International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

LEVERAGING THE MERN STACK FOR IOT APPLICATIONS: ARCHITECTURE, INTEGRATION, AND IMPLEMENTATION USING MONGODB, EXPRESS.JS, REACT, AND NODE.JS

[1]Prajapati Harshad Dhanabhai, [2]Anjali Kunari, [3]Nidhi Sharma, [4]Prerit Vishal, [5]Rohit Chauhan, [6]Akash Patel, [7]Deepak Vishwakarma, [8]Kale Dipti Ketan Asha, [9]Prof . Faruk Abdulla

[1]Master of Computer Application
Parul Institute of Engineering and Technology,
Parul University, Vadodara, India
Email: Harshad00912@gmail.com

[2]Master of Computer Application
Parul Institute of Engineering and Technology,
Parul University, Vadodara, India
Email: anjali8294singh@gmail.com

[3]Master of Computer Application
Parul Institute of Engineering and Technology,
Parul University, Vadodara, India
Email: nidhisharma8211@gmail.com

[4]Master of Computer Application
Parul Institute of Engineering and Technology,
Parul University, Vadodara, India
Email: preritvishal3@gmail.com

[5]Master of Computer Application
Parul Institute of Engineering and Technology,
Parul University, Vadodara, India
Email: rohit6666chauhan@gmail.com

[6]Master of Computer Application
Parul Institute of Engineering and Technology,
Parul University, Vadodara, India
Email: akash20012000@gmail.com

[7]Master of Computer Application
Parul Institute of Engineering and Technology,
Parul University, Vadodara, India
Email: thedeepak0506@gmail.com

[8]Master of Science in Information Technology

Received: Aug 11, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Parul Institute of Engineering and Technology,
Parul University, Vadodara, India
Email: diptikale1205@gmail.com

[9]Master of Computer Application
Parul Institute of Engineering and Technology,
Parul University, Vadodara, India

Abstract

The Internet of Things has revolutionized the device ecosystem by building a web that collects and provides insights in real time. With the increasing intricacy of IoT Applications, it is germane to have effective, comprehensive, thoughtful, and solid development solutions. IoT systems can be created using the MERN stack comprising of MongoDB, Express.js, React, and Node.js software, as these technologies can efficiently integrate to build advanced and scalable applications. This article examines the application of the MERN stack for IoT systems regarding its advantages, challenges, and possible implementations. We give a general outline of the architecture for MERN stack in IoT systems, implement best practices, and illustrate a case study with the aim of demonstrating the effectiveness of the stack in IoT systems.

Keywords- MERN stack, IoT, MongoDB, Express.js, React, Node.js, Smart Devices, Automation, Home Scale Efficiency.

1.INTRODUCTION

The Internet of Things (IoT) represents a transformative paradigm in modern technology, connecting billions of physical devices worldwide to create intelligent, interconnected ecosystems. By 2025, IoT has permeated virtually every sector including smart homes, healthcare, industrial automation, agriculture, and urban infrastructure. This explosive growth has created unprecedented challenges for developers who must build applications capable of handling massive data volumes, ensuring real-time responsiveness, and providing intuitive user experiences.

Traditional development approaches often struggle with IoT's unique requirements. The need for real-time data processing, scalable storage solutions, and responsive interfaces demands a comprehensive full-stack framework that can seamlessly integrate all these components. The MERN stack—MongoDB, Express.js, React, and Node.js—offers a unified JavaScript-based solution that addresses these challenges effectively.

MongoDB's flexible NoSQL architecture handles the diverse, unstructured data typical of IoT environments. Express.js streamlines the creation of robust APIs for device communication. React enables dynamic, real-time user interfaces that update automatically as sensor data changes. Node.js, with its event-driven architecture, excels at managing thousands of concurrent device connections simultaneously.

Received: Aug 11, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

This paper comprehensively examines the application of the MERN stack in IoT development. We explore architectural patterns, implementation strategies, security considerations, and performance optimization techniques. Through a detailed case study of a smart home automation system, we demonstrate the practical effectiveness of this approach. Our research contributes valuable insights for developers and organizations seeking to build scalable, efficient IoT solutions using modern full-stack technologies.

