WEATHER APP.

A PROJECT REPORT

Submitted by

Ashutosh Kumar



Lovely Professional University, Punjab

Name- Ashutosh Kumar

Section- K24DJ

Registration Number - 12404541  PAGE-1

Roll Number- 42

INDEX

|  |  |  |
| --- | --- | --- |
| NO. | CHAPTER NAME | PAGE NO. |
| 1. | Title Page | 1 |
| 2. | Introduction and Technology used | 3-5 |
| 3. | Workflow(flowchart) | 6-7 |
| 4. | Working of project | 8-9 |
| 5. | Code snippet (Html) | 10 |
| 6. | Code snippet (CSS) +Javascript | 11-14 |
| 7. | Screenshots of website, webpage | 15 |
| 8. | conclusions | 16 |
| 9. | LINK TO OPEN THE WEBSITE | 17 |
|  |  |  |

PAGE-2

# INTRODUCTION

Here's an introduction for a weather app project built using HTML, CSS, and JavaScript:

---

### Weather App

This Weather App is a simple and interactive application that allows users to check the current weather conditions for any city around the world. Built with HTML, CSS, and JavaScript, the app fetches real-time weather data from the OpenWeatherMap API and displays it dynamically on the screen.

#### Key Features:

Real-Time Data: Get up-to-date weather information including temperature, humidity, wind speed, and a brief weather description.

Responsive Design: The app is designed to be visually appealing on any device, with CSS styling that adapts to both mobile and desktop screens.

PAGE-3

Easy to Use: Users can simply enter the name of a city and click the "Get Weather" button to instantly receive the latest weather data for that location.

#### Technologies Used:

1. HTML: Provides the basic structure of the app, including the input field, button, and areas to display weather information.

2. CSS: Enhances the visual appeal of the app with a modern, clean design. It includes responsive styles to ensure a consistent look across devices.

3. JavaScript: Handles the API calls to fetch data from OpenWeatherMap, processes the JSON response, and dynamically updates the DOM to show weather information.

#### How It Works:

1. The user enters the name of a city and clicks "Get Weather."

2. JavaScript sends a request to the OpenWeatherMap API, passing in the city name and API key.

3. The API returns the current weather data, which JavaScript then displays on the page.

This project demonstrates the power of combining HTML, CSS, and JavaScript to create an interactive web application that provides real time information to users. It’s great starter project for anyone

PAGE-4

interested in learning about API integration and dynamic DOM manipulation in JavaScript.

# TECHNOLOGY-USED

**Technologies Used in the Weather App**

1. **HTML (HyperText Markup Language)**:
   * Provides the basic structure of the app.
   * Defines the layout and elements like input fields, buttons, and containers where weather information will be displayed.
2. **CSS (Cascading Style Sheets)**:
   * Styles the app to make it visually appealing.
   * Used for layout, colors, fonts, and responsiveness, ensuring the app looks good on different devices (mobile, tablet, and desktop).
   * Adds modern design elements such as shadows, gradients, and centered content.
3. **JavaScript**:
   * Handles the logic of the application, including fetching data and updating the interface dynamically.
   * Uses the Fetch API to make HTTP requests to the OpenWeatherMap API, retrieving real-time weather information based on the user's input.
   * Manipulates the DOM (Document Object Model) to display weather data like temperature, humidity, and wind speed.
4. **OpenWeatherMap API**:
   * Provides real-time weather data for cities around the world.
   * Returns data in JSON format, which includes information such as temperature, weather description, humidity, and wind speed.
   * Requires an API key to authenticate requests and access data. PAGE-5
5. **VS Code (Visual Studio Code)** (or any preferred code editor):
   * Used for writing, editing, and running the HTML, CSS, and JavaScript code.
   * Offers features like syntax highlighting, debugging, and live server extensions to preview the app during development.

These technologies work together to create a fully functional, interactive weather app that retrieves and displays real-time weather information based on the user’s input.

WORKFLOW

**Workflow for the Weather App**

1. **UI Structure with HTML**:
   * Create the HTML file to define the basic layout of the app.
   * Add an input field where users can type a city name, a button labeled "Get Weather," and sections to display weather information (like city name, temperature, humidity, and description).
   * Link the CSS and JavaScript files to the HTML file.
2. **Styling the App with CSS**:
   * Use CSS to style the elements: center the weather container on the page, add colors, fonts, and spacing to make it visually appealing.
   * Add responsive design elements to ensure the app looks good on both desktop and mobile devices.
   * Include transitions, shadows, and gradients for a modern design.
3. **Connecting to the OpenWeatherMap API with JavaScript**:
   * In the JavaScript file, define an apiKey variable with your OpenWeatherMap API key.
   * Set up a function called getWeather() that:
     + Fetches the city name entered by the user. PAGE-6
     + Constructs the API endpoint URL using the city name, API key, and metric units.
     + Sends a GET request to the OpenWeatherMap API.
4. **Handling the API Response**:
   * Use the Fetch API to handle the API call in getWeather():
     + If the API request is successful, retrieve the JSON response data.
     + Extract necessary data like temperature, weather description, humidity, and wind speed.
     + Update the relevant HTML elements to display this data on the page.
     + If an error occurs (e.g., city not found or invalid API key), display an error message.
5. **Displaying Weather Information Dynamically**:
   * Use JavaScript to manipulate the DOM and update the weather information in real-time.
   * Display the weather data within the designated HTML elements (temperature, humidity, wind speed, etc.).
   * Show or hide certain elements or messages based on the success or failure of the API call.
6. **Testing and Debugging**:
   * Run the app locally using a Live Server extension in VS Code.
   * Check the app by entering different city names to confirm it returns accurate weather data.

Project structure

1.index.html- The main HTML file containing the structure of the website  PAGE-7

2.style.css- The CSS file for styling the layout and design

3.Javascript.js- Handles the logic of the application, including fetching data and updating the interface dynamically.

## Working of project

**1. User Interface (HTML & CSS)**

* The app starts with a simple interface that includes an input field for the city name, a "Get Weather" button, and sections where the weather data (such as temperature, description, humidity, etc.) will be displayed.
* CSS is used to style the app, ensuring it is visually appealing and responsive to different screen sizes.

**2. User Input**

* The user types in the name of a city (e.g., "Amritsar") in the input field.
* When the "Get Weather" button is clicked, JavaScript captures the input and initiates the process of fetching weather data.

**3. API Call (JavaScript and OpenWeatherMap API)**

* The JavaScript function getWeather() takes the city name entered by the user and appends it to an API request URL.
* The function also includes the OpenWeatherMap API key and any query parameters, like units (metric for Celsius).
* JavaScript then uses the Fetch API to make an HTTP GET request to the OpenWeatherMap API with the constructed URL.

**4. Fetching Weather Data PAGE-8**

* The API returns data in JSON format if the request is successful.
* This JSON response contains various pieces of weather information for the specified city, such as temperature, humidity, weather description, and wind speed.

**5. Displaying Weather Data**

* JavaScript parses the JSON response and extracts relevant information.
* It then updates specific HTML elements on the page to display this information.
  + For example, the city name, temperature, weather description, humidity, and wind speed are dynamically updated with the data received from the API.

**6. Error Handling**

* If the API request fails (e.g., due to an incorrect city name or an invalid API key), the app catches the error.
* JavaScript displays an error message alert or updates the HTML to show a "City not found" message, enhancing the user experience and guiding the user to try again.

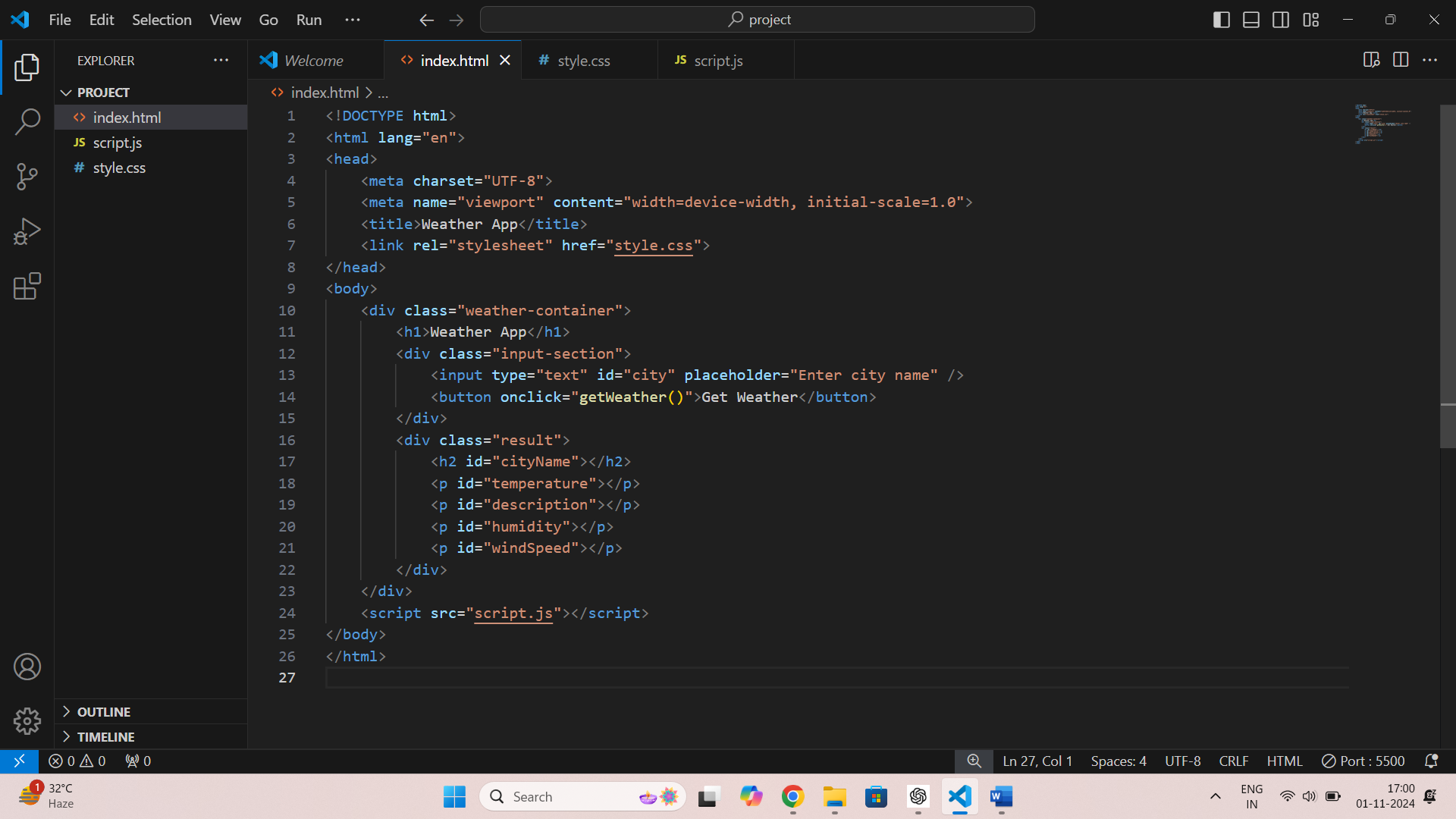
**7. Real-Time Updates**

* Each time the user enters a new city name and clicks the button, the app repeats steps 3–6, fetching and displaying the updated weather information without reloading the page.

SCREENSHOT OF CODE.

