Table Implementation

This was my thought process while designing the table. I started with the simplest approach and gradually added complexity based on styling needs, flexibility, and testability.

1. Vanilla Table (HTML & CSS)

The first approach used plain HTML & CSS to create a simple table with a hover effect. The button inside a row appears only when hovered, achieved using CSS pseudo-classes (:hover). This method is lightweight, efficient, and works across all browsers without JavaScript.

2. Styled Table (Tailwind CSS)

Next, Tailwind CSS was used to enhance styling while keeping the structure the same. The hover effect still relied on CSS, but Tailwind made styling easier and more maintainable.

3. State-Based Table (React + Tailwind)

For a more dynamic approach, React's useState was used to control button visibility. Instead of CSS-only hover, useState tracked hover state explicitly. This should only be used when the table gets complicated, making it testable and flexible.

4. Testing the Implementation

Tests were written using Jest & React Testing Library, focusing on:

- Is table even rendering?
- Is the correct data visible?
- Are the table headers visible?
- Verifying button visibility on hover.
- Ensuring it disappears when the mouse leaves.

5. What to use & why?

For simple tables, CSS-only hover is best due to its efficiency and simplicity. However, when interactions become more complex, using state (useState) or event listeners (onMouseEnter) improves testability and control.

(*Mock data in this project was generated using ChatGPT for testing purposes.*)

Nishant Bhandari