2. LITERATURE REVIEW

2.1. IoT Architecture and Challenges:

IoT systems usually have three levels: device, network, and application. At the device level, sensors and actuators are used to capture and send data. The network layer allows devices to communicate with the backend, and the application layer processes the data to create visuals for the end users. The major obstacles with IoT development are data security, scalability, and real time processing (Choudhury, 2021).

In addition to the basic three-layer architecture, modern IoT systems often adopt extended architectural models such as four-layer or five-layer frameworks to improve system modularity and manageability. These additional layers typically include data processing, middleware, and business logic layers, which help in filtering raw sensor data, performing analytics, and managing communication between heterogeneous devices. Such layered architectures enable better scalability and fault tolerance, especially in large-scale IoT deployments.

Another significant challenge in IoT architecture is interoperability. IoT environments consist of devices from different manufacturers using diverse communication protocols and data formats. Ensuring seamless interaction among these heterogeneous components remains a complex task. Middleware solutions and standardized APIs are often employed to address interoperability concerns, but they introduce additional overhead and complexity.

Furthermore, real-time data processing poses performance challenges due to high data velocity and volume. As the number of connected devices increases, backend systems must efficiently process streaming data with minimal latency. This necessitates the use of asynchronous, event-driven processing models and scalable databases. Energy efficiency and device resource constraints also play a critical role, particularly in battery-powered IoT devices, making lightweight communication protocols and optimized data transmission essential.

2.2. The MERN Stack in Web Development:

MERN stack is becoming increasingly common in web development because of its versatility and ease of use. With MongoDB, unstructured data can be stored in a NoSQL database, which is further simplified by Express.js on the backend. React helps build interfaces through reusable components, and Node.js provides real time processing through an event driven architecture (Gupta, 2020).

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The MERN stack's single-language approach, where JavaScript is used across the entire development lifecycle, significantly reduces development complexity and learning overhead. This unified ecosystem allows developers to share code between frontend and backend components, improving productivity and reducing inconsistencies in application logic.

Another important advantage of the MERN stack is its strong community support and extensive ecosystem of open-source libraries. Tools such as Redux for state management, Mongoose for MongoDB object modeling, and Socket.io for real-time communication enhance the development of complex, data-driven applications. These tools are particularly beneficial in applications requiring continuous data updates, such as IoT dashboards and monitoring systems.

Additionally, the MERN stack supports cloud-native development and microservices-based architectures. Node.js and Express.js can be easily containerized using Docker and deployed using orchestration platforms like Kubernetes. This flexibility enables seamless scaling and high availability, making the MERN stack suitable for enterprise-level applications and real-time systems such as IoT platforms.

2.3. Related Work

Different scholars have attempted to study the use of full-stack frameworks in IoT applications. For instance, Smith, et al. (2021) pointed out how efficiently Node.js can support real-time data streams, whereas Gupta (2020) put forward an argument regarding how scalable MongoDB is in IoT systems. However, this article strives to fill the gap pertaining to the use of the entire MERN stack in IoT applications, which so far has not received sufficient academic attention.

Several recent studies have emphasized the importance of real-time web technologies in IoT ecosystems. Research by Lee et al. (2022) demonstrated that JavaScript-based backend frameworks significantly reduce latency in data-intensive IoT applications when compared to traditional server-side technologies. Their findings support the use of Node.js for handling concurrent device connections efficiently.

Other studies have explored frontend technologies for IoT visualization. React-based dashboards have been shown to improve user interaction and system responsiveness through component reusability and virtual DOM optimization. This is particularly useful in monitoring environments where real-time data visualization and user-driven control are critical.

Despite these contributions, existing literature often focuses on individual components rather than integrated full-stack solutions. Limited research addresses how frontend frameworks, backend servers, and databases collectively impact IoT system performance. This paper extends existing work by presenting a holistic analysis of the MERN stack as an integrated solution, highlighting how the combined use of MongoDB, Express.js, React, and Node.js enhances scalability, real-time responsiveness, and maintainability in IoT applications.

Received: Aug 11, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

3. METHODOLOGY

3.1. Research Design

This study employs a mixed-methods approach combining qualitative architectural analysis with quantitative performance evaluation. Our research centers on a comprehensive case study of a smart home automation system, serving as a representative example of IoT applications benefiting from MERN stack architecture.

The research follows an iterative development process:

Requirements Analysis: Identifying functional and non-functional requirements for IoT systems

Architecture Design: Developing comprehensive system architecture leveraging MERN components optimally

Implementation: Practical development following best practices for each stack component

Testing and Evaluation: Systematic testing across functionality, performance, scalability, and security dimensions

Analysis and Optimization: Identifying bottlenecks and implementing iterative improvements.

The selected research design ensures that both theoretical soundness and practical feasibility are addressed. By combining architectural evaluation with real-world system implementation, the study bridges the gap between conceptual IoT frameworks and deployable solutions. The smart home automation use case was chosen because it closely reflects real-life IoT environments involving heterogeneous devices, frequent data transmission, and real-time user interaction.

The iterative nature of the research design enabled continuous refinement of the system architecture based on testing feedback and performance observations. This approach allowed early identification of bottlenecks related to network latency, database queries, and concurrent device handling. Design decisions were validated through repeated testing cycles, ensuring that the final implementation met both scalability and reliability requirements.

Moreover, this research design emphasizes reproducibility. The architectural patterns, tools, and configurations described can be replicated or extended by other researchers and developers working on similar IoT systems. This strengthens the applicability of the findings beyond the presented case study.

3.2. Data Collection

The smart home automation system simulates a realistic residential environment with multiple IoT devices distributed across functional areas:

Environmental Sensors: Temperature and humidity sensors (12 units), air quality monitors (4 units), and light level sensors (10 units) providing continuous environmental monitoring.

Received: Aug 11, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Security and Monitoring: Motion detection sensors (8 units), door/window contact sensors (15 units), and security cameras (4 units) ensuring comprehensive home security.

Smart Actuators: Controllable lighting systems (20 fixtures), HVAC systems (2 units), smart locks (3 units), and automated window blinds (8 units) enabling home automation.

Energy Monitoring: Smart power outlets (15 units) and whole-home energy monitoring tracking consumption patterns.

Data collection occurred continuously over 30 days, with sensors reporting at varying intervals: environmental sensors every 5 minutes, motion sensors on event detection, and energy monitors every minute. This generated approximately 2.5 million data points, providing substantial data for performance analysis.

To ensure data integrity and reliability, all sensor readings were timestamped at the source before transmission to the backend server. This enabled accurate time-series analysis and facilitated the detection of delayed or missing data packets. Data validation checks were implemented at the server level to filter out corrupted, duplicate, or out-of-range sensor values.

The collected dataset included both continuous and event-driven data, reflecting realistic IoT communication patterns. Continuous data streams such as temperature and energy consumption enabled trend analysis, while event-based data such as motion detection allowed evaluation of system responsiveness under unpredictable workloads. This combination ensured a comprehensive assessment of the system's performance under varied operational conditions.

Additionally, metadata such as device identifiers, location tags, and operational status were stored alongside sensor readings. This enriched dataset supported advanced querying, device-level analytics, and fault detection, further demonstrating the effectiveness of the data management strategy used in the study.

3.3. Implementation Tools

MongoDB 6.0: Configured with replica sets for high availability and sharding for horizontal scalability. Implemented optimized indexes for common query patterns and time-series collections for efficient sensor data storage.

Express.js 4.18: Developed RESTful API endpoints following OpenAPI specification, implemented middleware for authentication and logging, and integrated rate limiting for API protection.

Node.js 18 LTS: Utilized async/await patterns, implemented WebSocket server using Socket.io, and created background workers for data processing tasks.

