**Project Based Learning Report on**

**WEATHER FORECASTING APPLICATION**

Submitted to

Department of Data Science

Bachelor of Technology

in

CSE-DS

(2022-2026)

T.SWATHI (22R11A6737)

K.DIVYA (22R11A6726)

D.VYBHAVI (22R11A6711)

Under the Guidance of :

# Ms. M.V. LAVANYA, Asst. Professor,EA,GCET



Department of CSE-DS

Accredited by NBA

**Geethanjali College of Engineering and Technology (UGC Autonomous)** (Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi) Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.No | Index | Page No. |
| 1. | ABSTRACT | 3 |
| 2. | INTRODUCTION | 4 |
| 3. | OBJECTIVE | 5 |
| 4. | SOFTWARE & HARDWARE REQUIREMENTS | 7 |
| 5. | MODULES | 8 |
| 6. | CODE | 11 |
| 7. | OUTPUT | 23 |
| 8. | CONCLUSION | 26 |
| 9. | REFERENCES | 27 |

# ABSTRACT

Weather forecasting application is web based application through which you will get all the reports related to weather forecasting of any locations .Its geographical locator which will be received through your browser setting and server configuration will automatically identify the location and able to present its weather details such as temperature ,direction of wind ,rains ,humidity ,etc. To change the location you need to select the options provided below to gets its details .Its new avatars and feed burner will also allow its users to get the weather reports directly to their mail, when they were not able to access their particular domain or even when the server is down. Its user’s friendly tools are simple to use ,that even child can handle it and get information on particular geographical area.

## 2. INTRODUCTION

An introduction for a weather forecasting application could focus on its ability to empower users with up-to-date and accurate weather information. It might highlight the app's features, such as real-time updates, comprehensive forecasts, and user-friendly interfaces. Additionally, the introduction could emphasize the app's reliability, the sources of its data (such as meteorological stations or satellite information), and how it simplifies decision-making by providing detailed weather predictions. Some introductions might also touch on additional features like interactive maps, personalized notifications, or severe weather alerts, aiming to showcase the app's convenience and usefulness in daily life.

# 3. OBJECTIVE

The objective of a weather forecasting application typically revolves around providing users with reliable, accurate, and easily accessible weather information. Some key objectives of such an application could include:

Accuracy: Offering precise and reliable weather forecasts by amalgamating data from multiple sources and employing advanced prediction models to enhance the reliability of forecasts.

User-Friendly Interface: Designing an intuitive and easy-to-navigate interface for users to access weather information effortlessly. This involves presenting forecasts in a clear and understandable manner, catering to users of varying technical expertise.

Real-Time Updates: Providing real-time updates and notifications to keep users informed about sudden weather changes or critical alerts, ensuring they can plan and adapt accordingly.

Customization and Personalization: Allowing users to personalize their weather experience by selecting favorite locations, setting specific weather alerts, or accessing detailed forecasts for activities like outdoor sports, travel, or work.

Severe Weather Alerts: Issuing warnings or alerts for severe weather conditions like storms, hurricanes, or heavy rainfall to help users take necessary precautions and ensure safety.

Reliable Data Sources: Using credible and diverse sources of weather data, such as meteorological stations, satellites, or weather models, to ensure the accuracy and comprehensiveness of forecasts.

Community Engagement: Encouraging user feedback and engagement to continuously improve the application's features and functionalities, thereby meeting the evolving needs of its user base.

Overall, the primary objective is to empower users with timely, accurate, and easily understandable weather information that aids in planning and decision-making in various aspects of their lives.

1. **SOFTWARE & HARDWARE REQUIREMENTS**

**Software Requirements:**

* Operating System linux , Windows
* Technology HTML , CSS, Javascript

**Hardware Requirements**:

* Processor-2.0GHZ
* Ram 2GB
* Hard Disk Drive 160GB

# MODULES

* User Authentication
* User Profile
* Saved Locations
* Personalized Weather
* Weather Condition
* Notifications
* Security
* Feedback Submission
* Logout Functionality

**USER AUTHENTICATION** **:**

* Login Screen : Collects user credentials (username/password or alternative authentication methods).
* Registration Screen : Allows new users to create account.

**USER PROFILE :**

* Profile Page : Displays user details and preferences.
* Profile Editing : Enables users to update their profile information.

**SAVED LOCATIONS :**

* Favorite Locations : Allows users to save and manage their preferred locations.
* Recent Searches : Displays a history to recently searched locations.

