1   Introduction

Code Review is a systematic examination(also referred as Peer review) of the source code.The
review process is performed to find the mistakes which was ignored in the initial development
phase and arr usually performed to improve the overall quality of the software by improving
internal code quality and maintainability, also find performance problems, security vulnera-
bilities etc [1].

Manual Review

Figure 1: No Meaningful Names Error

Figure 2: Javadoc Comments Absent

A method can have only one return statement and that should be the end of the state-ment.

```
81          base = Double.parseDouble(sa);
82          return base;
83      }
84
85⊝    static int floorCeil(double s,double x,double base) {
86          if(x==1) {
87              return 0;
88          }
89          else if(s==0) {
90              double b1=base;
91              double s1=s+1;
92              for(int v=1;v<=s1;v++) {
93                  base=base*b1;
94              }
95              if(base<x) {
96                  return 1;
97              }
98              else if(base>x) {
99                  return 1;
100             }
101         }
102         else {
103             double b1=base;
104             for(int v=1;v<s;v++) {
105                 base=base*b1;
106             }
107             if(base<x) {
108                 return 1;
109             }
110             else if(base>x) {
111                 return 2;
112             }
113             else {
114                 return 0;
115             }
116         }
117         return -1;
118     }
```

Figure 3: Multiple Return Error

SonarQube

SonarQube is a free and open source static code analysis tool to detect bugs, code smells, security vulnerabilities. SonarQube supports more than 20 programming languages and it has inbuilt rules for most of the languages.I have hosted SonarQube on my system and used it to perform the code review and used the java rule-set(consisted of 351 active rules) to perform the analysis.
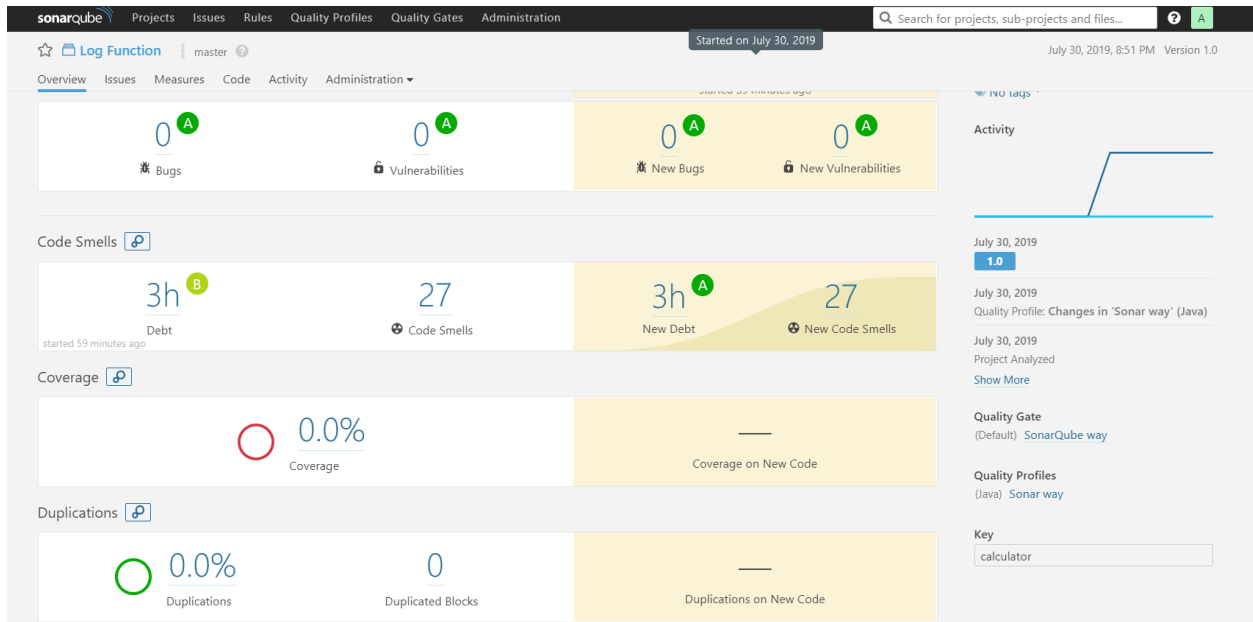
Review Report

