9. Build a To-do List Application using React with state and prop

```
import React, { useState } from 'react';
import './App.css'; // Assuming you create this file for external styles

// --- Icon Components (Inline SVG to avoid external dependencies) ---

const Trash2Icon = (props) => (
  <svg {...props} xmlns="http://www.w3.org/2000/svg" width="20" height="20"
viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2"
strokeLinecap="round" strokeLinejoin="round" className="icon-trash">
    <path d="M3 6h18"/>
    <path d="M19 6v14c0 1-1 2-2 2H7c-1 0-2-1-2-2V6"/>
    <path d="M10 11v6"/>
    <path d="M14 11v6"/>
    <path d="M14 2h-4c-1 0-2 1-2 2v2h8V4c0-1-1-2-2-2z"/>
  </svg>
);

const PlusIcon = (props) => (
    <svg {...props} xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2"
strokeLinecap="round" strokeLinejoin="round" className="icon-plus">
        <path d="M12 5v14"/><path d="M5 12h14"/>
    </svg>
);
const NewTaskInput = ({ value, onChange, onAdd }) => (
  <div className="new-task-input-container">
    <input
      type="text"
      placeholder="What needs to be done?"
      value={value}
      onChange={(e) => onChange(e.target.value)}
      onKeyDown={(e) => {
        if (e.key === 'Enter') {
          onAdd(value);
        }
      }}
      className="task-input-field"
    />
    <button
      onClick={() => onAdd(value)}
      className="add-task-button"
      disabled={value.trim() === ''}
    >
      <PlusIcon />
    </button>
  </div>
);
```

```
/ --- Main Application Component ---

const App = () => {
  // State for the list of tasks
  const [todos, setTodos] = useState([
    { id: 1, text: "Build a responsive React component", completed: true },
    { id: 2, text: "Use standard CSS for styling", completed: false },
    { id: 3, text: "Implement state management with hooks", completed: false
},
  ]);

  // State for the current input value of the new task
  const [newTaskText, setNewTaskText] = useState('');

  /**
   * Adds a new task to the list.
   * @param {string} text - The text of the new task.
   */
  const addTask = (text) => {
    if (text.trim() === '') return;

    const newTodo = {
      id: Date.now(), // Simple unique ID generation
      text: text.trim(),
      completed: false,
    };

    setTodos([...todos, newTodo]);
    setNewTaskText(''); // Clear the input field
  };

  /**
   * Toggles the completion status of a task by its ID.
   * @param {number} id - The ID of the task to toggle.
   */
  const toggleTask = (id) => {
    const updatedTodos = todos.map(todo =>
      todo.id === id ? { ...todo, completed: !todo.completed } : todo
    );
    setTodos(updatedTodos);
  };

  /**
   * Deletes a task by its ID.
   * @param {number} id - The ID of the task to delete.
   */
  const deleteTask = (id) => {
    const updatedTodos = todos.filter(todo => todo.id !== id);
```

```
  setTodos(updatedTodos);
};

// The TodoItem component displays a single task item.
const TodoItem = ({ todo, onToggle, onDelete }) => {
  return (
    <li className="todo-item">
      <div
        className="todo-item-content"
        onClick={() => onToggle(todo.id)}
      >
        <input
          type="checkbox"
          checked={todo.completed}
          onChange={() => onToggle(todo.id)}
          className="todo-checkbox"
        />
        <span className={`todo-text ${todo.completed ? 'completed' : ''}`}>
          {todo.text}
        </span>
      </div>

      {/* Delete Button */}
      <button
        onClick={(e) => {
          e.stopPropagation(); // Prevent toggling when deleting
          onDelete(todo.id);
        }}
        className="delete-button"
        aria-label="Delete task"
      >
        <Trash2Icon />
      </button>
    </li>
  );
};

return (
  <div className="app-container">
    <div className="todo-app-wrapper">
      <h1 className="app-title">
        To-Do List
      </h1>
      <p className="app-subtitle">
        Manage your daily tasks efficiently.
      </p>

      {/* Task Input (uses props for communication) */}
```

```
        <NewTaskInput
          value={newTaskText}
          onChange={setNewTaskText}
          onAdd={addTask}
        />

        {/* Task List */}
        {todos.length === 0 ? (
            <p className="empty-list-message">
                🎉 All done! Add a new task above.
            </p>
        ) : (
          <ul className="todo-list">
            {/* List items (uses props for communication) */}
            {todos.map(todo => (
              <TodoItem
                key={todo.id}
                todo={todo} // Passing the task object as a prop
                onToggle={toggleTask} // Passing the toggle handler as a prop
                onDelete={deleteTask} // Passing the delete handler as a prop
              />
            ))}
          </ul>
        )}
      </div>
    </div>
  );
};

export default App;
```

OUTPUT: