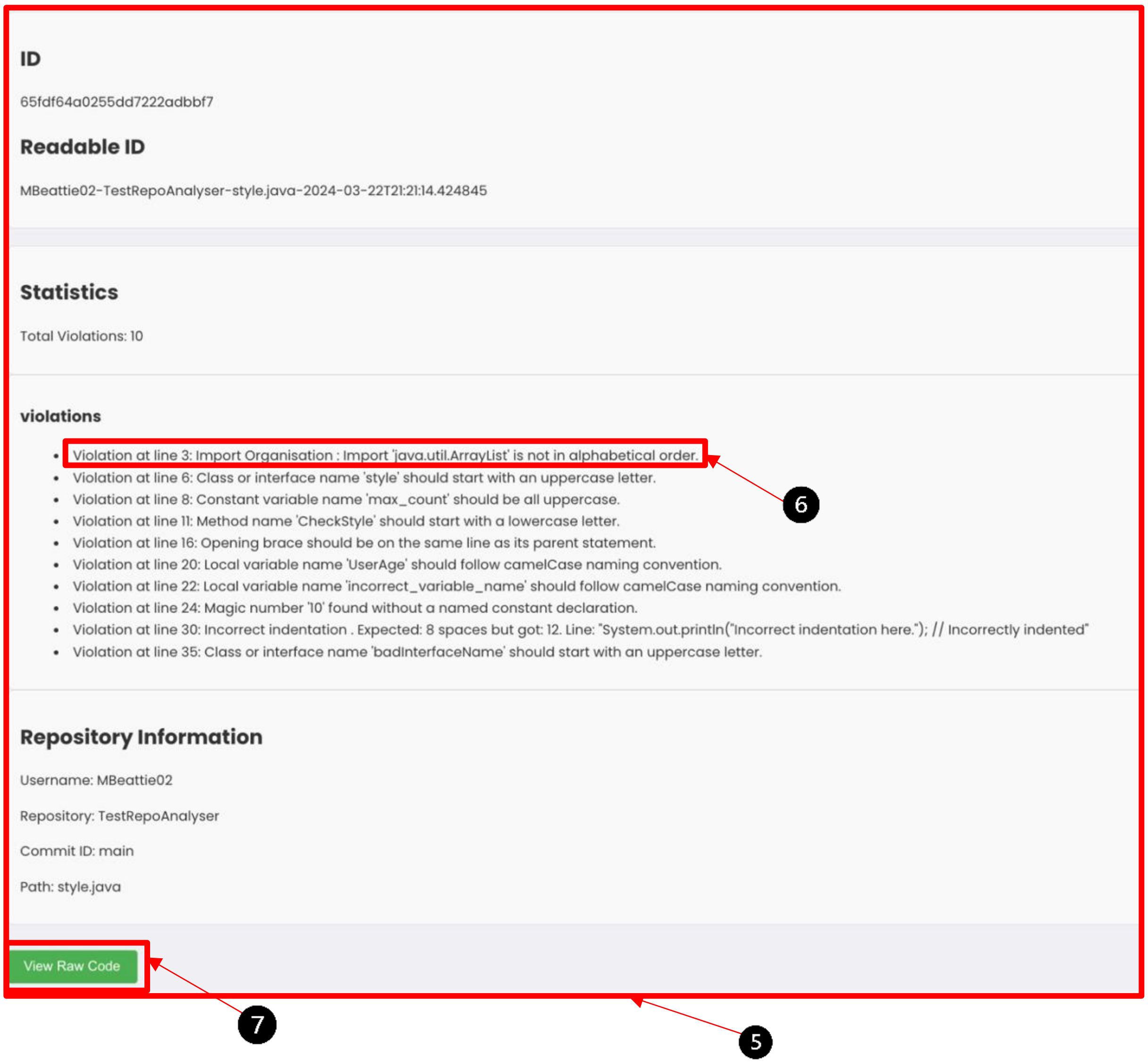
Once you select a repository from the drop-down you will continue to be presented with dropdowns to select from until you reach the root of the project which holds the .java file you would like to analyse.



4. Once all query boxes have been filled in you should then press the submit button to begin code review of the codebase.



5. After successful code review the results of the code review run will be displayed. This includes:
    a. Unique ID of the code review run
    b. Statistics of the total number of vulnerabilities
    c. List of the specific vulnerabilities that have been detected.
    d. Information of the repository that has been analysed.



ID

65fdf64a0255dd7222adbbf7

Readable ID

MBeattie02-TestRepoAnalyser-style.java-2024-03-22T21:21:14.424845

Statistics

Total Violations: 10

violations

- Violation at line 3: Import Organisation : Import 'java.util.ArrayList' is not in alphabetical order.
- Violation at line 6: Class or interface name 'style' should start with an uppercase letter.
- Violation at line 8: Constant variable name 'max_count' should be all uppercase.
- Violation at line 11: Method name 'CheckStyle' should start with a lowercase letter.
- Violation at line 16: Opening brace should be on the same line as its parent statement.
- Violation at line 20: Local variable name 'UserAge' should follow camelCase naming convention.
- Violation at line 22: Local variable name 'incorrect_variable_name' should follow camelCase naming convention.
- Violation at line 24: Magic number '10' found without a named constant declaration.
- Violation at line 30: Incorrect indentation . Expected: 8 spaces but got: 12. Line: "System.out.println("Incorrect indentation here."); // Incorrectly indented"
- Violation at line 35: Class or interface name 'badInterfaceName' should start with an uppercase letter.

