Traffic Telligence code

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>TrafficTelligence</title>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.9.1/chart.min.js"></script>
 <style>
* {
 margin: 0;
 padding: 0;
 box-sizing: border-box;
 }
 body {
 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
 background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
 min-height: 100vh;
 color: #333;
 }
 .container {
 max-width: 1400px;
 margin: 0 auto;
```

```
padding: 20px;

}

.header {

text-align: center;

color: white;

margin-bottom: 30px;

}

.header h1 {

font-size: 2.5rem;

margin-bottom: 10px;

text-shadow: 2px 2px 4px rgba(0,0,0,0.3);

}

.header p {

font-size: 1.1rem;

opacity: 0.9;

}

.dashboard {

display: grid;

grid-template-columns: 1fr 1fr 1fr;

gap: 20px;

margin-bottom: 30px;

}

.card {

background: rgba(255, 255, 255, 0.95);

border-radius: 15px;
```

```
padding: 25px;

box-shadow: 0 8px 32px rgba(0,0,0,0.1);

backdrop-filter: blur(10px);

border: 1px solid rgba(255,255,255,0.2);

}

.card h3 {

color: #4a5568;

margin-bottom: 20px;

font-size: 1.3rem;

}

.input-group {

margin-bottom: 15px;

}

.input-group label {

display: block;

margin-bottom: 5px;

font-weight: 600;

color: #2d3748;

}

.input-group input, .input-group select {

width: 100%;

padding: 12px;

border: 2px solid #e2e8f0;

border-radius: 8px;

font-size: 16px;
```

```
transition: border-color 0.3s ease;
}
.input-group input:focus, .input-group select:focus {
outline: none;
border-color: #667eea;
box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}
.btn {
background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
color: white;
border: none;
padding: 12px 24px;
border-radius: 8px;
cursor: pointer;
font-size: 16px;
font-weight: 600;
transition: transform 0.2s ease;
width: 100%;
}
.btn:hover {
transform: translateY(-2px);
box-shadow: 0 4px 12px rgba(102, 126, 234, 0.3);
}
.results {
display: grid;
```

```
grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
gap: 20px;
margin-bottom: 30px;
}
.metric-card {
background: rgba(255, 255, 255, 0.95);
border-radius: 15px;
padding: 20px;
text-align: center;
box-shadow: 0 8px 32px rgba(0,0,0,0.1);
}
.metric-value {
font-size: 2rem;
font-weight: bold;
color: #667eea;
margin-bottom: 5px;
}
.metric-label {
color: #718096;
font-size: 0.9rem;
}
.chart-container {
background: rgba(255, 255, 255, 0.95);
border-radius: 15px;
padding: 25px;
```

```
box-shadow: 0 8px 32px rgba(0,0,0,0.1);

margin-bottom: 20px;

}

/* Traffic Map Styles */

.map-container {

background: rgba(255, 255, 255, 0.95);

border-radius: 15px;

padding: 25px;

box-shadow: 0 8px 32px rgba(0,0,0,0.1);

margin-bottom: 20px;

}

.traffic-map {

position: relative;

width: 100%;

height: 400px;

background: linear-gradient(45deg, #f7fafc 25%, transparent 25%),

linear-gradient(-45deg, #f7fafc 25%, transparent 25%),

linear-gradient(45deg, transparent 75%, #f7fafc 75%),

linear-gradient(-45deg, transparent 75%, #f7fafc 75%);

background-size: 20px 20px;

background-position: 0 0, 0 10px, 10px -10px, -10px 0px;

border-radius: 10px;

overflow: hidden;

}

.road {
```

```
position: absolute;

background: #4a5568;

z-index: 1;

}

.road-horizontal {

height: 8px;

width: 100%;

}

.road-vertical {

width: 8px;

height: 100%;

}

.intersection {

position: absolute;

width: 20px;

height: 20px;

background: #2d3748;

border-radius: 50%;

z-index: 3;

transform: translate(-50%, -50%);

}

.traffic-sensor {

position: absolute;

width: 12px;

height: 12px;
```

```
border-radius: 50%;

z-index: 4;

transform: translate(-50%, -50%);

cursor: pointer;

transition: all 0.3s ease;

box-shadow: 0 0 10px rgba(0,0,0,0.3);

}

.traffic-sensor:hover {

transform: translate(-50%, -50%) scale(1.5);

}

.sensor-low { background: #48bb78; animation: pulse-green 2s infinite; }

.sensor-medium { background: #ed8936; animation: pulse-orange 2s infinite; }

.sensor-high { background: #f56565; animation: pulse-red 2s infinite; }

@keyframes pulse-green {

0%, 100% { box-shadow: 0 0 10px rgba(72, 187, 120, 0.5); }

50% { box-shadow: 0 0 20px rgba(72, 187, 120, 0.8); }

}

@keyframes pulse-orange {

0%, 100% { box-shadow: 0 0 10px rgba(237, 137, 54, 0.5); }

50% { box-shadow: 0 0 20px rgba(237, 137, 54, 0.8); }

}

@keyframes pulse-red {

0%, 100% { box-shadow: 0 0 10px rgba(245, 101, 101, 0.5); }

50% { box-shadow: 0 0 20px rgba(245, 101, 101, 0.8); }

}
```

```css
.traffic-flow {
position: absolute;
width: 6px;
height: 6px;
background: white;
border-radius: 50%;
z-index: 2;
animation: flow 3s linear infinite;
}
@keyframes flow {
0% { opacity: 0; }
10% { opacity: 1; }
90% { opacity: 1; }
100% { opacity: 0; }
}
.map-legend {
display: flex;
justify-content: space-around;
margin-top: 20px;
padding: 15px;
background: rgba(247, 250, 252, 0.8);
border-radius: 10px;
}
.legend-item {
display: flex;
```

