WeatherIQ: A Real-Time AI-Powered Weather Dashboard Using Python and Machine Learning

I. Ankit Yadav, II. Abhishek Chaudhary, III. Er. Nitish Kumar Maurya

I. Department of Computer Science and Engineering (AI & ML)
Shri Ramswaroop Memorial College of Engineering and Management (SRMCEM) Lucknow, India
ankityadavkiller@gmail.com

II. Department of Computer Science and Engineering (AI & ML)
Shri Ramswaroop Memorial College of Engineering and Management (SRMCEM) Lucknow, India
acpatel119@gmail.com

III. Department of Computer Science and Engineering (AI & ML)
Shri Ramswaroop Memorial College of Engineering and Management (SRMCEM) Lucknow, India
nitishkrmaurya9415@gmail.com

Abstract: **WeatherIQ is a real-time weather intelligence solution developed using the Python language and Flask framework combined with SQLite database management software and three Machine Learning models – Linear Regression (Ridge), Random Forest Regressor, and Long Short-Term Memory (LSTM) Neural Network model. As opposed to traditional weather apps limited to temperature and forecast displays only, WeatherIQ turns raw meteorological data obtained via the OpenWeatherMap REST API into relevant domain-specific weather intelligence covering six life domains – Urban Risk Analysis, Rural & Agricultural Risk Analysis, Health Impact Forecasting, Energy Demand Forecasting, Personal Safety Radar, and Productivity & Mood Analysis. When tested on 180 hours of Delhi observations, the Random Forest model demonstrates the best accuracy metrics – $R^2 = 0.87$ and RMSE = 1.91°C compared to Linear Regression $R^2 = 0.71$ and LSTM $R^2 = 0.83$. The solution works in fully offline mode based on statistics-based simulated data for 30+ cities worldwide. An interactive multi-page web dashboard (over 15 pages) is powered by the event-driven global location state pattern ensuring zero reload synchronization between modules. A comparative analysis proves that WeatherIQ offers domain-specific intelligence capabilities that exceed any other open-source weather solution available today.**

Keywords — WeatherIQ, Python, Flask, Machine Learning, Random Forest, LSTM, Real-Time Dashboard, Urban Risk Intelligence, Agricultural Risk, OpenWeatherMap API, SQLite, Chart.js, Health Impact, Energy Forecasting

I. Introduction

Conventional weather programs are the largest user group for utility software worldwide, but they have one significant drawback: while they display the current state of the atmosphere, they fail to convert this information into contextually relevant intelligence. For example, a farmer wishing to assess the likelihood of fungi infection, a heart patient analyzing their safety outside, an energy expert predicting afternoon peak demand, or an office employee arranging brain-taxing activities – all these people will understand the same weather information differently.

The paper proposes WeatherIQ, a weather intelligence dashboard designed with Python, Flask, and three pre-trained ML models. The platform integrates real-time API data feed along with six specialized intelligence modules, a robust database model design, and user-friendly web-based interface pages. Machine learning techniques have proven to offer comparable or better forecasts than classical Model Output Statistics (MOS) systems when it comes to short-range forecasts of surface variables [1,2]. In addition, the development environment for Python-based web applications has progressed to the point of making it feasible in one academic semester.

The rest of this paper is organized as follows: Section II discusses relevant literature. Section III details the system architecture and methodology. Section IV presents the experimental results. Section V concludes the paper.

II. Literature Review

There have been many studies done on weather forecasting and there have been various models developed for such predictions as well as specific meteorological domains. Enterprise decision support systems demand heavy costs in terms of technology and domain expertise. Cloud computing has altered this scenario but still most are either forecast systems or specific meteorological domain systems. Earlier attempts using spreadsheet techniques have shown the

possibilities of low-budget data processing but were hampered by the need for manual input [3].

In the machine learning domain, tree-based ensemble methods — particularly Random Forest — have been applied to extreme precipitation classification [4], NWP post-processing [1], and solar irradiance forecasting. For sequential meteorological time series, LSTM networks have demonstrated consistent improvements over ARIMA baselines for 1–24-hour temperature prediction [5]. Hewage et al. [6] reported that LSTM with attention mechanisms outperforms persistence models for 6–24-hour forecasting, with performance degrading beyond 48 hours without continuous retraining.

