Feature Documentation Rule Report

This document summarizes the setup and usage of the Feature Completion Documentation Generator rule in Cursor AI. It includes the rule configuration, a screenshot of its implementation, the prompt used, and the result obtained.

Rule Configuration:

Below is the configuration of the rule file (`feature-docs.mdc`) used to automatically generate Developer Documents and User Manuals without manual prompts.

****Rule Name:**** Feature Completion Documentation Generator

****Scope:**** Apply this rule when finalizing a feature (e.g., at final commit or end of development chat)

****Trigger:**** After code changes for a feature are done, generate documentation for that feature.

****Instructions for Cursor AI:****

When a new feature or code update has been completed, ****automatically generate two documents**** describing the change:

1. ****Developer Document (Technical):**** Write a markdown document aimed at developers/maintainers that explains the feature's implementation details. Include information such as:

  - ****Architecture & Code Changes:**** Describe any new modules, files, or architectural changes. Mention file paths or project structure updates (e.g., new directories, moved files).

  - ****APIs & Endpoints:**** List new public APIs, endpoints, or functions introduced by this feature. Provide brief descriptions of their purpose and how they interact with existing components.

  - ****Logic Flow:**** Explain the logic of important algorithms or workflows added. (For example, describe how data flows through the new feature's functions or how different components interact in this update.)

  - ****Dependencies:**** Note any new libraries or dependencies added to the project, and why they were needed.

  - ****Testing Instructions:**** Provide guidance on how to test this feature. This could include new unit tests to run (`/path/to/testfile`), manual test steps, or configuration needed to verify the feature is working.

2. ****User Manual Document (Non-Technical):**** Write a markdown document for end users that explains the feature in simple, user-friendly terms. Include:

  - ****Feature Summary:**** A plain-language overview of what the feature is and what benefit or functionality it adds for the user. (For example, “This update adds a new ****Search**** function that helps you find items by name.”)

  - ****How to Use:**** Step-by-step instructions on how the user can access or activate this new feature in the UI. Reference any menus, buttons, or screens by name. (E.g., “Go to the ****Settings**** page and click on ****Advanced****, then ...”).

  - ****UI Changes:**** Describe any changes in the interface or user workflow. Mention new buttons, icons, form fields, or options the user will see, and what they do. Keep descriptions visual and easy to follow (as if writing a help guide).

  - ****Behaviour Changes: **** Explain any changes in app behavior from the user’s perspective. For instance, if a process is now automated or an existing feature behaves differently after this update, clarify that in simple terms.

  - ****Tips/Notes:**** Include any helpful tips, limitations, or prerequisites for using the feature (if needed). For example, if the user needs certain permissions or must enable a setting, mention that. Keep this section brief and jargon-free.

****Guiding Principles:****

- ****No Unimplemented Features:**** Only document what has actually been implemented in this update. Do not invent features or describe functionality that isn’t present in the code. The content should be based on the final code changes for the feature.

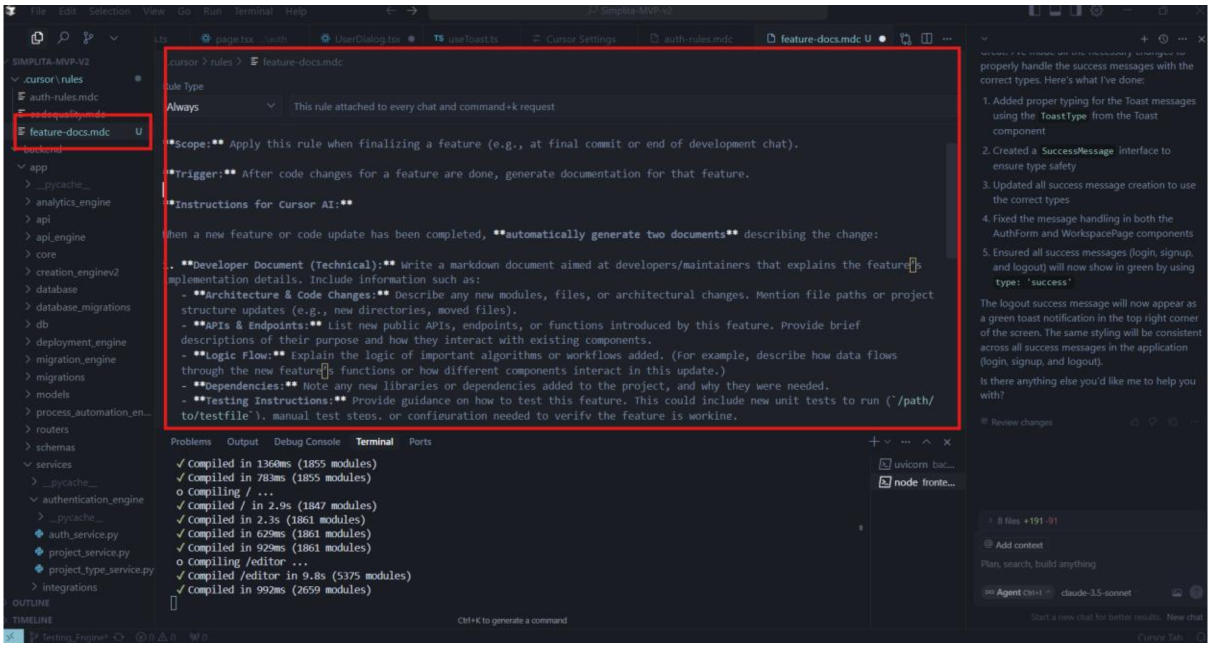
- ****Clarity and Accuracy:**** Use clear, concise language appropriate to the audience of each document. In the Developer Document, it’s okay to use technical terms and reference the code; in the User Manual, use everyday language and avoid internal code references.

