Lesson 07 Demo 01

Styling Essential Elements Using Tailwind

Objectives: To style and configure essential UI elements, including buttons, dropdowns, badges, spinners, and progress bars, using Tailwind CSS utility classes, ensuring a responsive design and smooth user interaction

Tools required: Visual Studio Code, Web browser, Tailwind CSS

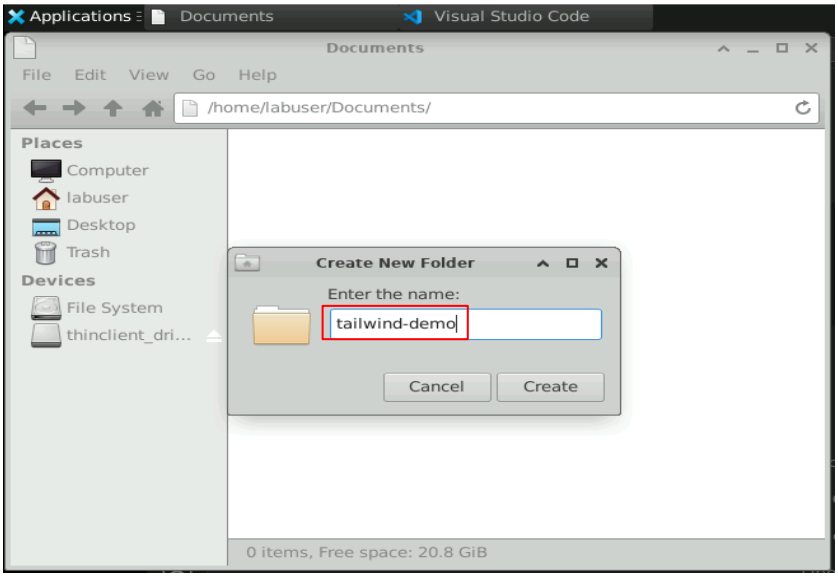
Prerequisites: NA

Steps to be followed:

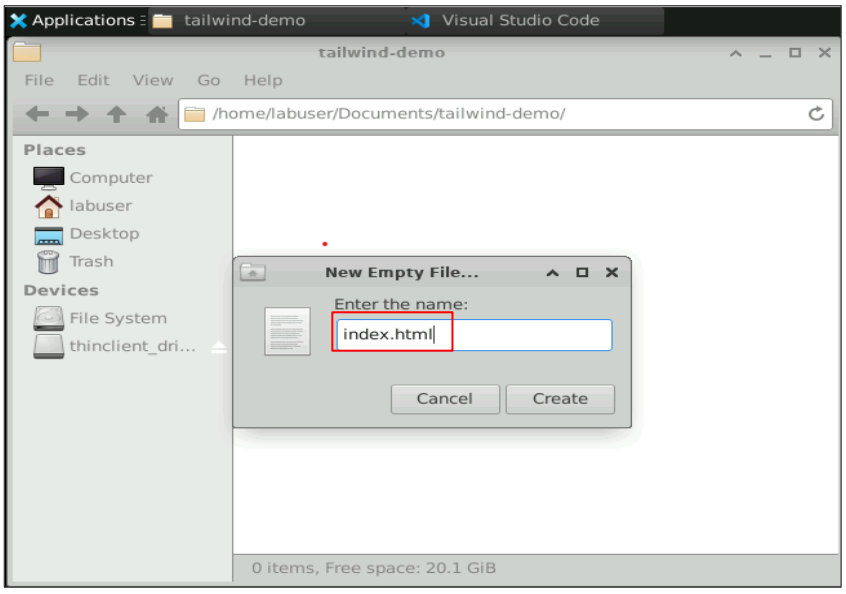
1. Set up the project folder and base HTML structure
2. Create and style the button components
3. Create a dropdown menu with three selectable options
4. Design badge elements
5. Add a loading spinner
6. Build a progress bar
7. Run the implementation in a web browser

Step 1: Set up the project folder and base HTML structure

