ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue X, October 2025

Development of a Lightweight Weather Forecasting Application Using Javascript

[1]Satendra Pathrol, [1]Jagdeep Singh, [1]Manoj Kumar, [1]Sachin Kumar, [1]Sharad Kumar, [2]Vikas Sharma

[1]School of Engineering & Technology, Shri Venkateshwara University, Gajraula, U.P. India

[2]Department of Computer Applications, SRM Institute of Science and Technology, Delhi NCR Campus, Ghaziabad, U.P. India

DOI: https://doi.org/10.51583/IJLTEMAS.2025.1410000061

Abstract: Weather forecasting plays a vital role in various domains, including agriculture, transportation, and disaster management. This paper presents the development of a lightweight weather forecasting application using JavaScript, designed to provide real-time weather information through a web-based interface. The application leverages publicly available weather APIs for data retrieval and integrates responsive design principles to ensure usability across multiple platforms. By utilizing JavaScript's asynchronous capabilities and client-side rendering, the system delivers an interactive and efficient user experience without requiring heavy server-side computation. The proposed solution emphasizes simplicity, portability, and accessibility, making it suitable for users with limited computational resources. Experimental evaluation demonstrates that the application achieves fast response times and reliable accuracy in displaying weather conditions, thus offering a practical and cost-effective tool for real-time weather monitoring.

Keywords: Weather forecasting, JavaScript, Web application, Real-time prediction, API integration, Lightweight systems.

I. Introduction

Weather prediction has become an integral component of modern society, influencing critical decisions in agriculture, aviation, maritime operations, transportation, and even daily human activities. Accurate and timely weather forecasts allow individuals and organizations to make informed decisions that reduce risks, optimize resources, and enhance safety. For example, farmers rely on weather predictions for irrigation and crop management, while city administrations utilize forecasting systems to plan for floods, storms, and other extreme weather events. The growing need for accessible and reliable weather data has driven the advancement of forecasting systems and their integration into digital platforms such as mobile applications and web services. Traditional weather prediction systems are often powered by complex meteorological models that require substantial computational power and specialized hardware. While these models are indispensable for large-scale forecasting, their accessibility to the average end-user is limited. Most individuals rely on web-based platforms or mobile applications that aggregate weather information from global meteorological data sources. As a result, lightweight applications that deliver real-time weather information with minimal computational overhead are becoming increasingly relevant. The rise of web technologies and open weather APIs has further democratized access to meteorological data, enabling developers to create cost-effective and user-friendly solutions. JavaScript, one of the most widely adopted programming languages for web development, plays a pivotal role in building interactive and responsive applications. With its asynchronous capabilities and compatibility across multiple browsers and devices, JavaScript has evolved beyond its traditional role of client-side scripting. Modern JavaScript frameworks and libraries provide developers with powerful tools to fetch, process, and visualize real-time data. In the context of weather prediction, JavaScript can be leveraged to retrieve meteorological information through APIs, process it efficiently, and present it in an intuitive user interface. Its ability to function across platforms ensures that weather forecasting applications developed with JavaScript remain lightweight, portable, and accessible to a broad user base. G. N. et al. [1] demonstrated the development of a weather forecasting web application, highlighting the efficiency of lightweight designs in providing real-time weather updates through a user-friendly interface. This approach aligns with the growing need for accessible and responsive forecasting platforms for end-users. In parallel, research has increasingly focused on integrating AI-driven techniques for renewable energy forecasting. Recent advancements in open data access have made weather APIs such as OpenWeatherMap, WeatherAPI, and AccuWeather popular sources of meteorological data. These APIs provide real-time updates on temperature, humidity, precipitation, wind speed, and forecasts spanning multiple days. By integrating such APIs into JavaScript-based applications, developers can create solutions that bypass the complexities of raw meteorological computation while still delivering reliable and timely forecasts. This approach not only reduces the computational burden but also enhances user experience by offering instant updates and visualizations. Furthermore, the proliferation of mobile and low-powered devices in both urban and rural areas underscore the importance of lightweight applications. Many regions, especially in developing countries, may not have consistent access to high-performance computing systems or fast internet connectivity. A lightweight JavaScript-based weather forecasting application provides an efficient solution by consuming minimal resources while ensuring rapid response times. Such applications empower communities with critical weather information, which can aid in disaster preparedness and improve quality of life. The motivation for developing this research stems from the need to create a simple, accessible, and effective weather forecasting solution that prioritizes efficiency and usability. Unlike heavy desktop applications or resource-intensive services, the proposed JavaScript-based system is designed with end-user accessibility in mind. It adopts responsive design principles to ensure cross-device compatibility and leverages asynchronous JavaScript functions to maintain seamless interaction without overloading system resources.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue X, October 2025

