IMPLEMENTATION OF PASSWORD HASHING TO ENHANCE WEB-BASED LOGIN SYSTEM SECURITY

M. Syahrudin[1]
[1]Information System, Faculty of Engineering and Computer Science, Islamic University of Indragiri,
Email: msyahrudin128@gmail.com[1],

ABSTRACT

User data security is a critical aspect of web-based system development, particularly for login systems that utilize passwords for authentication. Storing passwords in plaintext poses a security risk should the database be accessed by unauthorized parties. This study aims to implement password hashing in a web-based login system using the Laravel framework to enhance the security of stored user authentication data. The research methodology encompasses requirements analysis, system design, implementation, testing, and results analysis. Implementation involves hashing passwords prior to database storage, while the login process relies on verifying the input against the stored hash value. Testing was conducted using the black-box method across various scenarios, including user registration, database password checks, login attempts with correct and incorrect passwords, and the logout process. The results demonstrate that user passwords were successfully stored as hashes rather than plaintext. Furthermore, the authentication process functioned correctly, as the system was able to verify user-entered passwords against the stored hash values. These findings indicate that implementing password hashing with Laravel is an effective measure for improving the security of user credential storage in web-based login systems.

Keywords: password hashing, web security, login system, Laravel, authentication.

1 INTRODUCTION

The rapid advancement of information technology has driven various organizations, institutions, and companies to utilize web-based systems to support their operations. Web-based systems facilitate easy information access and data management while enabling the rapid and efficient delivery of services. A widely implemented feature in these systems is the login mechanism, which serves to authenticate users and ensure that only those with appropriate access rights can enter the system. Typically, these systems use a combination of a username or email address and a password for authentication; consequently, the security of login credentials is a critical aspect of web-based application development.

Passwords are a key component of the user authentication process. If passwords are stored insecurely, they become vulnerable to theft and misuse by malicious actors. Storing passwords as plain text is a risky practice because unauthorized access to the database could expose all user passwords immediately. Such a scenario can lead to serious security consequences, including account takeovers, information theft, and unauthorized access to private or sensitive data. Furthermore, since users often reuse the same password across multiple services, a password leak in one system could compromise the security of their accounts on other systems.

One method to enhance password storage security is hashing. Hashing involves transforming data into a fixed-length value using a hash algorithm. Unlike encryption—which is reversible using a specific key—hashing is designed as a one-way process, meaning the hash value cannot easily be converted back into the original password. In authentication systems, user passwords are converted into hash values before being stored in the database. When a user logs in, the entered password undergoes the appropriate hashing process and is then verified against the stored hash value. This approach eliminates the need for the system to store the actual user password in the database.

The implementation of password hashing has become increasingly important as threats to web applications continue to evolve. Attacks on information systems can be carried out in

various ways, including database theft, the exploitation of application vulnerabilities, and unauthorized access to user credentials. Consequently, securing login systems requires more than just providing an authentication page; it also necessitates careful attention to how user credentials are processed and stored. Implementing an appropriate hashing mechanism serves as a layer of protection to mitigate risk should authentication data fall into the hands of unauthorized parties.

In web application development, Laravel is a PHP framework that offers a range of features to help developers build applications in a structured and secure manner. Laravel supports authentication mechanisms and provides password hashing capabilities through built-in security components. By leveraging these features, developers can implement hashing without having to manually create hashing algorithms. Furthermore, Laravel supports hashing algorithms specifically designed for password storage, enabling more structured hash generation and password verification within the application.

Although frameworks provide built-in security features, the proper understanding and implementation of password hashing remain essential. Errors in password management—such as storing passwords in plaintext, using unsuitable hashing algorithms, or performing verification incorrectly—can compromise the security of login systems. Therefore, it is necessary to examine and test password hashing implementations in web-based login systems to understand how these mechanisms are applied and how they contribute to the secure storage of user credentials.

