

Development of a Sensitive-Data Monitoring System for Witness and Victim Protection: Role-Based Security Integration and Usability Evaluation

Erikson Putra Perdana Lumban Tobing*
Informatics Engineering

Faculty of Computer Science, Universitas Katolik Santo
Thomas
Medan, Indonesia
erikson28@gmail.com

Zakarias Sitummorang
Data Science

Faculty of Computer Science, Universitas Katolik Santo
Thomas
Medan, Indonesia
zakarias65@ust.ac.id

Lamhot Sitorus

Informatics Engineering
Faculty of Computer Science, Universitas Katolik Santo
Thomas
Medan, Indonesia
lamhot68@yahoo.com

Abstract—Data on applications, case reviews, and protection services for witnesses and victims at the Medan Representative Office of the Witness and Victim Protection Agency (LPSK) had previously been managed through a legacy application built without a modern framework, which constrained data-handling efficiency, feature coverage, and the security controls required for information tied directly to personal safety. Existing literature widely discusses public-service monitoring in general terms but rarely demonstrates concrete, tested technical safeguards for highly sensitive data such as the identities of protected individuals. This study develops a web-based monitoring platform on the Laravel framework that consolidates the application, review, service, and financial workflows into a single system, protected by layered security consisting of Role-Based Access Control (RBAC), role-based data masking, Bcrypt password hashing, CSRF protection, rate limiting, UUID-based routing combined with policy-based authorization, and audit trails. A Research and Development (R&D) approach following the 4D model (Define, Design, Develop, Disseminate) was used. A requirements-to-feature traceability analysis was performed to confirm that every identified requirement is addressed by a specific module and, where applicable, a specific security mechanism. Black-box functional testing across 35 scenarios and security testing across 10 scenarios each achieved a 100% success rate, and each security scenario is reported individually with its attack/input condition, expected result, and actual result. A User Acceptance Test with 5 respondents at LPSK Medan produced a mean System Usability Scale (SUS) score of 93.6 (Grade A, Excellent), which is interpreted strictly as perceived usability within this limited, single-office sample rather than as evidence of broader user acceptance. The principal contribution of this work is the integrated combination of RBAC, role-based data masking, and audit trails, implemented and empirically validated within a highly sensitive government information system, and positioned against related Laravel-based and RBAC-based studies through a state-of-the-art comparison. System scalability was not verified

through load testing, and the security evaluation did not include independent penetration testing, both of which remain limitations and directions for future work.

Keywords—monitoring system, witness and victim protection, information system security, role-based access control, usability



© 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution ShareAlike (CC BY SA) license (<https://creativecommons.org/licenses/by-sa/4.0/>).

I. INTRODUCTION

The digitalization of public services has become an urgent requirement for state institutions, including those that manage high-risk data such as the Witness and Victim Protection Agency (LPSK). LPSK is an independent state institution established under Law No. 13 of 2006 concerning the Protection of Witnesses and Victims[1], as amended by Law No. 31 of 2014 [2], with the authority to provide protection, assistance, compensation, and restitution to witnesses and victims of criminal offenses. The obligation to safeguard the data it manages is further reinforced by Law No. 27 of 2022 concerning Personal Data Protection, which requires data controllers to implement adequate technical and administrative measures, including encryption and access restricted to authorized parties [3]. Because the data handled by LPSK covers the identity, location, and case history of protected persons, any leakage or manipulation of that data threatens personal safety rather than merely causing administrative loss.

At the Medan Representative Office, the process of tracking applications, case reviews, protection services, and financial

administration previously relied on a system built without a modern framework, leaving the code base difficult to maintain and standard security mechanisms such as CSRF protection, input validation, and data encryption largely absent. This situation produced three core problems: low data-management efficiency, limited feature coverage, and insufficient scalability to accommodate growing data volumes and user numbers.