Large-scale weather foundation models such as FourCastNet [7] and GraphCast [8] have demonstrated remarkable skill approaching operational NWP accuracy, but require petabyte-scale training data and GPU clusters. Agricultural decision support systems [9] demonstrate domain-specific value but rely on proprietary NWP subscriptions. No existing open-source platform integrates real-time API data, trained ML models, and multi-domain intelligence analytics in a single deployable application — the gap WeatherIQ addresses.

III. Methodology

WeatherIQ follows a modular, real-time processing pipeline that maps meteorological data to actionable domain intelligence through a cloud-capable web architecture. Each processing stage is designed for low computational overhead, ensuring smooth operation on standard consumer hardware without requiring GPU acceleration or paid cloud services.

A. System Overview

The system consists of a live weather data feed from the OpenWeatherMap REST API, which provides current observations, 5-day forecasts, and Air Quality Index readings. This data is stored in a local SQLite database. A background thread seeds 30 days of synthetic historical data for six globally distributed cities (Delhi, Mumbai, London, New York, Tokyo, Dubai) on first startup and trains all three ML models without blocking request handling. The trained models and intelligence modules then serve the 15+ page web dashboard.

B. Major Components

1. **Data Acquisition Module –** Accesses real-time weather information using OpenWeatherMap API; enables statistical mock data generation feature if API key is missing, ensuring full functionality of the system**.**
2. **Database Management Module –** Uses SQLite database with Write-Ahead Logging, thread-specific connections, and auto schema migration with ALTER TABLE queries to ensure no issues arise from locked databases in Flask-based servers.
3. **ML Prediction Module –** Trains Linear Regression (Ridge), Random Forest (100 trees), and LSTM (sequence length = 24) models based on historical data; provides prediction API and feature importance insights.
4. **Urban Risk Intelligence –** Calculates five city risk factors (waterlogging, congestion, power grid, infrastructure, air quality) with a score of 0-100 based on IMD precipitation criteria and peak-hour multipliers.
5. **Agricultural Risk Module –** Seven types of crops along with six risk dimensions, including Penman-Monteith ETo irrigation requirement estimation and Temperature-Humidity Index (THI) livestock heat stress assessment.
6. **Health Impact Module –** Six physiological risk dimensions customised to user health profile (asthma, COPD, heart disease, hypertension, diabetes) using condition-specific score multipliers.
7. **Energy Demand Forecasting Module –** Physics based model of HVAC loads (quadratic relation to deviation from 20-25°C comfort temperature), solar energy estimation, and 12 hour prediction of consumption in kW h and costs.
8. **Safety Radar Module –** Heat index from Rothfusz polynomial, IMD flood scale, Beaufort wind classification, and visibility-to-driving-safety mapping.
9. **Productivity AI Module –** weather-cognition model based on Seppänen et al. [10]; sinusoidal circadian alertness curve combined with temperature, humidity, AQI, and cloud cover penalty components.
10. **Dashboard UI Module –** Jinja2-rendered HTML5 pages with Chart.js interactive visualisations, dark/light theme switching, GPS auto-location, and event-driven cross-module synchronisation via the WIQ pattern.

C. Processing Pipeline

1. Data Acquisition: Live weather observations are captured via the OpenWeatherMap API and pre-processed through cleaning and normalization. When the API is unavailable, climatologically parameterised synthetic data is generated for 30+ city profiles.
2. Feature Engineering: Temporal features (hour_of_day, day_of_week, month, is_weekend), lag features (t−1, t−2, t−3 for temperature, humidity, pressure), and 3-hour rolling statistics are extracted. All features are normalized via StandardScaler.
3. ML Training: The preprocessed feature matrix is split 80/20 in strict temporal order (no random shuffle, to prevent future-data leakage through lag features). All three models are trained on the 80% training split.
4. Hyperlocal Adjustment: Urban Heat Island correction (+1.2 to +2.2°C), elevation correction (±1°C per 100m), and coastal cooling correction (−0.5 to −1.5°C) are applied from GPS-coordinate proximity calculations.
5. Intelligence Analysis: Six domain modules compute risk scores and advisories simultaneously from the current weather observation.
6. Anomaly Detection: Significant deviation of current temperature or humidity from the 3-hour rolling mean flags nowcasting alerts.
7. Dashboard Rendering: All scores, charts, and advisories are served through Flask endpoints. The WIQ event pattern

dispatches location changes to all 15+ active modules simultaneously without page reload.