- ****Structure & Format:**** Present each document in an organized way (with headings, bullet points, or numbered steps as appropriate) to enhance readability. Ensure the tone of the Developer Document is explanatory and factual, while the User Manual is friendly and instructional.

****(End of Rule Template)****]]]]]]]]]]]]]]]]]]]]]]]]]]]

Screenshot of Rule in Cursor AI IDE:

Insert the screenshot below showing the rule configuration in the Cursor AI IDE.



This rule attached to every chat and command+k request

```
**Scope:** Apply this rule when finalizing a feature (e.g., at final commit or end of development chat).

**Trigger:** After code changes for a feature are done, generate documentation for that feature.

**Instructions for Cursor AI:**

When a new feature or code update has been completed, **automatically generate two documents** describing the change:

1. **Developer Document (Technical):** Write a markdown document aimed at developers/maintainers that explains the feature's
implementation details. Include information such as:
  - **Architecture & Code Changes:** Describe any new modules, files, or architectural changes. Mention file paths or project
  structure updates (e.g., new directories, moved files).
  - **APIs & Endpoints:** List new public APIs, endpoints, or functions introduced by this feature. Provide brief
  descriptions of their purpose and how they interact with existing components.
  - **Logic Flow:** Explain the logic of important algorithms or workflows added. (For example, describe how data flows
  through the new feature's functions or how different components interact in this update.)
  - **Dependencies:** Note any new libraries or dependencies added to the project, and why they were needed.
  - **Testing Instructions:** Provide guidance on how to test this feature. This could include new unit tests to run (`/path/
  to/testfile`), manual test steps, or configuration needed to verify the feature is working.
```

```
✓ Compiled in 1360ms (1855 modules)
✓ Compiled in 783ms (1855 modules)
o Compiling / ...
✓ Compiled / in 2.9s (1847 modules)
✓ Compiled in 2.3s (1861 modules)
✓ Compiled in 629ms (1861 modules)
✓ Compiled in 929ms (1861 modules)
o Compiling /editor ...
✓ Compiled /editor in 9.8s (5375 modules)
✓ Compiled in 992ms (2659 modules)
```

properly handle the success messages with the correct types. Here's what I've done:

1. Added proper typing for the Toast messages using the `ToastType` from the Toast component
2. Created a `SuccessMessage` interface to ensure type safety
3. Updated all success message creation to use the correct types
4. Fixed the message handling in both the AuthForm and WorkspacePage components
5. Ensured all success messages (login, signup, and logout) will now show in green by using `type: 'success'`

The logout success message will now appear as a green toast notification in the top right corner of the screen. The same styling will be consistent across all success messages in the application (login, signup, and logout).

Is there anything else you'd like me to help you with?

Prompt Used:

The rule was configured to trigger on chat finalization, so no manual prompt is required. However, for reference, the original manual prompt was:

"Please generate the Dev Doc and User Manual for my last feature":

Sample Data:

1. **Task Prompt:**
   Change the button label from "Generate Test Cases for Selected Pages" to "Generate Test Cases".
2. After generating and accepting the code,
3. Ask Cursor Chat: “Please generate the Dev Doc and User Manual for my most recent feature.”

4. Dev Documentation:

Feature Documentation: Update "Generate Test Cases" Button UI

Date: October 26, 2023

Author: AI Assistant **Version:** 1.0

---

1. Developer Documentation

1.1. Introduction This document outlines the technical changes made to update the user interface of the "Generate Test Cases" button within the test case generation panel. The primary goals were to simplify the button's text and standardize its icon.**1.2. Files Modified** The following file was modified:

- frontend/src/components/testing_engine/Chat Panel/Test Case panel/GenTestCase.tsx

1.3. Changes Implemented

- **Button Text Simplification:**
- The text on the button was changed from "Generate Test Cases for Selected Pages" to "Generate Test Cases".
- This change was made within the <span> element inside the button component.
- **Icon Update:**
- The previous inline SVG icon (a gear-like or abstract representation) was replaced.
- The Lightbulb icon from the lucide-react library was imported and implemented.