This research focuses on implementing password hashing in a web-based login system using the Laravel framework. The implementation involves hashing passwords before storing them in the database and verifying them during user authentication. Through this implementation, the study aims to demonstrate the differences between storing passwords without hashing protection and storing them using hashing mechanisms. Additionally, the research seeks to provide insight into utilizing Laravel's built-in security features to build more secure authentication systems.

Based on the foregoing, password hashing is a crucial step in enhancing the security of web-based login systems. Using Laravel as a development framework facilitates the implementation of this mechanism, as it offers security features that developers can readily utilize. Consequently, this research—titled "Implementation of Password Hashing to Enhance Web-Based Login System Security"—was conducted to implement and analyze password hashing within a Laravel-based login system, aiming to mitigate the risks associated with insecure credential storage and improve the protection of user authentication data.

2 RESEARCH METHOD

This study employs a research and development method focused on implementing password hashing in a web-based login system. The research involved building a login system using the Laravel framework and subsequently applying a hashing mechanism to user passwords to enhance the security of authentication data. The research process was conducted systematically, spanning requirements analysis, system design, implementation, and testing of the developed system.

2.1 Requirement Analysis

The first stage of the research involves analyzing system requirements. This stage identifies the core requirements for the login system, such as user registration, login functionality, user data validation, data storage in the database, and the logout process. In addition to functional requirements, the research also addresses security needs, particularly regarding user password management.

Passwords are not stored as plaintext; instead, they are processed using a hashing mechanism. Consequently, the password data stored in the database is not the actual password entered by the user. This analysis serves as the foundation for determining the system design and the security mechanisms to be implemented.

2.2 Design System

Once the system requirements have been determined, the next stage is system design. The system is designed as a web-based application utilizing Laravel as the primary framework, with a database serving as the storage medium for user information.

During the authentication process, users first register by entering the required data, including a password. This password undergoes a hashing process before being stored in the database. When a user logs in, the system receives the provided email or username and password. The password is not compared directly against the stored data; instead, it is verified using Laravel's built-in mechanisms to match the input password with the stored hash value.

2.3 Implementation

The implementation phase involved building a login system using the Laravel framework. This process encompassed creating registration and login pages, implementing authentication and session management, and establishing a connection between the application and the database.

During registration, user-entered passwords are processed using Laravel's built-in hashing capabilities. The resulting hash value is then stored in the password column of the users table, ensuring that the actual password is not stored directly in the database.

During the login process, the system receives the authentication data provided by the user. Laravel then verifies the entered password against the stored hash value. If the verification is successful, the user logs in and gains access to pages requiring authentication; conversely, if the password does not match, the system rejects the login attempt and notifies the user that the provided authentication data is incorrect.

2.4 Testing

Once the system was successfully implemented, testing was conducted to verify that the login feature and password hashing mechanism functioned according to the design. Black-box testing was employed, evaluating system functions based on inputs and resulting outputs without directly examining the program's code structure.

Tests covered several scenarios, including registration with a valid password, verification of password data stored in the database, logging in with the correct password, logging in with an incorrect password, and logging out of the system. Password data testing was performed to ensure that passwords were not stored in plain text but had been converted into hash values.

Additionally, the password verification process was tested to confirm that users entering the correct password could access the system, while those entering an incorrect password could not authenticate. The results of each test were recorded and compared against the expected outcomes.

2.5 Analysis Result

The test results were used to evaluate the system's success in securely storing passwords and verifying them during login. If the passwords stored in the database were in hashed form and the login process successfully performed verification, the hashing implementation could be considered to have met the research objectives.

Based on these results, the study drew conclusions regarding the implementation of password hashing in a Laravel-based login system and its contribution to enhancing the security of user authentication data storage. This method aims to demonstrate that implementing appropriate security mechanisms during the password storage phase is a crucial aspect of building a more secure web-based login system.

