Applying CQRS and Event-Driven Patterns in Large-Scale Healthcare Applications

Author Name: Deepak Singh
Affiliation: Gainwell Technologies, USA
Role: Advisory Solution Architect
Email: deepaksingh1981@gmail.com

Abstract: ***The study looks into using Command Query Responsibility Segregation (CQRS) and Event-Driven Architecture (EDA) together in big healthcare systems. It is important to determine if using modern architectural designs improves scalability, immediate responsiveness and flexibility when working with large amounts of data. Secondary data gathered using qualitative and quantitative methods, and an explanatory research design, which involved studying cases and analysing graphs, is used. It was observed that CQRS and EDA allow for building modular systems, make different systems work together and make decisions more quickly with up-to-date data. In addition, case studies point out how healthcare has benefited and how the system has become more reliable. Even though there are many benefits, there are still problems like the tougher design process and combining with older technology. The research supports adopting plans in stages, strengthening skills and putting money into monitoring devices to make adaptation simpler. All in all, the analysis shows that these architectural approaches improve the flexibility, fortitude and preparedness of healthcare systems.***

Key terms: ***CQRS, Event-Driven Architecture, healthcare systems, software architecture, system scalability, data interoperability, real-time processing, microservices, event sourcing, system reliability, healthcare informatics.***

I. INTRODUCTION

A. Background to the Study

With the increase in digital records, use of wearable devices and the rise of telehealth, healthcare data is growing at a rapid rate. Traditional systems that are all in one piece usually become stressed as the system grows in importance for reliability, scalability and performance. As a result, developers have started to use Command Query Responsibility Segregation (CQRS) and Event-Driven Architecture (EDA) to solve these issues. They let different processes work independently, allow quick responses to changing situations and boost reliability, which is very useful for large healthcare projects today.

B. Overview

CQRS means the command and query responsibilities are developed separately, which helps systems manage each area on its own. With Event-Driven Architecture, systems using CQRS can use events to communicate with each other, encouraging loose coupling and asynchronous methods. Combining these patterns ensures the creation of applications that can increase in size, work flexibly and remain highly available in the healthcare area [1]. It investigates the way CQRS and EDA combine in healthcare to support managing patient records, offering advice to doctors and overseeing hospital activities by handling large amounts of data and numerous users at once.

C. Problem Statement

Doi:10.48047/ijiee.2024.14.4.9

Most healthcare organisations still depend on old-style frameworks that are tough to upgrade and adapt as needs change. Such systems usually produce results too late, lead to delays in key medical decisions and are not flexible enough for quick advances [2]. As the demand for integrated healthcare services increases and rules get stricter, it is necessary to use architectural styles that can support flexible, scalable and straightforward projects. It studies how CQRS and EDA deal with these challenges by providing a new method for building and running healthcare software systems.

D. Objectives

The objectives of this study include: 1) To analyse how CQRS and Event-Driven Architecture overcome problems with scalability, maintainability and performance in healthcare applications. 2) To explore examples in which these techniques have been successfully used in hospitals and medical centres. 3) To give guidance and suggestions for using CQRS and EDA in future projects related to healthcare software.

E. Scope and Significance

This study centres on major healthcare applications such as Electronic Health Records (EHR), hospital information systems and platforms used for managing patients. It highlights the ways CQRS and EDA, when used, reduce costs and errors while making systems available, consistent and able to work together [3]. The main emphasis is on backend software architecture, but issues related to development, analysing data and connecting systems are considered as well. This study is significant because it can help guide system designers and health IT specialists toward solutions that boost care quality, help patients and streamline healthcare duties.

II. LITERATURE REVIEW

A. Effectiveness of CQRS and Event-Driven Architecture in Enhancing Scalability and Performance in Healthcare Applications

Combining CQRS and EDA has been very helpful in making healthcare applications scalable, maintainable and fast. Using CQRS, read and write activities can be managed and grown independently. Such separation is most useful in healthcare because there are many more read operations than write operations, like updating patient records ***[Refer to figure 1].*** Splitting these operations up improves how these systems handle lots of requests at once, making the whole system more responsive and efficient [4].

EDA allows services using CQRS to communicate through events asynchronously. Real-time processing and quick response become essential in healthcare because needed to quickly send alerts and monitor patient details [5]. Event-driven systems are capable of telling healthcare providers right away if a patient's condition changes, allowing quick responses to the situation.