**PERSONALIZED WEATHER :**

* Customized Dashboard : Tailors the main weather screen based on user preferences.
* Default Location : Sets a default location for quick access.

**WEATHER CONDITION :**

* Current Conditions : Displays temperature , humidity , wind , speed etc…
* Daily Forecast : Presents extended daily forecasts.
* Hourly Forecast : Shows hourly weather predictions.

**NOTIFICATIONS :**

* Customized Alerts : Lets users set personalized weather alerts based on their preferences.
* Notification History : Shows a log of past notifications

**SECURITY :**

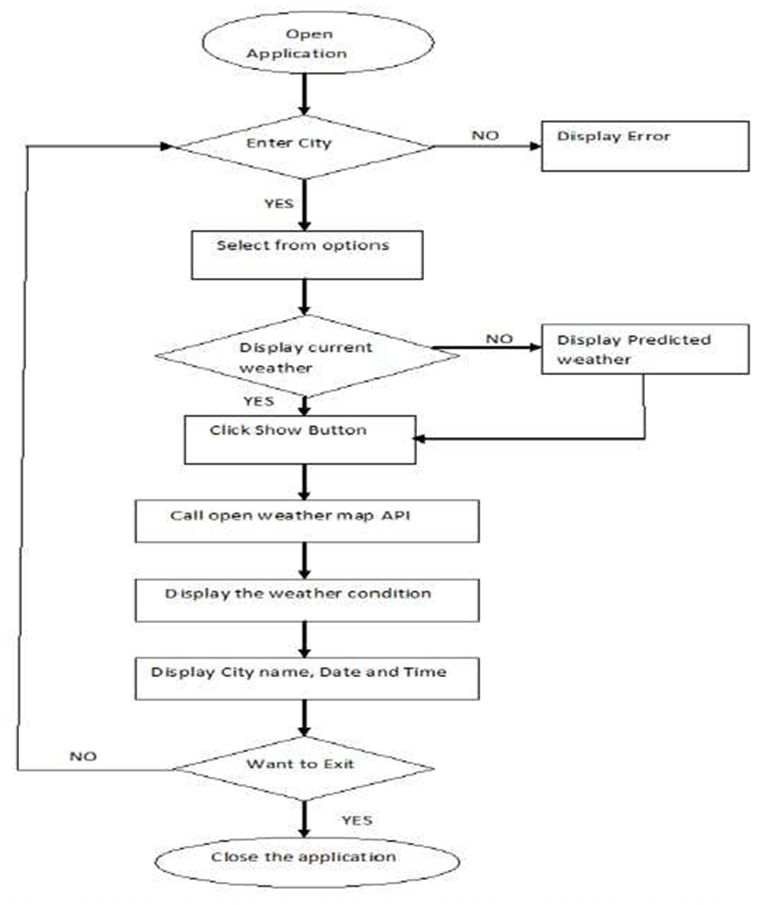
* Two-Factor Authentication (2FA) : Enhances security with an additional layer of authentication.
* Password Recovery : Provides a way for users to recover their account in case of forgotten passwords.

**FEEDBACK SUBMISSION :**

Allows users to submit feedback directly from their profile.

**LOGOUT FUNCTIONALITY:**

Logout Button: Allows users to securely log out of their accounts



**6.CODE**

**weather.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Weather Forecast</title>

<link rel="stylesheet" href="styleweather.css">

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

</head>

<body>

<h1>WEATHER FORECAST</h1>

<div class="container">

<div class="weather-input">

<h3>Enter City Name</h3>

<input class="city-input" type="text" placeholder="E.g Hyderabad,Vijayawada">

<button class="search-btn">Search</button>

<div class="separator"></div>

<button class="location-btn">Use Current Location</button>

</div>

<div class="weather-data">

<div class="current-weather">

<div class="details">

<h2>\_\_(\_\_\_\_)</h2>

<h4>Temp: \_\_\_c</h4>

<h4>Wind: \_\_ M/s</h4>

<h4>Humidity: \_%</h4>

</div>

</div>

<div class="days-forecast">

<h2>5-Day Forecast</h2>

<ul class="weather-cards">

<!-- Cards will be populated by JavaScript -->

</ul>

</div>

</div>

</div>

</body>

</html>

**styleweather.css**

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Open Sans', sans-serif;

}

body {

background: #E3F2FD;

}

h1 {

color: #fff;

padding: 18px 0;

text-align: center;

font-size: 1.75rem;

background: #5372F0;

