**MODULE-1**

**Program-1:-**

Dynamic Inventory Management System: Design an inventory management system for a warehouse using a 1-D array. The array should store product details (ID, name, quantity, and price). Implement operations for adding new products, updating inventory levels, and calculating the total inventory value dynamically.

**Code:-**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Inventory Management System</title>

<style>

body {

font-family: 'Arial', sans-serif;

margin: 0;

background-color: #e9ecef;

color: #495057;

}

h1, h2 {

text-align: center;

color: #ee00ff;

}

.container {

max-width: 800px;

margin: 20px auto;

padding: 20px;

background: white;

border-radius: 8px;

box-shadow: 0 2px 15px rgba(0, 0, 0, 0.1);

}

input, button {

padding: 12px;

margin: 5px 0;

border-radius: 4px;

border: 1px solid #ced4da;

width: calc(100% - 22px);

box-sizing: border-box;

font-size: 1rem;

}

button {

background-color: #ee00ff;

color: white;

cursor: pointer;

border: none;

}

button:hover {

background-color: #ee00ff;

}

table {

margin-top: 20px;

border-collapse: collapse;

width: 100%;

}

th, td {

border: 1px solid #ddd;

padding: 12px;

text-align: left;

}

th {

background-color: #ee00ff;

color: white;

}

tr:hover {

background-color: #f1f1f1;

}

#totalValue {

font-weight: bold;

margin-top: 10px;

text-align: center;

font-size: 1.2em;

}

.button-container {

display: flex;

justify-content: center;

margin-top: 20px;

}

.button-container button {

margin: 0 10px;

}

</style>

</head>

<body>

<div class="container">

<h1>Inventory Management System</h1>

<h2>Add Product</h2>

<input type="text" id="productId" placeholder="Product ID" required>

<input type="text" id="productName" placeholder="Product Name" required>

<input type="number" id="productQuantity" placeholder="Quantity" required>

<input type="number" id="productPrice" placeholder="Price" required>

<button onclick="addProduct()">Add Product</button>

<h2>Update Inventory</h2>

<input type="text" id="updateProductId" placeholder="Product ID" required>

<input type="number" id="updateQuantity" placeholder="New Quantity" required>

<button onclick="updateProduct()">Update Inventory</button>

<h2>Inventory</h2>

<div class="button-container">

<button onclick="calculateTotalValue()">Calculate Total Value</button>

</div>

<div id="totalValue"></div>

<table id="inventoryTable">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Quantity</th>

<th>Price</th>

</tr>

</thead>

<tbody>

</tbody>

</table>

</div>

<script>

let inventory = [];

function addProduct() {

const id = document.getElementById('productId').value;

const name = document.getElementById('productName').value;

const quantity = parseInt(document.getElementById('productQuantity').value);

const price = parseFloat(document.getElementById('productPrice').value);

const product = { id, name, quantity, price };

inventory.push(product);

document.getElementById('productId').value = '';

document.getElementById('productName').value = '';

document.getElementById('productQuantity').value = '';

document.getElementById('productPrice').value = '';

displayInventory();

}

function updateProduct() {

const id = document.getElementById('updateProductId').value;

const newQuantity = parseInt(document.getElementById('updateQuantity').value);

const product = inventory.find(item => item.id === id);

if (product) {

product.quantity = newQuantity;

document.getElementById('updateProductId').value = '';

document.getElementById('updateQuantity').value = '';

displayInventory();

} else {

alert('Product not found!');

}

}

function displayInventory() {

const tbody = document.querySelector('#inventoryTable tbody');

tbody.innerHTML = '';

inventory.forEach(item => {

const row = document.createElement('tr');

row.innerHTML = `<td>${item.id}</td><td>${item.name}</td><td>${item.quantity}</td><td>${item.price.toFixed(2)}</td>`;

tbody.appendChild(row);

});

}

function calculateTotalValue() {

const total = inventory.reduce((sum, item) => sum + (item.quantity \* item.price), 0);

document.getElementById('totalValue').innerText = `Total Inventory Value: $${total.toFixed(2)}`;

