Project Work

:- Project Report Format :-

1. Introduction

* Overview

  A brief description about your project

* Purpose

2. Literature Survey

* Existing problem

  Existing approaches or method to solve this problem

* Proposed Solution

  what is the method or solution suggest by you ?

3. Theoritical Analysis

* Block Diagram

  Diagrammatic overview of the project

* Hardware / software Designing

  Hardware and software requirements of the project

4. Result

  Final Findings (output) of the project along with Screenshots

5. Advantages and Disadvantages

List of advantages and disadvantages of the proposed solution

6. Applications

The areas where this solution can be applied

7. Conclusion

conclusion summarizing the entire work and findings.

8. Future Scope

Enhancements that can be made in the future

Project Report Format

Introduction :-

Overview :-

weather App is a one step solution for staying up- to- date with real time weather forecasts

This project is an exciting endevar in "Front-end-Development" aimed at prouding users with a sleek and intutive weather application. Our mission is to deliver an engaging user experience by presenting weather data in a visually appealing and information manner.

purpose :-

weather plays a significant role in our daily use influencing our activities, clothing choices and overall well being people constantly seek accurate weather information to plan their schedule accordingly. while many weather application exist. wheather app stands out by priortizing user experience and simplicity.

The purpose of a weather app project is to create software application that provide users with real- time weather information and forcasts for a specific location or multiple locations This type of app is designed to be accessible on various platforms, such as smartphone. and tablets to help users stay informed about the current weather conditions and plan their activities accordingly.

Objective :-

Real-time weather Data :-

The app should be able to fetch and display current weather conditions, including temperature, humidity and speed. for the users chosen location.

weather Forecasts :-

Providing accurate weather forecasts for the next few days is crucial. as it help users plan ahead for events.

location Based Services :-

The app should be able to determine the user's location or allow them to input a specific location for weather information.

User-friendly interface :-

The app should have an intuitive and visually appealing interface, making it easy for users to understand and navigate

Customisation :-

users may want to customize the app to display weather units in their preffered format (or) choose the time format

weather Alerts :-

The app may include a feature to send weather alerts & notifications to users for severe weather conditions like storms, humicanes or temperatures.

Maps and Rador :-

Including weather maps and rador data can apply help users visulalize weather patterns and tracle storms in real-time.

Energy Efficiency :-

weather app projects may focus on optimizing battery usage, especially for mobile devices.

Integration with APIe :-

The app may utilize third party weather APIs to access the accurate and up-to date weather data.

Cross-platform compatability :-

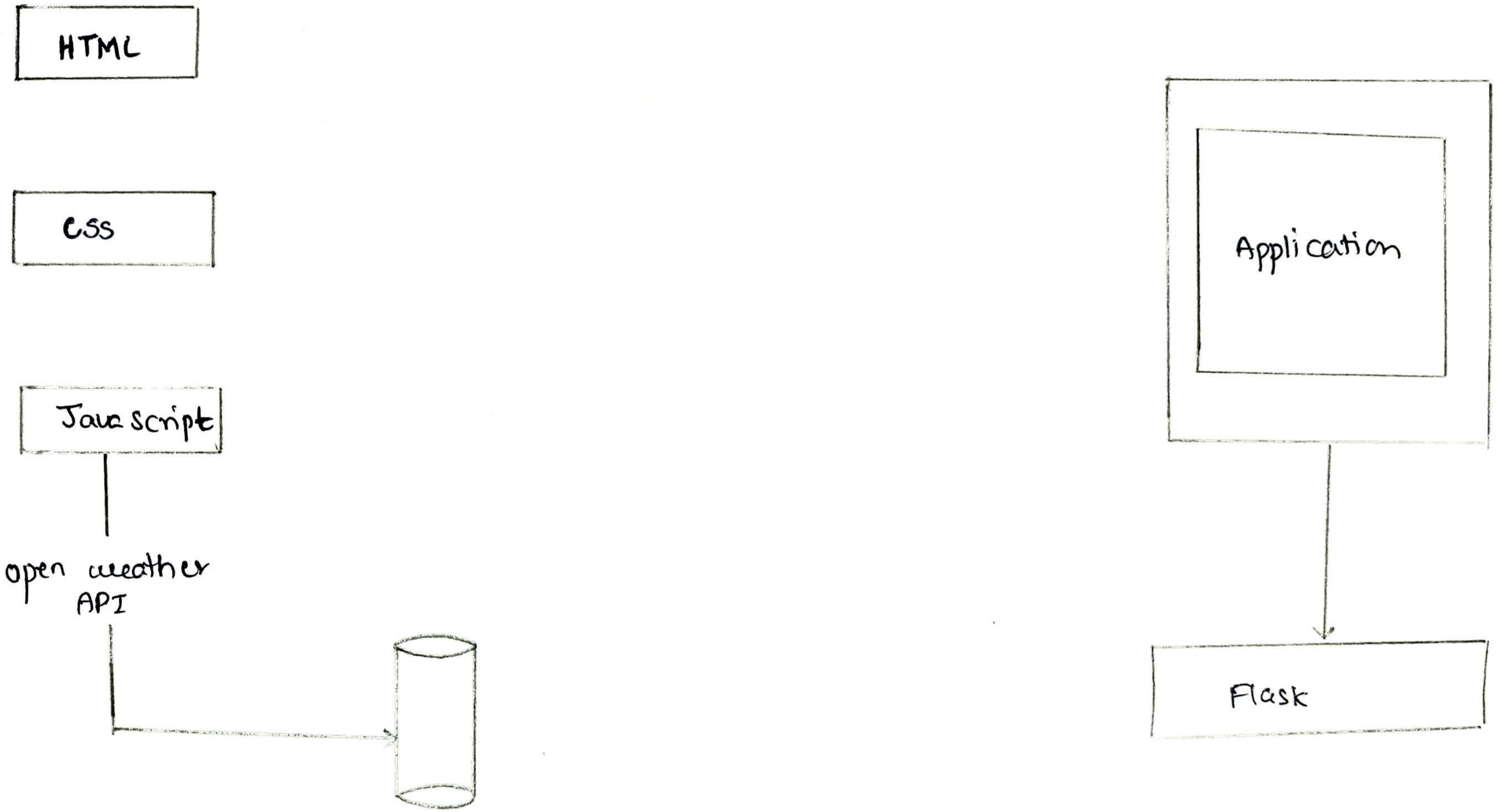
To reacher a broader audience, tha app should should be compatible with different operating systems such as android, ios, windows

