AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
E-ISSN:3069-0102
VOL.6, NO. 2 (2026)

COMPREHENSIVE SQL INJECTION ATTACK AND AUTHENTICATION BYPASS TESTING USING VULNERABLE WEB APPLICATIONS

[1] Dr B.Ravi kumar, [2] L Nandhini,[3] G Ramcharan , [4] R Dimple,[5] S Shivakumar
[1]AssistantProfessor, [2345]Students
Department of Cyber Security
Siddhartha Institute of Technology & Sciences, Narapally
dr.brkou@siddhartha.org.in, 24TQ1A6214@siddhartha.co.in, 24TQ1A6224@siddhartha.co.in,
24TQ1A6231@siddhartha.co.in, 24TQ1A6222@siddhartha.co.in

Abstract

SQL Injection is one of the most critical web application vulnerabilities, allowing attackers to manipulate database queries by inserting malicious SQL code into user inputs. This project focuses on the comprehensive analysis and testing of SQL Injection attacks and authentication bypass techniques using intentionally vulnerable web applications in a controlled environment. The study demonstrates how weak input validation and improper authentication mechanisms can lead to unauthorized access, data leakage, and compromise of sensitive information.

Various SQL Injection techniques, including error-based, union-based, and login bypass attacks, are implemented to evaluate their impact on web security. Authentication bypass is specifically analyzed by injecting crafted SQL payloads into login forms to gain access without valid credentials. The testing is conducted on educational vulnerable applications, examining different input points such as login forms, search fields, and URL parameters.

The project also highlights effective prevention techniques such as input validation, parameterized queries, prepared statements, and secure authentication practices. By comparing vulnerable and secured implementations, it demonstrates how proper coding practices can mitigate SQL Injection risks. Additionally, the study emphasizes ethical testing by conducting all experiments in authorized environments. Overall, this project provides practical insights into identifying, exploiting, and preventing SQL Injection vulnerabilities, making it valuable for students, developers, and cybersecurity professionals.

I. Introduction

Web application security has become a critical aspect of modern software development, as most online systems rely on databases to store and manage sensitive information. With the rapid growth of web applications across sectors such as education, banking, healthcare, and business, the risk of cyberattacks has also increased significantly. Attackers often exploit weaknesses in application logic and insecure database connections to gain unauthorized access or manipulate stored data. Among various database-related threats, SQL Injection is considered one of the most severe vulnerabilities because it directly targets database queries and authentication mechanisms. This project focuses on understanding SQL Injection attacks and authentication bypass techniques through practical testing using vulnerable web applications in a controlled environment. Web applications play a vital role in everyday life, as they are widely used in areas such as education, banking, healthcare,

Received: 28-04-2026 | Accepted: 02-06-2026 | Published: 11-06-2026

e-commerce, communication, and business operations. Organizations depend on these systems to store, process, and exchange large amounts of information over the internet. Since these applications handle sensitive data such as usernames, passwords, financial records, and personal details, ensuring their security has become a top priority.

II. Literature Survey

Web application security has become a major area of research due to the increasing dependence of modern applications on databases for storing and managing sensitive information such as user credentials, personal records, financial transactions, and confidential organizational data. With the rapid growth of internet-based services, the frequency and sophistication of cyberattacks targeting web applications have also increased. Researchers in cybersecurity have consistently identified SQL Injection as one of the most dangerous and widely exploited vulnerabilities because it directly interacts with the database layer.

Various studies highlight that even minor coding errors, such as improper input validation or insecure query construction, can create serious security weaknesses. These vulnerabilities allow attackers to manipulate backend database queries, leading to unauthorized access, data leakage, or modification of critical information. Literature on SQL Injection primarily focuses on different attack techniques, authentication weaknesses, testing methodologies, and preventive measures to ensure secure web application development.

Researchers have also compared SQL Injection with other web vulnerabilities such as Cross-Site Scripting (XSS) and session hijacking. These comparisons reveal that SQL Injection is particularly severe because it targets the database directly, which is the core component of any application. While front-end attacks mainly affect user interaction, SQL Injection can expose or compromise the entire backend system if not properly secured.

Recent studies emphasize the use of automated vulnerability assessment tools to detect SQL Injection flaws. These tools scan web applications to identify insecure input fields, weak query handling, and potential attack points. However, research also indicates that automated tools alone are not sufficient, as some complex vulnerabilities may go undetected. Therefore, manual testing and expert analysis are considered essential for comprehensive security evaluation.