}

.container {

display: flex;

gap: 35px;

padding: 30px;

}

.weather-input input {

width: 100%;

height: 46px;

outline: none;

padding: 0 17px;

margin: 10px 0 20px 0;

font-size: 1.07rem;

border-radius: 4px;

border: 1px solid;

}

.weather-input .separator {

height: 1px;

width: 100%;

background: #bbb;

margin: 25px 0;

display: flex;

align-items: center;

justify-content: center;

}

.weather-input .separator::before {

content: "or";

color: #6C757D;

font-size: 1.18rem;

padding: 0 15px;

margin-top: -4px;

background: #E3F2FD;

}

.weather-input button {

width: 100%;

padding: 10px 0;

cursor: pointer;

outline: none;

border: none;

color: #fff;

font-size: 1rem;

border-radius: 4px;

background: #5372F0;

}

.weather-input .location-btn {

background: #6C757D;

}

.weather-data {

width: 100%;

}

.current-weather {

color: #fff;

display: flex;

border-radius: 5px;

padding: 20px 70px 20px 20px;

background: #5372F0;

justify-content: space-between;

}

.current-weather h2 {

font-size: 1.7rem;

}

.weather-data h4 {

margin-top: 12px;

font-size: 1rem;

font-weight: 500;

}

.current-weather .icon {

text-align: center;

}

.current-weather .icon img {

max-width: 120px;

margin-top: -15px;

}

.current-weather .icon h6 {

margin-top: -10px;

text-transform: capitalize;

}

.days-forecast h2 {

font-size: 1.5rem;

margin: 20px 0;

}

.weather-cards {

display: flex;

gap: 20px;

}

.weather-cards .card {

list-style: none;

color: #fff;

padding: 18px 16px;

border-radius: 5px;

background: #6C757D;

width: calc(100% / 5);

}

.weather-cards .card img {

max-width: 70px;

margin: 5px 0 -12px 0;

}

**scriptweather.js**

const cityInput = document.querySelector(".city-input");

const searchButton = document.querySelector(".search-btn");

const locationButton = document.querySelector(".location-btn");

const currentWeatherDiv = document.querySelector(".current-weather");

const weatherCardsDiv = document.querySelector(".weather-cards");

const API\_KEY = "18f0d725ca6bb3fa6415390695a612cc";

const createWeatherCard = (cityName, weatherItem, index) => {

const date = weatherItem.dt\_txt.split(" ")[0];

const iconUrl = https://openweathermap.org/img/wn/${weatherItem.weather[0].icon}@4x.png;

if (index === 0) {

return `

<div class="details">

<h2>${cityName} (${date})</h2>

<h4>Temperature: ${(weatherItem.main.temp - 273.15).toFixed(2)}°C</h4>

<h4>Wind: ${weatherItem.wind.speed} M/s</h4>

<h4>Humidity: ${weatherItem.main.humidity}%</h4>

</div>

<div class="icon">

<img src="${iconUrl}" alt="weather-icon">

<h4>${weatherItem.weather[0].description}</h4>

</div>`;

} else {

const iconUrl = https://openweathermap.org/img/wn/${weatherItem.weather[0].icon}@2x.png;

return `

<li class="card">

<h3>${date}</h3>

<img src="${iconUrl}" alt="weather-icon">

<h4>Temperature: ${(weatherItem.main.temp - 273.15).toFixed(2)}°C</h4>

<h4>Wind: ${weatherItem.wind.speed} M/s</h4>

<h4>Humidity: ${weatherItem.main.humidity}%</h4>

</li>`;

}

};

const getWeatherDetails = (cityName, lat, lon) => {

const WEATHER\_API\_URL = https://api.openweathermap.org/data/2.5/forecast?lat=${lat}&lon=${lon}&appid=${API\_KEY};

fetch(WEATHER\_API\_URL)

.then(res => res.json())

.then(data => {

const uniqueForecastDays = [];

const fiveDaysForecast = data.list.filter(forecast => {

const forecastDate = new Date(forecast.dt\_txt).getDate();

if (!uniqueForecastDays.includes(forecastDate)) {

uniqueForecastDays.push(forecastDate);

return true;

}

return false;

});

cityInput.value = "";

weatherCardsDiv.innerHTML = "";

currentWeatherDiv.innerHTML = "";

fiveDaysForecast.forEach((weatherItem, index) => {

if (index === 0) {

currentWeatherDiv.innerHTML = createWeatherCard(cityName, weatherItem, index);

