TRAFFIC MANAGEMENT SYSTEM

INTRODUCTION:

Here is a sample code which I have created using html, css, and javascript to create a website for traffic management system.

SAMPLE CODE:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Real-Time Traffic Information</title>

    <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />

    <style>

      /* Reset default margin and padding */

      * {

          margin: 0;

          padding: 0;

          box-sizing: border-box;

}


body {
```

```
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    margin: 0;
}

h1 {
    font-size: 24px;
    margin-bottom: 20px;
    text-align: center;
}

#map {
    height: 60vh;
    width: 100%;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}
```

```
#search-bar {
    background-color: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
    margin-top: 20px;
    width: 80%;
    max-width: 400px;
}

label, input, button {
    margin: 10px 0;
    display: block;
}

input {
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    width: 100%;
}

button {
```

```
    background-color: #3887be;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}

button:hover {
    background-color: #326fa7;
}
    </style>
</head>
<body>
    <h1>Real-Time Traffic Information</h1>
    <div id="search-bar">
        <label for="location">From: </label>
        <input type="text" id="location" placeholder="Your location">
        <label for="destination">To: </label>
        <input type="text" id="destination" placeholder="Your destination">
        <button id="search-button">Search</button>
</div>
```

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


<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
<script>
    // Create a Leaflet map instance
    var map = L.map('map').setView([37.7749, -122.4194], 12); // Default to
San Francisco


    // Add a basemap (OpenStreetMap)
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        maxZoom: 19,
    }).addTo(map);


    // Function to add a route on the map
    function addRoute(location, destination) {
        var routeUrl =
`https://www.mapquestapi.com/directions/v2/route?key=QwNMrtsZNUVQXE
Rv9agY4GfMbXjZ5tXT&from=${location}&to=${destination}`;


        fetch(routeUrl)
.then(function(response) {
    return response.json();
})
.then(function(data) {
```

```
                if (data.info.statuscode === 0) {
                    var coordinates = data.route.shape.shapePoints;
                    var polyline = L.polyline(L.Polyline.fromEncoded(coordinates), {
color: 'blue' }).addTo(map);
                    map.fitBounds(polyline.getBounds());
                } else {
                    alert("Route not found. Please check your input.");
                }
            });
        }

        // Event listener for the search button
        document.getElementById('search-button').addEventListener('click',
function() {
            var location = document.getElementById('location').value;
            var destination = document.getElementById('destination').value;
            addRoute(location, destination);
        });
    </script>
</body>
</html>
```

CODE EXPLANATION:

The provided HTML and JavaScript code is for a web page that allows users to input a source (From) and a destination (To), and then displays the route

between these two locations on a map using Leaflet, a popular open-source mapping library. Here's an explanation of the code:

HTML Section:

1. The HTML document is structured with the usual **<!DOCTYPE>**, **<html>**, **<head>**, and **<body>** elements.
2. Inside the **<head>**, metadata like character set and viewport settings are defined, and the web page's title is set to "Real-Time Traffic Information."
3. The page loads the Leaflet CSS by including the **<link>** tag for the Leaflet CSS from a content delivery network (CDN).
4. Custom styles are defined within a **<style>** element. These styles apply to various elements on the page to improve its appearance and layout. This includes resetting default margin and padding, setting a background color, styling the search bar, and more.

JavaScript Section:

1. The JavaScript code is wrapped inside a **<script>** element at the bottom of the HTML document to ensure it runs after the page is loaded.
2. It starts by creating a Leaflet map instance with **L.map('map')** and sets its initial view to San Francisco (latitude 37.7749, longitude -122.4194) with a zoom level of 12.
3. A base map from OpenStreetMap is added to the map using **L.tileLayer**. This provides the underlying map data for visualization.
4. The **addRoute** function is defined to fetch and display a route on the map. It takes the "From" (location) and "To" (destination) as input.
5. Inside the **addRoute** function:
6. A URL is constructed with the source and destination locations, and a MapQuest API key.
7. A **fetch** request is made to the MapQuest Directions API to obtain the route information.
8. When the response is received, it's parsed as JSON, and the code checks if the status code (**data.info.statuscode**) is 0, which indicates a successful route retrieval.

9. If the route is found, it retrieves the route's shape coordinates from **data.route.shape.shapePoints**. It creates a polyline on the map using Leaflet to visualize the route and adjusts the map view to fit the bounds of the polyline.
10. If the route is not found (status code is not 0), an alert is shown indicating that the route was not found. This handles cases where the input locations are not valid.
11. An event listener is added to the "Search" button. When the button is clicked, it triggers the **addRoute** function with the values entered in the "From" and "To" input fields.

Overall, this code provides a simple web application that allows users to search for routes between two locations and visualizes the route on a map using Leaflet. Make sure to replace the API key with your own when you deploy the application.

CONCLUSION:

- We use the Leaflet library for mapping, which is a popular open-source alternative to Mapbox.
- We load the OpenStreetMap as the base map.
- We make use of the MapQuest Open Traffic Data API to fetch traffic incidents data (you need to replace **'YOUR_MAPQUEST_API_KEY'** with your actual MapQuest API key).
- You should implement the logic to process and display the traffic data on the map as needed. This could involve adding markers, popups, or other visual elements to represent traffic incidents.
- Remember to obtain your MapQuest API key and adapt the traffic data handling according to your specific requirements.