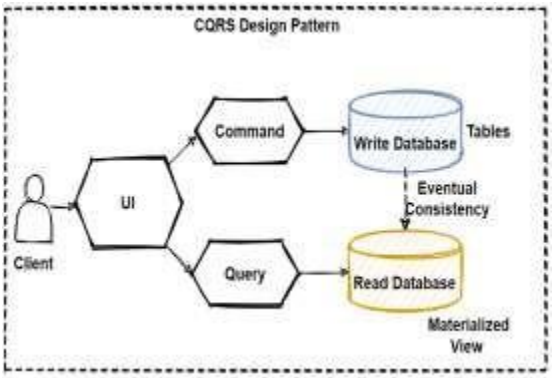

Figure 1: CQRS pattern

Source: [16]

Also, implementing CQRS and EDA leads to a more modular design, which helps with maintenance. Each piece of the system can be created, turned on and enlarged as needed, which lowers the complexity of a monolithic

Doi:10.48047/ijiee.2024.14.4.9

system. Being modular makes development simpler and allows updates and maintenance to be done swiftly, which is key in the fast-developing healthcare field [5]. As a result, using CQRS and EDA together in healthcare systems deals with significant challenges by increasing scalability, improving speed and lowering maintenance efforts for more effective and superior healthcare applications.

B. Real-World Implementations of CQRS and Event-Driven Patterns in Healthcare Systems

Using CQRS and EDA in healthcare systems has led to much better performance, scalability and the ability to maintain these systems. The use of CQRS in healthcare has greatly improved application performance. Using CQRS, some systems have shown read speeds of 42,000 operations per second and write speeds of 8,000 transactions per second, keeping write consistency at 99.95% [6]. The result is that read operations are now 84% more efficient, and write conflicts have decreased by 71% [6]. The use of both CQRS and EDA allows the database to handle around 6 million fewer events daily and improve response times by 73%, which helps the system remain available 99.97% of the time [6].

CQRS and EDA are designed to help independently scale different system components. Additionally, this has resulted in expenses being cut by 52% for infrastructure and 64% for operations. Also, adopting these patterns has made it possible to trace the code 76% faster and spend 65% less time on data reconciliation, bringing about better maintainability [6].

In urgent healthcare situations, EDA makes sure data is processed and delivered immediately, for example, in monitoring patients. As a result, critical alerts are now responded to in about 300 milliseconds instead of 1.2 seconds, helping doctors intervene faster [7]. Practical examples show CQRS and EDA improve healthcare systems' performance, responsiveness and scalability, helping both patients and the practice be more efficient and effective.

C. Challenges and Limitations in Implementing CQRS and Event-Driven Patterns in Healthcare Applications

Though CQRS and EDA have impressive benefits for scalability and performance, they introduce many obstacles for healthcare use. Using CQRS and EDA in a system leads to a more complicated architecture. Managing each read and write process independently needs extra infrastructure and coordination. Because of this complexity, it often takes longer to create the app and to maintain it. Companies using CQRS state that time-to-development increased by 30%, simply because separate models and the logic for keeping them in sync are required [8].

CQRS divides the data into read and write models, and this separation can result in problems with consistency where the read data lags behind the write data. When instant data accuracy matters in healthcare, these delays may cause problems. [9] When systems face many transactions, it can take up to several seconds for data written to be available for reading, which might influence clinical actions.

Most healthcare organisations use legacy systems that do not work with CQRS or EDA. Combining these patterns may require refactoring or making adapters, which can be very time-consuming and resource-consuming. The case study found that connecting CQRS to an existing healthcare system was about 40% more difficult than simply starting with a brand new system [8].

Doi:10.48047/ijiee.2024.14.4.9

Monitoring and debugging in an event-driven system may be difficult because events happen independently from each other. It is necessary to use advanced tools and logging to watch for patterns in flows between services. If the system is not regularly monitored, it can take extra time to understand the source of any issues, and this can lead to important healthcare services being postponed.

