



FARMER WEATHER ALERT SYSTEM: A REAL-TIME WEATHER NOTIFICATION PLATFORM

¹Mr.Abhishek Rai, ¹Mr.Ashwaj H S, ¹Mr.Deepak Krishna M, ¹Mr.Likhith Rai, ²Dr.Sandeep.Bhat

¹3rd year Student, ²Professor

¹Department of Computer Science and Engineering,
¹Srinivas Institute of Technology, Mangaluru, Karnataka, India

Abstract : This paper presents a web-based Farmer Weather Alert System designed to deliver timely and localized weather warnings to rural farmers. Agriculture is critically dependent on weather, and unforeseen meteorological events such as storms, excessive rainfall, or high winds can severely damage crops and infrastructure. Hence, a reliable and proactive alert system is imperative for empowering farmers to make informed decisions. The system integrates a Python Flask web backend with the OpenWeatherMap API to retrieve real-time weather conditions and Twilio's communication API to send SMS and voice alerts. The platform supports bilingual communication (English and Kannada), ensuring inclusivity across rural Karnataka. Alerts are triggered when predefined weather thresholds are crossed, and all communications are logged for transparency. The system includes a web-based admin interface to manage farmer profiles, update alert thresholds, and review notification history. With modular design and cost-effective implementation, this solution can be readily adapted for broader agricultural and disaster management applications.

I. INTRODUCTION

Agriculture is the backbone of many economies, especially in developing countries like India, where a large portion of the population depends on farming for their livelihood. Weather conditions significantly influence agricultural productivity, and timely weather information is crucial for farmers to make decisions related to sowing, irrigation, pesticide application, and harvesting. However, in recent years, unpredictable and extreme weather patterns—such as unseasonal rainfall, cyclones, droughts, and high wind speeds—have intensified due to climate change. These changes have led to frequent crop failures, livestock losses, and financial hardship for farmers, many of whom lack access to timely and localized weather updates.

Conventional methods of weather broadcasting, such as television and radio, often fail to reach remote rural regions promptly or in a language that farmers understand. Even when available, the information is often too general, making it difficult for farmers to take location-specific precautions. The digital divide further widens this gap, as many farmers are not equipped to access mobile apps or internet-based services.

To address these challenges, we propose the **Farmer Weather Alert System**, a real-time, bilingual weather alert platform that sends SMS and voice alerts directly to farmers' mobile phones. Built using Python, Flask, Twilio API, and OpenWeatherMap API, the system ensures timely delivery of alerts regarding high-impact weather events. The use of Excel for data management keeps the system lightweight and accessible for administrators without a technical background. The bilingual feature, supporting both English and Kannada, ensures that critical information is understood by a diverse farmer population.

This project represents an innovative intersection of agriculture and information technology, aimed at making weather forecasting more actionable and inclusive. By empowering farmers with early warnings, the system not only helps protect crops and livestock but also contributes to sustainable agricultural practices and food security.

II. PROBLEM STATEMENT

2.1 Agricultural Vulnerability to Weather Extremes

Agriculture in India, particularly in rural regions, is deeply reliant on seasonal weather conditions. However, with the increasing unpredictability of climatic patterns due to global climate change, farmers are exposed to a wide range of meteorological risks. These include unseasonal rains, prolonged dry spells, hailstorms, heatwaves, and cyclonic winds. Such weather anomalies can lead to drastic consequences, such as delayed sowing, waterlogging, pest outbreaks, reduced crop yield, and post-harvest losses. The lack of timely intervention often results in irreversible damage to crops and significant economic setbacks to farmers, who are frequently dependent on a single harvest for their livelihood.

2.2 Limitations of Conventional Forecast Dissemination

Despite advancements in meteorological science, the mode of weather forecast dissemination remains a critical bottleneck. Government forecasts are often communicated via television, newspapers, or radio broadcasts, which may not reach farmers in time or with sufficient localization. Additionally, these channels generally deliver generalized information over large geographic regions, making them ineffective for micro-level agricultural decision-making. Furthermore, illiteracy and language diversity in rural India limit the accessibility and comprehension of these broadcasts. There is an urgent need for a solution that delivers localized, real-time weather alerts directly to farmers in a language and format they can easily understand.