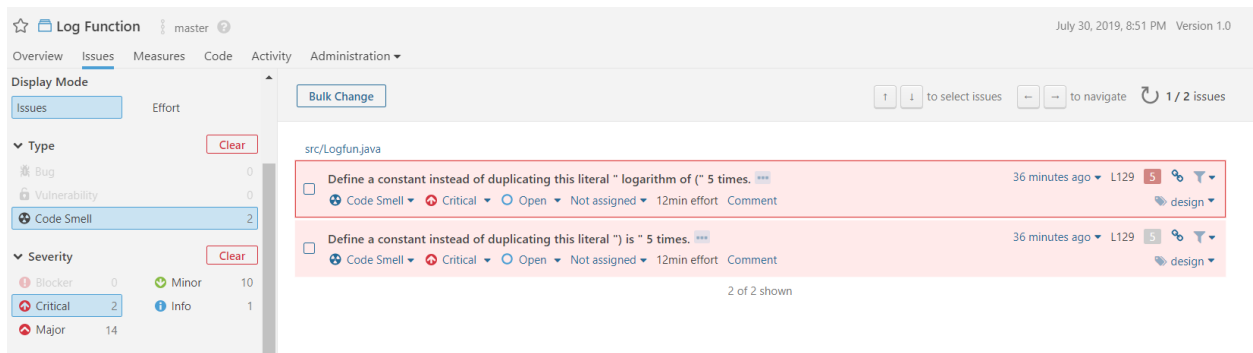

Figure 4: SonarQube Dashboard. (Log Function)
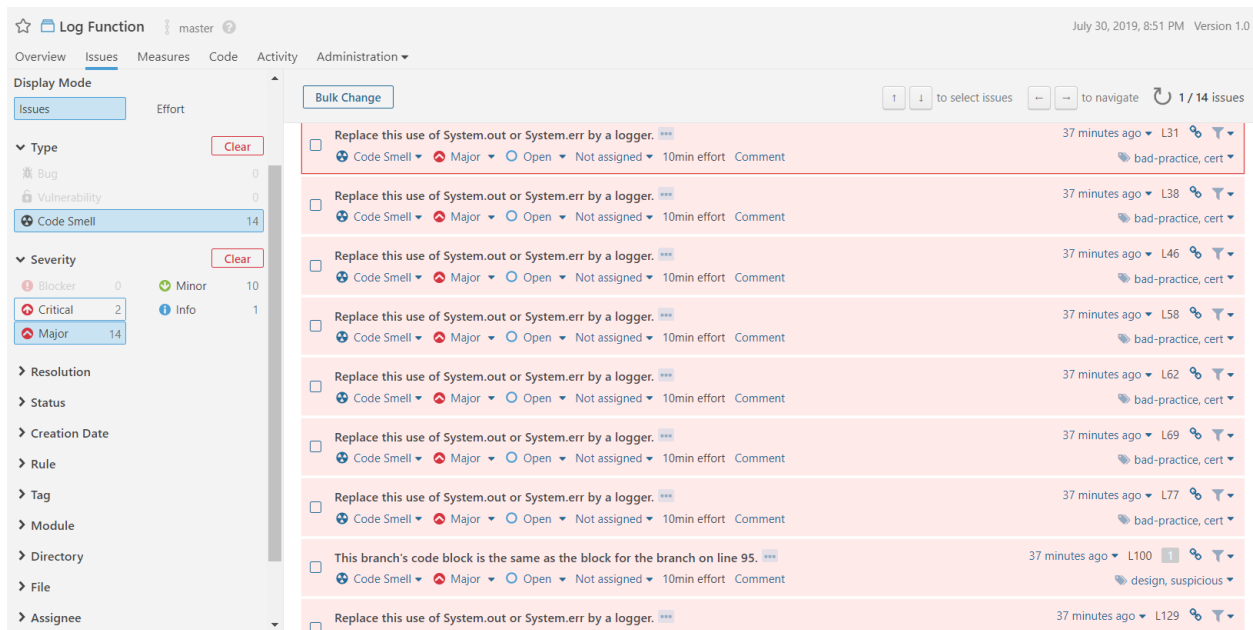

