Project Report format

1. INTRODUCTION
 1. Overview
  A brief discription about your project

 2. purpose
  The use of this project. what can be achived using this

2. LITERATURE SURVEY
 1. Existing problem
 Existing approaches or method to solve this problem
 2. proposed solution
 what is the method or solution suggested by you?

3. THEORITICAL ANALYSIS
 1. Block diagram
 Diagrammatic overview of the project
 2. Hardware / software designing
 Hardware and software requirements of the project

4. Result
 Finding (output) of the project along with screen shots

5. ADVANTAGES & DISADVANTAGES
 list of advantages and disadvantages of the proposed solution.

6. Application:

The area where this solution can be applied

7. conclusion:

Conclusion summarizing the entair [illegible] work and findings

8. FUTURE SCOPE

Enhancement that can be made in the future.

PROJECT REPORT FORMAT

INTRODUCTION

→ over view:

weather APP is an one step solution for staying up-to-date with real-time weather forecasts.

This project is an existing endever in "Front-End Development" aimed to providing users with a sleek and inductive weather application. our mission is to deliver an engaging user experience by presenting weather data in a visually appealing and informative

⇒ Purpose:-
weather plays a significant role in our daily lives, influencing our purpose activities, clothing choices & overall well-being. People constanity seek accurate weather information to plan their scheduler, accordinly, while many weather application exist. weather app stands out by priortiting user experience and simplicity.

The purpose of a weather app project is to create a software application that provide users with real-time weathers information and forecasts for a specific location (or) for a multiple location.

LITERATURE SURVEY

=> Existing Problem:

Real-Time weather Data:

The app should be able to fetch and display current weather conditions, including temperature, humidity, wind speed & visibility.

weather fore casts:

providing accurate weather forecasts for the next few days in crucial as it help users plan ahead for events, travel or out door activities

Location Based services:

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

User-friendly Interface:

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

customization:

users may want to customize the app to display weather units in their preferred format

⇒ **Proposed solution:**

weather Alerts: The app may include a feature to send weather alerts and notifications to users for server weather conditions.

maps and Radar:
Including weather maps and radar data can help users visualize weather patterns and track storms.

Integration with API's:
The app may utilize third-party weather API's to access accurate and up-to-date weather dat

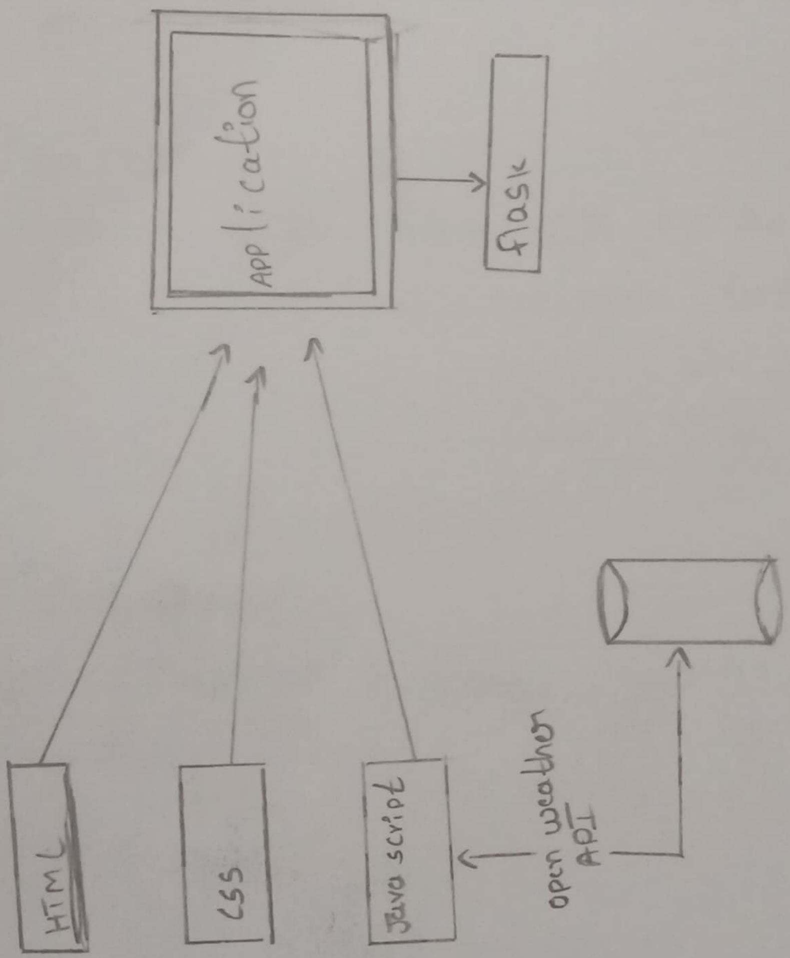
Cross-platform compatability:
To reacher a boader audience, the app should be compatible with different operating systems such as android, ios, web browsers

offline access:
All though real time data is essential, the app might consider providing basic weather information even when the device is off line.

over all, the primary purpose of weather app project is to offer users a convenient and reliable tool to acces weather information.