2.3 Digital Divide and Technological Barriers

Many rural communities are still affected by the digital divide — the gap in access to modern information and communication technologies. While mobile phone penetration has improved significantly, internet literacy remains low, especially among elderly and economically disadvantaged farmers. Thus, expecting farmers to install and use mobile apps or access web portals to monitor weather conditions is impractical in many cases. Therefore, any viable weather alert system must be able to deliver essential information using simple technologies such as SMS and voice calls, which are widely used and understood.

2.4 Objective and Scope of the Solution

To address the above challenges, the primary objective of this project is to design and develop a weather alert platform tailored specifically for farmers. The platform must be capable of pulling real-time meteorological data from reliable APIs, evaluating the data against user-defined thresholds, and triggering alerts when conditions exceed safe limits. These alerts must be delivered in the local language (Kannada), via SMS and voice calls, ensuring inclusivity and accessibility. The system should also allow for administrator control through a web interface, where threshold settings can be configured, user data managed, and alert logs reviewed. Such a system would bridge the communication gap between meteorological data providers and end-user farmers, offering a scalable and replicable model for rural agritech solutions.

III. SYSTEM ARCHITECTURE AND DESIGN

3.1 System Overview

The Farmer Weather Alert System follows a modular and layered architecture designed to promote separation of concerns, scalability, and ease of integration. The overall design consists of three distinct layers: the presentation layer (frontend), the application logic layer (backend), and the data management layer. These layers work together to fetch weather data, assess potential weather hazards based on predefined thresholds, and notify registered farmers via SMS or voice call. The system targets agricultural administrators and rural development officers who are responsible for managing the registration of farmers and monitoring regional weather forecasts. These users access the system through a web interface, while the farmers, who are the primary beneficiaries, receive the alerts on their mobile devices in their preferred language. The architecture emphasizes automation, inclusivity, and ease of use, making it appropriate for deployment even in areas with limited technical infrastructure.

3.2 Frontend Design

The frontend component is implemented as a web interface using HTML5, CSS3, JavaScript, and Flask's templating engine. It provides administrators with intuitive access to the main system functionalities, including farmer data registration, threshold configuration, alert log review, and real-time weather monitoring. The design ensures responsiveness so that it can be accessed on various devices such as laptops, tablets, and smartphones. The homepage includes navigational links to all major modules and displays summarized weather and alert information. The user registration module enables the input and editing of farmer details including name, contact number, geographical location, and language preference. A separate interface allows the administrator to define and modify the threshold limits for various weather parameters like wind speed and rainfall. The system also includes a real-time dashboard that presents weather information retrieved from external APIs, ensuring that administrators can monitor the status across different farming zones. An alert log page provides access to the historical data of alerts sent, including timestamps, recipients, and types of notification. To maintain data integrity and usability, the frontend employs form validation techniques and user feedback prompts during data entry and configuration.

3.3 Backend Design

The backend of the system is developed using Flask, a lightweight Python web framework that supports modular design and rapid deployment. It acts as the operational core of the application, orchestrating interactions between the frontend, the external APIs, and the data repository. At regular intervals, the backend fetches current weather data using the OpenWeatherMap API, based on the geographical coordinates associated with each farmer. The retrieved data includes various weather parameters such as temperature, humidity, precipitation levels, and wind speed. This data is evaluated against the administrator-defined thresholds. If any parameter exceeds the acceptable limit, the backend dynamically generates an alert message tailored to the severity and nature of the condition.

Twilio's communication API is integrated within the backend to facilitate the dispatch of notifications. Depending on the farmer's preferences and urgency of the situation, the alert is sent either as an SMS or as a voice call. The backend supports multilingual alerts, converting the generated messages into Kannada or English. To automate weather checks and alert generation, the backend employs a job scheduler, which allows these tasks to run periodically without manual initiation. The scheduler is configured to execute time-based tasks using Python libraries, ensuring consistency and reducing latency in critical scenarios. Error handling mechanisms are included to deal with failures in API requests, connectivity issues, or unsuccessful message deliveries. In addition, access to the backend is protected through login authentication and route protection, ensuring that only authorized personnel can interact with sensitive data or configurations.

