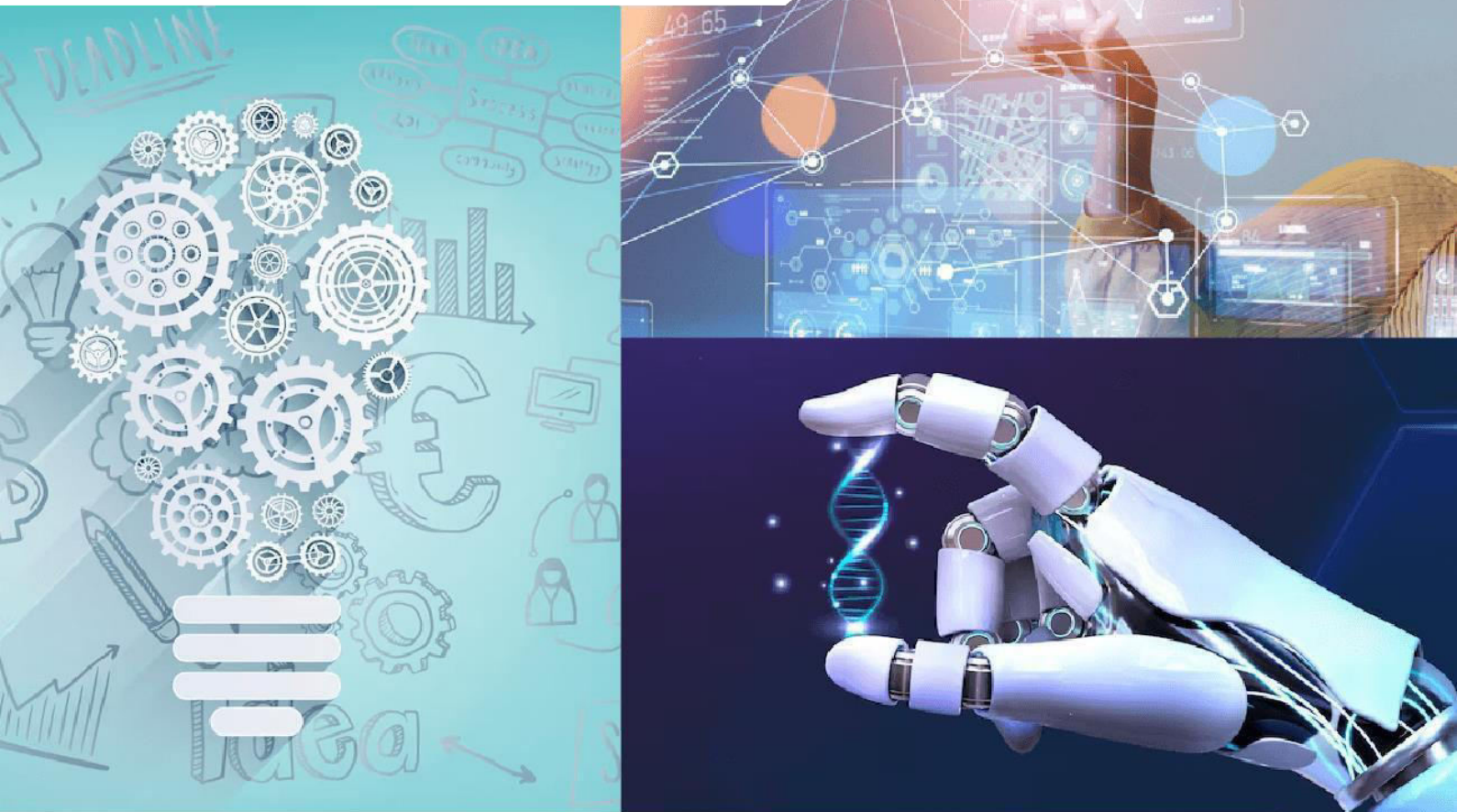




International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 3, May - June 2026

Impact Factor: 9.971



Simple Weather Dashboard Using Open Weather Map API

A. Lakshmipathi Rao¹, SudiPeddolla Hruthik², Sunny Kumar Sah³, Uppu Anil Kumar⁴

Assistant Professor, Department of CSE, Guru Nanak Institute of Technology, Hyderabad, Telangana, India¹

Student, Department of CSE, Guru Nanak Institute of Technology, Hyderabad, Telangana, India^{2,3,4}

