A Project Report

on

SMART CITY PLANNER

Submitted in partial fulfillment of requirements for the award of the course

of

**ADI1201 – DATA EXPLORATION AND VISUALIZATION**

Under the guidance of

### Ms. A Jeyashri

### Assistant Professor / AI

Submitted By



**Sukant R (927623BAD114)**

**Perumal P (927623BAD074)**

**Sakthi Sanjuktha S (927623BAD096)**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**M.KUMARASAMY COLLEGE OF ENGINEERING**

(Autonomous)

**KARUR – 639 113**

DECEMBER 2024

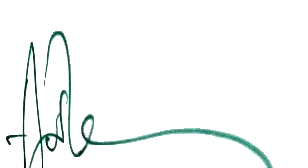
# M. KUMARASAMY COLLEGE OF ENGINEERING

**(Autonomous Institution affiliated to Anna University, Chennai)**

# KARUR – 639 113

**BONAFIDE CERTIFICATE**

Certified that this project report on **“ SMART CITY PLANNER”** is the Bonafide work of **SUKANT R (927623BAD114), PERUMAL P (927623BAD074), SAKTHI SANJUKTHA S (927623BAD096)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



|  |  |
| --- | --- |
| Signature | Signature |
| **Mrs. A.JEYASHRI** | **Dr.A.SELVI Ph.D.,** |
| **SUPERVISOR,** | **HEAD OF THE DEPARTMENT,** |
| Department of Artificial Intelligence  and Data Science, | Department of Artificial Intelligence  and Data Science, |
| M.Kumarasamy College of Engineering, | M.Kumarasamy College of Engineering, |
| Thalavapalayam, Karur -639 113. | Thalavapalayam, Karur -639 113. |



**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**VISION OF THE INSTITUTION**

             To emerge as a leader among the top institutions in the field of technical education

**MISSION OF THE INSTITUTION**

* Produce smart technocrats with empirical knowledge who can surmount the global challenges
* Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
* Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

**VISION OF THE DEPARTMENT**

To produce competent industry relevant education, skillful research, technical and innovative computer science professionals acquaintance with managerial skills, human and social values.

**MISSION OF THE DEPARTMENT**

* To impart technical knowledge through innovative teaching, research, and consultancy.
* To develop and to promote student ability thereby to compete globally through excellence in education.
* To facilitate the development of academic-industry Collaboration.
* To produce competent engineers with professional ethics, technical competence and a spirit of innovation and managerial skills.

**PROGRAM EDUCATIONAL OBJECTIVES (PEOS)**

**PEO 1:** To acquire technical knowledge and proficiency required for the employment and higher education in the contemporary areas of computer science or management studies**.**

**PEO 2:** To apply their competency in design and development of innovative solutions for real-world problems.

**PEO 3:** To demonstrate leadership qualities with high ethical standards and collaborated with other industries for the socio-economical growth of the country.

**PROGRAM OUTCOMES**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**PROGRAM SPECIFIC OUTCOMES (PSOs)**

* **PSO1: Professional Skills:**  Ability to apply the knowledge of computing techniques to design and develop computerized solutions for the problems.
* **PSO2: Successful career:** Ability to utilize the computing skills and ethical values in creating a successful career.

# 

# ABSTRACT

# The "Smart City Planner: Analyzing Urban Data for Better Living" project uses IoT technology and real-time data analytics to address key urban challenges in traffic management, energy usage, and environmental monitoring. The project aims to develop an integrated platform, "Info-Share," which collects data from various city-wide IoT sensors to optimize urban operations. In traffic management, real-time data will be used to adjust traffic signals dynamically, reducing congestion and improving traffic flow. Energy usage will be monitored to identify inefficiencies and promote energy-saving strategies, reducing consumption and costs. The environmental monitoring system will track air and water quality, noise levels, and temperature, issuing alerts when thresholds are exceeded, ensuring a healthier urban environment. The platform will facilitate data sharing among city departments, businesses, and citizens, fostering collaboration and informed decision-making. Ultimately, the project aims to create a sustainable, efficient, and livable city environment through the use of real-time data and smart technologies.

# ABSTRACT WITH POs AND PSOs MAPPING

|  |  |  |
| --- | --- | --- |
| **ABSTRACT** | **POs MAPPED** | **PSOs MAPPED** |
| The "Smart City Planner" project uses IoT sensors and real-time data analytics to address urban challenges in traffic management, energy usage, and environmental monitoring. By developing the "Info-Share" platform, the project optimizes traffic flow through dynamic signal adjustments, monitors energy consumption to reduce inefficiencies, and tracks air, water, and noise quality to ensure a healthier environment. The platform facilitates data exchange among city departments, businesses, and citizens to support informed decision-making, fostering a sustainable, efficient, and livable urban environment. | **POs 1**(3)  **POs 2**(1)  **POs 3**(3)  **POs 4**(2)  **POs 5**(2)  **POs 6**(3)  **POs 8**(1)  **POs 9**(2)  **POs 11**(1)  **POs 12**(3) | **PSO 1**(3)  **PSO 2**(2) |

Note: 1- Low, 2-Medium, 3- High

**SUPERVISOR HEAD OF THE DEPARTMENT**

# 

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** No. | TITLE | **PAGE** No. |
| 1 | ABSTRACT | vii |
| 2 | INTRODUCTION | 1 |
|  | 1.1 Objective | 1 |
|  | 1.2 Overview | 1 |
|  | 1.3 Technology Used | 2 |
| 3 | PROJECT METHODOLOGY | 4 |
|  | 2.1 Proposed Work | 4 |
|  | 2.2 Block Diagram | 4 |
| 4 | METHODOLOGY | 5 |
|  | 3.1 Data Collection | 5 |
|  | 3.2 Data Integration and Processing | 5 |
|  | 3.3 Data Analysis and Visualization | 5 |
|  | 3.4 Web Dashboard Development | 6 |
|  | 3.5 Data Storage | 6 |
| 5 | RESULTS AND DISCUSSION | 7 |
| 6 | CONCLUSION | 9 |
| 7 | REFERENCES | 10 |
| 8 | APPENDIX | 11 |

# CHAPTER 1

# INTRODUCTION

# Objective

The objective of the "Smart City Planner" project is to leverage IoT sensors and real-time data to optimize traffic management, reduce energy consumption, and monitor environmental conditions in urban areas. The project aims to dynamically adjust traffic signals, track and reduce city-wide energy inefficiencies, and monitor air, water, and noise quality to ensure a healthier environment. By developing the "Info-Share" platform, the project facilitates seamless data exchange among city departments, businesses, and citizens, promoting collaboration in decision-making. Ultimately, it seeks to create a sustainable, efficient, and livable urban environment through data-driven insights and real-time monitoring.