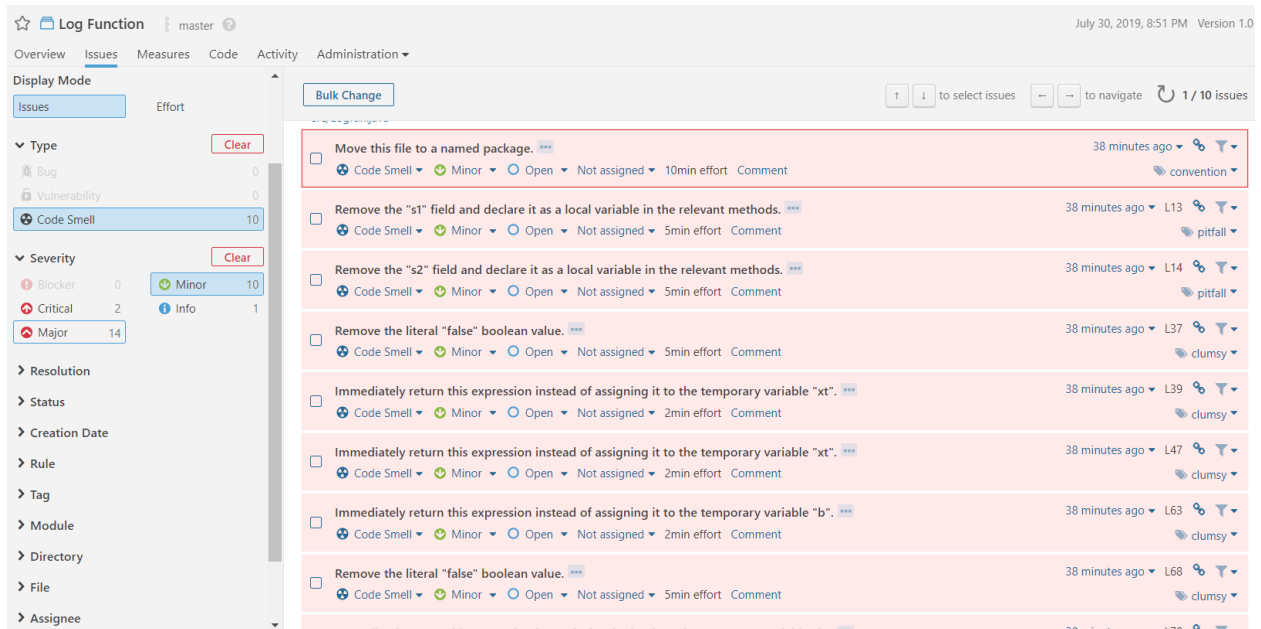Figure 5: Critical Errors
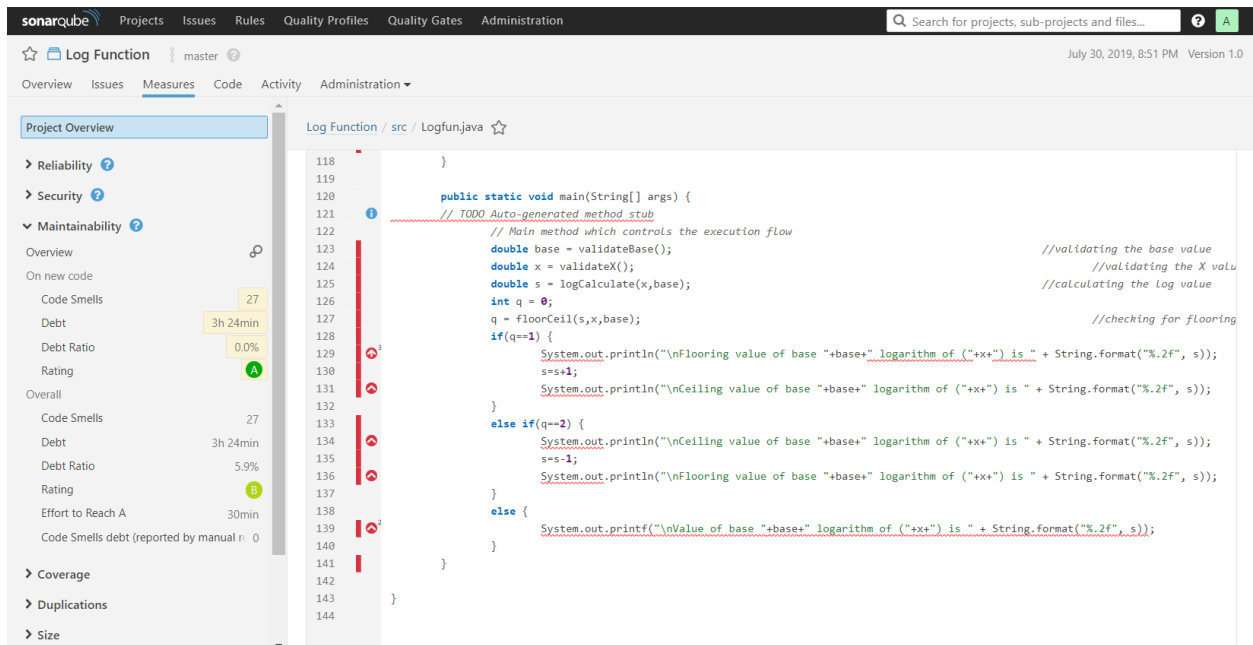
Figure 6: Major Errors