3 RESULT AND DISCUSSION

3.1 System Implementation Results

Research results indicate that a password hashing mechanism can be implemented in a web-based login system using the Laravel framework. This implementation is applied to the user registration and authentication processes. When a user registers, the entered password is

not stored directly in the database; instead, it is first processed using the hashing mechanism provided by Laravel. The following is the source code used to handle the registration process with password hashing implemented.

```php
public function register(Request $request)
{
    $messages = [
        'username.required' => 'Username wajib diisi.',
        'username.unique' => 'Username sudah digunakan. Silakan pilih username lain.',
        'username.max' => 'Username maksimal :max karakter.',
        'password.required' => 'Password wajib diisi.',
        'password.min' => 'Password minimal :min karakter.',
        'password.confirmed' => 'Konfirmasi password tidak cocok.',
    ];

    // Accept lowercase role values from the form (admin, teacher, student)
    $data = $request->validate([
        'username' => ['required', 'string', 'max:255', 'unique:users,username'],
        'password' => ['required', 'string', 'min:8', 'confirmed'],
    ], $messages);

    $user = User::create([
        'username' => $data['username'],
        'password' => Hash::make($data['password']),
    ]);

    RegistrationStatus::create([
        'user_id' => $user->id,
        'status'  => 'Mendaftar',
    ]);

    return redirect()->route('login')->with('success', 'Registrasi berhasil. Silakan login.');
}
```

Fig 1. Source Code Registration

After the hashing process is performed as shown in Figure 1 above, the system stores the resulting hash in the user table within the database. Consequently, the original password entered by the user is not stored directly. When user data is inspected, the value in the password column appears as a random string of characters with a specific format and length, rather than the actual password used by the user.

This implementation demonstrates that the hashing process has been successfully integrated into the system workflow without altering how users interact with the login page. Users continue to enter their passwords as usual, while the system automatically handles password security on the application side.

3.2 Registration Process Results

During the registration process, users are required to enter specific information, such as their name, email, and password. Once the data is submitted, the system validates the provided information. If all data meets the requirements, the password is processed using a hashing function before the user data is stored.

id	email	email_verified_at	username	password
1	admin@example.com	NULL	admin	$2y$12$jNM4Qtxu86s1RUjm5gOGpeBb.y5NHJqpsflwkLSZBDi...
2	teacher1@example.com	NULL	teacher1	$2y$12$OvHy1F9CP4glx.28b3ZaNOVg0E5/uldXdqFTBSKF87J...
3	teacher2@example.com	NULL	teacher2	$2y$12$/GTbAexJT.7rK7qEz3FfqehAUSTYanXXqVDcPqVpHuD...
4	teacher3@example.com	NULL	teacher3	$2y$12$.slPDgGiWk.BNICQPtPllOT7xzMfjhtpMgOMR8FAEJi...
5	teacher4@example.com	NULL	teacher4	$2y$12$MxVYtyCZ7IccEg7TvxLzaOuwit4zbxPEuBeap/PY7QI...
6	teacher5@example.com	NULL	teacher5	$2y$12$kU4i9VanZ3hXURmmfiX8zu6Uf3HCzhogh2.IKWvzc66...
7	teacher6@example.com	NULL	teacher6	$2y$12$wpreC/J88Ye3LKfBjCNZOuvLIJwjurXLntTomHZJ6lF...
8	teacher7@example.com	NULL	teacher7	$2y$12$ND.P4RTye3dxw8LbrqNJKumWaECsNwG3xSz0B0/tPKn...
9	teacher8@example.com	NULL	teacher8	$2y$12$pZvyDQJJsxxYZq94GlqNj.FdkPWbB1aAB4ihIM.5oL1...
10	teacher9@example.com	NULL	teacher9	$2y$12$LflJ9mSOjQNVR.Rws6mm3uzBb4AMg0wyBt3siY/2ZwD...
11	teacher10@example.com	NULL	teacher10	$2y$12$nig150jW.SM1PvdOi7hq0e62VDrDe/GPRX0qFnW.MeS...
12	teacher11@example.com	NULL	teacher11	$2y$12$Inwyk3fYIPnKeHec0/B30OZr5GAUz0MXsY/Fq8wyDQP...
13	teacher12@example.com	NULL	teacher12	$2y$12$v3GZ7apl1o2svBv4/7dVbOPAe1nvoOYvBHJ1qwdhqnG...

