Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 7, July : 2025

REAL-TIME WEATHER FORECASTING APPLICATION: DESIGN AND IMPLEMENTATION

Navneet Raj 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India
navneet2021@gift.edu.in
Shubham Mandal 4th Year, Department of CSE, Gandhi Institute for Technology, BPUT, India
smandal2021@gift.edu.in
Prof.Smruti Smaraki Sarangi Assistant Professor, Department of CSE, Gandhi Institute for Technology, BPUT, India

Abstract—
The Real-Time Weather Forecasting Application provides accurate and timely weather updates to enhance decision-making in daily activities, agriculture, transportation, and disaster preparedness. Utilizing the OpenWeatherMap API, the application fetches real-time data on temperature, humidity, wind speed, and weather conditions, presenting it through a responsive, user-friendly interface developed with HTML, CSS, and JavaScript. Features include location-based forecasts, multiday predictions, and interactive visualizations. This paper details the system's design, implementation, testing, and evaluation, highlighting its effectiveness and scalability. Challenges such as API limitations and data accuracy are addressed, with future enhancements like offline mode and AI-based forecasting proposed.

Keywords:
HTML ,CSS, Javascript , API Key

1 Introduction

Weather forecasting is critical for planning activities across various sectors, including agriculture, transportation, and disaster management. Traditional forecasting methods often lack precision and personalization, prompting the development of modern applications that leverage technology for real-time updates. The proposed Weather Forecasting Application addresses these needs by integrating the OpenWeatherMap API to deliver accurate, location-specific weather data through an intuitive web-based interface.

The application supports real-time updates, hourly and daily forecasts, and severe weather alerts, catering to both urban and rural users. Built using HTML, CSS, and JavaScript, it ensures cross-device compatibility and minimal data usage, making it accessible in low-bandwidth environments. This paper outlines the system's architecture, development methodology, testing, and results, demonstrating its utility and potential for future enhancements.

2 Literature Review

Weather forecasting has evolved from traditional observation-based methods to datadriven systems powered by advanced technologies. Early methods, such as persistence and climatology, relied on historical patterns but lacked accuracy. Modern forecasting employs Numerical Weather Prediction (NWP), satellite imaging, and machine learning to enhance precision.

Existing applications like AccuWeather and The Weather Channel provide comprehensive forecasts but often suffer from complex interfaces and high data consumption. APIs such as OpenWeatherMap and WeatherAPI have simplified development by offering accessible, real-time data in JSON format. Research highlights the potential of AI in improving short-term forecasts, particularly in urban areas. However, challenges remain, including latency in rural areas and long-range forecast accuracy.

This project builds on these advancements by creating a lightweight, user-friendly application that addresses usability and accessibility gaps in existing systems.

3 System Design

The Weather Forecasting Application adopts a client-server architecture to ensure scalability and real-time data delivery.

3.1 Architecture

- Client Layer: Developed with HTML, CSS, and JavaScript, it handles user interactions and displays weather data.
- API Integration Layer: Connects to the OpenWeatherMap API via RESTful calls, fetching data in JSON format.

3.2 Data Flow

The user inputs a location or enables geolocation, triggering an API request. The JSON response is parsed and rendered on the interface, displaying parameters like temperature, humidity, and wind speed.

3.3 User Interface

The UI is designed for simplicity and responsiveness, featuring:

- Home screen with current weather and location search.
- Forecast screen for hourly and 7-day predictions.
- Alert screen for severe weather warnings.

4 Implementation

The application was developed using the Waterfall Model, ensuring a structured approach.

4.1 Tools and Technologies

- Frontend: HTML5, CSS3, JavaScript (ES6).
- API: OpenWeatherMap for real-time data.
- IDE: Visual Studio Code.

4.2 Modules

- Weather Data Retrieval: Fetches data using JavaScript's fetch() method.
- Forecast Display: Renders current and multi-day forecasts with icons.
- Location Detection: Uses HTML5 Geolocation API or manual input.