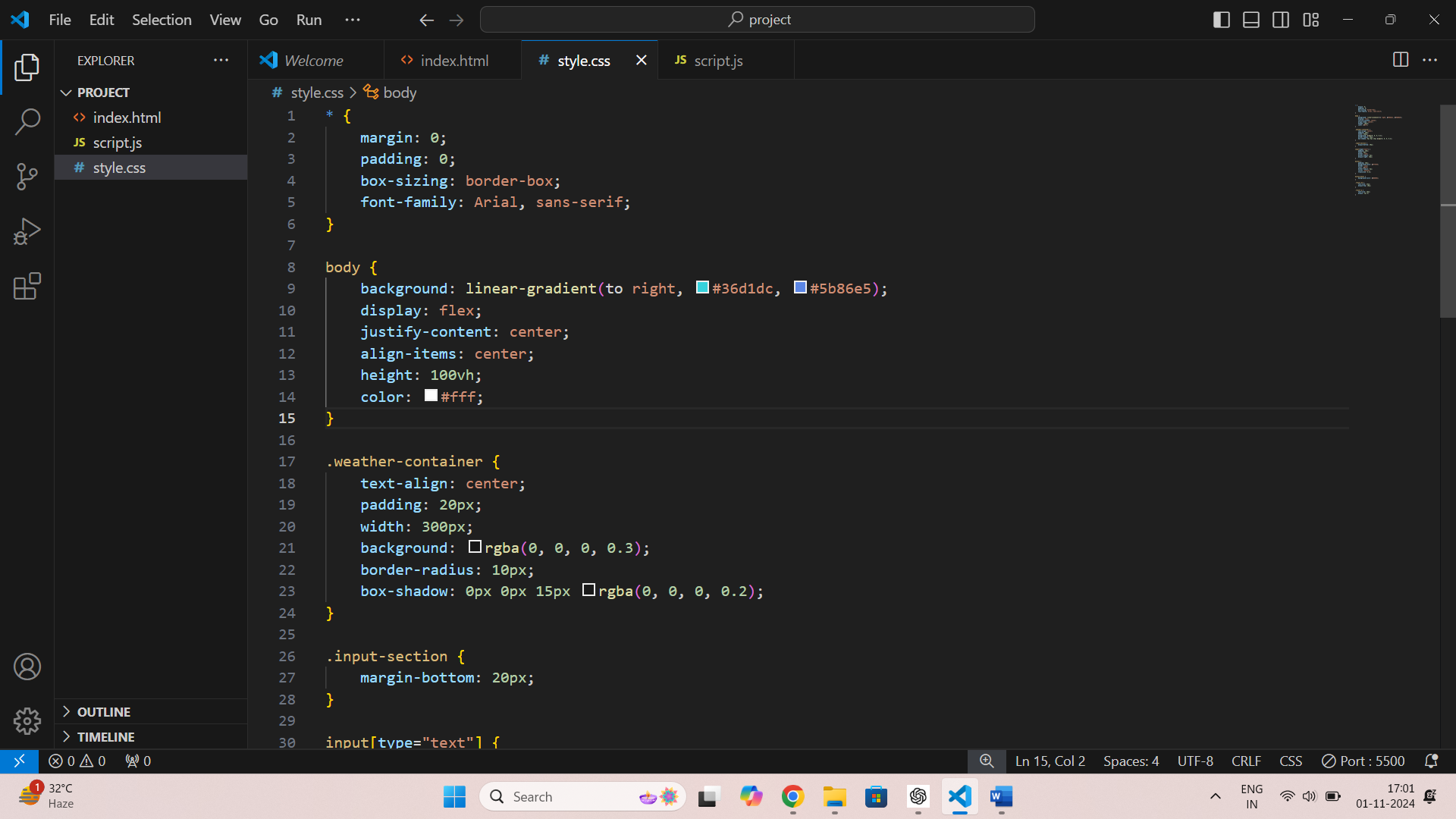
PAGE-9

INDEX.HTML



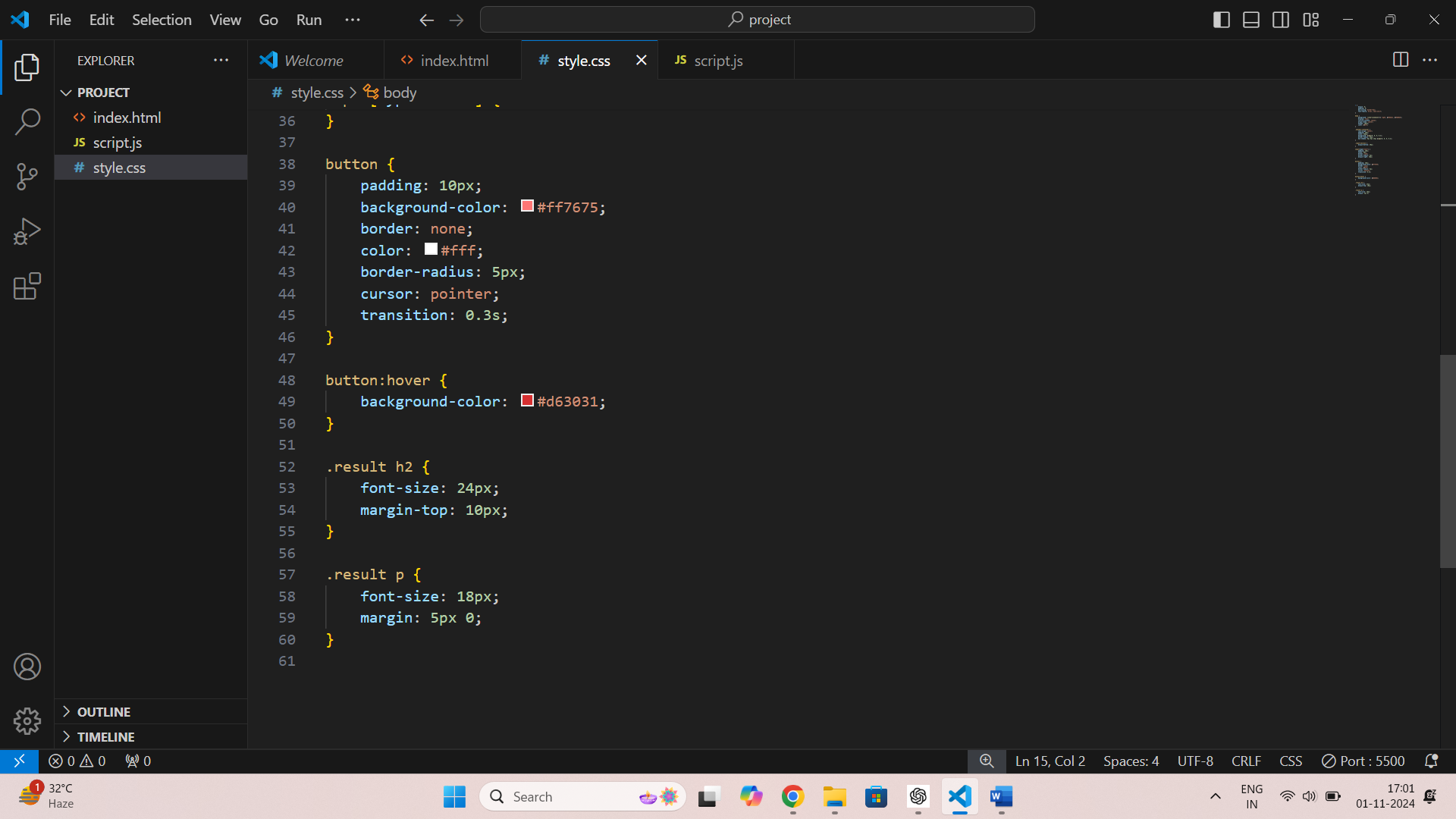
PAGE-10

# STYLE.CSS

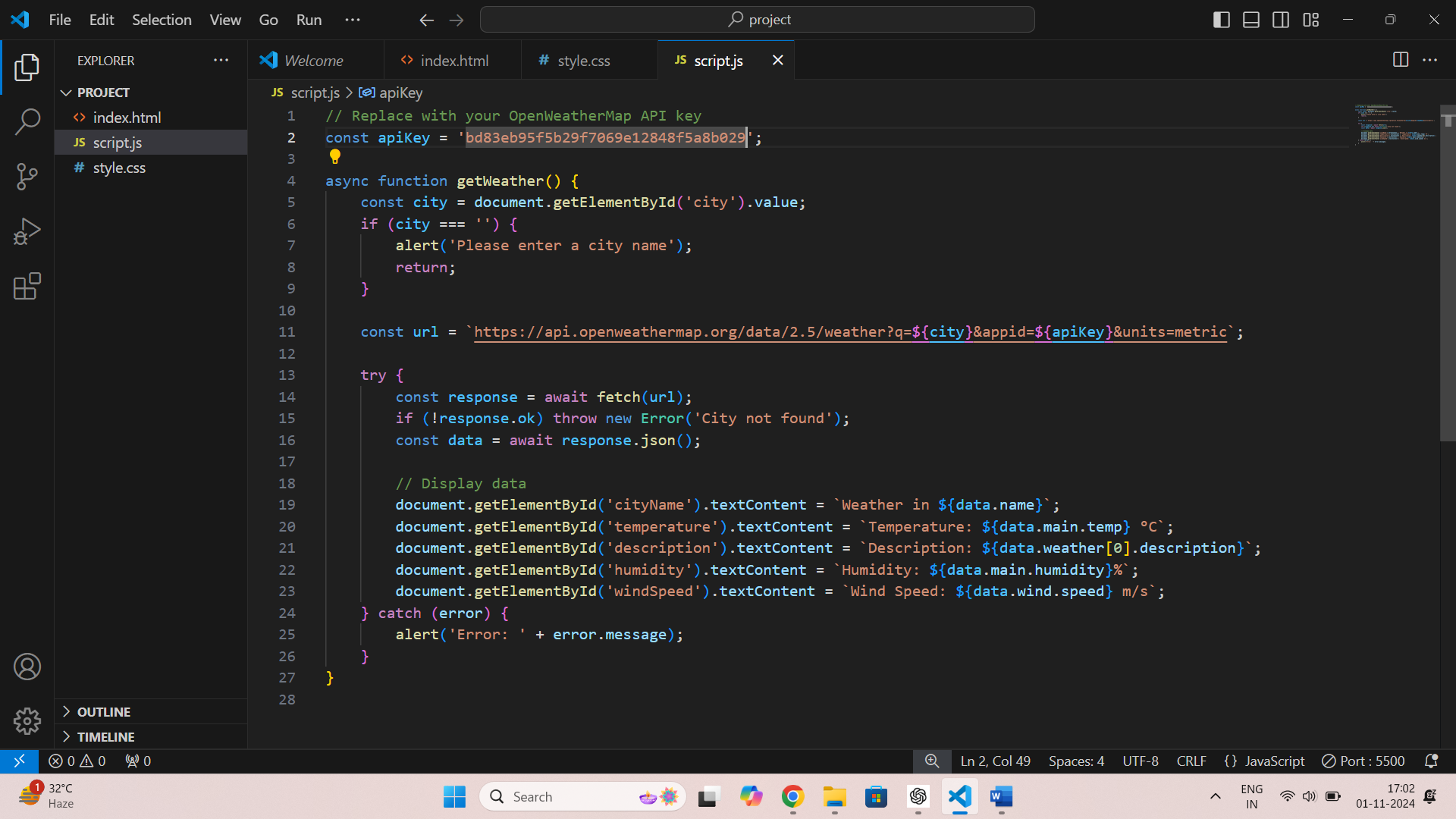
 PAGE-11

# 

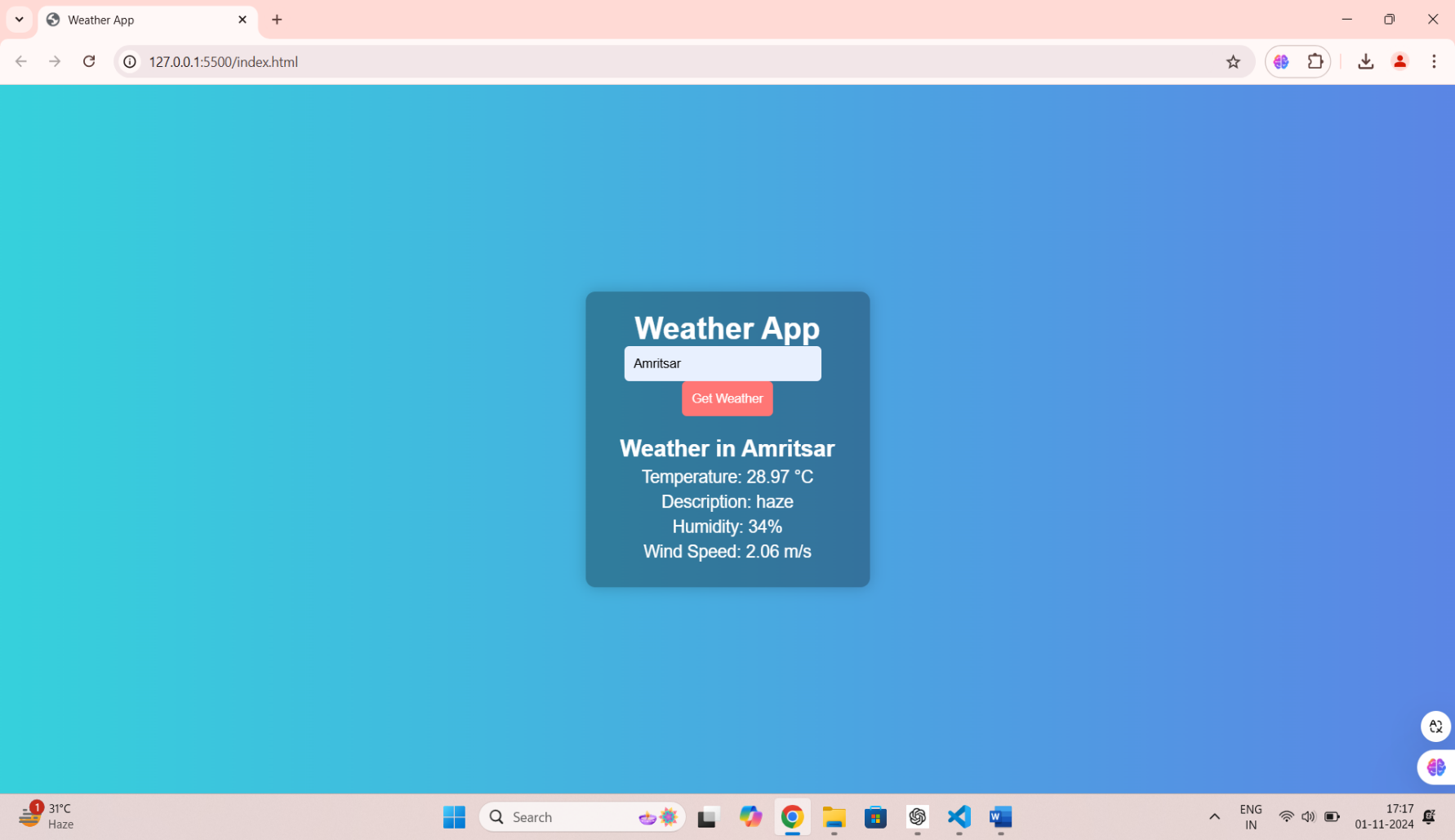
PAGE-12



PAGE-13

SCRIPT.JS

PAGE-14



PAGE-15

CONCLUSION:

In summary, the weather app developed provides users with accurate, real-time weather updates, forecasts, and alerts, enhancing their ability to plan daily activities effectively. By leveraging reliable weather APIs and a user-friendly interface, the app offers a seamless experience for users to access vital information quickly.

The app's features, such as location-based forecasts, customizable notifications, and an intuitive design, contribute to its usability and appeal. Furthermore, the incorporation of visual elements, such as interactive maps and graphs, enriches user engagement.

Looking ahead, future enhancements could include integrating advanced features such as machine learning algorithms for personalized forecasts, support for additional languages, and offline access to weather data. Additionally, expanding the app’s reach to incorporate climate change information and environmental alerts could further enhance its value to users.

Overall, this weather app stands as a significant tool for navigating daily weather challenges, with the potential for continuous improvement and adaptation to user needs.

PAGE-16

ADDRESS OF THE WEBPAGE:

CLICK ON THE LINK BELOW TO VISIT THE WEBSITE

THANK YOU