# Overview

The "Smart City Planner" project aims to enhance urban living through the integration of IoT sensors and real-time data analytics. It focuses on three key areas: traffic management, energy consumption, and environmental monitoring. The system collects data from IoT sensors to optimize traffic flow, reduce congestion, and dynamically adjust traffic signals. It also tracks city-wide energy usage to identify inefficiencies and promote energy-saving measures. Additionally, the project monitors air, water, and noise quality to ensure a healthier urban environment. The "Info-Share" platform facilitates seamless data exchange between city departments, businesses, and citizens, fostering collaboration and informed decision-making. The goal is to create a sustainable, efficient, and livable urban environment through proactive monitoring and intelligent resource management.

* 1. **Technology Used**

**1.Python for Data Processing and Analysis**

Python is a highly versatile and powerful language, especially for data analytics. With libraries like Pandas (for data manipulation), NumPy (for numerical computing), and SciPy (for scientific computing), Python is ideal for processing and analyzing large datasets from IoT sensors. Python also supports machine learning libraries like Scikit-learn and TensorFlow, should the project require predictive analytics in the future.

**2.Flask for Dashboard Interface**

Flask is a lightweight Python framework used to build web applications. In this project, Flask will be used to create the dashboard interface for displaying real-time data and analytics. It will allow users (city planners, residents, etc.) to interact with the data collected from IoT sensors, view visualizations, and receive alerts or notifications. Flask is chosen for its simplicity, flexibility, and ease of integration with other Python libraries.

**3.IoT Devices for Real-Time Data Collection**

IoT (Internet of Things) devices such as sensors will be used to collect real-time data on various parameters such as traffic flow, air quality, energy usage, and environmental conditions. These sensors will be deployed throughout the city to gather data continuously, ensuring that the system can monitor urban conditions in real time.

**4. Data Visualization Libraries (Matplotlib & Folium)**

Matplotlib is a widely used library for creating static, animated, and interactive visualizations in Python. It will be used to generate charts and graphs for visualizing traffic data, energy consumption trends, and environmental parameters.

Folium is a Python library specifically used for interactive mapping. It integrates well with geographic data, making it ideal for visualizing traffic patterns, environmental conditions, or energy usage on maps. Folium is particularly useful when dealing with location-based data and creating interactive maps for the dashboard.

This technology stack is designed to provide a robust, scalable, and flexible solution for real-time monitoring, data analysis, and decision-making, ensuring the project's success in optimizing urban life through smart technologies.