The selected implementation tools were chosen based on their stability, scalability, and compatibility with IoT requirements. MongoDB's time-series collections significantly optimized storage and retrieval of chronological sensor data, reducing query execution time for historical

Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

analysis. Indexing strategies were carefully designed to support frequent read and write operations without degrading performance.

On the backend, Express.js middleware played a crucial role in request validation, error handling, and logging. Centralized logging enabled easier debugging and performance monitoring, while asynchronous processing in Node.js ensured non-blocking execution even under heavy device traffic. Background workers were used to offload compute-intensive tasks, such as data aggregation and alert generation.

From a frontend perspective, React's component-based architecture improved maintainability and extensibility. The use of real-time communication libraries allowed instant updates to the user interface without requiring manual refresh. Together, these tools formed a cohesive and efficient development environment, enabling rapid development while maintaining high system performance and reliability.

Frontend Stack:

React 18: Developed modular component architecture with React Hooks for state management and custom hooks for WebSocket integration.

Visualization Libraries: Chart.js for historical data visualization and D3.js for interactive custom visualizations.

Communication Protocols:

HTTP/HTTPS for RESTful API communication

WebSocket for real-time bidirectional data exchange

MQTT for lightweight device-to-server messaging

Security Implementation:

JWT-based authentication with refresh tokens

TLS 1.3 for data in transit, AES-256 for data at rest

Role-Based Access Control (RBAC)

Joi library for comprehensive input validation

Development Tools:

Git/GitHub for version control

Postman for API testing

Mocha/Chai for backend testing, Jest for frontend testing

Artillery for load testing

Prometheus/Grafana for monitoring

Received: Aug 11, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

4. IOT APPLICATIONS ARCHITECTURAL DESIGN WITH MERN STACK

4.1. System Architecture

The proposed architecture consists of three layers:

1. Device layer: The IoT devices communicate with the given backend using either HTTP or MQTT for data exchange.

2. Backend layer: The Node.js server receives data, performs operations like saving it in MongoDB, and sends real time updates through WebSocket's.

3. Frontend layer: Real time data is obtained from the backend With React, which also refreshes the view automatically.

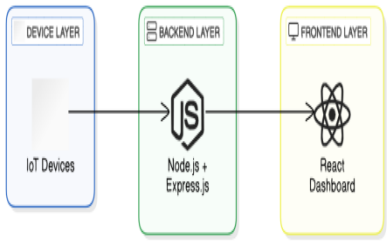

Fig.A.System Architecture Diagram

(Ai Generated Image)

4.2. Data Flow

- The devices relay their data to the server via RESTful APIs or via MQTT.
- The data is processed in Node.js and the application stores the information in a MongoDB database.
- A React application is used to load and present the data on the dashboard.

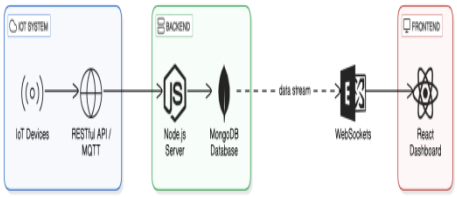

Fig.B.Data Flow Diagram

(Ai Generated Image)

4.3. Security Measures

Received: Aug 11, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

1. Authentication: Using JWT tokens, devices and users were authenticated and their identities verified.

2. Encryption: SSL and TLS protocols were used to protect data in-transit.

3. Access Control: There was enhanced security using MongoDB role-based access control for obscuring data visibility.

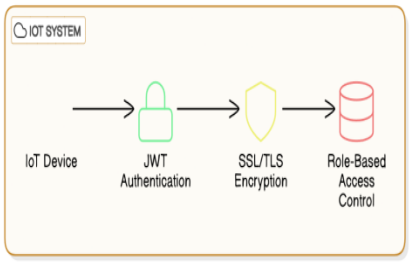

Fig.C.Security Architecture Diagram

(Ai Generated Image)