Fig 2. Process Results

As part of the research experiment shown in Figure 2 above, when a user enters "password123" as their password, the system does not store that value directly in the database. Instead, the password is converted into a hash value, which is then stored as the user's password data.

This result demonstrates that anyone with access to the database contents cannot immediately determine the user's actual password simply by viewing the password column. This provides an additional layer of protection for authentication data, particularly in the event of a data breach or unauthorized access to the database.

3.3 Login Process Results

```php
public function login(Request $request)
{
    $messages = [
        'username.required' => 'Username wajib diisi.',
        'password.required' => 'Password wajib diisi.',
    ];

    $credentials = $request->validate([
        'username' => ['required', 'string'],
        'password' => ['required', 'string'],
    ], $messages);

    $remember = $request->boolean('remember');

    if (Auth::attempt($credentials, $remember)) {
        // Regenerate session to prevent fixation and store role in session
        $request->session()->regenerate();
        $user = Auth::user();
        if ($user) {
            $request->session()->put('user_id', $user->id);
            $request->session()->put('role', $user->role);
            if ($user->role === 'Student') {
                // Assuming there's a Student model related to User
                $student = Student::where('user_id', $user->id)->first();
                if ($student) {
                    $request->session()->put('student_id', $student->id);
                }
            } elseif($user->role === 'Teacher') {
                // Assuming there's a Teacher model related to User
                $teacher = Teacher::where('user_id', $user->id)->first();
                if ($teacher) {
                    $request->session()->put('teacher_id', $teacher->id);
                }
            } elseif($user->role === 'Candidate') {
                // Additional candidate-specific session handling can go here
            }
        }

        return redirect()->intended('/');
    }

    return back()->withErrors([
        'username' => 'Username atau password salah.',
    ])->onlyInput('username');
}
```

Fig 3. Source Code Login

In the login process shown in Figure 3, the user enters a previously registered email and password. The system then searches for user data based on the provided email. Once the user data is located, the system proceeds to verify the password.

The password entered by the user is not compared against the hash value using a simple text comparison. Instead, the system utilizes Laravel's built-in verification mechanism to check if the password matches the hash value stored in the database.

Test results indicate that when the user enters the correct password, the system successfully recognizes the match and grants access to pages requiring authentication. Conversely, if the entered password is incorrect, the authentication process is rejected, and the user remains on the login page.

This demonstrates that the implementation of hashing does not hinder the authentication process. Even though the stored password has been converted into a hash value, the system remains capable of verifying the password entered by the user.

3.4 Password Security Test Results

Security testing was conducted by examining the stored password data following user registration. This test aimed to ensure that passwords were not stored in plaintext.

The results showed that the password value in the database had been converted into a hash and differed from the original password entered during registration. Thus, the hashing implementation functioned as intended by the research.

Testing also involved evaluating various login scenarios. The results are summarized in Table 1.

Table 1. Testing Results

No	Test Scenario	Expected results	Reults
1	Register with valid data.	User data successfully saved.	Success
2	Password verification against the database	Data is stored in the database in the form of a hash.	Success
3	Log in using the correct password.	The user has successfully logged into the system.	Success
4	Log in using the wrong password	The system denied access.	Success
5	Log in with an unregistered email	The system denied access.	Success
6	Logout	The user logs out of the system.	Success

Based on the test results, all key functions related to authentication operate as designed. Passwords are successfully secured prior to storage, while the login process continues to perform verification correctly.

3.5 Discussion

Implementing password hashing significantly enhances the security of login systems. Before hashing is applied, passwords stored as plain text are immediately exposed if an unauthorized party gains access to the database. This poses a high risk, as password information is a critical component of user authentication data.