Repository Information

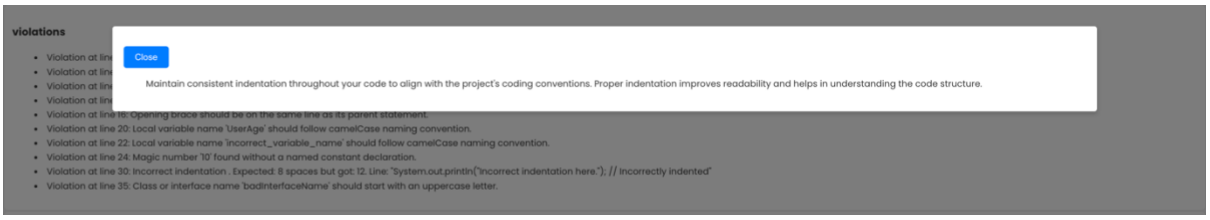
Username: MBeattie02

Repository: TestRepoAnalyser

Commit ID: main

Path: style.java

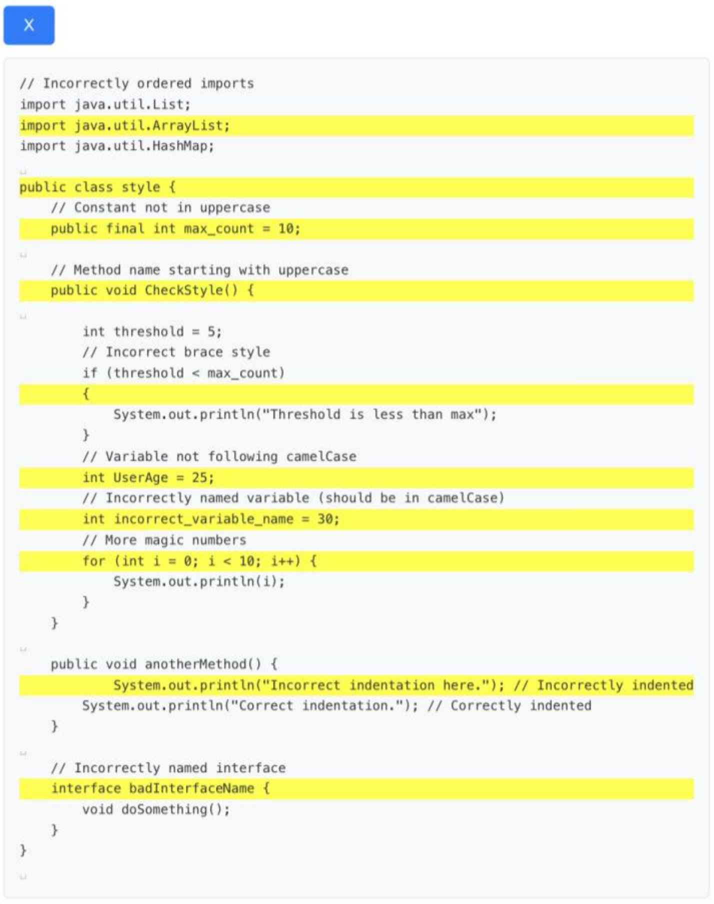
6. To get a more detailed understanding of the issue that has been detected you can click on the violation. This will display a more detailed explanation of the issue in a pop-up modal once clicked.



Maintain consistent indentation throughout your code to align with the project's coding conventions. Proper indentation improves readability and helps in understanding the code structure.

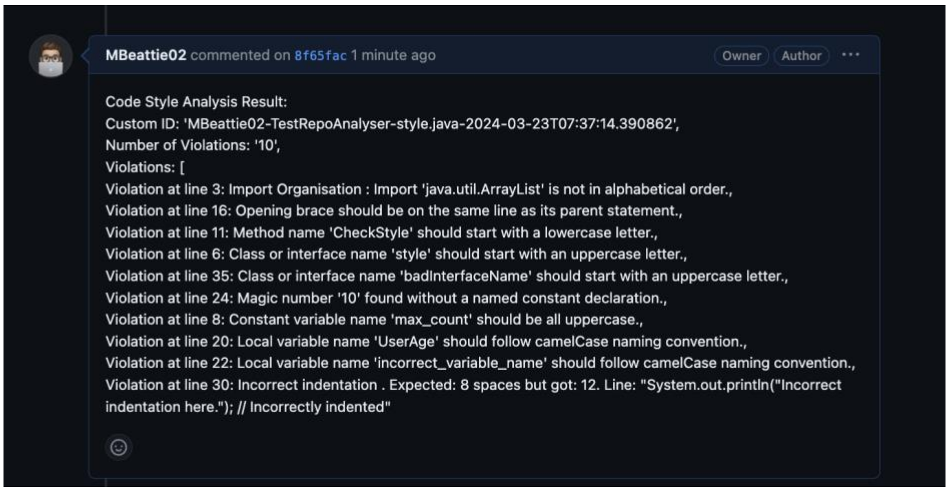
- Violation at line 16: Opening brace should be on the same line as its parent statement.
- Violation at line 20: Local variable name 'UserAge' should follow camelCase naming convention.
- Violation at line 22: Local variable name 'incorrect_variable_name' should follow camelCase naming convention.
- Violation at line 24: Magic number '10' found without a named constant declaration.
- Violation at line 30: Incorrect indentation . Expected: 8 spaces but got: 12. Line: "System.out.println("Incorrect indentation here."); // Incorrectly indented"
- Violation at line 35: Class or interface name 'badInterfaceName' should start with an uppercase letter.

7. To view the raw code of the code review, click the View Raw Code button. This will display a modal with the vulnerability lines highlighted.



```
// Incorrectly ordered imports
import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;

public class style {
    // Constant not in uppercase
    public final int max_count = 10;

    // Method name starting with uppercase
    public void CheckStyle() {

        int threshold = 5;
        // Incorrect brace style
        if (threshold < max_count)
        {
            System.out.println("Threshold is less than max");
        }
        // Variable not following camelCase
        int UserAge = 25;
        // Incorrectly named variable (should be in camelCase)
        int incorrect_variable_name = 30;
        // More magic numbers
        for (int i = 0; i < 10; i++) {
            System.out.println(i);
        }
    }

    public void anotherMethod() {
            System.out.println("Incorrect indentation here."); // Incorrectly indented
        System.out.println("Correct indentation."); // Correctly indented
    }

    // Incorrectly named interface
    interface badInterfaceName {
        void doSomething();
    }
}
```

8. You also have the choice to post the results of the code review to GitHub. If the radio button is selected once the code review is completed it will post the result of the code review to the GitHub commit of the file which the tool has analysed.



```
Code Style Analysis Result:
Custom ID: 'MBeattie02-TestRepoAnalyser-style.java-2024-03-23T07:37:14.390862',
Number of Violations: '10',
Violations: [
Violation at line 3: Import Organisation : Import 'java.util.ArrayList' is not in alphabetical order.,
Violation at line 16: Opening brace should be on the same line as its parent statement.,
Violation at line 11: Method name 'CheckStyle' should start with a lowercase letter.,
Violation at line 6: Class or interface name 'style' should start with an uppercase letter.,
Violation at line 35: Class or interface name 'badInterfaceName' should start with an uppercase letter.,
Violation at line 24: Magic number '10' found without a named constant declaration.,
Violation at line 8: Constant variable name 'max_count' should be all uppercase.,
Violation at line 20: Local variable name 'UserAge' should follow camelCase naming convention.,
Violation at line 22: Local variable name 'incorrect_variable_name' should follow camelCase naming convention.,
Violation at line 30: Incorrect indentation . Expected: 8 spaces but got: 12. Line: "System.out.println("Incorrect
indentation here."); // Incorrectly indented"
```

