**Title: Weather App Project**

**Project Overview:**

The weather app creation project aims to users by providing easy access to weather information for any city worldwide. The user experience involves, entering a city name into an input field and initiating the retrieval of weather data by clicking a button or pressing enter. By using JavaScript, the app connects to an open weather API, enabling the weather data specific to the user's specified city. The representation of weather conditions involves the utilization of icons or images such as a sun or a cloud, showing visual understanding of the current weather. In handling errors while fetching data from the weather API, the 'try and catch' technique is used to ensure smooth functionality. A Time Lapse background video is added, enhancing the overall visual experience of the application. The CSS features are optimized to create attractive buttons.

**Objectives:**

List the specific objectives or milestones you aimed to achieve during the internship.

**Technologies Used:**

* **HTML**: Markup language for structuring the web page.
* **CSS**: Stylesheet language for UI design and layout.
* **JavaScript**: Programming language for interaction and API integration.

**Hosting:**

* Infinity Free: Platform to host the app online.

**Methodology:**

Explain the approach you took to accomplish the project goals. Careful consideration of available data and simplicity of integration, an appropriate weather API was chosen. Structured HTML components were used in the implementation, along with JavaScript features for data retrieval and user interactions, and 'try and catch' blocks were used to strengthen the code's error management. A significant amount of effort was devoted to improving the app's visual attractiveness and usability. This was accomplished by optimizing the CSS, adding weather symbols for visual aids, and making sure the app would work on a variety of devices.

**Implementation Details:**

1. **Features and Functionality:**

**Input Interface:** Users can input the name of a city into an input field. Retrieval of weather data occurs upon clicking a designated button or pressing the 'Get Weather' key.

**Weather Data Display:** Display of real-time weather information for the specified city. Presentation of temperature data, typically in Celsius. Showcase of humidity levels in percentage. Indication of the speed of wind in meters per second.

**Visual Representation:** Utilization of weather icons or images (e.g., sun, cloud) to visually depict the prevailing weather conditions.

**Error Handling:** Implementation of 'try and catch' blocks to manage potential errors during the process of fetching data from the weather API. Display of informative messages in case of unsuccessful data retrieval or API errors.

**2. Codebase Overview:**

Give an overview of the codebase structure, organization, and key modules.

**HTML:** The main HTML file containing the structure of the weather app. Structured elements for input fields, buttons, weather data display, and navigation bar .

**CSS Styling:** Contains the CSS styles for the entire app, including layout, colors, fonts, buttons, and responsiveness.

**JavaScript Functionality:** The primary JavaScript file responsible for the app's functionalities. Handles user interactions like button clicks or 'Enter' key press for weather data retrieval. Functions to connect to the open weather API, fetch weather data, and update the UI dynamically. Functions to dynamically update HTML elements with weather information fetched from the API.

**3. Challenges Faced:**

We faced challenges with adjusting the background time lapse video and to show the error message when the data entered is incorrect. We addressed them using position fixed and z-index, adjusting the order of video to be displayed on the application. Using ‘throw’ exception we displayed the error box as City not found, please enter valid data.

**4. Results and Achievements:**

Present the outcomes of the project, showcasing any quantitative or qualitative results. The weather app project provide users with access to real-time weather data for any city worldwide. With successful API integration, the app displayed temperature, humidity, wind speed, and weather conditions specific to user-input cities. Visual representation of weather conditions via icons enhanced the app's usability, while CSS improved the interface's design. Efficient error handling, implemented through 'try and catch' blocks, ensured uninterrupted functionality, supported by informative error messages for users. We achieved the quality and functionality, resulting in a successful application of a valuable and impactful project.

**Conclusion:**

The development of a functional and user-friendly application, providing access to real-time weather data for global cities, stands as a significant accomplishment. This project has not only delivered a valuable tool but also served as a platform for personal growth and learning, encompassing new technical skills and an understanding of effective project management. Overall, the weather app project marks a successful journey in delivering a functional, aesthetically pleasing, and informative application, contributing positively to both user needs and personal development.