1.1 Open Visual Studio Code and create a new project folder named **tailwind-demo**



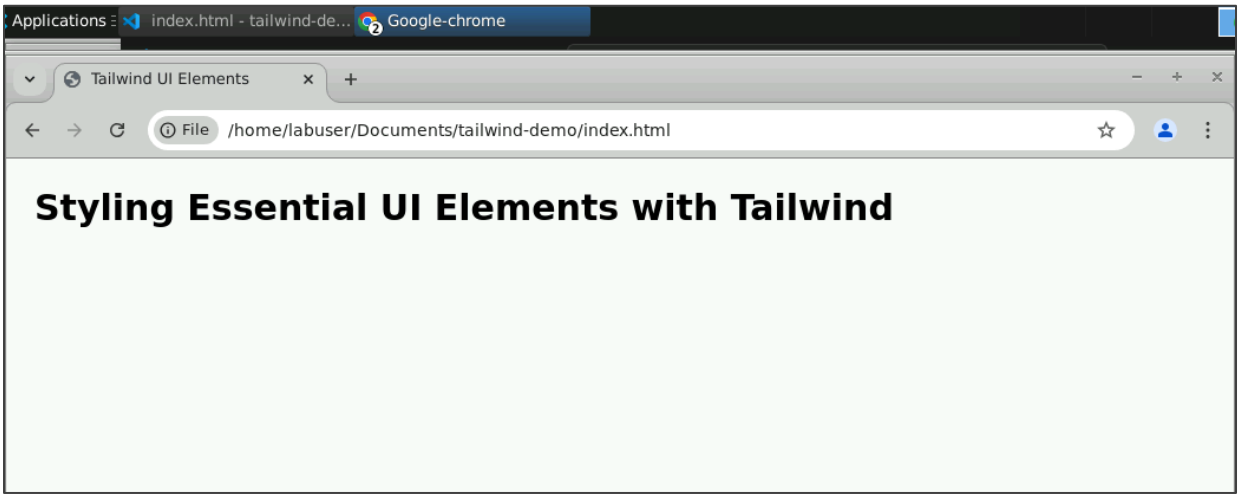
1.2 Inside this folder, create a file named **index.html**



1.3 Add the following boilerplate code to the index.html file:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tailwind UI Elements</title>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-gray-100 p-6">
  <div class="container mx-auto">
<h1 class="text-3xl font-bold mb-6">Styling Essential UI Elements with Tailwind</h1>
  </div>
</body>
</html>
```

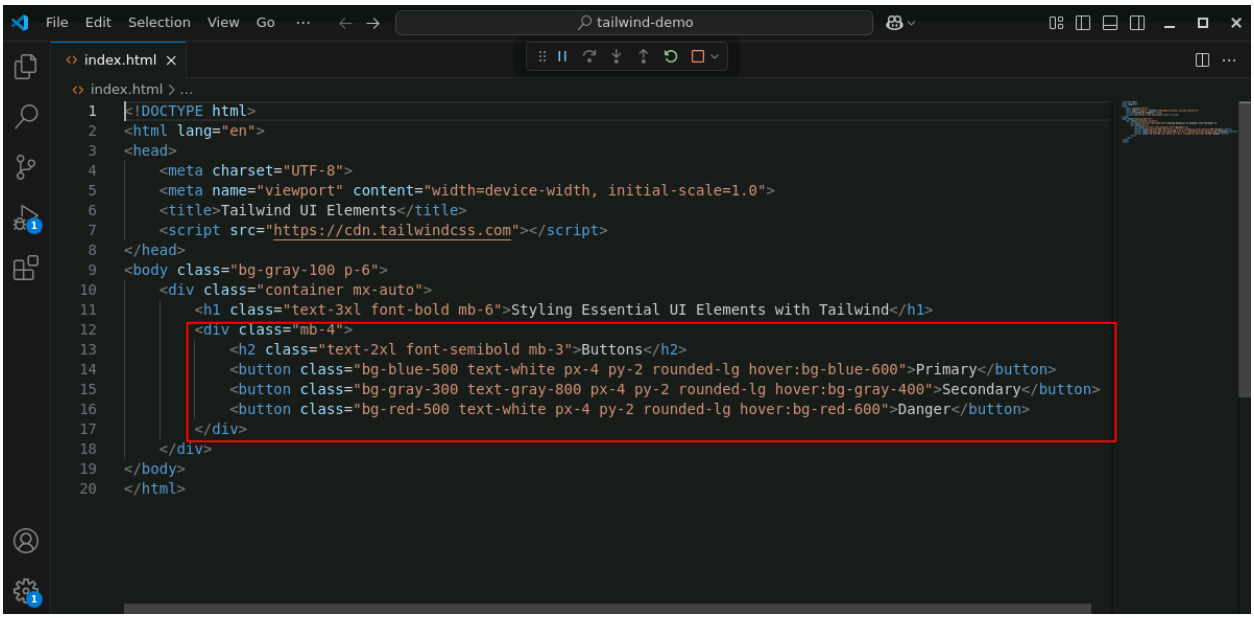
1.4 Save the file and open it in a web browser to verify that the basic structure of the web page is displayed correctly



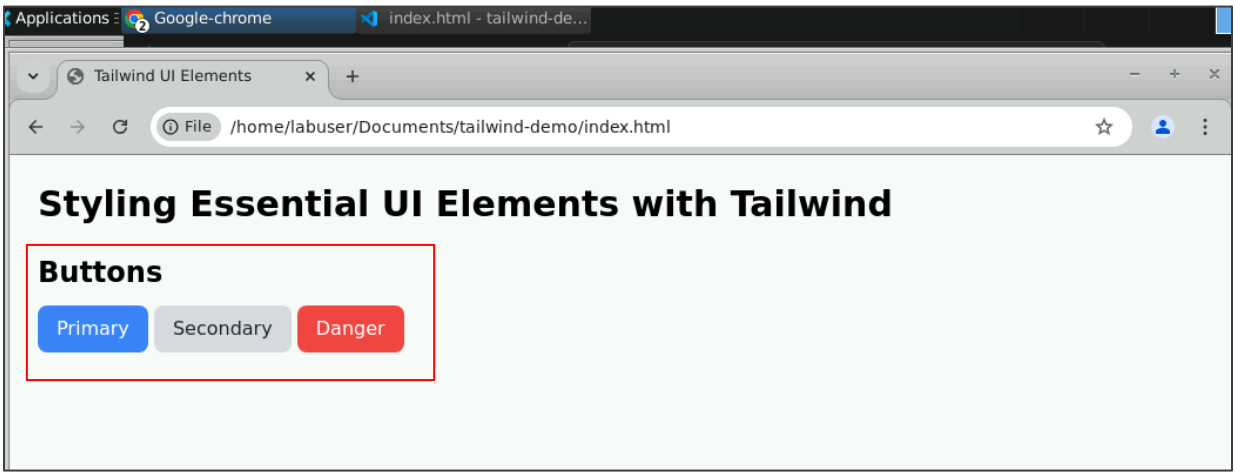