} else {

weatherCardsDiv.insertAdjacentHTML("beforeend", createWeatherCard(cityName, weatherItem, index));

}

});

})

.catch(() => {

alert("An error occurred while fetching the weather details!");

});

};

const getCityCoordinates = () => {

const cityName = cityInput.value.trim();

if (!cityName) return;

const GEOCODING\_API\_URL = http://api.openweathermap.org/geo/1.0/direct?q=${cityName}&limit=1&appid=${API\_KEY};

fetch(GEOCODING\_API\_URL)

.then(res => res.json())

.then(data => {

if (!data.length) return alert(No coordinates found for ${cityName});

const { name, lat, lon } = data[0];

getWeatherDetails(name, lat, lon);

})

.catch(() => {

alert("An error occurred while fetching the coordinates!");

});

};

const getUserCoordinates = () => {

navigator.geolocation.getCurrentPosition(

position => {

const { latitude, longitude } = position.coords;

const REVERSE\_GEOCODING\_URL = http://api.openweathermap.org/geo/1.0/reverse?lat=${latitude}&lon=${longitude}&limit=1&appid=${API\_KEY};

fetch(REVERSE\_GEOCODING\_URL)

.then(res => res.json())

.then(data => {

const { name, lat, lon } = data[0];

getWeatherDetails(name, latitude, longitude);

})

.catch(() => {

alert("An error occurred while fetching the coordinates!");

});

},

error => {

if (error.code == error.PERMISSION\_DENIED) {

alert("Geolocation request denied. Please reset location permission to grant access again.");

}

}

);

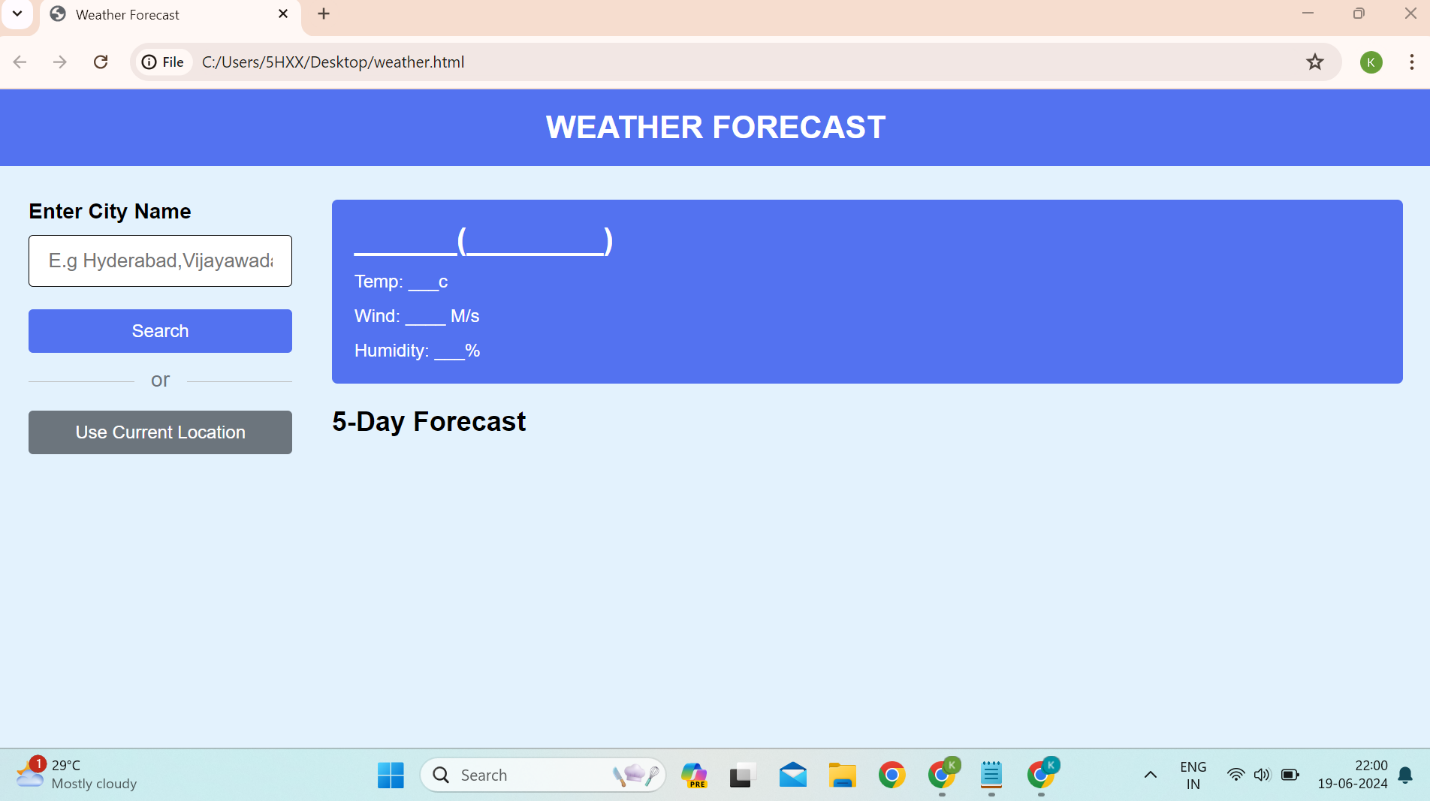
};