3.4 Data Management Layer

The data management layer is responsible for storing and retrieving all persistent information used by the system. For simplicity and ease of use, Microsoft Excel files are used as the primary data storage medium. This approach eliminates the need for database administration expertise while still supporting structured data handling. The system uses Python's Pandas library to interact with these Excel files programmatically. There are two major spreadsheets used within the system: one for managing farmer records and the other for logging the alert history. The farmer record sheet contains fields such as name, contact number, district or village, and the preferred language for alerts. The alert log captures all outgoing communication attempts, along with relevant metadata including time, alert type, delivery method, and recipient. This data structure ensures traceability and transparency, allowing system administrators to audit past actions and assess the effectiveness of alert responses.

Although Excel serves as a functional and accessible data store for a limited number of users, the system is designed to be scalable. In future iterations, the data layer can be easily migrated to a relational database management system such as MySQL or PostgreSQL. This would improve query performance, enhance security, and support concurrent access in large-scale deployments.

3.5 Integration Architecture

The system relies on two main external services to perform its core functions. The first is the OpenWeatherMap API, which supplies real-time and forecasted meteorological data. It is queried using HTTP requests with specific geographic parameters such as latitude and longitude corresponding to the farmer's registered location. The response from this API is parsed to extract relevant fields like wind speed, rainfall, and temperature. The second external service is the Twilio API, which is responsible for the actual delivery of alerts. It supports SMS messaging as well as automated phone calls, both of which are critical for reaching farmers with varying literacy levels and device capabilities.

To maintain security, both APIs are accessed using unique authentication tokens stored in environment variables. This approach protects sensitive credentials from being exposed in the application's source code. The integration is implemented using RESTful communication, ensuring compatibility and flexibility across different platforms. These external services are stable and widely adopted, which makes the overall system highly reliable and easily maintainable.

IV. SYSTEM FEATURES

4.1 Real-Time Weather Monitoring and Alerts

The primary function of the Farmer Weather Alert System is to monitor real-time weather conditions and generate alerts based on pre-configured thresholds. The system utilizes the OpenWeatherMap API, which provides continuous weather updates including temperature, humidity, wind speed, precipitation levels, and atmospheric pressure. These updates are pulled periodically by the backend scheduler and evaluated using conditional logic defined by the system administrator. When any weather parameter surpasses the predefined safety threshold, the system immediately triggers an alert generation protocol. This automation ensures that alerts are disseminated without manual intervention, allowing for faster responses during critical weather events. The real-time nature of the data acquisition process significantly enhances the system's reliability and accuracy in forecasting potentially damaging conditions.

4.2 Multilingual Communication Support

To ensure accessibility and inclusivity for a linguistically diverse farmer population, the system supports multilingual communication. Specifically, alerts can be generated and transmitted in both English and Kannada, which are widely used across Karnataka and other South Indian regions. The language preference is recorded during the registration of each farmer and stored in the backend dataset. During alert creation, the system dynamically constructs the message content based on the weather condition and automatically translates it into the desired language. This capability is vital in overcoming barriers related to illiteracy and language comprehension. By delivering information in the local dialect, the system empowers farmers to take timely and informed decisions, thereby reducing the risk of misinterpretation and delayed response.

4.3 SMS and Voice Call Integration via Twilio

The system integrates the Twilio API to dispatch notifications in the form of SMS and voice calls. Twilio is a reliable cloud communications platform that allows for programmable messaging and voice services. Upon identification of a hazardous weather condition, the backend constructs a message and invokes the Twilio API to send it to the farmer's registered phone number. If the alert is classified as severe—such as cyclone warnings or high wind velocity—the system is programmed to also initiate an automated voice call, ensuring the alert is noticed even in situations where the farmer may not be able to read a text message. This dual-mode alert mechanism enhances the probability of successful delivery and acknowledgment of the weather warning. Moreover, the integration ensures that the communication layer is resilient, scalable, and well-suited for deployment in rural areas with varying network availability.