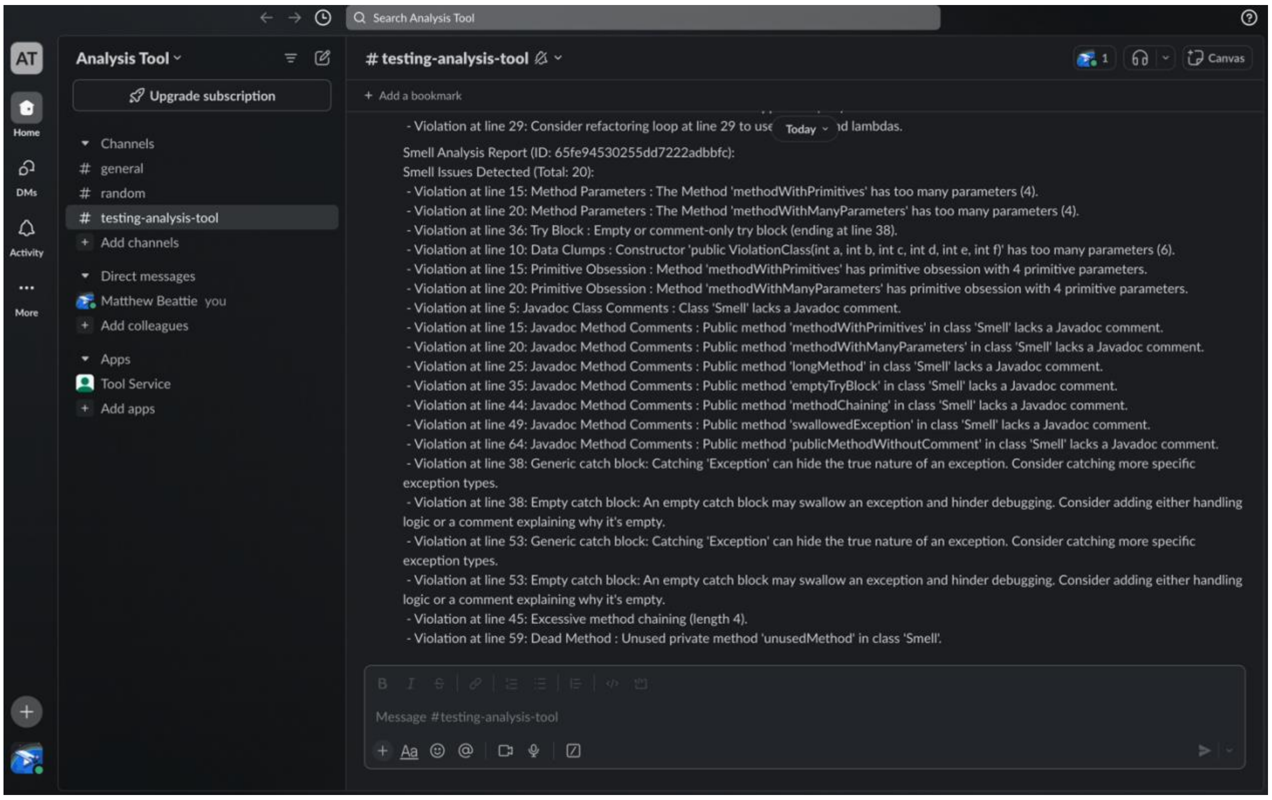
Slack

A slack workspace is available allowing you to receive alerts each time that an code review run has been completed.

The workspace can be joined at:

https://join.slack.com/t/analysis-tool/shared_invite/zt-2fex2ajk3-fpU5hvX5WE1jSyHOMOqFtw

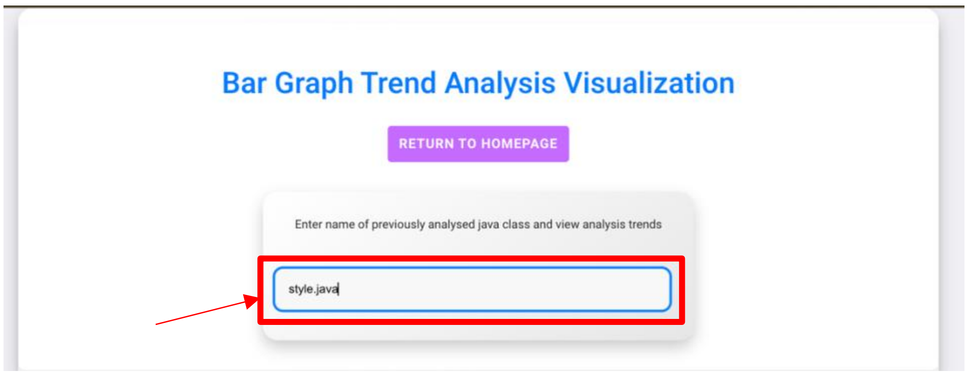
(Link valid for 30 days)