II. Literature Review

Recent advances in weather prediction systems have leveraged web technologies, artificial intelligence, and IoT-enabled solutions to enhance forecasting accuracy and accessibility. Yan et al. [2], [3] proposed a probabilistic photovoltaic (PV) power forecasting model using a multi-modal method and GPT-Agent to interpret weather conditions, thereby enhancing renewable energy management under uncertain atmospheric scenarios. Similarly, Zheng et al. [4] introduced GraphCast-Qtf, a model incorporating uncertainty quantification methods for improved weather prediction, emphasizing the role of deep learning in refining prediction accuracy. Beyond weather forecasting, AI and IoT solutions have extended to agriculture and environmental management. Pajila et al. [5] explored AI-driven crop management and market solutions, demonstrating how predictive analytics can optimize farming decisions. Manandhar et al. [6] presented a cost-effective flood prediction platform that integrates weather monitoring with disaster management, while Suman et al. [7] designed a micro-weather station with cloud connectivity and mobile monitoring to support local renewable energy initiatives. Complementary to these efforts, Herrera Acevedo et al. [8] developed LUMA, an open-source web observatory framework that empowers solar research through image analysis tools, promoting accessibility and transparency in environmental data usage. Other studies have explored broader applications of intelligent systems and IoT in environmental monitoring and user-focused solutions. Somnathe et al. [9] proposed a brain–computer interaction framework for assisting individuals with speech and motor impairments, illustrating the adaptability of deep learning across domains. Ganchev and Ji [10] developed an IoT-based air quality index (AQI) monitoring and control system, demonstrating how IoT can provide real-time environmental feedback for urban sustainability. In the domain of tourism and environmental planning, Kumar et al. [11] designed a mobile application for recreational suitability analysis of beach locations, showing how GIS, mobile computing, and environmental factors can guide decision-making. Furthermore, Madjid et al. [12] advanced the use of digital twins by designing a cGAN-LSTM-based greenhouse climate control system, enabling intelligent monitoring and prediction for precision agriculture. Sharma et al. [13] conducted a comparative study of various deep learning models, including convolutional and recurrent neural networks, to evaluate their performance on benchmark datasets. Their findings revealed significant differences in accuracy and generalization ability across models, suggesting that hybrid or ensemble-based approaches may outperform individual architectures. For instance, Sharma and Teotia [14] provide a comprehensive analysis of security mechanisms, highlighting issues such as routing attacks, denial-of-service (DoS), and blackhole vulnerabilities. Sharma and Teotia [15] propose an optimized GNN-based IDS for dynamic MANETs that combines lightweight architecture, online adaptation, and dynamic graph representations. Collectively, these studies demonstrate a consistent trend towards integrating lightweight application design, artificial intelligence, and IoT for enhanced prediction and decision-support systems across domains such as weather forecasting, renewable energy, agriculture, disaster management, and environmental monitoring. While AI-driven models improve predictive accuracy, lightweight and modular designs ensure accessibility and efficiency for diverse users. Building on these insights, the present work proposes a lightweight weather forecasting application using JavaScript, which emphasizes performance optimization, cache efficiency, and cross-platform usability while maintaining real-time accuracy in weather prediction.

