Student Information

First Name: Abdullah

Last Name: Elmi

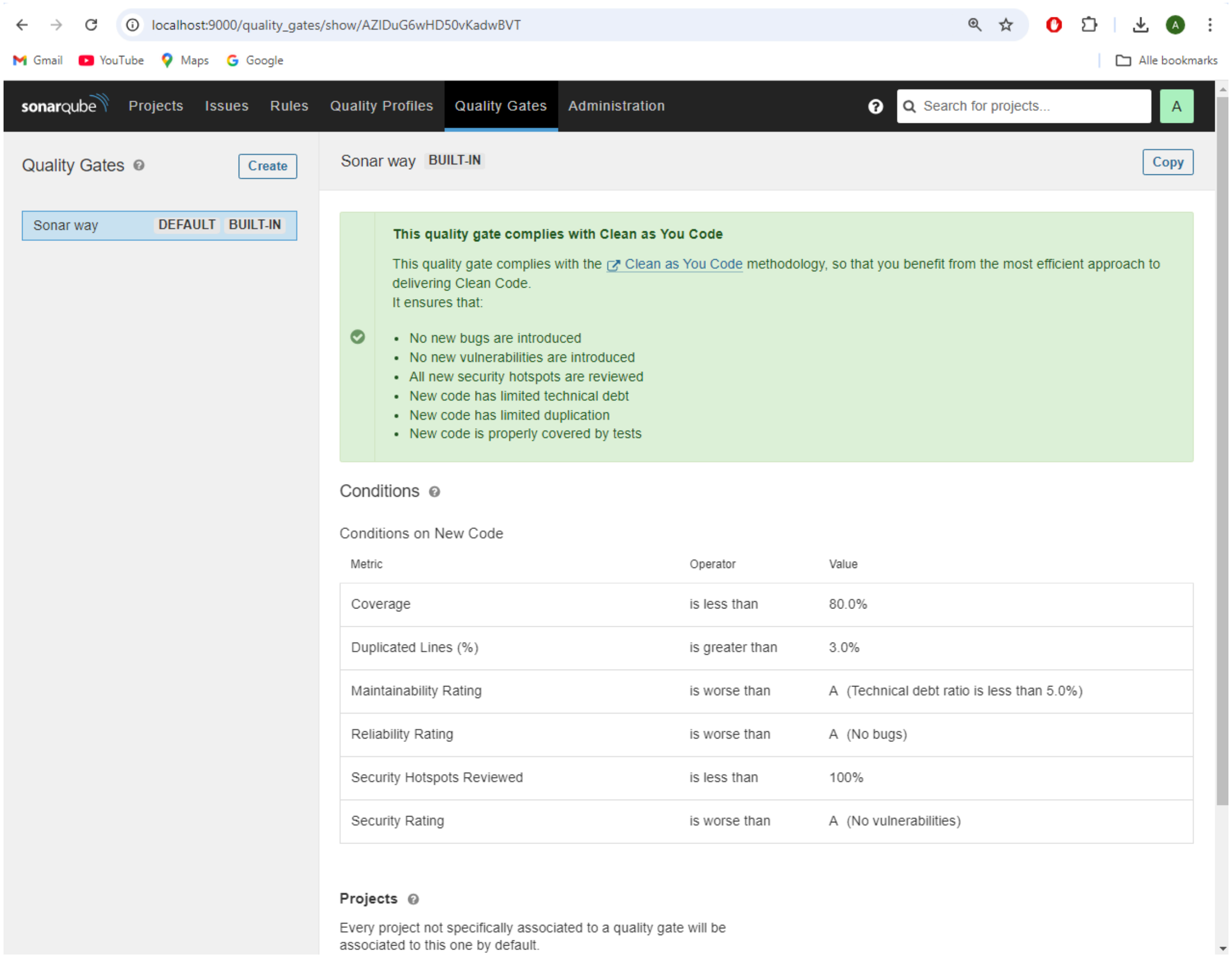
Student number: 500908247

Assignment 1

1. Git log

```
27b4fc1x 2025-09-14     chore: updated pictures to change comments on GitLab
185b42dx 2025-09-14     chore: added new photos to put in README.md
6b305aax 2025-09-14     Merge branch 'feat/bsn-validator'
d18afd3x 2025-09-14     chore: added a png and changed the README.md
054cbb7x 2025-09-14     Merge branch 'main' of https://gitlab.fdmci.hva.nl/se-specialization-25-26-semester-1/tse2
49487f7x 2025-09-14     chore: reset main to tests-only state for review
cf84898x 2025-09-14     chore: update pom.xml to see the SonarQube Coverage
5a0a827x 2025-09-14     chore: ignore build and IDE files; untrack target/, .idea/, *.iml
169952ax 2025-09-14     chore: update .gitignore to exclude target directory
894b569x 2025-09-14     test: update BSN validator tests with valid 8-digit example
3fadfb9x 2025-09-14     feat: implement BSN validator logic (11-proef & input checks)
a84b08ex 2025-09-14     test: add failing BSN validator tests (valid/invalid cases)
bfabfa8x 2025-09-14     Initial commit
```

2. Sonarqube



This quality gate complies with Clean as You Code

This quality gate complies with the Clean as You Code methodology, so that you benefit from the most efficient approach to delivering Clean Code.
It ensures that:

- No new bugs are introduced
- No new vulnerabilities are introduced
- All new security hotspots are reviewed
