WEATHER DASHBOARD USING API

A MINI-PROJECT REPORT

Submitted by

KRISHNA PRASAD CA	211701025
ANSHUMAN KAILASH	211701006

in partial fulfilment for the course

CD19643 – WEB ESSENTIALS

for the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND DESIGN

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR

THANDALAM

CHENNAI - 602 105

MAY 2024

RAJALAKSHMI ENGINEERING COLLEGE

CHENNAI - 602105

BONAFIDE CERTIFICATE

Certified that this project report **"WEATHER DASHBOARD USING API"** is the bonafide work of **"KRISHNA PRASAD CA (211701025) ANSHUMAN KAILASH (211701006)"** who carried out the project work for the subject CD19643- Web Essentials under my supervision.

SIGNATURE

Prof. Uma Maheshwar Rao ,
Head of the Department
Associate Professor
Department of Computer Science and Design
Rajalakshmi Engineering College
Chennai - 602105

SIGNATURE

Mr.N.Duraimurugan, M.Tech.,
Supervisor
Associate Professor
Department of Computer Science and Engineering
Rajalakshmi Engineering College
Chennai - 602105

Submitted to Project and Viva Voce Examination for the subject

CD19643 – Web Essentials held on____________________.

Internal Examiner External Examiner

ABSTRACT

A weather dashboard using API is a versatile tool designed to provide real-time and forecasted weather data to users through an interactive, user-friendly interface. The core functionality of the dashboard revolves around integrating with weather APIs, such as OpenWeatherMap or WeatherStack, to fetch data including temperature, humidity, wind speed, precipitation, and other meteorological variables. This data is then processed and displayed in various formats such as graphs, charts, and maps, allowing users to quickly interpret current weather conditions and trends.

The development of the weather dashboard involves several key components. Firstly, it requires a robust backend system capable of making API calls to retrieve and parse weather data. This system often utilizes languages like Python or JavaScript and frameworks such as Flask or Node.js. The frontend is built using modern web technologies like HTML5, CSS3, and JavaScript libraries such as React or Vue.js to ensure a responsive and intuitive user experience.

One of the primary benefits of a weather dashboard is its ability to present complex data in a simplified manner. Users can customize their view based on location, preferred units of measurement, and specific weather parameters of interest. Advanced features might include weather alerts, historical data analysis, and predictive models using machine learning algorithms to forecast future conditions with higher accuracy.

Moreover, the dashboard can be enhanced with geolocation services, enabling automatic updates based on the user's current position. Integration with additional APIs, such as Google Maps, can provide visual context to weather data, displaying information on a map for easier comprehension.

In summary, a weather dashboard using API serves as an essential tool for individuals and organizations requiring accurate and up-to-date weather information. Its development encompasses a blend of backend data processing and frontend design, culminating in a platform that transforms raw weather data into actionable insights, thus aiding in daily planning and decision-making.

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Mr.S.Meganathan, B.E, F.I.E.,** our Vice Chairman **Mr. Abhay Shankar Meganathan, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) Thangam Meganathan, Ph.D.,** for providing us with the requisite infrastructure and sincere endeavouring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N.Murugesan, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to our **Prof. Uma Maheshwar Rao** Associate Professor and Head of the Department of Computer Science and Design for his guidance and encouragement throughout the project work. We convey our sincere thanks to our internal guide and Project Coordinator, **Dr.N.Duraimurugan, M.Tech., PhD.,** Department of Computer Science and Engineering, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

KRISHNA PRASAD CA (211701025)

ANSHUMAN KAILASH (211701006)

CHAPTER 1

INTRODUCTION

The proposed project is a comprehensive weather forecasting website that utilizes the OpenWeatherMap API to provide users with real-time weather data. This project aims to deliver precise and current weather information, such as temperature, humidity, wind speed, and atmospheric conditions, for any specified location worldwide. By integrating the powerful OpenWeather API, the website ensures that users receive reliable and up-to-date weather data, which is essential for planning daily activities, travel, or outdoor events. The user interface of the website is meticulously designed to be both intuitive and responsive, ensuring seamless navigation across various devi ces, including desktops, tablets, and mobile phones. One of the standout features of the website is its search functionality, allowing users to effortlessly find weather information for their desired locations. Additionally, the website offers a detailed 5-day weather forecast, providing users with extended insights into future weather conditions. Visual elements play a significant role in enhancing user engagement on the website. The use of weather-specific icons and dynamic background images that change based on the current weather conditions helps to create an immersive experience. These visual cues not only make the website aesthetically pleasing but also allow users to quickly grasp the weather situation at a glance. To make complex weather data more understandable, the website incorporates user-friendly charts and graphs. These visual representations of weather trends help users to comprehend changes over time easily. The project also prioritizes accessibility, ensuring that all users, including those with disabilities, can effectively interact with the website. Features such as keyboard navigation and screen reader compatibility are integrated to comply with web accessibility standards. the website includes personalized features, such as location-based weather alerts and the ability to save favorite locations for quick access. Security measures are also a critical component, with the website employing secure connections to protect user data and ensure privacy. The back-end architecture of the website is built to handle a high volume of API requests efficiently, ensuring that performance remains optimal even during peak usage times. Overall, this weather forecasting website project combines advanced technology, user-centric design, and robust performance to deliver a valuable and reliable service. It empowers users with accurate weather information, helping them make informed decisions and stay prepared for any weather conditions .

