**Cursor AI Rules in Angular**

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

1. **Go to Cursor AI's official website:**
   * Visit [Cursor AI](https://www.cursor.com) 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 Angular Project (if you don't have one)**

1. **Initialize an Angular App:**
   * Open a terminal or command prompt and run the following command to create a new Angular app:

ng new my-angular-app --routing --style=scss

cd my-angular-app

1. **Install Dependencies:**
   * If you plan to use additional dependencies (like Angular Material or RxJS), install them as well:

npm install @angular/material rxjs

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

1. **Open Cursor AI:**
   * Launch the Cursor AI application you downloaded earlier.
2. **Create or Open a Project in Cursor:**
   * Open your Angular 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 a .cursorrules file to set your coding standards and architecture.

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

1. **Create the .cursorrules File:**
   * In the root of your Angular 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 Angular development:

# PROJECT OVERVIEW

- \*\*Tech Stack\*\*: Angular 15, Node 18, TypeScript, SCSS, Jasmine, Firebase.

- \*\*Description\*\*: Angular frontend with Firebase backend, using TypeScript and SCSS for styling.

# CODE STYLE

- \*\*Naming\*\*:

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

- Directives/Services: `camelCase` (e.g., `userProfileService`).

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

- Constants: `UPPER\_SNAKE\_CASE` (e.g., `MAX\_RETRIES`).

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

# FOLDER STRUCTURE

- `/src/app/components/`: Angular components.

- `/src/app/services/`: Services for business logic.

- `/src/app/models/`: TypeScript interfaces and models.

- `/src/app/shared/`: Shared components, directives, and pipes.

# PROJECT RULES

- \*\*State Management\*\*: Use `NgRx` for state management if the app is large.

- \*\*Lazy Loading\*\*: Use lazy loading for feature modules to optimize performance.

- \*\*No inline styles\*\*: Use SCSS files for all styling to ensure maintainability.

# TOOLING & DEPENDENCY RULES

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

- \*\*Angular Version\*\*: Stick to the current stable Angular version (currently Angular 15).

# 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 `README.md` and `DEVELOPMENT.md` updated for new developers.

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

1. **Activate Rules in Cursor:**
   * Cursor will automatically detect the .cursorrules file when the project is loaded.
2. **Review and Apply Rules:**
   * Review if all your rules are correctly applied and validated by Cursor AI.
   * If any issues arise, modify the .cursorrules file and restart Cursor AI.