**Q4: Job Queue System**: Simulate a **job processing system** where jobs (like printing documents) are queued. Implement the queue with the ability to dynamically prioritize certain jobs (e.g., emergency print requests) using a **priority queue**.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Job Queue System</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

margin: 0;

background-color: #f4f4f9;

}

.job-system {

background: #ffffff;

padding: 20px;

border-radius: 10px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

text-align: center;

width: 400px;

}

h1 {

margin-bottom: 20px;

font-size: 1.5em;

}

input, button, select {

padding: 10px;

margin: 10px 0;

border: 1px solid #ddd;

border-radius: 5px;

width: 100%;

}

button {

background-color: #007BFF;

color: white;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

.queue {

margin-top: 20px;

text-align: left;

}

.queue p {

margin: 5px 0;

padding: 8px;

background-color: #f9f9f9;

border: 1px solid #ddd;

border-radius: 5px;

}

</style>

</head>

<body>

<div class="job-system">

<h1>Job Queue System</h1>

<input type="text" id="jobName" placeholder="Enter Job Name">

<select id="jobPriority">

<option value="Emergency">Emergency</option>

<option value="Regular">Regular</option>

</select>

<button id="addJob">Add Job</button>

<button id="processJob">Process Job</button>

<div class="queue">

<h3>Current Job Queue:</h3>

<div id="queueDisplay">No jobs in the queue.</div>

</div>

</div>

<script>

class PriorityQueue {

constructor() {

this.queue = [];

}

// Enqueue: Add job to the queue

enqueue(job) {

if (job.priority === "Emergency") {

this.queue.unshift(job); // Emergency jobs are prioritized

} else {

this.queue.push(job); // Regular jobs are added to the end

}

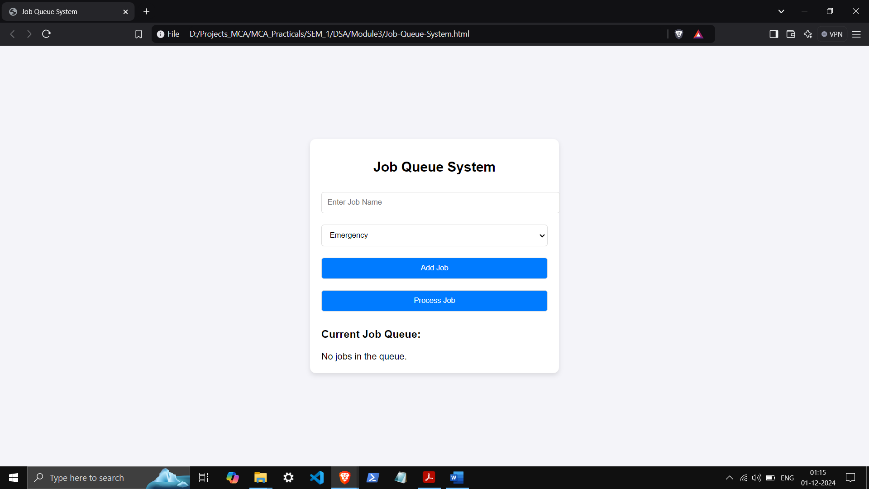
}

// Dequeue: Process the next job

dequeue() {

return this.queue.shift(); // Remove and return the first job

}



// Display the queue

display() {

if (this.queue.length === 0) {

return "No jobs in the queue.";

}

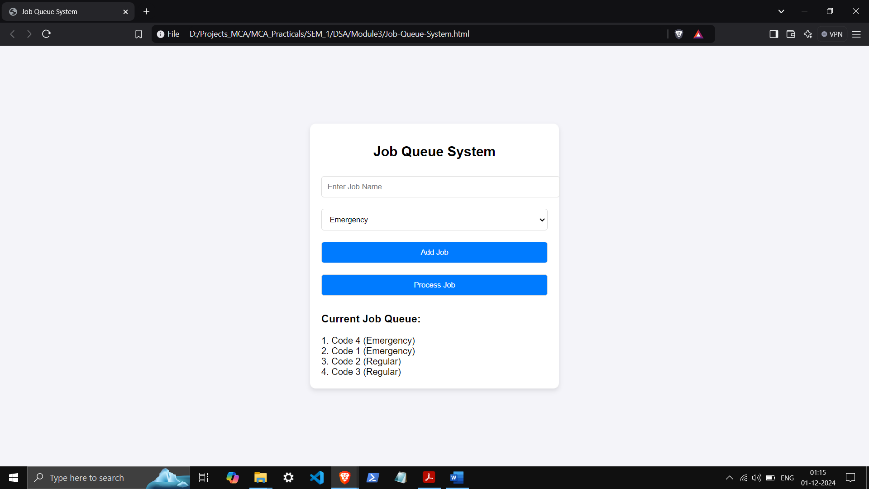
return this.queue

.map((job, index) => `${index + 1}. ${job.name} (${job.priority})`)

.join("<br>");

}

}



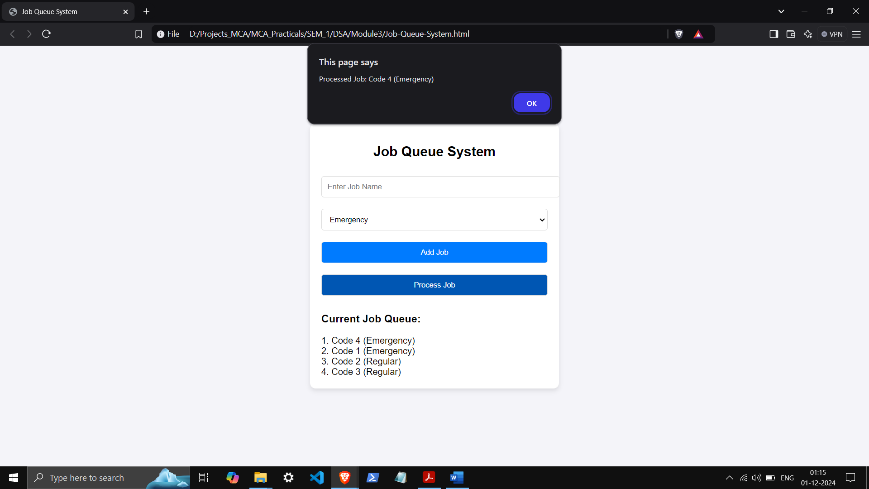
const jobQueue = new PriorityQueue();

document.getElementById("addJob").addEventListener("click", () => {

const jobName = document.getElementById("jobName").value.trim();

const jobPriority = document.getElementById("jobPriority").value;

if (!jobName) {

 alert("Please enter a job name.");

return;

}

jobQueue.enqueue({ name: jobName, priority: jobPriority });

document.getElementById("queueDisplay").innerHTML = jobQueue.display();

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

});

document.getElementById("processJob").addEventListener("click", () => {

const processedJob = jobQueue.dequeue();

if (processedJob) {

alert(`Processed Job: ${processedJob.name} (${processedJob.priority})`);

} else {

alert("No jobs to process.");

}

document.getElementById("queueDisplay").innerHTML = jobQueue.display();

});

</script>

</body>

</html>