**Task 2b - Documentation**

**Tools used** : CodeMR

**Code Metrics** : Complexity, Coupling, Lack of Cohesion, Size

**Explanation of Metrics:**

1. **Complexity:**

Complexity gives a measure of how hard is the code to understand and explains how entities interact with one another. Higher levels of complexity in the code increases risk of unintentionally interfering with interactions and increases the chance of introducing defects while making changes in future.

1. **Coupling :**

Coupling gives a measure of the extent to which classes are coupled with each other. It is used to identify whether the system is tightly coupled or loosely coupled.

Characteristics of tightly coupled systems:

* A change in a class usually forces a ripple effect of changes in other classes.
* Require more effort and/or time due to the increased dependency.
* Might be harder to reuse a class because dependent classes must be included.

1. **Lack of Cohesion :**

Measure how well the methods of a class are related to each other. High cohesion tend to be preferable, because high cohesion is associated with several desirable traits of software including robustness, reliability, reusability, and understandability. In contrast, low cohesion is associated with undesirable traits such as being difficult to maintain, test, reuse, or even understand.

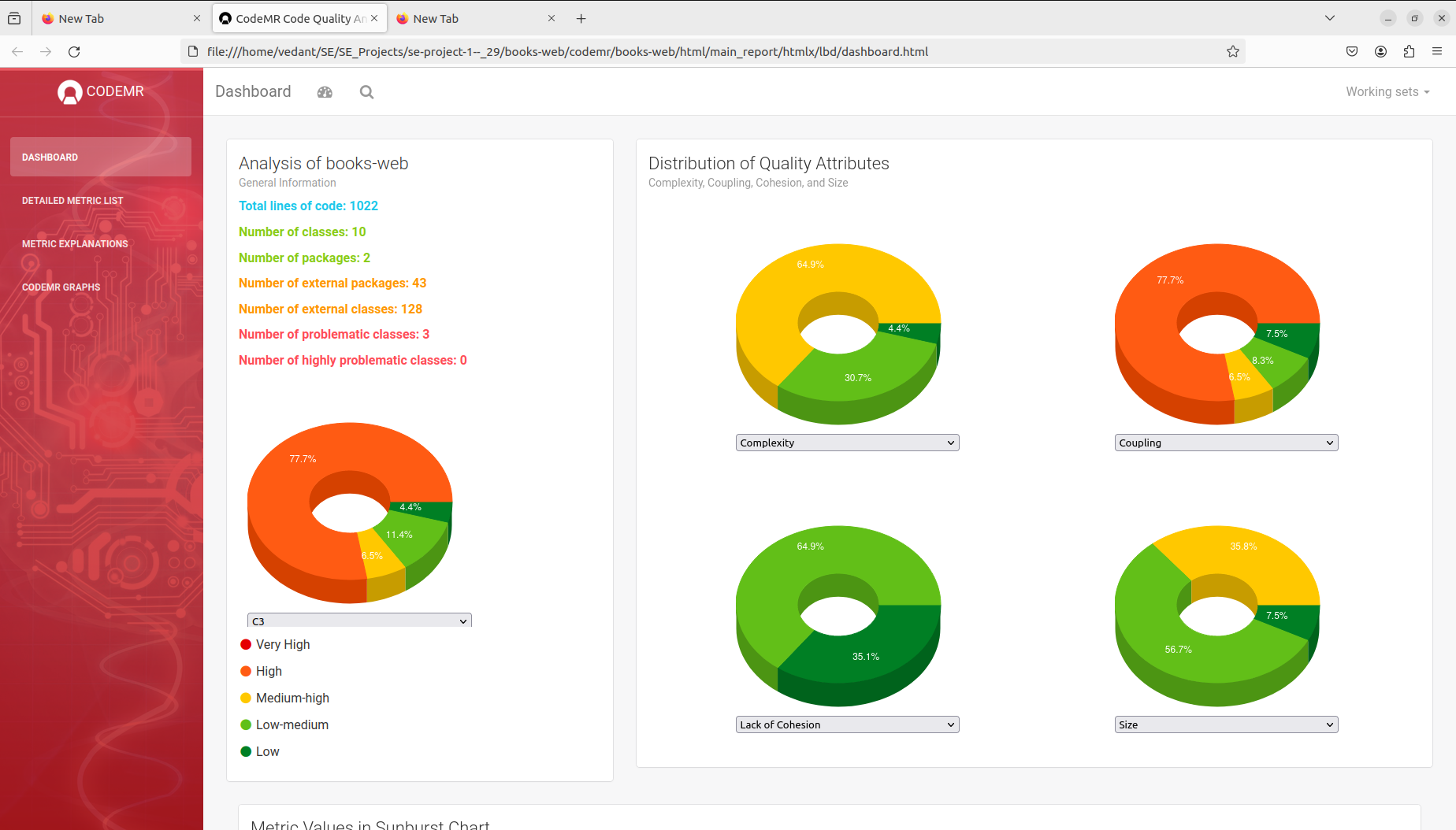
1. **Size :**

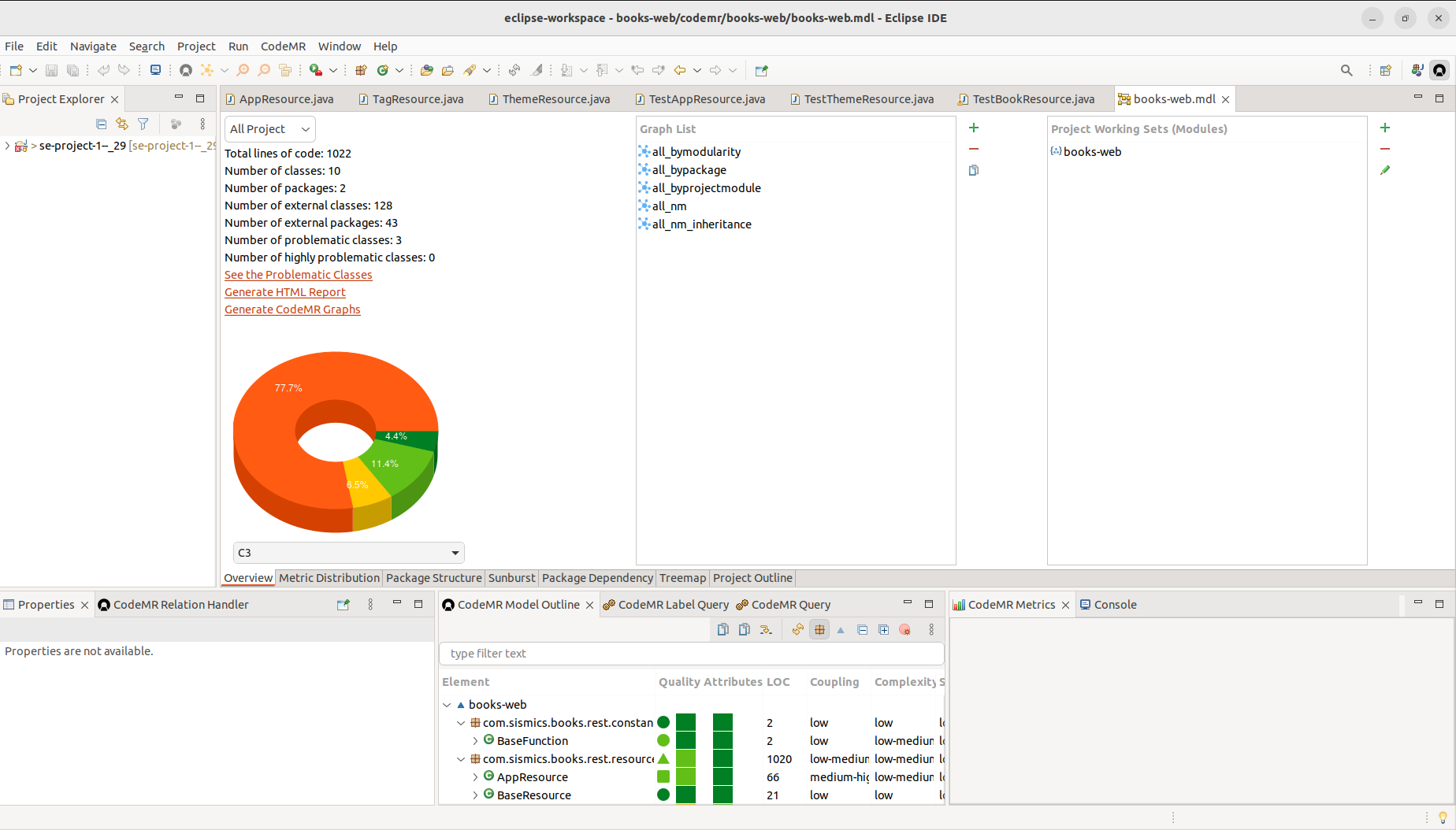
It gives a measure of the number of lines or methods in the code. A very high count might indicate that a class or method is trying to do too much work and should be split up. It might also indicate that the class might be hard to maintain.