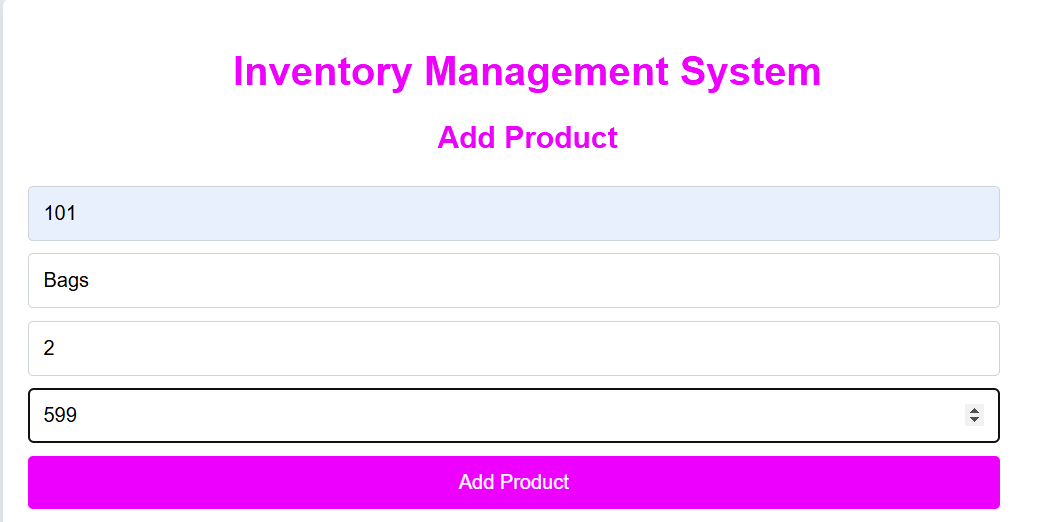
}

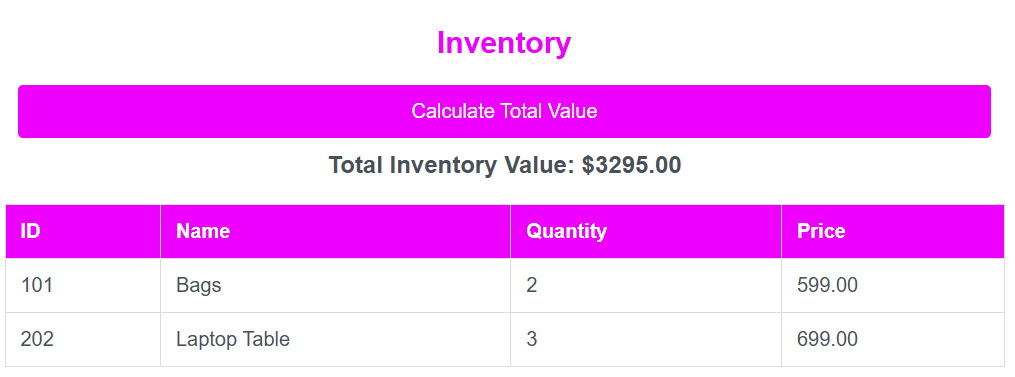
</script>

</body>

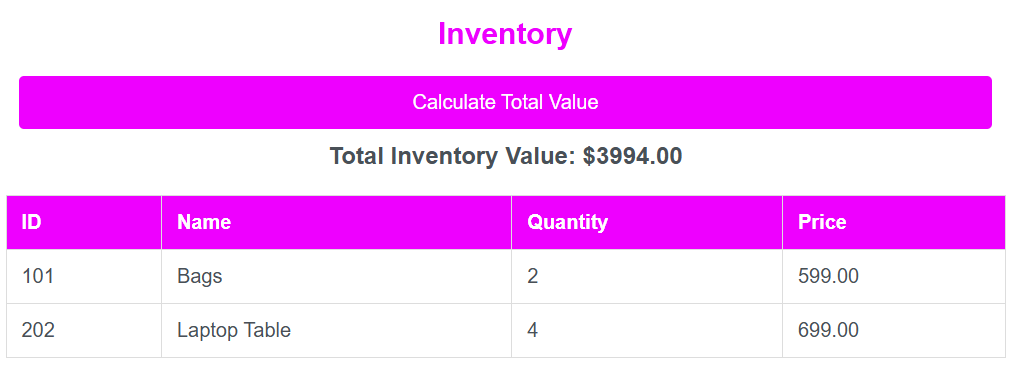
</html>

Output :-









**MODULE-1**

**Program-2 :-**

Real-time Seating Arrangement: Create a 2-D array to represent seating arrangements for a movie theater. The system should handle seat booking and cancellation requests dynamically. Add constraints for group bookings, ensuring they are seated together.