```
align-items: center;

gap: 8px;

}

.legend-dot {

width: 12px;

height: 12px;

border-radius: 50%;

}

.area-label {

position: absolute;

background: rgba(255, 255, 255, 0.9);

padding: 4px 8px;

border-radius: 6px;

font-size: 12px;

font-weight: 600;

color: #2d3748;

z-index: 5;

box-shadow: 0 2px 8px rgba(0,0,0,0.1);

}

.scenarios {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(350px, 1fr));

gap: 20px;

}

.scenario-card {
```

```css
background: rgba(255, 255, 255, 0.95);
border-radius: 15px;
padding: 25px;
box-shadow: 0 8px 32px rgba(0,0,0,0.1);
}
.scenario-card h4 {
color: #4a5568;
margin-bottom: 15px;
font-size: 1.2rem;
}
.scenario-card p {
color: #718096;
line-height: 1.6;
margin-bottom: 15px;
}
.status-indicator {
display: inline-block;
width: 10px;
height: 10px;
border-radius: 50%;
margin-right: 8px;
}
.status-low { background-color: #48bb78; }
.status-medium { background-color: #ed8936; }
.status-high { background-color: #f56565; }
```

```
.user-location-marker {

position: absolute;

width: 16px;

height: 16px;

background: #3b82f6;

border: 3px solid white;

border-radius: 50%;

z-index: 6;

transform: translate(-50%, -50%);

box-shadow: 0 0 15px rgba(59, 130, 246, 0.6);

animation: pulse-blue 2s infinite;

}

@keyframes pulse-blue {

0%, 100% { box-shadow: 0 0 15px rgba(59, 130, 246, 0.6); }

50% { box-shadow: 0 0 25px rgba(59, 130, 246, 0.9); }

}

.location-analysis {

background: rgba(255, 255, 255, 0.95);

border-radius: 15px;

padding: 25px;

box-shadow: 0 8px 32px rgba(0,0,0,0.1);

margin-bottom: 20px;

}

.analysis-grid {

display: grid;
```

```
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

gap: 15px;

margin-top: 20px;

}

.analysis-item {

background: rgba(103, 126, 234, 0.1);

padding: 15px;

border-radius: 10px;

text-align: center;

}

.analysis-value {

font-size: 1.5rem;

font-weight: bold;

color: #667eea;

margin-bottom: 5px;

}

.analysis-label {

color: #4a5568;

font-size: 0.9rem;

}

.location-recommendations {

background: rgba(52, 211, 153, 0.1);

border-left: 4px solid #34d399;

padding: 15px;

margin-top: 15px;
```

```
border-radius: 0 8px 8px 0;
}
.location-recommendations h5 {
color: #047857;
margin-bottom: 10px;
}
.location-recommendations ul {
color: #065f46;
margin-left: 20px;
}
.tooltip {
position: absolute;
background: rgba(0, 0, 0, 0.9);
color: white;
padding: 8px 12px;
border-radius: 6px;
font-size: 12px;
pointer-events: none;
z-index: 1000;
opacity: 0;
transition: opacity 0.3s ease;
}
@media (max-width: 768px) {
.dashboard {
grid-template-columns: 1fr;
```

```
 }

 .header h1 {
 font-size: 2rem;
 }
 .map-legend {
 flex-direction: column;
 gap: 10px;
 }
 }
 </style>
</head>
<body>
 <div class="container">
 <div class="header">
 <h1>□ TrafficTelligence</h1>
 <p>Advanced Traffic Volume Estimation with Machine Learning</p>
 </div>
 <div class="dashboard">
 <div class="card">
 <h3>□ Current Location Analysis</h3>
 <div class="input-group">
 <button class="btn" onclick="getCurrentLocation()" id="locationBtn">□ Get My
Location</button>
 </div>
```

```
<div id="locationInfo" style="display: none;">

<div class="input-group">

<label>Your Coordinates</label>

<input type="text" id="coordinates" readonly>

</div>

<div class="input-group">

<label>Nearest Traffic Zone</label>

<input type="text" id="nearestZone" readonly>

</div>

<div class="input-group">

<label>Current Traffic Volume</label>

<input type="text" id="userLocationVolume" readonly>

</div>

<div class="input-group">

<label>Traffic Status</label>

<input type="text" id="userLocationStatus" readonly>

</div>

</div>

</div>

<div class="card">

<h3>⍰ Traffic Prediction Input</h3>

<div class="input-group">

<label for="hour">Hour of Day (0-23)</label>

<input type="number" id="hour" min="0" max="23" value="8">

</div>
```

```
<div class="input-group">
<label for="day">Day of Week</label>
<select id="day">
<option value="0">Sunday</option>
<option value="1" selected>Monday</option>
<option value="2">Tuesday</option>
<option value="3">Wednesday</option>
<option value="4">Thursday</option>
<option value="5">Friday</option>
<option value="6">Saturday</option>
</select>
</div>
<div class="input-group">
<label for="weather">Weather Condition</label>
<select id="weather">
<option value="clear" selected>Clear</option>
<option value="rain">Rain</option>
<option value="snow">Snow</option>
<option value="fog">Fog</option>
</select>
</div>
<div class="input-group">
<label for="temperature">Temperature (°C)</label>
<input type="number" id="temperature" value="22">
</div>
```