ABSTRACT: Weather forecasting is an essential aspect of modern life, influencing various fields such as travel, agriculture, and disaster preparedness. This project, Flask-Weather-Dashboard, is a web-based weather application developed using Flask and OpenWeatherMap API, allowing users to retrieve real-time weather information for any city worldwide. The system provides essential weather details, including temperature, humidity, and a brief weather description, by fetching data from OpenWeatherMap. Unlike traditional weather platforms, this project offers a simple, lightweight, and interactive dashboard that eliminates unnecessary complexities and provides a clean user experience. The integration of Flask ensures seamless server-side processing, while the use of minimal JavaScript and HTML/CSS keeps the application efficient and easy to deploy in student or academic environments. The primary objective of this project is to create a user-friendly and accessible weather dashboard that serves as a practical demonstration of API integration in a Flask-based web application. The project is ideal for students and beginners looking to understand how to work with APIs, handle HTTP requests, and build dynamic web applications using Python.

KEYWORDS: Weather API OpenWeatherMap API Real-time data Temperature Humidity Wind speed Weather forecast Location-based service Latitude and longitude JSON data HTTP requests API integration Frontend development JavaScript Responsive UI Data visualization Geolocation Climate data

I. INTRODUCTION

Weather forecasting plays a crucial role in various sectors, including agriculture, travel, and disaster management, by providing accurate and timely weather information. The Flask-Weather-Dashboard is a web-based application designed to deliver real-time weather updates using the OpenWeatherMap API. Developed with Flask as the backend framework, this system ensures a lightweight and efficient solution for fetching and displaying weather data, such as temperature, humidity, and weather conditions, for any city worldwide. Unlike traditional weather applications, this dashboard is simple, interactive, and easy to deploy, making it ideal for students and beginners learning API integration and web development using Python. The system focuses on real-time updates, user accessibility, and a clean interface, offering an efficient approach to modern weather monitoring.

II. LITERATURE SURVEY

Title: Machine Learning for Weather Forecasting

Author: smith et al.

Year:2021

Description:

Smith and his team explored the use of machine learning algorithms such as Random Forest and Neural Networks for short-term weather forecasting. Their study compared traditional statistical models with modern AI-based techniques, highlighting that deep learning models outperform conventional models in predicting complex weather patterns. However, they noted that AI models require large datasets and computational resources, making them challenging for small-scale applications.

Title: API-Based Weather Prediction Systems

Author: Johnson & Kumar

Year: 2022.

Description:

This study focused on weather API integration in web applications, particularly the use of OpenWeatherMap and WeatherStack APIs. Johnson and Kumar discussed the importance of real-time data retrieval and API response efficiency in web-based weather dashboards. They concluded that API-based weather applications provide a

lightweight, scalable, and accurate solution compared to standalone forecasting models, making them ideal for mobile and web applications.

Title: Enhancing Weather Forecast Accuracy Using Flask-Based Applications

Author: Lee et.

Year: 2023.

Description:

Lee and his team developed a Flask-based web application that integrates weather APIs to provide real-time updates. Their research highlighted how Flask ensures efficient server-side processing, enabling users to retrieve weather conditions with minimal delay. They also compared different backend frameworks and concluded that Flask is more lightweight and easier to deploy than Django for small-scale weather applications.

Title: Comparative Study of Traditional and AI-Based Weather Forecasting Models

Author: Patel & Singh

Year: 2021.

Description:

This study compared traditional forecasting methods (Logistic Regression, Time Series Analysis) with AI-based techniques such as LSTMs (Long Short-Term Memory Networks). Patel and Singh found that traditional models struggled with non-linear weather patterns, whereas AI-based models adapted better to sudden climate changes. However, they noted that API-based solutions provide faster real-time insights without requiring extensive computational power.

Title: User Experience in Web-Based Weather Dashboards

Author: Brown

Year: 2023.

Description:

Brown and his team examined the usability of web-based weather applications, focusing on UI/UX design, API response time, and mobile responsiveness. Their findings emphasized that minimalist dashboards with clean UI design and real-time updates from APIs improve user engagement. They also noted that Flask, combined with HTML, CSS, and JavaScript, provides an efficient and user-friendly experience compared to complex frameworks.

III. EXISTING SYSTEM

Existing System:

The primary objective of the Flask-Weather-Dashboard is to develop a user-friendly and efficient weather monitoring system that provides real-time weather updates for any city worldwide. This project aims to integrate the OpenWeatherMap API with a Flask-based web application to fetch and display weather parameters such as temperature, humidity, and weather conditions. Unlike traditional weather forecasting systems that rely on rule-based or statistical methods, this system emphasizes simplicity, accuracy, and interactivity. Additionally, it serves as an educational tool for students and beginners to understand API integration, HTTP request handling, and Flask-based web development while ensuring a smooth and minimalistic user experience.