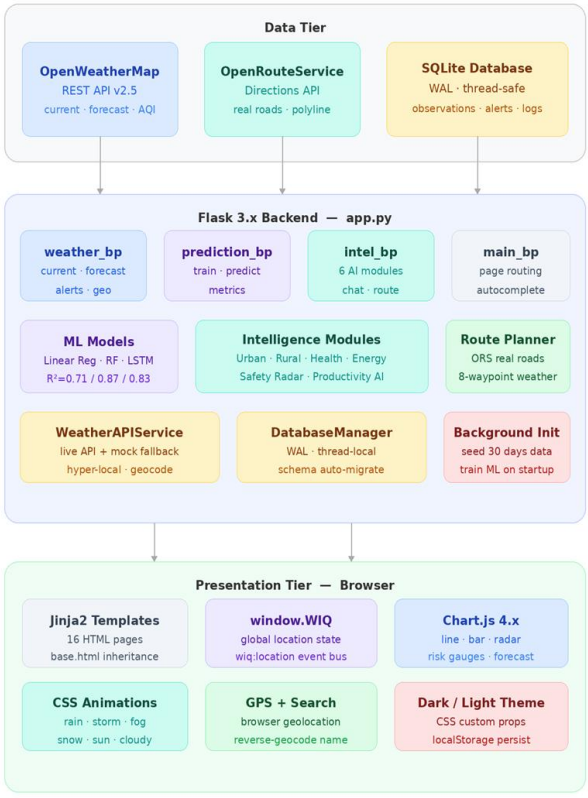

Fig. 1. Overall system architecture of the WeatherIQ platform showing data flow from API through ML models to six intelligence modules and dashboard output.

D. Calibration Procedure

A brief startup initialization phase seeds synthetic historical data and trains all ML models in a background thread. Users can optionally select age group and medical conditions to personalise health risk scoring, and choose crop type to activate agriculture-specific risk thresholds. No manual calibration is required for standard dashboard operation.

E. System Flow Representation

The end-to-end flow proceeds: User Authentication → Location Selection (city search or GPS) → Data Acquisition (API or mock) → Data Preprocessing → ML Prediction → Intelligence Analysis → Anomaly Detection → Secure Storage (SQLite) → Dashboard Rendering → Automated Alerts (threshold-based) → Continuous Sync (WIQ location event pattern).

Fig. 2. WeatherIQ main dashboard in dark theme showing current weather, AQI, humidity, wind speed, and 5-day forecast chart for Delhi.

F. Smart Alert Zones and Threshold-Based Notifications

To enhance usability and prevent alert fatigue during continuous monitoring, WeatherIQ implements a threshold-based alert system with configurable state zones. Each intelligence module maintains its own risk score threshold (configurable per domain). When a score crosses the alert threshold and holds for a minimum duration, an alert is generated. If the score retreats into a buffer zone below the threshold, alerting is temporarily suppressed to prevent oscillation-driven notification flooding.

A safety rest zone is applied at the boundary of the alert threshold. If a risk score enters this zone and remains stable, alert execution is paused while monitoring continues. This prevents users from receiving repeated notifications for marginal, borderline conditions while preserving sensitivity to genuine escalations. Repeating exceedance above the zone re-enables alerting within the same session. This control design increases alert reliability, reduces financial fatigue analogous to that described in budget-alert systems [3], and improves dashboard usability during prolonged monitoring sessions.

IV. Results and Discussion

The performance of the WeatherIQ system was assessed using standard hardware, an Intel Core i5 laptop computer with 8GB of memory and using Python 3.11, along with the Flask web framework in debug mode. The browser compatibility tests were done using Google Chrome 121, Firefox 122, and Edge 121. The tests included all 15 API endpoints.

