CodeQual Analysis Report

Pull Request Analysis Report

Confidence

25%

PR Decision: Blocked

This PR cannot be merged due to critical security issues.

Blocking Issues

- CRITICAL SQL Injection vulnerability in auth module
- HIGH Exposed API keys in configuration

Positive Findings

- ✅ Good test coverage (85%)
- ✅ Follows established coding patterns
- ✅ Proper error handling implemented

Current PR Issues

CRITICAL

SQL Injection Vulnerability

User input is directly concatenated into SQL query without sanitization.

```
const query = "SELECT * FROM users WHERE id = " + userId;
```

> **Recommendation:** Use parameterized queries or prepared statements.

HIGH

Hardcoded API Keys

API keys are exposed in the source code.

```
const API_KEY = 'sk_live_abcd1234';
```

> **Recommendation:** Store API keys in environment variables and use a secrets management system.

Repository Issues

Existing issues in your codebase that need attention

CRITICAL

SQL Injection Vulnerability

Direct SQL query construction with user input allows SQL injection attacks.

```
const query = "SELECT * FROM users WHERE id = " + req.params.id;
db.execute(query);
```

> **Recommendation:** Use parameterized queries or prepared statements to prevent SQL injection.

CRITICAL

Hardcoded API Keys Exposed

Sensitive API keys are hardcoded in the source code and exposed in version control.

```
const API_KEY = "sk-proj-abcd1234efgh5678ijkl9012mnop3456"; const
stripe = new Stripe(API_KEY);
```

> **Recommendation:** Store sensitive credentials in environment variables and never commit them to version control.

HIGH

Memory Leak in Event Handlers

Event listeners are not properly cleaned up, causing memory leaks.

```
window.addEventListener('resize', handleResize); // Missing:
window.removeEventListener('resize', handleResize);
```

> **Recommendation:** Always remove event listeners in cleanup functions or component unmount lifecycle.

HIGH

Unvalidated User Input

User input is being used without proper validation.

```
const userInput = req.body.search; db.query(`SELECT * FROM
products WHERE name LIKE '%${userInput}%'`);
```

> **Recommendation:** Validate and sanitize all user input before using it in queries.

HIGH

Cross-Site Scripting (XSS) Risk

User content is rendered without proper escaping, allowing XSS attacks.

```
document.getElementById('output').innerHTML = userComment; //
Should use textContent or properly escape HTML
```

> **Recommendation:** Always escape user-generated content before rendering it in HTML or use safe DOM methods like textContent.

MEDIUM

Complex Function

Function has cyclomatic complexity of 15

```
function complexFunc() { /* 50+ lines */ }
```

> **Recommendation:** Break down complex functions.

MEDIUM

Long Method

Method exceeds 50 lines

```
public void longMethod() { /* many lines */ }
```

> **Recommendation:** Split into smaller methods.

MEDIUM

Deep Nesting

Code nesting depth exceeds 4 levels

```
if (a) { if (b) { if (c) { if (d) { } } } }
```

> **Recommendation:** Refactor to reduce nesting.

MEDIUM

Duplicate Code

Similar code found in 3 locations

```
const result = data.map(x => x * 2);
```

> **Recommendation:** Extract common functionality.

MEDIUM

Missing Tests

No tests for critical functions

```
// No test coverage for this function
```

> **Recommendation:** Add unit tests.

MEDIUM

Performance Issue

Inefficient loop in hot path

```
for (let i = 0; i < arr.length; i++) { arr.filter() }
```

> **Recommendation:** Optimize algorithm.

MEDIUM

Error Handling

Missing error handling in async code

```
async function() { await fetch(url); }
```

> **Recommendation:** Add try-catch blocks.

LOW

Mixed Quotes

Inconsistent quote usage

```
const a = "hello"; const b = 'world';
```

> **Recommendation:** Use single quotes consistently.

LOW

Inconsistent Naming

Variable naming convention

```
const user_name = "John";
```

> **Recommendation:** Use camelCase consistently.

LOW

Missing Comments

Complex logic without documentation

```
function xyz(a, b, c) { return a * b + c; }
```

> **Recommendation:** Add explanatory comments.

LOW

Unused Variable

Variable declared but never used

```
const unusedVar = "never used";
```

> **Recommendation:** Remove unused code.