Step 2: Create and style the button components

2.1 Modify the index.html file and add the following code for button styles inside the <body> tag:

```
<div class="mb-4">
        <h2 class="text-2xl font-semibold mb-3">Buttons</h2>
        <button class="bg-blue-500 text-white px-4 py-2 rounded-lg hover:bg-blue-600">Primary</button>
        <button class="bg-gray-300 text-gray-800 px-4 py-2 rounded-lg hover:bg-gray-400">Secondary</button>
        <button class="bg-red-500 text-white px-4 py-2 rounded-lg hover:bg-red-600">Danger</button>
    </div>
```



2.2 Save and refresh the browser to see the styled buttons

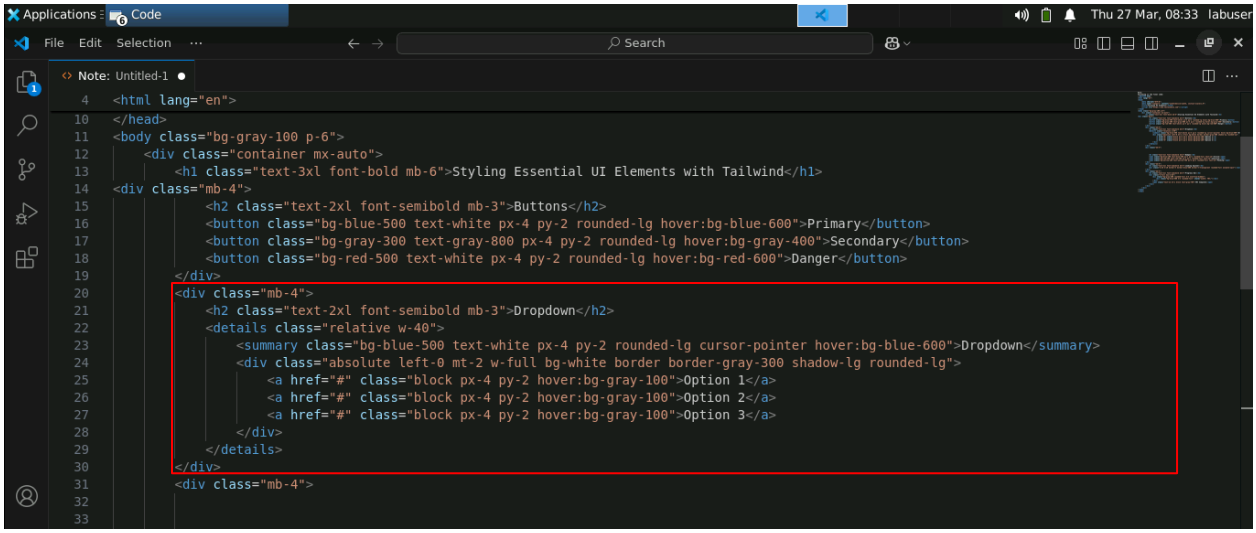


By completing this step, you have styled three types of buttons: Primary, Secondary, and Danger. Each button uses distinct colors.

