Traffic management system

AGENDA:

To develop the web application that displays our output data to the end user.

TEAM MEMBERS:

* PRABU T (MENTOR)
* REVU BHARATH KUMAR (LEADER)
* THIYAGARAJAN D (MEMBER)
* SUDHAKAR V (MEMBER)
* MEESALA PRAVEEN (MEMBER)
* MUSALI DIVAKAR (MEMBER)

**OBJECTIVES:**

* **Reducing Congestion:** One of the primary objectives is to reduce traffic congestion. Traffic management systems aim to alleviate the gridlock that leads to delays, frustration, and increased pollution.
* **Enhancing Safety:**These systems help improve road safety by monitoring and responding to accidents, hazards, and other issues in real-time. They can also control traffic signals and signage to promote safer driving.
* **Efficient Resource Allocation:**By collecting and analyzing traffic data, these systems can allocate resources more efficiently, such as adjusting traffic signal timings to match traffic patterns or deploying emergency services to accident sites more quickly.
* **Environmental Benefits:**Traffic management systems can help reduce emissions and fuel consumption by optimizing traffic flow, reducing idling, and minimizing stop-and-go patterns.



**PLATFORM:**

* **Operating System Platforms:**Examples include Windows, macOS, and Linux, which provide the foundation for running various applications.
* **Cloud Computing Platforms:**Providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud offer cloud platforms that allow businesses to build, deploy, and scale applications in the cloud.
* **Mobile App Development Platforms:** iOS and Android are platforms for developing mobile applications, each with its own set of development tools and environments.
* **Web Development Platforms:**Web development platforms like WordPress, Drupal, and Ruby on Rails provide tools and frameworks for building websites and web applications.
* **E-commerce Platforms:** Platforms like Shopify and Magento are designed specifically for creating and managing online stores.
* **IoT (Internet of Things) Platforms:**These platforms help developers build and manage IoT applications and devices.

# **WEB DEVELOPMENT PLATFORM:**

* **Web Development Frameworks:** Use frameworks like Django, Ruby on Rails, or Express.js to expedite the development process.
* **Content Management Systems (CMS):** If your website will involve content management, consider using platforms like WordPress, Joomla, or Drupal.
* **Database Management Systems:**Tools like MySQL, PostgreSQL, and MongoDB are commonly used for database management.
* **Version Control Systems:**Git is essential for tracking changes in the codebase and collaborating with a team.
* **Development Environments:** IDEs (Integrated Development Environments) like Visual Studio Code, PyCharm, and Sublime Text are popular for web development.

**CODE FOR WEBSITE :**

1. **Server-Side Python Code:**

**from flask import Flask, render\_template, request**

**app = Flask(\_\_name\_\_)**

**# Store received sensor data in memory for demonstration.**

**sensor\_data = []**

**@app.route('/')**

**def index():**

**return render\_template('index.html', sensor\_data=sensor\_data)**

**@app.route('/receive\_data', methods=['POST'])**

**def receive\_data():**

**data = request.get\_json()**

**sensor\_data.append(data)**

**return 'Data received and stored'**

**if \_\_name\_\_ == '\_\_main\_\_':**

**app.run(debug=True)**

1. **HTML and Template for Web Page:**

**<!-- templates/index.html -->**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Traffic Congestion Monitor</title>**

**</head>**

**<body>**

**<h1>Traffic Congestion Monitor</h1>**

**<div id="congestion-display">**

**<table>**

**<tr>**

**<th>Timestamp</th>**

**<th>Congestion Level</th>**

**</tr>**

**{% for data in sensor\_data %}**

**<tr>**

**<td>{{ data.timestamp }}</td>**

**<td>{{ data.congestion\_level }}</td>**

**</tr>**

**{% endfor %}**

**</table>**

**</div>**

**</body>**

**</html>**

1. **Python Code for Bluetooth (Bluetooth LE):**

**import pygatt**

**import json**

**def send\_data\_via\_bluetooth(data):**

**device\_address = 'your\_device\_address'**

**client = pygatt.GATTToolBackend()**

**client.start()**

**try:**

**device = client.connect(device\_address)**

**device.char\_write('your\_characteristic\_uuid', json.dumps(data).encode('utf-8'))**

**finally:**

**client.stop()**

**# Usage:**

**data = {"timestamp": "2023-10-25 10:00:00", "congestion\_level": 30}**

**send\_data\_via\_bluetooth(data)**

1. **Python Code for Wi-Fi (HTTP Post Request):**

**import requests**

**import json**

**def send\_data\_via\_wifi(data):**

**server\_url = 'http://your-server-ip-or-domain/receive\_data'**

**headers = {'Content-Type': 'application/json'}**

**response = requests.post(server\_url, data=json.dumps(data), headers=headers)**

**# Usage:**

**data = {"timestamp": "2023-10-25 10:00:00", "congestion\_level": 30}**

**send\_data\_via\_wifi(data)**

THANK YOU