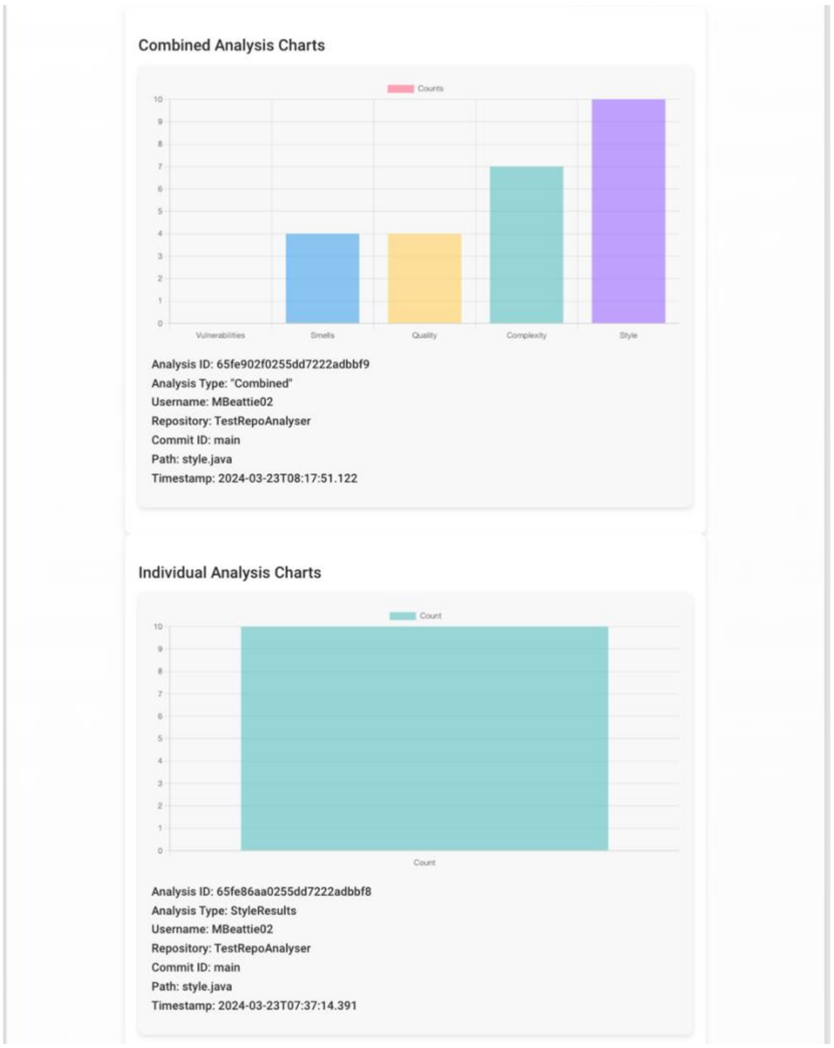
```
 - Violation at line 29: Consider refactoring loop at line 29 to use [Today] nd lambdas.
Smell Analysis Report (ID: 65fe94530255dd7222adbbfc):
Smell Issues Detected (Total: 20):
 - Violation at line 15: Method Parameters : The Method 'methodWithPrimitives' has too many parameters (4).
 - Violation at line 20: Method Parameters : The Method 'methodWithManyParameters' has too many parameters (4).
 - Violation at line 36: Try Block : Empty or comment-only try block (ending at line 38).
 - Violation at line 10: Data Clumps : Constructor 'public ViolationClass(int a, int b, int c, int d, int e, int f)' has too many parameters (6).
 - Violation at line 15: Primitive Obsession : Method 'methodWithPrimitives' has primitive obsession with 4 primitive parameters.
 - Violation at line 20: Primitive Obsession : Method 'methodWithManyParameters' has primitive obsession with 4 primitive parameters.
 - Violation at line 5: Javadoc Class Comments : Class 'Smell' lacks a Javadoc comment.
 - Violation at line 15: Javadoc Method Comments : Public method 'methodWithPrimitives' in class 'Smell' lacks a Javadoc comment.
 - Violation at line 20: Javadoc Method Comments : Public method 'methodWithManyParameters' in class 'Smell' lacks a Javadoc comment.
 - Violation at line 25: Javadoc Method Comments : Public method 'longMethod' in class 'Smell' lacks a Javadoc comment.
 - Violation at line 35: Javadoc Method Comments : Public method 'emptyTryBlock' in class 'Smell' lacks a Javadoc comment.
 - Violation at line 44: Javadoc Method Comments : Public method 'methodChaining' in class 'Smell' lacks a Javadoc comment.
 - Violation at line 49: Javadoc Method Comments : Public method 'swallowedException' in class 'Smell' lacks a Javadoc comment.
 - Violation at line 64: Javadoc Method Comments : Public method 'publicMethodWithoutComment' in class 'Smell' lacks a Javadoc comment.
 - Violation at line 38: Generic catch block: Catching 'Exception' can hide the true nature of an exception. Consider catching more specific
exception types.
 - Violation at line 38: Empty catch block: An empty catch block may swallow an exception and hinder debugging. Consider adding either handling
logic or a comment explaining why it's empty.
 - Violation at line 53: Generic catch block: Catching 'Exception' can hide the true nature of an exception. Consider catching more specific
exception types.
 - Violation at line 53: Empty catch block: An empty catch block may swallow an exception and hinder debugging. Consider adding either handling
logic or a comment explaining why it's empty.
 - Violation at line 45: Excessive method chaining (length 4).
 - Violation at line 59: Dead Method : Unused private method 'unusedMethod' in class 'Smell'.
```

Graph Analysis

The graph analysis page allows you to visually view the results of all the code review runs that has been carried out on a given java file. This facilitates the easy viewing of trends and the number of vulnerabilities that have been detected.


Bar Graph Trend Analysis Visualization
RETURN TO HOMEPAGE
Enter name of previously analysed java class and view analysis trends
style.java

To interact with and view the graphs for each code review run you firstly need to enter the name of a java class file which has been previously analysed in the query form box. The name of the file must be followed by .java . if the file exists the code review runs relating to this class will be displayed.



Analysis ID: 65fe902f0255dd7222adbbf9
Analysis Type: "Combined"
Username: MBeattie02
Repository: TestRepoAnalyser
Commit ID: main
Path: style.java
Timestamp: 2024-03-23T08:17:51.122

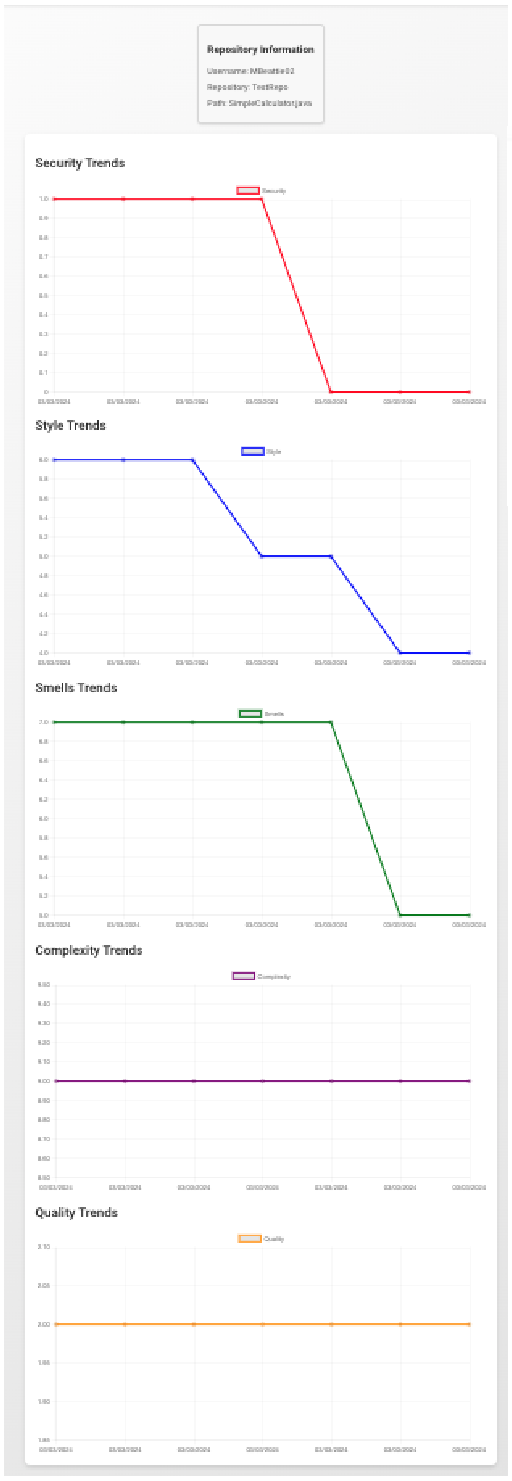
Analysis ID: 65fe86aa0255dd7222adbbf8
Analysis Type: StyleResults
Username: MBeattie02
Repository: TestRepoAnalyser
Commit ID: main
Path: style.java
Timestamp: 2024-03-23T07:37:14.391