```
<div class="input-group">
<label for="event">Special Event</label>
<select id="event">
<option value="none" selected>None</option>
<option value="concert">Concert</option>
<option value="sports">Sports Event</option>
<option value="festival">Festival</option>
<option value="holiday">Holiday</option>
</select>
</div>
<button class="btn" onclick="predictTraffic()">□ Predict Traffic Volume</button>
</div>
<div class="dashboard">
<div class="card">
<h3>□ Traffic Prediction Input</h3>
<div class="input-group">
<label for="hour">Hour of Day (0-23)</label>
<input type="number" id="hour" min="0" max="23" value="8">
</div>
<div class="input-group">
<label for="day">Day of Week</label>
<select id="day">
<option value="0">Sunday</option>
<option value="1" selected>Monday</option>
<option value="2">Tuesday</option>
```

```
<option value="3">Wednesday</option>

<option value="4">Thursday</option>

<option value="5">Friday</option>

<option value="6">Saturday</option>

</select>

</div>

<div class="input-group">

<label for="weather">Weather Condition</label>

<select id="weather">

<option value="clear" selected>Clear</option>

<option value="rain">Rain</option>

<option value="snow">Snow</option>

<option value="fog">Fog</option>

</select>

</div>

<div class="input-group">

<label for="temperature">Temperature (°C)</label>

<input type="number" id="temperature" value="22">

</div>

<div class="input-group">

<label for="event">Special Event</label>

<select id="event">

<option value="none" selected>None</option>

<option value="concert">Concert</option>

<option value="sports">Sports Event</option>
```

```
<option value="festival">Festival</option>
<option value="holiday">Holiday</option>
</select>
</div>
<button class="btn" onclick="predictTraffic()"> Predict Traffic Volume</button>
</div>
<div class="card">
<h3> Real-time Controls</h3>
<div class="input-group">
<label for="location">Monitor Location</label>
<select id="location">
<option value="downtown" selected>Downtown Core</option>
<option value="highway">Highway Junction</option>
<option value="suburban">Suburban Area</option>
<option value="industrial">Industrial Zone</option>
</select>
</div>
<div class="input-group">
<label for="timeRange">Analysis Time Range</label>
<select id="timeRange">
<option value="1">Last 1 Hour</option>
<option value="6" selected>Last 6 Hours</option>
<option value="24">Last 24 Hours</option>
<option value="168">Last Week</option>
</select>
```

```
</div>
<button class="btn" onclick="updateDashboard()">□ Update Dashboard</button>
<button class="btn" onclick="generateRecommendations()" style="margin-top:
10px;">□ Get Recommendations</button>
</div>
</div>
</div>
<div class="location-analysis" id="locationAnalysis" style="display: none;">
<h3>□ Your Location Traffic Analysis</h3>
<div class="analysis-grid">
<div class="analysis-item">
<div class="analysis-value" id="userTrafficVolume">-</div>
<div class="analysis-label">Current Volume</div>
</div>
<div class="analysis-item">
<div class="analysis-value" id="userTrafficTrend">-</div>
<div class="analysis-label">Traffic Trend</div>
</div>
<div class="analysis-item">
<div class="analysis-value" id="userWaitTime">-</div>
<div class="analysis-label">Est. Wait Time</div>
</div>
<div class="analysis-item">
<div class="analysis-value" id="userAlternativeRoutes">-</div>
<div class="analysis-label">Alternative Routes</div>
```

```
</div>
</div>
<div class="location-recommendations" id="userRecommendations">
<h5>□ Personalized Recommendations</h5>
<ul id="recommendationsList">
<li>Getting your location data...</li>
</ul>
</div>
</div>
<div class="results" id="results">
<div class="metric-card">
<div class="metric-value" id="currentVolume">1,247</div>
<div class="metric-label">Current Volume (vehicles/hour)</div>
</div>
<div class="metric-card">
<div class="metric-value" id="predictedVolume">1,389</div>
<div class="metric-label">Predicted Next Hour</div>
</div>
<div class="metric-card">
<div class="metric-value" id="congestionLevel">Medium</div>
<div class="metric-label">Congestion Level</div>
</div>
<div class="metric-card">
<div class="metric-value" id="accuracy">94.2%</div>
<div class="metric-label">Model Accuracy</div>
```