**Appendix:**

**1. Code Snippets:**

**HTML:**

<!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" type="text/css" href="style.css">

</head>

<body>

<video autoplay muted loop>

<source src="https://cdn.pixabay.com/vimeo/256955049/saigon-14385.mp4?width=1280&hash=3e5f8905ae9db3698a79a23e073f7818717a8b4d" type="video/mp4">

</video>

<div class="container">

<h1>Weather App</h1>

<div class="d-flex justify-content-row">

<input type="text" id="locationInput" placeholder="Enter a city">

<button id="searchButton">Get Weather</button>

</div>

<div id="weather-info" class="d-flex flex-column align-items-center">

<img id="weather-icon">

<p id="temperature"></p>

<p id="weather-description"></p>

<div class="additional-info">

<p id="feelsLike"></p>

<p id="humidity"></p>

<p id="windSpeed"></p>

</div>

</div>

</div>

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

</body>

</html>

**CSS Styling:**

body {

margin: 0;

padding: 0;

overflow: hidden;

}

video {

position: fixed;

top: 0;

left: 0;

min-width: 100%;

min-height: 100%;

z-index: -1;

}

.container {

max-width: 500px;

margin: 0 auto;

text-align: center;

padding: 20px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);

margin-top: 50px;

background-color: rgba(255, 255, 255, 0.5);

}

h1 {

font-size: 40px;

font-family: Monotype Corsiva;

}

input[type="text"] {

width: 90%;

padding: 10px;

margin: 10px 0;

border: 3px solid #ccc;

border-radius: 3px;

}

button {

background-color: #007BFF;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 10px;

cursor: pointer;

margin: 10px 10px 10px 10px;

}

.weather-container {

margin-top: 20px;

display: flex;

justify-content: center;

align-items: center;

flex-direction: column;

}

.additional-info {

display: flex;

justify-content: space-around;

margin-top: 10px;

}

.box-container {

padding: 10px;

border: 2px solid #ccc;

border-radius: 5px;

width: 100px;

height: 100px;

box-sizing: border-box;

}

#temperature,#weather-description{

font-size: 25px;

font-weight: bold;

font-family: Monotype Corsiva;

}

#feelsLike,

#humidity,

#windSpeed {

font-size: 20px;

font-weight: bold;

font-family: Monotype Corsiva;

}

**JavaScript:**

const cityInput = document.getElementById('locationInput');

const searchButton = document.getElementById('searchButton');

const temperatureContainer = document.getElementById('temperature');

const weatherDescriptionContainer = document.getElementById('weather-description');

const weatherIcon = document.getElementById('weather-icon');

const feelsLikeContainer = document.getElementById('feelsLike');

const humidityContainer = document.getElementById('humidity');

const windSpeedContainer = document.getElementById('windSpeed');

const apiKey = '72d217a137d58992bdcacab92cb8d82f';

searchButton.addEventListener('click', getWeather);

async function getWeather() {

const city = cityInput.value;

try {

const response = await fetch(https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiKey});

if (!response.ok) {

if (response.status === 404) {

throw new Error(City not found. Please enter a valid city name.);

} else {

throw new Error(Failed to fetch weather data. Status: ${response.status});

}

}

const data = await response.json();

const humidity = data.main.humidity;

const windSpeed = data.wind.speed;

const feelsLike = Math.round(data.main.feels\_like - 273.15);

const temperature = Math.round(data.main.temp - 273.15);

weatherIcon.src = https://openweathermap.org/img/wn/${data.weather[0].icon}.png;

temperatureContainer.textContent = Temperature: ${temperature}°C;

weatherDescriptionContainer.textContent = data.weather[0].description;

feelsLikeContainer.textContent = Feels Like: ${feelsLike}°C;

humidityContainer.textContent = Humidity: ${humidity}%;

windSpeedContainer.textContent = Wind Speed: ${windSpeed} m/s;

} catch (error) {

console.error('Error fetching weather data:', error.message || error);