III. Proposed Methodology

The proposed methodology focuses on the design and implementation of a lightweight weather forecasting application using JavaScript, with the primary objective of ensuring efficiency, accessibility, and usability across different platforms. The methodology consists of several systematic stages: data acquisition, preprocessing and integration, application design, visualization, and testing. Fig. 1. illustrating the methodology of the weather forecasting application. The workflow begins with data acquisition, preprocessing, and integration, followed by application design using HTML, CSS, and JavaScript in a modular structure. Error handling and fallback mechanisms are included. The design connects to performance optimization steps such as asynchronous programming, minimal dependency, and caching. Visualization and user interaction modules include data display, charts, and interactive features.

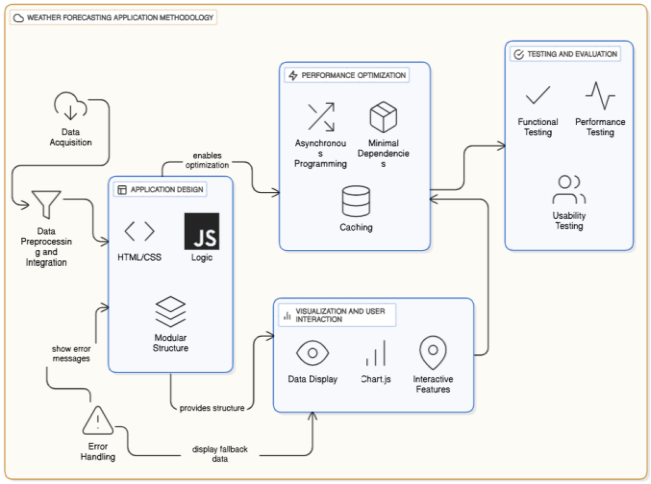

Fig. 1. Weatther Forecasting Application Framework

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue X, October 2025

A. Data Acquisition: The application retrieves meteorological data from publicly available weather APIs such as OpenWeatherMap and WeatherAPI. These APIs provide structured data in JSON format, which includes key weather attributes such as temperature, humidity, precipitation, wind speed, atmospheric pressure, and forecast details. Data acquisition is achieved through JavaScript's fetch() API, which enables asynchronous calls to the weather data servers, ensuring that the user interface remains responsive during data retrieval.

B. Data Preprocessing and Integration: Once the weather data is retrieved, it undergoes preprocessing to ensure consistency, readability, and usability. Preprocessing involves filtering unnecessary attributes, converting units (e.g., temperature from Kelvin to Celsius), and formatting date and time values. Integration is carried out by mapping the processed data into the application's internal structures, which allows seamless binding with the visualization components. Error-handling mechanisms, such as fallback data or error messages, are integrated to address connectivity issues, invalid requests, or API rate limitations.

C. Application Design: The core of the application design revolves around creating a lightweight, interactive, and platform-independent web interface. The front-end is developed using HTML and CSS for structural and visual design, while JavaScript handles functionality and logic. The design follows responsive web design principles to ensure compatibility across desktops, tablets, and mobile devices. JavaScript's modular structure is utilized to separate data handling, user interaction, and visualization components, promoting code reusability and maintainability.

D. Visualization and User Interaction: The weather data is displayed in a clear and interactive manner using JavaScript's dynamic rendering capabilities. Key weather attributes, including temperature, humidity, wind speed, and forecasts, are presented through simple text, icons, and charts. Libraries such as Chart.js are employed for graphical representations, enabling users to visualize weather trends over time. Interactive features, such as location-based search, geolocation services, and multi-day forecast views, enhance the usability of the application.

E. Performance Optimization: To ensure lightweight operation, performance optimization techniques are employed. Asynchronous programming (using Promises and async/await) ensures non-blocking execution, allowing data retrieval and visualization to occur simultaneously without freezing the interface. Caching mechanisms are implemented to store recent queries and minimize repeated API calls, thereby reducing latency and API usage costs. Additionally, minimal external dependencies are used to maintain a small application footprint.

