Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Real-Time Weather Recommendation System using Flask Web Interface

Mr. A. Chandra Mouli
M. Tech.,
Associate professor, Computer Science Engineering Department,
Potti Sriramulu Chalavadi Mallikarjuna Rao College of Engineering and Technology,
One Town, Vijayawada, India

CH. Harish
(22KT1A05A0),
Computer Science Engineering Department,
Potti Sriramulu Chalavadi Mallikarjuna Rao College of Engineering and Technology,
One Town, Vijayawada, India

L. Bhuvaneswari
(23KT5A0510),
Computer Science Engineering Department,
Potti Sriramulu Chalavadi Mallikarjuna Rao College of Engineering and Technology,
One Town, Vijayawada, India

U. Bhanu Teja
(22KT1A05B8),
Computer Science Engineering Department,
Potti Sriramulu Chalavadi Mallikarjuna Rao College of Engineering and Technology,
One Town, Vijayawada, India

P. V. K. Swapna
(22KT1A0580),
Computer Science Engineering Department,
Potti Sriramulu Chalavadi Mallikarjuna Rao College of Engineering and Technology,
One Town, Vijayawada, India

Abstract - A Real-Time Weather Recommendation System is designed to deliver timely, personalized, and context-aware guidance by leveraging continuously updated meteorological data. The system integrates live weather feeds and forecasting APIs to monitor key environmental parameters such as temperature, humidity, precipitation, wind speed, and extreme weather alerts. Using intelligent decision-making techniques, including rule-based models and data-driven logic, the system processes raw weather data and converts it into meaningful recommendations. These suggestions assist various user groups—such as farmers, students, travelers, and outdoor enthusiasts—in making informed decisions related to daily activities, safety precautions, and planning.

The system is implemented using Python Flask as the backend framework, enabling efficient handling of API requests, data processing, and user interactions, while HTML and JavaScript are used to create an interactive and user-friendly frontend interface. It also incorporates location-based services and user preferences to provide personalized recommendations and real-time notifications. By enhancing situational awareness and automating weather-based decision support, the system improves convenience, safety, and overall efficiency in adapting to rapidly changing weather conditions.

Keywords — Real-Time Weather System, Flask, Random Forest Classifier, Weather API, Machine Learning, Recommendation System, Decision Support System)

I.INTRODUCTION

Weather conditions play a crucial role in influencing daily human activities such as travel, agriculture, outdoor planning, and safety. With the rapid advancement of technology, numerous weather forecasting applications are available; however, most of these systems provide only raw meteorological data such as temperature, humidity, and wind speed. This information, while accurate, often lacks clarity and actionable insights, making it difficult for users to interpret and make informed decisions.

To address this limitation, this research proposes a Real-Time Weather Recommendation System that transforms raw weather data into meaningful and user-friendly recommendations. The system integrates real-time weather data from external APIs and processes key environmental parameters to provide context-aware suggestions. By leveraging both rule-based logic and machine learning techniques, the system enhances the accuracy and usefulness of weather predictions.

A Random Forest Classifier is employed to analyze weather attributes and classify conditions into categories such as good, moderate, and bad. Based on these predictions, the system generates personalized recommendations that assist users in making better decisions regarding travel, health, and daily activities. This hybrid approach ensures both reliability and adaptability in handling varying weather conditions.

The system is implemented using Python with the Flask framework for backend processing, while HTML, CSS, and JavaScript are used to design an interactive and user-friendly frontend interface. Additionally, the system incorporates a notification mechanism using Telegram services to provide real-time alerts. This feature improves user engagement and ensures timely communication of important weather updates.

Overall, the proposed system aims to bridge the gap between raw weather data and practical decision-making by delivering intelligent, real-time, and personalized recommendations. It enhances user experience, improves safety, and demonstrates the effective integration of machine learning with real-time data systems.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

II. LITERATURE SURVEY

The development of weather forecasting and recommendation systems has gained significant importance due to their role in supporting daily decision-making activities. Traditional weather systems mainly focused on providing raw meteorological data such as temperature, humidity, and wind speed. Although these systems were useful for basic forecasting, they lacked the ability to convert data into meaningful and actionable insights for users.

