

Team 5 - Code review report  
(Emil, Emmi, Musa, Verner)

## Objective

As a team, we conducted a code review using SonarQube and code coverage metrics, also for proper practice we used a default ESLint configuration and a common formatting and naming etiquette. The findings were meant to increase maintainability and readability of code.

## Summary

Tool	Report
Code Coverage	71.1%
SonarQube - Reliability	B - 110 open issues
SonarQube - Maintainability	A - 153 open issues
SonarQube - Duplications	0.0%
SonarQube - Security	A - 0 open issues

## Common & Priority Issues

### Reliability:

1. Missing prop validation
  - Reliability and maintainability low to medium priority
  - When passing props to components it is good practice to validate the existence of the prop
2. Visible, non-interactive elements with click handlers must have at least one keyboard listener.
  - Reliability low priority
  - Non interactive elements are not meant to have click handlers, causes readability issues

### Maintainability:

1. Components imported multiple times
  - Maintainability low priority
  - Affects readability
2. Multiple checks instead of optional chaining (?)
  - Maintainability medium priority
  - Makes code more readable and concise
3. Unused imports
  - Maintainability low priority
  - Takes up less space

Examples:

## 1. SonarQube report

The screenshot shows the SonarQube Main Branch Summary report for the 'main' branch. The summary indicates a 'Passed' status with 4k Lines of Code, Version 1.0, and a last analysis 1 hour ago. The Quality Gate is 'Sonar way'. The report includes sections for Security, Reliability, Maintainability, Accepted Issues, Coverage, Duplications, and Security Hotspots. A large green checkmark icon is prominently displayed.

Category	Value	Status
Security	0 Open issues	A
Reliability	110 Open issues	B
Maintainability	153 Open issues	A
Accepted Issues	0	C
Coverage	71.1%	D
Duplications	0.0%	E
Security Hotspots	3	F

## 2. Prop validation

The screenshot shows the prop validation interface for the file 'src/components/editor/ImageComponent.jsx'. It lists several issues related to prop validation, each with a checkbox, severity buttons (Reliability: Low/Medium, Maintainability: Low/Medium), and a 'Consistency' button. The issues are:

- 'node' is missing in props validation (Reliability: Low, Maintainability: Medium)
- 'deleteNode' is missing in props validation (Reliability: Low, Maintainability: Medium)
- 'updateAttributes' is missing in props validation (Reliability: Low, Maintainability: Medium)
- 'node.attrs' is missing in props validation (Reliability: Low, Maintainability: Medium)
- 'node.attrs.width' is missing in props validation (Reliability: Low, Maintainability: Medium)
- 'node.attrs' is missing in props validation (Reliability: Low, Maintainability: Medium)

Each issue row includes a 'Select issues' dropdown, navigation buttons, and a timestamp (e.g., 1 month ago). A summary at the top right indicates 110 issues, 1d 1h effort, and a 'react' context.

### 3. Missing non interactive elements

The screenshot shows a code review interface with several issues listed:

- L64 · 5min effort · 1 month ago · ⚡ Code Smell · ⚠ Major  
'node.attrs.title' is missing in props validation  
Reliability Low Maintainability 🌐 Medium  
react +
- L64 · 5min effort · 1 month ago · ⚡ Code Smell · ⚠ Major  
Avoid non-native interactive elements. If using native HTML is not possible, add an appropriate role and support for tabbing, mouse, keyboard, and touch inputs to an interactive content element.  
Reliability 🌐 Medium Maintainability Low  
accessibility react +
- L85 · 5min effort · 1 month ago · ⚡ Code Smell · ⚠ Major  
src/components/editor/ImageUploadButton.jsx  
'editor' is missing in props validation  
Reliability Low Maintainability 🌐 Medium  
react +
- L6 · 5min effort · 1 month ago · ⚡ Code Smell · ⚠ Major  
'editor.chain' is missing in props validation  
Consistency

### 4. Example error

The screenshot shows a code editor with a specific line highlighted:

```
79
80           > <Trash2 size={10} />
81           </button>
82         </div>
83
84         /* Resize handle */
85       <div
86         style={{ ... }}
```

A tooltip box appears over the highlighted line:

Avoid non-native interactive elements. If using native HTML is not possible, add an appropriate role and support for tabbing, mouse, keyboard, and touch inputs to an interactive content element.