LOW

Magic Numbers

Hard-coded values without context

```
if (value > 86400) { return true; }
```

> **Recommendation:** Use named constants.

LOW

Long Lines

Lines exceeding 120 characters

```
const veryLongLine = "This is a very long line that exceeds the
recommended character limit and should be broken down";
```

> **Recommendation:** Break into multiple lines.

LOW

Missing Semicolons

Inconsistent semicolon usage

```
const a = 1 const b = 2
```

> **Recommendation:** Use semicolons consistently.

LOW

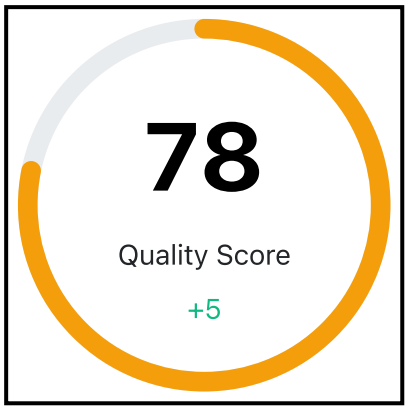
Trailing Spaces

Whitespace at end of lines

```
const text = "hello world"
```

> **Recommendation:** Remove trailing whitespace.

Quality Metrics



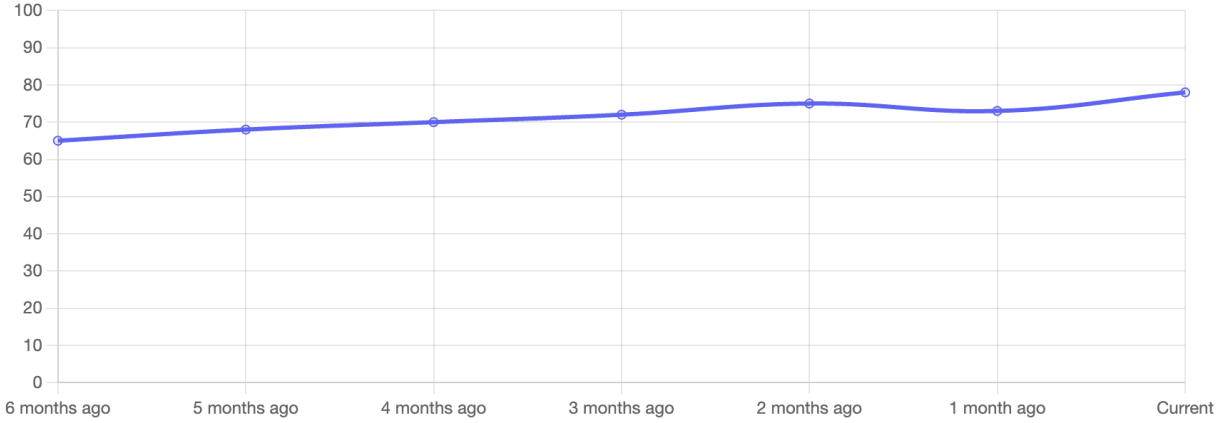
0-40: Poor

41-60: Fair

61-80: Good

81-100: Excellent

Score Trend



Skills Assessment

Security	65/100
Code Quality	82/100
Performance	78/100
Architecture	90/100

Improvement Suggestions

Improve Security Skills — Focus on learning about input validation and secure coding practices.

Performance Optimization — Learn about memory management and efficient algorithms.

Educational Resources

Estimated learning time: 45 minutes

Preventing SQL Injection

Learn how to protect your application from SQL injection attacks.

Start Learning →

🔐 Secure API Key Management

Best practices for storing and using API keys securely.

Start Learning →

💾 Memory Management in JavaScript

Understanding and preventing memory leaks in your applications.

Start Learning →

PR Comment Preview

```
## CodeQual Analysis Report

**Decision:** 🚫 Blocked

This PR cannot be merged due to critical security issues:

### 🚨 Critical Issues (2)
- **SQL Injection vulnerability** in auth module
- **Exposed API keys** in configuration

### ✅ Positive Findings
- Good test coverage (85%)
- Follows established coding patterns
- Proper error handling implemented

### 📊 Code Quality Score: 78/100

⛔ These critical security issues must be resolved before this PR
can be merged.
```

CodeQual v2.1.0

GitHub Documentation Send Feedback

Generated on 7/2/2025, 9:59:03 PM | Report ID: abc-123-def-456