AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
Peer Reviewed, Referred & Indexed Journal
E-ISSN:3069-0102
VOL.6, NO. 2(2026)

DESIGN AND IMPLEMENTATION OF A VULNERABLE WEB APPLICATION FOR SQL INJECTION ATTACK DEMONSTRATION AND PREVENTION

[1] Ravinder, [2] G.Neha,[3] A.Sri vidhya, [4] Chandrakanth

[1]Assistant Professor, [234]Students

Department of Cyber Security

Siddhartha institute of technology & sciences

shirisha.ravi@siddhartha.co.in, 23TQ1A6201@siddhartha.co.in, 23TQ1A6214@siddhartha.co.in, 23TQ1A6206@siddhartha.co.in

ABSTRACT

Web applications play a vital role in modern digital environments by providing users with access to various online services such as e-commerce, banking, social networking, and information management systems. These applications interact with databases to store and retrieve important data, including user credentials and personal information. However, if web applications are developed without proper security measures, they become vulnerable to cyber attacks. One of the most common and critical vulnerabilities affecting web applications is SQL Injection.

SQL Injection is a type of cyber attack in which malicious SQL commands are inserted into input fields such as login forms or search boxes. When an application fails to properly validate or sanitize user inputs, the attacker's input becomes part of the SQL query executed by the database. This allows attackers to manipulate database queries to bypass authentication mechanisms, retrieve sensitive information, modify records, or even delete entire databases. Because of its severe impact on application security, SQL Injection is considered one of the most dangerous web vulnerabilities and is included in the OWASP Top 10 web application security risks. The main objective of this project is to design and implement a vulnerable web application that demonstrates how SQL Injection attacks occur and how they can be prevented using secure coding techniques. The application includes a login interface where users enter their username and password to access the system. In the initial implementation, the application is intentionally designed with insecure SQL queries that directly concatenate user input with SQL statements. This allows attackers to manipulate the query logic and bypass authentication using SQL Injection techniques.After demonstrating the vulnerability, the project implements a secure solution to prevent SQL Injection attacks.The prevention mechanism uses parameterized queries through Prepared Statement, which separates SQL commands from user input. By treating user input as data rather than executable code, this method effectively prevents SQL Injection attacks and enhances the security of the application. The system is developed using HTML and CSS for the frontend user interface, Spring Boot for backend application logic, and MySQL as the database management system. The project provides a practical understanding of SQL Injection vulnerabilities and highlights the importance of secure coding practices in webapplication development. Through this implementation, developers and students can better understand how web application vulnerabilities arise and how thy can be mitigated using proper security techniques

.

I INTRODUCTION

Web applications rely heavily on databases to store and manage user information such as login credentials,personal details, and transaction data. In many applications, user inputs from forms such as login pages or search fields are directly used to construct SQL queries. If these inputs are not properly validated or handled securely,the application

Received: 04-03-2026 | Accepted: 13-03-2026 | Published: 01-04-2026 | www.ajaccm.com

becomes vulnerable to SQL Injection attacks.SQL Injection is a serious security vulnerability in which attackers insert malicious SQL commands into input fields to manipulate database queries.When SQL Injection vulnerabilities exist, attackers can bypass authentication mechanisms, gain unauthorizedaccess to sensitive information, modify or delete database records, and potentially compromise the entire system.

Such attacks can lead to significant data breaches, financial losses, and damage to the reputation of organizations.Many real-world web applications have suffered from SQL Injection attacks due to improper coding practices and lack of secure input handling.

The main problem addressed in this project is the presence of insecure database query construction in web applications, which makes them susceptible to SQL Injection attacks. There is a need to understand how these vulnerabilities occur and how they can be effectively prevented using secure coding techniques.This project aims to design and implement a vulnerable web application that demonstrates SQL Injection attacks in a controlled environment. It also focuses on implementing prevention techniques using secure methods such as parameterized queries to ensure safe interaction between the application and the database. The primary objective of this project is to design and implement a web application that demonstrates SQLInjection vulnerabilities and their prevention using secure coding techniques. The project aims to provide a practical understanding of how insecure database query construction can lead to serious security risks in web applications. Another important objective is to develop a vulnerable login system that interacts with a database and intentionally allows SQL Injection attacks for demonstration purposes. This helps in understanding how attackerscan manipulate input fields to bypass authentication mechanisms and gain unauthorized access to a system. The project also aims to highlight the importance of secure programming practices in web application development. By implementing both vulnerable and secure versions of database queries, the project demonstrates how improper input handling can lead tovulnerabilities and how these issues can be prevented.