Prior research on public-service monitoring systems shows that digital monitoring improves efficiency and transparency, although implementation often falls short due to weak data integration and the absence of real-time dashboards [4]. A systematic review of indicators for evaluating digital public services further emphasizes that data security and user satisfaction are essential dimensions that must be measured rather than assumed [5]. Studies on the role of digitalization in Indonesian government services also point out that limited staff competence, weak infrastructure, and inadequate data-security standards remain common obstacles, particularly for services that handle sensitive data [6].

Several studies have demonstrated the suitability of the Laravel framework for building monitoring systems in other domains, such as construction project monitoring and evaluation [7] and financial administration archiving for a government statistics agency [8], with Laravel reported to outperform native PHP in terms of code structure and built-in security [9]. However, these studies focus primarily on monitoring functionality and do not explicitly test layered security mechanisms for extremely sensitive data such as the identity of protected individuals.

These four groups of studies share a common pattern: monitoring and public-service digitalization are examined at the level of requirements, indicators, or basic functionality, but are rarely accompanied by a concrete design and technical proof of the security mechanisms actually implemented and tested. In a context involving the safety of individuals, such as witness and victim protection, this omission constitutes a relevant research gap, because the security needs of such a domain cannot be addressed through general web-application security principles alone; they require additional mechanisms that specifically shield the identity of protected persons from unauthorized internal access.

This study proposes a web-based monitoring system built on the Laravel framework that integrates the application, review, service, and financial protection processes into a single platform, supported by a defense-in-depth security approach comprising Role-Based Access Control (RBAC), role-based data masking to conceal the identity of protected persons, Bcrypt password hashing, CSRF protection, rate limiting, UUID-based identifiers combined with policy-based authorization to prevent Insecure Direct Object Reference (IDOR), and an audit trail built on the Spatie Activity Log package. The system follows a Repository and Service architectural pattern to support long-term maintainability.

The main contribution of this study is not the use of Laravel per se - several comparable studies have already applied Laravel-based RBAC in general administrative domains [10],

[11], or in broader centralized authorization and database-security domains [12], [13] - but rather the integrated combination of RBAC, role-based data masking, and audit trails, implemented and empirically tested within a highly sensitive government information system, a combination that has not been extensively addressed in either the public-service monitoring literature or the RBAC literature cited above. This positioning is examined directly in Section III.E through a state-of-the-art comparison (Table V). Based on this background, the study aims to: (1) design and develop an integrated witness and victim protection monitoring system, with explicit traceability from identified requirements to implemented features; (2) implement and test the effectiveness of its layered security mechanisms, including the specific authorization mechanism used to prevent IDOR; and (3) evaluate user acceptance through functional, security, and usability testing, with a usability result interpreted within the scope and limits of the evaluation sample.

II. METHOD

This study applies a Research and Development (R&D) approach using the 4D model developed by Thiagarajan, Semmel, and Semmel [14], consisting of four stages: Define, Design, Develop, and Disseminate. This model was selected because it provides a systematic progression from needs analysis to testing and handover of the system to end users.

In the Define stage, the condition of the existing monitoring system at LPSK Medan was analyzed through observation, interviews with staff and technical personnel, and a literature review, in order to identify functional and security requirements. The Design stage covered database design, process modeling using the Unified Modeling Language (Use Case Diagram, Activity Diagram, Class Diagram), and interface design. The Develop stage implemented the system using the Laravel framework, the PHP programming language, and a MySQL 8.0.30 database, following a Repository Pattern and Service Pattern to separate data-access logic from business rules. The modules built include the dashboard, application intake, case review, protected-person services, and finance (budget and expenditure).