- New code has limited technical debt
- New code has limited duplication
- New code is properly covered by tests

Conditions

Conditions on New Code

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A (Technical debt ratio is less than 5.0%)
Reliability Rating	is worse than	A (No bugs)
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A (No vulnerabilities)

Projects

Every project not specifically associated to a quality gate will be associated to this one by default.

The SonarQube analysis shows that my project passed the Quality Gate with the highest possible rating (A) for Reliability, Security, and Maintainability. There are no bugs, vulnerabilities, or code smells detected. The test coverage is 97%, which means almost all lines in the BsnValidator class are tested. In addition, the analysis shows 0% code duplication, which indicates that the implementation is clean and without redundant code.

3. Test Driven Development

Your best test class code snippets with a rationale why the unit tests are “good” tests. Provide a link to the Test class and the class under test in Git.

```
 @Nested
@DisplayName("Valid BSNs")
class ValidCases {
    @ParameterizedTest
    @DisplayName("Accepts known valid BSNs (11-proef, 9 digits)")
    @ValueSource(strings = {"286497153", "459980889", "808026598"})
    void validSamplesPass(String bsn) {
        assertTrue(validator.isValid(bsn));
    }

    @ParameterizedTest
    @DisplayName("8-digit BSNs are validated with an implied leading zero")
```