**Code:-**

**HTML File :-**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

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

<title>Movie Theater Seating</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="container">

<h1>Movie Theater Seating Arrangement</h1>

<div class="controls">

<div>

<h3>Book Seats</h3>

<input type="number" id="numSeats" placeholder="Number of Seats" min="1">

<button onclick="bookSeats()">Book</button>

<p class="error-message" id="bookError"></p>

</div>

<div>

<h3>Cancel Seat</h3>

<input type="number" id="row" placeholder="Row Number" min="0">

<input type="number" id="col" placeholder="Seat Number" min="0">

<button onclick="cancelSeat()">Cancel</button>

<p class="error-message" id="cancelError"></p>

</div>

</div>

<div class="seating-grid" id="seatingGrid"></div>

</div>

<script src="script.js"></script>

</body>

</html>

**CSS File:-**

\* {

box-sizing: border-box;

margin: 0;

padding: 0;

}

body {

font-family: Arial, sans-serif;

background-color: #f8f9fa;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

padding: 20px;

}

.container {

max-width: 800px;

width: 100%;

background-color: #ffffff;

border-radius: 8px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

padding: 20px;

text-align: center;

}

h1 {

margin-bottom: 20px;

color: #333;

}

.controls {

display: flex;

justify-content: space-between;

margin-bottom: 20px;

}

.controls div {

flex: 1;

margin: 0 10px;

}

input {

width: 80%;

padding: 8px;

margin-top: 5px;

}

button {

padding: 8px 12px;

margin-top: 10px;

background-color: #007bff;

color: #fff;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 14px;

}

button:hover {

background-color: #0056b3;

}

.error-message {

color: red;

font-size: 12px;

margin-top: 5px;

display: none;

}

.seating-grid {

display: grid;

grid-template-columns: repeat(10, 1fr); /\* Adjust to the number of seats per row \*/

gap: 5px;

}

.seat {

width: 30px;

height: 30px;

background-color: #28a745;

border-radius: 4px;

display: flex;

justify-content: center;

align-items: center;

cursor: pointer;

color: #fff;

font-size: 14px;

}

.seat.booked {

background-color: #dc3545;

cursor: not-allowed;

}

**JS File:-**

const rows = 10; // Number of rows

const cols = 10; // Number of seats per row

let seatingArrangement = Array.from({ length: rows }, () => Array(cols).fill(false));

createSeatingGrid();

function createSeatingGrid() {

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

seatingGrid.innerHTML = "";

for (let row = 0; row < rows; row++) {

for (let col = 0; col < cols; col++) {

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

seat.classList.add("seat");

seat.textContent = `${row}-${col}`;

if (seatingArrangement[row][col])

{

seat.classList.add("booked");

}

seat.dataset.row = row;

seat.dataset.col = col;

seatingGrid.appendChild(seat);

}

}

}

function bookSeats() {

const numSeats = parseInt(document.getElementById("numSeats").value);

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

bookError.style.display = "none";

if (isNaN(numSeats) || numSeats <= 0) {

bookError.textContent = "Enter a valid number of seats.";

bookError.style.display = "block";

return;

}

let booked = false;

for (let row = 0; row < rows; row++) {

for (let col = 0; col <= cols - numSeats; col++) {

if (seatingArrangement[row].slice(col, col + numSeats).every(seat => !seat)) {

for (let i = 0; i < numSeats; i++) {

seatingArrangement[row][col + i] = true;

}

booked = true;

break;

}

}

if (booked) break;

}

if (!booked) {

bookError.textContent = "Cannot book seats together. Try fewer seats.";

bookError.style.display = "block";

}

createSeatingGrid();

}

function cancelSeat() {

const row = parseInt(document.getElementById("row").value);

const col = parseInt(document.getElementById("col").value);

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

cancelError.style.display = "none";

if (isNaN(row) || isNaN(col) || row < 0 || row >= rows || col < 0 || col >= cols) {

cancelError.textContent = "Enter valid row and seat numbers.";

cancelError.style.display = "block";

return;

}

if (!seatingArrangement[row][col]) {

cancelError.textContent = "This seat is not booked.";

cancelError.style.display = "block";

return;