Figure 7: Minor Errors

Figure 8: Function Main Method Implementation
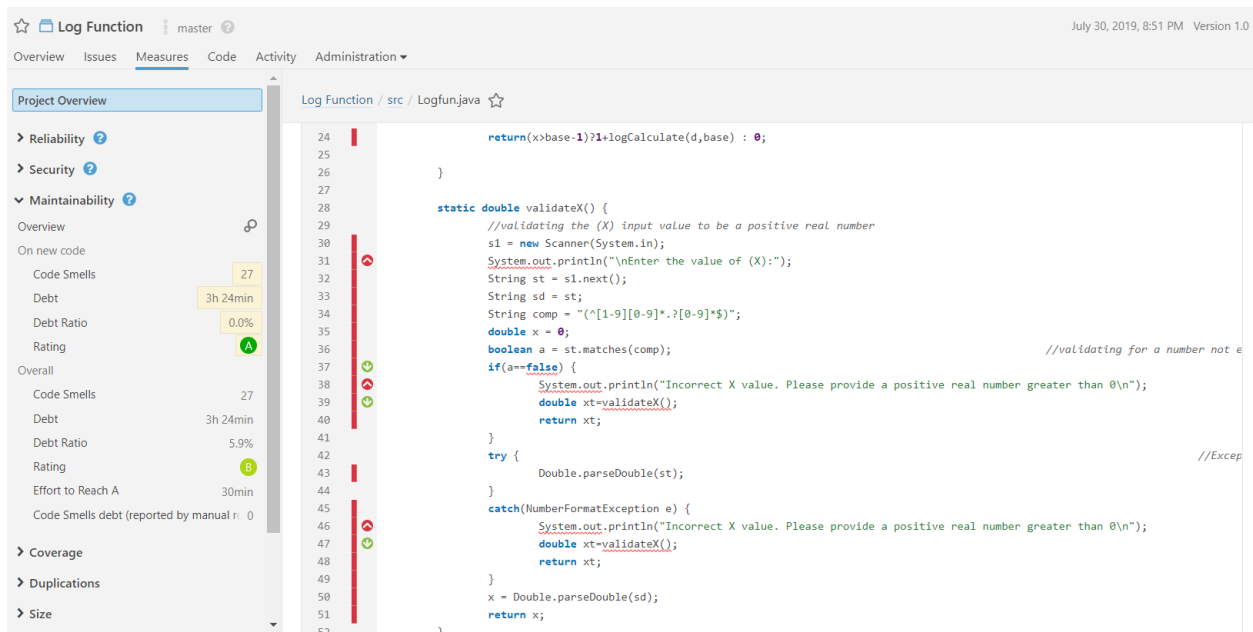
Figure 9: Function Method Implementation