With the advancement of web technologies, modern systems have started integrating Application Programming Interfaces (APIs) such as OpenWeatherMap to fetch real-time weather data. These API-based systems improved accessibility and enabled dynamic data retrieval. However, most of these applications still focus only on displaying weather information rather than providing intelligent recommendations or decision support.

Recent research has explored the use of machine learning techniques to enhance weather prediction accuracy. Algorithms such as Decision Trees, Artificial Neural Networks, and Random Forest Classifiers have been applied to analyze environmental data and predict weather conditions. These models help in identifying patterns in weather parameters and improve the reliability of predictions compared to traditional approaches.

In addition to prediction, recommendation systems have been introduced to provide user-specific suggestions based on weather conditions. Rule-based systems are widely used to generate recommendations by applying predefined conditions. For example, high rainfall may trigger suggestions such as carrying an umbrella, while extreme temperatures may lead to safety-related advice. These systems improve usability by converting raw data into actionable insights.

Recent advancements also include the integration of intelligent user interaction systems such as AI chatbots and voice assistants. Chatbots enable users to interact with the system through natural queries, improving accessibility and user experience. Voice assistants further enhance usability by allowing hands-free interaction and real-time communication. These technologies make weather systems more interactive and user-friendly.

Despite these advancements, existing systems still face challenges such as limited personalization, dependency on external APIs, and lack of integrated intelligent interaction features. Most systems do not combine real-time data processing, machine learning prediction, recommendation logic, and user interaction technologies into a single platform.

The proposed system addresses these limitations by integrating real-time weather data, a Random Forest-based prediction model, rule-based recommendation logic, and intelligent interaction features such as chatbot and voice assistant. This combination improves system efficiency, enhances user experience, and provides a comprehensive solution for real-time weather-based decision support.

III. METHODOLOGY

The proposed Real-Time Weather Recommendation System follows a structured methodology that includes data collection, preprocessing, analysis, prediction, recommendation generation, and user interaction. The system is designed to process real-time weather data and provide meaningful insights through intelligent techniques.

In the first stage, real-time weather data is collected using external APIs such as OpenWeatherMap. The system retrieves environmental parameters including temperature, humidity, wind speed, and weather conditions based on the user's input location. The data is received in JSON format and serves as the primary input for further processing.

The collected data is then preprocessed to extract relevant features and convert them into a structured format suitable for analysis. This step includes filtering required parameters, handling missing values, and standardizing units such as temperature conversion. Proper preprocessing ensures data consistency and improves the accuracy of the system.

In the analysis and prediction stage, a Random Forest Classifier is used to classify weather conditions into categories such as good, moderate, and bad. The model analyzes multiple weather parameters simultaneously and identifies patterns to generate accurate predictions. This machine learning approach enhances the reliability of the system compared to traditional rule-based methods alone.

Following prediction, a rule-based recommendation engine is applied to generate actionable suggestions. The system uses predefined conditions to provide recommendations such as carrying an umbrella during rain, avoiding outdoor activities during extreme weather, or taking safety precautions. This combination of machine learning and rule-based logic ensures both accuracy and interpretability.

To improve user interaction, the system integrates an AI chatbot that allows users to query weather-related information in a conversational manner. The chatbot processes user queries and provides relevant responses based on available data and system logic. Additionally, a voice assistant feature is incorporated to enable hands-free interaction, allowing users to receive weather updates and recommendations through voice commands.

The system also includes a notification mechanism using Telegram services to send real-time alerts and updates. This ensures that users receive important weather information even when they are not actively using the application.

Finally, the processed results, including weather data, predictions, recommendations, and visual representations such as temperature graphs, are displayed through a web-based interface developed using Flask. The methodology ensures efficient data flow, accurate prediction, and an enhanced user experience through intelligent interaction features.