}

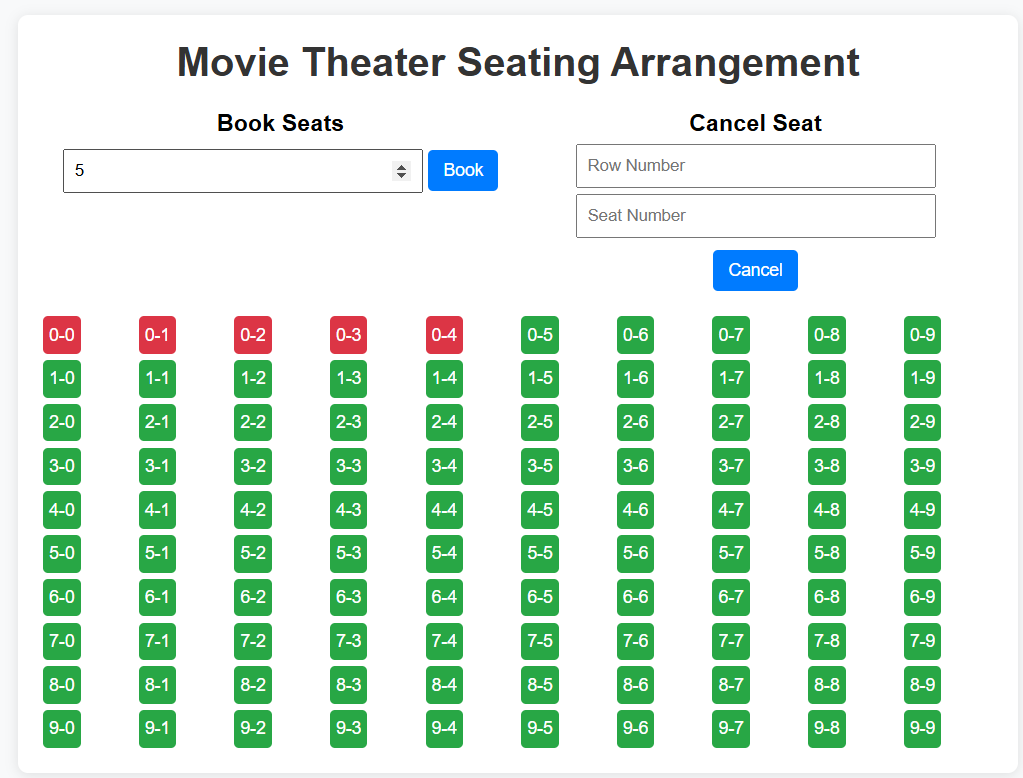
seatingArrangement[row][col] = false;

createSeatingGrid();

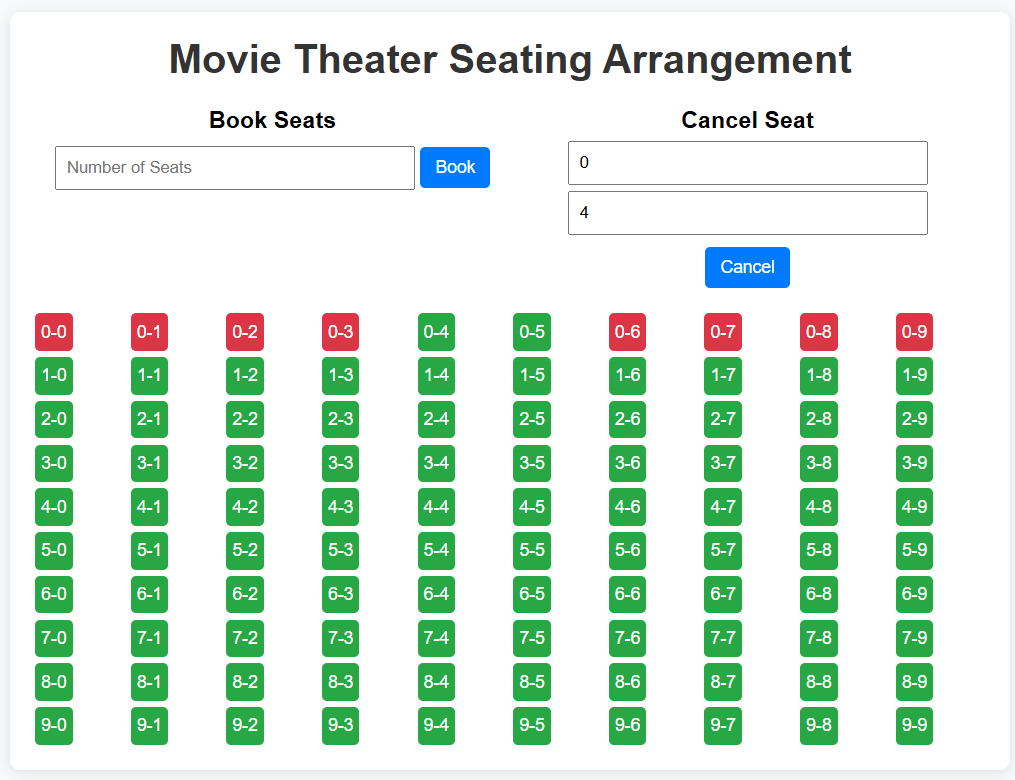
}

**Output :-**

Book the ticket in group of Movie Theater.



Cancle the ticket of Movie Theater.



**MODULE-1**

**Program-3 :-**

Traffic Data Analysis: Collect real-time traffic data (e.g., car counts, speed, and entry/exit times) on multiple road lanes using a dynamic array. Analyze traffic flow patterns for peak hours using array operations like sorting, searching, and aggregation.