CHAPTER 2

OBJECTIVE

The primary objective of the weather forecasting website project is to provide users with accurate and up-to-date weather information for any specified location globally. By leveraging the OpenWeather API, the website aims to deliver comprehensive weather data including current conditions, temperature, humidity, wind speed, and a detailed 5-day forecast. This information is essential for users to plan their daily activities, travel, and outdoor events effectively.

Key objectives of the project include:
1.Real-time Weather Data: To provide real-time weather updates that are precise and reliable, ensuring users have the most current information available.

2. User-friendly Interface: To design an intuitive and responsive user interface that facilitates seamless navigation and access to weather data across various devices, including desktops, tablets, and mobile phones.

3. Advanced Search Functionality: To incorporate a robust search feature that allows users to easily find weather information for any desired location.

4. Visual Enhancements: To use weather-specific icons, dynamic background images, and visual cues to create an engaging and immersive user experience.

5. Data Visualization: To include user-friendly charts and graphs that help users understand weather trends and changes over time.

6. Accessibility: To ensure the website is accessible to all users, including those with disabilities, by integrating features such as keyboard navigation and screen reader compatibility.

7. Personalization: To offer personalized features such as location-based weather alerts and the ability to save favorite locations for quick access.

8. Security: To implement robust security measures that protect user data and maintain privacy, including the use of secure connections.

9. Scalability and Performance: To build a back-end architecture capable of efficiently handling a high volume of API requests, ensuring optimal performance even during peak usage times.

By achieving these objectives, the weather forecasting website aims to empower users with reliable weather information, enhancing their ability to make informed decisions and stay prepared for various weather conditions.

CHAPTER 3

FUNCTIONAL OVERVIEW

3.1 Start: Initializing Application Environment and Components

- In this phase, the application initializes its core components, including libraries and dependencies, to create a robust environment for user interaction and data fetching.

2. Initialize Application: Setting Up Initial Configurations and Environment

- This step involves configuring the application settings, loading necessary libraries, and establishing a connection to the OpenWeather API, ensuring seamless integration and smooth operation.

3. User Accesses Website: Ensuring Connectivity and Displaying Loading Screen

- Upon user access, the website checks for internet connectivity, and if available, displays a loading screen while the application initializes, providing users with a responsive and engaging experience.

4. Location Detection: Utilizing Geolocation API and Manual Entry Prompt

- Using the Geolocation API, the application detects the user's location automatically. If access is denied, users are prompted to enter their location manually, ensuring accurate weather data retrieval.

5. Fetch Weather Data: Retrieving Current and Forecast Data from OpenWeather API

- This stage involves sending API requests to OpenWeather, retrieving current weather, 5-day forecast, and hourly forecast data, ensuring real-time and accurate weather updates for users.

6. Handle API Response: Processing and Parsing API Data

- Upon receiving the API response, the application processes the data, checks for errors, and parses it into usable formats for further manipulation and display on the user interface.

7. Process Weather Data: Extracting Relevant Information from API Response
- In this phase, the application extracts relevant weather information such as temperature, humidity, and weather conditions from the API response, preparing it for display to the user.

8. Update User Interface: Displaying Weather Information with Appropriate Icons
- The user interface is updated with the retrieved weather data, including the current weather, 5-day forecast, and hourly forecast, presented with visually appealing icons and data for easy interpretation.

9. Data Visualization: Creating Interactive Charts and Maps for Weather Trends
- To enhance user experience, the application creates interactive charts and weather maps to visually represent weather data trends and patterns, providing users with a comprehensive view of weather conditions.

10. User Interactions: Enabling Search, Favorites, and Custom Alerts
- Users can interact with the website by searching for different locations, saving favorites, and setting custom weather alerts, enhancing personalization and user engagement.

11. Dynamic Backgrounds: Changing Backgrounds Based on Weather Conditions
- The website dynamically changes its background based on current weather conditions, creating an immersive and visually engaging experience for users.

12. Responsive Design: Ensuring Adaptability Across Devices
- The website layout is designed to adapt seamlessly across desktops, tablets, and mobile devices, ensuring consistent functionality and user experience across various screen sizes.

13. Accessibility Features: Enhancing Accessibility for All Users
- Features like keyboard navigation and screen reader compatibility are implemented to make the website accessible to users with disabilities, ensuring inclusivity and equal access to information.

14. Performance Optimization: Improving Speed and Efficiency

- Performance optimization techniques such as API request optimization, data caching, and monitoring performance metrics are employed to ensure fast and efficient operation of the website.

15. Security Measures: Protecting User Data and Privacy

- The application implements security measures such as data encryption and user authentication to protect user data from unauthorized access, ensuring privacy and confidentiality.

16. End User Interaction: Saving User Preferences and State

- User preferences and state are saved to allow users to resume their interactions with the website seamlessly, providing a personalized and uninterrupted user experience.

17. End: Properly Closing the Application and Releasing Resources

- In the final stage, the application closes all processes, releases resources efficiently, and ensures proper shutdown, maintaining system stability and reliability.

CHAPTER 4
TECHNICAL IMPLEMENTATION

4.1.1 Frontend Development:

HTML/CSS: The frontend of the website is built using HTML for structuring the content and CSS for styling and layout design.

Responsive Design: The website is designed to be responsive, ensuring optimal viewing and interaction experience across a wide range of devices and screen sizes.

User Interface (UI): The UI is designed with a focus on usability and accessibility, featuring intuitive navigation, clear call-to-action buttons, and user-friendly forms.