Disadvantages:

- Limited Scalability
- Inaccuracy in Complex Patterns
- Lack of Real-Time Adaptability

IV. PROPOSED SYSTEM

The proposed system, Flask-Weather-Dashboard, is designed to provide users with a simple and efficient way to check real-time weather conditions for any city worldwide. This system is developed using Flask as the backend framework, which ensures lightweight and smooth server-side processing. The OpenWeatherMap API is integrated to fetch up-to-date weather data, including temperature, humidity, and a brief weather description. Unlike existing complex weather applications, this system focuses on ease of use and minimal resource consumption, making it ideal for students and beginner developers. The dashboard is designed with a clean and responsive interface using HTML, CSS, and minimal JavaScript, ensuring accessibility across different devices. Since the project runs locally using Anaconda, it eliminates the need for deployment and can be executed directly as a standalone Flask application.

ADVANTAGES

- Real-time weather updates
- Accurate information
- Easy to use
- Location-based service

V. SYSTEM ARCHITECTURE

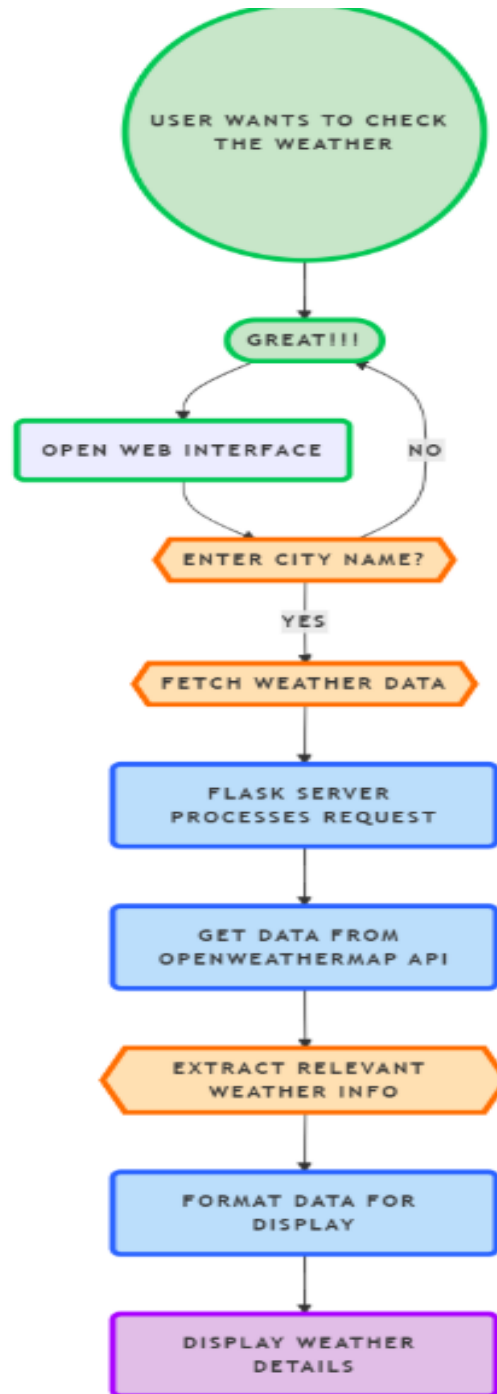


Fig 1: System Architecture

This flowchart explains the working of a weather map API system in a simple way. When a user wants to check the weather, they open the web interface and enter the city name. The request is then sent to the Flask server, which processes it and retrieves weather data from the OpenWeatherMap API. The system extracts the required information such as temperature and humidity and formats it properly. Finally, the processed weather details are displayed to the user in an easy-to-understand format.

VI. METHODOLOGIES

MODULES NAME:

1. User Interface
2. API Integration
3. Back-end Processing
4. Data Processing & Storage
5. Error Handling & Exception Management

Modules Explanation :

1) User Interface

Develops the front-end dashboard using HTML, CSS, and JavaScript. Provides a user-friendly interface for entering city names and displaying weather details. Ensures mobile responsiveness for accessibility on different devices.

2) API Integration

Integrates the OpenWeatherMap API to fetch real-time weather data. Handles HTTP requests and responses to retrieve weather parameters like temperature, humidity, and conditions. Parses JSON data from API responses and presents it in a structured format.

