**Train Route Simulation (Circular Linked List):**

**Simulate a train system where each station is represented as a node in a circular linked list. The train moves around the stations in a loop. Add operations to dynamically insert new stations or remove old ones, with proper memory management.**

<html lang="en">

<head>

<title>Train Route Simulation</title>

<style>

body {

font-family: Arial, sans-serif;

text-align: center;

background-color: #f4f4f9;

margin: 0;

padding: 0;

}

h1 {

color: #333;

}

.station-list, .controls {

margin: 20px auto;

max-width: 500px;

text-align: left;

}

.station-list ul {

list-style: none;

padding: 0;

}

.station-list ul li {

padding: 10px;

border: 1px solid #333;

margin: 5px;

background: #fff;

display: flex;

justify-content: space-between;

}

.train-status {

font-weight: bold;

color: #2a9d8f;

margin-top: 20px;

}

.controls input, .controls button {

margin: 5px;

padding: 10px;

border: none;

font-size: 1em;

cursor: pointer;

}

.controls button {

background-color: #2a9d8f;

color: white;

border-radius: 5px;

}

</style>

</head>

<body>

<h1>Train Route Simulation</h1>

<div class="controls">

<input type="text" id="stationName" placeholder="Enter station name">

<button onclick="addStation()">Add Station</button>

<button onclick="removeStation()">Remove Current Station</button>

<button onclick="startTrain()">Start Train</button>

</div>

<div class="station-list">

<h2>Stations on Route</h2>

<ul id="stationList"></ul>

</div>

<div class="train-status" id="trainStatus">Train is currently stopped.</div>

<script>

class Station {

constructor(name) {

this.name = name;

this.next = null;

}

}

class TrainRoute {

constructor() {

this.head = null;

this.current = null;

}

addStation(name) {

const newStation = new Station(name);

if (!this.head) {

this.head = newStation;

this.head.next = this.head;

} else {

let temp = this.head;

while (temp.next !== this.head) {

temp = temp.next;

}

temp.next = newStation;

newStation.next = this.head;

}

this.updateStationList();

}

removeStation() {

if (!this.head) {

alert("No stations to remove.");

return;

}

if (this.head.next === this.head) {

this.head = null;

this.current = null;

} else {

let prev = this.head;

while (prev.next !== this.current) {

prev = prev.next;

}

if (this.current === this.head) {

this.head = this.head.next;

}

prev.next = this.current.next;

}

this.current = this.head;

this.updateStationList();

}

startTrain() {

if (!this.head) {

alert("No stations on the route. Please add stations first.");

return;

}

this.current = this.head;

this.moveTrain();

}

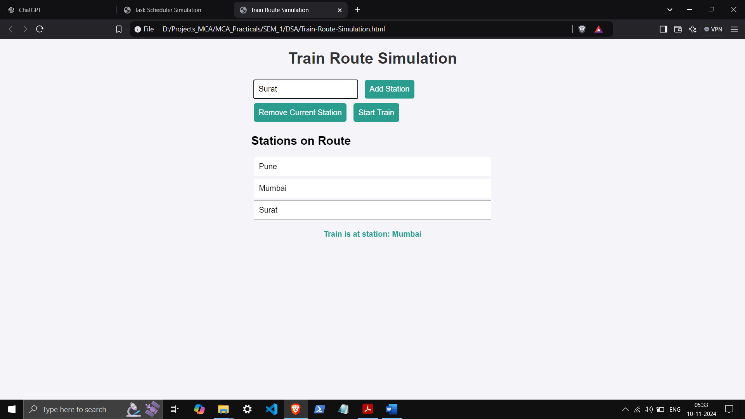
moveTrain() {

if (!this.current) return;

document.getElementById("trainStatus").textContent = `Train is at station: ${this.current.name}`;

this.current = this.current.next;

setTimeout(() => this.moveTrain(), 2000);

 }

updateStationList() {

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

stationList.innerHTML = "";

if (!this.head) return;

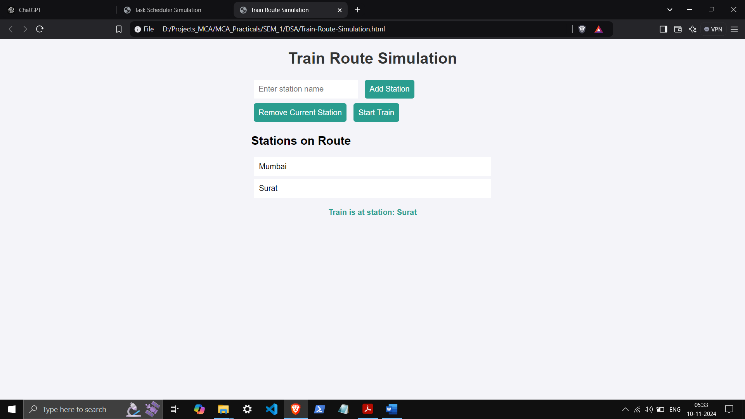
let temp = this.head;

do {

const li = document.createElement("li");

li.textContent = temp.name;

stationList.appendChild(li);

 temp = temp.next;

} while (temp !== this.head);

}

}

const trainRoute = new TrainRoute();

function addStation() {

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

if (stationName === "") {

alert("Please enter a valid station name.");

return;

}

trainRoute.addStation(stationName);

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

}

function removeStation() {

trainRoute.removeStation();

}

function startTrain() {

trainRoute.startTrain();

}

</script>

</body>

</html>