III. METHODOLOGY

A. Research Design

The study relies on explanatory research to uncover how CQRS and EDA shape healthcare systems with significant numbers of users. The goal is to describe how and why these architectural designs improve a system's ability to perform well, increase in size and be maintained, while stating what challenges can occur. The research looks at practical examples to find out how different architectural choices affect the way a system performs. It is useful for explaining how various parts of a system work and communicate, which makes it perfect for reviewing systems like those used in healthcare IT.

B. Data Collection

The study makes use of secondary data collected through both qualitative and quantitative methods. In qualitative data analysis, case studies of health organisations and technology companies that have implemented CQRS and EDA are used to uncover common trends and ideas. Examples of sources are scholarly journals, industry bulletins and technology briefs. Quantitative data is taken from research studies that present metrics, charts and graphs such as throughput, latency and error rates for the system, both before and after the new approach was applied. They help in reviewing trends and make it possible to compare different data.

C. Case Studies/Examples

Case Study 1: SafetyCulture's Implementation of CQRS and Event Sourcing

The Australian technology firm SafetyCulture started using CQRS and Event Sourcing to improve the way their systems could scale and be supported. Separating commands from queries makes it easier for them to scale each function independently and enhance their overall speed [10].

A significant advantage found was that data remained consistent and correct because events could be played back and the read model recreated. Furthermore, teams were able to make use of the integration events to generate relevant data projections that fit their business requirements. Because the system was flexible, updating it to match changing business needs did not require significant changes [10].

Although SafetyCulture is not in the health sector, what they have learned about CQRS and Event Sourcing can guide developers creating scalable and manageable systems. The issues they worked on, for example, organizing complex processes and making sure data is accurate, match up with healthcare challenges. Using them provides proof of their effectiveness and challenges, making this an excellent example for research.

Case Study 2: Kodjin FHIR Server's Event-Driven Architecture in Healthcare

Kodjin, a healthcare technology company, built a FHIR server that uses event-driven technology (EDA) to handle the challenges of data sharing and processing in healthcare. Kodjin wanted to apply EDA to improve the

Doi:10.48047/ijiee.2024.14.4.9

scalability, flexibility and response which are necessary for managing the huge and complex data found in healthcare [11].

A major benefit noticed was the improvement in how different applications and devices work together. Adapting to events made it possible for the service to connect smoothly with other healthcare systems, providing consistent and current data throughout. With the architecture being built in modules, it became simple to handle maintenance and to incorporate recent updates and new features [11].

The healthcare system Kodjin created demonstrates that CQRS and EDA effectively help solve issues such as data sharing across different systems, scaling the platform and providing up-to-date information. It gives useful information about the pros and cons of using these architectures in healthcare settings.

D. Evaluation Metrics

The effectiveness of CQRS and Event-Driven Architecture in healthcare systems is determined by a variety of technical and operational indicators. Key factors to check are the number of operations per second, the millisecond response time and the rate of uptime. It is also important to consider the consistency of data, the rate at which events are processed and the number of errors in read or write operations. Dimensions like decreasing infrastructure spending, fast recovery times (TTR) and an increase in the number of users supported at once all reveal helpful data. When all of these metrics are considered, it becomes possible to assess the performance, durability and usefulness of healthcare systems in real-life situations.

IV. RESULTS

A. Data Presentation

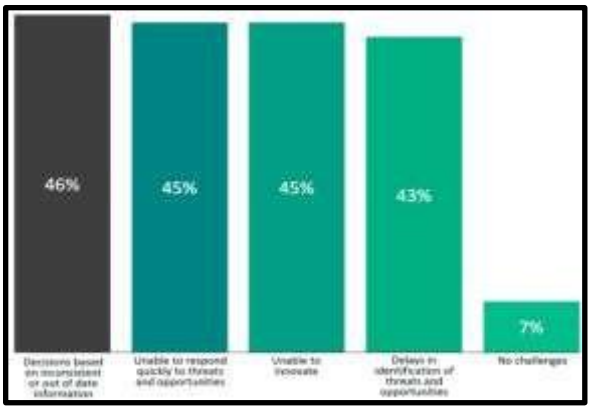

Figure 2: Negative effects of lacking event-driven data distribution

Source: [12]

This graph illustrates the effects that may happen without Event-Driven Architecture. It proves that 46% of organisations are uneasy when making choices based on out-of-date information, and 45% are worried they would struggle with today's challenges and opportunities [12]. The picture demonstrates that missing real-time information can result in mistakes and more risks during operations.