**Code:-**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Traffic Data Analysis</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

background-color: #f2f2f2;

}

h1 {

text-align: center;

color: #4CAF50;

}

.container {

max-width: 800px;

margin: 20px auto;

padding: 20px;

background: white;

border-radius: 8px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

}

input, button {

padding: 10px;

margin: 5px 0;

width: calc(100% - 22px);

box-sizing: border-box;

font-size: 1rem;

}

button {

background-color: #007bff;

color: white;

cursor: pointer;

border: none;

}

button:hover {

background-color: #0056b3;

}

#results {

margin-top: 20px;

text-align: center;

}

#trafficDataTable {

margin-top: 20px;

border-collapse: collapse;

width: 100%;

}

#trafficDataTable th, #trafficDataTable td {

border: 1px solid #ddd;

padding: 8px;

}

#trafficDataTable th {

background-color: #4CAF50;

color: white;

}

</style>

</head>

<body>

<div class="container">

<h1>Traffic Data Analysis</h1>

<input type="text" id="lane" placeholder="Lane (e.g. Lane 1)">

<input type="number" id="carCount" placeholder="Number of Cars" min="1">

<input type="number" id="averageSpeed" placeholder="Average Speed (km/h)">

<input type="text" id="entryTime" placeholder="Entry Time (HH:MM)">

<input type="text" id="exitTime" placeholder="Exit Time (HH:MM)">

<button onclick="addTrafficData()">Add Traffic Data</button>

<button onclick="analyzeTrafficData()">Analyze Traffic Data</button>

<div id="results"></div>

<table id="trafficDataTable">

<thead>

<tr>

<th>Lane</th>

<th>Car Count</th>

<th>Average Speed</th>

<th>Entry Time</th>

<th>Exit Time</th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

<script>

let trafficData = [];

function addTrafficData() {

const lane = document.getElementById('lane').value;

const carCount = parseInt(document.getElementById('carCount').value);

const averageSpeed = parseFloat(document.getElementById('averageSpeed').value);

const entryTime = document.getElementById('entryTime').value;

const exitTime = document.getElementById('exitTime').value;

const data = { lane, carCount, averageSpeed, entryTime, exitTime };

trafficData.push(data);

document.getElementById('lane').value = '';

document.getElementById('carCount').value = '';

document.getElementById('averageSpeed').value = '';

document.getElementById('entryTime').value = '';

document.getElementById('exitTime').value = '';

displayTrafficData();

}

function displayTrafficData() {

const tbody = document.querySelector('#trafficDataTable tbody');

tbody.innerHTML = '';

trafficData.forEach(data => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${data.lane}</td>

<td>${data.carCount}</td>

<td>${data.averageSpeed}</td>

<td>${data.entryTime}</td>

<td>${data.exitTime}</td>

`;

tbody.appendChild(row);

});

}

function analyzeTrafficData() {

if (trafficData.length === 0) {

document.getElementById('results').innerText = 'No traffic data available.';

return;

}

// Calculate total car count and average speed

const totalCars = trafficData.reduce((sum, data) => sum + data.carCount, 0);

const averageSpeed = trafficData.reduce((sum, data) => sum + data.averageSpeed, 0) / trafficData.length;

// Find peak hours

const peakHours = {};

trafficData.forEach(data => {

const entryHour = data.entryTime.split(':')[0];

peakHours[entryHour] = (peakHours[entryHour] || 0) + data.carCount;

});

// Find the hour with the maximum cars

const peakHour = Object.keys(peakHours).reduce((a, b) => peakHours[a] > peakHours[b] ? a : b);

document.getElementById('results').innerHTML = `

<h2>Traffic Analysis Results</h2>

<p>Total Cars: ${totalCars}</p>

<p>Average Speed: ${averageSpeed.toFixed(2)} km/h</p>

<p>Peak Hour: ${peakHour}:00 with ${peakHours[peakHour]} cars</p>

`;