4.1.2. API Integration:

- Obtain API key from OpenWeather API by registering on their website.
- Implement API requests in the backend to fetch weather data such as current weather, forecast, and historical data.
- Ensure proper handling of API responses and error cases.

4.1.3. Backend Development

- Set up a backend server using Node.js.
- Develop endpoints to handle API requests from the frontend.
- Implement data processing logic to format and filter weather data before sending it to the frontend.

4.1.4. Location Detection:

- Integrate Geolocation API to automatically detect the user's location.
- Provide an option for manual location entry in case geolocation is not available or denied by the user.

4.1.5. Data Processing and Display:

- Parse the JSON response received from the API to extract relevant weather information.
- Format the data to display current weather conditions, forecast, and additional details such as temperature, humidity, and wind speed.
- Use icons and graphics to represent different weather conditions for better visualization.

4.1.6. User Interactions:

- Enable users to search for weather information by entering a location or using their current location.
- Implement features like saving favorite locations, setting up custom alerts, and sharing weather updates on social media platforms.

4.1.7. Data Visualization:

- Utilize charts, graphs, and maps to visualize weather trends, temperature variations, and precipitation forecasts.
- Enhance the user experience with dynamic backgrounds or animations based on the current weather conditions.

4.1.8. Testing and Quality Assurance:

- Conduct thorough testing to identify and fix any bugs or issues in the application.
- Perform functional testing to ensure all features work as expected across different devices and browsers.
- Test for performance, scalability, and security vulnerabilities.

4.1.9. Deployment:

- Deploy the application to a web server or cloud platform such as AWS, Heroku, or Netlify.
- Configure domain settings and SSL certificates for secure communication.
- Monitor server logs and application metrics to ensure uptime and performance.

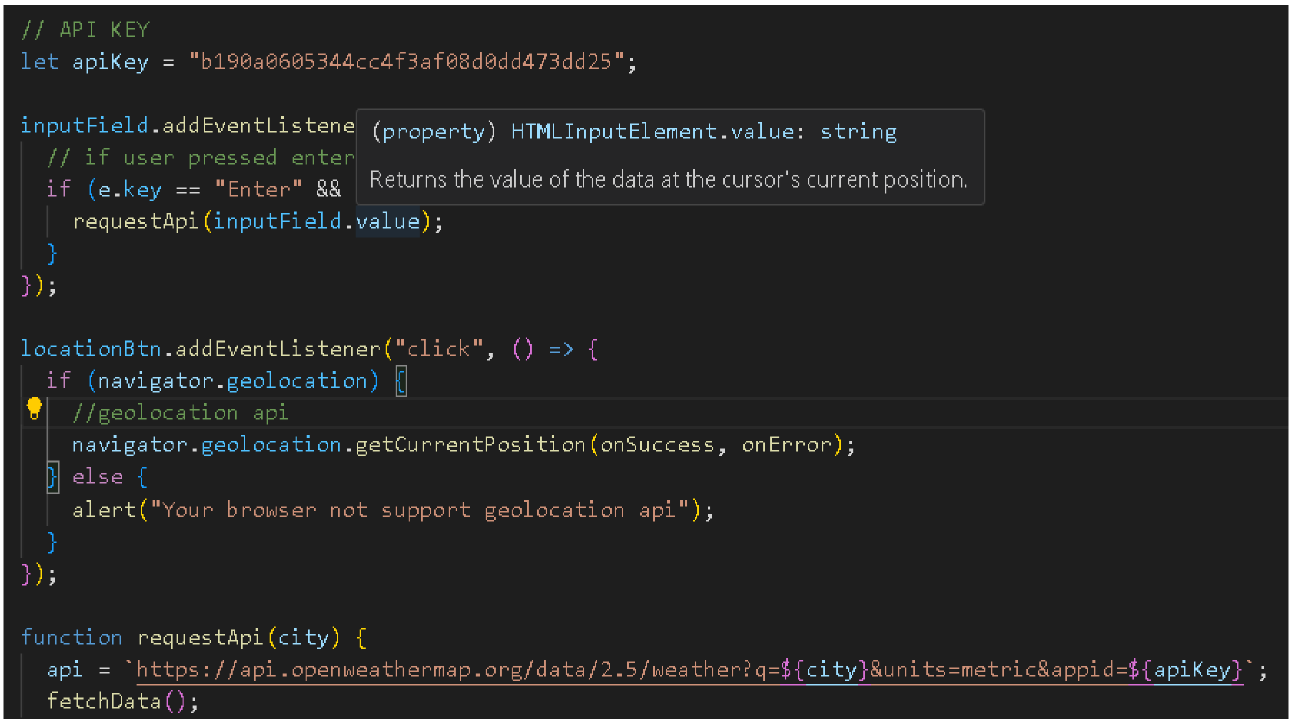
4.1.10. Documentation and Training:

- Create user documentation and guides explaining how to use the application.
- Provide training sessions or tutorials for users to learn about the features and functionalities of the weather application.

4.1.11. Maintenance and Updates:

- Regularly update the application with new features, bug fixes, and improvements based on user feedback and market trends.
- Monitor API usage and adapt to any changes or deprecations in the OpenWeather API.
- Maintain data integrity by ensuring accurate and up-to-date weather information is displayed to users.