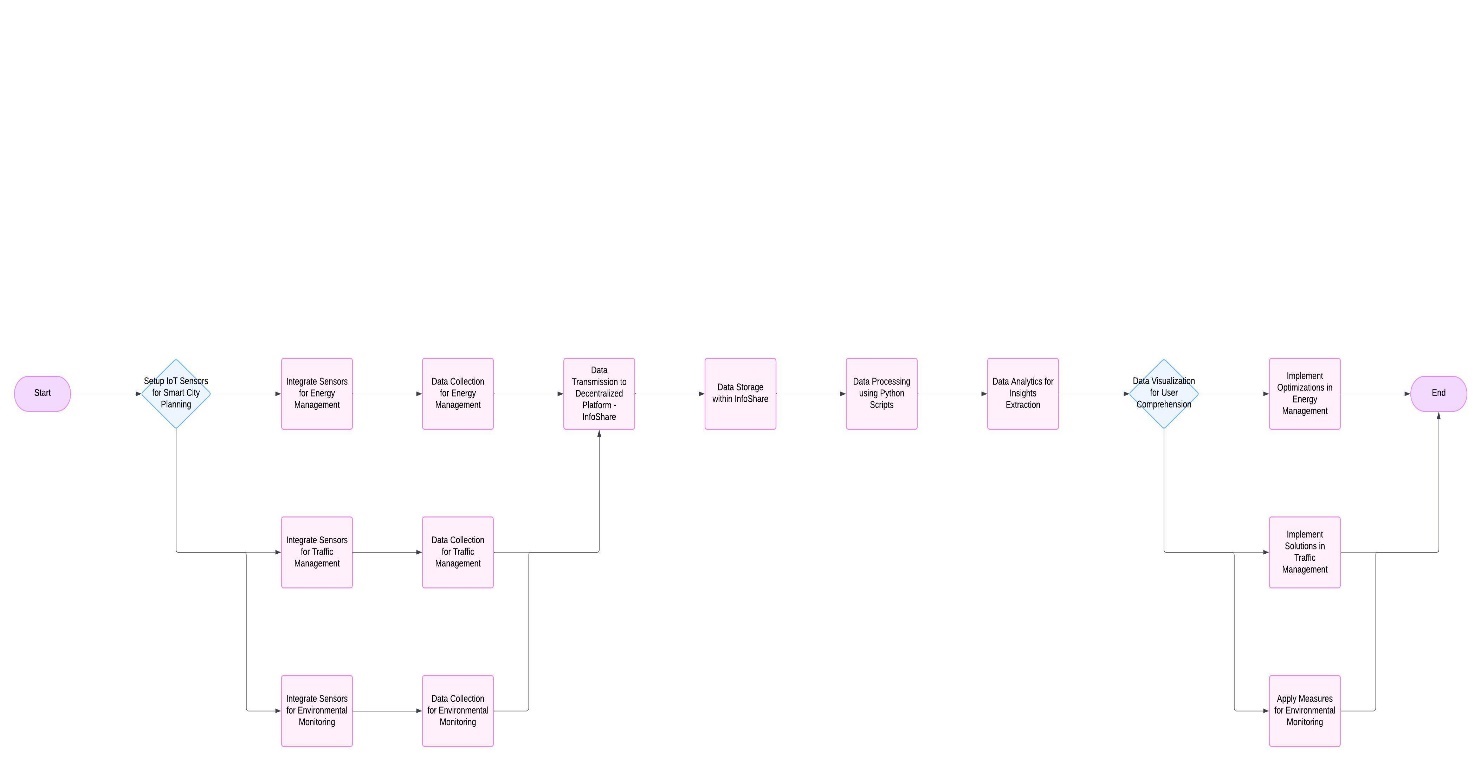
**CHAPTER 2**

**PROJECT METHODOLOGY**

**2.1Proposed Work**

The proposed work for the "Smart City Planner" project involves designing a system to collect and analyze real-time data from IoT sensors deployed across the city. The project will focus on optimizing traffic flow, monitoring energy usage, and tracking environmental conditions. Data will be processed and analyzed to generate insights, trigger alerts, and provide actionable recommendations. A web-based dashboard, developed using Flask, will display the data through visualizations like charts and maps. Cloud-based storage will be used for scalability and efficient data management, enabling real-time access and future growth of the system.

**2.2 Block Diagram**

****

**CHAPTER 3**

**METHODOLOGY DESCRIPTION**

## 3.1 Deployment of IoT Devices for Data Collection

## Appropriate sensors are chosen to monitor real-time data for the key areas—traffic flow (e.g., traffic cameras, vehicle count sensors), energy usage (e.g., smart meters), and environmental monitoring (e.g., air quality, temperature, noise level sensors).IoT sensors are deployed across the city to collect real-time data on traffic, energy consumption, and environmental parameters. These sensors continuously monitor the selected metrics and send data to the central system for processing.

## 3.2 Data Integration and Processing

## The raw data from multiple IoT sensors is collected and integrated into a central data processing platform. Raw data from IoT devices often contains noise or missing values. Data cleaning techniques are applied to ensure the data is accurate, consistent, and usable for analysis.Real-time data is processed using algorithms to analyze and interpret the collected information. For example, traffic data can be used to dynamically adjust traffic signals, while energy data can be analyzed to detect inefficiencies or promote energy-saving strategies

## 3.3 Data Analysis and Visualization

## The collected data is analyzed to provide actionable insights, like adjusting traffic signals to improve flow or identifying energy-saving opportunities. Data is visualized through charts, graphs, and maps to make the information accessible and easy to interpret for users.The processed data is visualized through interactive dashboards. Libraries like Matplotlib and Folium are used.

## 3.4 Web Dashboard Development

## The user interface for the system is developed using Flask, a lightweight web framework. It provides a web-based dashboard where users (city planners, businesses, and residents) can access real-time data and visualizations. The system allows different access levels (e.g., admin, planners, public) to ensure that sensitive data is only accessible by authorized personnel. The dashboard allows users to interact with the data, explore different aspects of urban management (e.g., traffic, energy usage, air quality), and receive notifications or alerts when issues arise.

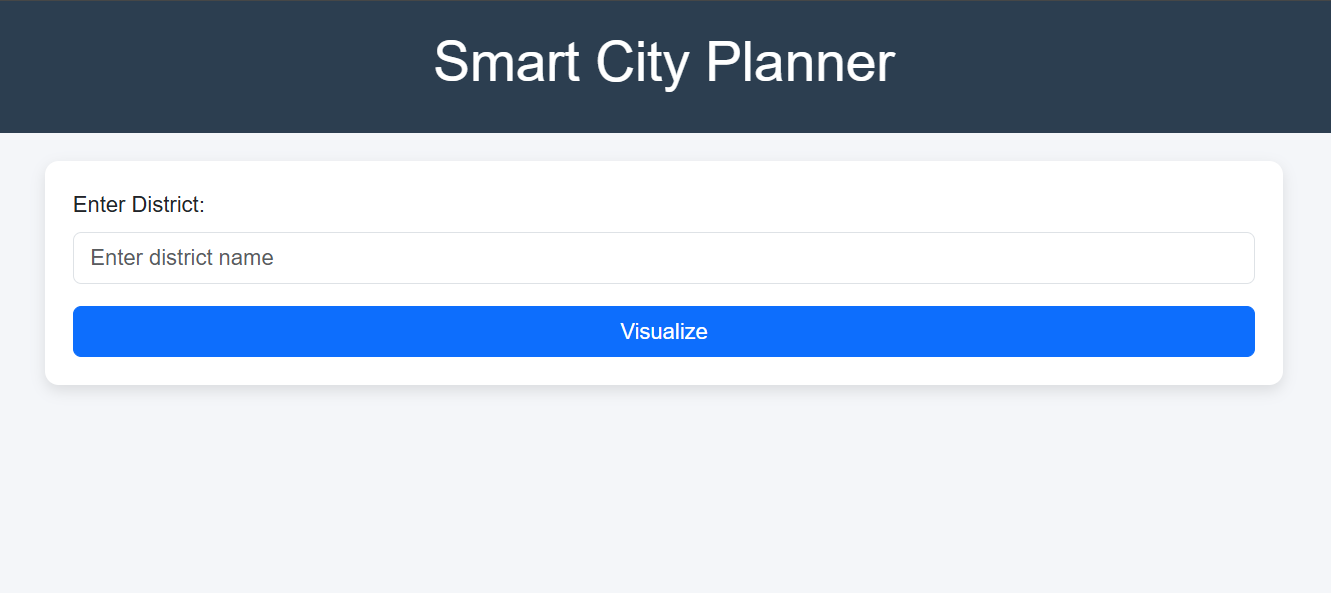
## 3.5 Data Storage

## Data collected from IoT devices is stored in the cloud, ensuring scalability and reliability. Cloud infrastructure allows for easy management of large data volumes and ensures the system can grow as more sensors are added over time.