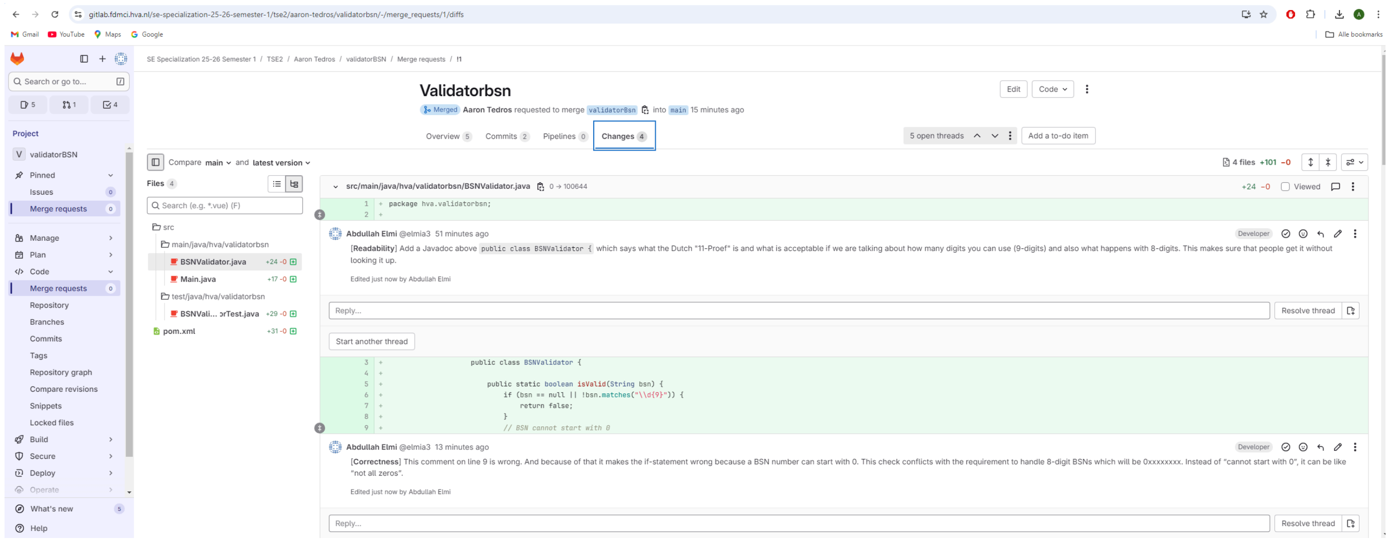
```
        @ValueSource(strings = {"72124581"})
            // This should be 072124581 and valid!
        void eightDigitInputsPass(String eightDigits) {
            assertTrue(validator.isValid(eightDigits));
        }
    }

    @Nested
    @DisplayName("Invalid inputs")
    class InvalidCases {
        @ParameterizedTest
        @DisplayName("Rejects bad formats and lengths")
        @ValueSource(strings = {"", "   ", "abc^%$gdebhe", "123-456-789","1234567890"})
        void badFormatsFail(String bsn) {
            assertFalse(validator.isValid(bsn));
        }

        @ParameterizedTest
        @DisplayName("Rejects numbers that fail the 11-proef")
        @ValueSource(strings = {"111111111", "123456789", "000000001"})
        void checksumFails(String bsn) {
            assertFalse(validator.isValid(bsn));
        }

        @ParameterizedTest
        @DisplayName("All zeros are invalid (8 or 9 digits)")
        @ValueSource(strings = {"00000000", "000000000"})
        void zerosAreInvalid(String bsn) {
            assertFalse(validator.isValid(bsn));
        }
    }
```

Class under test

Test class

4. Code Reviews

These are the screenshots of the code review I have done in GitLab on another student which is Aaron Tedros.



gitlab.fdmci.hva.nl/se-specialization-25-26-semester-1/tse2/aaron-tedros/validatorbsn/-/merge_requests/1/diffs

SE Specialization 25-26 Semester 1 / TSE2 / Aaron Tedros / validatorBSN / Merge requests / !1

Validatorbsn

Merged Aaron Tedros requested to merge `validatorBsn` into `main` 15 minutes ago

Overview 5 Commits 2 Pipelines 0 Changes 4

5 open threads | Add a to-do item

Compare main and latest version — 4 files +101 −0

Files 4
- src
  - main/java/hva/validatorbsn
    - BSNValidator.java +24 -0
    - Main.java +17 -0
  - test/java/hva/validatorbsn
    - BSNVali…orTest.java +29 -0
- pom.xml +31 -0

src/main/java/hva/validatorbsn/BSNValidator.java 0 → 100644 +24 −0

```
1 + package hva.validatorbsn;
2 +
```

Abdullah Elmi @elmia3 51 minutes ago — Developer

[**Readability**] Add a Javadoc above `public class BSNValidator {` which says what the Dutch "11-Proef" is and what is acceptable if we are talking about how many digits you can use (9-digits) and also what happens with 8-digits. This makes sure that people get it without looking it up.