The results will be displayed as either combined code review if all code review types were run or individual code review if only one code review type was run.

Trend Analysis

The graph analysis page allows you to visually view trends in the code review runs carried out. This allows for easy comparison of the number of vulnerabilities detected between runs to check if code fixes implemented have had any impact on the number of vulnerabilities detected.

To interact and view the graphs for each code review run you firstly need to enter the name of a java class file which has been previously analysed in the query form box. The name of the file must be followed by .java . if the file exists the analysis trends relating to this class will be displayed.

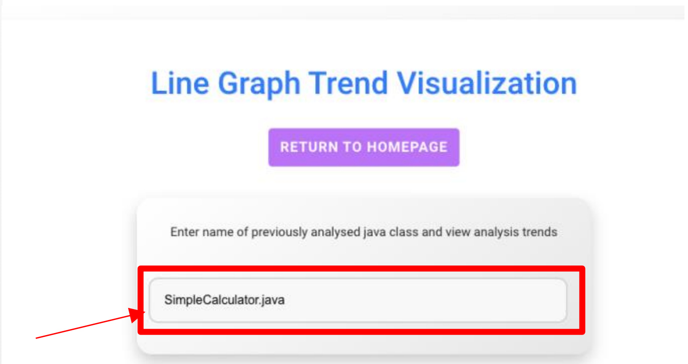


Repository Information

Username: MBeattie02

Repository: TestRepo

Path: SimpleCalculator.java



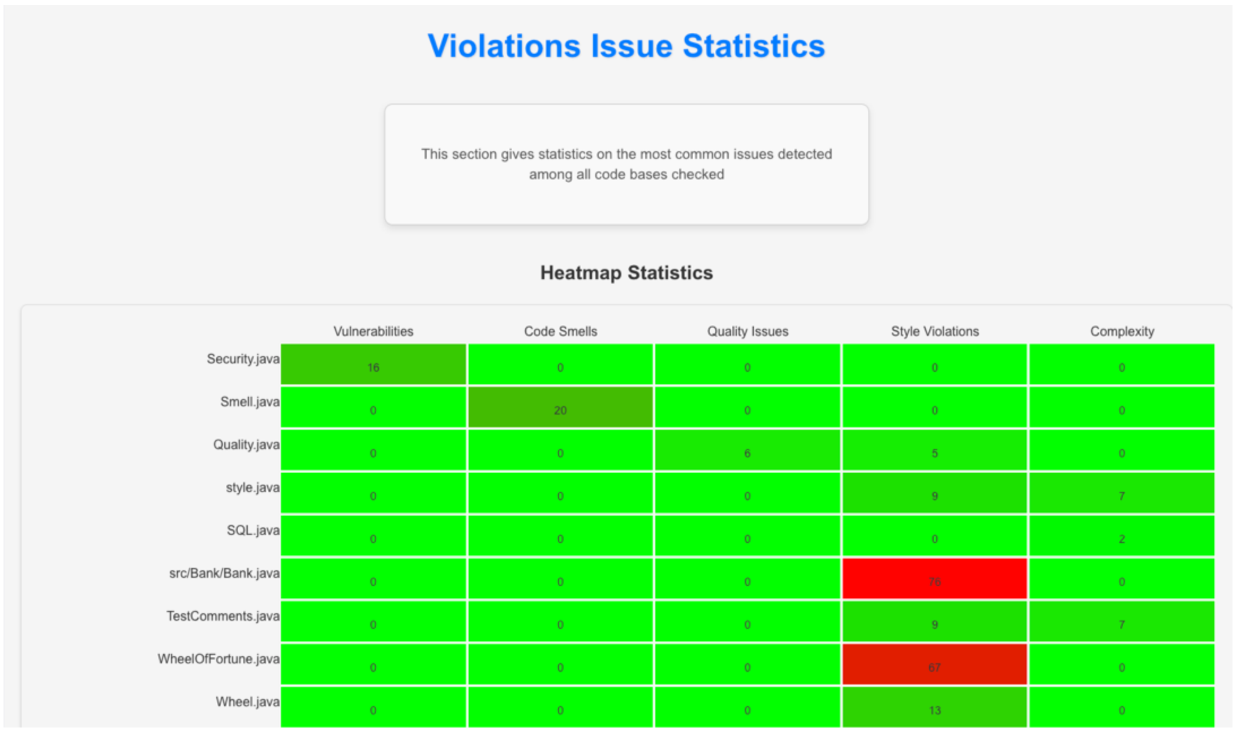
Enter name of previously analysed java class and view analysis trends

SimpleCalculator.java

Statistical Analysis

The statistical analysis page is used to get insights on the most common issues detected in the code repositories analysed. No user interaction is needed on this page.

Firstly, a heat map is displayed of all the classes which have been analysed. The darker the colour of the square the more vulnerabilities the class has.



