**Q6: Bank ATM Queue Simulation**: Implement a **bank ATM queue** where customers are queued for transactions. Simulate different types of transactions (deposit, withdrawal, balance check) with varying processing times. Use a **deque** (double-ended queue) to allow priority transactions at either end.

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bank ATM Queue Simulation</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

margin: 0;

background-color: #f4f4f9;

}

.atm-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, select, button {

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 {

padding: 8px;

background-color: #f9f9f9;

border: 1px solid #ddd;

border-radius: 5px;

}

</style>

</head>

<body>

<div class="atm-system">

<h1>ATM Queue Simulation</h1>

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

<select id="transactionType">

<option value="Deposit">Deposit (2s)</option>

<option value="Withdrawal">Withdrawal (3s)</option>

<option value="Balance Check">Balance Check (1s)</option>

</select>

<button id="addToQueue">Add to Queue</button>

<button id="processQueue">Process Queue</button>

<div class="queue">

<h3>Current Queue:</h3>

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

</div>

</div>

<script>

class Deque {

constructor() {

this.queue = [];

}

enqueue(customer, priority = false) {

if (priority) {

this.queue.unshift(customer); // Add priority customers to the front

} else {

this.queue.push(customer); // Add regular customers to the end

}

}

dequeue() {

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

}

isEmpty() {

return this.queue.length === 0;

}

display() {

if (this.isEmpty()) {

return "No customers in queue.";

}

return this.queue

.map(

(customer, index) =>

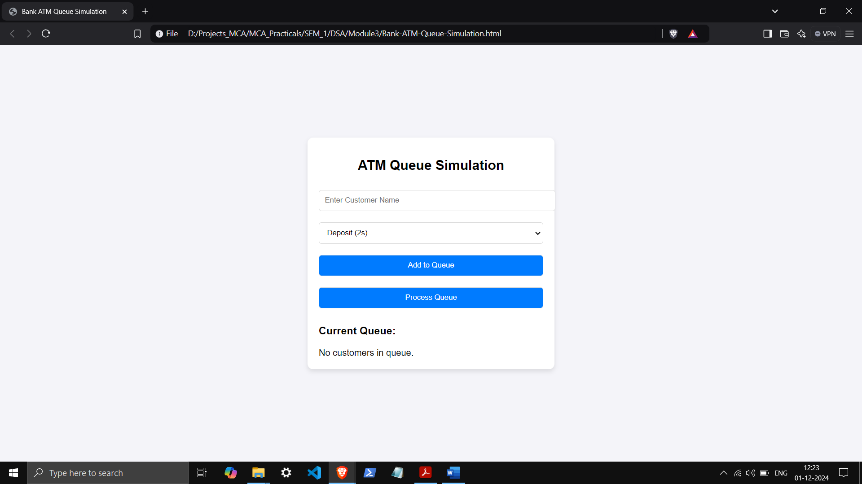
`${index + 1}. ${customer.name} (${customer.transaction})`

)

.join("<br>");

}

}

 const atmQueue = new Deque();

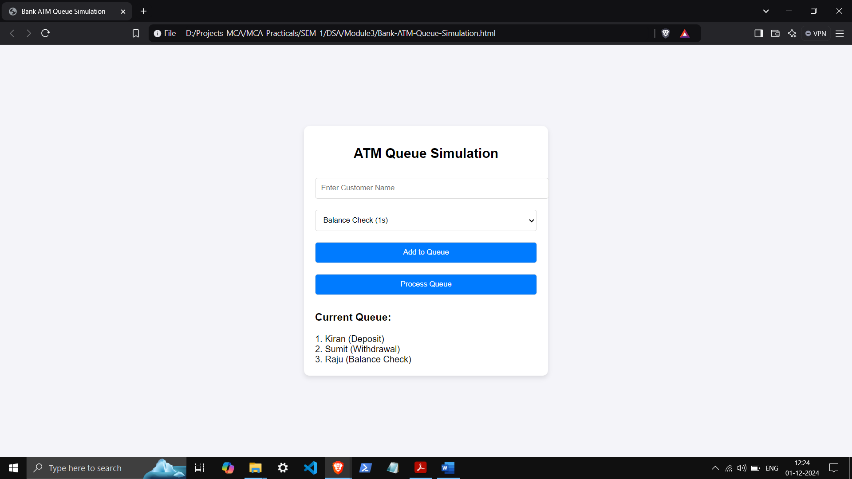
document.getElementById("addToQueue").addEventListener("click", () => {

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

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

if (!customerName) {

alert("Please enter the customer's name.");

 return;

}

// Add customer to the queue

atmQueue.enqueue({ name: customerName, transaction: transactionType });

document.getElementById("queueDisplay").innerHTML = atmQueue.display(); document.getElementById("customerName").value = "";

});

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

if (atmQueue.isEmpty()) {

alert("No customers to process.");

return;

}

const currentCustomer = atmQueue.dequeue();

const processingTimes = {

"Deposit": 2000,

"Withdrawal": 3000,

"Balance Check": 1000,

};

alert(`Processing ${currentCustomer.name}'s ${currentCustomer.transaction}...`);

setTimeout(() => {

alert(`${currentCustomer.name}'s ${currentCustomer.transaction} completed!`);

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

}, processingTimes[currentCustomer.transaction]);

});

</script>

</body>

</html>