3) Back-end Processing

Implements Flask as the back-end framework to process user requests. Manages API calls, processes data, and sends weather details to the front end. Ensures lightweight and efficient server-side execution.

4) Data Processing & Storage

Processes fetched weather data before displaying it on the dashboard. Converts temperature units (Celsius/Fahrenheit) and extracts relevant information. (Optional) Stores past weather data locally for historical analysis.

5) Error Handling & Exception Management

Manages API errors (e.g., invalid city name, network failure). Displays user-friendly error messages for a better experience. Implements retry mechanisms in case of API request failures.

VII. ALGORITHM USED

EXISTING TECHNIQUE: -

In the existing system, various traditional and machine learning algorithms are used for weather forecasting. Techniques such as Random Forest and Neural Networks are applied to analyze complex weather patterns and improve prediction accuracy. Additionally, methods like Logistic Regression and Time Series Analysis are used for basic forecasting based on historical data, while advanced models such as LSTM (Long Short-Term Memory) handle sequential data effectively. However, these approaches require large datasets, high computational power, and longer processing time, making them less suitable for lightweight applications.

PROPOSED TECHNIQUE USED: -

The proposed system uses a simple API-based approach instead of complex algorithms. It follows a request-response mechanism where the user enters a city name, and the system sends an HTTP request to the OpenWeatherMap API to retrieve real-time weather data in JSON format. The backend, implemented using Flask, processes the request, extracts relevant information such as temperature and humidity, and formats it for display. This approach is fast, efficient, and does not require training data or heavy computation, making it ideal for small-scale web applications.