Offline Access :-

Although real-time data is essential, the app might consider providing basic weather information even when the device is offline

Overall, the primary purpose of a weather app project is to offer users a convenient and reliable tool to access weather information.

Block Diagram



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Weather App</title>

    <link rel="stylesheet" href="weather.css" />
  </head>
  <body>
    <div class="wrapper">
      <div class="shape shape-1"></div>
      <div class="shape shape-2"></div>
      <div class="container">
        <div class="search-container">
          <input
            type="text"
            placeholder="Search a location"
            id="city"
            list="cities"
            value=""
          />
          <datalist id="cities">
            <option value="mumbai">
            <option value="delhi">
            <option value="Bangalore">
            <option value="kolkata">
            <option value="Chennai">
            <option value="Ahmedabad">
            <option value="Hyderabad">
            <option value="pune">
            <option value="surat">
```

```
<option value="America">
<option value="Armenia">
<option value="Bahamas">
<option value="Belgium">
<option value="Brazil">
<option value="Canada">
<option value="Denmark">
<option value="England">
<option value="France">
<option value="Germany">
<option value="Hungary">
<option value="Iceland">
<option value="Ireland">
<option value="Italy">
<option value="Japan">
<option value="Kenya">
<option value="Latvia">
<option value="Mexico">
<option value="Netherlands">
<option value="Norway">
<option value="Poland">
<option value="Portugal">
<option value="Romania">
<option value="Russia">
<option value="Spain">
<option value="Sweden">
<option value="Switzerland">
<option value="Turkey">
<option value="Ukraine">
<option value="Vietnam">
<option value="Wales">
<option value="Zimbabwe">
```

```
                <option value="chandigarh">
                <option value="gujrat">
                <option value="punjab">
                <option value="simhachalam">
                <option value="rk-beach">
                <option value="arunachal pradesh">
                <option value="parvathipuram">
                <option value="paderu">
                <option value="odisha">
                <option value="maddilapalem">
            </datalist>
            <button id="search-btn">Find</button>
        </div>
        <div id="result"></div>
    </div>
  </div>
  <!-- Script -->
  <script src="weather.js"></script>
 </body>
</html>
```

```
let result = document.getElementById("result");
let searchBtn = document.getElementById("search-btn");
let cityRef = document.getElementById("city");
const key ="1946852246b6fe5011a8a2565d4b9718"
//Function to fetch weather details from api and display them
let getWeather = () => {
  let cityValue = cityRef.value;
  //If input field is empty
  if (cityValue.length == 0) {
    result.innerHTML = `<h3 class="msg">Please enter a city name</h3>`;
  }
  //If input field is NOT empty
  else {
    let url = `https://api.openweathermap.org/data/2.5/weather?q=${cityValue}&appid=${key}&units=metric`;
    //Clear the input field
    cityRef.value = "";
    fetch(url)
      .then((resp) => resp.json())
      //If city name is valid
      .then((data) => {
        console.log(data);
        console.log(data.weather[0].icon);
        console.log(data.weather[0].main);
        console.log(data.weather[0].description);
        console.log(data.name);
        console.log(data.main.temp_min);
        console.log(data.main.temp_max);
        result.innerHTML = `
        <h2>${data.name}</h2>
        <h4 class="weather">${data.weather[0].main}</h4>
        <h4 class="desc">${data.weather[0].description}</h4>
        <img src="https://openweathermap.org/img/w/${data.weather[0].icon}.png">
```