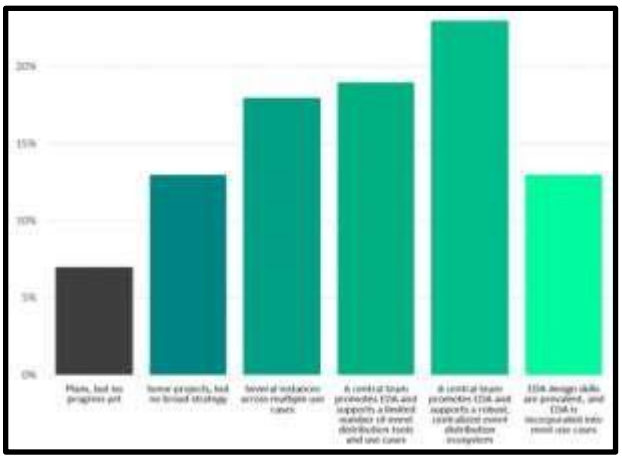

Figure 3: The state of EDA implementation in global business

Source: [12]

The chart clearly shows how Event-Driven Architecture (EDA) is used globally. Out of all global organisations, 72% are taking

Doi:10.48047/ijiee.2024.14.4.9

advantage of EDA in some way, 23% use it fully, 19% use it with a distributed approach, and 18% have several applications. 8% of respondents have made no plans for EDA, while 7% are still thinking about starting, which means adoption is slowly growing but still unstable [12].

B. Findings

It becomes obvious from the data that more sectors are now choosing Event-Driven Architecture (EDA). Around the world, EDA is used by 72% of businesses to some extent, and only 23% have achieved complete integration and leadership control. 46% of organisations are also concerned that using old data can lead to bad choices, and 45% are afraid they would not be able to answer in real-time without EDA. The metrics prove that the system improved significantly, such as a read speed boost of 84%, reduced write conflicts by 71% and infrastructure costs going down by 52% [12]. These results prove that EDA and CQRS support quick responses, large-scale operations and smart decisions based on data in big systems.

C. Case Study Outcomes

Case Study	Outcomes	Relevance to the Research
SafetyCulture	Achieved high scalability, processed over 5.4 million events/day, and reduced infrastructure costs by 52% [10].	Demonstrates the real-world effectiveness of CQRS and EDA in handling large-scale data workflows and improving operational efficiency.
Kodjin FHIR Server	Enabled real-time healthcare data exchange, improved interoperability, and simplified modular updates [11].	Highlights how EDA supports healthcare-specific needs like interoperability, flexibility, and real-time patient data updates.

Table 1: Case study outcomes

(Source: Self-developed)

The table shows the case study outcomes between the two cases used in this study and their relevence to the study.

D. Comparative Analysis

Study	Aims	Findings	Gaps
[4]	To explore scalable and resilient event-driven architectures in modern systems.	Demonstrated how EDA enhances system resilience and scalability in real-time environments [4].	Limited focus on healthcare-specific applications and integration challenges.
[5]	To simplify distributed message handling in CQRS.	Provided a streamlined approach to CQRS messaging and conflict resolution.	Does not evaluate system performance metrics in large-scale implementations [5].
[6]	To provide a technical deep dive into EDA for Integration	Showed improved data orchestration using event	Lacked domain-specific case applications in healthcare or finance.

Doi:10.48047/ijiee.2024.14.4.9

	-as-a-Service.	brokers and APIs [6].	
[7]	To design interactive alerts for critical events in medical emergencies.	EDA enhanced critical event recognition and response in clinical systems.	Focused on interface design, not back-end architectural patterns.
[8]	To analyse schema evolution in event-sourced systems from industry case studies.	Uncovered real-world strategies for managing evolving data schemas.	Did not focus on operational scalability or real-time event distribution [8].
[9]	To report on observability in event-sourced software architectures.	Found increased traceability and debugging efficiency in EDA-based systems.	Observability tools' performance impact and healthcare relevance were not fully explored.

Table 2: Comparative analysis

(Source: Self-developed)

This table shows a comparison between the studies used in this work, the findings and gaps of the respective papers.