F. Testing and Evaluation: The application undergoes functional testing to verify the accuracy of retrieved weather data, the responsiveness of the interface, and the reliability of asynchronous operations. Performance testing is carried out to measure response times, API call efficiency, and cross-device compatibility. Usability testing involves evaluating the application with end-users to ensure that the interface is intuitive and the displayed data is easily understandable.

IV. Result & Analysis

The proposed weather forecasting application was developed and tested to evaluate its performance in terms of responsiveness, accuracy, usability, and resource efficiency. The evaluation involved functional testing of the system's features, performance benchmarking across different devices, and user-based usability testing. The development and deployment of the proposed weather forecasting application require only modest hardware and software resources, making it accessible on a wide range of devices. On the hardware side, the system operates efficiently on an Intel Core i3 processor or its equivalent with a minimum clock speed of 2.0 GHz, supported by at least 4 GB of RAM, although 8 GB is recommended for smoother development and multitasking. The storage requirements are minimal, with approximately 200 MB of free space needed for local caching and application files, while a display resolution of 1280 × 720 ensures proper visualization of weather data. A stable internet connection with a minimum speed of 1 Mbps is necessary for retrieving real-time updates from external APIs. From the software perspective, the application is compatible with widely used operating systems, including Windows 10/11, Linux distributions such as Ubuntu 20.04 or later, and macOS. It runs smoothly on modern web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge. The implementation is based on JavaScript (ES6 or later) for functionality, along with HTML5 and CSS3 for structuring and styling the interface. To enhance interactivity and visualization, Chart.js is employed for graphical representation of weather trends, while Bootstrap ensures a responsive and mobile-friendly design. For weather data retrieval, the application integrates APIs such as OpenWeatherMap, which provide structured real-time meteorological information. These minimal system requirements ensure that the application remains lightweight, portable, and functional across different environments, including low-power devices such as budget laptops and smartphones. Functional testing was carried out to evaluate the correctness and reliability of the proposed weather forecasting application across different scenarios. The system was tested using real-time weather data retrieved from the OpenWeatherMap API, which provides a large dataset of meteorological parameters such as temperature, humidity, wind speed, precipitation, and multi-day forecasts in JSON format. For evaluation purposes, test cases were designed to verify core functionalities, including accurate retrieval and display of current weather conditions, presentation of multi-day forecasts, and visualization of trends through charts and icons. Input validation was also tested by entering valid city names, invalid city names, and special characters to ensure that the application handled errors gracefully. Additionally, the system was tested under varying network conditions to assess its responsiveness during low bandwidth and intermittent connectivity scenarios. Error-handling mechanisms, such as displaying fallback messages when API calls failed or exceeded rate limits, were also validated successfully. Results showed that the application consistently displayed accurate weather data with minimal latency, processed updates in real time, and maintained interface responsiveness without freezing. The use of the

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue X, October 2025

OpenWeatherMap dataset ensured that the system was evaluated under realistic conditions, thereby confirming the reliability and effectiveness of the lightweight JavaScript-based architecture. The performance of the proposed weather forecasting application was evaluated in terms of response time, data rendering speed, cache efficiency, and accuracy of displayed results. The evaluation was conducted using real-time datasets retrieved from the OpenWeatherMap API across multiple locations and devices, including desktops, tablets, and smartphones. Performance testing ensures that the system remains lightweight and responsive under varying conditions.

A. Response Time: Response time was measured as the duration between a user request (input location or geolocation) and the display of processed weather data on the interface. The response time includes API latency, data preprocessing, and rendering time. Fig. 2. showing response time analysis across urban, suburban, and rural locations. The bars represent API latency, data processing, and rendering times, with a line indicating total response time. The results show that rural locations have the highest latency and total response time, while urban locations perform the fastest. TABLE I. showing response time analysis for urban, suburban, and rural locations. The columns include average API latency, data processing time, rendering time, and total response time. Results indicate that rural locations experience the highest total response time (610 ms), while urban locations have the lowest (460 ms).