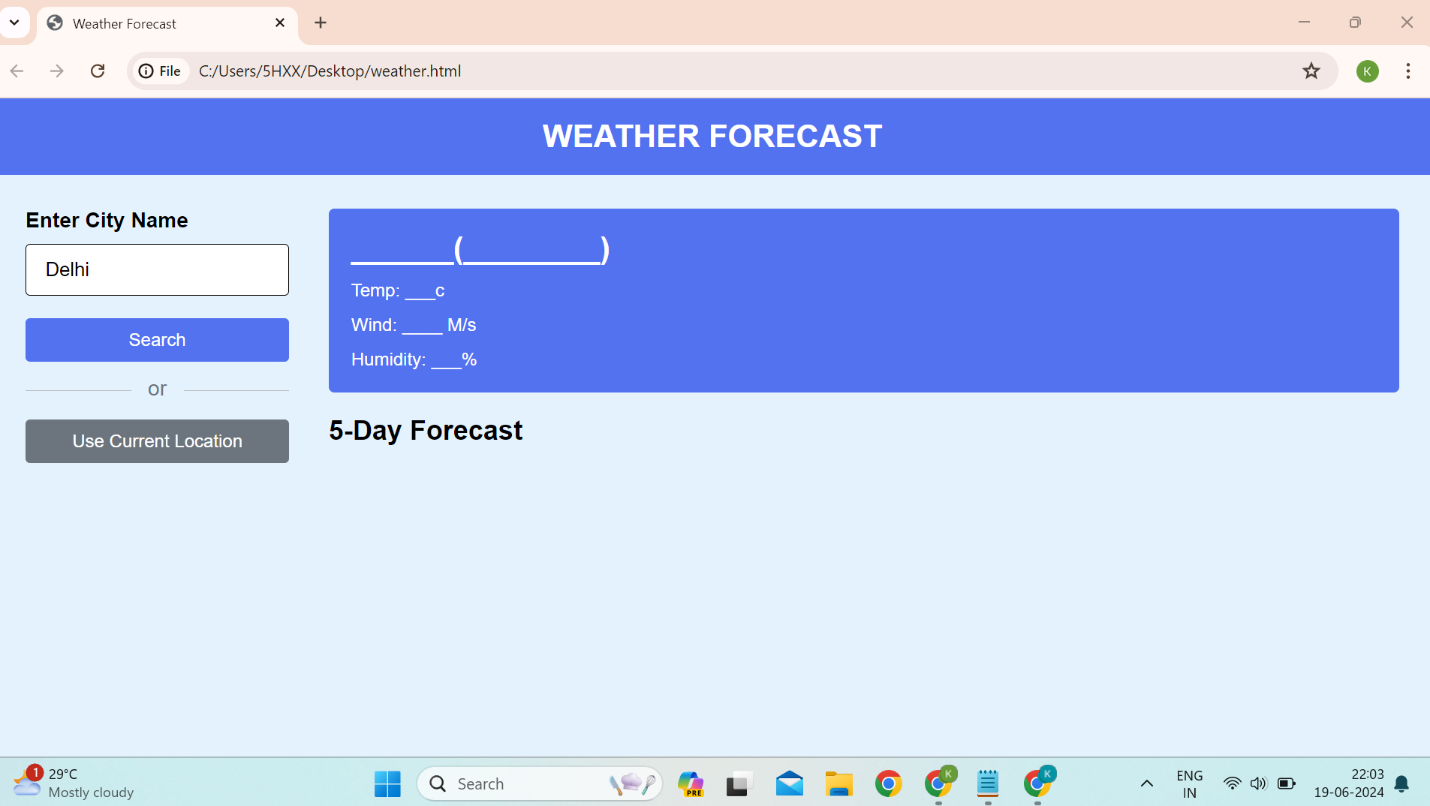
locationButton.addEventListener("click", getUserCoordinates);