Security mechanisms were implemented during the Develop stage, comprising: (1) Role-Based Access Control, implemented with the Spatie Permission package to restrict access for four user roles (SuperAdmin, Admin, Treasurer, Staff) in line with the principles of separation of duties and need-to-know, consistent with earlier applications of Laravel-based RBAC in other administrative systems [10], [11]; (2) role-based data masking through an Accessor on the Eloquent Model to conceal the name of an applicant or protected person from roles without a direct need to know it; (3) Bcrypt password hashing with a cost factor of 12; (4) CSRF protection using Laravel's built-in token combined with the SameSite=Strict cookie attribute, consistent with common CSRF mitigation practice for web applications [15]; (5) rate limiting on the login endpoint and the dashboard API; (6) UUIDs used for route model binding in place of sequential numeric identifiers, combined with per-model Policy classes invoked through

Laravel's authorize() call at every state-changing controller action, jointly preventing IDOR by removing identifier predictability and enforcing a role-based authorization decision on the specific resource being accessed, following IDOR-prevention techniques for website-based applications [16]; and (7) automatic audit trails through an Observer pattern on Eloquent models, recording every create, update, and delete operation to an audit_logs table.

The Disseminate stage in this study is defined as the introduction and handover of the system to end users within the LPSK Medan Representative Office as the organizational unit under study; dissemination to other LPSK representative offices lies outside the scope of this research.

System evaluation used three testing methods. First, Black Box Testing covering 35 scenarios across the authentication, user management, staff management, application, review, protected-person services, finance, dashboard, system configuration, and audit-log modules, focused on the correspondence between input and expected output without inspecting the underlying code. Second, scenario-based security testing covering 10 security components (data masking, role-based masking bypass, Bcrypt hashing, CSRF protection, login rate limiting, API rate limiting, IDOR prevention, RBAC, SQL injection prevention, and cross-site scripting prevention), each documented individually with its attack/input condition, expected result, and actual result (TABLE 3) rather than only an aggregate success rate. Third, a User Acceptance Test (UAT) involving 5 respondents at the LPSK Medan Representative Office – the Head of the Representative Office, the Head of General Affairs and Administration, an administrative staff member, a treasurer, and a staff member – using a 10-item, five-point Likert System Usability Scale (SUS) questionnaire. Following Brooke's scoring method [17], the contribution score for odd-numbered items is calculated as (response value - 1) and for even-numbered items as (5 - response value); the summed contribution scores are multiplied by 2.5 to yield a SUS score on a 0–100 scale, which is then averaged and interpreted using the adjective rating scale of Bangor, Kortum, and Miller [18]. As detailed in Section III.D, the SUS instrument measures perceived usability and is interpreted here only as such, not as a general measure of user acceptance.

III. RESULTS AND DISCUSSION

A. Requirements Analysis

Analysis of the legacy system showed that the recording of applications, risk assessments, and case-progress reports was still handled through separate administrative documents, so information was not centralized and both searching and validating data took longer than necessary. The key requirements identified were integration of data across all service stages (from application intake to case closure) within a single platform; a dashboard that presents a real-time summary of active cases and threat levels for management; and security requirements covering role-based access restriction and the

concealment of protected-person identities from internal parties without a direct need to know.

To make explicit how each identified requirement was translated into a concrete system feature - rather than remaining at a descriptive level - TABLE 1 maps every requirement to the module(s) that implement it and, where the requirement has a security dimension, to the corresponding security mechanism(s) verified in Section III.D. This mapping shows that each requirement identified in the Define stage is addressed by at least one concrete module and, for security-related requirements, by a specific mechanism that is independently exercised in the testing described in Section III.D - closing the gap between the descriptive requirements analysis and the developed system.

