**Task Scheduler Simulation:**

**Implement a priority-based task scheduler using a doubly linked list. Each task has a priority and duration. When new tasks are added, they should be inserted at the correct position based on priority. Simulate task execution and removal once completed.**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Task Scheduler Simulation</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

}

.container {

max-width: 500px;

width: 100%;

text-align: center;

}

.task-input {

margin-bottom: 10px;

}

.task-list, .execution-log {

margin-top: 20px;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

max-height: 200px;

overflow-y: auto;

}

.task-item {

margin: 5px 0;

padding: 5px;

border-bottom: 1px solid #ddd;

}

.execution-log p {

margin: 5px 0;

}

</style>

</head>

<body>

<div class="container">

<h2>Task Scheduler Simulation</h2>

<div class="task-input">

<input type="text" id="taskPriority" placeholder="Priority (higher number = higher priority)" />

<input type="text" id="taskDuration" placeholder="Duration (ms)" />

<button onclick="addTask()">Add Task</button>

</div>

<div class="task-list" id="taskList">

<h3>Scheduled Tasks</h3>

<!-- Tasks will appear here -->

</div>

<button onclick="startExecution()">Start Execution</button>

<div class="execution-log" id="executionLog">

<h3>Execution Log</h3>

<!-- Execution log will appear here -->

</div>

</div>

<script>

class Task {

constructor(priority, duration) {

this.priority = priority;

this.duration = duration;

this.next = null;

this.prev = null;

}

}

class TaskScheduler {

constructor() {

this.head = null;

this.tail = null;

}

addTask(priority, duration) {

const newTask = new Task(priority, duration);

if (!this.head) {

this.head = this.tail = newTask;

} else {

let current = this.head;

while (current && current.priority >= newTask.priority) {

current = current.next;

}

if (!current) {

// Insert at the end

this.tail.next = newTask;

newTask.prev = this.tail;

this.tail = newTask;

} else if (!current.prev) {

// Insert at the beginning

newTask.next = this.head;

this.head.prev = newTask;

this.head = newTask;

} else {

// Insert in the middle

newTask.next = current;

newTask.prev = current.prev;

current.prev.next = newTask;

current.prev = newTask;

}

}

this.renderTasks();

}

executeTask() {

if (!this.head) return null;

const taskToExecute = this.head;

this.head = this.head.next;

if (this.head) {

this.head.prev = null;

} else {

this.tail = null;

}

return taskToExecute;

}

renderTasks() {

const taskList = document.getElementById("taskList");

taskList.innerHTML = "<h3>Scheduled Tasks</h3>";

let current = this.head;

while (current) {

const taskElement = document.createElement("div");

taskElement.className = "task-item";

taskElement.textContent = `Priority: ${current.priority}, Duration: ${current.duration} ms`;

taskList.appendChild(taskElement);

current = current.next;

}

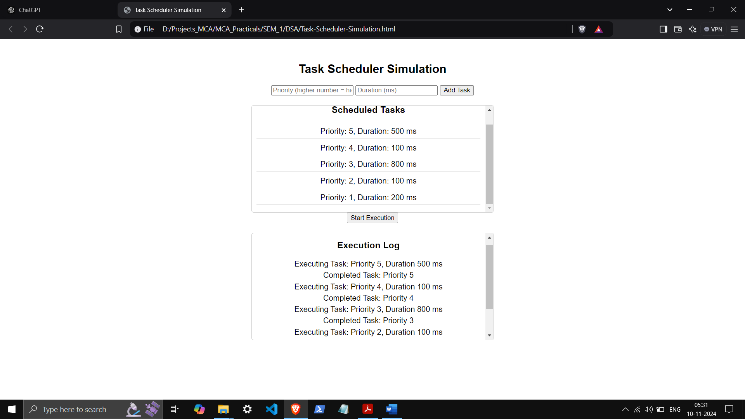
}

}

const scheduler = new TaskScheduler();

function addTask() {

const priority = parseInt(document.getElementById("taskPriority").value);

 const duration = parseInt(document.getElementById("taskDuration").value);

if (!isNaN(priority) && !isNaN(duration)) {

scheduler.addTask(priority, duration);

} else {

alert("Please enter valid numbers for priority and duration.");

}

document.getElementById("taskPriority").value = "";

document.getElementById("taskDuration").value = "";

}

async function startExecution() {

const log = document.getElementById("executionLog");

log.innerHTML = "<h3>Execution Log</h3>";

let task = scheduler.executeTask();

while (task) {

const logEntry = document.createElement("p");

logEntry.textContent = `Executing Task: Priority ${task.priority}, Duration ${task.duration} ms`;

log.appendChild(logEntry);

await new Promise(resolve => setTimeout(resolve, task.duration));

const completedEntry = document.createElement("p");

completedEntry.textContent = `Completed Task: Priority ${task.priority}`;

log.appendChild(completedEntry);

task = scheduler.executeTask();

}

}

</script>

</body>

</html>