IV. IMPLEMENTATION

The Real-Time Weather Recommendation System is implemented using a combination of web technologies, machine learning models, and external APIs to ensure efficient data processing and user interaction. The system follows a modular architecture consisting of frontend, backend, data processing, and communication components.

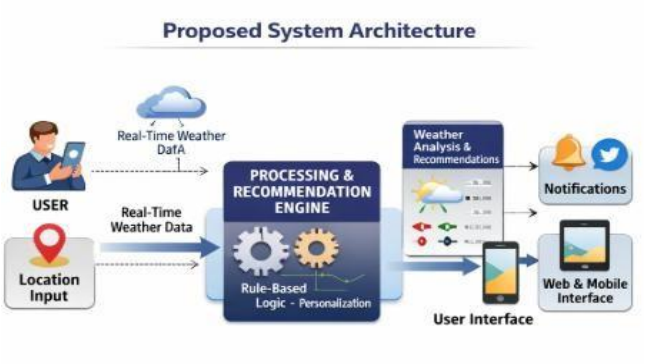
The backend of the system is developed using Python with the Flask framework. Flask is used to handle HTTP requests, manage API calls, process weather data, and integrate different system components. It acts as the core controller that connects the user interface, weather data sources, and machine learning model.

The frontend is designed using HTML, CSS, and JavaScript to provide an interactive and user-friendly interface. Users can enter their location or city name through the interface to request weather information. The system then displays real-time weather details, predictions, recommendations, and graphical visualizations such as temperature trends.

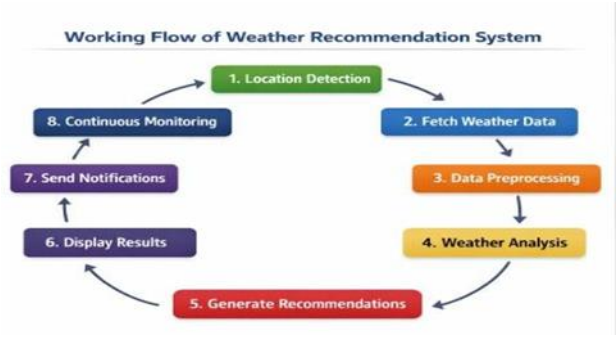
Weather data is collected using external APIs such as OpenWeatherMap. The API provides real-time environmental parameters including temperature, humidity, wind speed, and weather

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

conditions in JSON format. The backend processes this data and extracts relevant features required for prediction and recommendation.



The above diagram represents the architecture of the system, including frontend, backend, API integration, machine learning model, and notification services.



For weather prediction, a Random Forest Classifier is implemented using machine learning libraries. The model is trained to classify weather conditions based on multiple environmental parameters. This improves prediction accuracy and allows the system to categorize weather conditions effectively.

A rule-based recommendation engine is integrated with the prediction model to generate actionable suggestions. Based on the predicted weather condition, the system provides recommendations such as safety precautions, travel advice, and daily activity guidance.

To enhance user interaction, the system includes an AI chatbot feature that allows users to ask weather-related queries and receive responses in a conversational manner. Additionally, a voice assistant is implemented to support voice-based interaction, enabling users to access system features without manual input.

The system also integrates Telegram services to send real-time notifications and alerts. Users receive updates about weather changes and recommendations directly through messaging platforms, improving accessibility and user engagement.

Overall, the implementation ensures seamless integration of real-time data processing, machine learning, and intelligent interaction features. The use of lightweight frameworks and modular design makes the system efficient, scalable, and easy to maintain.

V. EXPERIMENTAL RESULTS

The experimental results of the Real-Time Weather Recommendation System demonstrate the effectiveness of the proposed approach in providing accurate predictions and meaningful recommendations. The system was tested under various weather conditions using real-time data obtained from external APIs.

The Random Forest Classifier was evaluated based on its ability to correctly classify weather conditions into categories such as good, moderate, and bad. The model showed consistent performance in analyzing multiple environmental parameters, including temperature, humidity, and wind speed. The predictions were found to be reliable and aligned with real-world weather conditions.