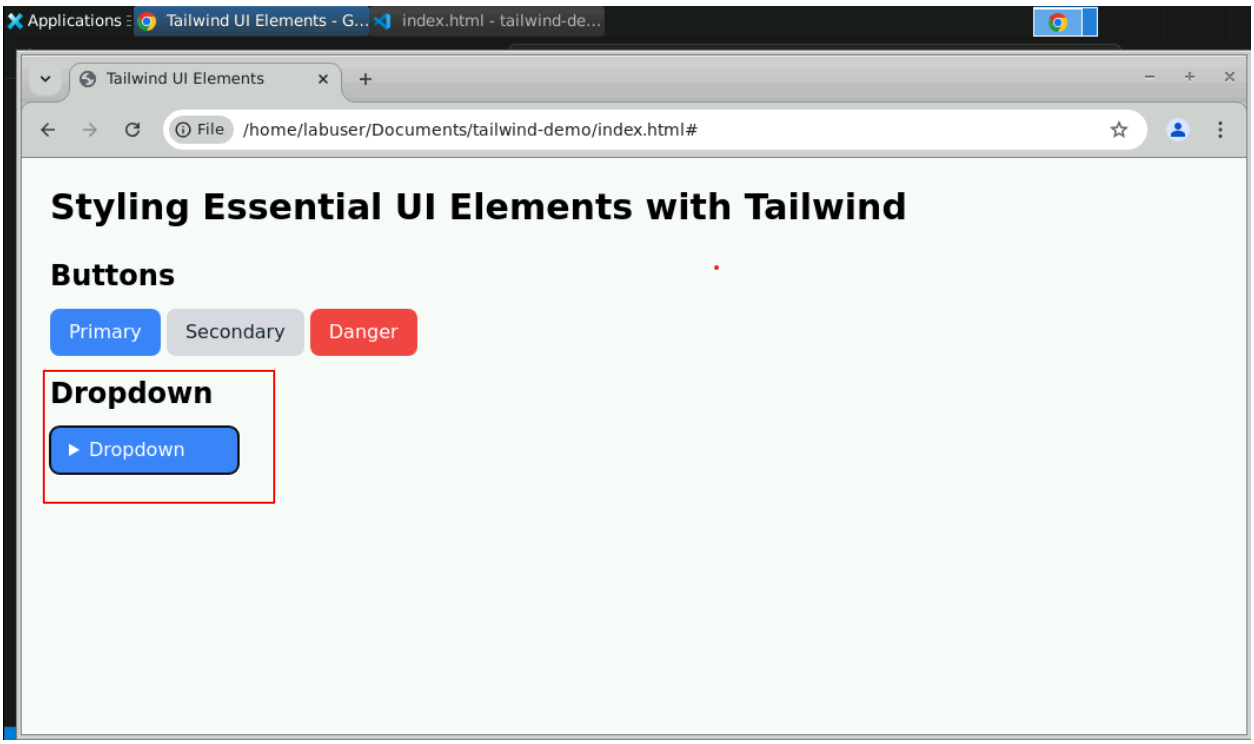
Step 3: Create a dropdown menu with three selectable options

3.1 Modify the index.html file and add the following dropdown code inside the <body> tag to create an interactive menu with selectable options:

```
<div class="mb-4">
        <h2 class="text-2xl font-semibold mb-3">Dropdown</h2>
        <details class="relative w-40">
          <summary class="bg-blue-500 text-white px-4 py-2 rounded-lg cursor-pointer hover:bg-blue-600">Dropdown</summary>
          <div class="absolute left-0 mt-2 w-full bg-white border border-gray-300 shadow-lg rounded-lg">