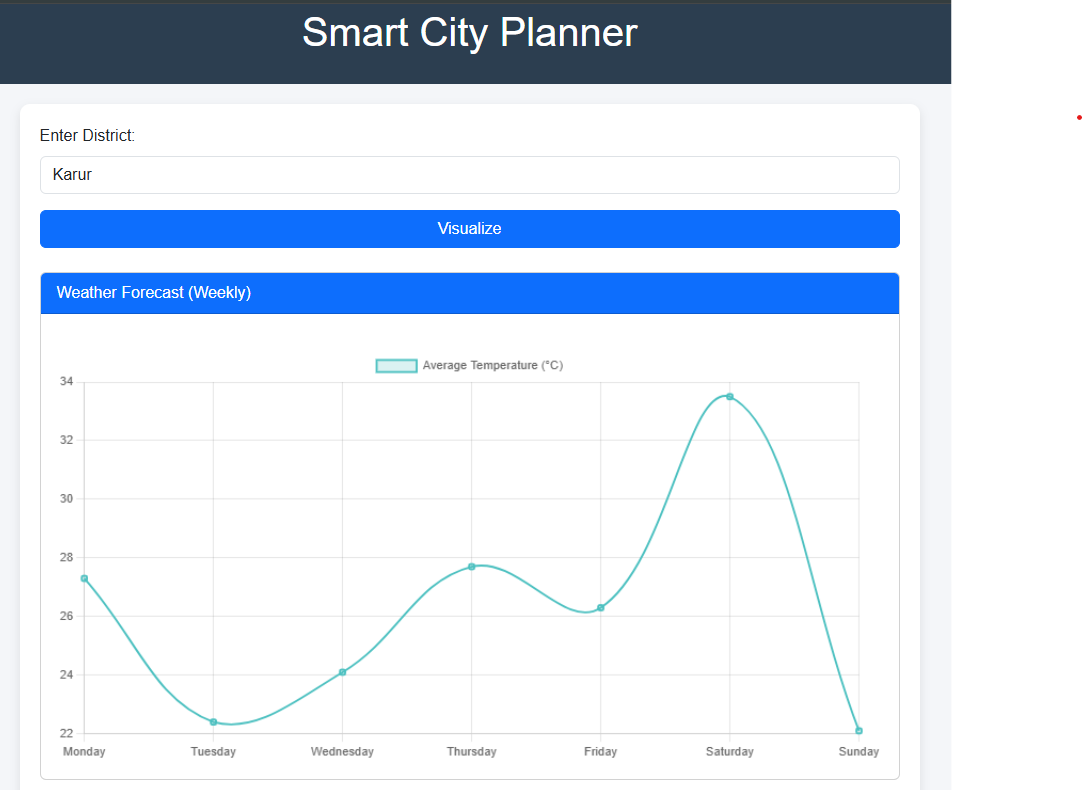
**CHAPTER 4**

**RESULTS AND DISCUSSION**

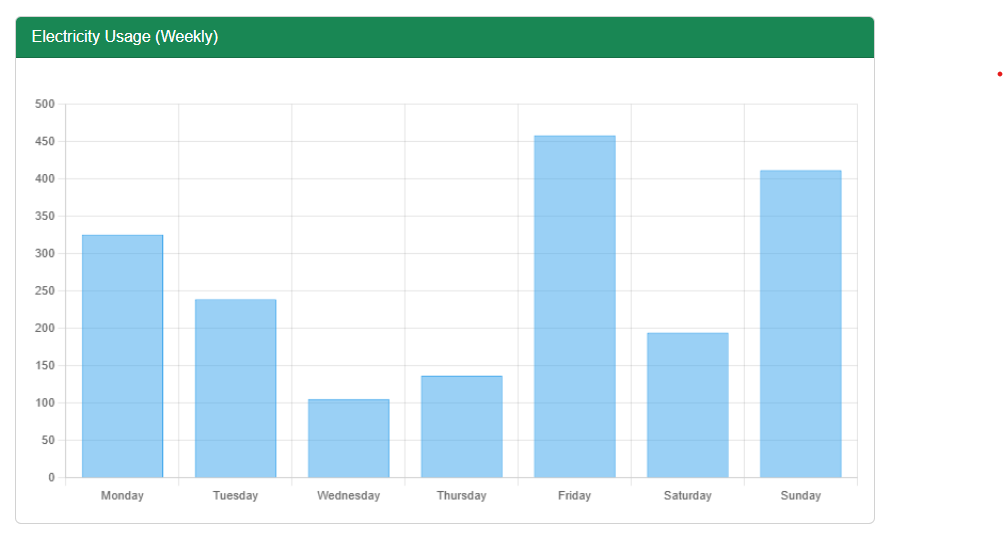
**Dashboard:**

****

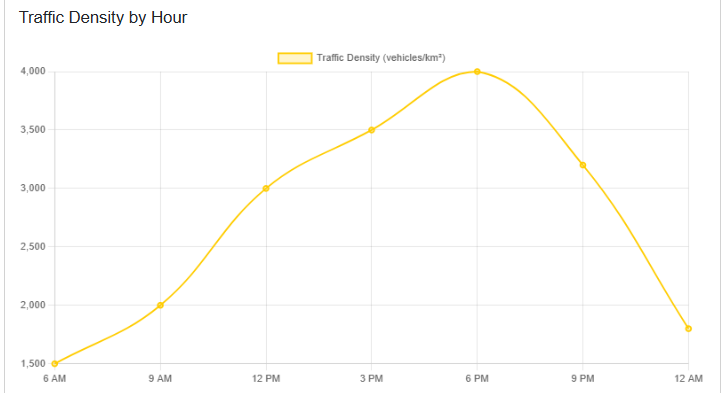
**Weather Report:**

****

**Power Report:**

****

**Traffic Report:**

****

**CHAPTER 5**

**CONCLUSION**

In the "Smart City Planner" project, users can enter the name of a district (e.g., "Chennai") and click the "Visualize" button to retrieve data specific to that district. The system then generates relevant visualizations, such as maps, charts, and insights, based on the data for that location. The "Smart City Planner" project's visualization page includes various sections to provide a comprehensive view of urban data. It features weather trends displayed as line charts, energy usage statistics in bar charts, traffic density analysis using pie and line charts, and a map that pinpoints the selected district, allowing users to explore different aspects of urban management visually.

The **Visualization Page** in the "Smart City Planner" project includes sections for displaying weather trends with line charts, energy usage statistics in bar charts, traffic density analysis using pie and line charts, and an interactive map pinpointing the selected district with relevant data overlays.

