**Cursor AI Rules in React**

**Step 1: Download and Install Cursor AI**

1. **Go to Cursor AI's official website**:
   * Visit [Cursor AI](https://www.cursor.so/) to access the official page.
2. **Sign Up/Login**:
   * Create an account if you don’t have one, or log in to your existing account.
3. **Download Cursor**:
   * After logging in, download the **Cursor AI** application for your platform (Windows/macOS/Linux).
4. **Install the Application**:
   * Run the downloaded installer and follow the on-screen instructions to complete the installation.

**Step 2: Set Up a New React Project (if you don't have one)**

1. **Initialize a React App**:
   * Open a terminal or command prompt and run the following command to create a new React app (using Create React App):

npx create-react-app my-app --template typescript

cd my-app

1. **Install Dependencies**:
   * If you are planning to use **React Router** or any other dependencies (like Bootstrap for styling), install them as well:

npm install react-router-dom bootstrap

**Step 3: Integrate Cursor AI with Your React Project**

1. **Open Cursor AI**:
   * Launch the **Cursor AI** application you downloaded earlier.
2. **Create or Open a Project in Cursor**:
   * Open your **React project** folder in Cursor AI by navigating to the project directory in the **Cursor AI** interface.
3. **Enable Cursor Rules**:
   * Once your project is opened in Cursor, you can add **.cursorrules** to set your coding standards and architecture.

**Step 4: Add Rules to .cursorrules File**

1. **Create the .cursorrules File**:
   * In the root of your React project (where package.json is located), create a file called .cursorrules.
2. **Write the Rules**:
   * Open the .cursorrules file and start adding your **coding standards**, **architectural rules**, **tooling dependencies**, and other project-specific guidelines. Here’s an example of rules for React development:

# PROJECT OVERVIEW

- \*\*Tech Stack\*\*: React 19, Node 18, TypeScript, Tailwind CSS, Jest, Supabase.

- \*\*Description\*\*: React frontend with Node backend, using TypeScript and Tailwind for styling.

# CODE STYLE

- \*\*Naming\*\*:

- Components: `PascalCase` (e.g., `MyComponent`).

- Variables/functions/hooks: `camelCase` (e.g., `fetchData`).

- \*\*TypeScript\*\*: Avoid `any` type. Always use explicit types.

# FOLDER STRUCTURE

- `/src/components/`: React components.

- `/src/hooks/`: Custom hooks.

- `/src/utils/`: Utility functions.

- `/tests/`: Unit/integration tests.

# PROJECT RULES

- \*\*State Management\*\*: Avoid global state (e.g., Redux) unless necessary.

- \*\*Logging\*\*: Use `@/lib/logger`.

# TOOLING & DEPENDENCY RULES

- \*\*Packages\*\*: Do not add packages without approval. Use stable versions.

- \*\*React\*\*: Use React 19 and Node 18.

# WORKFLOWS

- \*\*Versioning\*\*: Use semantic versioning and update `CHANGELOG.md` with each feature.

- \*\*Commits\*\*: Use conventional commits (e.g., `feat:`, `fix:`).

# DEBUGGING & MAINTENANCE

- \*\*Issue Investigation\*\*: Explain issues clearly, present solutions, and ask for approval before making changes.

- \*\*Documentation\*\*: Keep `productContext.md` and `currentTasks.md` updated.

**Step 5: Test and Validate Cursor Rules**

1. **Activate Rules in Cursor**:
   * Cursor will automatically detect the .cursorrules file when the project is loaded.
   * Review if all your rules are correctly applied.

A screenshot of a computer screen

Description automatically generated