```
</div>

</div>

<div class="map-container">

<h3>� Real-time Traffic Map</h3>

<div class="traffic-map" id="trafficMap">

<!-- Roads -->

<div class="road road-horizontal" style="top: 20%;"></div>

<div class="road road-horizontal" style="top: 50%;"></div>

<div class="road road-horizontal" style="top: 80%;"></div>

<div class="road road-vertical" style="left: 25%;"></div>

<div class="road road-vertical" style="left: 50%;"></div>

<div class="road road-vertical" style="left: 75%;"></div>

<!-- Intersections -->

<div class="intersection" style="left: 25%; top: 20%;"></div>

<div class="intersection" style="left: 50%; top: 20%;"></div>

<div class="intersection" style="left: 75%; top: 20%;"></div>

<div class="intersection" style="left: 25%; top: 50%;"></div>

<div class="intersection" style="left: 50%; top: 50%;"></div>

<div class="intersection" style="left: 75%; top: 50%;"></div>

<div class="intersection" style="left: 25%; top: 80%;"></div>

<div class="intersection" style="left: 50%; top: 80%;"></div>

<div class="intersection" style="left: 75%; top: 80%;"></div>

<!-- Area Labels -->

<div class="area-label" style="left: 15%; top: 10%;">Industrial Zone</div>

<div class="area-label" style="left: 60%; top: 10%;">Suburban Area</div>
```

```
<div class="area-label" style="left: 35%; top: 35%;">Downtown Core</div>
<div class="area-label" style="left: 85%; top: 70%;">Highway Junction</div>
<!-- Traffic Sensors -->
<div class="traffic-sensor sensor-medium" style="left: 25%; top: 20%;" datalocation="Industrial-Main" data-volume="892"></div>
<div class="traffic-sensor sensor-high" style="left: 50%; top: 20%;" datalocation="Downtown-North" data-volume="1456"></div>
<div class="traffic-sensor sensor-low" style="left: 75%; top: 20%;" datalocation="Suburban-East" data-volume="634"></div>
<div class="traffic-sensor sensor-high" style="left: 25%; top: 50%;" datalocation="Downtown-West" data-volume="1523"></div>
<div class="traffic-sensor sensor-high" style="left: 50%; top: 50%;" datalocation="Downtown-Center" data-volume="1789"></div>
<div class="traffic-sensor sensor-medium" style="left: 75%; top: 50%;" datalocation="Suburban-Central" data-volume="1123"></div>
<div class="traffic-sensor sensor-low" style="left: 25%; top: 80%;" datalocation="Industrial-South" data-volume="567"></div>
<div class="traffic-sensor sensor-medium" style="left: 50%; top: 80%;" datalocation="Downtown-South" data-volume="987"></div>
<div class="traffic-sensor sensor-high" style="left: 75%; top: 80%;" datalocation="Highway-Main" data-volume="1698"></div>
</div>
<div class="map-legend">
<div class="legend-item">
<div class="legend-dot" style="background: #48bb78;"></div>
<span>Low Traffic (< 900)</span>
</div>
<div class="legend-item">
<div class="legend-dot" style="background: #ed8936;"></div>
```

```
<span>Medium Traffic (900-1400)</span>
</div>
<div class="legend-item">
<div class="legend-dot" style="background: #f56565;"></div>
<span>High Traffic (> 1400)</span>
</div>
<div class="legend-item">
<div class="legend-dot" style="background: #4a5568;"></div>
<span>Traffic Sensor</span>
</div>
</div>
</div>
<div class="chart-container">
<h3>□ Traffic Volume Prediction Chart</h3>
<canvas id="trafficChart" width="400" height="200"></canvas>
</div>
<div class="scenarios">
<div class="scenario-card">
<h4>□ Dynamic Traffic Management</h4>
<p><span class="status-indicator status-medium"></span><strong>Status:</strong>
Active Optimization</p>
<p>Signal timings automatically adjusted based on predicted volume. Estimated
congestion reduction: 23%</p>
<p><strong>Next Action:</strong> Implement alternate routing at 5:30 PM</p>
</div>
```

```
 <div class="scenario-card">
 <h4>⍰ Urban Development Planning</h4>
 <p><span class="status-indicator status-low"></span><strong>Status:</strong>
Analysis Complete</p>
 <p>Recommended infrastructure improvements identified for Q3 2025. Projected
traffic increase: 15% by 2026</p>
 <p><strong>Priority:</strong> Expand Highway Junction capacity</p>
 </div>
 <div class="scenario-card">
 <h4>⍰ Commuter Guidance</h4>
 <p><span class="status-indicator status-high"></span><strong>Status:</strong> High
Demand Period</p>
 <p>Alternative routes recommended for downtown area. Average time savings: 12
minutes per trip</p>
 <p><strong>Suggestion:</strong> Use suburban bypass until 6:00 PM</p>
 </div>
 </div>
 </div>
 <div class="tooltip" id="tooltip"></div>
 <script>
 // ML Model Simulation Class
 class TrafficPredictor {
 constructor() {
 this.weights = {
 hour: 0.4,
 day: 0.2,

 weather: 0.15,

 temperature: 0.1,

 event: 0.15

 };

 this.baseTraffic = 800;

 }

 predict(hour, day, weather, temperature, event) {

 let volume = this.baseTraffic;

 // Hour patterns (rush hours)

 if (hour >= 7 && hour <= 9) volume += 600; // Morning rush

 else if (hour >= 17 && hour <= 19) volume += 700; // Evening rush

 else if (hour >= 12 && hour <= 14) volume += 300; // Lunch hour

 else if (hour >= 22 || hour <= 5) volume -= 400; // Night

 // Day of week patterns

 if (day >= 1 && day <= 5) volume += 200; // Weekdays

 else volume -= 100; // Weekends

 // Weather impact

 switch(weather) {

 case 'rain': volume += 150; break;

 case 'snow': volume += 300; break;

 case 'fog': volume += 100; break;

 default: break;

 }

 // Temperature impact
```

```
if (temperature < 0) volume += 100;

else if (temperature > 35) volume += 50;

// Event impact

switch(event) {

case 'concert': volume += 400; break;

case 'sports': volume += 600; break;

case 'festival': volume += 300; break;

case 'holiday': volume -= 200; break;

default: break;

}

// Add some randomness

volume += Math.random() * 200 - 100;

return Math.max(0, Math.round(volume));

}

getCongestionLevel(volume) {

if (volume < 900) return 'Low';

else if (volume < 1400) return 'Medium';

else return 'High';

}

getSensorClass(volume) {

if (volume < 900) return 'sensor-low';

else if (volume < 1400) return 'sensor-medium';

else return 'sensor-high';

