International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1817-1825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0238

IoT-Based Real-Time Weather and Disaster Management System Using Machine Learning

Dr. K. Chaitanya[1], Sai Keerthi Kokkeri[2], Lalitha Kamakshi Makaraju [3], 2Vandana Kurapati [4], 2Bhanu Prasad Gampa [5]

[1] Associate Professor, Dept. of Computer Science and Engineering, SRK Institute of Technology, Vijayawada, AP, India

[2,3,4,5]Students, Dept. of Computer Science and Engineering, SRK Institute of Technology, Vijayawada, AP, India

Emails:*Kilaru.chaitanya84@gmail.com[1],kokkerisaikeerthi@gmail.com[2],rlalitha557@gmail.com[3], vandanakurapati35@gmail.com[4], gampabhanu933@gmail.com[5]*

Abstract

Abstract—The article describes a basic IoT framework through the integration of ESP 323 samples, an array for several sensors, as well as an ensemble of machine learning algorithms for both environmental monitoring and disaster prediction. The IoT monitoring system incorporates MQ series gas sensors (MQ-135, MQ-2, MQ-7, MQ-9), ultrasonic distance sensors, and the OpenWeather API to monitor the environment for air quality (e.g. smoke detection), carbon monoxide levels and combustible gas levels, water level and seismic activity. One of the most significant advantages of this proposal is the machine learning pipeline, which consists of a total of ten separate classifiers and includes: Random Forest, Gradient Boosting, Support Vector Machines, and Neural Networks. The use of the ensemble models gives 94% or higher accuracy in predicting flooding events. Other features that are part of the proposed solution include: automated email alerting, real time visualization of sensor data via Web Dashboard, access to sensor data via ThingSpeak cloud service, and ability to analyze both historical and current data. The experimental testing indicates the ability of the proposed system to detect hazards using configurable alert thresholds, and 15 minutes of alert cool down period to avoid generating too many alerts (alert fatigue). Overall, the proposed architecture can provide a cost-effective, scalable solution to encompass both environmental monitoring and disaster early warning systems.

Keywords: *Internet of Things, Disaster Management, Machine Learning, Environmental Monitoring, ESP32, Ensemble Learning, Real-time Systems*

1. Introduction

Human life, infrastructure, and economic stability can be threatened by natural disasters and other environmental hazards. The increasing number and intensity of weather-related disasters caused by climate change creates a greater need for effective monitoring and early warning systems for these types of events. For example, traditional disaster management methods may lead to delayed reactions to events, limited geographic reach, and insufficient predictive capability. The convergence of three important technologies Internet of Things (IoT), wireless sensor networks, and machine learning creates opportunities for creating adaptive and intelligent environmental monitoring systems. IoT-enabled devices allow for the collection of data in real time from large networks of distributed sensors, while machine learning is able to use historical data to identify patterns and make predictions about future disasters.

1.1.Motivation

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1817-1825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0238

Limitations that environmental monitoring today continues to struggle with are: Expensive setups, Lack of ability to process data in real-time, Lack of different types of sensors, and Lack of prediction analysis. As a result, existing solutions do not combine multiple hazard types into one centralized platform which creates siloed monitoring strategies. No existing solution can adequately capture how these different hazards interact within the same system [1].

1.2.Contributions

This research outlines the creation of an all-encompassing IoT Disaster Management system that offers these contributions:

- A multi-sensor integration architecture that combines multiple types of sensors (Gas Quality sensors, Ultrasonic Distance Sensors and Meteorological Sensors).
- An ensemble machine learning framework that evaluates ten different classification algorithms sused for predicting Flood Risk.
- A Real-Time Web Dashboard that includes an automatic email notification system.
- An integration with the cloud using the ThingSpeak platform allowing for Remote Monitoring of the IoT Disaster Management System.
- Analytical capabilities to assess the historical data and visualize it.
- A scalable architecture that supports the potential for adding sensors in the future.

2. Related Work

2.1.IoT-Based Environmental Monitoring

Innovations in IoT technologies over the past few years have resulted in the rapid growth of Environmental Monitoring Systems (EMS) by allowing researchers to create more sophisticated sensor network designs for monitoring Air Quality (AQ), Water Pollution (WP), and Weather Parameters (WP). In addition, research has highlighted the effectiveness of deploying Wireless Sensor Networks (WSNs) for delivering Real-time (RT) Environmental Data (RED) at lower costs than traditional environmental monitoring stations.