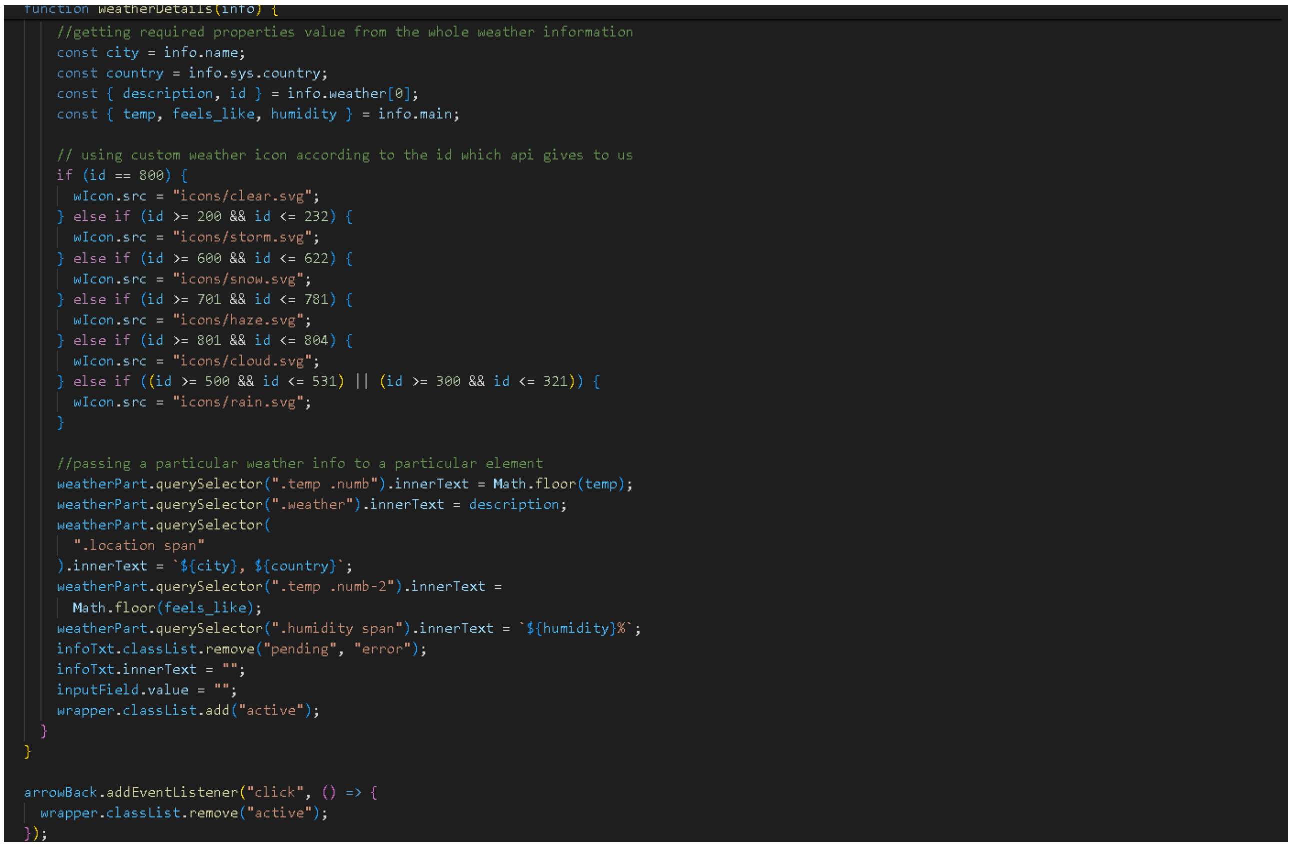
By following these steps, we have successfully implemented a weather application using the OpenWeather API, providing users with accurate and real-time weather forecasts and enhancing their overall experience.



```javascript
// API KEY
let apiKey = "b190a0605344cc4f3af08d0dd473dd25";

inputField.addEventListene
  // if user pressed enter
  if (e.key == "Enter" &&
    requestApi(inputField.value);
  }
});

locationBtn.addEventListener("click", () => {
  if (navigator.geolocation) {
    //geolocation api
    navigator.geolocation.getCurrentPosition(onSuccess, onError);
  } else {
    alert("Your browser not support geolocation api");
  }
});

function requestApi(city) {
  api = `https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${apiKey}`;
  fetchData();
```

```
(property) HTMLInputElement.value: string
Returns the value of the data at the cursor's current position.
```

Fig 4.1(a) API KEY REQUEST AND GEOLOCATION REQUEST



```
function weatherDetails(info) {
    //getting required properties value from the whole weather information
    const city = info.name;
    const country = info.sys.country;
    const { description, id } = info.weather[0];
    const { temp, feels_like, humidity } = info.main;

    // using custom weather icon according to the id which api gives to us
    if (id == 800) {
      wIcon.src = "icons/clear.svg";
    } else if (id >= 200 && id <= 232) {
      wIcon.src = "icons/storm.svg";
    } else if (id >= 600 && id <= 622) {
      wIcon.src = "icons/snow.svg";
    } else if (id >= 701 && id <= 781) {
      wIcon.src = "icons/haze.svg";
    } else if (id >= 801 && id <= 804) {
      wIcon.src = "icons/cloud.svg";
    } else if ((id >= 500 && id <= 531) || (id >= 300 && id <= 321)) {
      wIcon.src = "icons/rain.svg";
    }

    //passing a particular weather info to a particular element
    weatherPart.querySelector(".temp .numb").innerText = Math.floor(temp);
    weatherPart.querySelector(".weather").innerText = description;
    weatherPart.querySelector(
      ".location span"
    ).innerText = `${city}, ${country}`;
    weatherPart.querySelector(".temp .numb-2").innerText =
      Math.floor(feels_like);
    weatherPart.querySelector(".humidity span").innerText = `${humidity}%`;
    infoTxt.classList.remove("pending", "error");
    infoTxt.innerText = "";
    inputField.value = "";
    wrapper.classList.add("active");
  }
}

arrowBack.addEventListener("click", () => {
  wrapper.classList.remove("active");
});
```

Fig. 4.2(b). DATA VISUALIZATION AND ERROR HANDLING

4.2WORKFLOW:

The workflow for this project involves several key steps. First, project planning is essential to define the scope, objectives, and target audience while gathering requirements. Following this, integrate the OpenWeather API by obtaining an API key and implementing backend requests to fetch weather data. Frontend development comes next, focusing on designing a user-friendly interface using HTML, CSS, and JavaScript, and implementing interactive elements. Integrate the Geolocation API for automatic location detection and provide an option for manual entry.