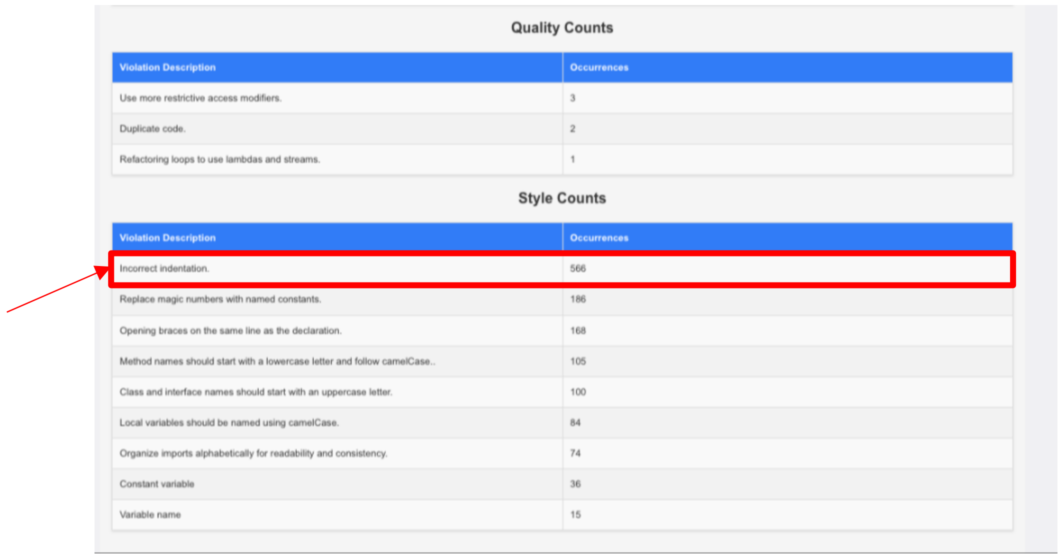
Violations Issue Statistics

This section gives statistics on the most common issues detected among all code bases checked

Heatmap Statistics

	Vulnerabilities	Code Smells	Quality Issues	Style Violations	Complexity
Security.java	16	0	0	0	0
Smell.java	0	20	0	0	0
Quality.java	0	0	6	5	0
style.java	0	0	0	9	7
SQL.java	0	0	0	0	2
src/Bank/Bank.java	0	0	0	76	0
TestComments.java	0	0	0	9	7
WheelOfFortune.java	0	0	0	67	0
Wheel.java	0	0	0	13	0

Below the heatmap is a table containing a count of all vulnerabilities that have been detected in the files analysed. This allows for quantitative analysis of the most prevalent vulnerabilities found in the code bases analysed and in need of the most improvement.



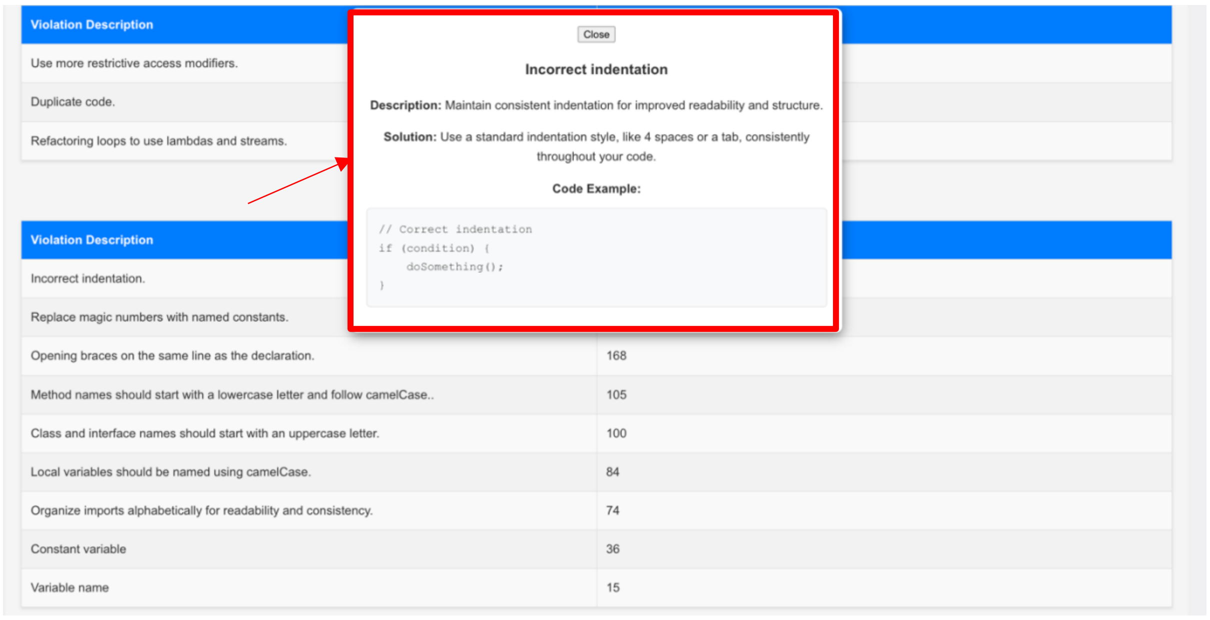
Quality Counts

Violation Description	Occurrences
Use more restrictive access modifiers.	3
Duplicate code.	2
Refactoring loops to use lambdas and streams.	1

Style Counts

Violation Description	Occurrences
Incorrect indentation.	566
Replace magic numbers with named constants.	186
Opening braces on the same line as the declaration.	168
Method names should start with a lowercase letter and follow camelCase..	105
Class and interface names should start with an uppercase letter.	100
Local variables should be named using camelCase.	84
Organize imports alphabetically for readability and consistency.	74
Constant variable	36
Variable name	15

You can also click on the vulnerability description to display a modal. This modal gives a description of what the issue is as well as the solution to fix it and a code snippet of a correct example.



Incorrect indentation

Description: Maintain consistent indentation for improved readability and structure.

Solution: Use a standard indentation style, like 4 spaces or a tab, consistently throughout your code.

Code Example:

```
// Correct indentation
if (condition) {
    doSomething();
}
```

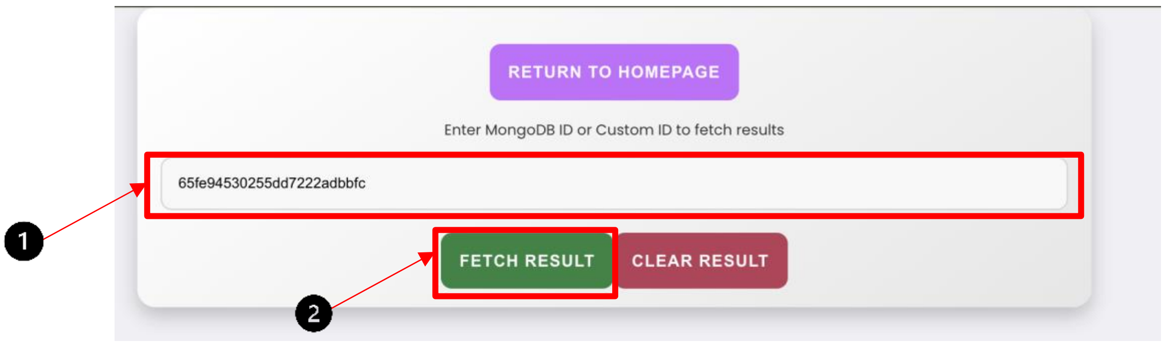
Violation Description	
Use more restrictive access modifiers.	
Duplicate code.	
Refactoring loops to use lambdas and streams.	