**REFERENCES:**

1. <https://leafletjs.com/reference.html>
2. <https://www.chartjs.org/docs/latest/>
3. <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
4. <https://www.iipa.org.in/cms/public/uploads/474091639042118.pdf>

**APPENDIX**

import pandas as pd

import random

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

import pandas as pd

import random

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

# Predefined list of districts with latitude and longitude

districts\_info = [

    {"Area": "Ariyalur", "Latitude": 11.1385, "Longitude": 79.0756},

    {"Area": "Chengalpattu", "Latitude": 12.697, "Longitude": 79.9828},

    {"Area": "Chennai", "Latitude": 13.0827, "Longitude": 80.2707},

    {"Area": "Coimbatore", "Latitude": 11.0168, "Longitude": 76.9558},

    {"Area": "Cuddalore", "Latitude": 11.748, "Longitude": 79.7714},

    {"Area": "Dharmapuri", "Latitude": 12.1357, "Longitude": 78.1612},

    {"Area": "Dindigul", "Latitude": 10.362, "Longitude": 77.9705},

    {"Area": "Erode", "Latitude": 11.341, "Longitude": 77.7172},

    {"Area": "Kallakurichi", "Latitude": 11.7386, "Longitude": 78.9604},

    {"Area": "Kancheepuram", "Latitude": 12.8342, "Longitude": 79.7036},

    {"Area": "Karur", "Latitude": 10.9601, "Longitude": 78.0766},

    {"Area": "Krishnagiri", "Latitude": 12.5186, "Longitude": 78.2133},

    {"Area": "Madurai", "Latitude": 9.9252, "Longitude": 78.1198},

    {"Area": "Mayiladuthurai", "Latitude": 11.1035, "Longitude": 79.655},

    {"Area": "Nagapattinam", "Latitude": 10.7656, "Longitude": 79.8425},

    {"Area": "Kanyakumari", "Latitude": 8.0883, "Longitude": 77.5385},

    {"Area": "Namakkal", "Latitude": 11.2186, "Longitude": 78.1676},

    {"Area": "Perambalur", "Latitude": 11.2333, "Longitude": 78.8833},

    {"Area": "Pudukottai", "Latitude": 10.3797, "Longitude": 78.82},

    {"Area": "Ramanathapuram", "Latitude": 9.3708, "Longitude": 78.8307},

    {"Area": "Ranipet", "Latitude": 12.9337, "Longitude": 79.3339},

    {"Area": "Salem", "Latitude": 11.6643, "Longitude": 78.146},

    {"Area": "Sivagangai", "Latitude": 9.8478, "Longitude": 78.488},

    {"Area": "Tenkasi", "Latitude": 8.9587, "Longitude": 77.3152},

    {"Area": "Thanjavur", "Latitude": 10.7867, "Longitude": 79.1391},

    {"Area": "Theni", "Latitude": 10.0104, "Longitude": 77.4777},

    {"Area": "Thiruvallur", "Latitude": 13.1431, "Longitude": 79.9085},

    {"Area": "Thiruvarur", "Latitude": 10.7672, "Longitude": 79.6366},

    {"Area": "Tuticorin", "Latitude": 8.7642, "Longitude": 78.1348},

    {"Area": "Tiruchirappalli", "Latitude": 10.7905, "Longitude": 78.7047},

    {"Area": "Thirunelveli", "Latitude": 8.7139, "Longitude": 77.7567},

    {"Area": "Tirupathur", "Latitude": 12.495, "Longitude": 78.5653},

    {"Area": "Tiruppur", "Latitude": 11.1085, "Longitude": 77.3411},

    {"Area": "Tiruvannamalai", "Latitude": 12.2253, "Longitude": 79.0747},

    {"Area": "The Nilgiris", "Latitude": 11.4143, "Longitude": 76.695},

    {"Area": "Vellore", "Latitude": 12.9165, "Longitude": 79.1325},

    {"Area": "Viluppuram", "Latitude": 11.9401, "Longitude": 79.4977},

    {"Area": "Virudhunagar", "Latitude": 9.5741, "Longitude": 77.9624},

]

# Generate synthetic dataset

data = []

for district in districts\_info:

    traffic\_density = random.randint(100, 5000)  # vehicles/km²

    traffic\_condition = random.choice(["Low", "Moderate", "High", "Congested"])

    electrical\_consumption = [round(random.uniform(50, 500), 2) for \_ in range(7)]  # Weekly kWh/day

    avg\_temperature = [round(random.uniform(20, 40), 1) for \_ in range(7)]  # Weekly °C

    weather\_condition = random.choice(["Sunny", "Rainy", "Cloudy", "Stormy", "Clear"])

    # Add data to the list

    data.append({

        "Area": district["Area"],

        "Latitude": district["Latitude"],

        "Longitude": district["Longitude"],

        "Traffic Density (vehicles/km²)": traffic\_density,

        "Traffic Condition": traffic\_condition,

        "Electrical Consumption (kWh/day)": electrical\_consumption,

        "Average Temperature (°C)": avg\_temperature,

        "Weather Condition": weather\_condition

    })

# Create a DataFrame

df = pd.DataFrame(data)

# Flask application remains the same as before.

# Flask Application

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

# Enhanced Template with Diverse Visualizations

template = """

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Smart City Planner</title>

    <!-- Bootstrap CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

    <!-- Chart.js -->

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <!-- jQuery -->

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <!-- Leaflet for Map -->

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

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

    <style>

        body {

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

            background-color: #f4f6f9;

        }

        header {

            text-align: center;

            background: #2c3e50;

            color: #fff;

            padding: 20px 0;

            margin-bottom: 20px;

        }

        main {

            max-width: 900px;

            margin: auto;

            background: #fff;

            padding: 20px;

            border-radius: 10px;

            box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);

        }

        .loader {

            display: none;

            border: 8px solid #f3f3f3;

            border-radius: 50%;

            border-top: 8px solid #3498db;

            width: 50px;

            height: 50px;

            animation: spin 1s linear infinite;

            margin: 20px auto;

        }

        @keyframes spin {

            0% { transform: rotate(0deg); }

            100% { transform: rotate(360deg); }

        }

        .card-body canvas {

            margin-top: 20px;

        }

        #map {

            height: 400px;

            margin-top: 20px;

        }

    </style>

</head>

<body>

    <header>

        <h1>Smart City Planner</h1>

    </header>

    <main>

        <!-- Input Form -->

        <form method="POST" id="districtForm">

            <div class="mb-3">

                <label for="district" class="form-label">Enter District:</label>

                <input type="text" class="form-control" id="district" name="district" placeholder="Enter district name">