- The new icon is displayed to the left of the button text.
- The Lightbulb component is used with size={18} and className="mr-2".
- **Conditional Rendering for Loading State:**
- The loading state (isGenerating === true) was also updated to display a more visually clear loading indicator:
- A div element with className="flex items-center".
- An inner div for the spinner: className="animate-spin rounded-full h-5 w-5 border-b-2 border-white mr-2".
- A span containing the text "Generating...".
- This ensures that the Lightbulb icon is only visible when the button is not in a loading state.

1.4. Code Snippet (Illustrative Change)

diff

Apply to feature-docs...

```
--- a/frontend/src/components/testing_engine/Chat Panel/Test Case panel/GenTestCase.tsx
+++ b/frontend/src/components/testing_engine/Chat Panel/Test Case panel/GenTestCase.tsx
@@ -1,5 +1,6 @@
 import React, { useState, useEffect } from 'react';
 import { toast } from 'react-hot-toast';
+import { Lightbulb } from 'lucide-react';

 interface GenTestCaseProps {
   setTestCases: React.Dispatch<React.SetStateAction<any[]>>;
@@ -435,19 +436,14 @@
       className={buttonClassName}
     >
```

```
      {isGenerating ? (
-         <>
-           <svg className="animate-spin h-4 w-
4" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
-             <circle className="opacity-
25" cx="12" cy="12" r="10" stroke="currentColor" strokeWidth="4"></circle>
-             <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-
8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135 5.824 3 7.938
l3-2.647z"></path>
-           </svg>
-           <span>Generating...</span>
-         </>
+         <div className="flex items-center">
+           <div className="animate-spin rounded-full h-5 w-5 border-b-2 border-white mr-
2"></div>
+           <span>Generating...</span>
+         </div>
      ) : (
         <>
-           <svg xmlns="http://www.w3.org/2000/svg" className="h-4 w-
4" fill="none" viewBox="0 0 24 24" stroke="currentColor">
-             <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M9.663 1
7h4.673M12 3v1m6.364 1.636l-.707.707M21 12h-1M4 12H3m3.343-5.657l-.707-
.707m2.828 9.9a5 5 0 117.072 0l-.548.547A3.374 3.374 0 0014 18.469V19a2 2 0 11-4 0v-
.531c0-.895-.356-1.754-.988-2.386l-.548-.547z" />
-           </svg>
-           <span>Generate Test Cases for Selected Pages</span>
+           <Lightbulb size={18} className="mr-2" />
+           <span>Generate Test Cases</span>
```

```
      </>
    )}
  </button>
```

1.5. Reason for Change

- To improve user experience by providing a more concise and universally understood button label.
- To enhance visual consistency and maintainability by utilizing an icon from the existing lucide-react library.

1.6. Potential Impacts

- No functional impact on the test case generation logic.
- Minor visual update to the button's appearance.
- Slight improvement in code readability and consistency within the GenTestCase.tsx component.

5.User Manual:

2.1. Feature Update: Enhanced "Generate Test Cases" Button We've updated the "Generate Test Cases" button to make it clearer and more visually intuitive!**2.2. What's New?**

- **Simpler Button Name:** The button previously labelled "Generate Test Cases for Selected Pages" now simply reads "**Generate Test Cases**".
- **New Icon:** We've added a new **Lightbulb icon (💡)** next to the button text. This icon visually represents the idea of generating new insights or, in this case, test cases.

2.3. How This Affects You

- **Functionality Remains the Same:** This is a visual update. The button still performs the exact same function: generating test cases based on the pages you have selected in the sidebar.
- **Easier to Understand:** The shorter label is more direct and easier to read.

- **Intuitive Design:** The new lightbulb icon provides a quick visual cue about the button's purpose.

2.4. Using the Updated Button

1. As before, select the pages for which you want to generate test cases from the sidebar.
2. Locate the button, which is now labelled "**Generate Test Cases**" and features a 💡 **Lightbulb icon**.

- *Previously:* The button had a different, more abstract icon and longer text.
- *Now:* Look for the 💡 and the shorter "Generate Test Cases" text.

3. Click the button.
4. If test case generation is in progress, the button will display a spinner and the text "Generating...".
5. Once completed, the generated test cases will appear as usual.

Visual Comparison:

- **Old Button:** Displayed a generic icon with the text "Generate Test Cases for Selected Pages".
- **New Button:** Displays a 💡 **Lightbulb icon** with the text "**Generate Test Cases**".

This update aims to improve your experience by making the interface cleaner and more user-friendly.