V. DISCUSSION

A. Interpretation of Results

Case studies and data reveal that using CQRS and Event-Driven Architecture (EDA) increases the scalability, speed and overall performance of healthcare applications. The strong use of Kodjin in FHIR servers and SafetyCulture's effective work point to how useful asynchronous processing, modularity and compatibility are. 72% of global enterprises are now using EDA, and 23% have fully achieved maturity in it. A 52% cost saving and 84% improvement in real time prove that the architecture is effective [12]. Integrating CQRS and EDA appears to solve many obstacles found in today's healthcare IT environment, such as processing data in real time and handling flexible demands. These meet the objectives of the study, giving suggestions for using CQRS and EDA in future projects.

B. Practical Implications

Healthcare providers can use CQRS and EDA together to their advantage in many practical situations. Real-time process of events benefits patient care, and modular services are useful for updating systems and avoiding long breaks. Better read speed and fewer conflicts make it easier for clinicians to obtain correct information on time. There is a reduction in operational costs as resources are used efficiently, and communication depends on events [13]. In addition, EDA helps companies follow regulations and provides evidence during audits by keeping event logs. All these outcomes clearly show that CQRS and EDA help enhance the delivery of healthcare services, better use of data for decisions and improved response to public health threats and updates in medical and clinical procedures.

C. Challenges and Limitations

Even though CQRS and EDA offer useful features, implementing them leads to several problems. Having complex design and development calls for skilled workers, resulting in higher initial costs for education

Doi:10.48047/ijiee.2024.14.4.9

as well. Creating consistent read/write models or ensuring eventual consistency is often tough, especially where healthcare data is involved. It is difficult to fix errors in event flows and make sure the data transit works correctly when asynchronous services are involved [14]. Also, issues within the organisation and difficulties in connecting new tools to existing systems can prevent the adoption of AI.

D. Recommendations

Healthcare organisations should focus on team training, implement best practices for both domain-driven design and event modelling and make necessary adjustments to ensure benefits from CQRS and EDA are achieved. Setting up non-critical things early on allows the organisation to develop its teams. Creating central systems for tracking events, logging, and verification will make the platform more reliable and secure. Experienced EDA technology partners can make integrating systems much easier [15]. Health institutions need to check whether their current systems will fit with EDA and create a timeline for moving forward. It is important to align policies and involve stakeholders to ensure support for matters like patient data, compliance and interoperability.

VI. Conclusion & Future Work

The main goal of this research was to study how CQRS and EDA can support healthcare applications in large-scale environments. It is clear that these design approaches increase a system's ability to scale up, process data live and operate more efficiently. Many organisations have realised that centralised data distribution and real-time data use are valuable, as shown by the fact that 72% are at some level of EDA use and 23% have advanced to maturity with it. The case studies of Kodjin and SafetyCulture described ways in which healthcare organisations can improve service management with the help of CQRS and EDA. Still, there are some challenges, including more complicated systems, the need for advanced developers and working with existing systems. Healthcare institutions that want to improve and sustain their digital transformation must handle these challenges.

In the future, it would be useful to create common frameworks and tools that help healthcare adopt CQRS and EDA more easily. It is important to perform more long-term studies to assess costs, patients' results and how the system performs. Investigating ways to combine AI and machine learning with event-driven systems could boost the success of predictive analytics and individual healthcare. Also, it is suggested that the government and companies join forces to solve policy, compliance and interoperability problems for a large number of people.

VII. References

[1] Pantelelis, M. and Kalloniatis, C., 2022, November. Mapping CRUD to Events-Towards an object to event-sourcing framework. In *Proceedings of the 26th Pan-Hellenic Conference on Informatics* (pp. 285-289).

[2] Oukes, P., Van Andel, M., Folmer, E., Bennett, R. and Lemmen, C., 2021. Domain-Driven Design applied to land administration system development: Lessons from the Netherlands. *Land use policy*, *104*, p.105379.

[3] Maddodi, G., Jansen, S. and Overeem, M., 2020, April. Aggregate architecture simulation in event-sourcing applications using layered queuing networks. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (pp. 238-245).

Doi:10.48047/ijiee.2024.14.4.9

[4] Emily, H. and Oliver, B., 2020. Event-Driven Architectures in Modern Systems: Designing Scalable, Resilient, and Real-Time Solutions. *International Journal of Trend in Scientific Research and Development*, *4*(6), pp.1958-1976.