Data processing is crucial to parse and format the API data to display current weather, forecasts, and additional details. Thorough functional testing is conducted across various devices and browsers to ensure performance and security. Deployment follows, where the application is deployed on a web server or cloud platform, with continuous monitoring of server logs and application metrics.

Lastly, ongoing maintenance involves regularly updating features, fixing bugs, and ensuring the weather data remains accurate.

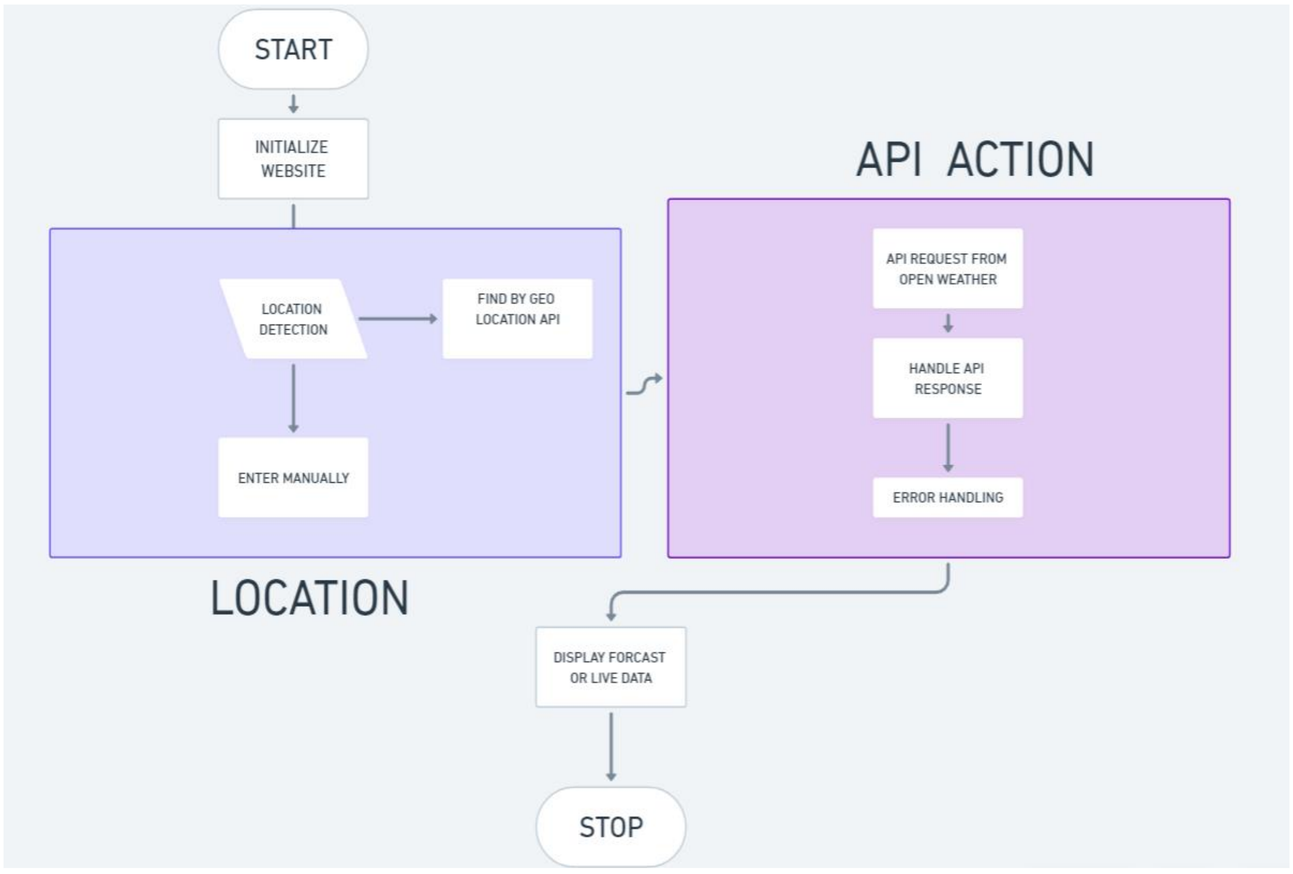


Fig 4.2(a) WORKFLOW DIAGRAM

4.4 USER INTERFACE:

The user interface of our financial website is designed with a primary focus on usability, accessibility, and intuitive navigation to enhance the overall user experience. The UI features a clean and modern design, with a visually appealing layout that facilitates easy comprehension and interaction for users of all levels of technical proficiency.

Key Features of the User Interface:

1. Responsive Design: The website is optimized to adapt seamlessly to various screen sizes and devices, ensuring consistent usability and accessibility across desktops, laptops, tablets, and mobile phones.