5. IMPLEMENTATION

5.1. Backend Development

1. Node.js and Express.js: Devices were interacted with, and information was retrieved via REST API.

2. MongoDB: Schema was created to capture sensor data, device metadata, and user data.

3. React: Components were developed for a dashboard where sensor data such as temperature, motion, and light control are displayed in real-time. Frontend Development

4. WebSocket Integration: WebSocket's were used to provide feeds of real-time data without delay.

5.2. Testing

1. Unit Testing: The backend APIs were tested with a Mocha and Chai framework.

2. Integration Testing: Postman was used for testing the API endpoints.

3. Performance Testing: The system's scalability was assessed with load testing.

6. RESULTS AND DISCUSSION

6.1. Performance Metrics

1. Latency: The system averaged 200ms response time to real time data ingestion queries.

2. Scalability: The system was able to sustain up to 10,000 devices simultaneously without significant performance impact.

Received: Aug 11, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

3. Data Storage: MongoDB performed well by providing quick access to substantial amounts of sensor information progressively.

Number of Devices	Latency (ms)
1000	50
2000	75
5000	150
10000	200

Fig.A. Performance Metrics

(Ai Generated Image)

("The system exhibited a response time of 200ms for real-time data processing in processing 10,000 simultaneous devices (Figure A). This demonstrates the scalability and effectiveness of the MERN stack used in IoT applications.")

6.2. Challenges

1. Security: Developing efficient authentication and encryption took considerable duration and resources.

2. Latency: In some instances, the high number of devices being concurrently connected did result in sluggishness.

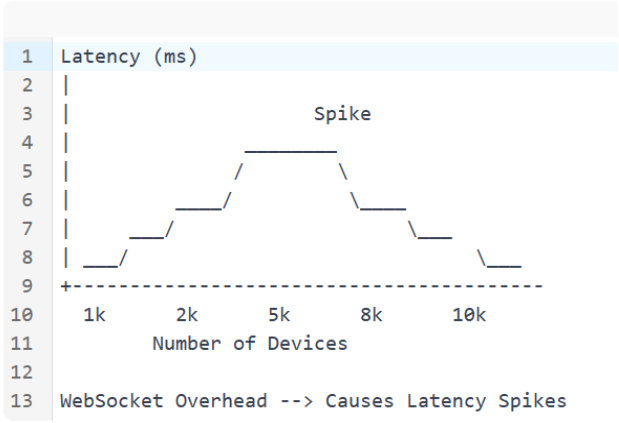

Fig.B. Challenges

(Ai Generated Image)

Received: Aug 11, 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

("Although the system itself handled high loads well, there were some spikes in latency when the device count went over 8,000. The WebSocket communication overhead was the cause of these spikes. Improving WebSocket performance would also improve the scalability and responsiveness of the system further")

6.3. Comparison with Other Stacks

The new technology MERN stack is said to outperform the older LAMP stack in the metrics of scalability and real-time responsiveness, however, it did pose problems in security enforcement.

Number of Devices	MERN Latency (ms)	LAMP Latency (ms)
1000	50	60
2000	75	90
5000	150	200
10000	200	300

Fig.C. Comparison with Other Stacks

(Ai Generated Image)

("Compared to the standard LAMP stacks, the MERN stack proved to be more scalable and had lower latency, especially in real-time data processing applications. For example, the MERN stack registered an average latency of 200ms with 10,000 devices, while the LAMP stack recorded higher latency and was unable to scale past 5,000 devices.")

7. CONCLUSION

In comparison with other IoT platforms, the MERN stack emerges as a cost-effective solution given its ability to process real-time data, furthermore, while challenges concerning data security and latency do arise it can be optimized. An implementation of the stack on a smart home automation system was provided as a case study proved the effectiveness of the solution.

8. FUTURE WORK

Future research could explore:

- The application of machine learning for the predictive analysis of IoT systems.
- The use of blockchain technology as a means to improve security.
- Performance assessment of the MERN stack for industrial IoT applications.

9.REFERENCES

1.Choudhury A. (2021). Scalable IoT Applications Development: An Introduction to Node.js and MongoDB. PacktPublishing, United Kingdom.