Project Overview

> Reliability ⊘
> Security ⊘
∨ Maintainability ⊘

Overview   🔗
On new code
    Code Smells   27
    Debt   3h 24min
    Debt Ratio   0.0%
    Rating   Ⓐ
Overall
    Code Smells   27
    Debt   3h 24min
    Debt Ratio   5.9%
    Rating   Ⓑ
    Effort to Reach A   30min
    Code Smells debt (reported by manual r 0

> Coverage
> Duplications
> Size

Log Function / src / Logfun.java ☆

```java
54          static double validateBase() {
55              //validating the base input value to be a positive real number
56              double base = 0;
57              s2 = new Scanner(System.in);
58              System.out.println("\nEnter the value of base:");
59              String st1 = s2.next();
60              String sd = st1;
61              if(st1.equalsIgnoreCase("1")) {                                      //base not equal to 1
62                  System.out.println("Incorrect base value. Please provide a positive real number except 1\n");
63                  double b=validateBase();
64                  return b;
65              }
66              String comp = "(^[1-9][0-9]*.?[0-9]*$)";
67              boolean d = st1.matches(comp);                                        //validating for a number not e
68              if(d==false) {
69                  System.out.println("Incorrect base value. Please provide a positive real number greater than 0\n");
70                  double b=validateBase();
71                  return b;
72              }
73              try {                                                                //Excep
74                  Double.parseDouble(st1);
75              }
76              catch(NumberFormatException e) {
77                  System.out.println("Incorrect base value. Please provide a positive real number\n");
78                  double b=validateBase();
79                  return b;
80              }
81              base = Double.parseDouble(sd);
82              return base;
```