}

}
```

```
const predictor = new TrafficPredictor();
let chart;
let userLocation = null;
let userLocationMarker = null;
// Location Analysis System
class LocationAnalyzer {
constructor() {
this.zones = {
downtown: { center: { lat: 17.385044, lng: 78.486671 }, radius: 2 },
highway: { center: { lat: 17.440000, lng: 78.520000 }, radius: 3 },
suburban: { center: { lat: 17.360000, lng: 78.450000 }, radius: 4 },
industrial: { center: { lat: 17.420000, lng: 78.420000 }, radius: 3 }
};
}
findNearestZone(userLat, userLng) {
let nearestZone = null;
let minDistance = Infinity;
for (const [zoneName, zone] of Object.entries(this.zones)) {
const distance = this.calculateDistance(
userLat, userLng,
zone.center.lat, zone.center.lng
);

if (distance < minDistance) {
minDistance = distance;
nearestZone = {
name: zoneName,
distance: distance,
...zone
};
}
}
return nearestZone;
}
calculateDistance(lat1, lng1, lat2, lng2) {
const R = 6371; // Earth's radius in km
const dLat = (lat2 - lat1) * Math.PI / 180;
const dLng = (lng2 - lng1) * Math.PI / 180;
const a = Math.sin(dLat/2) * Math.sin(dLat/2) +
Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *
Math.sin(dLng/2) * Math.sin(dLng/2);
const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
return R * c;
}
analyzeTrafficForLocation(zone, hour = new Date().getHours()) {
const baseVolume = predictor.predict(hour, new Date().getDay(), 'clear', 25, 'none');
let zoneMultiplier = 1;
let congestionFactor = 1;
switch(zone.name) {
case 'downtown':
```

```
zoneMultiplier = 1.5;
congestionFactor = 1.3;
break;
case 'highway':
zoneMultiplier = 1.8;
congestionFactor = 1.4;
break;
case 'suburban':
zoneMultiplier = 0.7;
congestionFactor = 0.8;
break;
case 'industrial':
zoneMultiplier = 0.9;
congestionFactor = 1.0;
break;
}
const volume = Math.round(baseVolume * zoneMultiplier);
const waitTime = Math.round((volume / 100) * congestionFactor);
const alternativeRoutes = this.getAlternativeRoutes(zone.name);
return {
volume,
waitTime,
alternativeRoutes,
trend: this.getTrafficTrend(hour),
recommendations: this.getLocationRecommendations(zone.name, volume, hour)
```

```
};

}

getTrafficTrend(hour) {

if (hour >= 7 && hour <= 9) return '� Increasing';

if (hour >= 17 && hour <= 19) return '� Peak';

if (hour >= 10 && hour <= 16) return '� Stable';

if (hour >= 20 && hour <= 23) return '� Decreasing';

return '� Low';

}

getAlternativeRoutes(zone) {

const routes = {

downtown: 3,

highway: 2,

suburban: 4,

industrial: 2

};

return routes[zone] || 2;

}

getLocationRecommendations(zone, volume, hour) {

const recommendations = [];


if (volume > 1400) {

recommendations.push("Consider using alternative routes - high congestion

detected");

recommendations.push("Allow extra 15-20 minutes for your journey");
```

```
}


if (hour >= 7 && hour <= 9) {
recommendations.push("Peak morning hours - traffic will increase");
} else if (hour >= 17 && hour <= 19) {
recommendations.push("Evening rush hour - expect delays");
}
switch(zone) {
case 'downtown':
recommendations.push("Use public transport if available");
recommendations.push("Consider walking for short distances");
break;
case 'highway':
recommendations.push("Maintain safe following distance");
recommendations.push("Check for construction updates");
break;
case 'suburban':
recommendations.push("Local roads may be faster");
break;
case 'industrial':
recommendations.push("Watch for heavy vehicle traffic");
break;
}
return recommendations;
}
```

```
}

const locationAnalyzer = new LocationAnalyzer();

function initChart() {

const ctx = document.getElementById('trafficChart').getContext('2d');

const hours = [];

const volumes = [];


// Generate 24 hours of sample data

for (let i = 0; i < 24; i++) {

hours.push(i + ':00');

volumes.push(predictor.predict(i, 1, 'clear', 22, 'none'));

}

chart = new Chart(ctx, {

type: 'line',

data: {

labels: hours,

datasets: [{

label: 'Traffic Volume',

data: volumes,

borderColor: '#667eea',

backgroundColor: 'rgba(102, 126, 234, 0.1)',

borderWidth: 3,

fill: true,

tension: 0.4

}]
});

},
options: {
responsive: true,
plugins: {
legend: {
display: false
}
},
scales: {
y: {
beginAtZero: true,
title: {
display: true,
text: 'Vehicles per Hour'
}
},
x: {
title: {
display: true,
text: 'Hour of Day'
}
}
}
}
});
```

```
}
function initTrafficMap() {
const sensors = document.querySelectorAll('.traffic-sensor');
const tooltip = document.getElementById('tooltip');
sensors.forEach(sensor => {
sensor.addEventListener('mouseenter', (e) => {
const location = e.target.dataset.location;
const volume = e.target.dataset.volume;
const congestion = predictor.getCongestionLevel(parseInt(volume));

tooltip.innerHTML = `
<strong>${location}</strong><br>
Volume: ${parseInt(volume).toLocaleString()} vehicles/hr<br>
Status: ${congestion} Traffic
`;
tooltip.style.opacity = '1';
});
sensor.addEventListener('mousemove', (e) => {
tooltip.style.left = e.pageX + 10 + 'px';
tooltip.style.top = e.pageY - 30 + 'px';
});
sensor.addEventListener('mouseleave', () => {
tooltip.style.opacity = '0';
});
});
```