Additionally, the project focuses on implementing a secure solution using parameterized queries through PreparedStatement to prevent SQL Injection attacks. This method ensures that user inputs are treated as data rather than executable SQL commands.Finally, the project aims to increase awareness about web application security among developers and students by providing a practical example of SQL Injection attacks and their mitigation techniques.The scope of this project is to design and implement a web application that demonstrates SQL Injection

II LITERATURE SURVEY

Web applications are widely used in modern computing environments for providing services such as online banking, e-commerce, and information management. These applications rely heavily on backend databases to store and retrieve data. However, improper handling of user inputs can expose applications to security vulnerabilities. Among these vulnerabilities, SQL Injection is one of the most common and critical threats affecting web applications. SQL Injection occurs when malicious SQL statements are inserted into input fields,allowing attackers to manipulate database queries and gain unauthorized access to sensitive information.(Wikipedia)With the increasing number of web-based applications, the importance of securing databases and preventing SQL Injection attacks has become a major concern for developers and researchers. Various studies have been conducted to analyze the causes of SQL Injection vulnerabilities and to develop techniques for detecting and preventing such attacks. The literature review helps in understanding existing research, identifying different types of SQL Injection attacks, and exploring methods that can be used to mitigate these vulnerabilities. The project titled “Design and Implementation of a Vulnerable Web Application for SQL Injection AttackDemonstration and Prevention” focuses on understanding how SQL Injection attacks occur in web applicationsand how they can be prevented through secure coding practices.SQL Injection is a type of injection attack where malicious SQL commands are inserted into user input fields such as login forms. If the application constructs SQL

Received: 04-03-2026 | Accepted: 13-03-2026 | Published: 01-04-2026 | www.ajaccm.com

queries using unsanitized input, attackers can manipulate the query logic to bypass authentication or retrieve confidential data from the database. (Cisco)

This project aims to build a web application that intentionally contains a SQL Injection vulnerability to demonstrate how attackers exploit insecure coding practices. After demonstrating the attack, the system implements prevention techniques using parameterized queries, which separate SQL commands from user input and protect the application from malicious manipulation. Several researchers have studied SQL Injection vulnerabilities and proposed various detection and prevention techniques. SQL Injection remains one of the most widely studied web security threats because of its potential to compromise sensitive data and disrupt database systems.Researchers have identified that SQL Injection vulnerabilities mainly occur due to improper validation of user inputs and insecure query construction. When developers directly concatenate user inputs into SQL statements, attackers can inject malicious commands that alter the intended behavior of the query.(ResearchGate) Studies also show that SQL Injection attacks can allow attackers to bypass authentication mechanisms, access restricted data, and manipulate database records. Because web applications handle large amounts of user data, these vulnerabilities can lead to serious security breaches and data leaksIn recent years, many advanced techniques have been developed to detect and prevent SQL Injection attacks.Traditional prevention techniques include input validation, parameterized queries, stored procedures, and access control mechanisms. Among these methods, parameterized queries and prepared statements are widelyrecommended because they prevent user inputs from being interpreted as executable SQL code. (OWASP Cheat Sheet Series)Recent research has also explored the use of machine learning and deep learning techniques to detect SQL Injection attacks automatically. These methods analyze patterns in SQL queries and identify malicious inputs that SQL Injection is based on the concept of code injection, where attackers insert malicious commands into input fields that are later executed by the database system. In a typical web application architecture, the user interacts with the frontend interface, which sends input data to the server. The server then constructs SQL queries and communicates with the database to retrieve or update information. If user input is not properly validated, attackers can inject SQL commands that modify the structure of the query.

For example, attackers may use logical conditions such as OR 1=1 or comment symbols to manipulate authentication queries. This allows them to bypass login systems and gain unauthorized access to the application.