Figure 10: Function Method Implementation

Log Function / src / Logfun.java ☆

```java
85          static int floorCeil(double s,double x,double base) {          //checking if ceiling/flooring values exists
86              if(x==-1) {
87                  return 0;
88              }
89              else if(s==0) {
90                  double b1=base;
91                  double s1=s+1;
92                  for(int v=1;v<=s1;v++) {
93                      base=base*b1;
94                  }
95                  if(base<x) {
96                      return 1;
97                  }
98                  else if(base>x) {
99                      return 1;
100                 }
101             }
102             else {
103                 double b1=base;
104                 for(int v=1;v<s;v++) {
105                     base=base*b1;
106                 }
107                 if(base<x) {
108                     return 1;
109                 }
110                 else if(base>x) {
111                     return 2;
112                 }
113                 else {
```
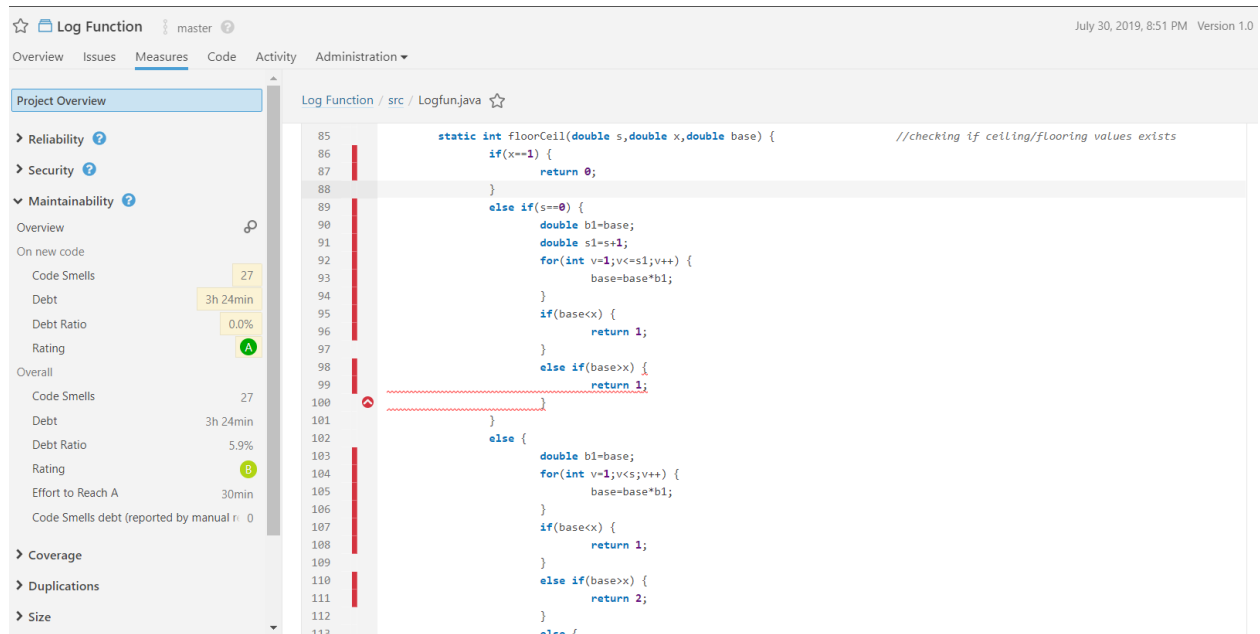
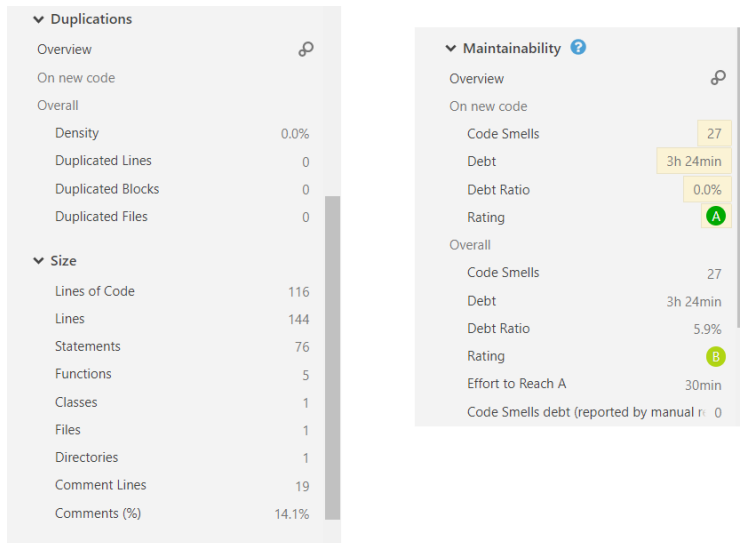Figure 11: Function Method Implementation