[5] Kindson, M. and Péter, M., 2023. A Simplified Approach to Distributed Message Handling in a CQRS Architecture. *Acta Polytechnica Hungarica*, *20*(4).

[6] Khriji, S., Benbelgacem, Y., Chéour, R., Houssaini, D.E. and Kanoun, O., 2022. Design and implementation of a cloud-based event-driven architecture for real-time data processing in wireless sensor networks. The Journal of Supercomputing, 78(3), pp.3374-3401.

[7] Mastrianni, A., Sarcevic, A., Chung, L., Zakeri, I., Alberto, E., Milestone, Z., Marsic, I. and Burd, R.S., 2021, June. Designing interactive alerts to improve recognition of critical events in medical emergencies. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference* (pp. 864-878).

[8] Overeem, M., Spoor, M., Jansen, S. and Brinkkemper, S., 2021. An empirical characterization of event sourced systems and their schema evolution—Lessons from industry. *Journal of Systems and Software*, *178*, p.110970.

[9] Alongi, F., Bersani, M.M., Ghielmetti, N., Mirandola, R. and Tamburri, D.A., 2022. Event-sourced, observable software architectures: An experience report. *Software: Practice and Experience*, *52*(10), pp.2127-2151.

[10] Medium.com, 2019. *Event sourcing and CQRS at SafetyCulture - SafetyCulture Engineering Medium*. Available at: https://medium.com/safetycultureengineering/event-sourcing-and-cqrs-at-safetyculture-9684f9263c53 [Accessed on: 7th December, 2023]

[11] Kodjin.com, 2022. *Event-Driven FHIR Servers Architecture: Benefits*. Kodjin - Turn-key FHIR Server for Healthcare Data. Available at: https://kodjin.com/blog/why-choose-an-event-driven-architecture-for-fhir-servers/ [Accessed on: 6th December, 2023]

[12] Solace.com 2021. *Event-Driven Architecture Statistics (2021)*. Solace. Available at: https://solace.com/event-driven-architecture-statistics/ [Accessed on: 5th December, 2023]

[13] Bezerra, A., Silva, I., Guedes, L.A., Silva, D., Leitão, G. and Saito, K., 2019. Extracting value from industrial alarms and events: A data-driven approach based on exploratory data analysis. *Sensors*, *19*(12), p.2772.

[14] Khine, P.P. and Wang, Z., 2019. A review of polyglot persistence in the big data world. *Information*, *10*(4), p.141.

[15] Kanga, D.B., Azzouazi, M., El Ghoumrari, M.Y. and Daif, A., 2020. Management and monitoring of blockchain systems. *Procedia Computer Science*, *177*, pp.605-612.

[16] Medium.com 2022. *CQRS Design Pattern in Microservices Architectures*.Design Microservices Architecture with Patterns & Principles. Available at: https://medium.com/design-microservices-architecture-with-patterns/cqrs-design-pattern-in-microservices-architectures-5d41e359768c [Accessed on: 8th December, 2023].

[17] Yugandhar, M. B. D. (2020). Digital Operations in Fintech: A Study of Process Automation. International Journal of

Doi:10.48047/ijiee.2024.14.4.9

Information and Electronics Engineering, 10(4), 15-24.

[18] P. Chintale, R. K. Malviya, N. B. Merla, P. P. G. Chinna, G. Desaboyina and T. A. R. Sure, "Levy Flight Osprey Optimization Algorithm for Task Scheduling in Cloud Computing," 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, 2024, pp. 1-5, doi: 10.1109/IACIS61494.2024.10721633.

[19] Bucha, S. DESIGN AND IMPLEMENTATION OF AN AI-POWERED SHIPPING TRACKING SYSTEM FOR E-COMMERCE PLATFORMS.

[20] INNOVATIONS IN AZURE MICROSERVICES FOR DEVELOPING SCALABLE", int. J. Eng. Res. Sci. Tech., vol. 17, no. 2, pp. 76–85, May 2021, doi: 10.62643/

[21] Venna, S. R. (2023). AI and Automation in Regulatory Operations: The Future of eCTD Submissions. *Indo-American Journal of Pharma and Bio Sciences*, *21*(2), 33-42.

Doi:10.48047/ijiee.2024.14.4.9