THEORITICAL ANALYSIS

=) Block Diagram:



⇒ <u>Hard ware / soft ware Designing</u>:

Hardware and software requirements of the project

Accessing a data base :

* The system should allow administrator to add historical weather data.

* The system should be able to recognize patterns in (s) temperature, humidity, and windeith use of historical data.

Software constraints:

* The development of the system will be constrained by the availability of required softcware such as webservers, data set.

Hardware Requirements:

* The system requires a database in order to store persistent data.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Weather App </title>
    <link rel="icon" type="image/x-icon" href="/Images/favicon.png">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.2/css/all.min.css"

integrity="sha512-1sCRPdkRXhBV2PBLUdRb4tMg1w2YPf37qatUFeS7zlBy7jJI8Lf4VHwWfZZfpXtYS
Ly85pkm9GaYVYMfw5BC1A=="
        crossorigin="anonymous" referrerpolicy="no-referrer" />
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@500&display=swap"
rel="stylesheet">
 <link rel="stylesheet" href="style.css">
</head>
<body>

    <!-- Main Container for the weather web app -->
    <div class="weatherContainer">
      <div id="location-details">
        <button id="get-location">get-location</button>
      </div>
        <h2 class="appHeading"> Weather App </h2>
        <!-- Input Divition for city name -->
        <div class="search">
            <form action="" id="weatherForm">
                <input type="text" name="" id="weatherInput"
                list="cities" placeholder="Enter your city here...">
                <datalist id="cities">
                    <option value="mumbai">
                    <option value="delhi">
                    <option value="Bengalore">
                    <option value="kolkata">
                    <option value="Chennai">
                    <option value="Ahmedabad">
                    <option value="Hyderabad">
                    <option value="pune">
                    <option value="surat">
                    <option value="Kanput">
                    <option value="jaipur">
                    <option value="lucknow">
                    <option value="tirupati">
                    <option value="vizag">
                    <option value="visakhapatnam">
                    <option value="vijaywada">
                    <option value="kurnool">
                    <option value="kakinada">
                    <option value="ananthapur">
                    <option value="srisailam">
                    <option value="vizinagaram">
                    <option value="agra">
                      <option value="italy">
                        <option value="devarapalli">
                        <option value="anakapalli">
                        <option value="london">
                        <option value="srikakulam">
                        <option value="rajam">
                        <option value="pendurthi">
                        <option value="kothavalasa">
                        <option value="thirchi">
<option value="guntur">
                        <option value="thirupathi">
                        <option value="england">
                        <option value="canada">
                        <option value="goa">
                          <option value="aruku">
                            <option value="s-kota">
                            <option value="kadapa">
                            <option value="America">
                            <option value="chandigarh">
                            <option value="gujarat">
                            <option value="punjab">
                            <option value="Manyam">
                            <option value="secundrabad">
                            <option value="arunachal pradesh">
                            <option value="parvathipuram">
                            <option value="paderu">
                            <option value="odisha">
                            <option value="bhuvaneswar">

                  </datalist>
                <button class="searchBtn" type="submit">
                    <ul>
                        <li><i class="fa-solid fa-magnifying-glass"></i></li>
                    </ul>
                </button>
            </form>

        </div>
        <!-- Outut Will show on the screen -->
        <h2 id="city"></h2>
        <img src="" alt="" id="weatherImage">
        <p class="weatherMain" id="weatherMain"></p>
        <h2 id="temp"><span class="temp"></span></h2>
        <div class="todayDates"></div>
        <div id="todayTime"></div>
    </div>
</body>
<script src="script.js"></script>
<script>
    let locationButton = document.getElementById("get-location");
let locationDiv = document.getElementById("location-details");
7
locationButton.addEventListener("click", () => {
  //Geolocation APU is used to get geographical position of a user and is available
inside the navigator object
  if (navigator.geolocation) {
    //returns position(latitude and longitude) or error
    navigator.geolocation.getCurrentPosition(showLocation, checkError);
  } else {
    //For old browser i.e IE
    locationDiv.innerText = "The browser does not support geolocation";
  }
});