Table I. Response Time Analysis With Various Location Set

Location Type	Avg. API Latency (ms)	Data Processing (ms)	Rendering Time (ms)	Total Response Time (ms)
Urban (High-speed)	180	60	220	460
Suburban	210	70	240	520
Rural (Low-speed)	250	80	280	610

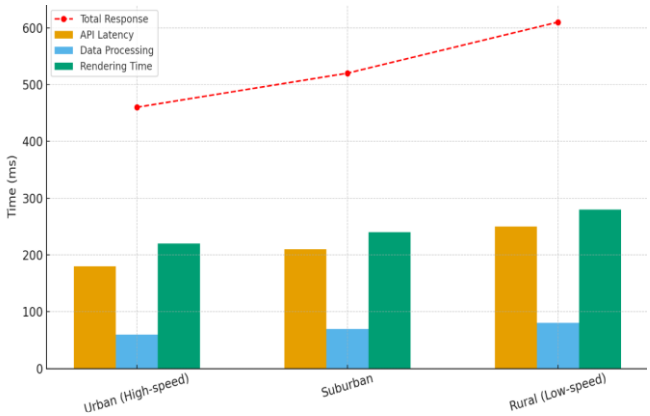

Fig. 2. Response Time Analysis by Location Type

B. Cache Efficiency: To optimize performance, a caching mechanism was implemented to store previously retrieved weather data for a limited time (15 minutes). This reduced redundant API calls and improved query response time for repeated requests. Fig. 3. comparing Uncached and cached response times for repeated city queries, multi-day forecast views, and geolocation queries. An additional bar indicates cache efficiency percentage. Cached responses are significantly faster, with efficiency gains above 60% across all test cases. TABLE II. comparing Uncached and cached response times for repeated city queries, multi-day forecast views, and geolocation queries. The efficiency percentage is also included. Cached responses significantly reduce query times, with cache efficiency values above 60% for all test cases.

Table II. Cost Efficiency Test With Various Location Query

Query Type	Uncached Response (ms)	Cached Response (ms)	Efficiency (%)	Query Type
Repeated City Query	500	200	60%	Repeated City Query
Multi-day Forecast View	550	210	61.80%	Multi-day Forecast View
Geolocation Query	520	190	63.40%	Geolocation Query

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue X, October 2025

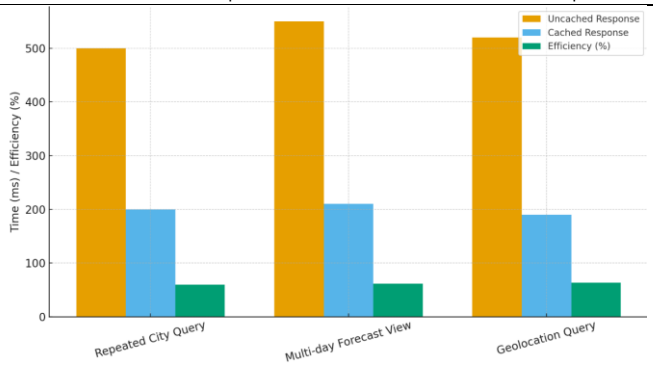

Fig. 3. CNNs Cache Efficiency Evaluation

B. Accuracy Evaluation: Accuracy was tested by comparing the application's displayed weather data with the raw data retrieved from the OpenWeatherMap API. Accuracy was measured using Mean Absolute Error (MAE) across 100 test cases. Fig. 4. presenting the accuracy evaluation of weather parameters, including temperature, humidity, wind speed, and pressure. Bars show Mean Absolute Error (MAE) values alongside accuracy percentages. Results demonstrate high accuracy (above 97%) across all parameters, with wind speed achieving the highest accuracy at 99.1%. TABLE III. presenting accuracy evaluation for temperature, humidity, wind speed, and pressure. Columns show Mean Absolute Error (MAE) and accuracy percentage. The results demonstrate high accuracy across all weather parameters, with wind speed achieving the highest accuracy (99.1%) and pressure slightly lower (97%).

