Todo with timer animation

You can add a small animation to visually show the countdown by adding a div with a progressing width at the bottom of each task item. Here’s an example that includes this animation:

1. Each task will have a progress div at the bottom of it.
2. The progress width will increase based on the timeoutDuration using CSS animation.

Here’s how to implement it:

jsx

Copy code

import React, { useState } from 'react';  
  
const TodoList = () => {  
 // State for tasks  
 const [tasks, setTasks] = useState([]);  
  
 // Function to add a new task with a timer  
 const addTask = (taskName, timeoutDuration) => {  
 const newTask = {  
 id: Date.now(),  
 name: taskName,  
 timeout: timeoutDuration  
 };  
 setTasks((prevTasks) => [...prevTasks, newTask]);  
  
 // Set up a timer to remove the task when timeout is reached  
 setTimeout(() => {  
 setTasks((prevTasks) => prevTasks.filter((task) => task.id !== newTask.id));  
 }, timeoutDuration);  
 };  
  
 return (  
 <div>  
 <h2>To-Do List</h2>  
 <ul>  
 {tasks.map((task) => (  
 <li key={task.id} className="task-item">  
 {task.name}  
 <div  
 className="progress-bar"  
 style={{  
 animation: `progressAnimation ${task.timeout}ms linear`  
 }}  
 ></div>  
 </li>  
 ))}  
 </ul>  
 {/\* Button to add a new task with a timer (example: 5 seconds) \*/}  
 <button onClick={() => addTask("New Task", 5000)}>Add Task with 5s Timer</button>  
 </div>  
 );  
};

export default TodoList;

**CSS for Animation**

You’ll need some CSS to style the task items and animate the progress-bar. Here’s an example:

css

Copy code

.task-item {

position: relative;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 4px;

}

.progress-bar {

position: absolute;

bottom: 0;

left: 0;

height: 4px;

background-color: #4caf50;

width: 0%;

}

/\* Define the animation \*/

@keyframes progressAnimation {

from {

width: 0%;

}

to {

width: 100%;

}

}

**Explanation of the CSS and Animation**

1. **.task-item:** Styles the task item container with padding and border.
2. **.progress-bar:** The bar at the bottom of each task. The width is initially set to 0% and then animated to 100%.
3. **@keyframes progressAnimation:** Defines the animation from width: 0% to width: 100%.
4. **Inline Style Animation Timing:** The animation property in the inline style (animation: progressAnimation ${task.timeout}ms linear) dynamically sets the duration based on each task’s timeout.

This animation will smoothly expand the progress bar from left to right until the timer ends, providing a visual countdown for each task.