            </div>

            <button type="submit" class="btn btn-primary w-100">Visualize</button>

        </form>

        <div class="loader"></div>

        {% if error %}

        <div class="alert alert-danger mt-3">{{ error }}</div>

        {% endif %}

        {% if data %}

        <!-- Dashboard Content -->

        <div id="dashboard">

            <!-- Weather Section -->

            <div class="card mt-4">

                <div class="card-header bg-primary text-white">

                    Weather Forecast (Weekly)

                </div>

                <div class="card-body">

                    <canvas id="weather-chart"></canvas>

                </div>

            </div>

            <!-- Electricity Section -->

            <div class="card mt-4">

                <div class="card-header bg-success text-white">

                    Electricity Usage (Weekly)

                </div>

                <div class="card-body">

                    <canvas id="electricity-chart"></canvas>

                </div>

            </div>

            <!-- Traffic Section -->

            <div class="card mt-4">

                <div class="card-header bg-warning text-dark">

                    Traffic Density Analysis

                </div>

                <div class="card-body">

                    <h5>Traffic Condition: <strong>{{ data["Traffic Condition"] }}</strong></h5>

                    <!-- Pie Chart for Vehicle Type Breakdown -->

                    <canvas id="traffic-vehicle-chart"></canvas>

                    <!-- Line Chart for Peak Hours -->

                    <h5 class="mt-4">Traffic Density by Hour</h5>

                    <canvas id="traffic-hourly-chart"></canvas>

                    <!-- Bar Chart for Average Speeds -->

                    <h5 class="mt-4">Average Speeds by Vehicle Type</h5>

                    <canvas id="traffic-speed-chart"></canvas>

                </div>

            </div>

            <!-- Map Section (Interactive Map) -->

            <div class="card mt-4">

                <div class="card-header bg-info text-white">

                    Location Map for District: {{ data["Area"] }}

                </div>

                <div class="card-body">

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

                </div>

            </div>

        </div>

        {% endif %}

    </main>

    <!-- JavaScript to Add Animations and Charts -->

    <script>

        // Function to simulate loading spinner

        $(document).ready(function () {

            $('#districtForm').on('submit', function () {

                $('.loader').show();

            });

        });

        {% if data %}

        // Weather Chart

        const weatherCtx = document.getElementById('weather-chart').getContext('2d');

        new Chart(weatherCtx, {

            type: 'line',

            data: {

                labels: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],

                datasets: [{

                    label: 'Average Temperature (°C)',

                    data: {{ data["Average Temperature (°C)"] | safe }},

                    borderColor: 'rgba(75, 192, 192, 1)',

                    backgroundColor: 'rgba(75, 192, 192, 0.2)',

                    borderWidth: 2,

                    tension: 0.4,

                    pointStyle: 'rectRounded',

                    pointHoverRadius: 8,

                }]

            },

            options: {

                responsive: true,

                plugins: {

                    legend: { display: true }

                },

                animation: {

                    duration: 2000,

                },

            }

        });

        // Electricity Chart

        const electricityCtx = document.getElementById('electricity-chart').getContext('2d');

        new Chart(electricityCtx, {

            type: 'bar',

            data: {

                labels: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],

                datasets: [{

                    label: 'Electricity Usage (kWh/day)',

                    data: {{ data["Electrical Consumption (kWh/day)"] | safe }},

                    backgroundColor: 'rgba(54, 162, 235, 0.5)',

                    borderColor: 'rgba(54, 162, 235, 1)',

                    borderWidth: 1,

                }]

            },

            options: {

                responsive: true,

                plugins: {

                    legend: { display: false }

                },

                animation: {

                    duration: 2000,

                },

            }

        });

        // Traffic Vehicle Breakdown Pie Chart

        const trafficVehicleCtx = document.getElementById('traffic-vehicle-chart').getContext('2d');

        new Chart(trafficVehicleCtx, {

            type: 'pie',

            data: {

                labels: ['Cars', 'Bikes', 'Buses', 'Trucks'],

                datasets: [{

                    data: [60, 20, 10, 10],  // Simulated data

                    backgroundColor: ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99'],

                    hoverOffset: 10,

                }]

            }

        });

        // Traffic Hourly Chart

        const trafficHourlyCtx = document.getElementById('traffic-hourly-chart').getContext('2d');

        new Chart(trafficHourlyCtx, {

            type: 'line',

            data: {

                labels: ['6 AM', '9 AM', '12 PM', '3 PM', '6 PM', '9 PM', '12 AM'],

                datasets: [{

                    label: 'Traffic Density (vehicles/km²)',

                    data: [1500, 2000, 3000, 3500, 4000, 3200, 1800],  // Simulated data

                    borderColor: '#ffcc00',

                    backgroundColor: 'rgba(255, 204, 0, 0.2)',

                    borderWidth: 2,

                    tension: 0.4,

                    pointStyle: 'circle',

                }]

            }

        });

        // Traffic Speed Analysis (Bar Chart)

        const trafficSpeedCtx = document.getElementById('traffic-speed-chart').getContext('2d');

        new Chart(trafficSpeedCtx, {

            type: 'bar',

            data: {

                labels: ['Cars', 'Bikes', 'Buses', 'Trucks'],

                datasets: [{

                    label: 'Average Speed (km/h)',

                    data: [60, 40, 30, 20],  // Simulated data

                    backgroundColor: '#ff6347',

                    borderColor: '#ff6347',

                    borderWidth: 1,

                }]

            },

            options: {

                responsive: true,

                plugins: {

                    legend: { display: false }

                }

            }

        });

        {% endif %}

    </script>

    <!-- Leaflet Map Initialization -->

{% if data %}

<script>

    const latitude = {{ data["Latitude"] }};

    const longitude = {{ data["Longitude"] }};

    const map = L.map('map').setView([latitude, longitude], 10);  // Set the map center dynamically

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

        attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

    }).addTo(map);

    // Add a marker for the district

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

        .bindPopup(`<b>${{ data["Area"] }}</b><br>Traffic: ${{ data["Traffic Condition"] }}.`)

        .openPopup();

</script>

{% endif %}

</body>

</html>

"""