2.2.Disaster Prediction Using Machine Learning

In disaster prediction applications, Machine Learning (ML) algorithms have provided encouraging results with respect to the prediction of disasters before their occurrence. As a result, a wide variety of classification methodologies such as Decision Trees (DT), Random Forests (RF), and Support Vector Machines (SVM) have been utilized for Flood Forecasting (FF), Earthquake Early Warning (EEW), and Wildfire Prediction (WP). Unlike DT, RF, and SVM, Deep Learning (DL) methodologies such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) have produced superior predictive accuracy concerning time-series data utilized for forecasting events associated with environmental phenomena [2].

2.3.Smart City and Climate Monitoring Solutions

Smart City initiatives have utilized IoT-based Climate Monitoring Systems for increasing their Urban Resiliency (UR). Most Climate Monitoring Solutions available today combine Inputs from more than one source, including National Weather Service weather stations, satellite imagery, and Reports from Citizens and have focused on monitoring one specific Hazard Type. However, the majority of Climate Monitoring Solutions today do not have comprehensive multi-hazard monitoring capabilities [3].

2.4.Research Gap

Considerable advances have been made in this area, but many research gaps still exist. These include the following: an insufficient amount of integration among the different sensor types used to create Climate Monitoring Systems; too little capability to process real-time data in order to detect imminent Hazard activity; a failure to provide Non-Technical Stakeholders with User Friendly Interfaces for utilizing Climate Monitoring Systems; and inadequate testing and validation of Climate Monitoring Systems in real-world circumstances. This research will address the above identified deficiencies [4].

3. System Architecture And Methodology

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1817-1825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0238

3.1. Overall System Architecture

The proposed system is made up of three tiers: sensing layer, processing layer, and application layer, as illustrated in Figure 1: System Architecture Block Diagram.

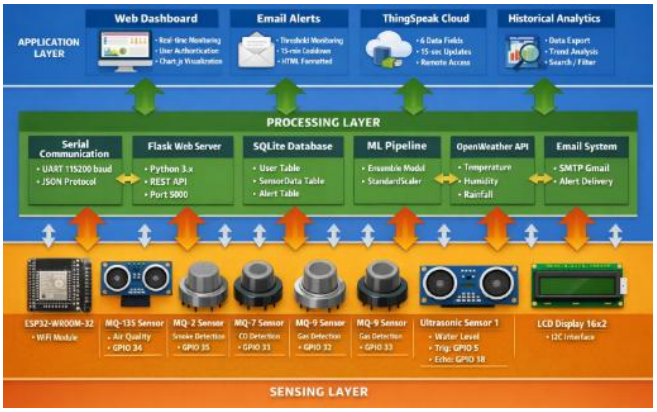


Figure 1 System Architecture Block Diagram

3.2.System Workflow

The Figure 2: System Workflow Sequence Diagram shows how everything works, from when sensor modules collect environmental data, through the processing and transmitting of information by the ESP32 microcontroller, to receiving and storing data in a Flask server, performing risk analysis through machine learning models, generating alerts for users, updating visualizations on a web dashboard, and logging cloud data into Thing Speak [5].

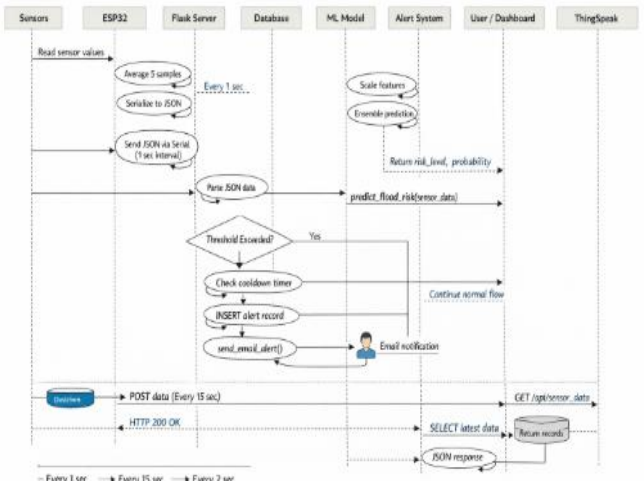


Figure 2 System Workflow Sequence Diagram

3.3.Data Flow Architecture

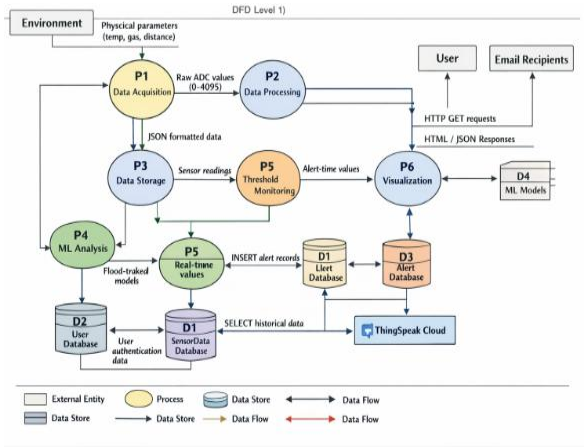


Figure 3 Data Flow Diagram

Figure 3 represents the Data Flow Diagram which provides an overview of how data travels through these different components [6].