4.4 Farmer Profile and Alert Management Interface

A critical feature of the system is the administrator web interface, which facilitates the management of farmer records and system configurations. The interface is developed using Flask and rendered through responsive web templates that enable interaction across desktop and mobile browsers. Through this interface, the administrator can register new farmers by entering details such as full name, contact number, village or region, and language preference. Records can be updated or deleted as needed, ensuring that the database remains current and accurate. The system also provides a live overview of active farmers, total alerts sent, and a weather conditions panel. In addition to user management, the admin interface allows for manual override of automatic alert functions. This gives administrators the flexibility to send custom notifications in exceptional scenarios, such as pest alerts or market price changes, thereby extending the utility of the platform beyond weather monitoring.

4.5 Customizable Weather Thresholds

One of the system's more advanced features is the ability for administrators to define and update the weather thresholds that trigger alerts. These thresholds represent the sensitivity settings for each weather parameter. For instance, an administrator might set the wind speed alert threshold to 40 km/h or define heavy rainfall as precipitation exceeding 30 mm per hour. These values can be modified through the admin dashboard, which immediately updates the evaluation logic in the backend. Customization of thresholds is essential for ensuring local relevance, as different regions may experience weather phenomena with varying severity. This flexibility allows the system to be adapted for different agricultural zones, crops, and climatic conditions. Additionally, this feature supports data-driven policymaking, where thresholds can be adjusted based on historical trends and agronomic advisories.

4.6 Alert Logging and Historical Recordkeeping

All alert activities conducted by the system are automatically recorded and stored in a structured Excel sheet, ensuring transparency and traceability. Each entry in the log contains detailed metadata, including the exact time and date of alert issuance, the type of weather event, the name and contact of the recipient farmer, the message content, and the delivery method (SMS or voice call). This alert history is viewable through the web interface, providing administrators with a chronological record of all communications. Such a record is essential for auditing purposes, post-event analysis, and validating the system's performance over time. In the case of disputes or inquiries regarding alerts, the log serves as an official document that can verify when and how messages were sent. Furthermore, the alert history enables longitudinal studies on the effectiveness of interventions and can inform future improvements to the platform.

4.7 Security, Accessibility, and User Experience

The system incorporates several measures to ensure that it is both secure and accessible. Access to the administrator portal is restricted through login authentication mechanisms, which prevent unauthorized individuals from altering farmer records or system thresholds. Form submissions are validated on both the client and server sides to prevent injection attacks or erroneous entries. Additionally, user interface components are designed with simplicity in mind, ensuring that users with minimal digital literacy can navigate the system effectively. The layout is optimized for small screen devices, allowing administrators in field conditions to use smartphones for updates and monitoring. Accessibility features, including clear typography and language labels, make the interface friendly for users across age groups and literacy levels. This attention to usability strengthens the system's adoption potential and ensures reliable performance under field conditions.

V. SYSTEM ILLUSTRATIONS

The Farmer Weather Alert System offers a streamlined and accessible user interface that allows administrators to monitor weather conditions, manage farmer data, and configure alerts efficiently. This chapter presents the core visual components of the system interface, each illustrated through screenshots that demonstrate its functionality in real-world usage.



Figure 1 Home Screen

Figure 1 presents the Home Screen, which serves as the main entry point for administrators. From this screen, users can navigate to other modules such as farmer registration, weather forecasts, threshold settings, and alert history. The design emphasizes clarity and accessibility, ensuring that system functions are quickly and easily identifiable.



Figure 2 Weather Forecast Dashboard

Figure 2 shows the **Weather Forecast Dashboard**, where current and forecasted weather data for registered farmer locations is displayed. This screen pulls data from the OpenWeatherMap API and includes essential parameters such as temperature, rainfall probability, wind speed, and humidity. The forecast is presented in both hourly and weekly formats, allowing administrators to anticipate trends and make proactive decisions.

