TASK 05

Weather App with Location

Weather App Build a web page that fetches weather data from a weather API based on the user's location or a user-inputted location. Display the current weather conditions, temperature, and other relevant information.

Weather App with Location
This is a basic structure for a weather app that uses an API to fetch weather data.

To build a weather app that fetches weather data from an API and displays it based on the user's location or a user-inputted location, follow these steps:

Set Up Your Environment: Create a new project directory and set up the necessary files.

HTML Structure: Create a basic HTML structure for your webpage.

CSS Styling: Add some styling to make the webpage look nice.

JavaScript Functionality: Use JavaScript to fetch data from a weather API and display it on the webpage.

Here’s a step-by-step guide:

1. Set Up Your Environment
    Create a new directory for your project and add the following files:

index.html
styles.css
script.js

2. HTML Structure
    In index.html, create the basic structure of the page:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Weather App</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Weather App</h1>
        <div class="search">
            <input type="text" id="locationInput" placeholder="Enter a
location">
            <button onclick="getWeatherByInput()">Get Weather</button>
        </div>
        <div class="weather-info">
            <h2 id="location"></h2>
            <p id="description"></p>
            <p id="temperature"></p>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

3. CSS Styling
    In styles.css, add some basic styling:

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

.container {
    text-align: center;
    background: #fff;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

.search-box {
    margin-bottom: 20px;
}

.search-box input {
    padding: 10px;
    width: 200px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.search-box button {
    padding: 10px 15px;
    border: none;
    background: #007bff;
    color: #fff;
    border-radius: 5px;
    cursor: pointer;
}

.weather-info {
    margin-top: 20px;
}

.weather-info div {
    margin: 10px 0;
}
```

Step 5: JavaScript Functionality

    In your script.js file, add the JavaScript to fetch and display the weather data:

```
const apiKey = 'YOUR_API_KEY'; // Replace with your OpenWeatherMap API key

async function fetchWeather() {
    const location = document.getElementById('location-input').value;
    if (!location) {
        alert('Please enter a location');
        return;
    }
```

```
    const url = `https://api.openweathermap.org/data/2.5/weather?q=$
{location}&appid=${apiKey}&units=metric`;
    try {
        const response = await fetch(url);
        const data = await response.json();

        if (data.cod === 200) {
            displayWeather(data);
        } else {
            alert(data.message);
        }
    } catch (error) {
        alert('Error fetching weather data');
    }
}

function displayWeather(data) {
    const weatherInfo = document.getElementById('weather-info');
    weatherInfo.innerHTML = `
        <div><strong>Location:</strong> ${data.name}, ${data.sys.country}</div>
        <div><strong>Temperature:</strong> ${data.main.temp} °C</div>
        <div><strong>Weather:</strong> ${data.weather[0].description}</div>
        <div><strong>Humidity:</strong> ${data.main.humidity}%</div>
        <div><strong>Wind Speed:</strong> ${data.wind.speed} m/s</div>
    `;
}
```

Step 6: Testing

1.Replace 'YOUR_API_KEY' in script.js with your actual API key from OpenWeatherMap.
2.Open index.html in your browser.
3.Enter a location and click the "Get Weather" button to see the current weather information for that location.

Additional Features
To enhance your weather app, consider adding:

Geolocation API: Automatically fetch weather data based on the user's current location.

Forecast Data: Display a 5-day weather forecast.

Styling: Improve the UI/UX with better design and animations.

Error Handling: Better handling for different types of errors (e.g., network errors, invalid location).

This basic setup will get you started on creating a functional weather app. Feel free to expand and customize it according to your needs!