🛠️ Code-Checker (ESLint Plugin) – Setup & Usage Guide

Code-Checker is a custom ESLint plugin used to enforce code quality, architecture standards, and API best practices across projects.

📦 Installation

1. Install the required packages:
   ```
   npm install eslint-plugin-code-checker
   npm install typescript (if needed)
   ```

2. Initialize Code-Checker in the project:
   ```
   npx code-checker init
   ```

3. Stage and commit to verify that ESLint is working correctly:
   ```
   git add .
   git commit
   ```

4. To run ESLint on all project files:
   ```
   npx eslint .
   ```

✅ Pre-commit Hook (lint-staged)

Code-Checker integrates with Husky and lint-staged to automatically run ESLint on staged files before every commit. This ensures that only compliant code is committed to the repository.

📏 Current Applicable Rules

The following custom rules are currently enforced by Code-Checker:

- **`no-console-left`** – Prevents leftover `console.log` and debugging statements
- **`require-try-catch-api`** – Enforces the use of `try-catch` blocks in API handlers
- **`warning-paginate-false`** – Warns the use of `paginate: false`
- **`limit-includes-api`** – Limits API `includes` usage to a maximum of three (3)

The actual rule configuration and severity levels are maintained in the different project ESLint configuration repository.

⚙️ Configuration & Rule Management (Code-Checker Repository)

All rule definitions, additions, and updates are maintained **within eslint-plugin-code-checker repository**.

This repository serves as the source of truth for:

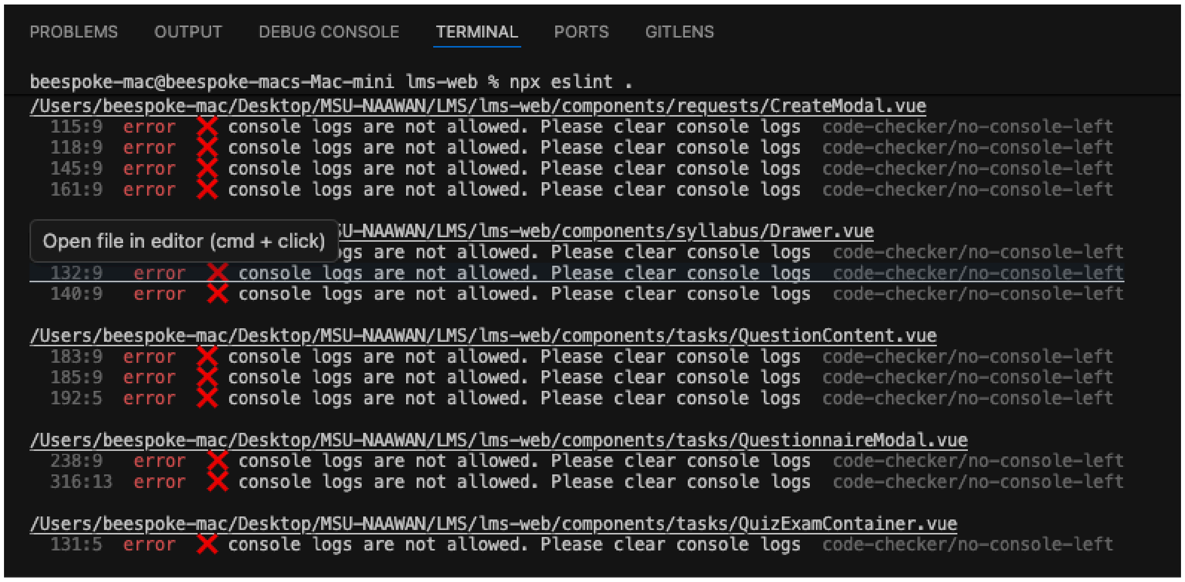
- Adding new custom ESLint rules
- Modifying existing rule logic
- Maintaining rule standards and validations

Projects that consume Code-Checker only enable or disable these rules through their ESLint configuration. Any changes to rule behavior must be implemented in this repository.

🧪 Recommended Workflow

- Rely on pre-commit hooks for immediate feedback
- Run `npx eslint .` before submitting pull requests
- Address lint issues locally prior to pushing changes

🧪 Sample Logs:



```
beespoke-mac@beespoke-macs-Mac-mini lms-web % npx eslint .
/Users/beespoke-mac/Desktop/MSU-NAAWAN/LMS/lms-web/components/requests/CreateModal.vue
  115:9  error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left
  118:9  error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left
  145:9  error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left
  161:9  error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left

U-NAAWAN/LMS/lms-web/components/syllabus/Drawer.vue
gs are not allowed. Please clear console logs  code-checker/no-console-left
  132:9   error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left
  140:9   error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left

/Users/beespoke-mac/Desktop/MSU-NAAWAN/LMS/lms-web/components/tasks/QuestionContent.vue
  183:9  error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left
  185:9  error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left
  192:5  error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left

/Users/beespoke-mac/Desktop/MSU-NAAWAN/LMS/lms-web/components/tasks/QuestionnaireModal.vue
  238:9   error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left
  316:13  error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left

/Users/beespoke-mac/Desktop/MSU-NAAWAN/LMS/lms-web/components/tasks/QuizExamContainer.vue
  131:5  error  ❌ console logs are not allowed. Please clear console logs  code-checker/no-console-left
```

Future Enhancements (Planned Rules)

Additional logic-focused rules may be introduced in future versions, such as:

- **no-hardcoded-role-checks**
  Avoid hardcoded role checks (e.g., `user.role === 'ADMIN'`). Use role constants or permission checks instead.

- **no-direct-status-mutation**
  Prevent direct updates to status fields without proper validation.

- **require-null-check-before-access**
  Require null/undefined checks (or optional chaining) before accessing nested properties (e.g., `user.profile.name`).

- **no-business-logic-in-controller**
  Discourage heavy business logic inside controllers; move logic to service layers.

- **require-transaction-on-multi-write**
  Enforce the use of transactions when performing multiple database write operations.

- **no-magic-numbers-domain**
  Prevent the use of magic numbers in domain logic (e.g., `status === 3`). Use enums or constants instead.

These rules are planned for future releases and may be added as the project's logic validation requirements evolve.