```
// Add traffic flow animation
createTrafficFlow();
}
function createTrafficFlow() {
const map = document.getElementById('trafficMap');

setInterval(() => {
// Create horizontal flows
for (let i = 0; i < 3; i++) {
const flow = document.createElement('div');
flow.className = 'traffic-flow';
flow.style.left = '0%';
flow.style.top = ${20 + i * 30}%;
flow.style.transform = 'translateY(-50%)';

map.appendChild(flow);

// Animate horizontally
flow.animate([
{ left: '0%' },
{ left: '100%' }
], {
duration: 3000,
easing: 'linear'
}).onfinish = () => flow.remove();
}

// Create vertical flows
for (let i = 0; i < 3; i++) {
const flow = document.createElement('div');
flow.className = 'traffic-flow';
flow.style.left = ${25 + i * 25}%;
flow.style.top = '0%';
flow.style.transform = 'translateX(-50%)';

map.appendChild(flow);

// Animate vertically
flow.animate([
{ top: '0%' },
{ top: '100%' }
], {
duration: 4000,
easing: 'linear'
}).onfinish = () => flow.remove();
}
}, 1000);
}
function updateTrafficMap() {
const sensors = document.querySelectorAll('.traffic-sensor');
const hour = parseInt(document.getElementById('hour').value);
const day = parseInt(document.getElementById('day').value);
const weather = document.getElementById('weather').value;
const temperature = parseInt(document.getElementById('temperature').value);
const event = document.getElementById('event').value;
sensors.forEach(sensor => {
// Generate new volume based on location and conditions
let baseVolume = parseInt(sensor.dataset.volume);
let newVolume = predictor.predict(hour, day, weather, temperature, event);

// Adjust based on location type
const location = sensor.dataset.location.toLowerCase();
if (location.includes('downtown')) {
newVolume *= 1.2;
} else if (location.includes('highway')) {
newVolume *= 1.5;
} else if (location.includes('suburban')) {
newVolume *= 0.8;
} else if (location.includes('industrial')) {
newVolume *= 0.9;
}
newVolume = Math.round(newVolume);
sensor.dataset.volume = newVolume;

// Update sensor class
sensor.className = traffic-sensor ${predictor.getSensorClass(newVolume)};
});
}
function predictTraffic() {
const hour = parseInt(document.getElementById('hour').value);
const day = parseInt(document.getElementById('day').value);
const weather = document.getElementById('weather').value;
const temperature = parseInt(document.getElementById('temperature').value);
const event = document.getElementById('event').value;
const prediction = predictor.predict(hour, day, weather, temperature, event);
const congestion = predictor.getCongestionLevel(prediction);
// Update display
document.getElementById('predictedVolume').textContent = prediction.toLocaleString();
document.getElementById('congestionLevel').textContent = congestion;

// Update current volume with some variation
const currentVolume = Math.round(prediction * (0.85 + Math.random() * 0.3));
document.getElementById('currentVolume').textContent =
currentVolume.toLocaleString();
// Update accuracy
const accuracy = (94 + Math.random() * 4).toFixed(1);
document.getElementById('accuracy').textContent = accuracy + '%';
// Update chart with new prediction
updateChart(hour, day, weather, temperature, event);
```

```
// Update traffic map
updateTrafficMap();
}
function updateChart(currentHour, day, weather, temperature, event) {
const volumes = [];
for (let i = 0; i < 24; i++) {
volumes.push(predictor.predict(i, day, weather, temperature, event));
}

chart.data.datasets[0].data = volumes;

// Highlight current hour
chart.data.datasets[0].pointBackgroundColor = volumes.map((_, i) =>
i === currentHour ? '#f56565' : '#667eea'
);
chart.data.datasets[0].pointRadius = volumes.map((_, i) =>
i === currentHour ? 8 : 4
);

chart.update();
}
function updateDashboard() {
const location = document.getElementById('location').value;
const timeRange = parseInt(document.getElementById('timeRange').value);
// Simulate different data based on location
```

```
 let baseMultiplier = 1;
 switch(location) {
 case 'downtown': baseMultiplier = 1.5; break;
 case 'highway': baseMultiplier = 1.8; break;
 case 'suburban': baseMultiplier = 0.7; break;
 case 'industrial': baseMultiplier = 0.9; break;
 }
 const currentVolume = Math.round(1200 * baseMultiplier);
 const predictedVolume = Math.round(currentVolume * (1 + (Math.random() * 0.4 - 0.2)));

 document.getElementById('currentVolume').textContent =
currentVolume.toLocaleString();
 document.getElementById('predictedVolume').textContent =
predictedVolume.toLocaleString();
 document.getElementById('congestionLevel').textContent =
predictor.getCongestionLevel(predictedVolume);

 // Update traffic map based on selected location
 highlightMapLocation(location);
 }
 function highlightMapLocation(location) {
 const sensors = document.querySelectorAll('.traffic-sensor');

 sensors.forEach(sensor => {
 const sensorLocation = sensor.dataset.location.toLowerCase();
```

```
// Remove any existing highlights
sensor.style.border = 'none';

// Highlight sensors matching the selected location
if ((location === 'downtown' && sensorLocation.includes('downtown')) ||
(location === 'highway' && sensorLocation.includes('highway')) ||
(location === 'suburban' && sensorLocation.includes('suburban')) ||
(location === 'industrial' && sensorLocation.includes('industrial'))) {

sensor.style.border = '3px solid #fff';
sensor.style.boxShadow = '0 0 15px rgba(255, 255, 255, 0.8)';
}
});
}
function getCurrentLocation() {
const btn = document.getElementById('locationBtn');
btn.textContent = '⍰ Getting Location...';
btn.disabled = true;
if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(
(position) => {
userLocation = {
lat: position.coords.latitude,
lng: position.coords.longitude
};

updateLocationInfo();
showLocationOnMap();
analyzeUserLocation();

btn.textContent = '⍰ Location Updated';
btn.disabled = false;