Furthermore, many authors stress the importance of integrating security into the Software Development Life Cycle (SDLC). Implementing secure coding practices, performing regular security testing, and applying proper validation techniques during development can significantly reduce vulnerabilities. Researchers conclude that a combination of secure design, continuous monitoring, and practical testing provides effective protection against SQL Injection and authentication bypass attacks.

Overall, the literature highlights the critical need for understanding both attack techniques and defense mechanisms. It supports the importance of practical testing using vulnerable applications, as carried out in this project, to gain real-world insights into web application security and improve defensive strategies.

Received: 28-04-2026 | Accepted: 02-06-2026 | Published: 11-06-2026

III. System Analysis

Web applications rely heavily on databases to store and manage sensitive data, making them a prime target for cyberattacks. SQL Injection is one of the most critical vulnerabilities that allows attackers to manipulate database queries through user input fields. The system must ensure secure handling of inputs, proper authentication, and safe database communication. Many applications fail due to weak validation and poor query construction. Therefore, analyzing how SQL Injection attacks occur is essential for improving security. The system should identify vulnerable entry points such as login forms, search fields, and URL parameters. It must also evaluate how authentication mechanisms can be bypassed using malicious inputs. Additionally, the system should compare vulnerable and secure implementations. Proper analysis helps in understanding attack patterns and designing effective countermeasures. This project focuses on identifying vulnerabilities and strengthening web application security.

Existing System

In the existing system, many web applications rely on basic input handling and simple authentication mechanisms. User inputs are often directly included in SQL queries without proper validation or sanitization. This makes applications vulnerable to SQL Injection attacks. Login systems sometimes use insecure query structures that allow attackers to bypass authentication using crafted inputs. Many applications lack proper use of prepared statements or parameterized queries. Security testing is often limited or not performed thoroughly during development. Automated tools may be used, but they may not detect all vulnerabilities. Developers sometimes focus more on functionality than security, leading to weak implementations. As a result, existing systems are often exposed to database-related attacks.

Disadvantages of Existing System

- Lack of proper input validation
- Vulnerable to SQL Injection attacks
- Weak authentication mechanisms
- Easy authentication bypass using malicious inputs
- Direct use of dynamic SQL queries
- Limited security testing during development
- Dependence only on automated tools
- Risk of data leakage and database compromise

Proposed System

The proposed system focuses on comprehensive testing of SQL Injection attacks and authentication bypass techniques in a controlled environment. It uses intentionally vulnerable web applications to demonstrate how attacks are performed. Different types of SQL Injection attacks such as error-based, union-based, and login bypass are tested. The system analyzes vulnerabilities in input fields like login forms, search boxes, and URL parameters. It also compares insecure and secure coding practices. Secure methods such as input validation, parameterized queries, and prepared statements are implemented. The system demonstrates how proper authentication mechanisms prevent unauthorized access. It emphasizes ethical testing and controlled

Received: 28-04-2026 | Accepted: 02-06-2026 | Published: 11-06-2026

experimentation. The proposed approach provides both attack and defense understanding. Overall, it helps improve web application security awareness and practices.

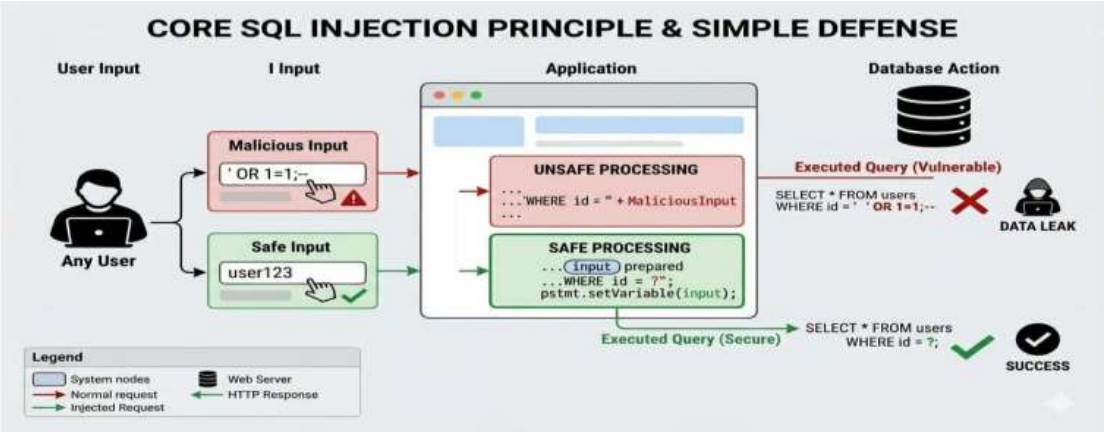
Advantages of Proposed System

- Provides practical understanding of SQL Injection attacks
- Helps identify real-world vulnerabilities
- Demonstrates authentication bypass techniques clearly
- Encourages secure coding practices
- Improves awareness of web security threats
- Supports both manual and controlled testing
- Shows comparison between vulnerable and secure systems
- Reduces risk of data breaches

IV. Methodology

The methodology begins with setting up a controlled environment using vulnerable web applications. Different input fields such as login forms, search boxes, and URL parameters are identified for testing. SQL Injection payloads are crafted and injected into these inputs to observe system behavior. Various attack techniques including error-based, union-based, and authentication bypass are tested. The results are analyzed to identify vulnerabilities and weaknesses. Secure coding techniques such as input validation and parameterized queries are then implemented. The same tests are repeated to verify if vulnerabilities are fixed. Both manual testing and tool-based analysis are performed for better accuracy. The system is evaluated based on security improvements and resistance to attacks. Finally, results are documented to demonstrate effectiveness.

System Architecture

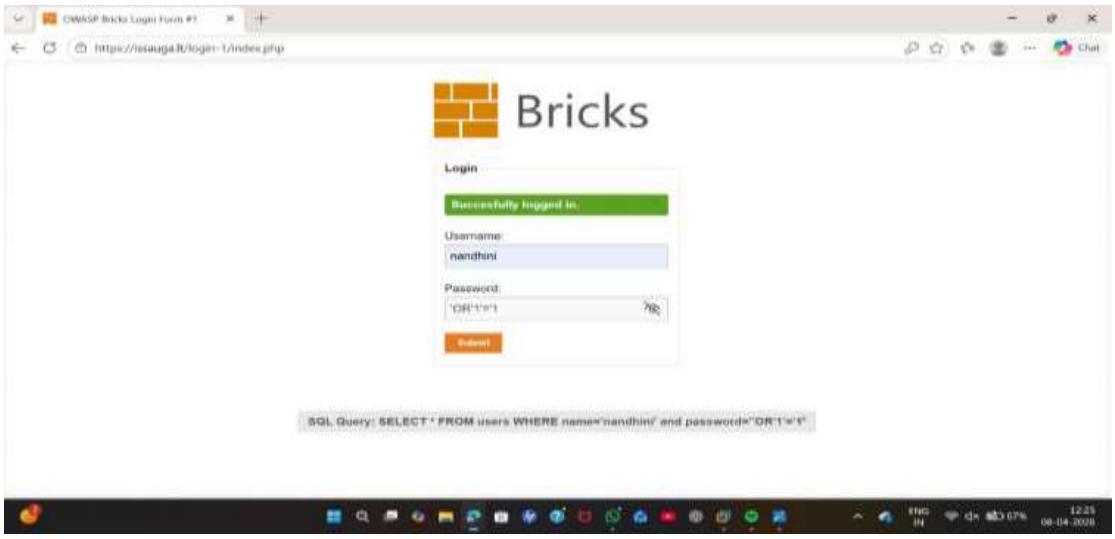


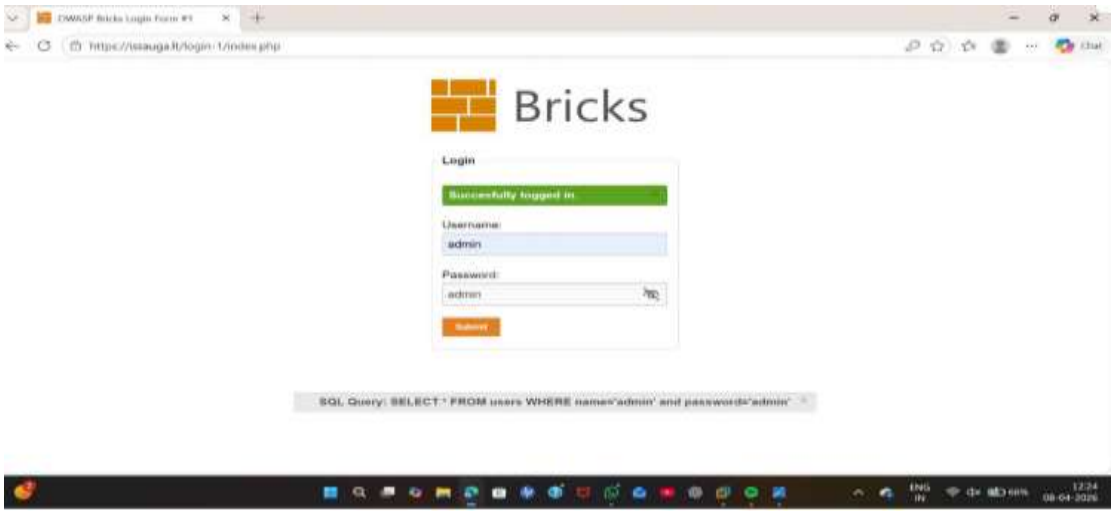
The system architecture consists of multiple components working together for testing and analysis. The user interface includes input fields such as login pages and search forms. The application layer processes user requests and interacts with the database. The database stores user data and responds to queries. The attack module injects SQL payloads into input fields to test vulnerabilities. The analysis module observes system responses and identifies weaknesses. The secure implementation module applies

Received: 28-04-2026 | Accepted: 02-06-2026 | Published: 11-06-2026

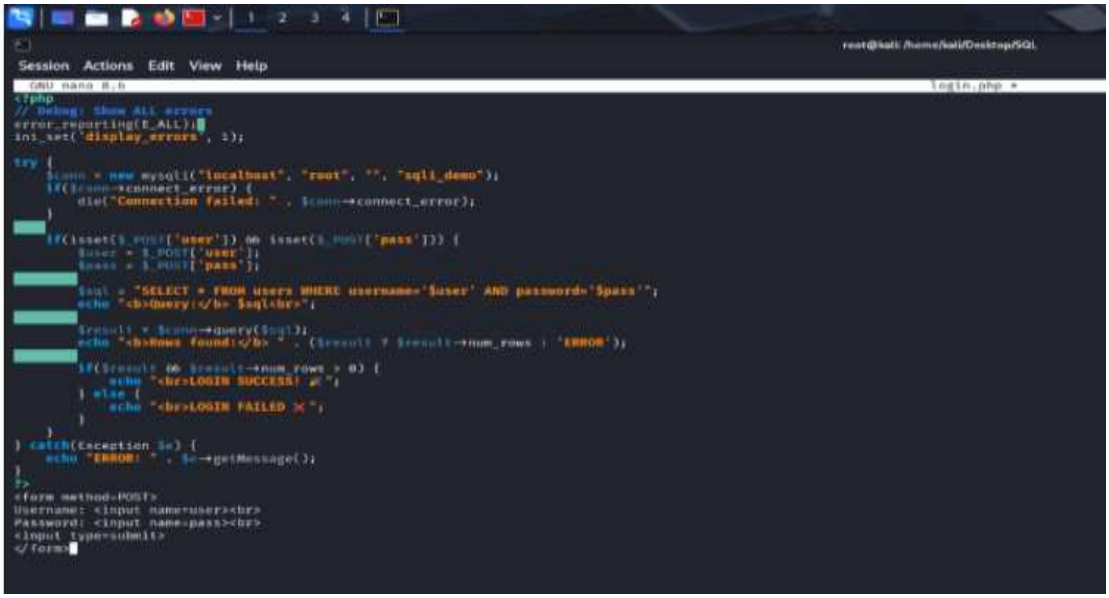
validation and safe query techniques. The testing module compares results before and after applying security measures. All operations are performed in a controlled and authorized environment. This architecture ensures effective testing, analysis, and improvement of web application security.

