# Reviewing Vibe Code

Ensuring Quality and Security in Node.js & React.js

Presented by: Sh Julkar Naen Nahian

Date: September 7, 2025



# Introduction

### What is Vibe Code?

Code generated by AI tools.

### Why Review It?

- Risks: Bugs, security flaws, or non-idiomatic code.
- Ensures alignment with project standards and performance needs.

### Objective

Create a robust review process for Vibe Code.

Made with GAMMA

# Challenges with Vibe Code (Node.js/React.js)

## Node.js Issues

- Improper async/await handling or callback hell.
- Security risks like unvalidated inputs or vulnerable npm packages.

## React.js Issues

- Inefficient component rendering or missing key props.
- State management errors or outdated hooks usage.

Context Gaps: AI may miss project-specific conventions (e.g., Redux patterns, Express middleware).

Made with GAMMA

# The Review Process

## Step-by-Step for Node.js & React.js

01

### Static Analysis

- Node.js: Run ESLint with eslint-plugin-node for API routes, middleware.
- React.js: Use eslint-plugin-react, eslint-plugin-react-hooks for components.

02

### Functional Review

- Node.js: Test API endpoints with tools like Postman or Jest.
- React.js: Verify component behavior, props, and state updates.

03

### Security Check

- Node.js: Scan for vulnerabilities with npm audit or Snyk.
- React.js: Check for XSS risks in dynamic rendering or user inputs.

04

### Performance Evaluation

- Node.js: Optimize event loop blocking or database queries.
- React.js: Use React DevTools to detect unnecessary re-renders.

05

### Code Style

- Ensure Prettier and ESLint enforce consistent style.
- Follow Airbnb’s JavaScript/React style guide.

Made with GAMMA

# Best Practices

## Node.js

- Validate inputs using libraries like Joi or express-validator.
- Use environment variables (dotenv) for sensitive data.
- Write unit tests with Jest/Mocha for API logic.

## React.js

- Ensure proper use of hooks (e.g., useEffect, useState).
- Optimize with memoization (React.memo, useMemo).
- Test components with React Testing Library.

## General

- Use GitHub Pull Requests for peer reviews.
- Document recurring issues to refine AI prompts.

Made with GAMMA

# Tools for Vibe Code Review



## Static Analysis

ESLint (eslint-plugin-react, eslint-plugin-node), Prettier.



## Security

Snyk, npm audit, OWASP Dependency-Check.



## Testing

Jest, Mocha, React Testing Library, Cypress (for E2E).



## Performance

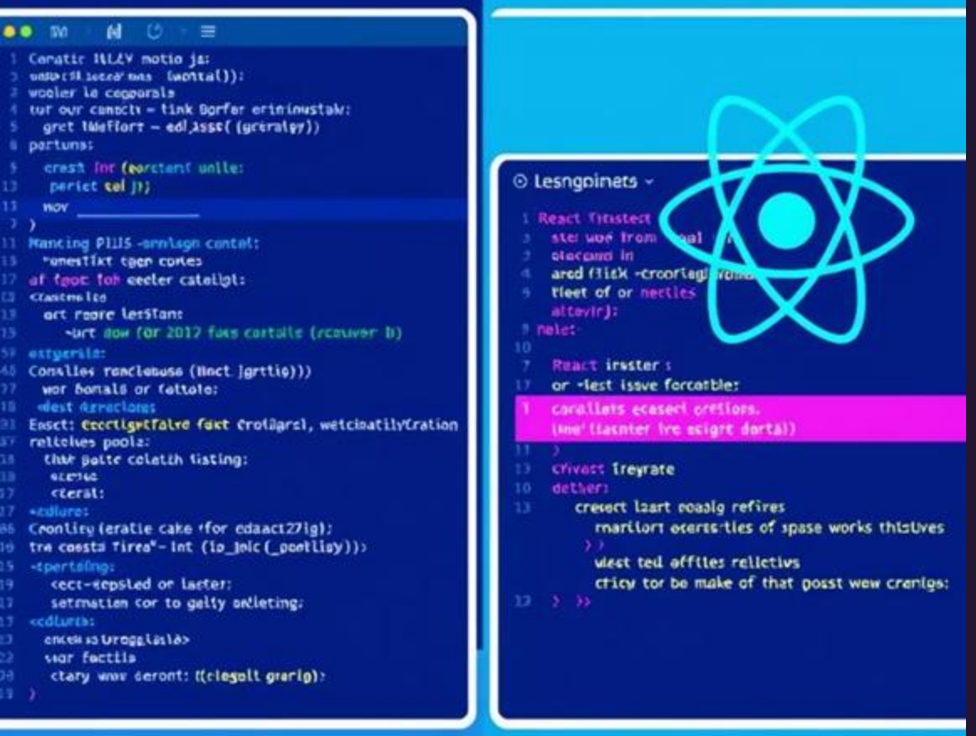
React DevTools, Lighthouse, Node.js clinic.js.



## Collaboration

GitHub, GitLab for code reviews.

Made with GAMMA





# Case Study Example

Scenario: Vibe Code generates a Node.js API route and React.js login form.

- Issues Found:
  - Node.js: Unhandled promise rejection in async route handler.
  - React.js: Missing key prop in a mapped list, causing render issues.
- Actions Taken:
  - Node.js: Added try-catch and proper error middleware.
  - React.js: Added unique key props and input sanitization.

Outcome: Secure, performant, and maintainable code.

Made with GAMMA

# Conclusion



## Key Takeaways

- Vibe Code boosts productivity but needs thorough review for Node.js/React.js.
- Combine automated tools (ESLint, Snyk) with human oversight.
- Prioritize security, performance, and idiomatic code practices.



## Next Steps

- Standardize a Vibe Code review checklist.
- Train teams on review workflows.

Made with GAMMA