// Re-enable button after 3 seconds
setTimeout(() => {
btn.textContent = '⍰ Get My Location';
}, 3000);
},
(error) => {
// Fallback to Hyderabad coordinates for demo
userLocation = {
lat: 17.385044,
lng: 78.486671
};

updateLocationInfo();
showLocationOnMap();
analyzeUserLocation();
```

```
btn.textContent = '⍰ Demo Location (Hyderabad)';
btn.disabled = false;

alert('Location access denied. Using demo location (Hyderabad, India) for
demonstration.');
}
);
} else {
alert('Geolocation is not supported by this browser.');
btn.textContent = '⍰ Get My Location';
btn.disabled = false;
}
}
function updateLocationInfo() {
const locationInfo = document.getElementById('locationInfo');
const coordinates = document.getElementById('coordinates');

coordinates.value = ${userLocation.lat.toFixed(6)}, ${userLocation.lng.toFixed(6)};
locationInfo.style.display = 'block';
}
function showLocationOnMap() {
const map = document.getElementById('trafficMap');

// Remove existing user location marker
if (userLocationMarker) {
```

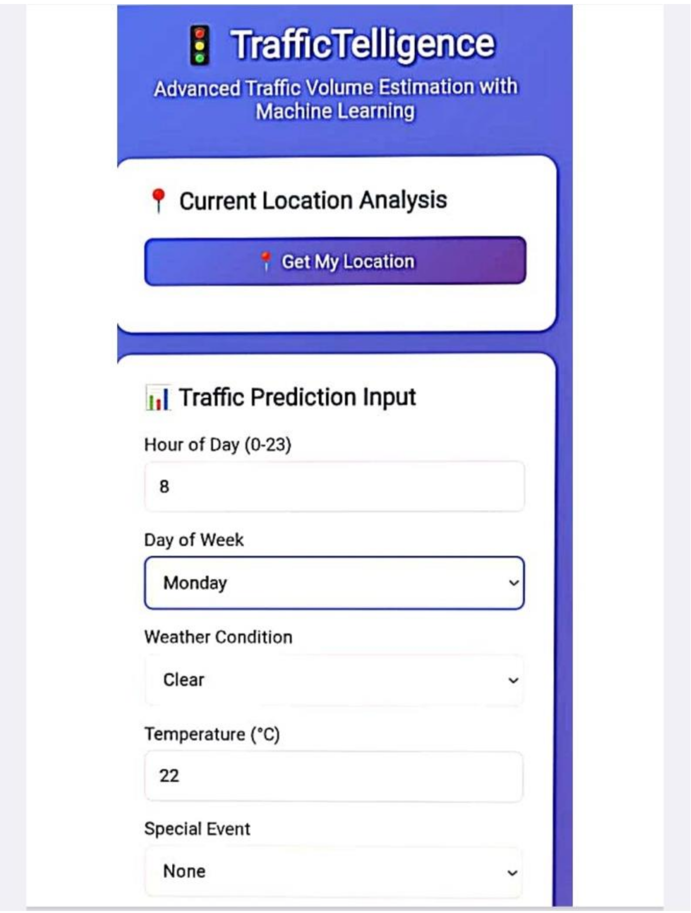
```
userLocationMarker.remove();

}


// Create new user location marker

userLocationMarker = document.createElement('div');

userLocationMarker.className = 'user-location-marker';


// Position marker (simplified positioning for demo)

const mapX = 30 + (Math.random() * 40); // Random position for demo

const mapY = 25 + (Math.random() * 50);


userLocationMarker.style.left = ${mapX}%;

userLocationMarker.style.top = ${mapY}%;


map.appendChild(userLocationMarker);


// Add tooltip for user location

userLocationMarker.addEventListener('mouseenter', (e) => {

const tooltip = document.getElementById('tooltip');

tooltip.innerHTML = `

<strong>□ Your Location</strong><br>

Lat: ${userLocation.lat.toFixed(4)}<br>

Lng: ${userLocation.lng.toFixed(4)}<br>

Click for detailed analysis

`;
```

```
 tooltip.style.opacity = '1';
 });

 userLocationMarker.addEventListener('mousemove', (e) => {
 const tooltip = document.getElementById('tooltip');
 tooltip.style.left = e.pageX + 10 + 'px';
 tooltip.style.top = e.pageY - 30 + 'px';
 });

 userLocationMarker.addEventListener('mouseleave', () => {
 const tooltip = document.getElementById('tooltip');
 tooltip.style.opacity = '0';
 });
 }
 function analyzeUserLocation() {
 const nearestZone = locationAnalyzer.findNearestZone(userLocation.lat,
userLocation.lng);
 const analysis = locationAnalyzer.analyzeTrafficForLocation(nearestZone);

 // Update UI with analysis results
 document.getElementById('nearestZone').value =
${nearestZone.name.charAt(0).toUpperCase() + nearestZone.name.slice(1)}
(${nearestZone.distance.toFixed(1)}km away);
 document.getElementById('userLocationVolume').value = ${analysis.volume}
vehicles/hour;
```

```
 document.getElementById('userLocationStatus').value =
predictor.getCongestionLevel(analysis.volume);

