Week-6

Aim:

To develop a weather application using HTML, CSS, and JavaScript that fetches and displays real-time weather data based on user input, while also utilizing localStorage to save the last searched city.

Description:

The project consists of an HTML file that provides a simple user interface with:
- An input field for entering a city name.
- A button to fetch weather information.
- A results section that dynamically updates with weather details such as temperature, condition, and humidity.

The application uses the Fetch API with async/await to retrieve weather data from the wttr.in API in JSON format. Errors such as invalid city names or network failures are handled gracefully.

The localStorage API is used to remember the last searched city so that when the page is reloaded, weather data for that city is automatically displayed. The UI is styled with CSS for a clean and user-friendly experience.

Tasks:

1. Design a simple UI with: input field for city name, a button to fetch weather details, and a section to display weather output.
2. Use the fetch API to retrieve weather data from wttr.in.
3. Handle asynchronous operations using async/await syntax and proper error handling with try...catch.
4. Dynamically update the DOM with the weather information.
5. Implement localStorage to save the last searched city and automatically fetch and display that city's weather on page reload.

Programs:

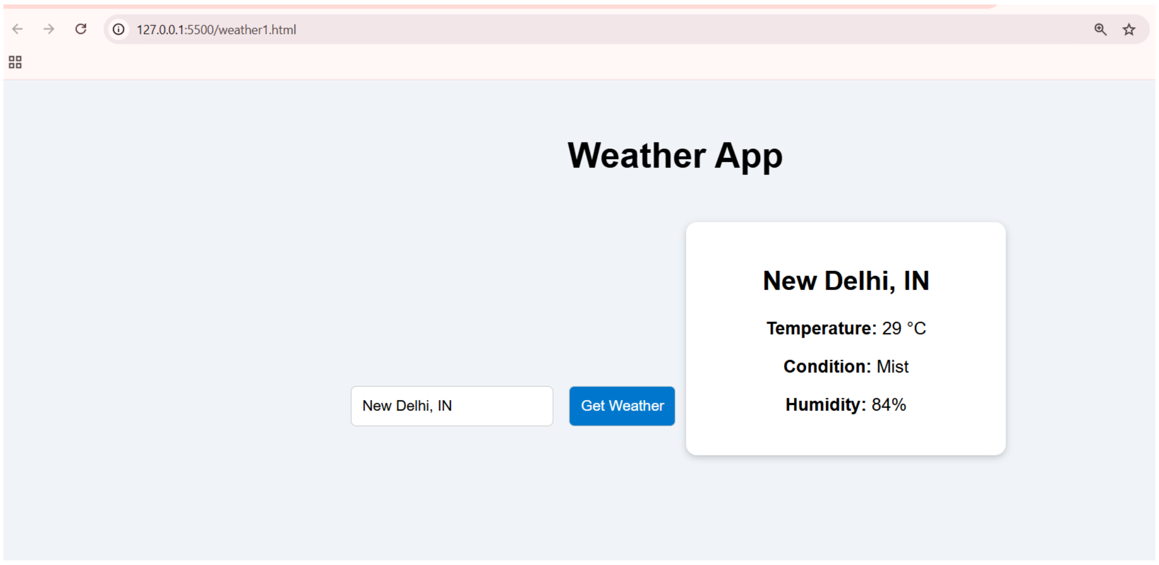
index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Weather App</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin: 50px;
      background: #f0f4f8;
    }
    .weather-box {
      margin-top: 20px;
      padding: 20px;
      display: inline-block;
      border-radius: 10px;
      background: #fff;
      box-shadow: 0 2px 6px rgba(0,0,0,0.2);
      min-width: 250px;
    }
    input, button {
      padding: 10px;
      margin: 5px;
      border-radius: 5px;
      border: 1px solid #ccc;
    }
    button {
      background: #0077cc;
      color: white;
      cursor: pointer;
    }
    button:hover {
      background: #005fa3;
    }
  </style>
</head>
<body>

  <h1>Weather App</h1>

  <input type="text" id="cityInput" placeholder="Enter city name" />
  <button id="getWeatherBtn">Get Weather</button>