Edited just now by Abdullah Elmi

Reply… | Resolve thread

Start another thread

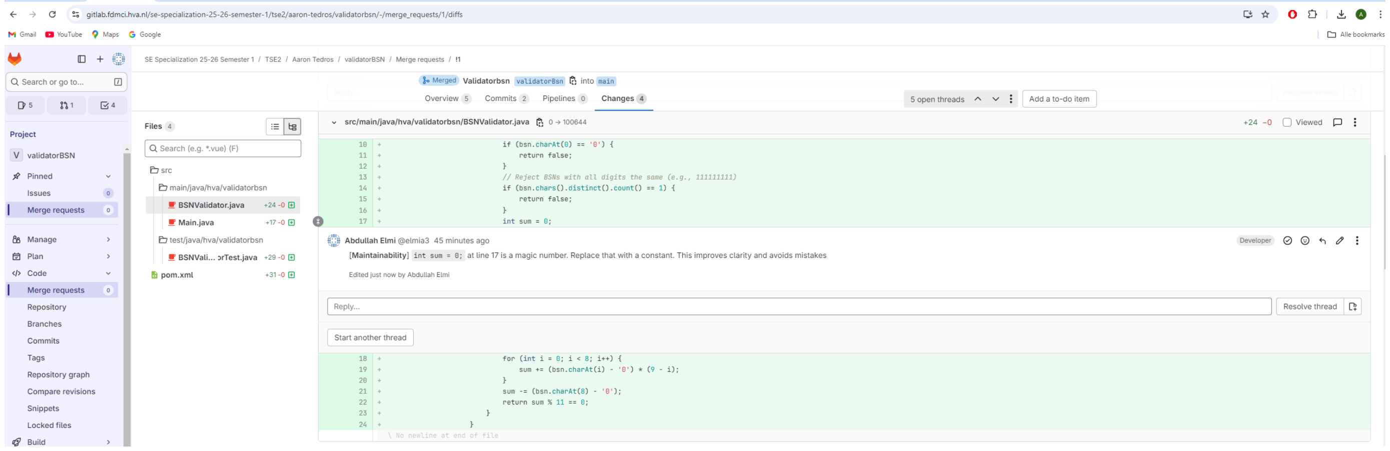
```
3 +                 public class BSNValidator {
4 +
5 +                     public static boolean isValid(String bsn) {
6 +                         if (bsn == null || !bsn.matches("\\d{9}")) {
7 +                             return false;
8 +                         }
9 +                         // BSN cannot start with 0
```

Abdullah Elmi @elmia3 13 minutes ago — Developer

[**Correctness**] This comment on line 9 is wrong. And because of that it makes the if-statement wrong because a BSN number can start with 0. This check conflicts with the requirement to handle 8-digit BSNs which will be 0xxxxxxxx. Instead of "cannot start with 0", it can be like "not all zeros".

Edited just now by Abdullah Elmi

Reply… | Resolve thread



gitlab.fdmci.hva.nl/se-specialization-25-26-semester-1/tse2/aaron-tedros/validatorbsn/-/merge_requests/1/diffs

SE Specialization 25-26 Semester 1 / TSE2 / Aaron Tedros / validatorBSN / Merge requests / !1

Merged Validatorbsn `validatorBsn` into `main`

Overview 5 Commits 2 Pipelines 0 Changes 4

5 open threads | Add a to-do item

src/main/java/hva/validatorbsn/BSNValidator.java 0 → 100644 +24 −0

```
10 +                         if (bsn.charAt(0) == '0') {
11 +                             return false;
12 +                         }
13 +                         // Reject BSNs with all digits the same (e.g., 111111111)
14 +                         if (bsn.chars().distinct().count() == 1) {
15 +                             return false;
16 +                         }
17 +                         int sum = 0;
```