```
          <img src="https://openweathermap.org/img/w/${data.weather[0].icon}.png">
          <h1>${data.main.temp} &#176C</h1>
          <div class="temp-container">
            <div>
              <h4 class="title">Humidity</h4>
              <h4 class="temp">${data.main.temp_min}&#176C</h4>
            </div>
            <div>
              <h4 class="title">temperature</h4>
              <h4 class="temp">${data.main.temp_max}&#176C</h4>
            </div>
          </div>
          `;
        })
        //If city name is NOT valid
        .catch(() => {
          result.innerHTML = `<h3 class="msg">Location not found</h3>`;
        });
    }
};
searchBtn.addEventListener("click", getWeather);
window.addEventListener("load", getWeather);
```

```css
{
  padding: 0;
  margin: 0;
  box-sizing: border-box;
  font-family: 'Lucida Handwriting', cursive;
}
:root {
  --white: #ffffff;
  --off-white: #e5e5e5;
  --transp-white-1: rgba(255, 255, 255, 0.25);
  --transp-white-2: rgba(255, 255, 255, 0.1);
  --pink: #f54cd0;
  --blue: #4475ef;
  --pink2: #e778cf;
  --black: #0e0a0;
  --shadow: rgba(2, 28, 53, 0.2);
}
body {
  height: 100vh;
  background: url("https://media0.giphy.com/media/67uxmHhIF3uh6Ph8ew/giphy.gif?cid=ecf05e47peyw4mgcagg8j02pp3fz1
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: 100% 100%;
}
.wrapper {
  font-size: 16px;
  width: 90vw;
  max-width: 28em;
  position: absolute;
  transform: translate(-50%, -50%);
  top: 50%;
  left: 50%;
```

```
}

.container {
    width: 100%;
    background: var(--transp-white-2);
    backdrop-filter: blur(10px);
    padding: 3em 1.8em;
    border: 2px solid var(--transp-white-2);
    border-radius: 6.6em;
    box-shadow: 0 1.8em 3.7em var(--shadow);
    box-shadow: 0px 0px 20px 2px white;
    text-align: center;
}

.search-container {
    font-size: 1em;
    display: grid;
    grid-template-columns: 9fr 3fr;
    gap: 1.25em;
}

.search-container input,
.search-container button {
    outline: none;
    font-size: 1em;
    border: none;
}

.search-container input {
    padding: 0.7em;
    background-color: transparent;
    border-bottom: 2px solid var(--transp-white-1);
    color: var(--white);
    font-weight: 300;
```

```
}
.search-container button:hover{
    color: white;
    background-color: green;
    cursor: pointer;
}
.search-container input::placeholder {
    color: var(--off-white);
}
.search-container input:focus {
    border-color: var(--pink2);
}
.search-container button {
    color: rgb(14, 13, 13);
    background-color: var(--white);
    border-radius: 0.3em;
}
#result h2 {
    color: white;
    text-transform: uppercase;
    letter-spacing: 0.18em;
    font-weight: 600;
    margin: 1.25em 0;
}
.weather {
    margin-top: -0.7em;
    color: white;
}
.desc{
    color: white;
}
```

```
#result img {
    margin: 0.6em 0 0 0;
    width: 6.2em;
    filter: drop-shadow(0 1.8em 3.7em var(--shadow));
}
#result h1 {
    font-size: 4em;
    margin: 0.3em 0 0.7em 0;
    line-height: 0;
    font-weight: 400;
    color: var(--white);
}
.temp-container {
    display: flex;
    justify-content: center;
}
.temp-container div {
    padding: 0.3em 1em;
}
.temp-container div:first-child {
    border-right: 1px solid var(--transp-white-1);
}
.temp-container .title {
    font-weight: 600;
    color: var(--white);
}
.temp-container .temp {
    font-weight: 300;
    color: var(--off-white);
}
```

```
h5 {
  margin-top: 1.5em;
  color: #fff;
  font-weight: 500;
  text-transform: uppercase;
  letter-spacing: 0.1em;
}