Violation Description	
Incorrect indentation.	
Replace magic numbers with named constants.	
Opening braces on the same line as the declaration.	168
Method names should start with a lowercase letter and follow camelCase..	105
Class and interface names should start with an uppercase letter.	100
Local variables should be named using camelCase.	84
Organize imports alphabetically for readability and consistency.	74
Constant variable	36
Variable name	15

Previous Results

The previous results page is used to retrieve the details of a previous code review run.

1. Firstly, you are required to enter either the MongoDB ID or the CustomID to fetch the results.

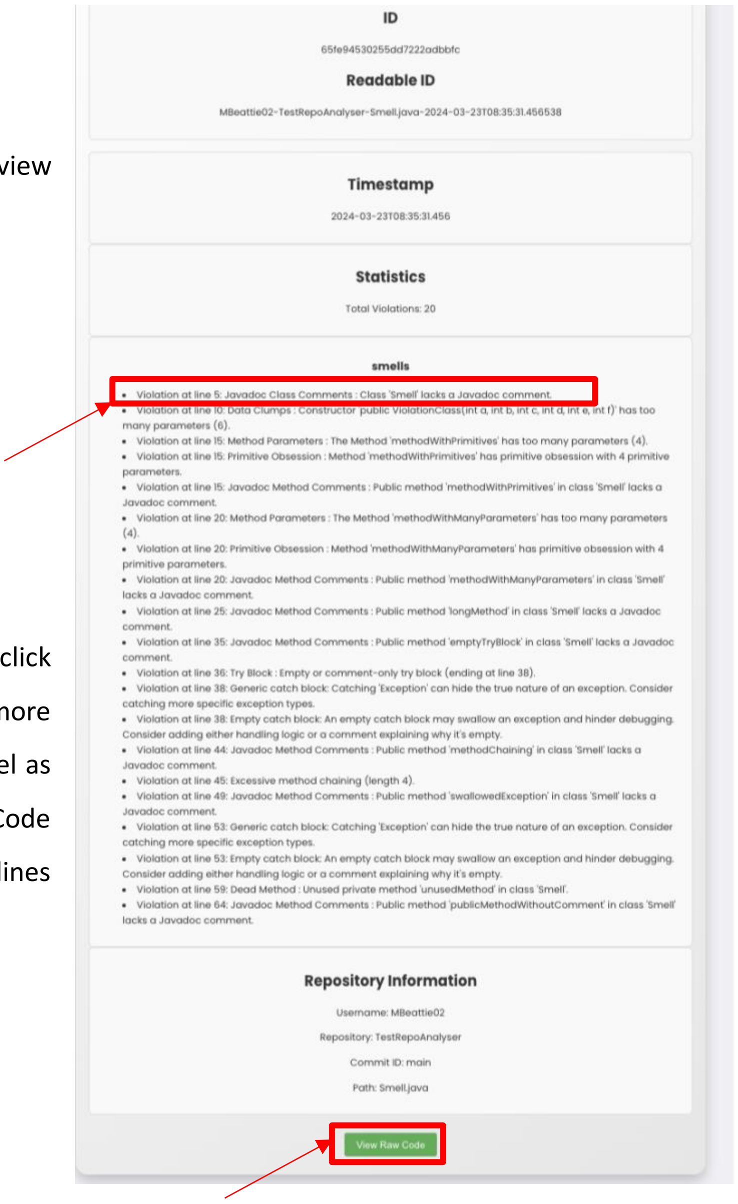


2. Once you have entered the ID, you then need to press the Fetch Result button to retrieve the details.

This will display all details of the code review run including:

- IDs
- Timestamp
- Statistics
- Vulnerabilities
- Repository Info

As on the query analysis page you can click on the vulnerability description to get a more detailed description displayed in a model as well as the option to view the Raw Code snippet with the vulnerability lines highlighted.



ID

65fe94530255dd7222adbbfc

Readable ID

MBeattie02-TestRepoAnalyser-Smell.java-2024-03-23T08:35:31.456538

Timestamp

2024-03-23T08:35:31.456

Statistics

Total Violations: 20

smells

- Violation at line 5: Javadoc Class Comments : Class 'Smell' lacks a Javadoc comment.
- Violation at line 10: Data Clumps : Constructor 'public ViolationClass(int a, int b, int c, int d, int e, int f)' has too many parameters (6).
- Violation at line 15: Method Parameters : The Method 'methodWithPrimitives' has too many parameters (4).
- Violation at line 15: Primitive Obsession : Method 'methodWithPrimitives' has primitive obsession with 4 primitive parameters.
- Violation at line 15: Javadoc Method Comments : Public method 'methodWithPrimitives' in class 'Smell' lacks a Javadoc comment.
- Violation at line 20: Method Parameters : The Method 'methodWithManyParameters' has too many parameters (4).
- Violation at line 20: Primitive Obsession : Method 'methodWithManyParameters' has primitive obsession with 4 primitive parameters.
- Violation at line 20: Javadoc Method Comments : Public method 'methodWithManyParameters' in class 'Smell' lacks a Javadoc comment.
- Violation at line 25: Javadoc Method Comments : Public method 'longMethod' in class 'Smell' lacks a Javadoc comment.
- Violation at line 35: Javadoc Method Comments : Public method 'emptyTryBlock' in class 'Smell' lacks a Javadoc comment.
- Violation at line 36: Try Block : Empty or comment-only try block (ending at line 38).
- Violation at line 38: Generic catch block: Catching 'Exception' can hide the true nature of an exception. Consider catching more specific exception types.
- Violation at line 38: Empty catch block: An empty catch block may swallow an exception and hinder debugging. Consider adding either handling logic or a comment explaining why it's empty.
- Violation at line 44: Javadoc Method Comments : Public method 'methodChaining' in class 'Smell' lacks a Javadoc comment.
- Violation at line 45: Excessive method chaining (length 4).
- Violation at line 49: Javadoc Method Comments : Public method 'swallowedException' in class 'Smell' lacks a Javadoc comment.
- Violation at line 53: Generic catch block: Catching 'Exception' can hide the true nature of an exception. Consider catching more specific exception types.
- Violation at line 53: Empty catch block: An empty catch block may swallow an exception and hinder debugging. Consider adding either handling logic or a comment explaining why it's empty.
- Violation at line 59: Dead Method : Unused private method 'unusedMethod' in class 'Smell'.
- Violation at line 64: Javadoc Method Comments : Public method 'publicMethodWithoutComment' in class 'Smell' lacks a Javadoc comment.