Abdullah Elmi @elmia3 45 minutes ago — Developer

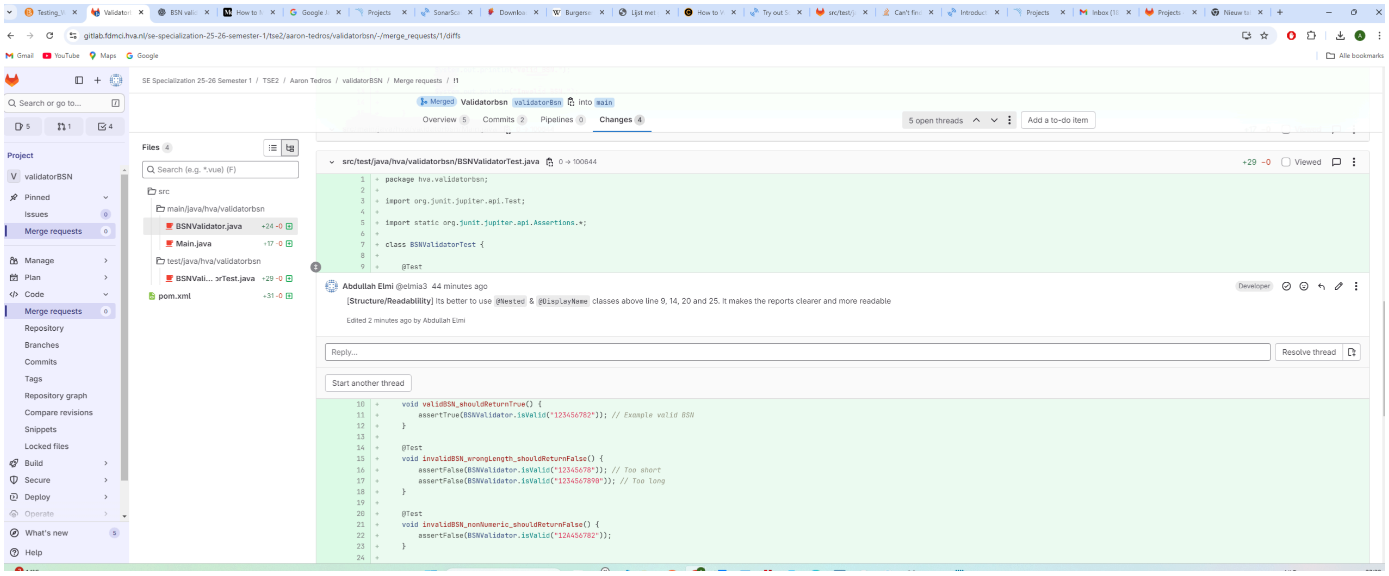
[**Maintainability**] `int sum = 0;` at line 17 is a magic number. Replace that with a constant. This improves clarity and avoids mistakes

Edited just now by Abdullah Elmi

Reply… | Resolve thread

Start another thread

```
18 +                         for (int i = 0; i < 8; i++) {
19 +                             sum += (bsn.charAt(i) - '0') * (9 - i);
20 +                         }
21 +                         sum -= (bsn.charAt(8) - '0');
22 +                         return sum % 11 == 0;
23 +                     }
24 +                 }
   \ No newline at end of file
```



gitlab.fdmci.hva.nl/se-specialization-25-26-semester-1/tse2/aaron-tedros/validatorbsn/-/merge_requests/1/diffs

SE Specialization 25-26 Semester 1 / TSE2 / Aaron Tedros / validatorBSN / Merge requests / !1

Merged Validatorbsn `validatorBsn` into `main`

Overview 5 Commits 2 Pipelines 0 Changes 4 — 5 open threads — Add a to-do item

Files 4: src / main/java/hva/validatorbsn — BSNValidator.java +24 -0, Main.java +17 -0; test/java/hva/validatorbsn — BSNVali…orTest.java +29 -0; pom.xml +31 -0