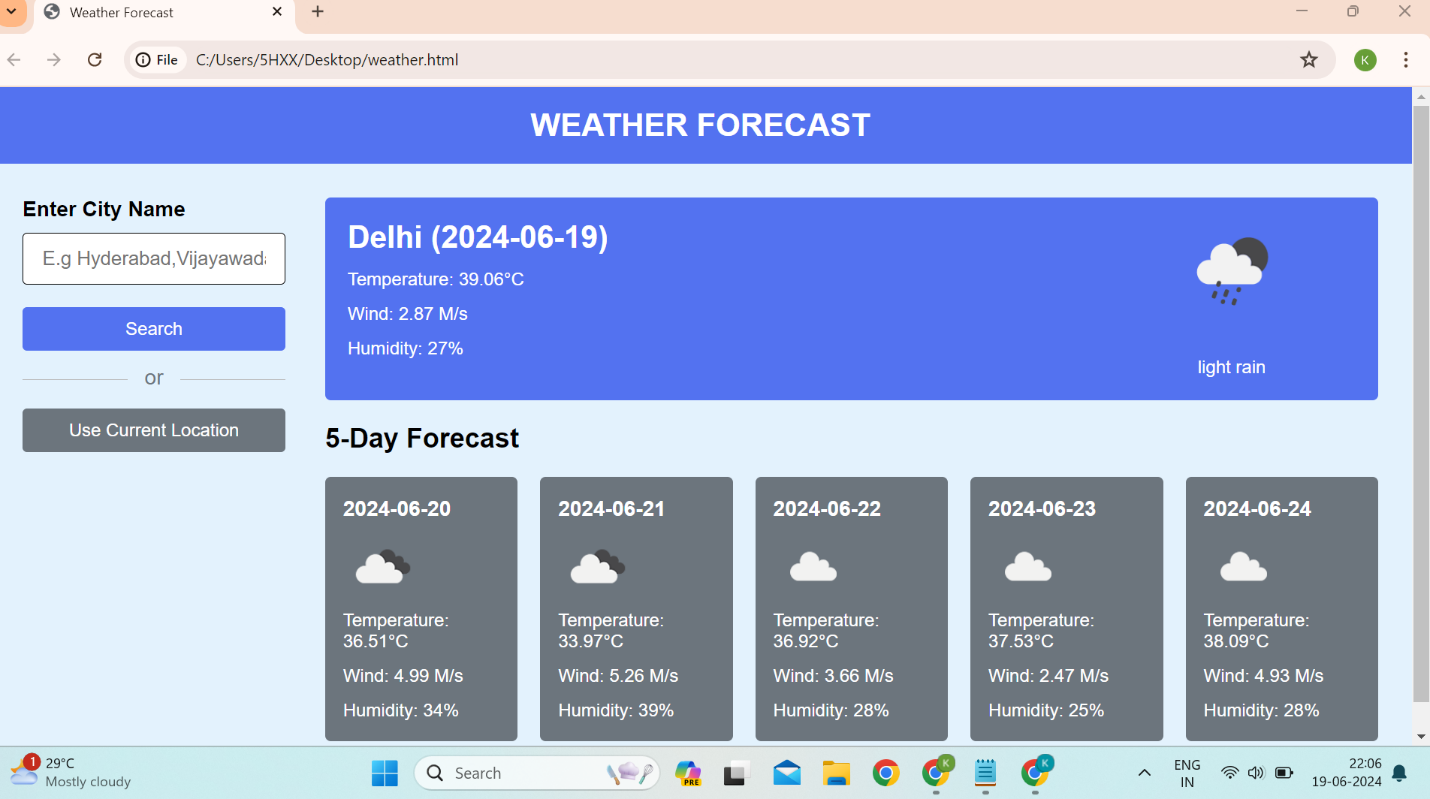
searchButton.addEventListener("click", getCityCoordinates);