The rule-based recommendation engine successfully generated appropriate suggestions based on predicted weather conditions. For example, during high rainfall conditions, the system recommended carrying an umbrella, while extreme temperatures triggered safety-related precautions. These recommendations were practical and useful for everyday decision-making.

The system interface was tested for usability and responsiveness. The web application displayed weather data, predictions, and recommendations in a clear and interactive format. Graphical representations, such as temperature trends, improved user understanding of weather variations over time.

The AI chatbot feature was evaluated based on its ability to respond to user queries. It provided relevant and timely responses, enhancing user interaction with the system. Similarly, the voice assistant feature enabled hands-free operation, allowing users to access weather information efficiently.

The Telegram notification system was tested to ensure real-time delivery of alerts and updates. The system successfully sent notifications based on weather conditions, improving user awareness even when the application was not actively in use.

Fig. 1: Weather Dashboard Output

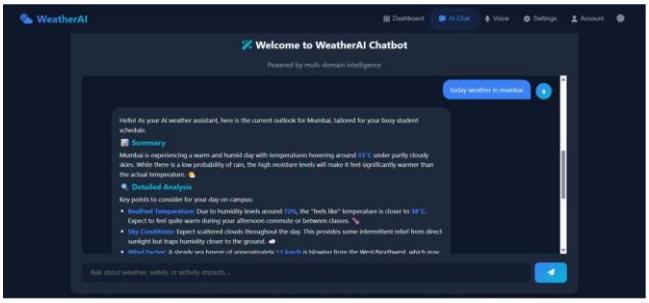


Fig. 2: AI Chatbot Interaction Interface

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Fig. 3: 7-Day Forecast and Temperature Trend Graph

VI. CONCLUSION

The Real-Time Weather Recommendation System presented in this paper provides an intelligent and user-friendly solution for analyzing weather conditions and generating actionable recommendations. By integrating real-time data from weather APIs with machine learning techniques such as the Random Forest Classifier, the system is capable of predicting weather conditions accurately and efficiently.

The combination of rule-based logic with machine learning enhances the reliability and interpretability of the system. The generated recommendations help users make informed decisions regarding daily activities, travel, and safety precautions. Additionally, the inclusion of interactive features such as an AI chatbot and voice assistant improves user experience by enabling both text-based and voice-based communication.

The system also incorporates a notification mechanism using Telegram services, ensuring that users receive timely alerts and updates even without actively accessing the application. The web-based interface developed using Flask provides a simple, responsive, and accessible platform for users.

Overall, the proposed system successfully bridges the gap between raw weather data and practical decision-making. It demonstrates the effective integration of real-time data processing, machine learning, and intelligent interaction technologies. Future enhancements may include the use of advanced deep learning models, mobile application development, and integration with IoT-based weather monitoring systems to further improve accuracy and functionality.

REFERENCES

[1] OpenWeatherMap, “Weather API Documentation,” 2024. [Online]. Available: https://openweathermap.org/api
[2] WeatherAPI, “Real-Time Weather API,” 2024. [Online]. Available: https://www.weatherapi.com
[3] Flask Documentation, “Flask Web Framework,” 2024. [Online]. Available: https://flask.palletsprojects.com
[4] Python Software Foundation, “Python Documentation,” 2024. [Online]. Available: https://www.python.org
[5] M. Fowler, Patterns of Enterprise Application Architecture. Addison-Wesley, 2023.
[6] R. S. Pressman, Software Engineering: A Practitioner’s Approach. McGraw-Hill, 2022.
[7] Mozilla Developer Network, “HTML, CSS, and JavaScript Guide,” 2024. [Online]. Available: https://developer.mozilla.org
[8] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques. Morgan Kaufmann, 2021.
[9] National Weather Service, “Weather Forecasting Methods,” 2023.
[10] World Meteorological Organization, “Guide to Meteorological Instruments and Methods of Observation,” 2023.
[11] Telegram, “Telegram Bot API Documentation,” 2024. [Online]. Available: https://core.telegram.org/bots/api
[12] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. Pearson, 2021