@app.route("/", methods=["GET", "POST"])

def index():

    error = None

    data = None

    if request.method == "POST":

        district = request.form.get("district", "").strip()

        if not district:

            error = "Please enter a valid district name."

        else:

            # Fetch data for the requested district

            district\_data = df[df["Area"].str.lower() == district.lower()]

            if district\_data.empty:

                error = f"No data found for district: {district}"

            else:

                data = district\_data.iloc[0].to\_dict()

    return render\_template\_string(template, error=error, data=data)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

import pandas as pd

import random

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

import pandas as pd

import random

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

# Predefined list of districts with latitude and longitude

districts\_info = [

{"Area": "Ariyalur", "Latitude": 11.1385, "Longitude": 79.0756},

{"Area": "Chengalpattu", "Latitude": 12.697, "Longitude": 79.9828},

{"Area": "Chennai", "Latitude": 13.0827, "Longitude": 80.2707},

{"Area": "Coimbatore", "Latitude": 11.0168, "Longitude": 76.9558},

{"Area": "Cuddalore", "Latitude": 11.748, "Longitude": 79.7714},

{"Area": "Dharmapuri", "Latitude": 12.1357, "Longitude": 78.1612},

{"Area": "Dindigul", "Latitude": 10.362, "Longitude": 77.9705},

{"Area": "Erode", "Latitude": 11.341, "Longitude": 77.7172},

{"Area": "Kallakurichi", "Latitude": 11.7386, "Longitude": 78.9604},

{"Area": "Kancheepuram", "Latitude": 12.8342, "Longitude": 79.7036},

{"Area": "Karur", "Latitude": 10.9601, "Longitude": 78.0766},

{"Area": "Krishnagiri", "Latitude": 12.5186, "Longitude": 78.2133},

{"Area": "Madurai", "Latitude": 9.9252, "Longitude": 78.1198},

{"Area": "Mayiladuthurai", "Latitude": 11.1035, "Longitude": 79.655},

{"Area": "Nagapattinam", "Latitude": 10.7656, "Longitude": 79.8425},

{"Area": "Kanyakumari", "Latitude": 8.0883, "Longitude": 77.5385},

{"Area": "Namakkal", "Latitude": 11.2186, "Longitude": 78.1676},

{"Area": "Perambalur", "Latitude": 11.2333, "Longitude": 78.8833},

{"Area": "Pudukottai", "Latitude": 10.3797, "Longitude": 78.82},

{"Area": "Ramanathapuram", "Latitude": 9.3708, "Longitude": 78.8307},

{"Area": "Ranipet", "Latitude": 12.9337, "Longitude": 79.3339},

{"Area": "Salem", "Latitude": 11.6643, "Longitude": 78.146},

{"Area": "Sivagangai", "Latitude": 9.8478, "Longitude": 78.488},

{"Area": "Tenkasi", "Latitude": 8.9587, "Longitude": 77.3152},

{"Area": "Thanjavur", "Latitude": 10.7867, "Longitude": 79.1391},

{"Area": "Theni", "Latitude": 10.0104, "Longitude": 77.4777},

{"Area": "Thiruvallur", "Latitude": 13.1431, "Longitude": 79.9085},

{"Area": "Thiruvarur", "Latitude": 10.7672, "Longitude": 79.6366},

{"Area": "Tuticorin", "Latitude": 8.7642, "Longitude": 78.1348},

{"Area": "Tiruchirappalli", "Latitude": 10.7905, "Longitude": 78.7047},

{"Area": "Thirunelveli", "Latitude": 8.7139, "Longitude": 77.7567},

{"Area": "Tirupathur", "Latitude": 12.495, "Longitude": 78.5653},

{"Area": "Tiruppur", "Latitude": 11.1085, "Longitude": 77.3411},

{"Area": "Tiruvannamalai", "Latitude": 12.2253, "Longitude": 79.0747},

{"Area": "The Nilgiris", "Latitude": 11.4143, "Longitude": 76.695},

{"Area": "Vellore", "Latitude": 12.9165, "Longitude": 79.1325},

{"Area": "Viluppuram", "Latitude": 11.9401, "Longitude": 79.4977},

{"Area": "Virudhunagar", "Latitude": 9.5741, "Longitude": 77.9624},

]

# Generate synthetic dataset

data = []

for district in districts\_info:

traffic\_density = random.randint(100, 5000) # vehicles/km²

traffic\_condition = random.choice(["Low", "Moderate", "High", "Congested"])

electrical\_consumption = [round(random.uniform(50, 500), 2) for \_ in range(7)] # Weekly kWh/day

avg\_temperature = [round(random.uniform(20, 40), 1) for \_ in range(7)] # Weekly °C

weather\_condition = random.choice(["Sunny", "Rainy", "Cloudy", "Stormy", "Clear"])

# Add data to the list

data.append({

"Area": district["Area"],

"Latitude": district["Latitude"],

"Longitude": district["Longitude"],

"Traffic Density (vehicles/km²)": traffic\_density,

"Traffic Condition": traffic\_condition,

"Electrical Consumption (kWh/day)": electrical\_consumption,

"Average Temperature (°C)": avg\_temperature,

"Weather Condition": weather\_condition

})

# Create a DataFrame

df = pd.DataFrame(data)

# Flask application remains the same as before.

# Flask Application

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

# Enhanced Template with Diverse Visualizations

template = """

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Smart City Planner</title>

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- Chart.js -->

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<!-- jQuery -->

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<!-- Leaflet for Map -->

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

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

<style>

body {

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

background-color: #f4f6f9;

}

header {

text-align: center;

background: #2c3e50;

color: #fff;

padding: 20px 0;

margin-bottom: 20px;

}

main {

max-width: 900px;

margin: auto;

background: #fff;

padding: 20px;

border-radius: 10px;

box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);

}

.loader {

display: none;

border: 8px solid #f3f3f3;

border-radius: 50%;

border-top: 8px solid #3498db;

width: 50px;

height: 50px;

animation: spin 1s linear infinite;

margin: 20px auto;

}

@keyframes spin {

0% { transform: rotate(0deg); }

100% { transform: rotate(360deg); }

}

.card-body canvas {

margin-top: 20px;

}

#map {

height: 400px;

margin-top: 20px;