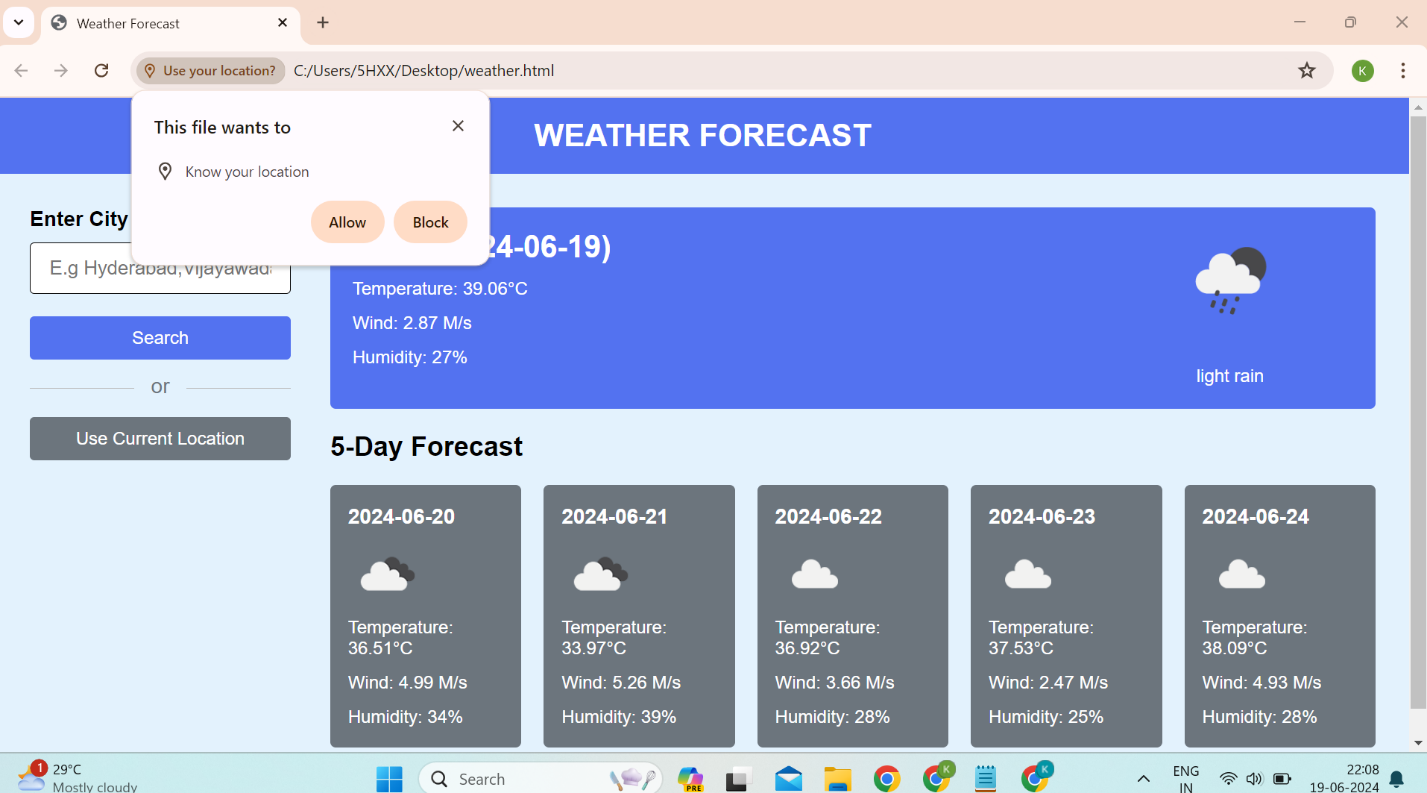
cityInput.addEventListener("keyup", e => e.key == "Enter" && getCityCoordinates());