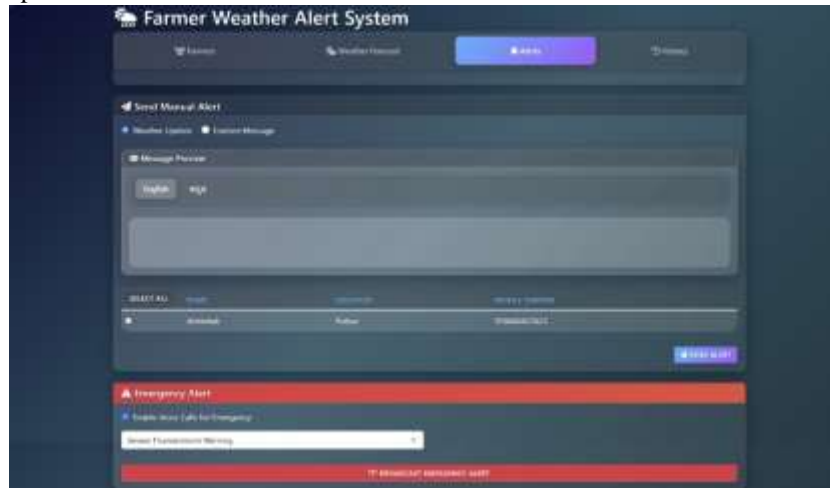


Figure 3 Alert Dashboard

Figure 3 depicts the **Alert Dashboard**, which is the operational center for weather alert management. This interface allows administrators to view ongoing conditions against set thresholds and provides options for manual or automated alert dispatch. The alert dashboard includes controls for emergency alerts, scheduled notifications, and real-time updates based on weather changes. It also features visual indicators to help identify critical areas requiring immediate attention.



Figure 4 History Dashboard

Figure 4 displays the **Alert History Panel**, which maintains a log of all alerts that have been sent to farmers. Each record includes details such as the date and time of the alert, weather event description, delivery method (SMS or call), and the recipient's contact. This historical record not only provides transparency but also allows system administrators to evaluate the timeliness and effectiveness of communications over time.

These figures collectively demonstrate the design focus on usability, transparency, and responsiveness. They reinforce how the system's interface supports the operational goal of delivering timely, reliable, and easily comprehensible weather alerts to farmers across diverse locations.

VI. CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The Farmer Weather Alert System represents a significant technological intervention aimed at improving the resilience of rural agricultural communities in the face of unpredictable and extreme weather conditions. Through its integration of real-time weather APIs, multilingual communication, and automated alert mechanisms, the system offers a practical and scalable solution to one of agriculture's most pressing challenges — timely and accurate access to localized weather information.

The system's core strength lies in its automation and accessibility. By combining a Python Flask backend with the OpenWeatherMap API and Twilio's messaging services, it eliminates the delay and ambiguity often associated with traditional weather communication methods. Farmers are notified of severe weather events directly through SMS or voice calls in a language they understand, allowing them to take preventive measures to safeguard crops, livestock, and equipment. The inclusion of a secure and responsive web interface for administrators enhances usability and supports real-time monitoring, user management, and historical data logging.

Furthermore, the system's use of Excel for data storage ensures ease of access and flexibility for small-scale deployments, especially in regions where technical infrastructure may be limited. The modular architecture also ensures that the platform can be maintained, upgraded, and scaled without the need for major restructuring.

In summary, the system demonstrates how modern web technologies, cloud APIs, and intuitive user interfaces can be effectively combined to serve real-world agricultural needs. It not only closes the communication gap between weather forecasting services and end-users but also promotes data-driven decision-making in the farming sector.

6.2 Future Scope

While the current implementation is effective for small to mid-scale use cases, there are several opportunities for further development and enhancement. One of the most promising directions is the integration of Internet of Things (IoT) devices such as local weather sensors and soil moisture monitors. These devices can provide hyper-localized data, improving the precision of alerts and enabling features like automatic irrigation activation during dry conditions or shutdowns during storms.

