# Introduction

## The Weather App is designed to allow users to input any city name and instantly receive up-to-date weather information, such as temperature, humidity and weather conditions. With smooth, efficient navigation and a minimalist layout, users can search, view, and understand weather conditions without any delays. The app is designed to be easy to use for anyone, regardless of their technical background. Users can quickly search for weather data without unnecessary complexity. It retrieves weather data from the OpenWeatherMap API, allowing users to view information such as temperature, humidity, and wind speed for different countries and cities around the world. The interface is designed to be user friendly.

# Objectives

## **To Provide Accurate and Real-Time Weather Information**

## To Ensure User-Friendly and Accessible Design

# Features

## Search Weather by City: Users can input a city name to get the current weather conditions.

## Geo-Location based Weather: Upon loading, the app detects the user’s location using the Geolocation API and displays the weather data for their current location.

## **Dynamic Weather Icons:** Different weather conditions (e.g., clouds, rain, mist) are represented by icons based on the weather data.

## **Error Handling:** Displays an error message if the entered city is not found or if there's a problem fetching the data.

# Technologies Used

## **HTML5:** Structure of the web page.

## **CSS3:** Styling and layout.

## **JavaScript:** Fetching weather data, handling user input, geolocation, and DOM manipulation.

## **OpenWeatherMap API:** Provides real-time weather data.

# Functionality

## **Fetch Weather data**

## The fetch weather function is responsible for retrieving weather data based on user input or the user's current location:

## async function fetchWeather(query) {

## const response = await fetch(`${weatherURL}&${query}&appid=${APIkey}`);

## const data = await response.json();

## if (response.status === 404 || !data.name) {

## showError();

## } else {

## updateWeather(data);

## }

## }

## **Updating the user UI**

## When the weather data is fetched successfully, the update weather function updates the DOM with the current weather information:

## function updateWeather(data) {

## cityElement.textContent = data.name;

## tempElement.textContent = `${Math.round(data.main.temp)} °C`;

## humidityElement.textContent = `${data.main.humidity}%`;

## windElement.textContent = `${data.wind.speed} km/hr`;

## const weatherCondition = data.weather[0].main;

## weatherIcon.src = getWeatherIcon(weatherCondition);

## weatherBox.style.display = "block";

## errorBox.style.display = "none";

## }

## **Displaying Appropriate Weather Icon**

## The get weather icon function returns the path of the weather icon based on the current weather condition:

## function getWeatherIcon(weatherCondition) {

## switch (weatherCondition) {

## case "Clear":

## return "pictures/clear.png";

## case "Clouds":

## return "pictures/clouds.png";

## case "Rain":

## return "pictures/rain.png";

## case "Drizzle":

## return "pictures/drizzle.png";

## case "Thunderstorm":

## return "pictures/thunderstorm.png";

## case "Snow":

## return "pictures/snow.png";

## case "Mist":

## return "pictures/mist.png";

## case "Haze":

## return "pictures/haze.png";

## case "Fog":

## return "pictures/fog.png";

## case "Smoke":

## return "pictures/smoke.png";

## case "Dust":

## return "pictures/dust.png";

## case "Squall":

## return "pictures/squall.png";

## case "Tornado":

## return "pictures/tornado.png";

## default:

## return "pictures/clear.png";

## }

## }

## **Geolocation support**

## The application detects the user’s current location using the browser’s Geolocation API. If permission is granted, it fetches weather data for the user's coordinates:

## function getCurrentLocationWeather() {

## if (navigator.geolocation) {

## navigator.geolocation.getCurrentPosition(

## (position) => {

## const lat = position.coords.latitude;

## const lon = position.coords.longitude;

## fetchWeather(`lat=${lat}&lon=${lon}`);

## },

## (error) => {

## showError();

## console.error("Error getting location:", error);

## }

## );

## } else {

## showError();

## console.error("Geolocation is not supported by this browser.");

## }

## }

## **Error handling**

## The show Error function is triggered when weather data cannot be fetched:

## function showError() {

## weatherBox.style.display = "none";

## errorBox.style.display = "block";

}