iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70589

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Cloudburst Prediction System

Amruth Raj P[1], Katta Vinod Kumar[2], K Vishnu Vardhan[3], Nisha L[4], Rajeshwari C Raikar[5], Mr. Rajan Thangamani[6]
[1, 2, 3, 4, 5]*Student, CST DevOps (of Aff) Presidency University (of Aff) Bengaluru, India*
[6]*Assistant Professor, Computer Science And Engineering (of Aff.) Presidency University(of Aff.) Bengaluru, India*

Abstract: ***The Cloudburst Forecasting System has been created to enhance the precision in cloudburst forecasting. By utilizing AI models, this system processes weather reports and satellite and radar imagery to gauge the potential for an impending cloudburst and issues adequate early warnings. Improved accuracy ensures less life and property being threatened. Having GIS technology support will assist in equitably allocating resources since decision-makers can then factor in some physical attributes of housing stock that influence the vulnerabilities of the affected group. This would further lead to the advanced planning and resource development at both self-help and community levels so that the response teams can undertake the forthcoming disaster response after the occurrence. Some positive implications include public preparedness drive, optimum use of resources, and a comprehensive approach to financial loss mitigation. However, the bigger impediments are going to continue into present weather models/systems, inadequate data for cloudburst events, along with computational power being the field evermore.***

I. INTRODUCTION

Many advanced weather forecasting models generate gigantic chunks of data in the name of temperature, humidity, atmospheric pressure, wind velocity, and cloud density. Despite virtually infinite opportunities that lie there for the application of such data in weather forecasting, the complexity and vastness mount up to a level where physical barriers stand in the way of value application toward an operational swift and intelligent decision. Such information could be ideal for forecasting temporary weather phenomena like a cloudburst. The agencies engaged in aftermath operations and the meteorologists could derive beneficial knowledge by correlating past scenarios with present atmospheric conditions, keeping in mind the possibility of occurrence or magnitude or impact area.

In the past, weather monitoring approaches were based on low-level threshold techniques: time-consuming, prone to errors, and limited when it came to scaling or generalizing to districts. Now that artificial intelligence and machine learning have emerged, it is possible to build independent data-driven systems that can capture complex atmospheric information accurately and with greater velocity. However, most of the present-day solutions tend to be very specialized to a given set of input parameters or lack tight integration with real-time live feeds and user-friendly interfaces for practical operation.

This work comes into the picture against such limitations through an AI web-based solution.

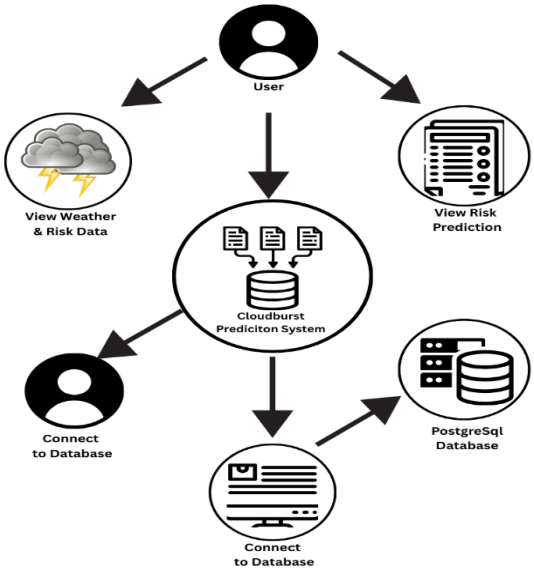


Fig. 1. Use Case Diagram

The workflow for a cloudburst prediction system involves:

1) Weather and Risk Data Collection: Weather data such like temperature, humidity, pressure, wind speed , and cloudiness , batch weather data for several cities is retrieved online or in batch mode and persisted in the database.
2) Computation of Risk Prediction: The cloudburst risk levels are computed using a predictive model or heuristic based on observed data by the system. Each record carries a timestamp and prediction confidence score.
3) Showing Weather and Risk Data: Users can also view through the API the general current weather condition with levels of associated risk in order to identify some areas that might be vulnerable.
4) Update data: Patient data can be streamlined to ensure that comparisons use the most recent information.
5) Access to Historical Risk Data: The previous year's risk forecast has been accessed. This allows for identifying what trends or patterns have occurred over the years.
6) Statistical SummaryGeneration: The captured statistics (high risk, low risk areas) generated for informing about decision making as well as emergency preparedness would be.
7) Data Base Integration: The backend system is supposed to be connected with PostgreSQL database for safety, efficiency and return of data.
8) Security of System and API Access: This will enable CORS secure front-end access to the APIs. The modular update and scalability are embedded with the Flask system architecture.