}

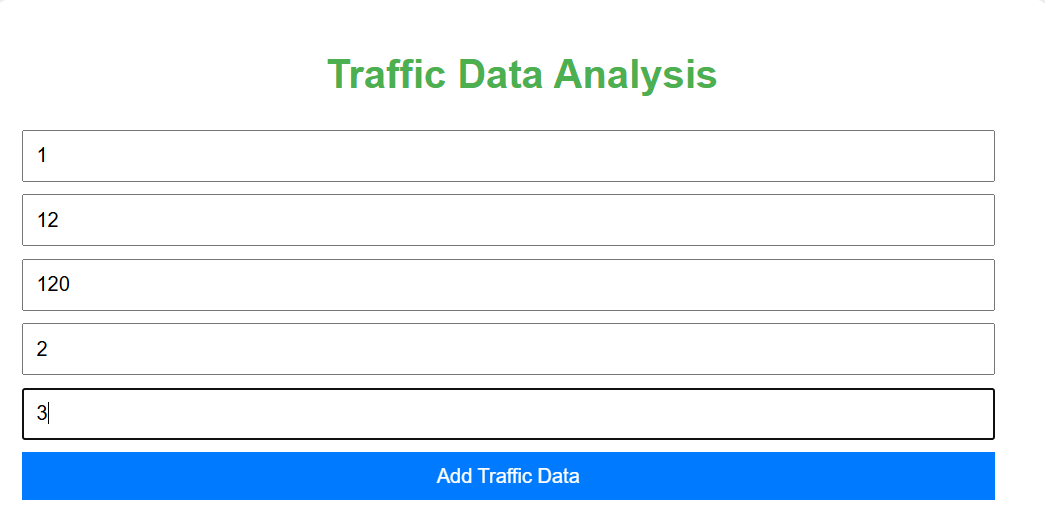
</script>

</body>

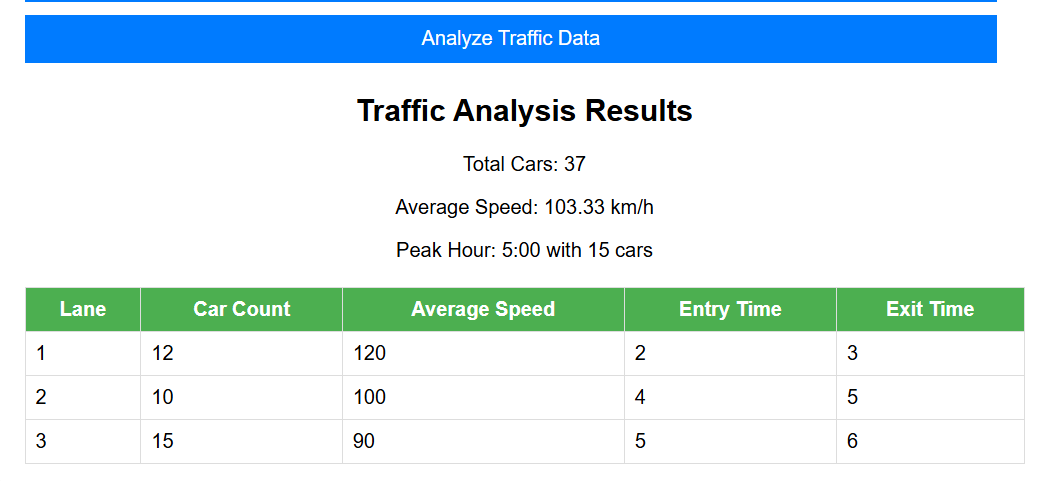
</html>

**Output :-**

Add Car Details for Analysis.



Analysis the traffic data and give result.



**MODULE-1**

**Program-4 :-**

Multi-Dimensional Data Representation: Design a multi-dimensional array to represent geographical data (e.g., latitude, longitude, altitude) for a drone-based mapping system. Implement functions to extract and manipulate data for different regions dynamically.

**Code:-**

**HTML File :-**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

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

<title>Drone-Based Mapping System</title>

<link rel="stylesheet" href="styles.css">

<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />

<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>

</head>

<body>

<div class="container">

<header>

<h1>Real-Time Drone-Based Mapping System</h1>

</header>

<section class="map-section">

<h2>Geographical Data Visualization</h2>

<div id="map" class="map"></div>

</section>

<section class="input-section">

<h2>Add Geographical Data</h2>

<label for="latitude">Latitude:</label>

<input type="number" id="latitude" step="0.0001">

<label for="longitude">Longitude:</label>

<input type="number" id="longitude" step="0.0001">

<label for="altitude">Altitude (m):</label>

<input type="number" id="altitude" step="1">

<button onclick="addGeoData()">Add Data</button>

</section>

<section class="filter-section">

<h2>Filter Data by Region</h2>

<label for="minLat">Min Latitude:</label>

<input type="number" id="minLat" step="0.0001">

<label for="maxLat">Max Latitude:</label>

<input type="number" id="maxLat" step="0.0001">

<label for="minLon">Min Longitude:</label>

<input type="number" id="minLon" step="0.0001">

<label for="maxLon">Max Longitude:</label>

<input type="number" id="maxLon" step="0.0001">

<button onclick="filterByRegion()">Filter Data</button>

</section>

<section class="sort-section">

<h2>Sort Data</h2>

<button onclick="sortDataByAltitude()">Sort by Altitude</button>

</section>

<section class="data-section">

<h2>Geographical Data List</h2>

<div id="geoDataList">No data available.</div>

</section>

</div>

<script src="script.js"></script>

</body>

</html>

**CSS File :-**

body {

font-family: 'Arial', sans-serif;

margin: 0;

padding: 0;

background-color: #f0f4f8;

}

.container {

width: 80%;

margin: 0 auto;

padding: 20px;

}

header {

text-align: center;

margin-bottom: 30px;

