**Q2: Online Ticketing System (Priority Queue)**: Design an **online ticketing system** using a **priority queue** where VIP customers are served first. Regular customers are served based on their order of arrival. Simulate ticket booking, cancellation, and serve operations, ensuring the system works under heavy traffic conditions

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Online Ticketing 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;

}

.ticket-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 {

padding: 10px;

margin: 10px 0;

border: 1px solid #ddd;

border-radius: 5px;

}

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="ticket-system">

<h1>Online Ticketing System</h1>

<input type="text" id="customerName" placeholder="Enter Customer Name">

<select id="customerType">

<option value="VIP">VIP</option>

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

</select>

<button id="bookTicket">Book Ticket</button>

<button id="serveCustomer">Serve Customer</button>

<div class="queue">

<h3>Queue Status:</h3>

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

</div>

</div>

<script>

class PriorityQueue {

constructor() {

this.queue = [];

}

// Enqueue: Add customer to the queue

enqueue(customer) {

if (customer.type === "VIP") {

this.queue.unshift(customer); // VIP customers get priority

} else {

this.queue.push(customer); // Regular customers are added to the end

}

}

// Dequeue: Serve the next customer

dequeue() {

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

}

// Display the queue

display() {

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

return "No customers in the queue.";

}

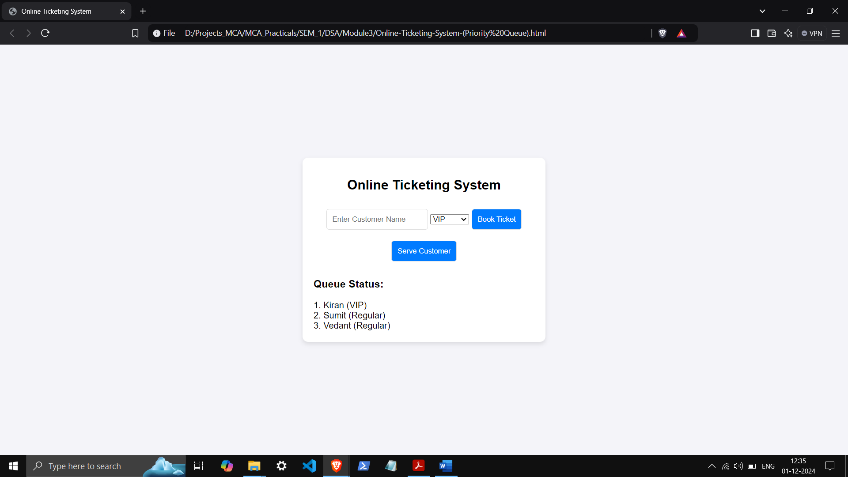
return this.queue

.map((customer, index) => `${index + 1}. ${customer.name} (${customer.type})`)

.join("<br>");

}

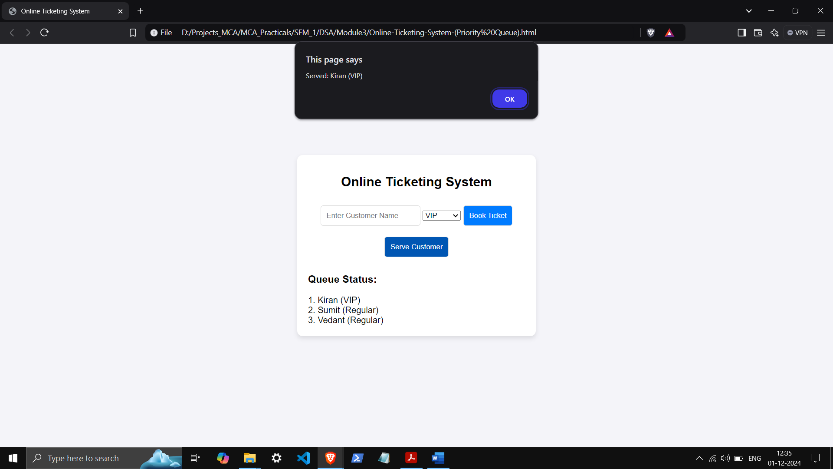
}

 const ticketQueue = new PriorityQueue();

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

const name = document.getElementById("customerName").value.trim();

const type = document.getElementById("customerType").value;

 if (!name) {

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

return;

}

ticketQueue.enqueue({ name, type });

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

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

});

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

const servedCustomer = ticketQueue.dequeue();

if (servedCustomer) {

alert(`Served: ${servedCustomer.name} (${servedCustomer.type})`);

} else {

alert("No customers to serve.");

}

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

});

</script>

</body>

</html>