@media screen and (max-width: 450px) {
  .wrapper {
    font-size: 14px;
  }
```

Advantages :-

1. Skill Environment :- Developing a weather app as a front-end-app project allows front end developers to improve their skills in HTML, CSS and java script, which are essential technologies for web development.

2. Real - world Application :- A weather app is pratical project that provides real-world value to users. it allows developers to work on something relevant and useful. their abilities to potential employers.

3. User Interface Design :- weather apps require an intuitive and visually appealing user interface. Building such an interface helps front-end-developers sharpen their design.

4. API Integration :- Integration weather data APIs into the app teaches developers how to work with external data sources and handle requests, a crucial aspect of modern web development.

5. Cross - browser Compability :- Front end developers must ensure the app functions correctly across different web browers and devices. Building a weather app gives them hands-on-experiences in dealing with cross-brocwer compability issue.

Disadvantages :-

1. limited scope :- A weather app, while useful may be considered a relatively simple object in terms of function Front-end-developers may miss the opportunity to work more complex applications that involve backened development.

2. Lack of Backened Experience :- Building a weather opp purely as a front-end project may not provide opportunities to gain experience in server-side programming, database management or backened architecture

3. Data limitations :- Front-end developers rely on weather APIs to fetch weather date. the amount of data and the available features are dependent on the capabilities of the chosen API, limiting the scope for data manipulation and analysis

4. Security Concerns :- Handling APIs and external data sources requires careful consideration of security to prevent data breathers.

5. Performance challenges :- Depending on the API and data methods, front-end-developers may encounter performance issues if the app requires frequent up dates (or) dates.

Applications :-

* Real - Time weather Information :- Display current weather conditions, including temperature, humidity, wind, speed along with an icon representing the weather type.

* Location - based Forecast :- Allow users to enter their location or use their devices gps to get localised weather forecasts for the current day and upcoming days.

* Multiple locations :- Enable to users to save and switch between multiple locations. so they can check the weather for places they frequently visit or plan to travel to.

* weather Radar and Maps :- Implement weather radar and interactive maps to visualize weather, patterns, including rain, snow.

* weather Alerts and warnings :- Display severe weather alerts and warnings for the users locations or selected regions, ensuring users stay informed about dangerous conditions.

* Historical weather Data :- offer access to historical weather app, allowing user to explore past weather patterns and trends.

* weather widgets :- Create small weather widgets that can be embedded on the websites or shared on social media platforms.

* Responsive Design :- Ensure the app is fully responsive and optimized for various devices including desktops and mobile phones

* Accessibility :- Make the app accesible to users with disabilities by adhering to accesibility standards and guidelines.

* offline support :- Implement caching and storage mechanisms to provide basic wheather information even when the users is offline

* social Media Integration :- Allow users to share weather updates on social media platforms.

Conclusion :-

project Overview : Briefly summarize the purpose of the weather app its target audience, and the technologies used in its development.

Feature and Functionality : Highlight the key features implemented in the app, such as real time weather data location based weather forecasts, interactive UI designs and any other unique functionalities

Design and user Experience :- Discuss the design choices made the brought out the project, including color schemes and overall user interface. Emphaize how the design element contribute and intuitive navigation.

Challenges Faced Describe any obstacles or challenges encounted during the development process such as third-party APIs, handling or ensuring campability Explain how you overcame these challanges.

Responsiveness and Campability : Discuss the effort put into ensuring that the weather app is responsive and compatible with various devices and browers. Mention any specific breakpoints or media queries used to optimize the app's display on different screen sizes.

Future Scope :-

Keep in mind that technology and trends are constantly evolving. so there might be even more exciting possibilities available in the future.

* Real-time data :- Improve the app by incorporation real-time weather data to provide users with the most up-to-data information.

* Interactive wheather Map :- Integrate an interactive map with various layers to display weather related data such as temperature, wind patterns and more. Zoom the map to explore weather conditions in different locations.

* weather Alerts :- Implement a notification system that sends weather alerts and warnings to users for severe weather conditions.

* Social Media Integration :- Enable users to share weather updates and images on social media platforms directly from the app.

* Voice commands :- Integrate voice recognition capabilities, allowing users to request weather information using voice commands. This could be particularly useful for hands free access on mobile devices or smart home devices

* weather Gamification :- Add gamification element by to the app, such as rewards or challengs based on weather condition (or) predictions.

* Multilingual Support :- Expands the apps usability by providing support for multiple languages, making it accessible to a broader audience.

* offline mode :- Develop a feature that allows users to access cached weather data and forecasts even when they have limited or no internet connectivity.

* Animations and Transitions :- Use suitable animations and smooth transitions to enhance the user experience and make the app feel more interactive.

* offline Support :- Implementing caching and storage mechnanisms to provide basic weather information even when the users is offline.

* Social Media Integration :- Allow users to share weather updates on social media platforms.

Search a location

Find

DELHI

33.05 °C

Humidity
33.05%

Temperature
33.05°C

Search a location

Find

MUMBAI

29.99 °C

Humidity	temperature
28.94°C	29.99°C