}

h1 {

font-size: 32px;

color: #333;

}

h2 {

color: #333;

}

.map-section {

margin-top: 20px;

}

#map {

height: 500px;

width: 100%;

}

.input-section, .data-section {

background-color: white;

padding: 20px;

border-radius: 8px;

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

margin-top: 20px;

}

input {

width: 100%;

padding: 8px;

margin: 5px 0;

border-radius: 4px;

border: 1px solid #ddd;

}

button {

padding: 10px 20px;

background-color: #007BFF;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

#geoDataList {

font-size: 14px;

color: #333;

}

**JS File :-**

// Initialize map using Leaflet

const map = L.map('map').setView([0, 0], 2); // Center at [0, 0] with zoom level 2

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

maxZoom: 18,

}).addTo(map);

// Initialize an array to store geographical data

let geoData = [];

// Function to add geographical data to the array

function addGeoData() {

const latitude = parseFloat(document.getElementById("latitude").value);

const longitude = parseFloat(document.getElementById("longitude").value);

const altitude = parseFloat(document.getElementById("altitude").value);

// Check if inputs are valid

if (isNaN(latitude) || isNaN(longitude) || isNaN(altitude)) {

alert("Please enter valid geographical data.");

return;

}

// Push new data to geoData array

geoData.push([latitude, longitude, altitude]);

// Add marker to the map

const marker = L.marker([latitude, longitude]).addTo(map);

marker.bindPopup(`Altitude: ${altitude} m`).openPopup();

// Update data list

updateGeoDataList();

}

// Function to display geographical data on the page

function updateGeoDataList() {

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

geoDataList.innerHTML = ""; // Clear the previous list

if (geoData.length === 0) {

geoDataList.textContent = "No data available.";

return;

}

geoData.forEach((data, index) => {

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

listItem.textContent = `Data Point ${index + 1} - Latitude: ${data[0]}, Longitude: ${data[1]}, Altitude: ${data[2]} m`;

geoDataList.appendChild(listItem);

});

}

// Add event listener for map clicks to fill form fields

map.on('click', function (e) {

const { lat, lng } = e.latlng;

// Simulate altitude data (since we don't have real elevation data)

const simulatedAltitude = (Math.random() \* 2000).toFixed(2); // Simulate altitude between 0 and 2000 meters

// Fill the form fields with the clicked location data

document.getElementById("latitude").value = lat.toFixed(4);

document.getElementById("longitude").value = lng.toFixed(4);

document.getElementById("altitude").value = simulatedAltitude;

// Optionally, center the map to the clicked location

map.setView([lat, lng], map.getZoom());

});

// Function to filter geographical data by region

function filterByRegion() {

const minLat = parseFloat(document.getElementById("minLat").value);

const maxLat = parseFloat(document.getElementById("maxLat").value);

const minLon = parseFloat(document.getElementById("minLon").value);

const maxLon = parseFloat(document.getElementById("maxLon").value);

// Filter geoData based on the region bounds

const filteredData = geoData.filter(data => {

return data[0] >= minLat && data[0] <= maxLat &&

data[1] >= minLon && data[1] <= maxLon;

});

// Update the displayed list with filtered data

updateFilteredGeoDataList(filteredData);

}

// Function to update the displayed list of filtered geographical data

function updateFilteredGeoDataList(filteredData) {

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

geoDataList.innerHTML = ""; // Clear the previous list

if (filteredData.length === 0) {

geoDataList.textContent = "No data found for the selected region.";

return;

}

filteredData.forEach((data, index) => {

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

listItem.textContent = `Data Point ${index + 1} - Latitude: ${data[0]}, Longitude: ${data[1]}, Altitude: ${data[2]} m`;

geoDataList.appendChild(listItem);

});

}

// Function to sort data by altitude

function sortDataByAltitude() {

const sortedData = [...geoData].sort((a, b) => b[2] - a[2]); // Sort in descending order by altitude