//Error Checks
const checkError = (error) => {
  switch (error.code) {
    case error.PERMISSION_DENIED:
      locationDiv.innerText = "Please allow access to location";
      break;
    case error.POSITION_UNAVAILABLE:
      //usually fired for firefox
      locationDiv.innerText = "Location Information unavailable";
      break;
    case error.TIMEOUT:
      locationDiv.innerText = "The request to get user location timed out";
  }
};

const showLocation = async (position) => {
  //We user the NOminatim API for getting actual addres from latitude and longitude
  let response = await fetch(

`https://nominatim.openstreetmap.org/reverse?lat=${position.coords.latitude}&lon=${
position.coords.longitude}&format=json`
  );
  //store response object
  let data = await response.json();
  locationDiv.innerText = `${data.address.city}, ${data.address.country}`;
};
</script>

</html>36j
```

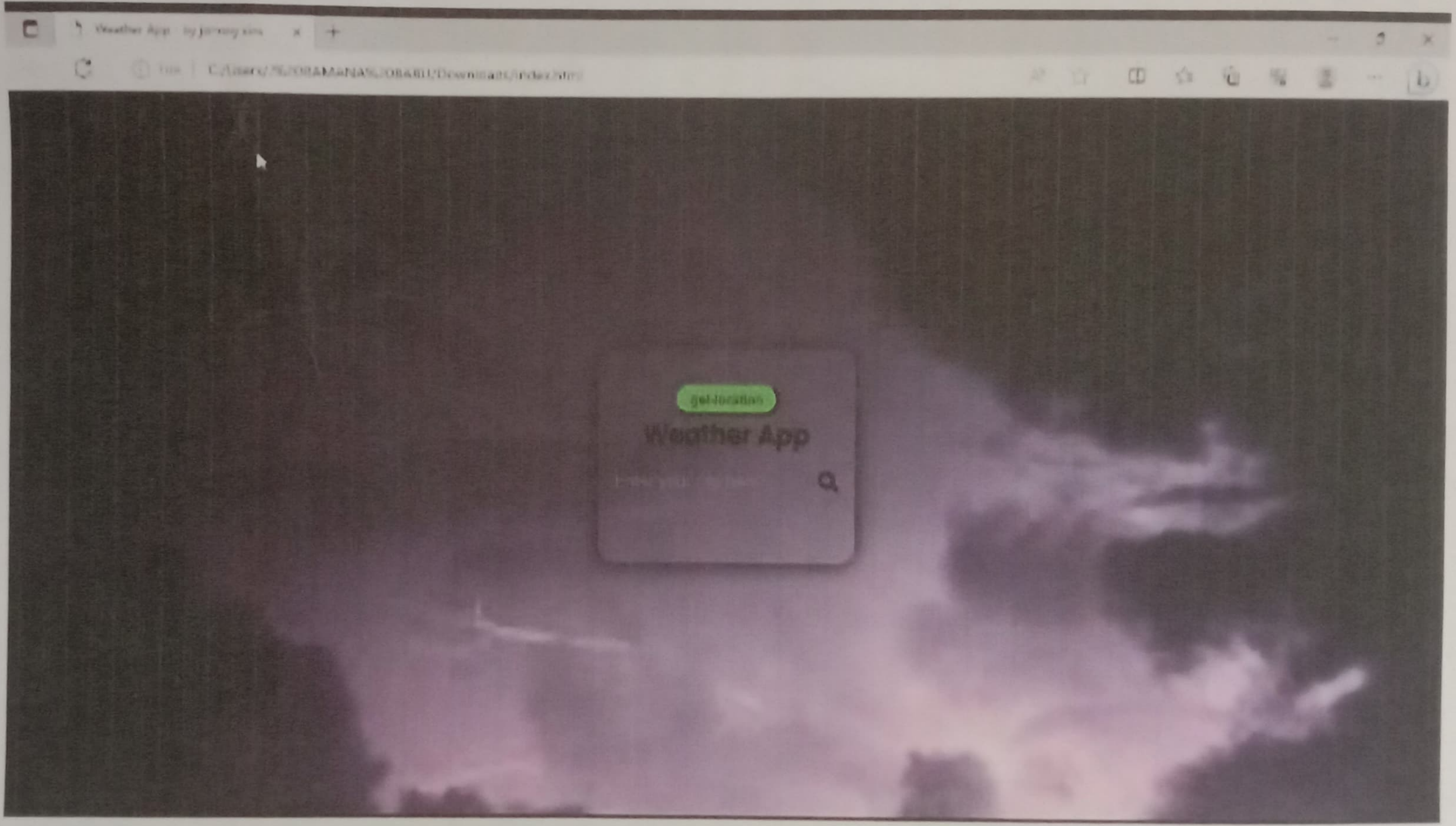
```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
.weatherContainer-#location-details{
height: fit-content;
width: 10px;
position: absolute;