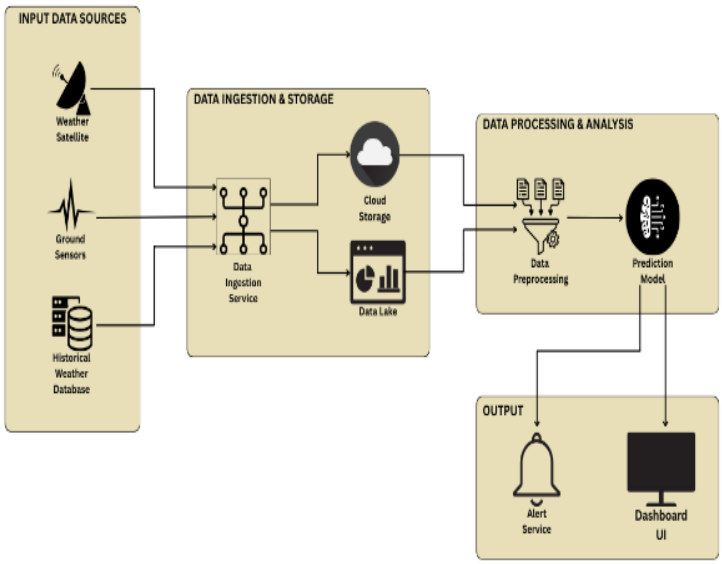


Fig. 2. Architecture

The Architecture includes:

1) Data Sources:GeoNames API and OpenWeatherMap API provide real-time weather data.
2) Data Layers:Separate Data Collector pulls the weather data from those two APIs.
3) Data Preprocessing:This cleans and organizes the raw data for further analysis.
4) Dimensionality reduction: Apply TF-IDF Vectorization for the lowering of dimension of patient data while retaining significant features.
5) Machine Learning Model:Trains a Random- forest model i.e. 100 trees on the engineered features.
6) Prediction Pipeline:Live prediction system feeds input data via the trained model and predicts as either "High Risk" or "Low Risk."
7) Database and storage: Predictions and their entire data takes place in a PostgreSQL database.
8) Continuous model improvement: Feedback loop exists for retraining the model for much greater accuracy in predictions.
9) Deployment:Entire flow could be deployed as end-to-end cloudburst prediction pipeline.

II. LITERATURE SURVEY

A. Literature Review

1) Cloudburst and Disaster Early Warning Systems:

Cloudbursts are brief and heavy showers of precipitation and hold the capacity to create flash floods, landslides, and significant damage to infrastructure. Various real-time flood detection and warning systems have been researched and analyzed by scientists and agencies. The conventional systems depend greatly on weather stations, satellite images, and Geographic Information Systems (GIS). Still, most of such solutions prove either not real-time responsive or available to the common public or local governments.

2) Web-based Weather Monitoring Tools:

Web-based weather monitoring tools have become increasingly popular over the last several years because of accessibility and cost-effectiveness. General weather data is available primarily on websites such as the Indian Meteorological Department's (IMD) Nowcasting portal and worldwide weather dashboards (e.g., Windy, AccuWeather). These are chiefly centralized and are not tailored for individual risk visualizations such as cloudburst probability or local risk assessment.

3) Use of React in Interactive Dashboards:

React is normally applied in the creation of dynamic and interactive single-page applications (SPAs). In environmental monitoring dashboards, React allows for modular development and live updates through APIs and WebSockets. Research has emphasized React's performance benefits and component reuse, which are perfect for dashboards with constant data refreshes and conditional rendering.

4) Flask for Lightweight Backend APIs:

Flask, as a Python micro web framework, is most commonly preferred for backend API services because it has low setup and high flexibility. It easily plays along with data processing libraries such as pandas and can handle RESTful API development. Comparatives between Django and Flask research reveal that Flask is best suited for small- to mid-scale applications where speedy development and customized routing are necessary—perfect for your real-time weather API.

5) Business Intelligence Integration in Power BI:

Microsoft Power BI is a business analytics service offered by Microsoft for interactive visualizations and embedded reporting. Recent emphasis has been shown depicting its integration into corporate dashboards, particularly environmental visualization of information. Power BI, as compared to open-source charting libraries, provides more drill-down, natural language querying, and cloud hosting options. Web application embedding of Power BI extends analysis capability without the need to rebuild visual pipelines.

6) AWS EC2 Cloud Deployment:

Cloud computing platforms such as Amazon EC2 enable scalable, on-demand hosting of end-to-end stack applications. Monitoring systems hosted by EC2 benefit from support such as remote access, scalability, and compatibility with other AWS services such as S3 or CloudWatch. Past studies in disaster monitoring platforms have indicated that EC2 is well-suited to host web services, APIs, and data visualization components with minimal downtime.

B. Research Gaps in Existing Solutions

1) Late or Inaccurate Alerts: In generating alerts, classical systems using synoptic meteorological models and satellite images tend to give such alerts either too late, during the cloudburst, or too early before the occurrence of the event. They lack prophetic capability for the early warning.
2) Lack of Granularity: Weather forecasts rendered by national and international forecasting systems across regions and districts overlook hyper-local aspects of cloudbursts. This leads to a mismatch between the information generated and the knowledge required to assess danger.
3) Lack of Real-Time Integration:Real-time weather data feeds usually remain unlinked from most of the models. These static forecasts or bulletins that are on a daily basis just cannot suffice for sudden changes in weather conditions.
4) Limited Use of Predictive Modeling: Some available tools operate on threshold-based rule systems or are expert systems. They do not fare well when generalized to different topographies and weather profiles.

5) Absence of User-Centric Interfaces: Most of these forecasting systems are technologically oriented and do not provide ease of use in a dashboard or through APIs for straightforward integration into apps or city-scale early-warning systems.

C. *Addressing the Gaps with the Proposed System*

1) Machine-Learning-Based Forecasting: Through applying a Random Forest classifier to past weather data with current situation inputs, this system identifies complicated nonlinear relations among parameters to predict more accurately.
2) City-Level Forecast: Your system addresses a couple of cities in India through GeoNames API. This is what allows local predictions, filling up the spatial vacuum existing with most national models.
3) OpenWeatherMap Live Weather Feed: The application gleans live weather data for temperature, humidity, pressure, wind speed, and cloudiness, among others, for dynamic risk computation.
4) Flask-Based API Interface: The fairly lightweight and secure REST API is said to expose the forecast to be integrated into third party systems or frontend dashboards.
5) React Interactive Dashboard:The UI portal allows users to view city-wise weather statistics, risk status, and real-time status boards—all near real-time updates.

The review points to the need for a robust, scalable, and interpretable solution for the analysis of a cloudburst. The system presented here supplements existing methods while addressing certain critical gaps in order to provide better results for cloudburst prediction.

III. METHODOLOGY

The system is a multistage pipeline to ensure accuracy and real-time response. A report of Indian cities is first received via the GeoNames API as it provides the necessary geospatial data (latitude-longitude). This data is then fed into the OpenWeatherMap API to obtain real-time weather data.

1) *Data Storage and Data Preprocessing:*

Structured weather data are stored in PostgreSQL.
The data are cleaned and validated before being fed into the model.

2) *Model Training:*

An offline Random Forest model would be trained by a labeled weather data set.
Cloudiness > 50% is proxy labeling of "High Risk."

3) *Prediction Pipeline:*

High-risk condition symbols are the output of the model for each new record.
Raw scores and predictions sit in a separate table for auditability.

4) *API and Frontend Integration*

Endpoints to be consumed include /api/weather, /api/risk, and /api/stats.
Predictions and city-level views are served by the React frontend.

IV. RISK PREDICTION

A. *Risk Prediction Model:*

The risk estimation model uses the Random Forest classification method: a very famous ensemble learning method known for its stability and interpretability. The model learns from a data set consisting of some relevant weather parameters: temperature, humidity, pressure, wind speed, and cloudiness. Each data record is labeled "High Risk" or "Low Risk" depending on some threshold values assigned to cloudiness (here, any value above 50%). These variables are normalized using MinMaxScaler in order for the values to be under a uniform range prior to learning, thus ensuring better accuracy in modeling. Given that this data will be used for learning purposes, it has been split into two sets sub-training and test sets, in the ratio 80:20, to establish enforceability of the model. At inference, the model takes real-time weather data and produces a binary prediction and confidence score. The forecast is indexed by timestamp and city ID so that one can perform time-series analysis. The model is therefore truly at the core of making decisions within the system that effectively translates the most basic weather weather inputs to actionable intelligence in cloudburst risk management.

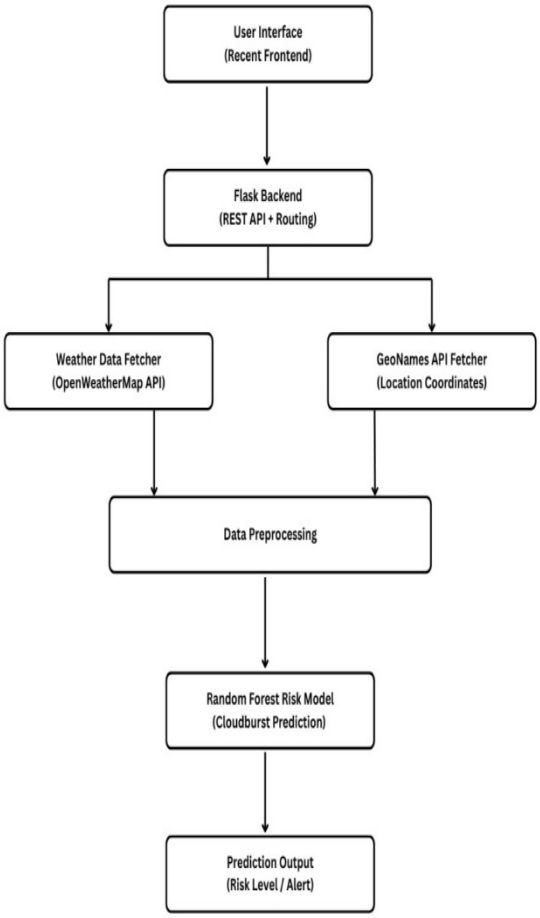


Fig. 3. Block Diagram

V. IMPLEMENTATION

The Cloudburst prediction system designs for Cloudburst harnessed an assortment of tools, libraries, and frameworks, so that it remains efficient, scalable, and easy to utilize.

1) Frontend (React+Vite):The frontend uses React and Tailwind CSS for a straightforward and capable UI. The buttons are switched immediately from the weather table view to the Recharts risk chart view and Power BI report view. The main app, App.jsx, loads components dynamically via React state.

2) Backend (Flask): The backend is constructed with Flask. The API (api.py) sports /api/weather and /api/risk endpoints, powered by respective helper functions defined in data.py.
They simulate or obtain weather and risk data via simple Python dictionaries and pandas.

3) Power BI Integration: Power BI is integrated into an iframe inside PowerBIEmbed.jsx.
The embedded URL of published Power BI report displays dynamic reports of risk analysis.
Smoothening of iframe allows interaction with the embedded report.

4) Deployment (AWS-EC2):The full stack is deployed on an AWS-based EC2 instance. The backend is an API implemented by Flask, while React is compiled and served statically.

Ports and security groups have been configured to give external access.

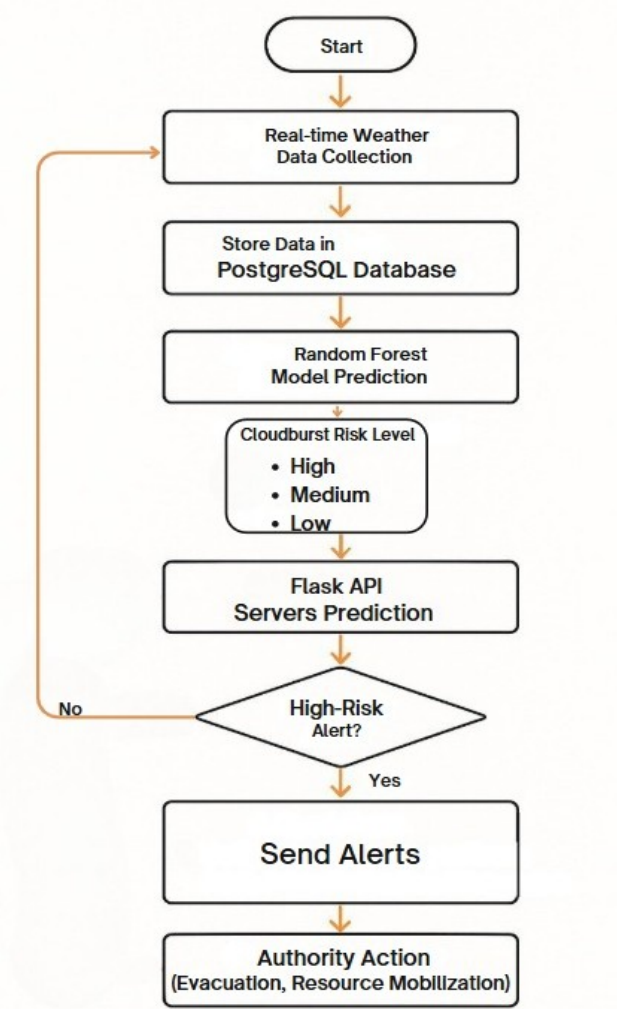


Fig. 4. Flow chart

VI. RESULTS AND DISCUSSIONS

1) Result:

The Cloudburst Risk Dashboard is a much potent combination for the real-time display of data, risk categorization, and analysis drawn into a web-based interface. It is recalled as showcased in the deliverables:

2) The Live Monitoring of Weather:

Weather data are given dynamically to all simulated data for different stations.

Data includes temperature, humidity, and accurate timestamp formatting.

3) Visualization of Risk Distribution:

Pie charts created with Recharts really show percentages of “Low”, “Moderate”, and “High” levels of risk.

Quite simple of a component visually, another update originates from the new data obtained from the backend.

4) Embedding of Power BI:

Power BI reports of production grade have been embedded into the dashboard.

Users can drill through charts and slicers to see detailed risk-trends and breakdowns.

Strongly interactive report responds to user filters within the iframe itself.

5) Responsive and Modular Interface:

Landing page buttons support toggling among components.

UI optimized for desktop and mobile view.

6) Successful Deployment on AWS:

The entire system is in the test phase of operation on an EC2 instance in the cloud.

The application is accessible through public IP/domain from anywhere, and it all just works fine in production.

7) Discussion:

The project was operationalized to monitor an environmental situation-making dashboard.

8) Performance and Modularity issues:

With React frontend setup for modularity at the component level, this ensured future maintainability and scalability (more visualization components or more data sources, for instance).

9) Backend flexibility:

Since Flask is a lightweight framework, it suited the API layer perfectly because of such simplicity. Since Python-based data simulation logic was available, it allowed for rapid prototyping.

10) Enterprise Analytics:

Embedding in Power BI gave the dwindling community a vibrant interactive analytical interface. Embedding into a React application indicates a front end that could promote enterprise analytics onto public dashboards.

11) Deployment View:

Hosting on AWS EC2 would enable worldwide reach for this dashboard. On the production side, there do remain some concerns regarding scalability, which are still kept away from being stripped (eg., API auth, HTTPS setup).

12) Limitations:

- Weather and risk data are now simulated.
- Real-time consumption from APIs like OpenWeatherMap or Indian Meteorological Department can be added to improve the dashboard.
- No alerting system is present currently (e.g., SMS/email).

VII. CONCLUSION

The process outlined comprises product design, development, and deployment for the Cloudburst Risk Dashboard using the latest web technology and bringing in real-time data visualization and cloud deployment. The system presents React frontend, Flask as backend API, and Power BI for enterprise-grade visual analytics to render an interactive tool for cloudburst risk monitoring that is user friendly.

The dashboard presents users the ability to see incoming weather information, to see risk level distributions via visualizations, and to engage with the BI reports in context. Hosting on AWS EC2 ensures global availability and remote monitoring capability.

Although this system presently works with mock data, it sets a very strong foundation for any enhancements that will be made in the future: real-time weather API integration, predictive analysis with machine learning, and alerting automation, just to name a few. Overall, the project supplements the fledgling body of disaster risk visualization and serves as a foundation for more responsive data-driven environmental monitoring systems.

VIII. FUTURE WORK

The configuration of the Cloudburst Risk Dashboard is only one possible idea for the time being. Most of these can be bettermented, perhaps, so that they will be more precise and user-friendly. More-efficiency is an option as well:

1) Working with Real-Time Data Sources:

For now, simulated weather and risk data are considered and used. But in the near future, meteorological APIs may start to bridge real-time data from:

- OpenWeatherMap, IMD Nowcast API, or NOAA for temperature, humidity, and rain.
- GIS passing layers for intensity mapping of rainfall and flood areas.

2) Machine Learning-Based Risk Prediction:

Machine learning would as well allow on-the-fly classification of risk based on data. Also, in minor modifications:

- Rainfall forecast on time series models such as LSTM, ARIMA.
- Cloudburst classification with historic data sets and classification rules.

3) Alert and Notification System:

Furthermore, warnings could be communicated to authorities or users through:

- Alerts via SMS or Emails in case of occurrences pertaining to any high-risk conditions.
- Interconnection with public warning systems or local emergency services.

4) Geospatial Visualization:

Provision of interactive maps based on Leaflet.js, Mapbox, or Google Maps API so as to:

- Geospatially visualize weather data.
- Highlight high-risk zones in accordance with overlays of live data.

5) User Width and Access Control:

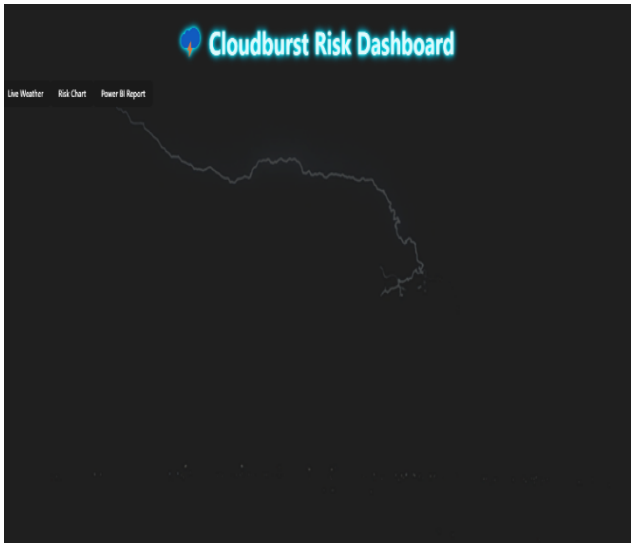
This type of deployment implements user-based access for:

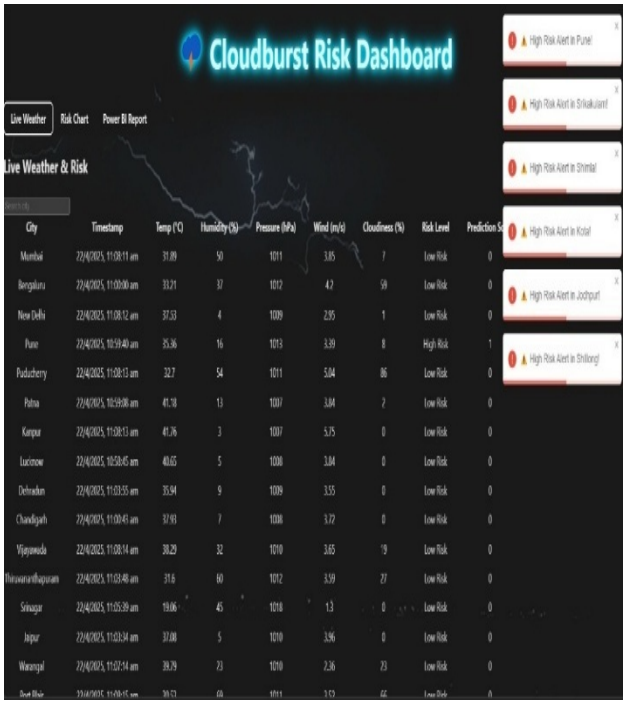
- Authorities only able to access the Dashboard.
- Login on the basis of Region to access data or type-trends-of-historical.

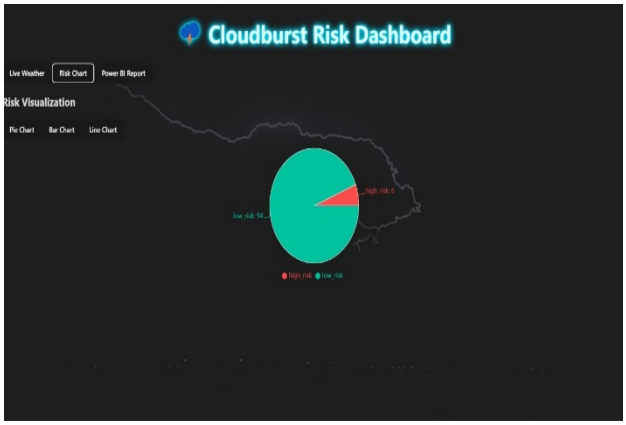
6) Scalability and Hosting Improvements:

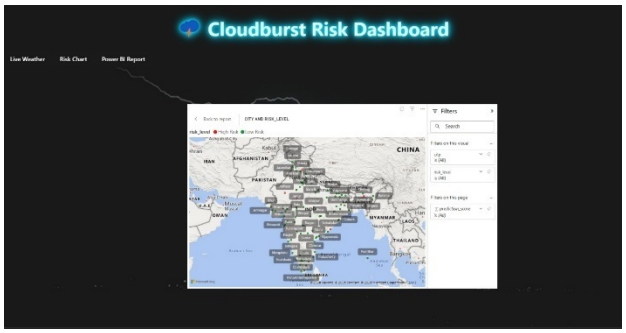
The changes for performance and scalability improvements go:

- Porting the back-end into Docker containers.
- Serverless architecture to be hosted by AWS Lambda, while CloudFront will be engaged as the CDN.
- Token-based mechanism to protect APIs, JWT implemented, and SSL mandated for enforcing HTTPS.









CITATIONS

[1] Banerjee, M., & Choudhary, S. (2022). Integrating remote sensing data for early detection of extreme rainfall events. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 15, 672-684. https://doi.org/10.1109/JSTARS.2022.3149876

[2] Girish, G. A., Anjum, A., Ganesh, G., Hegde, G. N., & Gowtham, S. V. (2024). The Cloudburst Prediction System. Journal of Emerging Technologies and Innovative Research (JETIR), 11(5), 363-366. Retrieved from www.jetir.org.

[3] Lee, J., Kim, H., & Park, C. (2024). Satellite-based rainfall prediction using convolutional neural networks. IEEE Transactions on Neural Networks and Learning Systems, 34(3), 10501062. https://doi.org/10.1109/TNNLS.2024.3267892

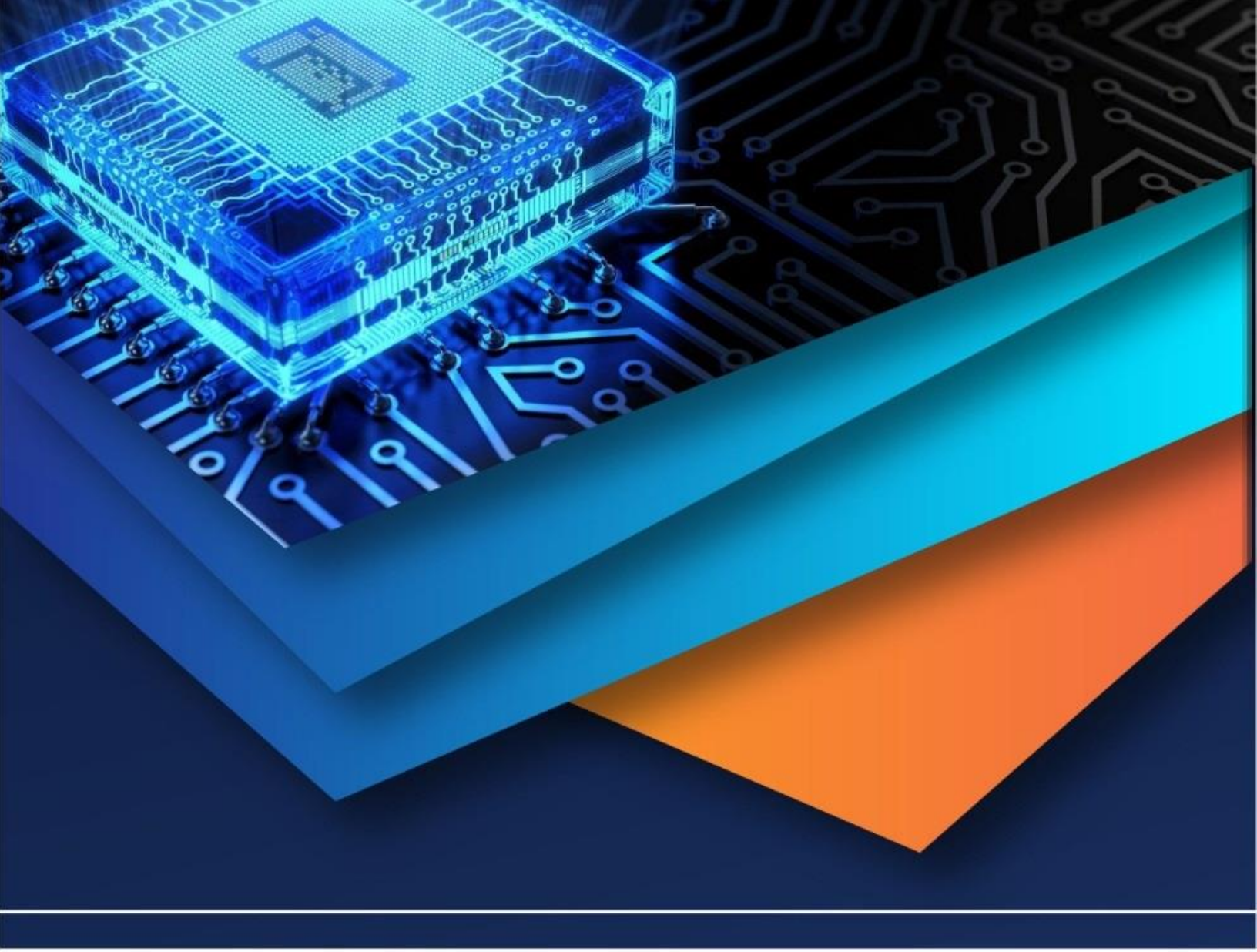
[4] Patel, A., & Gupta, M. R. (2021). Deep learning model for heavy rainfall and cloudburst forecasting. IEEE Access, 8, 122345-122356. https://doi.org/10.1109/ACCESS.2021.3098765

[5] Singh, S., Kumar, R., & Sharma, A. (2022). Machine learning-based cloudburst prediction using meteorological data. IEEE Transactions on Geoscience and Remote Sensing, 60, 1-10. https://doi.org/10.1109/TGRS.2022.3145678

[6] Sivaprakash, V., Sharathi, R., Karunya, V., & Umashankari, V. (2024). Cloudburst Prediction System (CBPS). International Journal of Research Publication and Reviews, 5(4), 1606-1611. Retrieved from www.ijrpr.com.

[7] Telsang, S., Sawale, P., Pujari, S., Pujari, V., Pungale, A., Dubewar, R., &Shendre, R. (2024). Cloudburst Prediction System Using Machine Learning. International Journal for Research in Applied Science & Engineering Technology (IJRASET), 12(11), 1680-1689. Retrieved from www.ijraset.com.

[8] Verma, K., Sharma, P., & Mehta, N. (2023). Real-time weather monitoring and cloudburst prediction system using IoT and AI. IEEE Internet of Things Journal, 9(5), 3356-3364. https://doi.org/10.1109/JIOT.2023.3265432



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY