**Enforce Rules / Standards**

**1. Visual Studio Code**

* **Settings and Extensions**: Use settings and extensions to enforce code standards.
  + Install extensions like ESLint, Prettier, and EditorConfig.
  + Create a .vscode/settings.json file to enforce consistent settings:

{

"editor.formatOnSave": true,

"eslint.autoFixOnSave": true,

"editor.codeActionsOnSave": {

"source.fixAll": true

}

}

**2. ESLint**

* **Configuration**: Set up an ESLint configuration to enforce coding standards.
  + Create an .eslintrc.js file:

module.exports = {

env: {

browser: true,

es2021: true,

},

extends: [

'eslint:recommended',

'plugin:react/recommended',

'plugin:@typescript-eslint/recommended'

],

parser: '@typescript-eslint/parser',

parserOptions: {

ecmaFeatures: {

jsx: true,

},

ecmaVersion: 12,

sourceType: 'module',

},

plugins: [

'react',

'@typescript-eslint'

],

rules: {

// Custom rules

'react/prop-types': 'off',

'no-console': 'warn',

// Add rules to enforce your project's coding standards

},

settings: {

react: {

version: 'detect',

},

},

};

**3. GitLab**

* **Pre-commit Hooks**: Use pre-commit hooks to enforce linting before code is committed.
  + Add a .pre-commit-config.yaml file:

repos:

- repo: https://github.com/pre-commit/mirrors-eslint

rev: v7.32.0

hooks:

- id: eslint

* + Install pre-commit and set up the hook:

pip install pre-commit

pre-commit install

* **CI/CD Pipeline: Set up linting in your GitLab CI/CD pipeline.**
  + Add a .gitlab-ci.yml file:

stages:

- lint

- test

lint:

stage: lint

script:

- npm install

- npm run lint

only:

- merge\_requests

- branches

**4. Jenkins**

* **Linting and Testing:** Integrate linting and testing in your Jenkins pipeline.
  + Add a Jenkinsfile:

pipeline {

agent any

stages {

stage('Install Dependencies') {

steps {

sh 'npm install'

}

}

stage('Lint') {

steps {

sh 'npm run lint'

}

}

stage('Test') {

steps {

sh 'npm test'

}

}

}

}

**5. Directory Structure Rules**

* **Enforce Directory Structure**: Use ESLint plugins like eslint-plugin-boundaries to enforce module boundaries and directory structure.
  + Install the plugin:

npm install eslint-plugin-boundaries --save-dev

* + Update your .eslintrc.js:

module.exports = {

// ...existing configuration

plugins: [

'boundaries',

// ...other plugins

],

settings: {

'boundaries/elements': [

{ 'type': 'api', 'pattern': 'src/api/\*' },

{ 'type': 'model', 'pattern': 'src/model/\*' },

{ 'type': 'store', 'pattern': 'src/store/\*' },

{ 'type': 'middleware', 'pattern': 'src/middleware/\*' },

{ 'type': 'router', 'pattern': 'src/router/\*' },

{ 'type': 'pages', 'pattern': 'src/pages/\*' },

]

},

rules: {

'boundaries/element-types': [2, {

'default': 'disallow',

'rules': [

{ 'from': 'api', 'allow': ['model', 'store', 'middleware'] },

{ 'from': 'model', 'allow': ['store', 'middleware'] },

{ 'from': 'store', 'allow': ['middleware'] },

{ 'from': 'middleware', 'allow': [] },

{ 'from': 'router', 'allow': ['pages'] },

{ 'from': 'pages', 'allow': ['router'] },

]

}]

}

};

**6. Prettier**

* **Code Formatting**: Use Prettier for consistent code formatting.
  + Create a .prettierrc file:

{

"singleQuote": true,

"semi": false,

"trailingComma": "es5",

"tabWidth": 2

}