(OWASP)Understanding this theory is essential for developing effective techniques to prevent SQL Injection attacks.deviate from normal query structures. Machine learning models have shown promising results in detecting previously unknown SQL Injection patterns. (MDPI) Despite these advancements, SQL Injection remains a persistent threat because many applications still use insecure coding practices. The literature review shows that SQL Injection continues to be a significant security threat to web applications. Most vulnerabilities occur due to improper input validation and insecure construction of SQL queries.Researchers have proposed various methods to detect and prevent SQL Injection attacks, including parameterized queries, input validation, machine learning-based detection, and secure development practices.Among these techniques, the use of prepared statements and parameterized queries is considered one of the most effective methods for preventing SQL Injection vulnerabilities. These methods ensure that user inputs are treated as data rather than executable SQL commands.Based on the insights obtained from previous research, this project focuses on developing a vulnerable web application to demonstrate SQL Injection attacks and implementing secure coding techniques to prevent them. The project aims to provide practical understanding of SQL Injection vulnerabilities and highlight the importance of secure programming practices in web application development.

III METHODOLOGY

Methodology describes the systematic approach followed for the design and implementation of the proposed system. It

Received: 04-03-2026 | Accepted: 13-03-2026 | Published: 01-04-2026 | www.ajaccm.com

explains the steps involved in developing the vulnerable web application used for demonstrating SQL Injection attacks and implementing secure coding techniques for preventing such attacks. The methodology focuses on understanding how SQL Injection vulnerabilities occur in web applications and how they can be mitigated using proper security practices.In this project, a web-based login system is developed that interacts with a database for user authentication. The system is initially designed with insecure query construction methods that allow SQL Injection attacks to occur.This helps in demonstrating how attackers can manipulate input fields to bypass authentication. After demonstrating the vulnerability, secure coding techniques such as parameterized queries using PreparedStatement are implemented to prevent SQL Injection attacks.The system follows a structured development approach involving frontend development, backend implementation, database integration, and security validation. The application is built using HTML and CSS for the user interface, Spring Boot for backend processing, and MySQL as the database management system. The methodology followed in this project involves severalstages including system design, implementation of the vulnerable application, demonstration of SQL Injection attacks, and implementation of prevention techniques.Initially, the login interface of the web application is designed using HTML and CSS. The user enters login credentials through this interface. These credentials are then sent to the backend server built using Spring Boot.In the vulnerable version of the application, the backend constructs SQL queries by directly concatenating user inputs into the SQL statement. This insecure practice allows attackers to manipulate the query structure using SQL Injection techniques. The system is then tested using malicious inputs to demonstrate authentication bypass.

After demonstrating the vulnerability, a secure version of the application is implemented using parameterized queries through PreparedStatement. This technique separates SQL commands from user inputs, ensuring that theinput is treated as data rather than executable SQL code. This prevents attackers from altering the SQL query structure.

Existing system

Several existing systems and applications are available that demonstrate web security vulnerabilities, particularly SQL Injection attacks. Popular tools such as DVWA (Damn Vulnerable Web Application), OWASP WebGoat, and bWAPP are widely used for educational and training purposes. These applications provide intentionally vulnerable environments where users can practice exploiting common web vulnerabilities, including SQL Injection, cross-site scripting, and authentication bypass attacks. For example, DVWA offers different security levels to help users understand how vulnerabilities behave under various configurations, while WebGoat, developed by OWASP, provides interactive lessons that guide users through attack scenarios and prevention techniques.

However, these existing systems have several limitations. Most of them require complex setup procedures, such as configuring local servers and databases, which can be difficult for beginners. Additionally, they are often not user-friendly for students who lack a strong background in cybersecurity. While these platforms effectively demonstrate how attacks are performed, they do not always provide a clear, step-by-step explanation of how to implement prevention techniques in real-world applications. Moreover, many of these tools are not integrated with modern web development frameworks like Django, making it harder for students to relate them to current industry practices.

In contrast, modern web applications in the industry are designed to be secure by default. They use techniques such as parameterized queries, prepared statements, input validation, and Object-Relational Mapping (ORM) frameworks to prevent SQL Injection attacks. However, these secure systems do not demonstrate how vulnerabilities occur internally, which makes it difficult for learners to understand the attack mechanism.

Therefore, while existing systems focus on vulnerable environments or secure implementations, there is a gap in providing a simple, beginner-friendly application that demonstrates both SQL Injection attacks and

Received: 04-03-2026 | Accepted: 13-03-2026 | Published: 01-04-2026 | www.ajaccm.com

their prevention in a clear and practical manner.