}
.weatherContainer #get-location{
    height: 30px;
    width: 100px;
    background-color: #4CAF50;
    border-radius: 50px;
}

body {
    width: 100%;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
    background-image:
url("https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQ4pqdGjHS_lBOpKuZZCatTEU
m0JjIhutE1uw&usqp=CAU");
    background-size: cover;
}

.weatherContainer {
    margin: 2rem;
    padding: 2rem 1rem;
    border-radius: 1rem;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    box-shadow: 0px 0px 19px 0px #0d0d0d;
    background :transparent;
}

#weatherForm input {
    border: black;
    color: black;
    outline: black;
    font-size: 1rem;
    font-family: sans-serif;
    background: transparent;
}
```

```css
#weatherForm ul li {
    list-style: none;
}

.searchBtn {
    background: transparent;
    border: none;
}

.searchBtn ul li {
    list-style: none;
    font-size: 1.2rem;
}

#weatherForm {
    display: flex;
    justify-content: space-around;
}

.appHeading {
    text-align: center;
    margin-bottom: 1rem;
    font-family: 'Poppins', sans-serif;
}

#city {
    text-transform: capitalize;
    text-align: center;
    margin-top: 1rem;
    font-family: 'Poppins', sans-serif;
}

.weatherContainer img {
    width: 50%;
}

#temp {
    word-spacing: -8px;
    font-size: 2.5rem;
    margin-bottom: 1rem;
    font-family: 'Poppins', sans-serif;
}

#temp sup {
    font-size: 1.5rem;
}

.weatherMain {
    font-family: sans-serif;
}

#todayTime,
.todayDates {
    font-family: sans-serif;
    line-height: 2rem;
}


/* Utility Classes */
.d-flex {
    display: flex;
}

.justify-space-around {
    justify-content: space-around;
}

.justify-space-center {
    justify-content: center;
}

.align-items-center {
    align-items: center;
}

.f-col {
    flex-direction: column;
}

/* Media Query for Responsive */
@media screen and (max-width: 307px) {
    #weatherForm {
        flex-direction: column;
    }

    #weatherForm input {
        text-align: center;
        margin-bottom: 1rem;
    }
}
```

```
const cityName = document.querySelector('#weatherInput');
const searchBtn = document.querySelector('#searchBtn');
const form = document.getElementById('weatherForm');
const myCity = document.getElementById('city');
const image = document.getElementById('weatherImage');
const weather = document.getElementById('weatherMain');
const temp = document.querySelector('.temp');
const dates = document.querySelector('.todayDates');
const times = document.getElementById('todayTime');
let date = new Date();

// Function work when user input the city name
form.addEventListener('submit', function (e) {

    // preventDefault() to stop page reload
    e.preventDefault();

    // Updating the city name
    let city = cityName.value;
    const myWeatherContainer = document.querySelector('.weatherContainer');
    const apiID ="d632731e811e8fa82bf0641a112b2df4";
    // API URL
    let url =
`https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${api
ID}`

    // fetching data from the weather api
    fetch(url).then((response) => {
        return response.json();
    }).then((data) => {

        const tempValue = Math.round(data['main']['temp']);
        const weatherMain = data['weather'][0]['main'];
        weather.innerHTML = weatherMain;

        // Updating the DOM
        myCity.innerHTML = city;
        temp.innerHTML = `${tempValue}`
        weather.innerHTML = `${weatherMain}`
        temp.innerHTML = `${tempValue}<span><sup>o</sup>C</span.`;

        // Updating the Images according to the weather
        if (weatherMain == 'Clear') {
            image.src = `./Images/sunny.png`
            myWeatherContainer.style.backgroundColor = '#ec6e4c'
        }
        if (weatherMain == 'Clouds') {
            image.src = `./Images/clouds.png`
            myWeatherContainer.style.backgroundColor = '#86d3d3'
        }
```

```
if (weatherMain == 'Rain') {
    image.src = `./Images/Rain.png`
    myWeatherContainer.style.backgroundColor = '#494bcf'
}
if (weatherMain == 'Drizzle') {
    image.src = `./Images/Drizzle.png`
    myWeatherContainer.style.backgroundColor = '#8ecfcf'
}
if (weatherMain == 'Haze') {
    image.src = `./Images/Drizzle.png`
    myWeatherContainer.style.backgroundColor = '#d8ced2'
}