TABLE 1. REQUIREMENTS-TO-FEATURE AND SECURITY MECHANISM MAPPING

Identified Requirement	Implementing Module / Feature	Security Mechanism(s) Addressing It
R1. Integrate application, review, service, and finance data within one platform	Application, Review, Protected-Person Services, and Finance modules, linked through a shared case record	Audit trail (Observer pattern) logs all cross-module changes; RBAC restricts module-level access
R2. Real-time dashboard of active cases and threat levels for management	Dashboard module with role-configurable statistics and charts	RBAC controls chart/menu visibility per role; API rate limiting protects the dashboard endpoint
R3. Role-based access restriction (separation of duties, need-to-know)	Four-role structure (SuperAdmin, Admin, Treasurer, Staff) enforced at menu, route, and resource level	RBAC (Spatie Permission) combined with per-model Policy-based authorization
R4. Concealment of protected-person identity from internal staff without a direct need to know	Applicant/protected-person name fields in the Application, Review, and Service modules	Role-based data masking (Eloquent Accessor)
R5. Protection of resource identifiers from guessing or enumeration	Record-level routes for Application, Review, and Service records (e.g., /permohonan/{uuid})	UUID-based route model binding combined with Policy authorization (IDOR prevention, see Section III.C)
R6. Prevent unrecoverable data loss and preserve traceability of changes	Soft delete on all core models (Application, Review, Service, Budget, Expenditure, Users)	Automatic audit trail (audit_logs table via Observer pattern)
R7. Secure authentication and protection against credential attacks	Login module	Bcrypt password hashing (cost factor 12); login rate limiting

Identified Requirement	Implementing Module / Feature	Security Mechanism(s) Addressing It
R8. Protection against common web-application attacks	All data-entry forms across modules	CSRF token + SameSite=Strict cookies; Eloquent parameter binding (SQL injection); Blade auto-escaping (XSS)

B. System Development

The developed system consists of five main modules. The dashboard module presents summary statistics of active cases and charts whose visibility can be configured per user role. The application module records protection requests along with a registration number issued automatically by the system. The review module links application data with the review team's eligibility analysis, including the recorded outcome of the decision meeting (accepted, rejected, or emergency protection). The protected-person services module records service realization (such as legal assistance and restitution), with the end date of each service calculated automatically using the Carbon helper library. The finance module manages budgets and expenditures with composite-key validation (budget code and year) to prevent duplicate entries, along with a dynamically calculated remaining budget.

All modules implement soft deletion, so historical data is not permanently lost due to user operational error, and every CRUD operation on the core models (Application, Review, Service, Budget, Expenditure, Users) is automatically recorded in the audit log through the Observer pattern, independent of whether developers remember to add logging in each controller.

C. Security Implementation

The primary threat in the witness and victim protection domain is the leakage of a protected person's identity to unauthorized parties, including from within the system's own user base. To address this, role-based data masking is implemented through an Eloquent Accessor that converts a name into an initialized format (e.g., "E*n T*g") for the Staff and Treasurer roles, while the SuperAdmin and Admin roles retain full visibility for high-level coordination needs. This mechanism operates at the presentation layer without altering the underlying data in the database.

The threat of request forgery (CSRF) is addressed through Laravel's built-in token verification combined with the SameSite=Strict cookie attribute, so that cross-origin requests are rejected before reaching business logic [15]. The threat of credential compromise is addressed through Bcrypt password hashing, which is deliberately slow to make brute-force attacks computationally impractical, supplemented by rate limiting on the login endpoint to restrict the number of attempts from a single IP address.

The threat of URL parameter manipulation (IDOR) is mitigated through two complementary mechanisms, rather than through UUIDs alone. First, resource-identifier obscurity:

UUIDs are used as the route model binding key in place of sequential numeric IDs, configured via the `getRouteKeyName()` method on the Eloquent model (e.g., `/permohonan/{uuid}`). This removes the predictability that would otherwise let an attacker enumerate resources by incrementing an integer parameter, and it is the mechanism verified by the numeric-ID manipulation scenario in TABLE 3(No. 7), where a manipulated integer parameter fails to resolve to any resource and returns HTTP 404. On its own, however, this mechanism only makes identifiers hard to guess; it does not decide whether a given authenticated user is entitled to act on a resource whose identifier they do possess. Second, and central to actual authorization, policy-based authorization: every state-changing controller action (view, update, delete) explicitly calls Laravel's `authorize()` method, which is bound to a corresponding Policy class for the resource (e.g., `PermohonanPolicy`, `PenelaahanPolicy`, `LayananPolicy`). Each Policy method checks the authenticated user's role and permissions against the specific model instance being requested before the operation proceeds. This means that even if a user obtains a syntactically valid UUID for a record outside the scope their role is permitted to access, the Policy check denies the request with an HTTP 403 response before any data is read or modified - independent of whether the UUID itself is valid. UUID-based routing therefore addresses identifier predictability, while Policy-based authorization enforces the access-control decision itself; it is this combination, and not the UUID in isolation, that prevents IDOR in this system. The authorization layer shares the same underlying Policy/Gate mechanism exercised in the RBAC test scenario (TABLE 3, No. 8), which confirms that a role without the required permission is denied access with HTTP 403 regardless of the identifier supplied in the request.