            <a href="#" class="block px-4 py-2 hover:bg-gray-100">Option 1</a>
            <a href="#" class="block px-4 py-2 hover:bg-gray-100">Option 2</a>
            <a href="#" class="block px-4 py-2 hover:bg-gray-100">Option 3</a>
          </div>
        </details>
      </div>
```



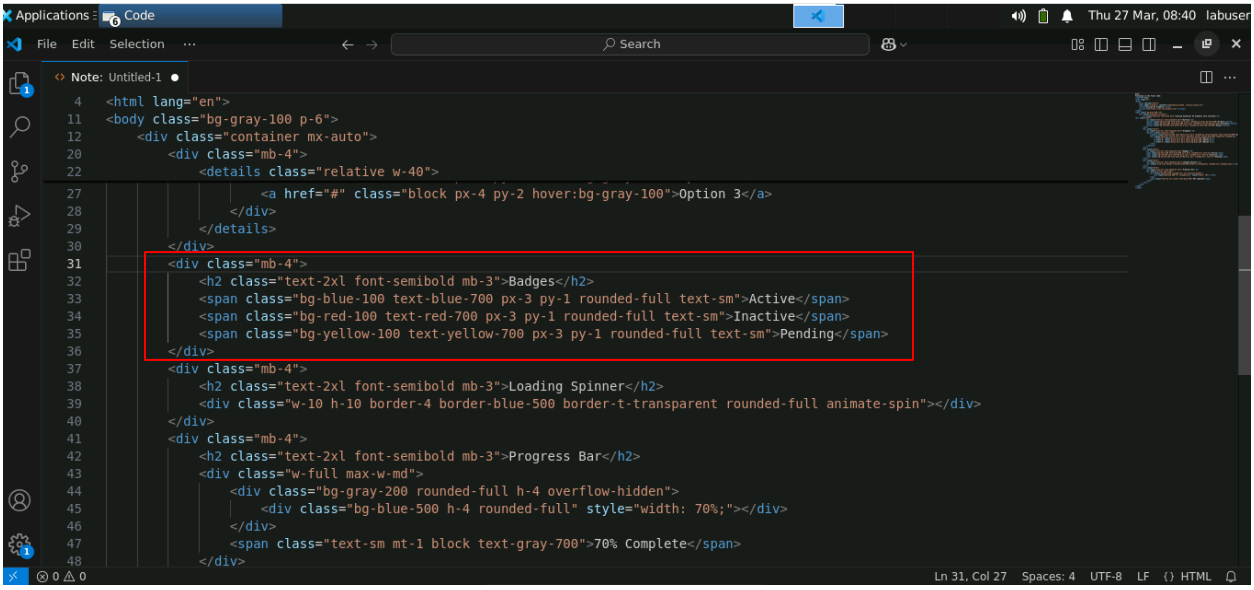
3.2 Save and refresh the browser to see the dropdown



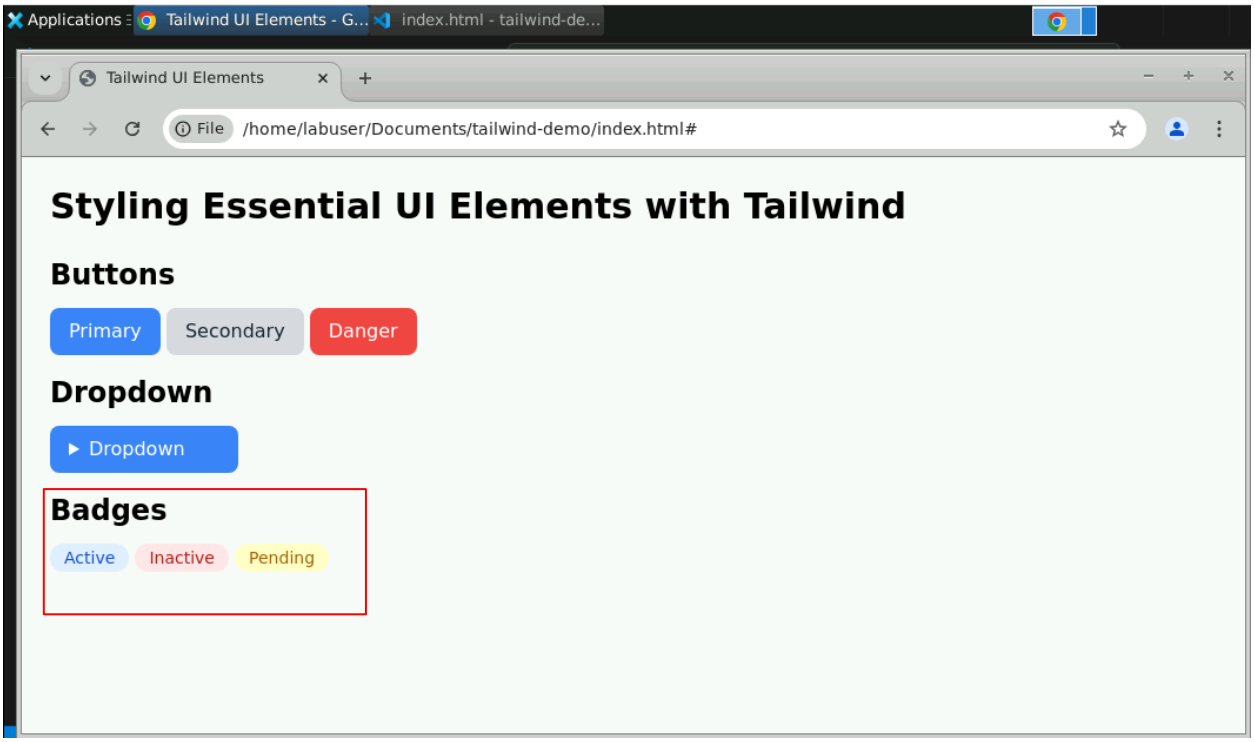
Step 4: Design badge elements

4.1 Modify the index.html file and add the following code for badge elements inside the <body> tag:

```
<div class=”mb-4”>
        <h2 class="text-2xl font-semibold mb-3">Badges</h2>