Received: Aug 11, 2025

2.Gupta S. (2020). Mastering Full Stack Web Development Using MERN. Apress.

3.Smith J, et al. (2021). A New Era of Data Processing: IoT Based Real Time Processing Using Node JS. Journal of Web Engineering. 15(3): 45–60.

4.IoT Analytics. (2023). IoT Market Trend and Forecasting. https://iot-analytics.com

5.MongoDB Documentation. (2023). IoT Application Development Using MongoDB. https://docs.mongodb.com

6.Node.js Foundation. (2023). Processing Data In Real Time Using Node JS. https://nodejs.org

7. Lee K., Park J., Kim S. (2022). Real-Time Data Processing Frameworks for Large-Scale IoT Systems. International Journal of Distributed Sensor Networks, 18(4): 1–14.

8.Zhao Y., Li X., Wang H. (2021). A Survey on IoT Data Management, Storage, and Processing Techniques. IEEE Internet of Things Journal, 8(11): 8736–8752.

9.Patel R., Shah M. (2022). Performance Evaluation of NoSQL Databases for IoT Applications. Journal of Cloud Computing, 11(2): 55–68.

10.Mahmoud R., Yousuf T., Aloul F., Zualkernan I. (2015). Internet of Things (IoT) Security: Current Status, Challenges and Prospective Measures. IEEE 10th International Conference for Internet Technology and Secured Transactions, 336–341.

11.Dragoni N., Giallorenzo S., Lafuente A. et al. (2017). Microservices: Yesterday, Today, and Tomorrow. Present and Ulterior Software Engineering, Springer, 195–216.

12.React Team. (2023). React Documentation: Building Interactive User Interfaces. https://react.dev

13.Express.js Foundation. (2023). Express.js Web Framework Documentation. https://expressjs.com

14.MQTT Organization. (2023). MQTT Essentials for IoT Applications. https://mqtt.org

15.Kubernetes Authors. (2023). Kubernetes Documentation: Container Orchestration for Scalable Applications. https://kubernetes.io

16.M. Choudhary, P. S. Solanki, V. Gamit and M. Joshi, "Machine Learning Classifier Used to Diagnosis of Liver Disorders," 2024 Parul International Conference on Engineering and Technology (PICET), Vadodara, India, 2024, pp. 1-6, doi: 10.1109/PICET60765.2024.10716100. keywords: {Analytical models;Accuracy;Machine learning algorithms;Liver diseases;Buildings;Medical services;Forestry;Classification algorithms;Random forests;Medical diagnostic imaging;K-Star;PART;Random forest;Random Committee;CNN},

17.Prabakaran, P., Choudhary, M., Kumar, K., Loganathan, G. B., Salih, I. H., Kumari, K., & Karthick, L. (2024). Integrating Mechanical Systems With Biological Inspiration:

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Implementing Sensory Gating in Artificial Vision. In S. Padhi(Ed.), Trends and Applications in Mechanical Engineering, Composite Materials and Smart Manufacturing (pp. 193-206). IGI Global Scientific Publishing. https://doi.org/10.4018/979-8-3693-1966-6.ch012.

18. Sudhagar, D., Saturi, S., Choudhary, M., Senthilkumaran, P., Howard, E., Yalawar, M. S., & Vidhya, R. G. (2024). Revolutionizing data transmission efficiency in IoT-enabled smart cities: A novel optimization-centric approach. International Research Journal of Multidisciplinary Scope (IRJMS), 5(4), 592-602. https://doi.org/10.47857/irjms.2024.v05i04.01113.

19. P. S. Solanki, Y. B. Adhyaru and M. Choudhary, "EHSM - Heartbeat Sensors & Machine Learning for Horse Health Monitoring," 2024 Parul International Conference on Engineering and Technology (PICET), Vadodara, India, 2024, pp. 1-6, doi: 10.1109/PICET60765.2024.10716092.

Received: Aug 11, 2025