 // Show detailed analysis
 document.getElementById('locationAnalysis').style.display = 'block';
 document.getElementById('userTrafficVolume').textContent =
analysis.volume.toLocaleString();
 document.getElementById('userTrafficTrend').textContent = analysis.trend;
 document.getElementById('userWaitTime').textContent = ${analysis.waitTime} min;
 document.getElementById('userAlternativeRoutes').textContent =
analysis.alternativeRoutes;

 // Update recommendations
 const recommendationsList = document.getElementById('recommendationsList');
 recommendationsList.innerHTML = '';
 analysis.recommendations.forEach(rec => {
 const li = document.createElement('li');
 li.textContent = rec;
 recommendationsList.appendChild(li);
 });
 }
 const recommendations = [
 "Increase signal cycle time by 15% during peak hours",
 "Activate dynamic lane control on Highway Junction",
 "Deploy mobile traffic units to downtown core",
 "Update navigation apps with alternate route suggestions",
 "Implement congestion pricing during rush hours",
 "Activate express bus services on high-demand routes",
 "Redirect traffic through suburban bypass routes",
 "Optimize traffic light coordination along main corridors",
 "Deploy additional traffic enforcement in high-congestion areas"
 ];
 const randomRec = recommendations[Math.floor(Math.random() *
recommendations.length)];
 alert(□ AI Recommendation:\n\n${randomRec}\n\nExpected Impact: 12-18% congestion
reduction);

 // Initialize the application
 document.addEventListener('DOMContentLoaded', function() {
 initChart();
 initTrafficMap();
 predictTraffic(); // Initial prediction

 // Auto-update every 30 seconds to simulate real-time data
 setInterval(() => {
 updateDashboard();
 updateTrafficMap();
 }, 30000);
 });
 </script>
```

```
</body>

</html>
```

OUTPUT:-



🚦 TrafficTelligence

Advanced Traffic Volume Estimation with Machine Learning

📍 Current Location Analysis

📊 Traffic Prediction Input

Hour of Day (0-23)

8

Day of Week

Monday

Weather Condition

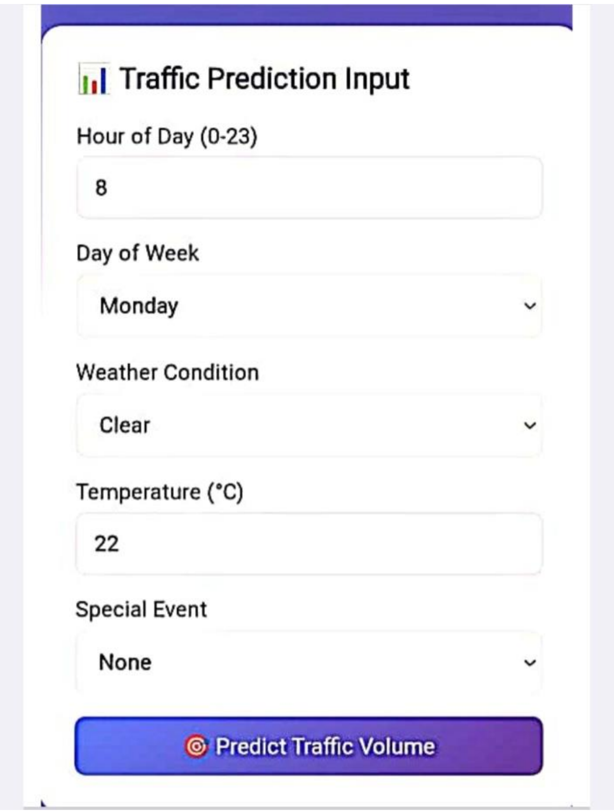
Clear

Temperature (°C)

22

Special Event

None



📊 Traffic Prediction Input

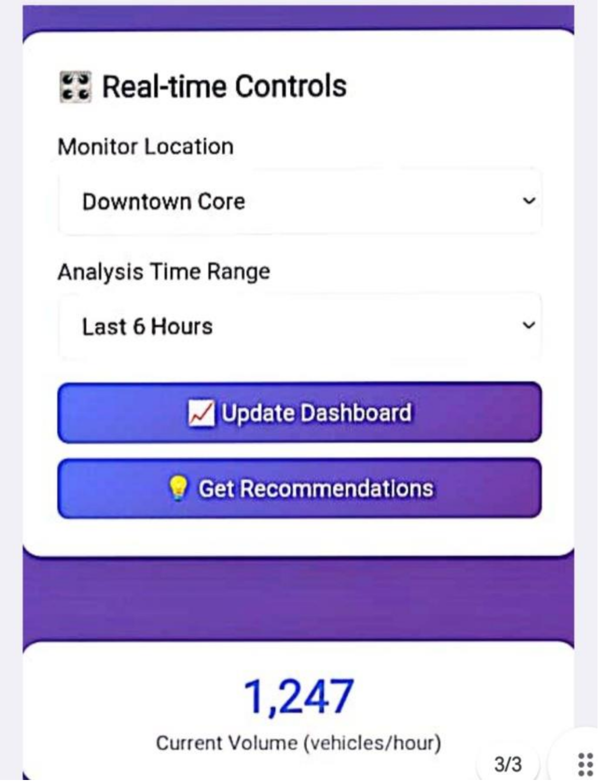
Hour of Day (0-23): 8

Day of Week: Monday

Weather Condition: Clear

Temperature (°C): 22

Special Event: None



🎛️ Real-time Controls

Monitor Location: Downtown Core

Analysis Time Range: Last 6 Hours

1,247

Current Volume (vehicles/hour)