Once hashing is implemented, user passwords are no longer stored in their original form; the system stores only the result of the hashing process. This makes the password data much harder to exploit directly in the event of a database breach. However, hashing does not render the system entirely immune to security threats. System security remains dependent on various other factors, such as server security, application configuration, session management, input validation, and protection against web application attacks.

Using Laravel in this study also facilitates the implementation of password hashing. The framework provides built-in features for creating and verifying hashes, eliminating the need for developers to build hashing mechanisms from scratch. This approach reduces the likelihood of implementation errors during the password securing process.

The research findings demonstrate that password hashing serves not merely as a method for transforming password formats but as an integral part of the authentication mechanism that helps protect user credentials. The original password remains known only to the user, while the application stores a hash representation used for verification.

Thus, the study indicates that implementing password hashing in a web-based login system using Laravel improves the security of stored user credentials compared to plaintext storage. Furthermore, this implementation allows the authentication process to function effectively without significantly impacting the user experience.

However, hashing should not be viewed as the sole security mechanism. To achieve a more robust login system, password hashing should be combined with other security practices, such as using HTTPS connections, validating input, properly managing sessions, limiting login attempts, and managing user access rights. By implementing these multiple layers of security, risks to the login system can be minimized, and the protection of user data enhanced.

4 CONCLUSION

Based on the research findings regarding the "Implementation of Password Hashing to Enhance Web-Based Login System Security" using the Laravel framework, it can be concluded that implementing password hashing successfully improves security during the storage of user authentication data. Passwords entered by users during registration are not stored directly as plaintext; instead, they are processed into hash values before being saved to the database. Consequently, the original user password cannot be directly ascertained simply by viewing the data stored in the database.

The implementation of hashing does not interfere with the user authentication process. During login, the system verifies the password entered by the user against the stored hash value. Test results indicate that users entering the correct password successfully log in, while incorrect passwords are rejected by the system. This demonstrates that the hashing mechanism can be effectively implemented in a web-based login system using Laravel.

Utilizing Laravel's built-in hashing features also simplifies the password security process, as developers do not need to create their own hashing algorithms. This implementation helps reduce the risk of errors in password management and provides an additional layer of protection for user credentials in the event of unauthorized database access.

However, password hashing is not the sole factor determining the security of a login system. Application security must still be supported by other mechanisms, such as the use of HTTPS, input validation, session security, login attempt limits, and user access control. Therefore, implementing password hashing serves as a crucial step in building a more secure web-based login system, particularly regarding the protection of user credential data.

REFERENCE

[1] R. Dafi Al Azhar and I. Sholihah Widiati, “Evaluasi Keamanan Penyimpanan Password Menggunakan Algoritma Hash: MD5, SHA-1, dan Bcrypt,” *Pros. Semin. Nas. Teknol. Inf. dan Bisnis*, pp. 1302–1305, 2025, doi: 10.47701/q885nj69.

[2] M Aditya Firmansyah, Qarinah, and Muhlis Tahir, “Integrasi Algoritma SHA-256 dan AES untuk Pengamanan Kredensial dan Data Sensitif pada Sistem Informasi Kepegawaian,” *J. Komput. Teknol. Inf. Sist. Komput.*, vol. 5, no. 1, pp. 462–468, 2026, doi: 10.62712/juktisi.v5i1.1032.

[3] J. Prastyia, G. Agam Alfarizi, S. Romadhon, and F. Pratama, “Pembuatan Website Company Profile Pt. Muhammad Syahri Purnama Menggunakan Metode Hashing,” *JATI (Jurnal Mhs. Tek. Inform.*, vol. 9, no. 1, pp. 1176–1182, 2024, doi: 10.36040/jati.v9i1.12550.

[4] Y. Arafat, T. Hidayat, M. Khozin, K. Kunci -Autentikasi, K. Siber, and S. Login, “Implementasi Penggunaan Bcrypt Dan Passkey Pada Sistem Login Web Di Smk Negeri 1 Xyz,” vol. 11, no. 1, pp. 70–76, 2026.