  <div id="weatherResult" class="weather-box"></div>

  <script>
    const cityInput = document.getElementById("cityInput");
    const getWeatherBtn = document.getElementById("getWeatherBtn");
    const weatherResult = document.getElementById("weatherResult");

    async function fetchWeather(city) {
      try {
        if (!city) {
          weatherResult.innerHTML = "<p>Please enter a city name.</p>";
          return;
        }

        const response = await
fetch(`https://wttr.in/${encodeURIComponent(city)}?format=j1`);
        if (!response.ok) {
          throw new Error("City not found or API error.");
        }

        const data = await response.json();
        displayWeather(city, data);

        localStorage.setItem("lastCity", city);
      } catch (error) {
        weatherResult.innerHTML = `<p style="color:red;">${error.message}</p>`;
      }
    }

    function displayWeather(city, data) {
      const current = data.current_condition[0];
      weatherResult.innerHTML = `
        <h2>${city}</h2>
        <p><strong>Temperature:</strong> ${current.temp_C} °C</p>
        <p><strong>Condition:</strong> ${current.weatherDesc[0].value}</p>
        <p><strong>Humidity:</strong> ${current.humidity}%</p>
      `;
    }

    getWeatherBtn.addEventListener("click", () => {
      const city = cityInput.value.trim();
      fetchWeather(city);
    });
```

```
    window.onload = () => {
      const lastCity = localStorage.getItem("lastCity");
      if (lastCity) {
        fetchWeather(lastCity);
      }
    };
  </script>

</body>
</html>
```

Output:

- User enters a city name (e.g., Hyderabad) and clicks Get Weather.
- The weather box displays:
  * City name.
  * Temperature in Celsius.
  * Weather condition (e.g., Partly Cloudy).
  * Humidity percentage.
- On reload, the app automatically fetches weather for the last searched city.



New Delhi, IN

Temperature: 29 °C

Condition: Mist

Humidity: 84%

Results:

As a result, the weather application was successfully implemented. Weather data is fetched using Fetch API and displayed dynamically. Errors such as invalid city inputs are handled gracefully. The localStorage integration ensures that the last searched city is remembered and auto-fetched. The UI is clean, interactive, and user-friendly.

VIVA QUESTIONS:

1. Which JavaScript method is used to make API calls to retrieve weather data, and how does it work?

- The **fetch()** method is commonly used.
- It sends an HTTP request to the weather API endpoint and returns a **Promise**.
- Example:
- fetch(`https://api.openweathermap.org/data/2.5/weather?q=London&appid=API_KEY`)
- .then(response => response.json())
- .then(data => console.log(data));
- It works by sending a request → receiving a response → converting it into JSON → using the data in JavaScript.

2. What is the role of async/await in handling asynchronous API requests, and how is it different from using .then()?

async/await allows writing asynchronous code in a synchronous-like style, making it easier to read and maintain.

Difference from .then():

.then() uses callback chaining → can become harder to read when there are multiple steps.

async/await makes code look cleaner and avoids callback hell.

Example:

```
async function getWeather(city) {
  try {
    let response = await
fetch(`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=API_KEY`);
    let data = await response.json();
    console.log(data);
  } catch (error) {
    console.error(error);
  }
}
```

3. How can you update the DOM to display weather details like temperature and description dynamically?

- By selecting HTML elements using **document.getElementById()** or **document.querySelector()** and then updating their textContent or innerHTML.

  Example:

- document.getElementById("temp").textContent = data.main.temp + " °C";

document.getElementById("desc").textContent = data.weather[0].description;

4. What is localStorage in JavaScript, and how can it be used to store the last searched city name?

- localStorage is a **browser storage mechanism** that stores data in key-value pairs and **persists even after page reload**.

Example:

```
// Save city
localStorage.setItem("lastCity", city);


// Retrieve city
let lastCity = localStorage.getItem("lastCity");
```

- This can be used to **remember the last searched city** and automatically load its weather when the user revisits.

5. How would you handle errors such as an invalid city name or a failed API request in a user-friendly way?

Use **try...catch** with async/await or .catch() with fetch().

Display an **error message** to the user instead of breaking the app.

Example:

```
async function getWeather(city) {
  try {
    let response = await fetch(`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=API_KEY`);
    if (!response.ok) {
      throw new Error("City not found");
    }
    let data = await response.json();
    // update DOM
  } catch (error) {
    document.getElementById("error").textContent = "❌ Invalid city name or network error.";
  }
}
```