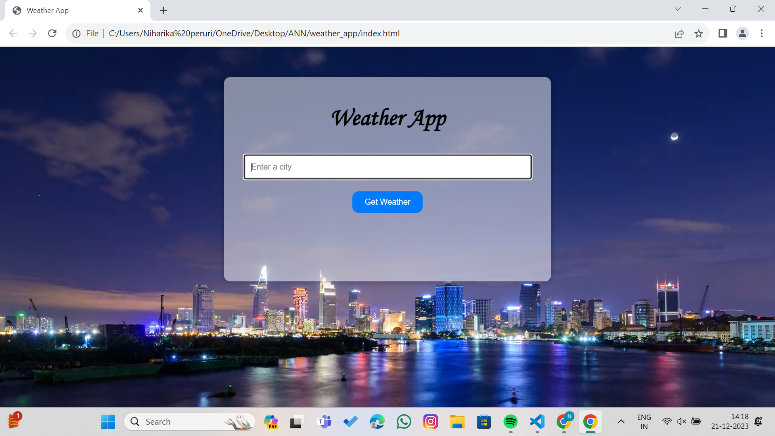
console.error('Error stack trace:', error.stack);

alert(error.message || 'An error occurred. Please try again.');

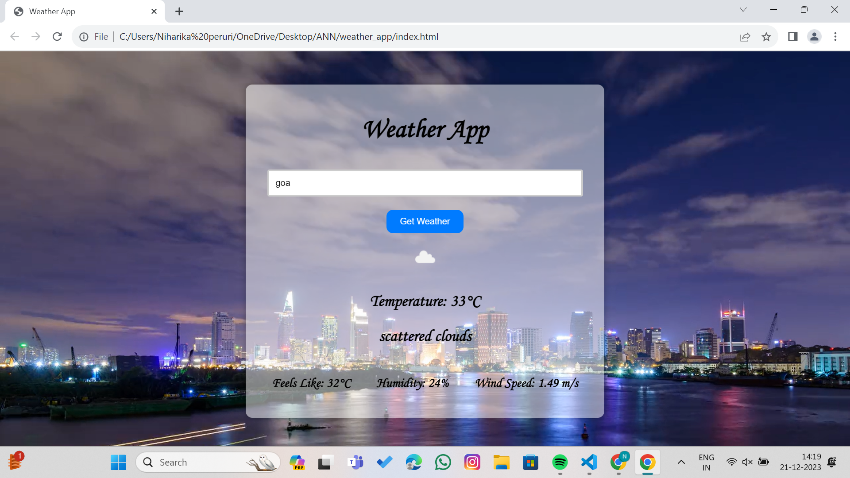
}

}

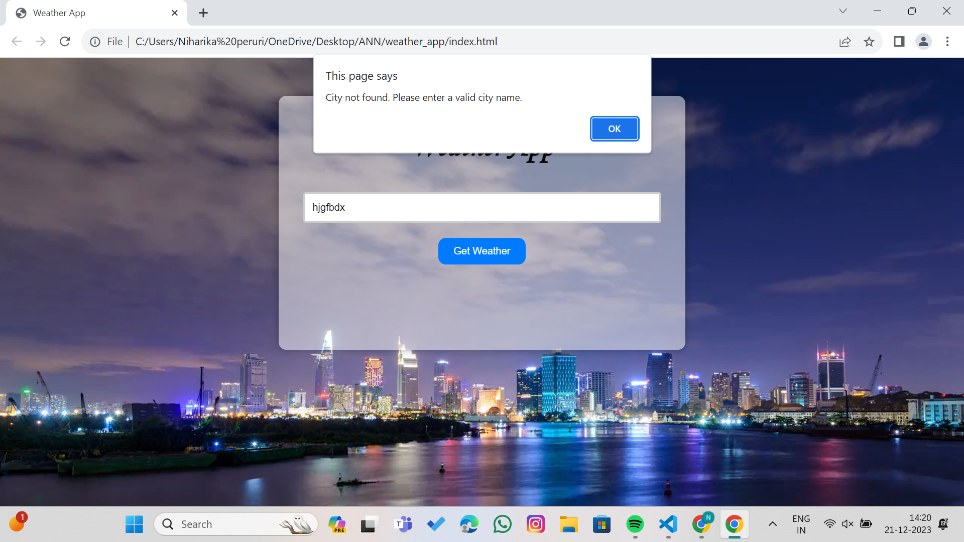
**2. Screenshots/Demo:**



*img.1:Homepage*



*img.2: Weather Description*



*img.3: Error Message*

*Appilication Link:* <weather-app-web.infinityfreeapp.com>