The threats of query and script injection (SQL Injection and XSS) are addressed through parameter binding in Eloquent/Query Builder and automatic HTML escaping in the Blade templating engine, respectively. All of these mechanisms were designed as a defense-in-depth strategy and verified through the testing scenarios presented in the following subsection.

D. Testing Results

All 35 Black Box Testing scenarios covering the authentication, user/staff management, application, review, service, finance, dashboard, configuration, and audit-log modules ran according to the expected results. TABLE 2 reports the aggregate outcome of the 10 security-testing scenarios; because an aggregate success rate alone does not show what each mechanism was actually tested against, TABLE 3 below reports every scenario individually, with its attack/input condition, expected result, and observed actual result.

The mean SUS score of 93.6 which can be seen in TABLE 4 lies well above the commonly used acceptability threshold of 68 [18], placing the system in Grade A, the Excellent category. This result should be interpreted specifically as a measure of

perceived usability - how easy and satisfying the interface was for the users who completed the assigned task scenarios - rather than as a measure of overall user acceptance. The SUS instrument does not evaluate perceived usefulness, intention to continue using the system, fit with organizational workflow, or long-term adoption, which are the concerns of separate acceptance models (e.g., TAM); usability and acceptance are therefore treated here as distinct constructs, with this study providing evidence only for the former. In addition, because the evaluation involved only five respondents, all drawn from a single LPSK representative office (Medan), the score of 93.6 demonstrates a very positive usability experience among the participating users of this pilot deployment, but it should not be generalized to the broader population of LPSK staff across other representative offices, nor treated as a validated measure of organization-wide acceptance. The five respondents -

representing the SuperAdmin, Admin, Treasurer, and Staff roles - successfully completed most of the assigned task scenarios, and the SUS score reflects their perceived ease of use within that limited sample; a larger, multi-site sample together with an acceptance-focused instrument is required before broader claims can be made, as noted in the Research Limitations.

TABLE 2. SUMMARY OF FUNCTIONAL AND SECURITY TESTING RESULTS

Test Type	Scenarios	Passed	Success Rate
Functional (Black Box)	35	35	100%
Security	10	10	100%
Total	45	45	100%