1. **C3 :**

It is maximum value of Coupling, Cohesion, Complexity metrics.

**Screenshots :**

****

****

**Insights related to current project :**

1. **Complexity :**

There are two classes with medium-high complexity,Six classes with low-medium complexity and two classes with low complexity.

1Classes having medium-high complexity (64.9%)

1.1)BookResources

1.2)UserResources

Classes having low-medium complexity (30.7%)

2.1)AppResources

2.2)TagResources

2.3)ConnectResources

2.4)LocaleResources

2.5)ThemeResources

2.6)Resources

3)Classes having low complexity(4.4%)

3.1) BaseResources

3.2) TextPlainMessageBodyWriter

The 'BookResource' class has a high complexity, which might indicate that it's doing too much or has too many responsibilities. This could make it more difficult to maintain or could increase the risk of defects.

**2) Coupling**

There are two classes with medium-high complexity,Six classes with low-medium complexity and two classes with low complexity.

1)Classes having high coupling (77.7%)

1.1)BookResources

1.2)UserResources

1.3)ConnectResources

2) Classes having medium-high coupling (6.5%)

2.1)AppResources

3)Classes having low-medium coupling (6.5%)

3.1)TagResources

3.2)Resources

4)Classes having low coupling (7.5%)

4.1)LocaleResources

4.2)ThemeResources

4.3) BaseResources

4.4) TextPlainMessageBodyWriter

The 'BaseResource' class has a high level of coupling, suggesting it might be too dependent on other classes or components. This can affect the modularity of the code, making changes more ripple-prone and potentially impacting reusability.

3)Lack of Cohesion:

There are 8 classes with low lack of Cohesion and 2 classes with low-medium lack of Cohesion.

1)Classes having low medium lack of cohesion(35.1%)

1.1)BookResources

1.2)UserResources

2) Classes having low Lack of cohesion(64.9%)

2.1)ConnectResources

2.2)AppResources

2.3)TagResources

2.4)TextPlainMessageBodyWriter

2.5)ThemeResources

2.6)LocaleResources

Several classes like 'BookResource', 'BaseResource', and 'LocaleResource' exhibit low to medium cohesion. This suggests that the classes may be grouping unrelated functionalities, which can be a sign that they should be refactored into more focused, cohesive units.

**4) Size**

There are four classes with low-medium size,five classes with low size and only 1 class with medium-high size.

1. Classes having low -medium size(56.7%)

1.1)ConnectResources

1.2)UserResources

1.3)TagResources

1.4)AppResources

2)Classes having low(7.5%)

2.1) ThemeResources

2.2)LocaleResources

2.3) BookResources

2.4)TextPlainMessageBodyWriter

3) Classes having medium-high size(35.8%)

3.1)BookResources

The 'BookResource' class also stands out in terms of size, which correlates with its high complexity. Large classes can be more challenging to understand, test, and maintain.