The line of code is:

```
86         className={`
87           absolute bottom-0 right-0 w-4 h-4 bg-blue-500 border-2 border-white rounded-tl-lg cursor-nw-resize opacity-0 group-hover-opacity-100
88           transition-opacity
89           isResizing ? 'opacity-100' : ''
90           onMouseDown={handleMouseDown}
91           title="Resize Image"
92         }
93         >Move size={12} className="absolute top-0 left-0 text-white" />
94       </div>
95     </div>
96   </NodeViewWrapper>
97 }
98 }
```

### 5. Sample solution

The screenshot shows a code review interface with a sample solution section:

**Avoid non-native interactive elements. If using native HTML is not possible, add an appropriate role and support for tabbing, mouse, keyboard, and touch inputs to an interactive content element.** ↗

Non-interactive DOM elements should not have an interactive handler [javascript:S6848](#)

Software qualities impacted: Reliability 🌐 Medium Maintainability Low

Tags: accessibility ...  
Line affected: L91  
Effort: 5 min  
Introduced: 1 month ago

Open · Not assigned · ⚡ Code Smell · ⚠ Major

Where is the issue? Why is this an issue? How can I fix it? Activity More info Open in IDE

The sample solution provides the following code examples:

**Noncompliant code example**

```
<div onClick={() => {}} /> // Noncompliant
```

**Compliant solution**

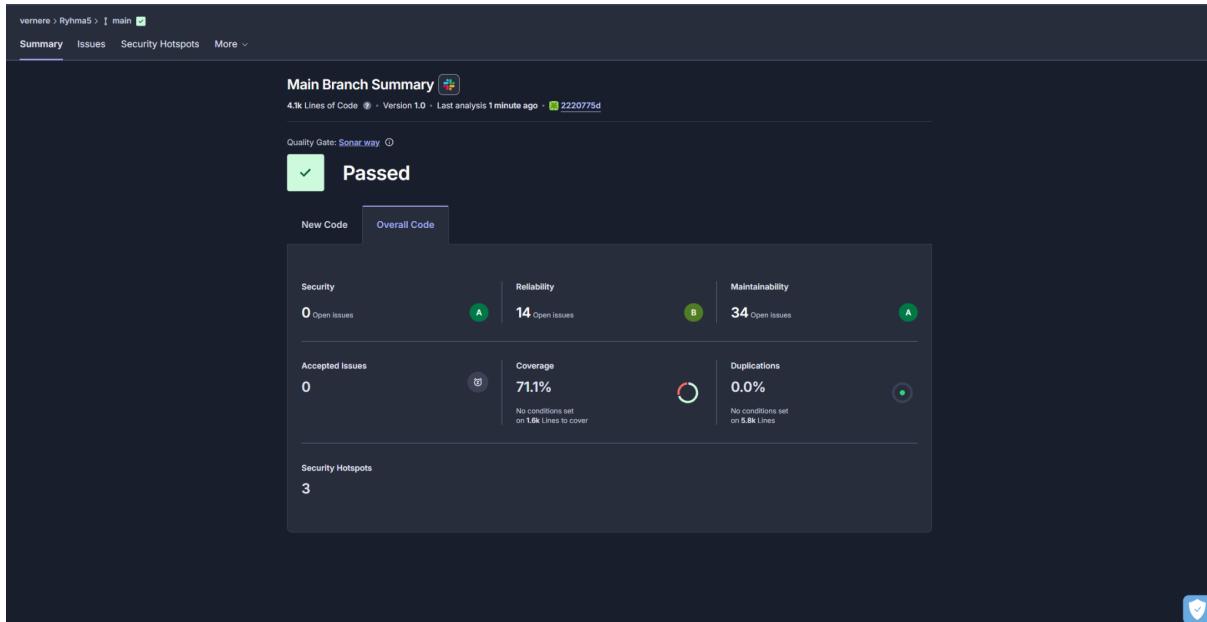
```
<div onClick={() => {}} role="button" />
```

### **Short-Term Actions(completed):**

- Added prop validation to all components that get passed props
- Upped code readability through removing duplicate imports
- Changed non-interactive elements with click handlers to have at least one keyboard listener

### **Long-Term Improvements:**

- Refactor collaboration invitation logic for better reliability



### **Conclusion:**

From the beginning the code was designed and implemented with clear intent. As a result the code quality was reasonably good starting the cleanup. With the cleanup and refactoring, issues were reduced from 263 to 48 ( 81.75% reduction), which can be considered a successful result. These corrections improved the software's reliability and maintainability.