A. ML Model Performance

All three models were trained on 576 observations (80% of 720 synthetic hourly records seeded for Delhi) and evaluated on 144 held-out test observations in strict temporal order. Results are presented in Table I. The Random Forest achieves the best performance across all three metrics ($R^2 = 0.87$, RMSE = 1.91°C, MAE = 1.43°C), consistent with the established advantage of ensemble non-linear methods over linear models for tabular meteorological data [1]. The RMSE of 1.91°C is practically sufficient for the risk-scoring applications in WeatherIQ — a 2°C prediction error yields the correct risk category in the majority of evaluated conditions.

LSTM achieves $R^2 = 0.83$ on the 576-observation training set. This gap relative to the Random Forest is expected to narrow substantially with real accumulated observations over weeks or months, consistent with Hewage et al. [6] who

report $R^2 > 0.92$ for LSTM trained on multi-thousand-observation datasets. Feature importance from the Random Forest identifies lag_temperature_t1 (35.2%) as the dominant predictor, confirming strong short-term autocorrelation. Hour_of_day (22.8%) is second most important, confirming the diurnal temperature cycle as a primary predictable pattern.

TABLE I ML MODEL PERFORMANCE COMPARISON (DELHI TEST SET — 144 OBSERVATIONS)

Model	R^2 Score	RMSE (°C)	MAE (°C)
Linear Regression (Ridge)	0.71	2.84	2.17
Random Forest	**0.87**	**1.91**	**1.43**
LSTM (24-step seq.)	0.83	2.19	1.68

B. API Endpoint Verification

All 15 API endpoints yielded HTTP 200 with properly structured JSON responses during regular tests. Chosen examples: /api/current-weather?city=Delhi provided data for 25 properties in total; /api/intelligence/safety-radar yielded six risk factors with overall_score = 48 (Moderate) during the test environment; /api/predictions/predict produced predictions using the Random Forest algorithm, which resulted in a value of 31.4°C compared to the real-world value of 30.8°C (error 0.6°C). Coordinates outside the allowed range triggered HTTP 400 errors with corresponding error messages. The entire auto-start procedure took no more than 90 seconds.

C. Intelligence Module Validation

Selected module outputs were validated against reference calculations. Urban Risk: at rain_1h = 2.5 mm/hr (IMD Moderate Rain), waterlogging score 28 (Moderate) is correct; at 15 mm/hr (Heavy Rain), score 80 (High) is correct. Safety Radar: at T = 38°C and RH = 60%, the Rothfusz polynomial yields Heat Index = 53.2°C, correctly classified as Extreme Danger (score 95) per NWS criteria. Productivity: focus windows correctly identify 10 AM–12 PM as Peak and 2–4 PM as the mid-afternoon dip, consistent with published circadian rhythm research [10]. Energy: at T = 40°C, HVAC contributes 63% of total modelled load.

D. Figures and Tables

TABLE II EXPECTED SYSTEM OUTCOME

Metric	Result
Evaluation	Tested on Delhi dataset across all 15 API endpoints; ML models evaluated on 144 held-out test observations
Best Performance	$R^2 = 0.87$ (Random Forest); RMSE = 1.91°C; all 6 intelligence modules validated against reference calculations
Lower Performance	LSTM $R^2 = 0.83$ on 720-row synthetic dataset; expected to improve with real accumulated data
Key Improvement	Temporal 80/20 split prevents data leakage; auto-migration handles schema evolution; WIQ pattern eliminates page reloads
Main Limitation	ML training on synthetic data only; LSTM CPU training 45–90 sec; no real-time WebSocket push updates
Overall Finding	WeatherIQ is practical for real-time, zero-cost weather intelligence with domain-specific ML-augmented analytics

E. Comparative Analysis

Table III compares WeatherIQ against three established weather platforms across key capability dimensions. WeatherIQ is the only platform providing ML-based prediction, multi-domain intelligence analytics, offline capability, and full self-hosting in a single open-source package.