// Updating dates
const currentMonth = date.getMonth();
switch (currentMonth) {
    case 0:
        dates.innerHTML = `${date.getDate()}, Jan`
        break;
    case 1:
        dates.innerHTML = `${date.getDate()}, Feb`
        break;
    case 2:
        dates.innerHTML = `${date.getDate()}, Mar`
        break;
    case 3:
        dates.innerHTML = `${date.getDate()}, Apr`
        break;
    case 4:
        dates.innerHTML = `${date.getDate()}, May`
        break;
    case 5:
        dates.innerHTML = `${date.getDate()}, Jun`
        break;
    case 6:
        dates.innerHTML = `${date.getDate()}, Jul`
        break;
    case 7:
        dates.innerHTML = `${date.getDate()}, Aug`
        break;
    case 8:
        dates.innerHTML = `${date.getDate()}, Sept.`
        break;
    case 9:
        dates.innerHTML = `${date.getDate()}, Oct.`
        break;
    case 10:
        dates.innerHTML = `${date.getDate()}, Nov`
        break;
    case 11:
```

```
                dates.innerHTML = `${date.getDate()}, Dec`
                break;
        }

        // Updating times
        function leftInterval() {
            const left = document.getElementById('todayTime')
            let leftDate = new Date();
            let hours = leftDate.getHours();
            let minutes = leftDate.getMinutes();
            let seconds = leftDate.getSeconds();

            if (hours == 0) {
                hours = 12;
            }

            if (hours > 12) {
                hours = hours - 12;
            }
            left.innerHTML = `${hours}h: ${minutes}m: ${seconds}s`
        }
        setInterval(leftInterval, 1000);
    })
})
```


Weather App - by jamesg.sim
get location
Weather App

ADVANTAGES AND DISADVANAGES

=> **Advantages:**

Skill enhancement:

Developing a weather app as a front-end project allows front end developers to improve their skills in HTML CSS and Java script.

Real - world Application:

A wheather app is a practical project that provides real-world value to users. It also allows developers. to work on something relevant

User Interface Design:

wheather apps require an intative and usually appealing user interface. Building such an interface helps to sharpen their design.

Reliance on Technology:

weather forecasting relies heavily on technology and if the technology fails or is un available, accurate predictions cannot be made.

Limitted Time frame:

forecasts are usually only accurate for a short time frame. making it difficult to plan head

⟹ Disadvantages:

Data Limitations:

front-end developers rely on weather API's to fetch weather data. The amount of data and the available features are dependent.

Lack of Back End Experience:

Building a weather app purely as a front-end project may not provide oppurtunities to gain experience in server-side programming.

Security concerns:

Handling API's and external data sources requires careful consideration of security to prevent data breaches to sensitive information

confusing Terminology:

The terminology used in weather forecast-ing can be confusing, making it difficult for some people to understand predictions.

Limited reach:

whether forecast are not available for many remote (or) sparsely populated areas, making it difficult for people in these areas to prepare for severe weather.

APPLICATIONS

Real-Time weather Information:

Display current weather conditions, including temperature, humidity wind speed and direction, along with an icon representing weather-type. Ex: Sunny, cloudy, Rainy.

Location-Based forecast:

Allow users to enter their location ore use their devices. GPS to get localized weather forecasts for the current day and upcoming days.

multiple Locations:

Enable users to save and switch between multiple locations. so they Can check weather for places they frequently visit.

weather Rader and maps:

Implement weather radar and interactive maps to visualize weather, patterns. including rain, snow and cloud cover.

weather Alerts and warnings:

Display severe weather alerts & warnings for the users location or selected regions, ensuring users stay informed about potentially dangerous conditions

Hourly and Daily forecasts:

provide detailed weather forecasts for the next few hours and several days ahead, giving users a comprehensive view of what to expect.

User preferences:

Let users customize the app by setting the temperature units.

Ex: celsius (or) fahrenheit

Historical weather Data:

offer access to historical weather data, allowing user to Explore past weather patterns.

social media Integration:

Allow Users to share weather updates on social media plat forms

responsive Designs:

Ensure the app is fully responsive and optimized for various devices

Accessibility:

make the app accessible to users with the disabilities by gadhering to accessibility standards and guidlines.

CONCLUSION

The weather apps are increasingly accurate and useful. and their benefits extend widely across the economy. while much has been accomplished in improving weather forecasts. their remains much room for improvements.

simultaneously. they are developing new technologies and observational networks that can enhance forecaster skill.

FUTURE SCOPE

The demand for weather and climate forecast information in support of critical decision-making has grown rapidly during the last decade. and will grow even faster in the coming years Great advances have been made in the utilization of predications in many years of human activities

The future of weather applications is promising with increasing demand for real time and accurate weather information.