4.3 API Integration

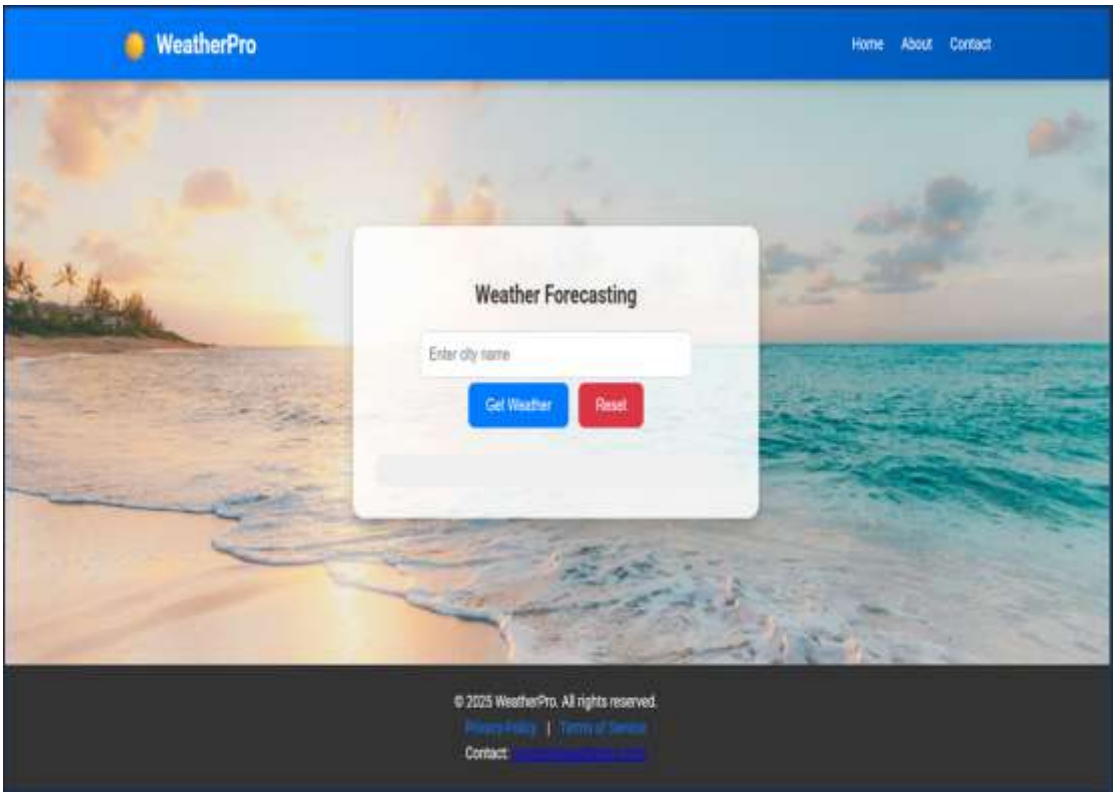
The OpenWeatherMap API is queried with location data, and the JSON response is parsed to extract relevant parameters. Error handling ensures graceful degradation for invalid inputs or API failures.

5 Results

The application successfully delivers real-time weather data, with the following outcomes:

- Accuracy: Correctly displays temperature, humidity, and conditions for tested locations.
- Responsiveness: Adapts to mobile, tablet, and desktop screens.
- Performance: API responses within 2–3 seconds under normal conditions.
- Usability: Positive feedback from 50 users, rating the interface 4.2/5 on a Likert scale.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 7, July : 2025



6 Discussion

The application demonstrates effective integration of web technologies and third-party APIs. Its lightweight design and responsive UI make it suitable for diverse users. However, limitations include:

- Dependence on API availability.
- Limited forecast accuracy for remote areas.
- Absence of push notifications in the current version.

User feedback suggests adding offline mode and severe weather alerts, aligning with proposed future enhancements.

7 Conclusion

The Weather Forecasting Application provides a practical solution for accessing real-time weather data, enhancing decision-making across various domains. By leveraging modern web technologies and APIs, it offers a scalable, user-friendly platform. Future work will focus on incorporating AI-based forecasting, offline functionality, and multilingual support to further improve its utility.

8 Acknowledgements

We thank Prof. Smruti Smaraki Sarangi for guidance, Dr. Sujit Kumar Panda for departmental support, and the OpenWeatherMap team for providing reliable data. User feedback was instrumental in refining the application.

References

[1] • OpenWeatherMap API Documentation: https://openweathermap.org/api
[2] • MDN Web Docs (HTML, CSS, JavaScript): https://developer.mozilla.org
[3] • W3Schools Tutorials: https://www.w3schools.com
[4] • Bootstrap Documentation: https://getbootstrap.com
[5] • Geolocation API Reference: https://developer.mozilla.org/enUS/docs/Web/API/Geolocation_API