Disadvantages of Existing System

Although existing systems such as DVWA, OWASP WebGoat, and bWAPP are useful for understanding web vulnerabilities, they have several disadvantages.

1. One major drawback is that they require complex installation and configuration processes
2. They mainly focus on demonstrating attacks rather than providing clear, step-by-step guidance on how to prevent SQL Injection in real-world applications.
3. They also lack real-time visualization of how SQL queries are manipulated during an attack,

Proposed System

The proposed system is a web-based application designed to demonstrate SQL Injection attacks and their prevention in a simple, clear, and interactive manner. Unlike existing tools such as DVWA and OWASP WebGoat, this system focuses on providing a beginner-friendly environment that combines both vulnerability demonstration and secure implementation within a single platform. The application will include modules such as a login page and data input forms that are intentionally vulnerable to SQL Injection, allowing users to observe how attackers can manipulate SQL queries to bypass authentication or access unauthorized data. Alongside this, the system will also provide a secure version of the same modules using techniques like parameterized queries, input validation, and ORM-based database interaction to prevent such attacks.

The proposed system will be developed using modern web technologies, such as Django for backend development and a relational database for data storage, ensuring relevance to current industry practices. It will also include features like real-time query display, where users can see how input data affects SQL queries, improving their conceptual understanding**.** The proposed system is a web-based application designed to provide a comprehensive and practical understanding of SQL Injection attacks along with effective prevention techniques. Unlike existing platforms such as DVWA and OWASP WebGoat, this system is specifically developed with a focus on simplicity, clarity, and ease of use for students and beginners. It will include intentionally vulnerable modules such as login forms, search fields, and data entry pages where users can perform SQL Injection attacks and observe how unauthorized access can be achieved. At the same time, the system will provide parallel secure implementations of these modules, allowing users to clearly compare vulnerable and protected code.

A key feature of the proposed system is real-time visualization, where users can see the actual SQL queries being executed based on their inputs. This helps in understanding how malicious inputs alter query structure and lead to security breaches. The system will also include detailed explanations, guided demonstrations, and step-by-step execution flows to improve conceptual clarity. Additionally, it will be built using modern technologies such as Django and integrated database systems, making it relevant to current industry standards and practices.

The application will also incorporate user-friendly interfaces, minimal setup requirements, and interactive learning features such as predefined attack examples and prevention techniques. By combining both attack simulation and secure coding practices in one platform, the proposed system aims to bridge the gap between theoretical learning and real-world implementation, making it an effective educational tool for understanding web application security.

Additionally, step-by-step explanations and comparisons between vulnerable and secure code will be provided to enhance learning. Overall, the system aims to bridge the gap between theoretical knowledge and practical implementation by offering a comprehensive, easy-to-use platform for understanding both SQL Injection attacks and their prevention.

Advantages of Proposed System

The proposed system offers several advantages over existing solutions such as DVWA and OWASP WebGoat.

1 its beginner-friendly design, which makes it easy for students with little or no cybersecurity background to understand SQL Injection concepts.

2.Another key advantage is that it demonstrates both vulnerable and secure implementations side-by-side

Received: 04-03-2026 | Accepted: 13-03-2026 | Published: 01-04-2026 | www.ajaccm.com

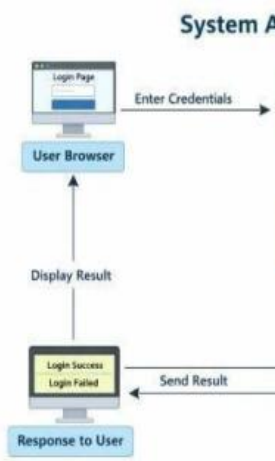
3.The application also emphasizes practical learning by providing step-by-step demonstrations of attacks and their prevention..

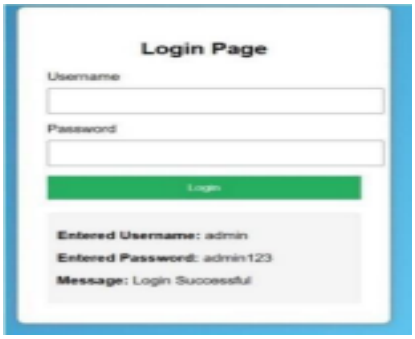
System Architecture

The proposed system follows a three-tier architecture, which separates the application into three main layers: presentation layer, application layer, and database layer. This architecture ensures better organization, scalability, and security while demonstrating SQL Injection vulnerabilities and prevention techniques.

