```jsx
import React, { useState } from 'react';

const Scheduler = () => {
  const [processes, setProcesses] = useState([]);
  const [processName, setProcessName] = useState('');
  const [burstTime, setBurstTime] = useState('');
  const [priority, setPriority] = useState('');
  const [algorithm, setAlgorithm] = useState('SJF');

  // Add process to the list
  const addProcess = () => {
    if (processName && burstTime && priority) {
      const newProcess = {
        name: processName,
        burstTime: parseInt(burstTime),
        priority: parseInt(priority),
      };
      setProcesses([...processes, newProcess]);
      setProcessName('');
      setBurstTime('');
      setPriority('');
    }
  };

  // Sorting for SJF
  const sjfScheduling = (processList) => {
    return processList.sort((a, b) => a.burstTime - b.burstTime);
  };

  // Sorting for Priority Scheduling
  const priorityScheduling = (processList) => {
    return processList.sort((a, b) => a.priority - b.priority);
  };

  // Handler for the algorithm to schedule
  const handleScheduling = () => {
    let scheduledProcesses = [];
    if (algorithm === 'SJF') {
      scheduledProcesses = sjfScheduling([...processes]);
    } else if (algorithm === 'Priority') {
      scheduledProcesses = priorityScheduling([...processes]);
    }
    return calculateTimes(scheduledProcesses);
  };

  // Calculate Completion Time and Turnaround Time
  const calculateTimes = (scheduledProcesses) => {
    let currentTime = 0;
```

```
    let totalTurnaroundTime = 0;
    let totalWaitingTime = 0;

    return scheduledProcesses.map((process, index) => {
      const waitingTime = currentTime;
      const turnaroundTime = waitingTime + process.burstTime;
      const completionTime = turnaroundTime;

      currentTime += process.burstTime;

      totalTurnaroundTime += turnaroundTime;
      totalWaitingTime += waitingTime;

      return {
        ...process,
        waitingTime,
        turnaroundTime,
        completionTime,
      };
    });
  };

  const scheduledProcesses = handleScheduling();

  return (
    <div className="container mx-auto p-4">
      <h1 className="text-2xl font-bold mb-4">Priority Scheduling
Algorithms</h1>

      <div className="mb-4">
        <input
          type="text"
          placeholder="Process Name"
          value={processName}
          onChange={(e) => setProcessName(e.target.value)}
          className="border p-2 rounded mr-2"
        />
        <input
          type="number"
          placeholder="Burst Time"
          value={burstTime}
          onChange={(e) => setBurstTime(e.target.value)}
          className="border p-2 rounded mr-2"
        />
        <input
          type="number"
          placeholder="Priority"
          value={priority}
```

```
          onChange={(e) => setPriority(e.target.value)}
          className="border p-2 rounded mr-2"
        />
        <button onClick={addProcess} className="bg-blue-500 text-white p-2
rounded">
          Add Process
        </button>
      </div>

      <div className="mb-4">
        <label className="mr-2">Select Algorithm:</label>
        <select value={algorithm} onChange={(e) =>
setAlgorithm(e.target.value)} className="border p-2 rounded">
          <option value="SJF">Shortest Job First (SJF)</option>
          <option value="Priority">Priority Scheduling</option>
        </select>
      </div>

      <h2 className="text-xl font-semibold mb-2">Scheduled Processes</h2>
      <table className="table-auto border-collapse border border-gray-400">
        <thead>
          <tr>
            <th className="border border-gray-300 p-2">Process Name</th>
            <th className="border border-gray-300 p-2">Burst Time</th>
            <th className="border border-gray-300 p-2">Priority</th>
            <th className="border border-gray-300 p-2">Waiting Time</th>
            <th className="border border-gray-300 p-2">Turnaround Time</th>
            <th className="border border-gray-300 p-2">Completion Time</th>
          </tr>
        </thead>
        <tbody>
          {scheduledProcesses.map((process, index) => (
            <tr key={index}>
              <td className="border border-gray-300 p-2">{process.name}</td>
              <td className="border border-gray-300 p-
2">{process.burstTime}</td>
              <td className="border border-gray-300 p-
2">{process.priority}</td>
              <td className="border border-gray-300 p-
2">{process.waitingTime}</td>
              <td className="border border-gray-300 p-
2">{process.turnaroundTime}</td>
              <td className="border border-gray-300 p-
2">{process.completionTime}</td>
            </tr>
          ))}
        </tbody>
      </table>
```

```
    </div>
  );
};

export default Scheduler;
```