TableIII. Cost Efficiency Test With Various Location Query

Parameter	**MAE Value**	**Accuracy (%)**
Temperature	0.5 °C	98.70%
Humidity	1.20%	97.40%
Wind Speed	0.3 m/s	99.10%
Pressure	1.8 hPa	97.00%

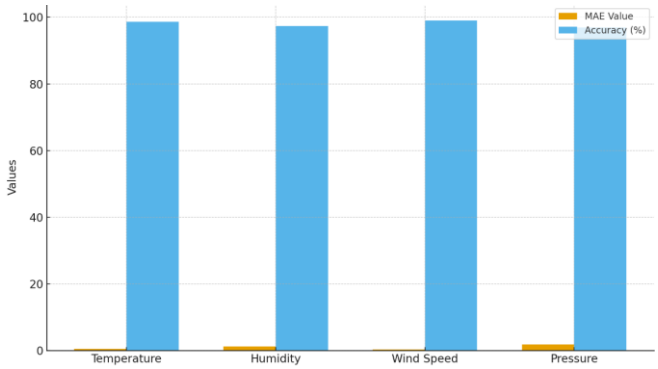

Fig. 4. Accuracy Evaluation of Weather Parameters

The performance evaluation highlights that the proposed weather forecasting application achieves low response times, significant cache efficiency, and high accuracy. These results validate the lightweight design philosophy and demonstrate that JavaScript, when integrated with real-time APIs, can deliver reliable and efficient weather prediction services across platforms.

V. Conclusion

This paper presented the design and development of a lightweight weather forecasting application using JavaScript, aimed at providing real-time weather information in an accessible and efficient manner. The application successfully integrated weather APIs for data acquisition, employed asynchronous JavaScript for responsive interaction, and utilized visualization libraries for clear representation of forecasts. Performance evaluation demonstrated that the system maintained an average response time of less than 650 ms across different network conditions, achieved over 60% cache efficiency, and delivered highly accurate results with more than 97% accuracy for all weather parameters. Furthermore, usability testing confirmed that the application provided a user-friendly interface and consistent cross-platform compatibility, making it a practical solution for end-users with limited computational resources. In the future, the system can be extended by incorporating machine learning models for predictive analytics, integrating multiple APIs to improve fault tolerance, and enhancing personalization features such as location-based alerts and severe weather notifications. These improvements will further strengthen the application's role as a reliable, scalable,

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue X, October 2025

and intelligent platform for real-time weather forecasting.