 <span class="bg-blue-100 text-blue-700 px-3 py-1 rounded-full text-sm">Active</span>
 <span class="bg-red-100 text-red-700 px-3 py-1 rounded-full text-sm">Inactive</span>
 <span class="bg-yellow-100 text-yellow-700 px-3 py-1 rounded-full text-sm">Pending</span>
</div>
```



4.2 Save and refresh the browser to see the badges

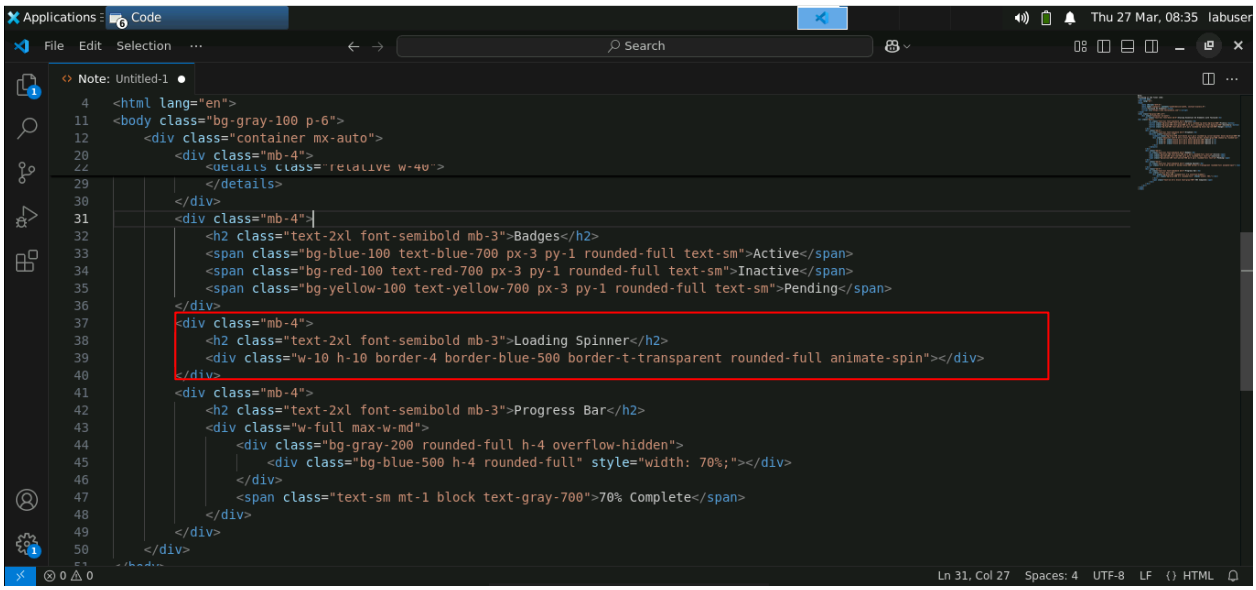


By following this step, we have designed the badges that can be used to visually highlight status indicators such as “Active,” “Inactive”, and “Pending.” Each badge will have a unique background and text color to represent its status clearly.

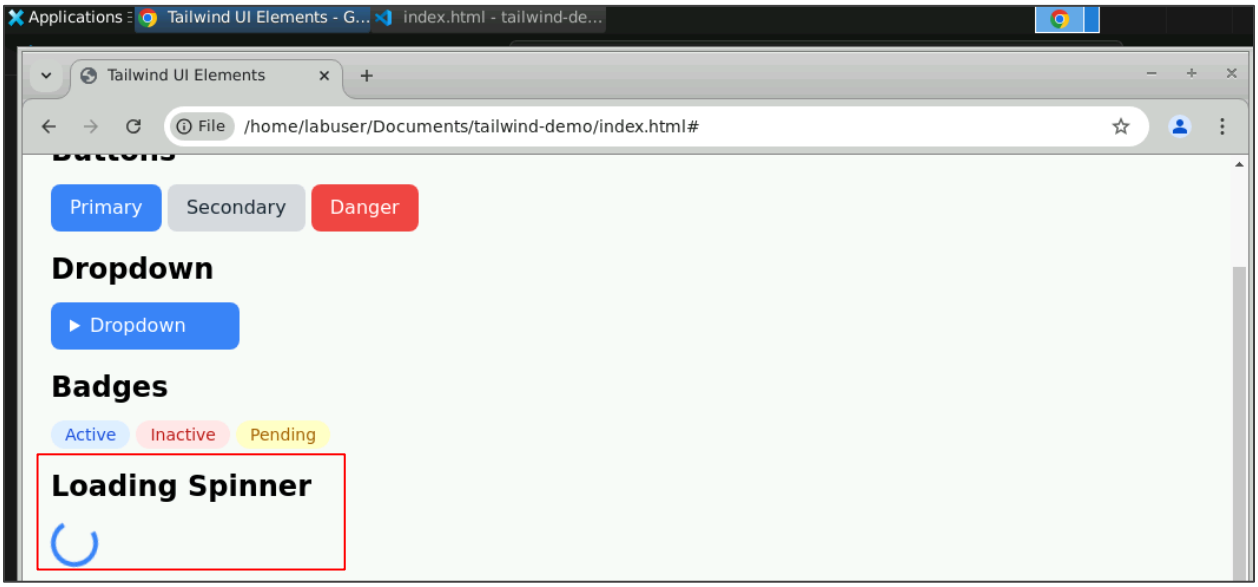
Step 5: Add a loading spinner

5.1 Modify the index.html file and insert code for a loading spinner inside the <body> tag:

** **
<h2 class="text-2xl font-semibold mb-3">Loading Spinner</h2>
** **
** **



5.2 Save and refresh the browser to see the loading spinner

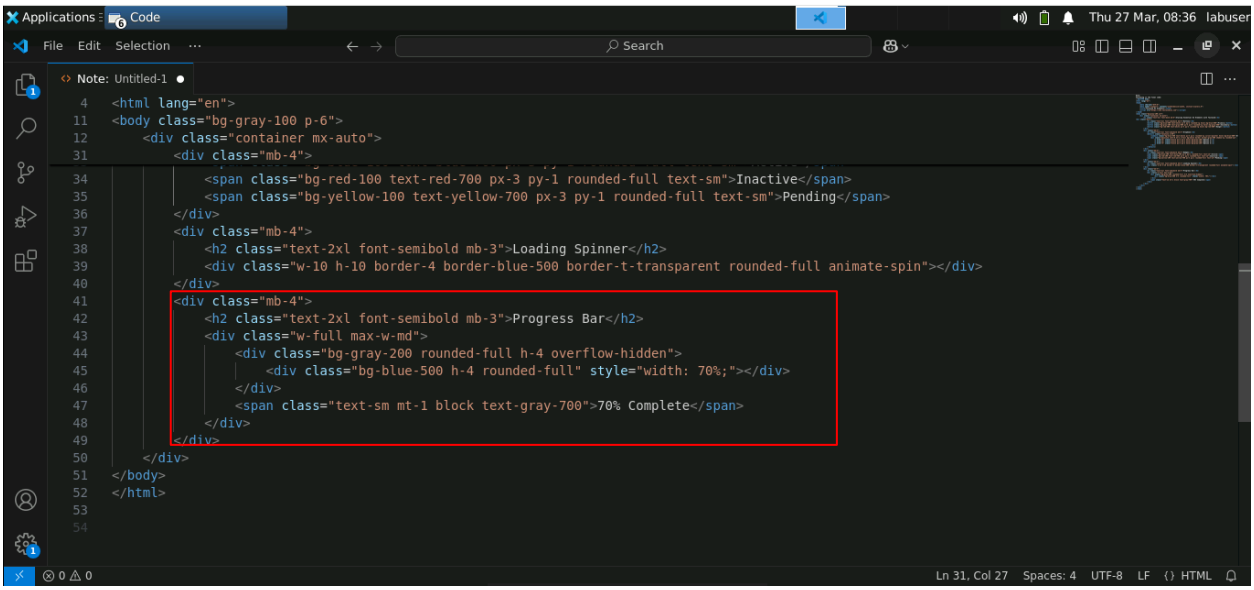


By following this step, you have inserted a loading spinner that visually indicates data is being fetched or a process is in progress. This helps enhance user experience by clearly signaling system activity during wait times.

Step 6: Build a progress bar

6.1 Modify the index.html file and add a code for progress bar inside the <body> tag:

```
<div class="mb-4">
 <h2 class="text-2xl font-semibold mb-3">Progress Bar</h2>
 <div class="w-full max-w-md">
   <div class="bg-gray-200 rounded-full h-4 overflow-hidden">
     <div class="bg-blue-500 h-4 rounded-full" style="width: 70%;"></div>
   </div>
   <span class="text-sm mt-1 block text-gray-700">70% Complete</span>
 </div>
</div>
```

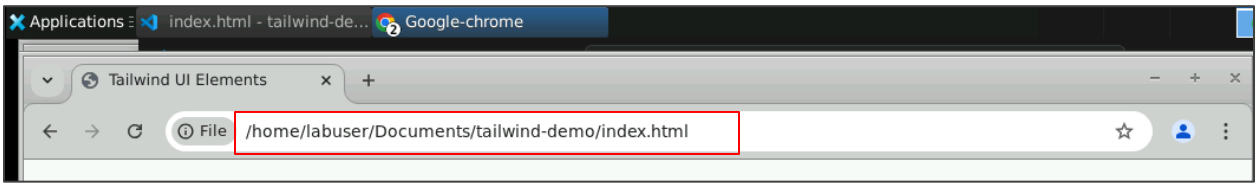


6.2 Save and refresh the browser to see the progress bar

By completing this step, you have added a progress bar that visually communicates the completion status of a task or process. It helps users understand how much of the task has been completed and how much remains, improving clarity and engagement.

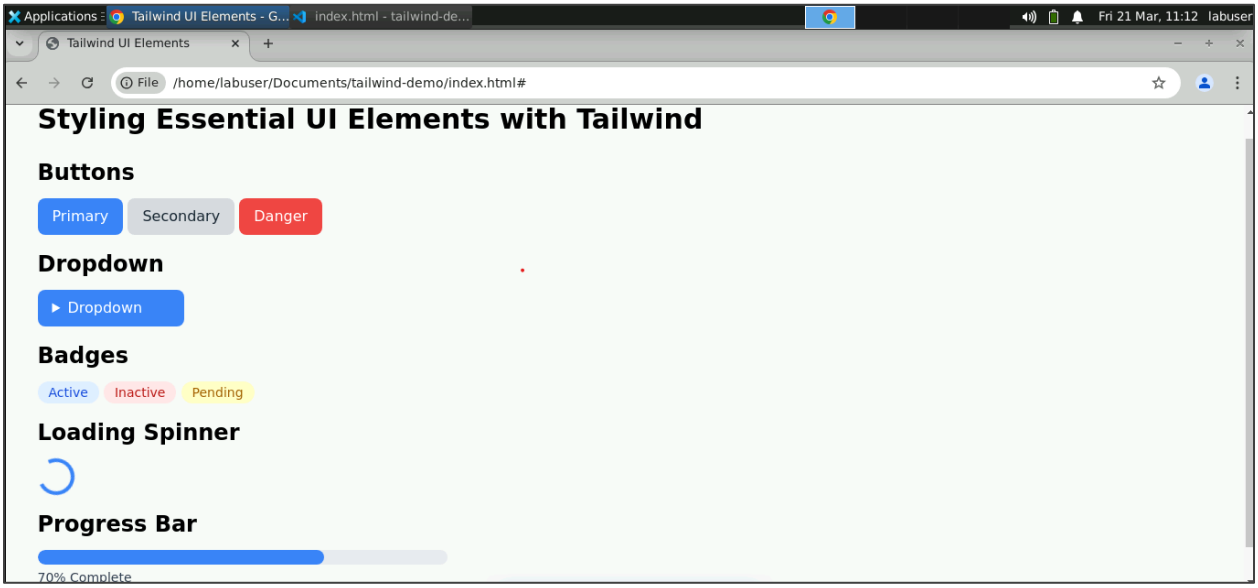
Step 7: Run the implementation in a web browser

7.1 Navigate to the project folder containing index.html file



By following this step, you navigate to the project folder named tailwind-demo and opens the index.html file in a browser.

7.2 Open the index.html in a web browser and verify all the UI elements



By following these steps, you have successfully styled essential UI elements using Tailwind CSS. Mastering Tailwind's utility classes allows you to build highly responsive, visually appealing, and maintainable web interfaces efficiently.