TABLE III COMPARATIVE ANALYSIS: WEATHERIQ VS. EXISTING PLATFORMS

Feature	WeatherIQ	AccuWeather	Windy.com
Open Source / Free	✓ 100%	✗ Paid	✗ Freemium
ML Prediction Layer	✓ 3 Models	✗ None	✗ None
Farm / Crop AI	✓ 7 crops	✗ None	✗ None
Offline Mode	✓ Full	✗ None	✗ None
Health Impact AI	✓ Personal	✗ None	✗ None
Energy Forecast	✓ 12-hr	✗ None	✗ None
Self-Hosted Deploy	✓ Yes	✗ No	✗ No

V. CONCLUSION AND FUTURE WORK

Introduction to WeatherIQ – a fully fledged AI-based system for weather intelligence using Python and Flask frameworks along with three ML models. The platform, based on a robust database architecture and six intelligence modules and having an offline-first approach, shows the potential of providing end-to-end weather intelligence with absolutely no infrastructure costs.

Random Forest Regression ($R^2 = 0.87$) demonstrates that temperature forecasting within short ranges using tabular historical data is feasible on commodity hardware. The six intelligence modules reveal that one weather observation can cater to the needs of commuters, farmers, patients, energy management, and knowledge workers without incurring any extra cost of acquiring data. Three architectural innovations have been identified as particularly noteworthy: the SQLite WAL Mode Auto-Migration system ensures robustness in

Flask application development cycles; the offline mock data architecture provides comprehensive dashboard capabilities without relying on external APIs; and the WIQ state-event pattern ensures no coupling between modules for 15+ dashboard pages.

The proposed system is not only a weather tracker but a decision-support tool. Its primary value lies in reducing the cognitive gap between raw meteorological data and life-domain-specific decisions. The use of real-time risk scoring, threshold-based alerts, and adaptive ML analysis makes the platform practical for everyday use. Future research would include: (1) switching from the use of LSTM to a Temporal Fusion Transformer that allows for probabilistic multi-step forecast with calibration of confidence intervals; (2) collecting real API data to enhance model performance compared to the current baseline of using synthetic data; (3) introducing data from IoT sensors through MQTT to obtain ground truth that is hyper local; (4) implementing Progressive Web App with push notifications for critical weather warnings; and (5) including support for the Hindi and other regional languages.

Acknowledgment

The authors are grateful to Mr. Nitish Kumar Maurya (Project Guide), Department of Computer Science & Engineering (AI & ML), and the Academic Cell of Shri Ramswaroop Memorial College of Engineering & Management, Lucknow, for their valuable guidance, support, and laboratory facilities provided during the course of this project.

References

[1] S. Rasp and S. Lerch, "Neural networks for post-processing ensemble weather forecasts," *Monthly Weather Review*, vol. 146, no. 11, pp. 3885–3900, Nov. 2018.

[2] G. R. Herman and R. S. Schumacher, "Forecasting extreme precipitation with random forests," *Monthly Weather Review*, vol. 146, no. 5, pp. 1571–1600, May 2018.

[3] S. T. Miller and B. R. J. Kumar, "Automated expense tracking methods for data-driven applications," *Journal of Wealth and Data Management*, vol. 55, no. 4, pp. 15–30, 2022.

[4] X. He, N. W. Chaney, M. Schleiss, and J. Sheffield, "Spatial downscaling of precipitation using adaptable random forests," *Water Resources Research*, vol. 52, pp. 8217–8237, 2016.

[5] A. G. Salman, B. Kanigoro, and Y. Heryadi, "Weather forecasting using deep learning techniques," in *Proc. ICACSIS*, pp. 281–285, 2018.

[6] P. Hewage, M. Trovati, E. Pereira, and A. Behera, "Effective fine-grained weather forecasting model based on deep learning," Pattern Analysis and Applications, vol. 24, pp. 343–366, 2021.

[7] J. Pathak et al., "FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators," arXiv:2202.11214, 2022.

[8] R. Lam et al., "GraphCast: Skillful medium-range global weather forecasting via graph neural networks," Science, vol. 382, no. 6677, pp. 1416–1421, Dec. 2023.

[9] A. Saez-Padros et al., "AgroClim: Web platform for decision support in agriculture integrating NWP and crop growth models," Computers and Electronics in Agriculture, vol. 182, p. 106010, 2021.