TASK-16

16. Write a Simple Counter:

– Display the total number of tasks and the number of completed tasks.

---

1. TodoApp.jsx (React Component):

```
// /src/components/TodoApp.jsx

import React, { Component } from 'react';
import '../styles/TodoApp.css'; // Import the CSS file for styling

class TodoApp extends Component {
 constructor(props) {
  super(props);
  this.state = {
   tasks: [], // Array to hold the list of tasks
   currentTask: '', // Input value for a new task
  };

  // Bind methods to 'this'
  this.addTask = this.addTask.bind(this);
  this.removeTask = this.removeTask.bind(this);
  this.toggleCompletion = this.toggleCompletion.bind(this);
  this.handleInputChange = this.handleInputChange.bind(this);
 }

 // Update state when the input field changes
 handleInputChange(event) {
  this.setState({ currentTask: event.target.value });
 }

 // Add a new task to the list
 addTask() {
  if (this.state.currentTask.trim() !== '') {
   this.setState((prevState) => ({
    tasks: [
     ...prevState.tasks,
     { name: prevState.currentTask, isCompleted: false }, // Add task object with name and
completion status
    ],
    currentTask: '', // Clear the input field after adding task
   }));
  }
```

```
  }

  // Remove a task by its index
  removeTask(taskIndex) {
    this.setState((prevState) => ({
      tasks: prevState.tasks.filter((_, index) => index !== taskIndex), // Remove task from the list
    }));
  }

  // Toggle the completion status of a task
  toggleCompletion(taskIndex) {
    this.setState((prevState) => {
      const updatedTasks = [...prevState.tasks]; // Make a copy of tasks
      updatedTasks[taskIndex].isCompleted = !updatedTasks[taskIndex].isCompleted; // Toggle
isCompleted
      return { tasks: updatedTasks }; // Return the updated tasks array
    });
  }

  render() {
    const totalTasks = this.state.tasks.length; // Calculate total number of tasks
    const completedTasks = this.state.tasks.filter(task => task.isCompleted).length; // Count
completed tasks

    return (
      <div className="todo-app">
        <h1>Todo List</h1>

        {/* Header showing total and completed tasks */}
        <div className="task-counter">
          <p>Total Tasks: {totalTasks}</p>
          <p>Completed Tasks: {completedTasks}</p>
        </div>

        {/* Input field and button to add tasks */}
        <div className="input-section">
          <input
            type="text"
            value={this.state.currentTask}
            onChange={this.handleInputChange}
            placeholder="Enter task"
          />
          <button onClick={this.addTask}>Add Task</button>
        </div>

        {/* Task list */}
        <div className="task-list">
```

```
      <ul>
       {this.state.tasks.map((task, index) => (
        <li
         key={index}
         className={task.isCompleted ? 'completed' : ''} // Add completed class if task is marked as completed
         style={{ backgroundColor: task.isCompleted ? 'green' : '' }} // Dynamic background color for completed tasks
        >
         {task.name}
         <button onClick={() => this.toggleCompletion(index)}>
          {task.isCompleted ? 'Undo' : 'Complete'} {/* Toggle text based on completion */}
         </button>
         <button onClick={() => this.removeTask(index)}>Remove</button>
        </li>
       ))}
      </ul>
     </div>
    </div>
   );
  }
}

export default TodoApp;
```

2. TodoApp.css (Styling):

```
/* /src/styles/TodoApp.css */

.todo-app {
 display: flex;
 flex-direction: column;
 align-items: center;
 width: 100%;
 max-width: 400px;
 margin: 50px auto;
 padding: 20px;
 background-color: #f9f9f9;
 border-radius: 10px;
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

h1 {
 font-size: 2em;
 margin-bottom: 20px;
}
```

```
.task-counter {
  width: 100%;
  margin-bottom: 20px;
  text-align: center;
  font-size: 1.2em;
  color: #333;
}

.input-section {
  display: flex;
  justify-content: space-between;
  width: 100%;
}

input[type='text'] {
  width: 70%;
  padding: 8px;
  font-size: 1.2em;
  margin-right: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

button {
  padding: 8px 16px;
  font-size: 1.2em;
  cursor: pointer;
  background-color: #4caf50;
  color: white;
  border: none;
  border-radius: 5px;
}

button:hover {
  background-color: #45a049;
}

.task-list {
  width: 100%;
  margin-top: 20px;
}

ul {
  list-style-type: none;
  padding: 0;
}
```

```
li {
  display: flex;
  justify-content: space-between;
  align-items: center;
  background-color: #f1f1f1;
  padding: 10px;
  margin-bottom: 10px;
  border-radius: 5px;
}

li.completed {
  text-decoration: line-through;
  background-color: #e0e0e0;
}

button {
  background-color: #f44336;
  color: white;
}

button:hover {
  background-color: #e53935;
}

li.completed {
  background-color: green;
  color: white;
}
```