**KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**KUMASI**

COLLEGE OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

PROJECT DOCUMENTATION



**TOPIC: DEVELOPING A WEB APPLICATION TO DETERMINE THE WEATHER USING HTML, CSS AND JAVASCRIPT**

**DOGBE FELIX - 4206920**

# **SUPERVISED BY:** **DR. (MR.) YAW MARFO MISSAH**

**DOCUMENTATION ON MY PROJECT**

**Introduction**

**Code Overview**

**Html Structure**

**CSS Stylling**

**JavaScript Functionality**

**Usage**

**Conclusion**

# CHAPTER 1: INTRODUCTION

In an increasingly interconnected world, having instant access to weather information is essential for planning daily activities, travel, and more. The proposed project aims to develop a user-friendly Weather Application that allows users to quickly and conveniently retrieve real-time weather data for any city around the globe. This application will serve as a valuable tool for individuals seeking accurate weather forecasts to make informed decisions.

Weather Application utilizes the latest web technologies to offer a user-friendly and visually appealing interface. Users can input the name of a city and instantly access important weather information like temperature, weather conditions, and date. This project is not only useful for individuals but also a great opportunity for developers interested in web development, API integration, and user interface design.

To create a seamless user experience, we will use HTML, CSS, and JavaScript. By integrating with the OpenWeatherMap API ,it will deliver accurate and current weather data to our users. Additionally, this project will provide insights into asynchronous programming, data visualization, and responsive design principles.

**OBJECTIVES**

The proposed project aims to develop a user-friendly and responsive weather application that provides real-time weather information to users based on their location input. The application will be implemented using JavaScript, CSS, and HTML, ensuring a seamless user experience across various devices and browsers. This project aims at:

1.Real-Time Weather Data: Create a web application that fetches and displays up-to-date weather information from a reliable weather API based on the user's location.

2.User-Friendly Interface: Design an intuitive and visually appealing user interface using CSS and HTML to ensure ease of use and accessibility.

3.Responsive Design: Develop a responsive layout that adapts to various screen sizes, including desktops, tablets, and mobile devices.

4.Location Detection: Implement a feature that allows users to automatically detect their location or manually input a location to retrieve weather data.

1. Weather Details: Display a range of weather details such as temperature, humidity, wind speed, and weather conditions (e.g., sunny, cloudy, rainy) for the selected location.

**PROJECT FEATURES**

1.Location Selection: Users can input their location or use browser-based location detection.

2.Current Weather: Display the current weather conditions for the selected location.

3.Weather Icons: Use appropriate weather icons to visually represent weather conditions.

4.Weather Details: Show additional details such as temperature, humidity, and wind speed.

5.Forecast Display: Provide a multi-day weather forecast with daily temperature and condition information.

**CODE OVERVIEW**

An overview of the code used to build the weather application reveals that it is composed of HTML, CSS, and JavaScript. The application fetches weather data from the OpenWeatherMap API and automatically updates the user interface to display the latest weather information.

The HTML file, index.html, comprises various sections that contribute to the application's user interface. These include elements for the application's title, search input, and weather details.

The CSS file, weather.css, offers styling for the application's layout and appearance. It defines styles for the container, input box, temperature display, and other elements.

The JavaScript file, weather.js, contains the programming logic to retrieve weather data from the OpenWeatherMap API and display it on the webpage. It listens for user input, fetches weather data, and updates the UI based on the received data.

**HTML STRUCTURE**

<!DOCTYPE html><html><head>

<title>My Weather Application</title>

<link rel="stylesheet" type="text/css" href="weather.css">

<link rel="stylesheet" type="text/css" href="https://fonts.google.com/"></head><body>

<!DOCTYPE html>: This declaration specifies the document type and version of HTML being used.

<html>: The root element of the HTML document, containing the entire webpage's content.

<head>: This section contains metadata about the webpage, like the title and external resource links.

<title>: Sets the title that appears in the browser tab.

<link>: Used to link external resources, like stylesheets. In this case, it links the weather.css stylesheet for styling.

<link>: Another <link> tag attempting to link a Google Fonts stylesheet, but the URL seems incomplete.

<div class="container"> <div class="heading"> <h1>Weather Application</h1> </div> <div class="input"> <input type="text" name="search" class="input-search" placeholder="Search for the city.."> </div> <div class="box"> <div class="location"> <div class="location-city"></div> <div class="location-date"></div> </div> <div class="temperature"> <div class="temperature-temp"><span></span></div> <div class="temperature-type"></div> <div class="temperature-range"></div> </div> </div> </div>

<div class="container">: A container to group and organize the main content of the webpage.

<div class="heading">: A section for the application's heading.

<h1>: A level 1 heading displaying "Weather Application".

<div class="input">: A section for user input.

<input>: An input field for users to type the city name to search for weather.

<div class="box">: A box containing weather-related information.

Nested <div> elements with classes for displaying location information (location) and temperature details (temperature).

<script src="weather.js"></script>

</body>

</html>

<script>: Links an external JavaScript file named weather.js to add functionality to the webpage.

</body>: Closes the body tag.

</html>: Closes the HTML document.