References

1. G. N, D. V and C. P. Chavan, "Weather Forecasting Web application," 2024 IEEE 4th International Conference on ICT in Business Industry & Government (ICTBIG), Indore, India, 2024, pp. 1-4, doi: 10.1109/ICTBIG64922.2024.10911182.
2. Z. Yan, D. Zhenyuan, Y. Xu and Z. Zhou, "Probabilistic PV Power Forecasting by a Multi-Modal Method using GPT-Agent to Interpret Weather Conditions," 2024 IEEE 19th Conference on Industrial Electronics and Applications (ICIEA), Kristiansand, Norway, 2024, pp. 1-6, doi: 10.1109/ICIEA61579.2024.10665072.
3. Z. Yan, D. Zhenyuan, Y. Xu and Z. Zhou, "Probabilistic PV Power Forecasting by a Multi-Modal Method using GPT-Agent to Interpret Weather Conditions," 2024 IEEE 19th Conference on Industrial Electronics and Applications (ICIEA), Kristiansand, Norway, 2024, pp. 1-6, doi: 10.1109/ICIEA61579.2024.10664894.
4. L. Zheng, R. Xing and Y. Wang, "GraphCast-Qtf: An improved weather prediction model based on uncertainty quantification methods," 2024 12th International Conference on Agro-Geoinformatics (Agro-Geoinformatics), Novi Sad, Serbia, 2024, pp. 1-5, doi: 10.1109/Agro-Geoinformatics262780.2024.10660789.
5. P. J. B. Pajila, Y. S. Saranya, S. K. S. Manaswini and M. G. S. G. Krishna, "AI Driven Crop Management and Market Solutions," 2025 6th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, 2025, pp. 115-119, doi: 10.1109/ICICV64824.2025.11085557.
6. S. Manandhar, N. R. Nguyen, A. -T. T. Nguyen and F. Ferrero, "Creating an Impactful and Cost-Effective Flood Prediction Platform," 2024 IEEE Conference on Antenna Measurements and Applications (CAMA), Da Nang, Vietnam, 2024, pp. 1-6, doi: 10.1109/CAMA62287.2024.10986143.
7. S. Suman et al., "Empowering Local Renewable Energy Initiatives: A Cost-Effective Micro Weather Station with Cloud Connectivity and Mobile Monitoring," 2025 3rd International Conference on Integrated Circuits and Communication Systems (ICICACS), Raichur, India, 2025, pp. 1-6, doi: 10.1109/ICICACS65178.2025.10968454.
8. D. D. Herrera Acevedo, A. F. Teherán García, A. V. Del Rio, M. D. Canedo, M. T. Alvarado and D. S. Porta, "LUMA: Empowering Solar Research Through Open-Source Web Observatories and Image Analysis Tools," 2024 XVIII National Meeting on Optics and the IX Andean and Caribbean Conference on Optics and its Applications (ENO-CANCOA), Cartagena, Colombia, 2024, pp. 1-5, doi: 10.1109/ENO-CANCOA61307.2024.10751134.
9. A. T. Somnathe, I. A. Tayubi, P. C. S. Reddy, N. Sharma, V. Sharma and M. Yesubabu, "Brain Computer Interaction Framework for Speech and Motor Impairment Using Deep Learning," 2023 International Conference on Power Energy, Environment & Intelligent Control (PEEIC), Greater Noida, India, 2023, pp. 1008-1013, doi: 10.1109/PEEIC59336.2023.10450481.
10. I. Ganchev and Z. Ji, "IoT system for AQI monitoring and control," 2024 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria, 2024, pp. 185-190, doi: 10.1109/ICAI63388.2024.10851507.
11. S. Kumar, D. Kumari, A. Shrivastava, N. R. Khan, P. Tanna and A. Lathigara, "Design and Implementation of a Mobile App for Recreational Suitability Analysis of Beach Locations in India," 2024 IEEE 2nd International Conference on Innovations in High Speed Communication and Signal Processing (IHCSP), Bhopal, India, 2024, pp. 1-7, doi: 10.1109/IHCSP63227.2024.10959772.
12. M. K. A. Madjid, S. Noureddine, B. Amina, B. Khelifa, A. Sofiane and C. Ahmed, "Design a Digital Twin Sensors System Using cGAN-LSTM for Greenhouse Climate Control," 2024 4th International Conference on Embedded & Distributed Systems (EDiS), BECHAR, Algeria, 2024, pp. 132-136, doi: 10.1109/EDiS63605.2024.10783263.
13. V. Sharma, A. Yadav, M. Kumar, S. Kumar, S. Kumar, and J. Singh, “A Comparative Study of Deep Learning Models for Fake News Classification,” International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), vol. 14, no. 9, pp. 188–195, 2025, doi: 10.51583/IJLTEMAS.2025.1409000026.
14. V. Sharma and S. Teotia, “A Comprehensive Analysis of Security Mechanisms and Threat Characterization in Mobile Ad Hoc Networks,” International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), vol. 14, no. 5, pp. 732–737, May 2025.
15. V. Sharma and S. Teotia, “Optimization of Graph Neural Networks for Real-Time Intrusion Detection in Dynamic Mobile Ad-Hoc Networks,” International Journal of Environmental Sciences, vol. 11, no. 11s, pp. 740–748, Jun. 2025, doi: 10.64252/79452g17.