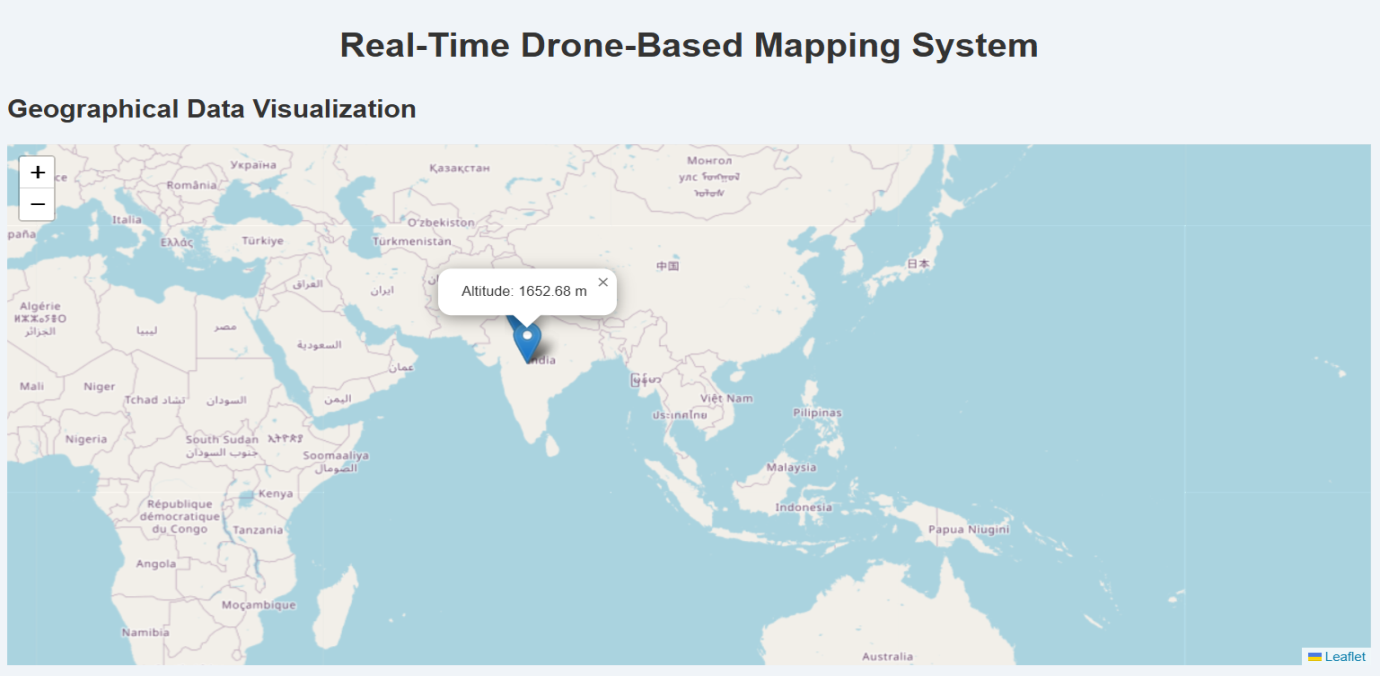
// Update the displayed list with sorted data

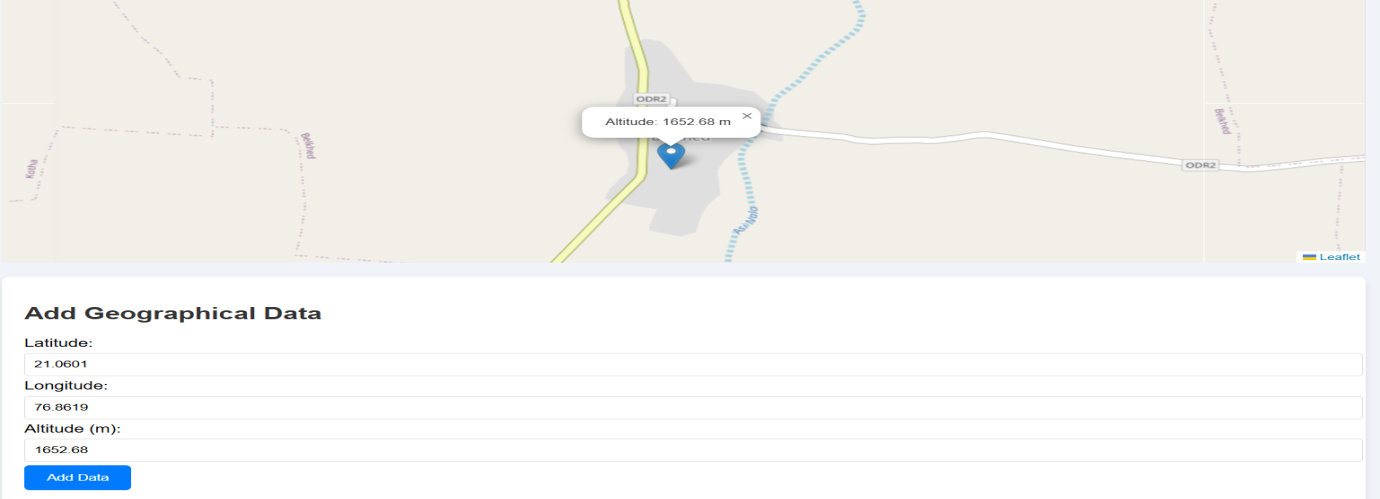
updateGeoDataList(sortedData);

}

**Output :-**

Add Data by choosing location from map.





Filter Data and find easily in data list.