**CSS STYLLING STRUCTURE**

\*{

margin:0;

padding:0;

box-sizing: border-box;

\*{}: Applies the following styles to all elements in the document.

margin:0; padding:0;: Sets the margins and padding of all elements to zero.

box-sizing: border-box;: Ensures that padding and border are included in the element's total width and height.

cssCopy code

body{

background-image: url('https://www.expenseanywhere.com/wp-content/uploads/2016/09/website-backgrounds-E280AB1E280AC-E280ABE280AC.jpg');

background-repeat: no-repeat;

background-size: cover;

/\*background-color: green;\*/

body{}: Applies styles to the entire body of the webpage.

background-image: url(...): Sets a background image for the body.

background-repeat: no-repeat;: Prevents the background image from repeating.

background-size: cover;: Makes the background image cover the entire body.

* **/\*background-color: green;\*/**: A commented-out line to set a background color (not currently in use).

cssCopy code

.container{

display: flex;

min-height: 100vh;

flex-direction: column;

}

.container{}: Styles for elements with the class "container".

display: flex;: Sets the container to use flex layout.

min-height: 100vh;: Sets the minimum height of the container to 100% of the viewport height.

flex-direction: column;: Displays child elements in a column layout.

**JAVASCRIPT FUNCTIONALITY**

window.onload = function () {

document.querySelector('.input-search').value = '';

}

window.onload = function () {...}: Defines a function to be executed when the window (webpage) finishes loading.

document.querySelector('.input-search'): Selects the HTML element with the class "input-search".

.value = '': Sets the value of the selected input element (search box) to an empty string.

const api\_details = {

url: "http://api.openweathermap.org/data/2.5/",

api\_key: "60bbd59ec7556e88c0f6b5a2080aebaa"

};

const api\_details = {...}: Defines an object named api\_details to store API-related information.

url: "http://api.openweathermap.org/data/2.5/": Specifies the base URL for the OpenWeatherMap API.

api\_key: "60bbd59ec7556e88c0f6b5a2080aebaa": Sets the API key required for accessing the OpenWeatherMap API.

const input = document.querySelector('.input-search');

input.addEventListener('keypress', showData);

const input = document.querySelector('.input-search');: Selects the input element with the class "input-search" and stores it in the input variable.

input.addEventListener('keypress', showData);: Adds an event listener to the input element that listens for a keypress event and calls the showData function.

async function showData(e) {

if (e.keyCode === 13) {

showResults(input.value);

}

async function showData(e) {...}: Defines an asynchronous function named showData that takes an event object e as a parameter.

if (e.keyCode === 13) {...}: Checks if the key code of the pressed key is 13 (Enter key).

showResults(input.value);: Calls the showResults function with the value of the input (city name) as an argument.

async function showResults(value) {

const data = await fetch(`${api\_details.url}weather?q=${value}&units=metric&APPID=${api\_details.api\_key}`);

const fdata = await data.json();

console.log(fdata);

if (fdata.message === "city not found") {

// ...

} else {

displayData(fdata);

}

async function showResults(value) {...}: Defines an asynchronous function named showResults that takes the value (city name) as a parameter.

const data = await fetch(...);: Uses the fetch API to retrieve weather data from the OpenWeatherMap API based on the provided URL.

await data.json();: Converts the fetched data into JSON format.

console.log(fdata);: Logs the fetched data to the browser's console.

if (fdata.message === "city not found") {...}: Checks if the fetched data indicates that the city was not found.

displayData(fdata);: Calls the displayData function to show the weather information on the webpage.

function displayData(data) {

const location\_city = document.querySelector('.location-city');

location\_city.innerText = `${data.name},${data.sys.country}`;

// ...

const lowHigh = document.querySelector('.temperature-range');

lowHigh.innerHTML = `${Math.round(data.main.temp\_min)}<span>&#730C</span>/${Math.round(data.main.temp\_max)}<span>&#730C</span>`;

}

function displayData(data) {...}: Defines the displayData function that takes the data object as a parameter.

const location\_city = document.querySelector('.location-city');: Selects the element with the class "location-city" to display the city name and country code.

location\_city.innerText = ...;: Sets the inner text of the element to show the city name and country code.

const lowHigh = document.querySelector('.temperature-range');: Selects the element with the class "temperature-range" to display the temperature range.

lowHigh.innerHTML = ...;: Sets the HTML content of the element to show the temperature range.

**USAGE**

Open the HTML file (index.html) in a web browser.

Enter the name of the city you want to get weather information for in the search input field.

Press the "Enter" key to trigger the search and fetch weather data.

The application will display the city name, current date, temperature, weather type, and temperature range.

If the city is not found, an appropriate message will be displayed

**CONCLUSION**

The Weather Application is a straightforward yet functional tool that allows users to quickly check weather conditions for various cities. Its combination of HTML, CSS, and JavaScript creates an engaging and informative user experience. The application's modular code structure and use of the OpenWeatherMap API make it a suitable starting point for developers looking to create weather-related web applications.

**REFERENCES: <https://chat.openai.com/>**

**GITHUB LINK: <https://github.com/Spar1001>**

# 