[5] M. C. Jayanti and D. A. Dermawan, “Implementasi Algoritma Hashing Pada Sistem Tracking Surat Himpunan Mahasiswa Prodi Dengan RAD (Studi Kasus : Sarjana Terapan Manajemen Informatika),” vol. 15, no. 1, pp. 66–76, 2026.

[6] S. Nasional, I. Fti, N. R. Anggraini, and B. A. Herlambang, “IN-FEST 2026 Implementasi Kriptografi Bcrypt Pada Sistem Informasi Dokumen Manajemen Mutu Berbasis Web IN-FEST 2026,” vol. 2026, pp. 60–68, 2026.

[7] T. Mary and N. Febriyani, “Peningkatan Keamanan Sistem Informasi Berbasis Laravel 12 dengan Rate Limiting dan Role-Based Access Control (RBAC),” *J. Teknol. Dan Sist. Inf. Bisnis*, vol. 7, no. 3, pp. 473–481, 2025, doi: 10.47233/jteksis.v7i3.1976.

[8] Ulil Asyhar, Budi Hartono, and Toni Wijanarko Adi Putra, "Implementasi Algoritma Base64 Pada Sistem Login Menggunakan Json Web Token (Jwt) Untuk Autentikasi Web," *J. Teknol. Inf. Dan Komun.*, vol. 17, no. 1, pp. 1–11, 2026, doi: 10.51903/jtikp.v17i1.1311.

[9] Nugroho Dwi Aji, Tomi Tri Sujaka, Husain, Ondi Asroni, and Kurniadin Abd. Latif, "Analisis Dan Implementasi Algoritma Bcrypt Dengan Affine Cipher Untuk Pengamanan Password Pada Aplikasi Web," *Cyber Secur. dan Forensik Digit.*, vol. 8, no. 1, pp. 1–9, 2025, doi: 10.14421/csecurity.2025.8.1.5076.

[10] R. Greslyana, S. Jatmika, and S. Arifin, "Development Of Adaptive Login Security Using A Combination Of AES And Bcrypt On The Laravel Framework," *ICoBITS*, vol. 1, pp. 452–460, 2026, doi: 10.32664/icobits.v1.31.

[11] N. Ramadani and E. maiyana, "Perancangan dan Implementasi Sistem Informasi Perpustakaan BerbasisWeb Menggunakan Framework Laravel untuk Optimalisasi PengelolaanBuku dan Transaksi Peminjaman," *Neotech J. Inf. Technol. Innov.*, vol. 1, no. 01, pp. 01–17, 2026, [Online]. Available: https://journal.torkataresearch.org/index.php/neotech/article/view/10

[12] I. Q. Lubis and A. Zulherry, "Implementasi keamanan website dari serangan Cross Site Request Forgery menggunakan algoritma HMAC-SHA256 pada framework Laravel," *Hello World J. Ilmu Komput.*, vol. 5, no. 1, pp. 47–58, 2026.

[13] G. Maulana and U. I. Indragiri, "Analisis Keamanan Sistem Autentikasi Login Berbasis Framework Laravel," vol. 1, no. 1, pp. 18–25, 2026.

[14] A. Fitriyani, H. Lubis, Y. Yaddarabullah, and R. Sari, "Keamanan Sistem Login Menggunakan Multifactor Authentication dan Algoritma Hashing," *J. Inf. Syst. Informatics Comput.*, vol. 9, no. 2, p. 263, 2025, doi: 10.52362/jisicom.v9i2.2137.

[15] D. Febrian *et al.*, "Implementation of Bcrypt Algorithm on Website-Based Hashing Generator Using Laravel Framework," *J. Inf. Syst. Informatics Comput.*, vol. 7, no. 2, p. 199, 2023, doi: 10.52362/jisicom.v7i2.1130.