2. Intuitive Navigation: Clear and logical navigation menus, buttons, and links are strategically placed throughout the website to guide users efficiently to their desired destinations and functionalities.

3. User-Friendly Forms: Input forms for login, account registration, and other interactions are designed to be user-friendly, with clear labels, instructions, and error messages to assist users in completing tasks accurately and efficiently.

4. Visual Consistency: Consistent use of colors, fonts, icons, and imagery throughout the website maintains visual coherence and reinforces brand identity, contributing to a cohesive and professional user experience.

5. Accessibility Features: Accessibility considerations are incorporated into the design to ensure that all users, including those with disabilities, can access and interact with the website effectively. This includes support for keyboard navigation, screen reader compatibility, and adherence to accessibility standards and guidelines.

6. Interactive Elements: Interactive elements such as buttons, sliders, dropdown menus, and modal dialogs are implemented judiciously to enhance user engagement and facilitate seamless interaction with the website's functionalities.

7. Feedback and Validation: Real-time feedback and validation mechanisms are integrated into form fields and user actions to provide users with immediate feedback on their inputs, helping to prevent errors and guide them through the user journey smoothly.

CHAPTER 6

OUTPUT SCREENSHOTS

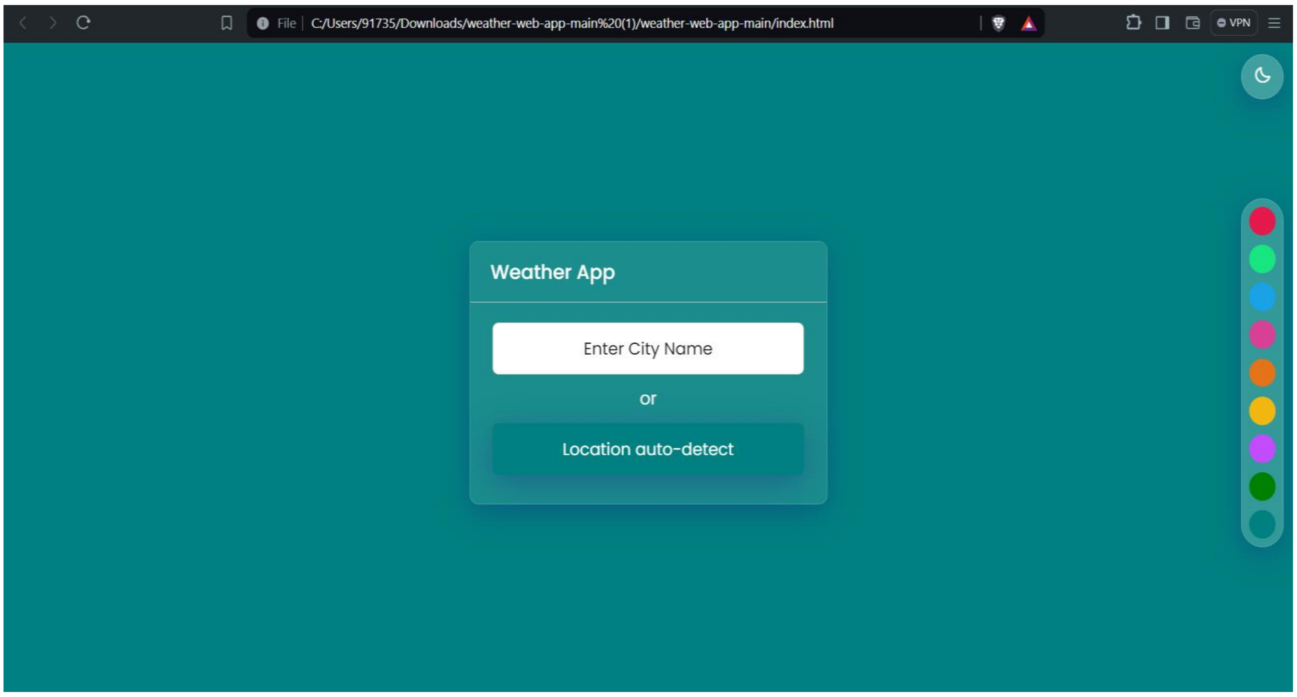

File
C:/Users/91735/Downloads/weather-web-app-main%20(1)/weather-web-app-main/index.html
VPN
Weather App
Enter City Name
or
Location auto-detect