The presentation layer (frontend) is responsible for user interaction. It includes web pages such as login forms, input fields, and result displays. Users interact with the system by entering data, which is then sent to the backend for processing. This layer is designed to be simple and user-friendly so that even beginners can easily perform and understand SQL Injection attacks.The application layer (backend) handles the core logic of the system. It processes user inputs, constructs SQL queries, and communicates with the database. In this system, two types of modules are implemented: vulnerable modules and secure modules. The vulnerable modules directly include user input in SQL queries, making them susceptible to SQL Injection attacks. In contrast, the secure modules use techniques such as parameterized queries, input validation, and ORM (Object-Relational Mapping) to prevent such attacks. This layer also manages features like real-time query display and comparison between insecure and secure executions.

The database layer is responsible for storing and managing data, such as user credentials and application records. It executes SQL queries received from the backend and returns results accordingly. In the vulnerable scenario, manipulated queries can alter database behavior, while in the secure scenario, proper query handling ensures data integrity and security.



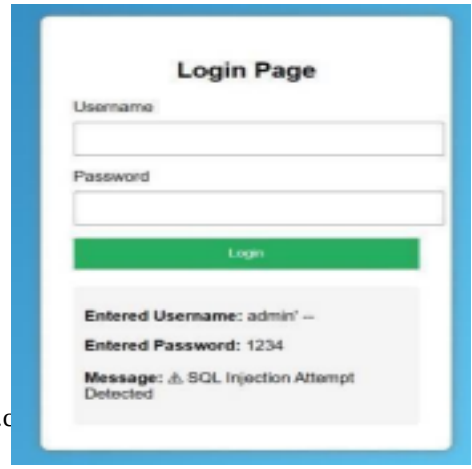


IV

Result&output

This chapter presents the results obtained from the implementation of the web application developed to demonstrate SQL Injection attacks and their prevention. The system was tested under different scenariosincluding valid login, invalid login, SQL Injection attack demonstration, and SQL Injection prevention.The screenshots included in this chapter illustrate how the system behaves under each test condition.

This screenshot shows the login interface when the user enters correct username and password. The backend application verifies the credentials stored in the MySQL database. Since the entered credentials match the database records, the system displays the message **"Login Successful."**



Received: 04-03-2026 | Accepted: 13-03-2026 | Published: 01-04-2026 | www.ajaccm.c

This screenshot demonstrates the SQL Injection attack on the vulnerable login system. The attacker enters a malicious SQL payload.
This modifies the SQL query executed by the backend application and bypasses the authentication process. As a result, the system incorrectly grants access and displays the message **"Login Successful."**

This screenshot shows the secure implementation of the system after applying SQL Injection prevention techniques. The application uses **PreparedStatement**, which prevents malicious input from altering the SQL query structure. When an SQL Injection attempt is made, the system detects the attack and displays the message **"Login Failed – SQL Injection Attempt Detected."**

V Conclusion

In conclusion, the project "Design and Implementation of a Vulnerable Web Application for SQL Injection Attack Demonstration and Prevention" provides a practical and effective approach to understanding one of the most critical web security vulnerabilities. By combining both vulnerable and secure modules within a single platform, the system helps users clearly visualize how SQL Injection attacks are performed and how they can be prevented using proper coding techniques. Unlike existing systems such as DVWA and OWASP WebGoat, the proposed system focuses on simplicity, real-time understanding, and ease of use, making it highly suitable for students and beginners.
The inclusion of features such as real-time query visualization, step-by-step demonstrations, and modern development practices enhances both conceptual clarity and practical knowledge. This project not only strengthens the user's understanding of SQL Injection attacks but also emphasizes the importance of secure coding practices in real-world web development. Overall, the system serves as a valuable educational tool that bridges the gap between theoretical knowledge and practical implementation, preparing learners to build more secure web applications in the future. the project "Design and Implementation of a Vulnerable Web Application for SQL Injection Attack Demonstration and Prevention" successfully highlights the importance of web application security by providing a practical platform for learning and experimentation. It enables users to gain a clear understanding of how SQL Injection attacks occur, how attackers exploit system vulnerabilities, and how such threats can be effectively mitigated using secure coding practices. By integrating both vulnerable and secure modules, the system offers a comparative learning approach that enhances both theoretical knowledge and hands-on experience.
Compared to existing tools like DVWA and OWASP WebGoat, the proposed system emphasizes simplicity, accessibility, and real-time learning, making it more suitable for beginners and academic use. The use of modern technologies and structured architecture ensures that the application remains relevant to current industry standards. Additionally, features such as query visualization and guided demonstrations help users better understand the internal working of

Received: 04-03-2026 | Accepted: 13-03-2026 | Published: 01-04-2026 | www.ajaccm.com

database interactions and security mechanisms.