}

</style>

</head>

<body>

<header>

<h1>Smart City Planner</h1>

</header>

<main>

<!-- Input Form -->

<form method="POST" id="districtForm">

<div class="mb-3">

<label for="district" class="form-label">Enter District:</label>

<input type="text" class="form-control" id="district" name="district" placeholder="Enter district name">

</div>

<button type="submit" class="btn btn-primary w-100">Visualize</button>

</form>

<div class="loader"></div>

{% if error %}

<div class="alert alert-danger mt-3">{{ error }}</div>

{% endif %}

{% if data %}

<!-- Dashboard Content -->

<div id="dashboard">

<!-- Weather Section -->

<div class="card mt-4">

<div class="card-header bg-primary text-white">

Weather Forecast (Weekly)

</div>

<div class="card-body">

<canvas id="weather-chart"></canvas>

</div>

</div>

<!-- Electricity Section -->

<div class="card mt-4">

<div class="card-header bg-success text-white">

Electricity Usage (Weekly)

</div>

<div class="card-body">

<canvas id="electricity-chart"></canvas>

</div>

</div>

<!-- Traffic Section -->

<div class="card mt-4">

<div class="card-header bg-warning text-dark">

Traffic Density Analysis

</div>

<div class="card-body">

<h5>Traffic Condition: <strong>{{ data["Traffic Condition"] }}</strong></h5>

<!-- Pie Chart for Vehicle Type Breakdown -->

<canvas id="traffic-vehicle-chart"></canvas>

<!-- Line Chart for Peak Hours -->

<h5 class="mt-4">Traffic Density by Hour</h5>

<canvas id="traffic-hourly-chart"></canvas>

<!-- Bar Chart for Average Speeds -->

<h5 class="mt-4">Average Speeds by Vehicle Type</h5>

<canvas id="traffic-speed-chart"></canvas>

</div>

</div>

<!-- Map Section (Interactive Map) -->

<div class="card mt-4">

<div class="card-header bg-info text-white">

Location Map for District: {{ data["Area"] }}

</div>

<div class="card-body">

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

</div>

</div>

</div>

{% endif %}

</main>

<!-- JavaScript to Add Animations and Charts -->

<script>

// Function to simulate loading spinner

$(document).ready(function () {

$('#districtForm').on('submit', function () {

$('.loader').show();

});

});

{% if data %}

// Weather Chart

const weatherCtx = document.getElementById('weather-chart').getContext('2d');

new Chart(weatherCtx, {

type: 'line',

data: {

labels: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],

datasets: [{

label: 'Average Temperature (°C)',

data: {{ data["Average Temperature (°C)"] | safe }},

borderColor: 'rgba(75, 192, 192, 1)',

backgroundColor: 'rgba(75, 192, 192, 0.2)',

borderWidth: 2,

tension: 0.4,

pointStyle: 'rectRounded',

pointHoverRadius: 8,

}]

},

options: {

responsive: true,

plugins: {

legend: { display: true }

},

animation: {

duration: 2000,

},

}

});

// Electricity Chart

const electricityCtx = document.getElementById('electricity-chart').getContext('2d');

new Chart(electricityCtx, {

type: 'bar',

data: {

labels: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],

datasets: [{

label: 'Electricity Usage (kWh/day)',

data: {{ data["Electrical Consumption (kWh/day)"] | safe }},

backgroundColor: 'rgba(54, 162, 235, 0.5)',

borderColor: 'rgba(54, 162, 235, 1)',

borderWidth: 1,

}]

},

options: {

responsive: true,

plugins: {

legend: { display: false }

},

animation: {

duration: 2000,

},

}

});

// Traffic Vehicle Breakdown Pie Chart

const trafficVehicleCtx = document.getElementById('traffic-vehicle-chart').getContext('2d');

new Chart(trafficVehicleCtx, {

type: 'pie',

data: {

labels: ['Cars', 'Bikes', 'Buses', 'Trucks'],

datasets: [{

data: [60, 20, 10, 10], // Simulated data

backgroundColor: ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99'],

hoverOffset: 10,

}]

}

});

// Traffic Hourly Chart

const trafficHourlyCtx = document.getElementById('traffic-hourly-chart').getContext('2d');

new Chart(trafficHourlyCtx, {

type: 'line',

data: {

labels: ['6 AM', '9 AM', '12 PM', '3 PM', '6 PM', '9 PM', '12 AM'],

datasets: [{

label: 'Traffic Density (vehicles/km²)',

data: [1500, 2000, 3000, 3500, 4000, 3200, 1800], // Simulated data

borderColor: '#ffcc00',

backgroundColor: 'rgba(255, 204, 0, 0.2)',

borderWidth: 2,

tension: 0.4,

pointStyle: 'circle',

}]

}

});

// Traffic Speed Analysis (Bar Chart)

const trafficSpeedCtx = document.getElementById('traffic-speed-chart').getContext('2d');

new Chart(trafficSpeedCtx, {

type: 'bar',

data: {

labels: ['Cars', 'Bikes', 'Buses', 'Trucks'],

datasets: [{

label: 'Average Speed (km/h)',

data: [60, 40, 30, 20], // Simulated data

backgroundColor: '#ff6347',

borderColor: '#ff6347',

borderWidth: 1,

}]

},

options: {

responsive: true,

plugins: {

legend: { display: false }

}

}

});

{% endif %}

</script>

<!-- Leaflet Map Initialization -->

{% if data %}

<script>

const latitude = {{ data["Latitude"] }};

const longitude = {{ data["Longitude"] }};

const map = L.map('map').setView([latitude, longitude], 10); // Set the map center dynamically

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

attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

}).addTo(map);

// Add a marker for the district

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

.bindPopup(`<b>${{ data["Area"] }}</b><br>Traffic: ${{ data["Traffic Condition"] }}.`)

.openPopup();

</script>

{% endif %}

</body>

</html>

"""

@app.route("/", methods=["GET", "POST"])

def index():

error = None

data = None

if request.method == "POST":

district = request.form.get("district", "").strip()

if not district:

error = "Please enter a valid district name."

else:

# Fetch data for the requested district

district\_data = df[df["Area"].str.lower() == district.lower()]

if district\_data.empty:

error = f"No data found for district: {district}"

else:

data = district\_data.iloc[0].to\_dict()

return render\_template\_string(template, error=error, data=data)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)