Figure 12: Quality Measures

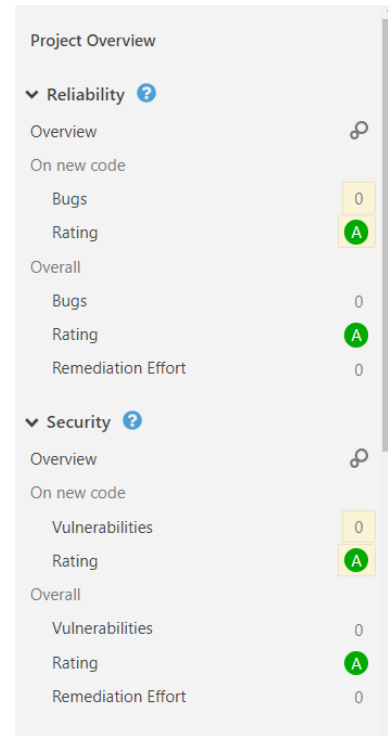

Figure 13: Quality Measures
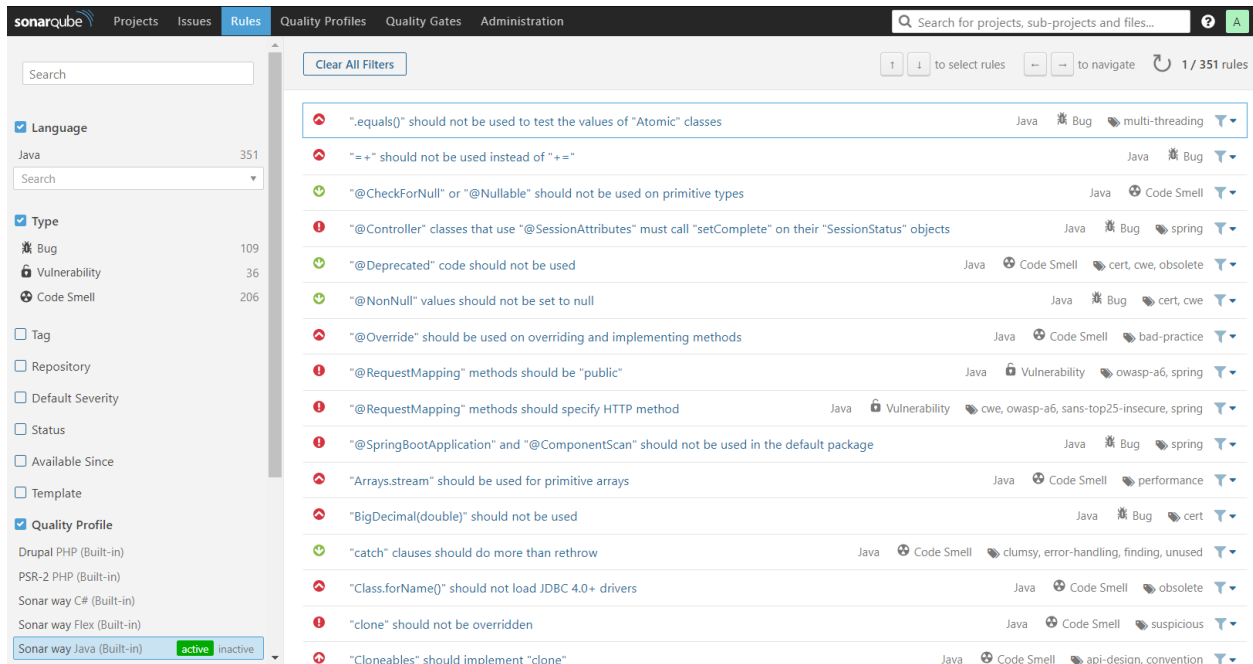

Figure 14: SonarQube Java Rules

Github

This is the Link to my project repository : F3: sinh(x) or you can go to the url : `https://github.com/Ruthvik-Shandilya/SOEN-6011`.

This is the Link to the reviewed project repository : F4: logb(x) or you can go to the url : `https://github.com/manikandan-ms/SOEN6011`.

References

[1] Code Review,
    texttthttps://en.wikipedia.org/wiki/Codereview