Fig 6.1 WEATHER DASHBOARD LIGHT MODE

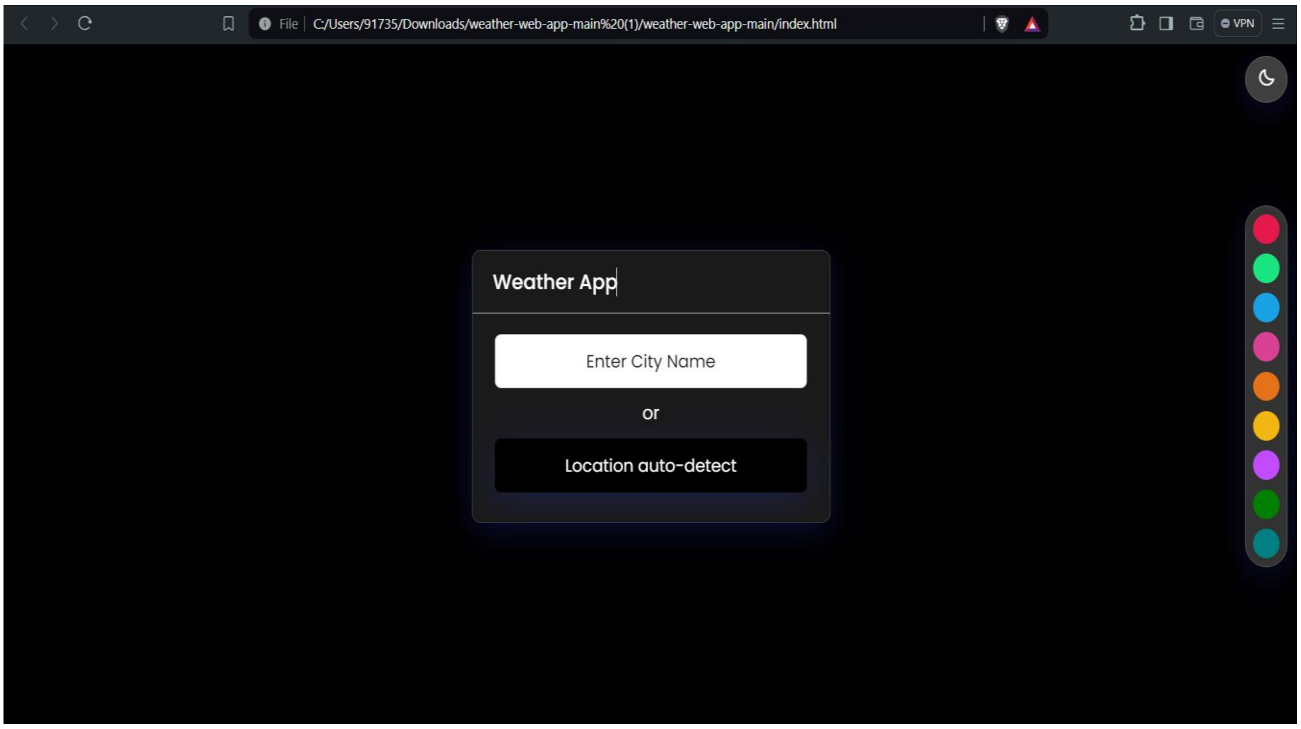

File
C:/Users/91735/Downloads/weather-web-app-main%20(1)/weather-web-app-main/index.html
VPN
Weather App
Enter City Name
or
Location auto-detect

Fig 6.2 WEATHER DASHBOARD DARK MODE

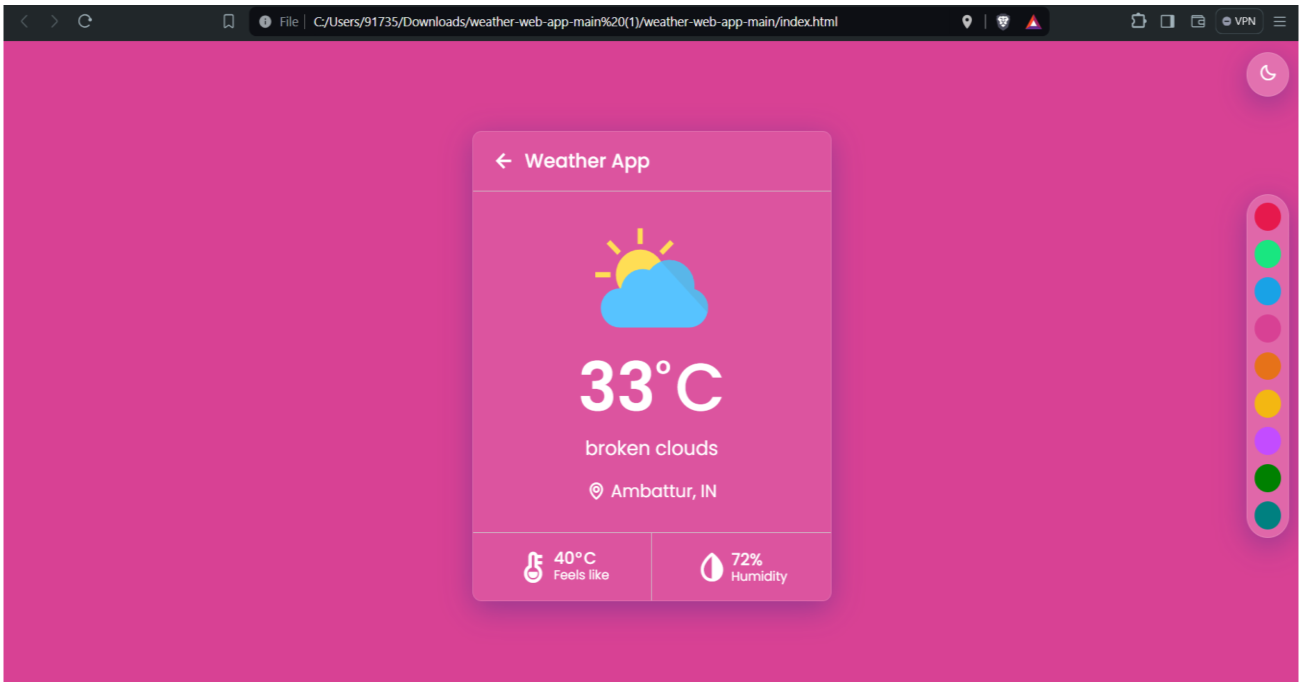

File
C:/Users/91735/Downloads/weather-web-app-main%20(1)/weather-web-app-main/index.html
VPN
Weather App
33°C
broken clouds
Ambattur, IN
40°C
Feels like
72%
Humidity