src/test/java/hva/validatorbsn/BSNValidatorTest.java 0 → 100644 +29 −0

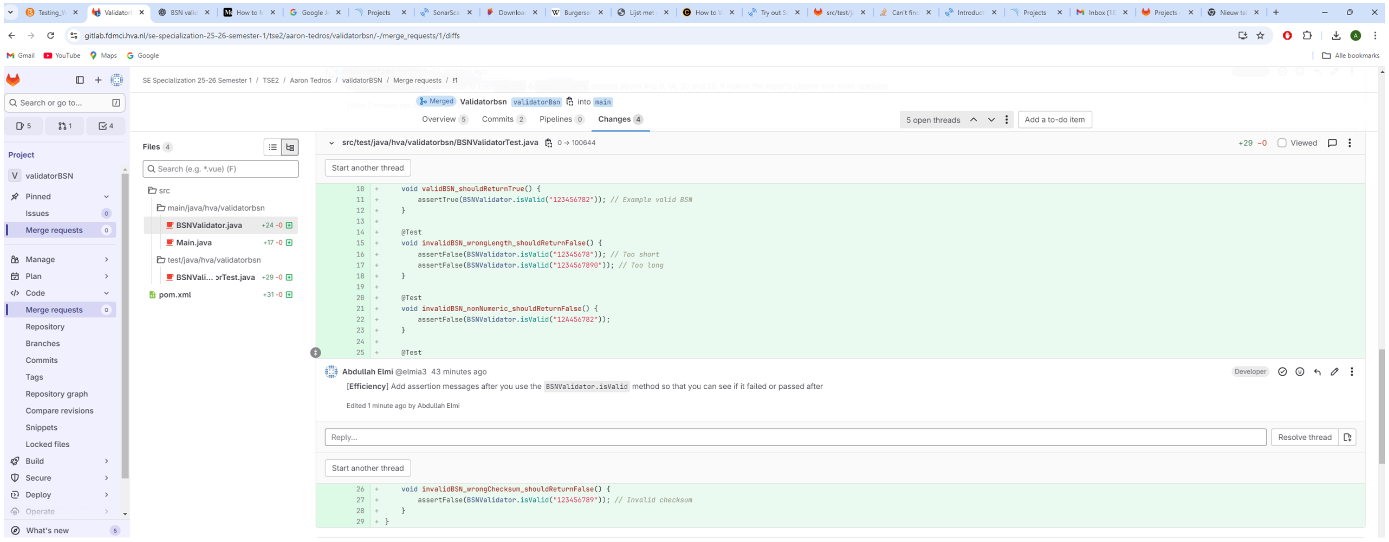
```
 1 + package hva.validatorbsn;
 2 +
 3 + import org.junit.jupiter.api.Test;
 4 +
 5 + import static org.junit.jupiter.api.Assertions.*;
 6 +
 7 + class BSNValidatorTest {
 8 +
 9 +     @Test
```

Abdullah Elmi @elmia3 44 minutes ago — Developer

[**Structure/Readablility**] Its better to use `@Nested` & `@DisplayName` classes above line 9, 14, 20 and 25. It makes the reports clearer and more readable

Edited 2 minutes ago by Abdullah Elmi

```
10 +     void validBSN_shouldReturnTrue() {
11 +         assertTrue(BSNValidator.isValid("123456782")); // Example valid BSN
12 +     }
13 +
14 +     @Test
15 +     void invalidBSN_wrongLength_shouldReturnFalse() {
16 +         assertFalse(BSNValidator.isValid("12345678")); // Too short
17 +         assertFalse(BSNValidator.isValid("1234567890")); // Too long
18 +     }
19 +
20 +     @Test
21 +     void invalidBSN_nonNumeric_shouldReturnFalse() {
22 +         assertFalse(BSNValidator.isValid("12A456782"));
23 +     }
24 +
```



src/test/java/hva/validatorbsn/BSNValidatorTest.java 0 → 100644 +29 −0

```
10 +     void validBSN_shouldReturnTrue() {
11 +         assertTrue(BSNValidator.isValid("123456782")); // Example valid BSN
12 +     }
13 +
14 +     @Test
15 +     void invalidBSN_wrongLength_shouldReturnFalse() {
16 +         assertFalse(BSNValidator.isValid("12345678")); // Too short
17 +         assertFalse(BSNValidator.isValid("1234567890")); // Too long
18 +     }
19 +
20 +     @Test
21 +     void invalidBSN_nonNumeric_shouldReturnFalse() {
22 +         assertFalse(BSNValidator.isValid("12A456782"));
23 +     }
24 +
25 +     @Test
```

Abdullah Elmi @elmia3 43 minutes ago — Developer

[**Efficiency**] Add assertion messages after you use the `BSNValidator.isValid` method so that you can see if it failed or passed after

Edited 1 minute ago by Abdullah Elmi

```
26 +     void invalidBSN_wrongChecksum_shouldReturnFalse() {
27 +         assertFalse(BSNValidator.isValid("123456789")); // Invalid checksum
28 +     }
29 + }
```

GitLab Comments Link