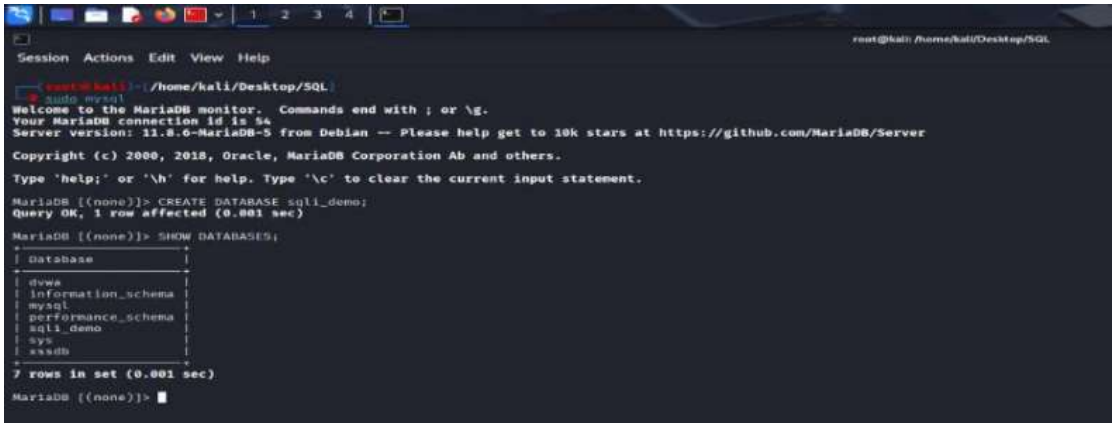
V. Result and Output





Received: 28-04-2026 | Accepted: 02-06-2026 | Published: 11-06-2026


Session Actions Edit View Help
GNU nano 8.6
login.php
<?php
// Debug: Show ALL errors
error_reporting(E_ALL);
ini_set('display_errors', 1);
try {
$conn = new mysqli("localhost", "root", "", "sqli_demo");
if($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
if(isset($_POST['user']) && isset($_POST['pass'])) {
$user = $_POST['user'];
$pass = $_POST['pass'];
$sql = "SELECT * FROM users WHERE username='$user' AND password='$pass'";
echo "Query: $sql
";
$result = $conn->query($sql);
echo "Rows found: " . ($result ? $result->num_rows : 'ERROR');
if($result && $result->num_rows > 0) {
echo "
LOGIN SUCCESS!";
} else {
echo "
LOGIN FAILED";
}
}
} catch(Exception $e) {
echo "ERROR: " . $e->getMessage();
}
?>
<form method=POST>
Username: <input name=user>
Password: <input name=pass>
<input type=submit>
</form>


Session Actions Edit View Help
/home/kali/Desktop/SQL
sudo mysql
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 54
Server version: 11.8.6-MariaDB-5 from Debian -- Please help get to 10k stars at https://github.com/MariaDB/Server
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> CREATE DATABASE sqli_demo;
Query OK, 1 row affected (0.001 sec)
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database |
+--------------------+
| dvwa |
| information_schema |
| mysql |
| performance_schema |
| sqli_demo |
| sys |
| xssdb |
+--------------------+
7 rows in set (0.001 sec)
MariaDB [(none)]>


Session Actions Edit View Help
7 rows in set (0.001 sec)
Database changed
Query OK, 0 rows affected (0.004 sec)
1 row in set (0.001 sec)
Query OK, 1 row affected (0.001 sec)
Query OK, 1 row affected (0.002 sec)
Query OK, 1 row affected (0.002 sec)
Query OK, 1 row affected (0.002 sec)
4 rows in set (0.001 sec)
Bye
/home/kali/Desktop/SQL

VI. Conclusion

The study of SQL Injection attacks and authentication bypass testing on vulnerable web applications clearly demonstrates the serious impact of insecure coding practices on application security. The experiments confirmed that improper input validation and lack of sanitization allow attackers to manipulate SQL queries, leading to unauthorized access to backend databases. Common input points such as login forms, search fields, and URL parameters were identified as major entry points for such attacks.

Authentication bypass testing further revealed that weak login mechanisms are highly vulnerable when SQL queries are directly constructed using user inputs. By injecting simple logical conditions, attackers can gain access to restricted areas without valid credentials. This highlights the critical risk associated with poor authentication design and insecure database interactions.

The analysis also showed that SQL Injection attacks can expose highly sensitive information, including usernames, passwords, database structures, and table details. Error messages generated by applications can unintentionally assist attackers by revealing backend information. Different attack types, such as error-based, union-based, and blind SQL Injection, demonstrated that even minor vulnerabilities can result in complete database compromise.

Overall, the project emphasizes that web application security depends on secure query handling, strong authentication mechanisms, and proper input validation. Implementing preventive measures such as prepared statements, parameterized queries, secure session management, and controlled error handling can significantly

Received: 28-04-2026 | Accepted: 02-06-2026 | Published: 11-06-2026

reduce risks. Therefore, adopting secure coding practices and conducting regular vulnerability testing are essential steps for building robust and secure web applications.

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: 28-04-2026 | Accepted: 02-06-2026 | Published: 11-06-2026