Fig 6.3 WEATHER FORECAST WITH GEO LOCATION

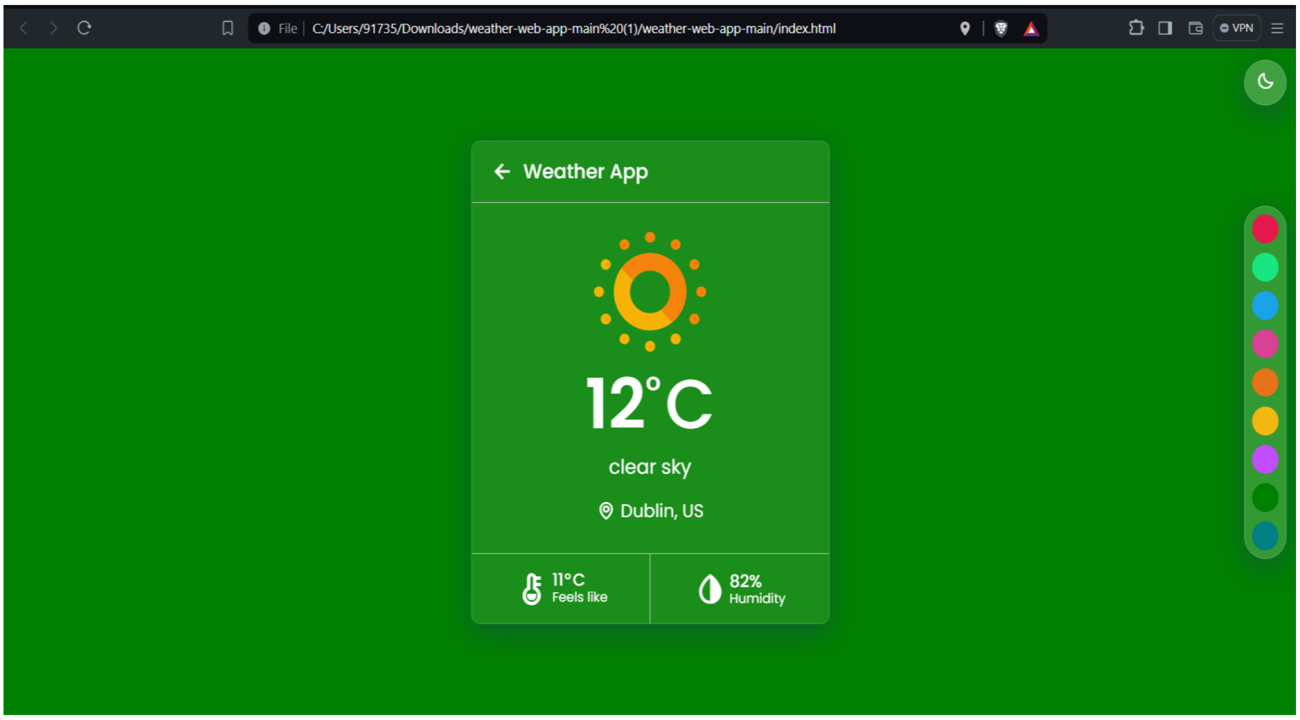

File
C:/Users/91735/Downloads/weather-web-app-main%20(1)/weather-web-app-main/index.html
VPN
Weather App
12°C
clear sky
Dublin, US
11°C
Feels like
82%
Humidity

Fig 6.4 WEATHER FORECAST FOR SPECIFIED LOCATION

CHAPTER 7
CONCLUSION

The "Weather Dashboard" project successfully integrates real-time weather data from the OpenWeather API into a user-friendly web application, fulfilling its primary objective of providing users with current weather conditions, forecasts, and detailed weather information for chosen locations. Utilizing search and geolocation functionalities, the application offers a seamless and intuitive user experience. Designed with clarity and accessibility in mind, the user interface ensures users can easily interact with the application to obtain necessary weather data. Throughout development, challenges such as asynchronous data fetching, API rate limits, and cross-browser compatibility were effectively managed through iterative development, rigorous testing, and adherence to best web development practices. Technologies like HTML, CSS, Bootstrap, and JavaScript were employed to create a responsive, aesthetically pleasing interface, while the OpenWeather API ensured reliable, up-to-date weather information.

The project also emphasizes robust error handling and user feedback mechanisms, informing users of issues like invalid location entries or network errors and guiding them to resolve these problems, thus enhancing usability and reliability. Looking ahead, future enhancements could include advanced features such as weather alerts, historical data, and user preferences for saving favorite locations, which would further augment functionality and user experience. The Weather Dashboard project has been a valuable learning experience, demonstrating the practical application of web development skills and API integration while emphasizing a user-centered design approach and the need for continuous improvement. Ultimately, this project not only achieves its goal of delivering real-time weather information but also establishes a strong foundation for future enhancements, showcasing the project's adaptability and potential for growth..

REFERENCES

[1] **Low-cost weather monitoring system with online logging and data visualization** Jess Christopher B. Lopez, Harreez M. Villaruz
https://www.researchgate.net/publication/304297354_Low-cost_weather_monitoring_system_with_online_logging_and_data_visualization

[2] **An Online Dashboard Platform for Weather Data of Major Sri Lankan Cities, and Global Climate Trends**
https://ieeexplore.ieee.org/document/10037514/authors#authors

[3] **OpenWeatherMap**
https://openweathermap.org/

[4]Indian meterogical department
https://mausam.imd.gov.in/chennai/