Furthermore, this project promotes awareness about secure web development practices and encourages developers to adopt preventive measures such as parameterized queries, input validation, and ORM usage. It acts as a bridge between classroom learning and real-world application development. Overall, the system not only demonstrates vulnerabilities but also empowers users with the knowledge and skills required to build secure and robust web applications, making it a valuable contribution to cybersecurity education.

References

[1] Kumar, R. D., Prudhviraj, G., Vijay, K., Kumar, P. S., & Plugmann, P. (2024). Exploring COVID-19 through intensive investigation with supervised machine learning algorithm. In Handbook of Artificial Intelligence and Wearables (pp. 145-158). CRC Press.

[2] Swathi, B., Vijay, K., Sushanth Babu, M., & Dinesh Kumar, R. (2024, November). Machine Learning Techniques in Cloud Based Intrusion Detection. In The International Conference on Artificial Intelligence and Smart Environment (pp. 557-564). Cham: Springer Nature Switzerland.

[3] Sv satyakrishna, shirisha rangu ,bhargavi nalacheruve.(2024) Prospective investigation on colorectal cancer with SMOTE on machine learning Algorithm

[4] Dr.G.Vishnu Murthy, BhargaviNalacheruve 1Professor, Department of computer Science & engineering, Anurag University, TS, India. 2Student, Department of computer Science & engineering, Anurag University, TS, India.

[5] V. N. S. Manaswini, K. K, C. Nigam, S. S. Ali, R. Niranjana, and Suman, “Real-Time Object Detection in Drone Surveillance Using YOLOv5,” in Proc. 2025 3rd Int. Conf. IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 2025, pp. 1–6, doi: 10.1109/ICICAT68430.2025.11414670.

[6] B. Soundarya, V. N. S. Manaswini, M. Ayyakrishnan, R. D. Kumar, “Contextual Analysis of Big Data Analytics in Intelligent Transportation Frameworks,” in Intersection of Artificial Intelligence, Data Science, and Cutting-Edge Technologies: From Concepts to Applications in Smart Environment, Lecture Notes in Networks and Systems, vol. 1353, Cham: Springer, 2025, doi: 10.1007/978-3-031-88304-0_79.

[7] R. D. Kumar, V. N. S.Manaswini, “Applications of blockchain in smart cities: detecting fake documents from land records using blockchain technology,” in Blockchain for Smart Cities, Elsevier, 2021, pp. 105–117, doi: 10.1016/B978-0-12-824446-3.00017-X.

[8] Tejavath Veeramma, Badarla Anil, Guguloth Ravinder, “An advanced movie recommender using collaborative filtering and sentiment analysis,” International Research Journal of Modernization in Engineering Technology and Science, vol. 7, no. 7, July 2025, doi: 10.56726/IRJMETS81618.

[9] Ravi Kumar Banoth, Ramana Murthy B V, “Automatic crop recommendation system using LightGBM and decision tree machine learning models,” Journal of Machine and Computing, vol. 5, no. 1, pp. 343, Jan. 2025, doi: 10.53759/7669/jmc202505026.

[10] Ravi Kumar Banoth, Dr. B.V. Ramana Murthy, “Smart agriculture through IoT and machine learning for analyzing carbon footprints,” in Proc. Int. Conf. Computer Science and Communication Engineering (ICCSCE), Apr. 2025. [11] Ravi Kumar Banoth, B. V. Ramana Murthy, “Soil image classification using transfer learning approach: MobileNetV2 with CNN,” SN Computer Science, vol. 5, art. no. 199, 2024, doi: 10.1007/s42979-023-02500-x.

Received: 04-03-2026 | Accepted: 13-03-2026 | Published: 01-04-2026 | www.ajaccm.com