Repository Information

Username: MBeattie02

Repository: TestRepoAnalyser

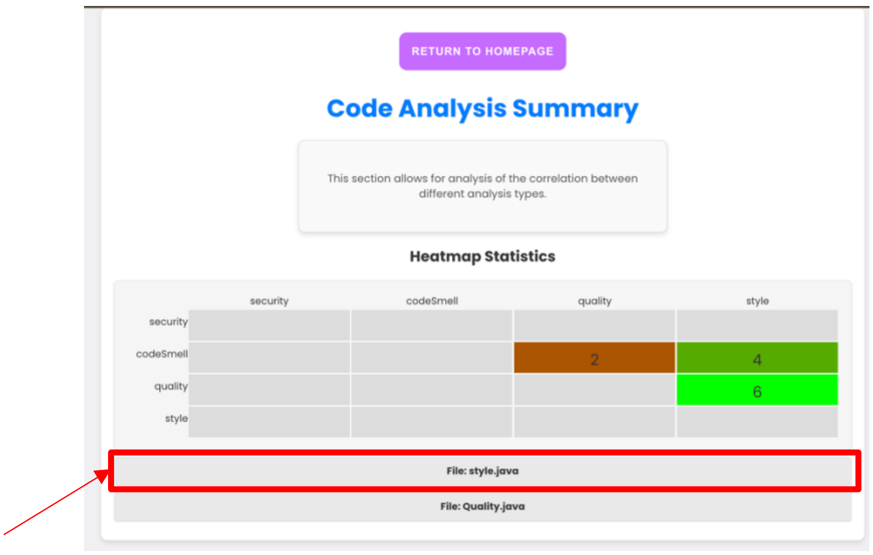
Commit ID: main

Path: Smell.java

Results Correlation

The statistical analysis page is used to check for any correlation between the different types on code review that the tool can perform. No user interaction is needed on this page.

Firstly, a heat map is displayed allowing for the convenient checking of correlations. The colours range from red to green depending on the number of correlations found.



Code Analysis Summary

This section allows for analysis of the correlation between different analysis types.

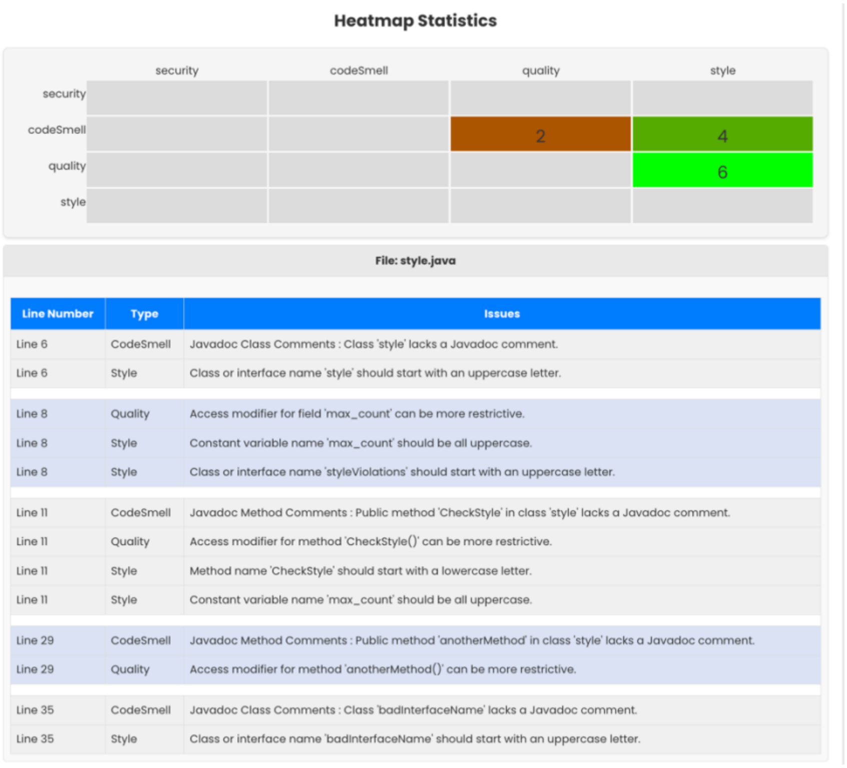
Heatmap Statistics

	security	codeSmell	quality	style
security				
codeSmell			2	4
quality				6
style				

File: style.java

File: Quality.java

You can also click on the file name to display a carousel of a more detailed description of the type of issue and line number which the correlation has been discovered on.



Heatmap Statistics

	security	codeSmell	quality	style
security				
codeSmell			2	4
quality				6
style				

File: style.java

Line Number	Type	Issues
Line 6	CodeSmell	Javadoc Class Comments : Class 'style' lacks a Javadoc comment.
Line 6	Style	Class or interface name 'style' should start with an uppercase letter.
Line 8	Quality	Access modifier for field 'max_count' can be more restrictive.
Line 8	Style	Constant variable name 'max_count' should be all uppercase.
Line 8	Style	Class or interface name 'styleViolations' should start with an uppercase letter.
Line 11	CodeSmell	Javadoc Method Comments : Public method 'CheckStyle' in class 'style' lacks a Javadoc comment.
Line 11	Quality	Access modifier for method 'CheckStyle()' can be more restrictive.
Line 11	Style	Method name 'CheckStyle' should start with a lowercase letter.
Line 11	Style	Constant variable name 'max_count' should be all uppercase.
Line 29	CodeSmell	Javadoc Method Comments : Public method 'anotherMethod' in class 'style' lacks a Javadoc comment.
Line 29	Quality	Access modifier for method 'anotherMethod()' can be more restrictive.
Line 35	CodeSmell	Javadoc Class Comments : Class 'badInterfaceName' lacks a Javadoc comment.
Line 35	Style	Class or interface name 'badInterfaceName' should start with an uppercase letter.