In addition, the Excel-based data management system, although convenient, may become inefficient as the number of registered users increases. Future versions of the system can adopt robust relational database solutions such as MySQL or PostgreSQL to handle larger datasets, allow advanced querying, and improve data security. This will also support concurrent access and multi-user environments.

Developing a dedicated mobile application is another key area of expansion. Such an app could allow farmers to register themselves, view upcoming forecasts, receive alerts with rich media content, and provide feedback on weather events. Mobile app support would improve accessibility and user engagement, especially for younger or tech-savvy farmers.

Language diversity can be further addressed by including additional regional languages beyond Kannada and English. This would make the system more inclusive and adaptable across different states and linguistic zones in India.

Lastly, potential collaborations with governmental meteorological departments, local agricultural extension services, and disaster management agencies could extend the system's impact. These collaborations could enable official verification of alerts, access to satellite-based forecasting data, and integration into broader rural development programs.

In conclusion, the Farmer Weather Alert System serves as a foundational framework upon which more complex, regionally adaptive, and intelligent agricultural support tools can be built. Its continued development could play a crucial role in transforming rural agriculture through timely, data-driven interventions

VII. REFERENCES

- [1] Lavanya, G., et al. "An automated low-cost IoT-based fertilizer intimation system for smart agriculture." *Sustainable Computing: Informatics and Systems* 28 (2020): 100300.
- [2] SriLakshmi, N., et al. "Integrated Agricultural Monitoring System with SMS Alerts and Web Application." *International Journal for Research Trends and Innovation (IJRTI)*, vol. 9, no. 3, 2024, pp. 59–63.
- [3] Singh, G., and Sharma, S. "Enhancing precision agriculture through cloud based transformative crop recommendation model." *Scientific Reports*, vol. 15, 2025, article 9138.
- [4] Rao, B. S., & Reddy, B. S. (2021). "IoT-Based Smart Agriculture Monitoring and Irrigation System." *International Journal of Engineering Research & Technology (IJERT)*, Vol. 10, Issue 4.
- [5] Sharma, A., & Verma, A. (2020). "Design of Weather Monitoring System Using OpenWeatherMap API and Python." *Journal of Emerging Technologies and Innovative Research (JETIR)*, Vol. 7, Issue 6.
- [6] Patel, K., & Patel, D. (2019). "Real-Time Smart Agriculture Monitoring System using IoT." *International Journal of Computer Science and Mobile Computing*, Vol. 8, Issue 5.
- [7] Agarwal, R., & Saxena, P. (2021). "Application of Twilio for Real-Time Notification in Disaster Management Systems." *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, Vol. 7, Issue 3.
- [8] Singh, N., & Gupta, M. (2022). "Smart Weather Forecasting Using IoT and Cloud Technology." *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, Vol. 11, Issue 1.
- [9] Kumar, P., & Joshi, M. (2020). "Decision Support Systems in Agriculture: A Review." *Agricultural Informatics Journal*, Vol. 5, Issue 2.
- [10] Raj, P., & Thomas, R. (2022). "A Cloud-Based Bilingual SMS Alert System for Farmers Using Weather APIs." *International Journal of Recent Technology and Engineering (IJRTE)*, Vol. 11, Issue 2.
- [11] OpenWeatherMap. (2023). "API Documentation." <https://openweathermap.org/api>
- [12] Twilio. (2023). "Twilio Programmable Messaging API." <https://www.twilio.com/docs/sms>
- [13] McKinney, W. (2018). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media
- [14] Ramesh, G., & Kumar, S. (2021). "Smart Farming through IoT: Current Trends, Applications, and Challenges." *International Journal of Scientific & Engineering Research (IJSER)*, Vol. 12, Issue 9.
- [15] Al-Garadi, M. A., et al. (2020). "A Survey of Communication Technologies for IoT-Based Smart Farming." *IEEE Communications Surveys & Tutorials*, Vol. 22, No. 4, pp. 3152–3184.