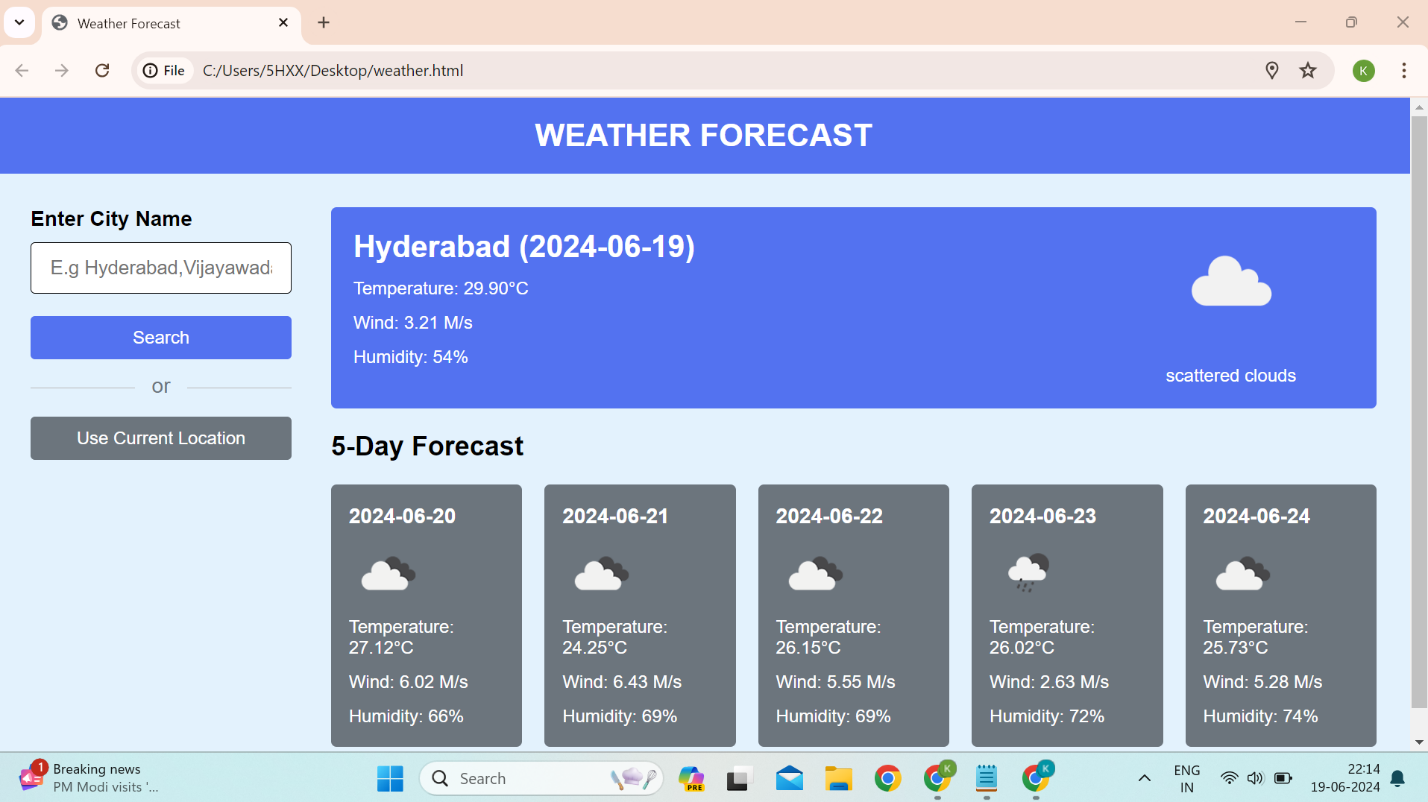
**7.OUTPUT:**











# CONCLUSION

# In conclusion, a weather forecasting application serves as a valuable tool for users to access accurate, real-time weather information. Through integration with APIs like Open Weather Map or others, these applications provide a range of functionalities:

# 1. Real-Time Updates: Users can access current weather conditions, forecasts, and precipitation probabilities for specific locations.

# 2. User-Friendly Interface: Intuitive interfaces present weather data in a clear and understandable manner, catering to users with varying technical expertise.

# 3. Customization: Users can personalize their experience by selecting favorite locations, setting alerts, or accessing detailed forecasts for specific activities.

# 4. Safety Alerts: Issuing warnings for severe weather conditions helps users take necessary precautions and ensures their safety.

# 5. Reliable Data Sources: By aggregating data from meteorological stations, satellites, and weather models, these apps aim to provide accurate and comprehensive forecasts.

# Developers creating weather forecasting applications need to consider user needs, design intuitive interfaces, ensure data accuracy, and potentially incorporate additional features like maps, historical data, or user community engagement.

# Ultimately, such applications aim to empower users with the knowledge needed to plan activities, make informed decisions, and stay safe in various weather conditions

# REFERENCES

Access to link resource at:-

* [www.wikipedia.com](http://www.wikipedia.com/)
* [www.google.com](http://www.google.com/)
* [www.netcad.com](http://www.netcad.com/)