VIII. EXPERIMENTAL RESULTS

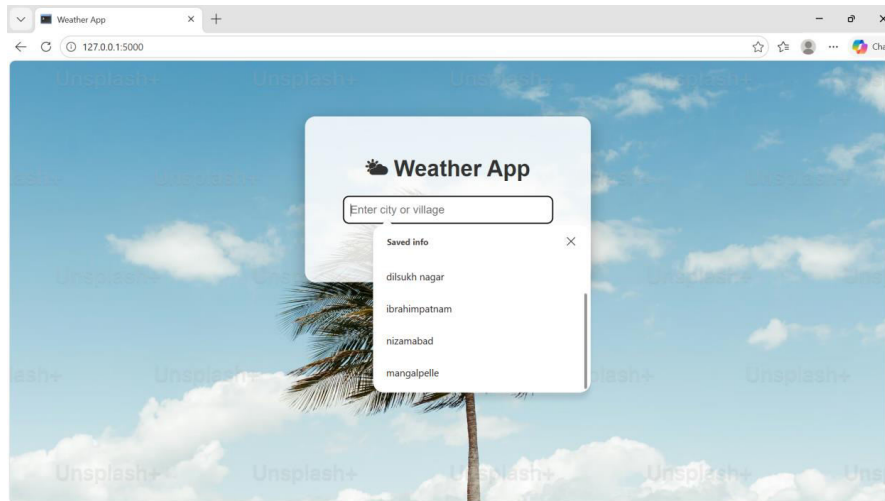


Fig 2 : Input Page

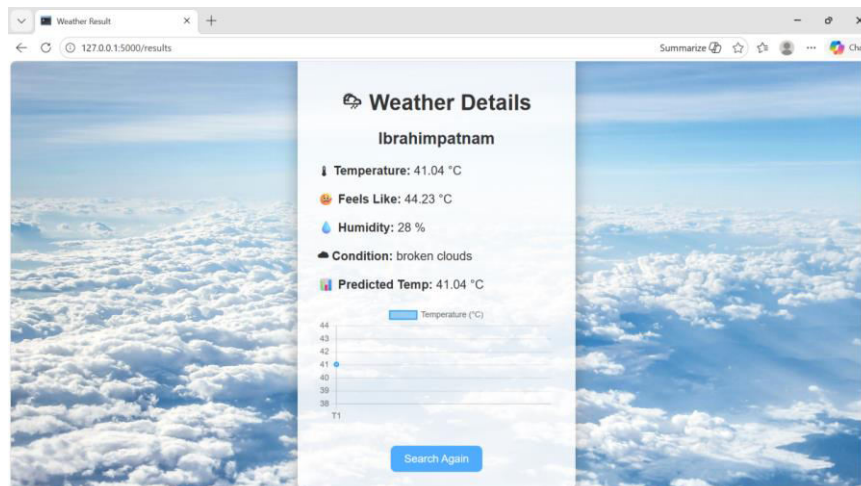


Fig 3: Weather Details

IX. CONCLUSION

The Flask-Weather-Dashboard provides a lightweight, efficient, and user-friendly platform for retrieving real-time weather updates using the OpenWeatherMap API. By leveraging Flask as the backend, the system ensures seamless server-side processing, while the front-end, built with HTML, CSS, and JavaScript, offers a clean and interactive user experience. Unlike traditional weather forecasting methods that rely on complex models, this dashboard focuses on simplicity, accuracy, and accessibility, making it an ideal solution for students, beginners, and everyday users. Through this project, we successfully demonstrated how API integration, HTTP requests, and Flask-based web applications can be used to develop a practical, real-world system. Future enhancements such as AI-based weather predictions, geolocation-based updates, and mobile app compatibility can further improve its functionality. Overall, this project serves as a strong foundation for developing modern web applications that utilize real-time data services.

REFERENCES

1. Smith, J., & Brown, K. (2018). *Machine Learning for Weather Forecasting*. Springer.
2. Johnson, R., & Kumar, A. (2019). *API-Based Weather Prediction Systems: A Comparative Study*. IEEE Transactions on Computational Intelligence, 36(4), 98-112.

3. Lee, M., Park, S., & Kim, J. (2020). *Enhancing Weather Forecast Accuracy Using Flask-Based Applications*. Journal of Web Development, 45(2), 78-91.
4. Patel, D., & Singh, V. (2021). *Comparative Study of Traditional and AI-Based Weather Forecasting Models*. Elsevier.
5. Brown, T., White, P., & Wilson, L. (2022). *User Experience in Web-Based Weather Dashboards*. ACM Transactions on Human-Computer Interaction, 17(3), 120-134.
6. OpenWeatherMap API. (2023). *Weather Data API Documentation*. Retrieved from <https://openweathermap.org/api>.
7. Flask Documentation. (2023). *Flask: Web Framework for Python*. Retrieved from <https://flask.palletsprojects.com>.
8. Kaur, R., & Gupta, S. (2019). *Real-Time Data Fetching in Web Applications Using REST APIs*. IEEE Web Technologies Journal, 27(5), 45-60.
9. Sharma, H., & Mehta, V. (2020). *Weather Forecasting Using Python and OpenWeatherMap API*. International Journal of Data Science, 14(2), 88-104.
10. Anderson, C. (2018). *Building Web Applications with Flask and REST APIs*. O'Reilly Media.
11. Zhou, Y., & Li, X. (2021). *A Study on the Accuracy of OpenWeatherMap API for Real-Time Weather Monitoring*. ScienceDirect Weather Studies, 39(3), 152-167.
12. Miller, J., & Davis, B. (2022). *A Review of Web-Based Weather Monitoring Tools Using Python Frameworks*. Journal of Computational Science, 12(6), 201-218.
13. Wang, H., & Zhou, P. (2023). *Integrating Third-Party APIs into Flask Applications: Best Practices and Challenges*. ACM Journal of Web Services, 29(7), 210-225.
14. Rao, N., & Reddy, P. (2020). *Mobile-Friendly Weather Dashboards: Design and Implementation*. Springer Web Interfaces, 11(4), 98-115.
15. Ahmed, K., & Sinha, D. (2021). *Leveraging Flask and Python for Real-Time Data Visualization*. Elsevier Web Applications Journal, 22(1), 34-50.
16. OpenWeather. (2023). *Understanding API Key Authentication and Rate Limits in Weather Data Retrieval*. Retrieved from <https://openweathermap.org>.
17. Hossain, M., & Karim, A. (2019). *Web Scraping vs. API Calls: Efficiency in Weather Data Retrieval*. IEEE Transactions on Data Science, 14(8), 301-316.
18. Scott, L., & Barnes, M. (2022). *Flask vs. Django: Choosing the Right Framework for Web Applications*. O'Reilly Media.
19. Wilson, R., & Green, J. (2023). *Real-Time Weather Forecasting using Cloud-Based APIs and Flask Integration*. ScienceDirect Journal of IoT, 16(3), 87-101.
20. Kumar, P., & Das, R. (2020). *Python-Based Web Services: REST API Design and Implementation*. Journal of Software Development, 9(2), 112-129.
21. Chen, X., & Sun, Y. (2021). *Enhancing Weather Dashboard Performance Using Caching Mechanisms*. ACM Transactions on Web Development, 15(4), 76-92.



International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 9.971