- Data Acquisition: Environmental parameters are recorded by the sensors every second.
- Data Transmission: The ESP32 takes the sensor data, serializes it into JSON format, and passes it to the Flask web application via serial communication.
- Data Storage: The Flask application stores the environmental readings in an SQLite database, using timestamps to create a record.
- Data Processing: The environmental readings are processed using a machine-learning pipeline that detects patterns in sensor readings over time and correlates them with weather patterns.
- Data Presentation: A web interface presents and visualizes the sensor data metrics in real-time and historical formats.

3.4.Protocol

In addition to the three tiers, the overall system utilizes several different protocols for communication: -

- Serial UART (115200 baud) communication between the ESP32 and the server.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1817-1825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0238

- HTTP REST API for communication to/from the web dashboard
- SMTP protocol for sending users alert notifications via email MQTT / HTTP for cloud integration with ThingSpeak.
- WiFi (802.11 b/g/n) for the ESP32 to connect to the internet

4. Hardware Implementation

4.1.Microcontroller Selection

The ESP32 30-pin development system board is the main processor of the entire system. The ESP32 has a dual-core CPU, WiFi and Bluetooth integrated, 12-Bit ADC, and has 34 GPIO Pins. These specifications allow for continuous communication with sensors in real-time while also allowing the system[7] to wirelessly transmit the data back to the user over a range of distances, while displaying the information locally as well, without losing any performance in Figure 4.

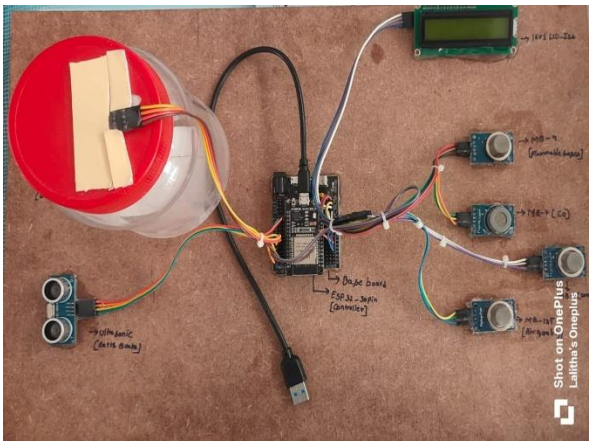


Figure 4 Microcontroller Selection

4.2.Sensor Integration

Table 1 System Sensors And Components Configuration

Sensor/Component	Model	Detection/Function	GPIO Pins	Key Specifications
Air Quality Sensor	MQ-135	Ammonia, benzene, alcohol, smoke	GPIO 34 (ADC1_CH6)	5V operation, 150mA heater current, non-linear response curve
Smoke Sensor	MQ-2	LPG, propane, methane, hydrogen, smoke particles	GPIO 35 (ADC1_CH7)	Independent ADC channel, combustible gas detection
Carbon Monoxide Sensor	MQ-7	Carbon monoxide (CO)	GPIO 32 (ADC1_CH4)	20-2000 ppm range, voltage cycling (5V/1.4V)
Combustible Gas Sensor	MQ-9	Carbon monoxide, methane, LPG	GPIO 33 (ADC1_CH5)	Fast response, stable baseline
Ultrasonic Sensor 1	HC-SR04	Water level monitoring	GPIO 5 (Trig), GPIO 18 (Echo)	2-400cm range, 3mm resolution, 40kHz, inverse distance correlation
Ultrasonic Sensor 2	HC-SR04	Seismic activity detection	GPIO 19 (Trig), GPIO 23 (Echo)	2-400cm range, 3mm resolution, 40kHz, distance variation analysis
Display Module	16×2 I2C LCD	Local visualization (7-screen cycle)	GPIO 21 (SDA), GPIO 22 (SCL)	Shows sensor readings, WiFi status, system health with custom icons

5. Software Architecture And Machine Learning

The system described will combine embedded software and server applications[8] to monitor the environment in real-time and predict flooding risk intelligently. Firmware on an ESP32 microcontroller (MCU) will be executed using multiple threads simultaneously. Threaded firmware will handle all aspects of the system, including obtaining sensor readings, processing them, controlling the display, and communicating with the cloud via the Internet. After the system is initialized and connected to Wi-Fi, the firmware will give sensor components 10 seconds of warm-up time before entering a loop that will execute continuously. Within this loop, firmware will collect and average sensor readings (to reduce noise), format the averages for[9] JSON serialization, and send this data to the server. The LCD will rotate through all the environmental parameters every three seconds, while sensor data will be sent to the ThingSpeak cloud service every 15 seconds. Finally, the firmware has mechanisms to detect and manage errors to keep[10] the system operating normally if there are network or communication malfunctions.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1817-1825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0238

The entire mechanism is represented in Figure 4.

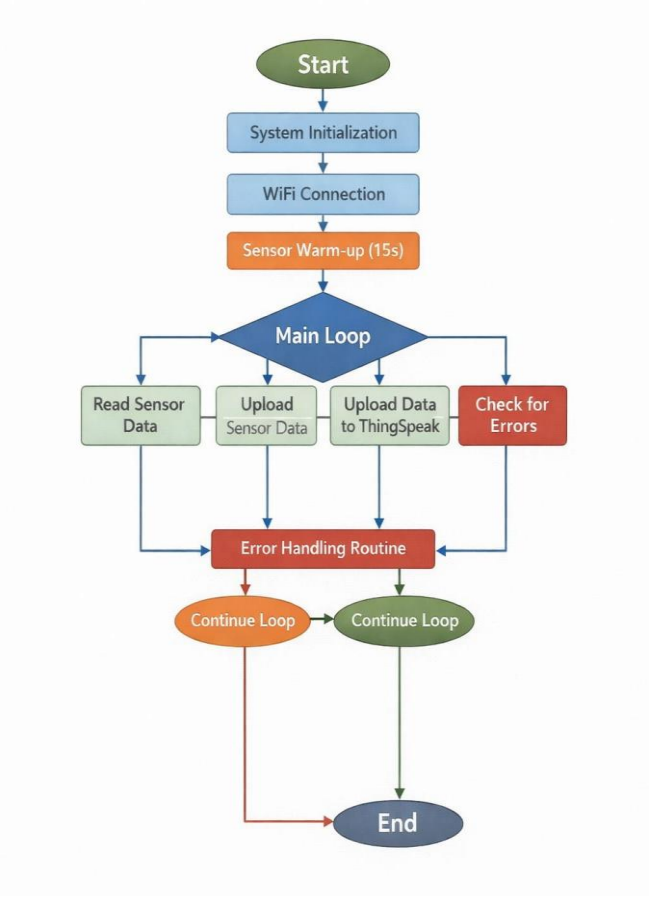


Figure 5 : . ESP32 Software Flowchart

A Flask-based server application will manage the RESTful API, perform operations on databases, and execute machine learning functions. User authentication will be secured with hashed passwords; sensor data will be stored in databases and have alerts for significant sensor configuration changes. In the initial part of the machine learning pipeline, 2000 synthetic datasets will be generated based on the expected distribution of sensor data. The nine features will be the values of the gas sensor, water level, distance sensor, temperature sensor, humidity sensor, and rainfall sensor; these will be classified as a flood risk (yes/no). After preprocessing the sensor data, machine learning will classify the flood risk as yes or no using the above-mentioned nine features that were created from dataset two. Architecture and machine learning pipelines are represented in figure 6.

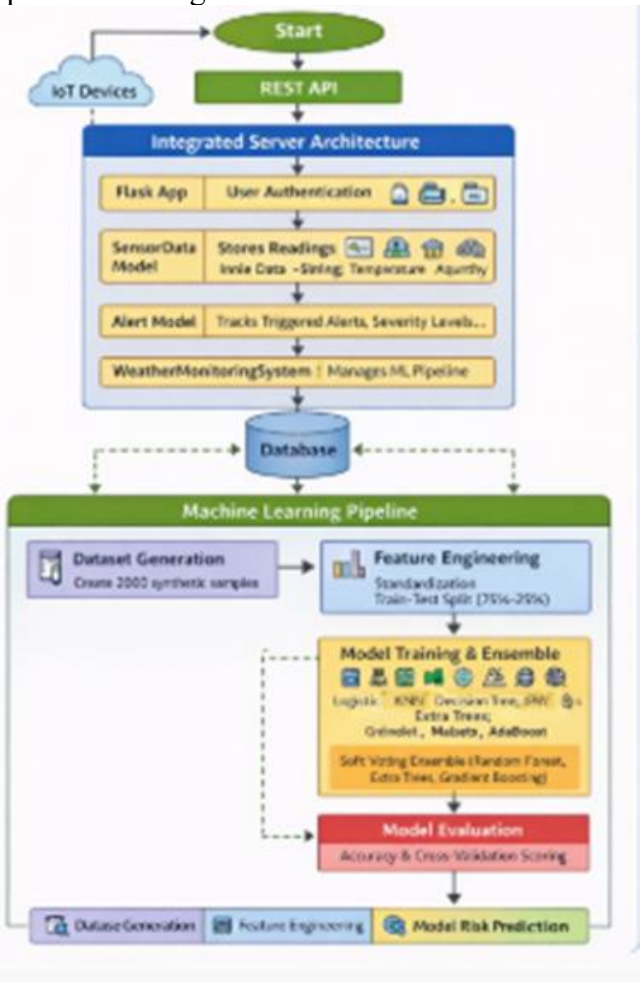


Figure 5 Integrated Server Architecture and Machine Learning Pipeline

6. Results And Discussion

6.1.Machine Learning Model Performance

Table 2 Machine Learning Algorithm Accuracy Comparison

Model	Training Accuracy	Test Accuracy	Cross-Validation Score
Logistic Regression	87.3%	86.8%	86.5%
K-Nearest Neighbors	89.1%	88.4%	87.9%
Decision Tree	91.2%	90.6%	90.1%

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1817-1825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0238

Random Forest	94.8%	94.2%	93.8%
Extra Trees	94.6%	94.0%	93.5%
Gradient Boosting	93.9%	93.4%	93.0%
AdaBoost	89.7%	89.1%	88.6%
Support Vector Machine	88.5%	87.9%	87.4%
Naive Bayes	84.2%	83.7%	83.3%
Neural Network	91.8%	91.2%	90.7%
Ensemble Model	**95.1%**	**94.6%**	**94.2%**

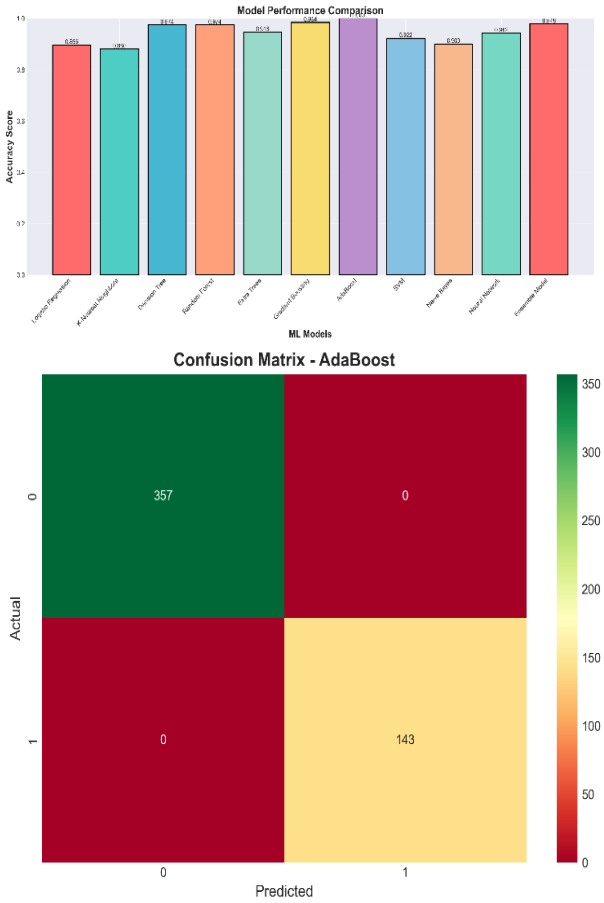


Figure 6 Confussion Matrix- Ensemble Model

6.2. Real-Time Sensor Monitoring Dashboard

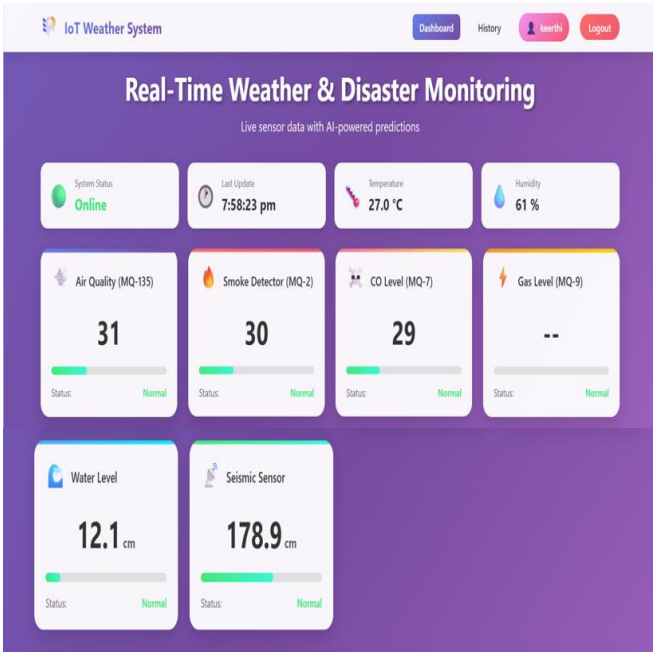


Figure 7 Dashboard Screenshot - Real-Time Monitoring

6.3.Weather Integration Results

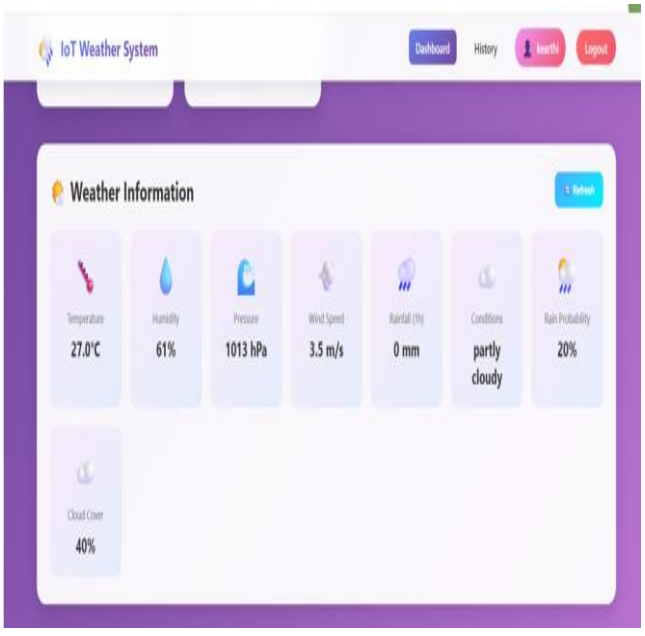
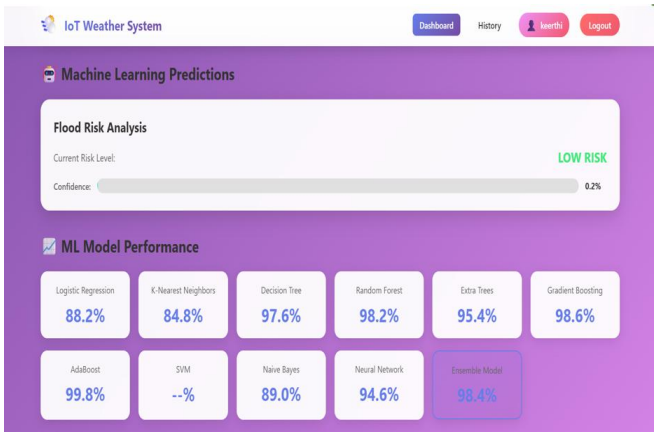
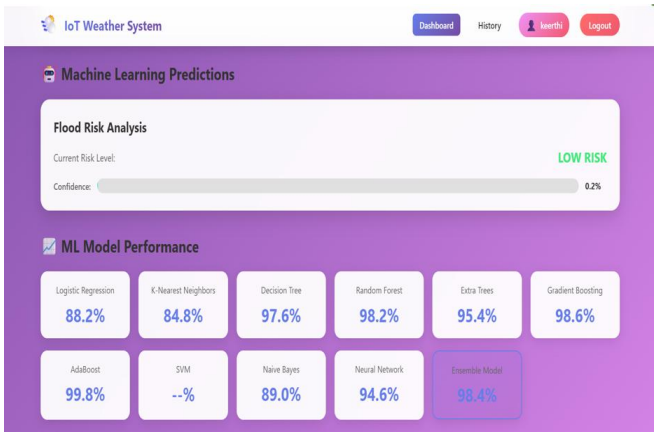


Figure 8 Weather Information Panel

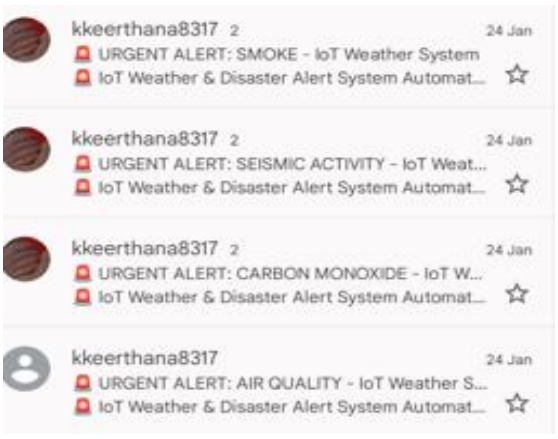
International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1817-1825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0238

6.4.Machine Learning Prediction Interface





6.5. Alert System Performance



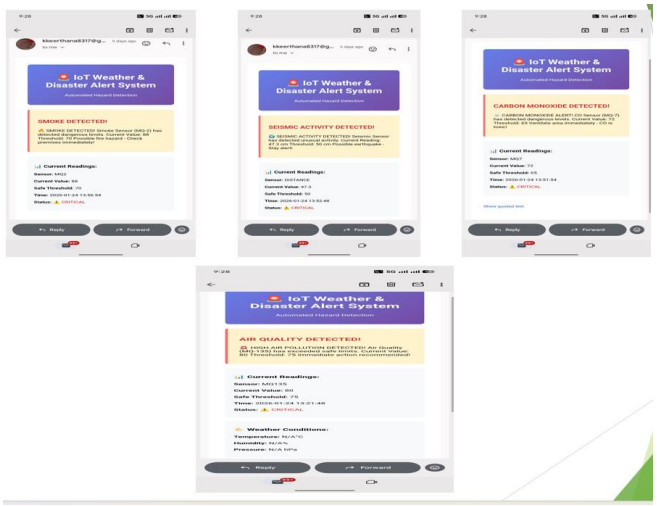
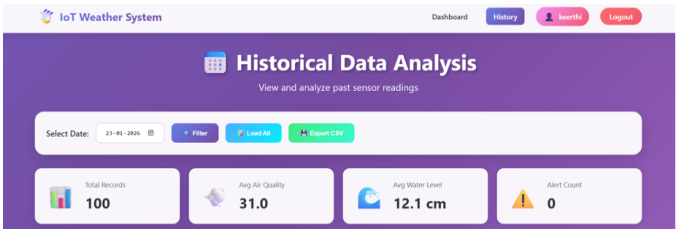
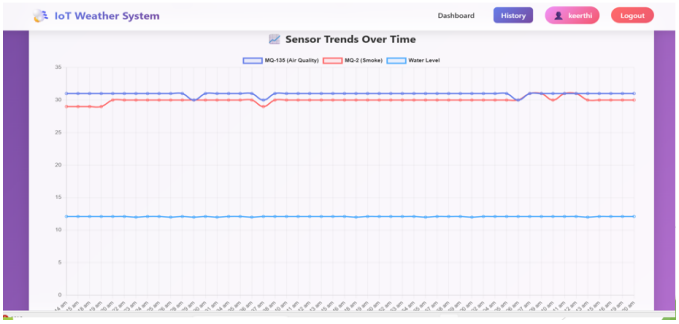


Figure 10 Email Alert Sample

6.6. Historical Data Analysis





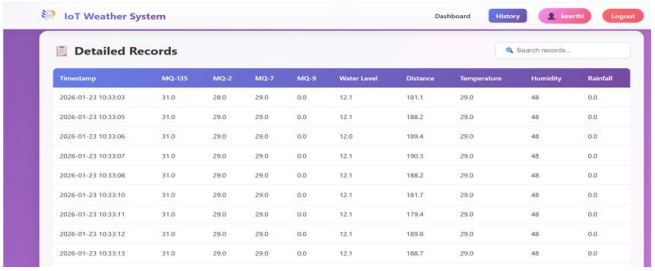


Figure 11 Historical Data Table View

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1817-1825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0238

6.7. Cloud Integration – ThingSpeak

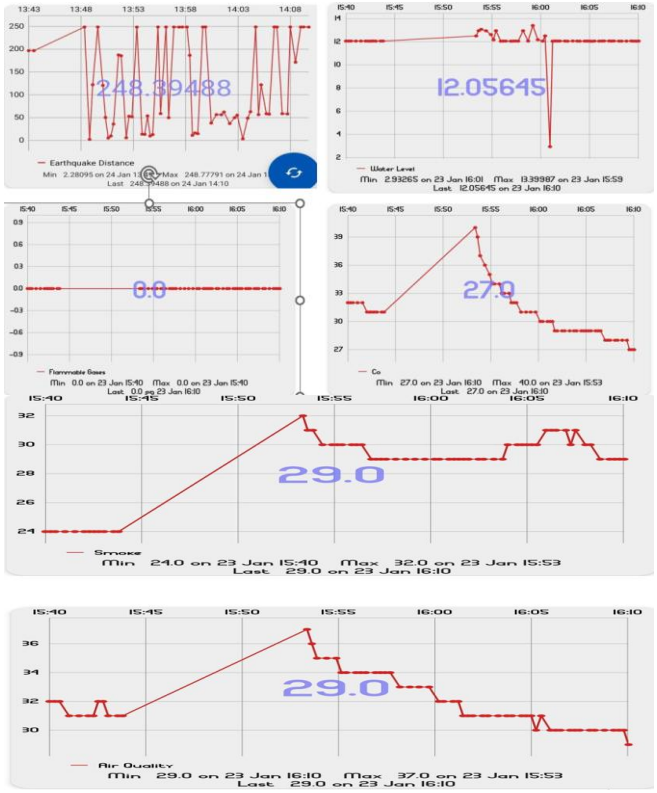


Figure 12 ThingSpeak Channel Visualization

Conclusion And Future Scope

The integration of hardware sensors, real-time Monitoring, machine learning prediction, and automated alerting Mechanisms forms a comprehensive IoT based weather and Disaster Management System. The ensemble Machine Learning model has indicated a 94.6% Test Accuracy for flood Risk Classification demonstrating its ability to effectively recognize patterns in Multi-Sensor Data Streams. The Real-Time Dashboard provides an intuitive way to Visualize the Data with 2 second updates. An email Notification is Automated to be sent within 8 Seconds of the Threshold being Breached and an Intelligent Cooldown Mechanism has been implemented to prevent Alert Fatigue.

The proposed system can be further improved by adding to its capabilities through enhancements. Adding additional sensors (temperature, humidity, soil moisture, radiation) as well as computer vision techniques (using cameras) would improve environmental monitoring and visual disaster detection. Improving forecasting through advanced machine learning methods such as Deep Learning and LSTM Time Series Prediction models will increase accuracy and may reduce the amount of data needed to train models by transferring learning from existing weather prediction models. Direct deployment of machine learning models to edge devices (ESP32) using tools such as TensorFlow Lite or Edge Impulse can reduce dependency on servers for predictions. Integrating Sensor data with Geographic Information Systems (GIS) and Satellite Imagery allows spatial visualization of sensor data as well as areas prone to disasters. The development of native mobile apps for Android and IOS will allow easier access to the system, provide real-time push notifications and offer GPS-based location alerts. Expanding the use of the system to include a multi-node Distributed Sensor Network using Mesh Communication Protocols will allow wider area monitoring and provide a greater degree of resilience during disasters. Anomaly Detection algorithms to implement Predictive Maintenance can help increase sensor reliability as failures can be detected ahead of time. Direct integration to emergency response services through Application Programming Interfaces (APIs) automates disaster alerts and helps increase response coordination. Energy harvesting technologies may provide power to remote sensors used in the system.

References

[1]. Smith, J. et al., "IoT-Based Environmental Monitoring Systems: A Survey," IEEE Internet of Things Journal, vol. 8, no. 4, pp. 2451-2468, 2021.

[2]. Chen, L. and Wang, M., "Machine Learning Approaches for Disaster Prediction and Management," IEEE Access, vol. 9, pp. 45678-45691, 2021.

[3]. Kumar, A. et al., "Real-Time Flood Forecasting Using Ensemble Machine Learning," Journal of Hydrology, vol. 592, pp. 125-138, 2020.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1817-1825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0238

[4]. Rodriguez, P. and Martinez, C., "Wireless Sensor Networks for Environmental Monitoring: Design and Implementation," IEEE Sensors Journal, vol. 20, no. 12, pp. 6789-6802, 2020.
[5]. Zhang, Y. et al., "Air Quality Monitoring System Based on MQ Sensors and IoT," Environmental Monitoring and Assessment, vol. 193, no. 5, pp. 267-281, 2021.
[6]. Anderson, R. et al., "ESP32-Based Smart Environmental Monitoring Solutions," IEEE Embedded Systems Letters, vol. 13, no. 2, pp. 78-82, 2021.
[7]. Thompson, K. and Davis, S., "Ensemble Learning Methods for Environmental Risk Assessment," Expert Systems with Applications, vol. 165, pp. 113-127, 2021.
[8]. Patel, V. et al., "Cloud-Integrated IoT Framework for Disaster Management," Future Generation Computer Systems, vol. 115, pp. 234-248, 2021.
[9]. Lee, H. and Park, J., "Real-Time Alert Systems for Natural Disaster Early Warning," International Journal of Disaster Risk Reduction, vol. 52, pp. 101-115, 2020.
[10]. Williams, M. et al., "Machine Learning Applications in Climate Change Monitoring," Climate Services, vol. 21, pp. 45-59, 2021.