TABLE 3. DETAILED SECURITY TESTING SCENARIOS

No.	Security Component	Attack / Input Condition	Expected Result	Actual Result
1	Role-Based Data Masking	Access the application/service list while authenticated with the Staff or Treasurer role.	Applicant/protected-person name is dynamically masked into an initialed, asterisked format (e.g., "E*n T*g").	The system displayed masked names for the Staff/Treasurer role in the browser. (Passed)
2	Data Masking Bypass (Full Visibility)	Access the application/service list while authenticated with the SuperAdmin or Admin role.	Applicant/protected-person name is displayed in full, without masking.	Full names were displayed correctly to support high-level coordination. (Passed)
3	Bcrypt Password Hashing	Inspect the password column of the users table directly in the database after registration.	Password is stored as an irreversible hash using the Bcrypt algorithm (cost factor 12).	Password was stored as a random hash string (\$2y\$12\$...) and could not be read manually. (Passed)
4	CSRF Protection	Submit a POST request (e.g., save application) with the @csrf token removed or manipulated.	System rejects the request and returns HTTP 419 (Page Expired) due to token-validation failure.	The request was rejected outright by the security middleware with HTTP 419. (Passed)
5	Login Rate Limiting	Submit more than 5 consecutive login attempts with incorrect credentials within 1 minute from the same IP.	System blocks the 6th and subsequent attempts for 1 minute and returns a "too many attempts" warning.	The IP was temporarily blocked; the brute-force login attempt was stopped and a lockout state was triggered. (Passed)
6	API Rate Limiting	Send more than 60 consecutive requests to the dashboard chart-data API endpoint within 1 minute.	System blocks subsequent requests and returns HTTP 429 (Too Many Requests).	The 61st request was blocked immediately with HTTP 429, protecting the database from overload. (Passed)
7	IDOR Prevention (identifier manipulation)	Access an application/service detail page by manipulating the URL to a sequential numeric ID (e.g., /permohonan/1) instead of the UUID.	Routing resolves exclusively through UUID model binding; a numeric parameter matches no bound resource.	The manipulated numeric ID failed completely; the system returned HTTP 404 because routing is bound exclusively to the UUID parameter. (Passed)
8	Role-Based Access Control (RBAC – Spatie)	Attempt to access the User & Access Management menu while authenticated with the Staff role.	System blocks the route through the authorization middleware/Policy and returns HTTP 403 Forbidden.	The Staff-role user was denied access to the sensitive page and automatically redirected to the 403 error view. (Passed)
9	SQL Injection Prevention	Enter a malicious SQL payload into a data-filter query field (e.g., ' OR 1=1 --).	Query Builder/Eloquent automatically sanitizes the parameter (parameter binding), treating the input as a literal string.	The malicious query was not executed; it was treated as literal text, so the database remained safe. (Passed)
10	Cross-Site Scripting (XSS) Prevention	Enter an HTML/script tag into an application form text field (e.g., <script>alert('hack')</script>).	Blade templating engine automatically performs HTML escaping using {{ }} syntax when rendering data.	The malicious script was rendered as plain text on the page and was never executed by the browser. (Passed)

TABLE 4. USER ACCEPTANCE TEST (UAT/SUS) RESULTS

Test	Respondents	Method/Scale	Result
UAT	5	SUS	Mean 93.6 (Grade A, Excellent)

E. Discussion

The 100% functional success rate is consistent with the findings of [4], that a digital monitoring system can operate in a structured manner when supported by adequate data integration; this study extends that finding by showing that integration involves not only cross-module data recording but also automated audit trailing that operates without manual developer intervention in each controller - an aspect that received little attention in that comparative study or in other Laravel-based monitoring studies that concentrate on basic functionality [7], [8].

On the security dimension, the testing results support the argument of [6] that data security and access management are critical requirements for services handling sensitive data. To make this claimed contribution concrete rather than descriptive, TABLE 5 positions this study against the Laravel-based and RBAC-based studies cited in Section I, in terms of the specific mechanisms each reports.

TABLE 5. COMPARISON WITH RELATED LARAVEL-BASED AND RBAC-BASED STUDIES

Study	Application Domain	RBAC	Role-Based Data Masking	Audit Trail	IDOR Protection	Formal Security Testing
Prasetyo & Cuhenda [7] (2025)	Construction project monitoring & evaluation	–	–	–	–	–
Supriatmaja et al. [8] (2024)	Government financial administration archiving	–	–	–	–	–
Mary & Febriyani [10] (2025)	General Laravel 12 administrative system	✓ (+ rate limiting)	–	–	–	Partial
Yuricha & Phan [11] (2023)	Cloud-based supply chain management	✓	–	–	–	–
Prasetya & Manongga [12] (2024)	Centralized authorization (Flask)	✓	–	–	–	–

Study	Application Domain	RBAC	Role-Based Data Masking	Audit Trail	IDOR Protection	Formal Security Testing
Sahyudhi & Susanto [13] (2025)	ERP database security	✓	–	–	–	–
This study (2026)	Witness and victim protection (government)	✓	✓	✓	✓ (UUID + Policy)	✓ (10 scenarios, TABLE 3)

✓ indicates the mechanism was reported as implemented and evaluated in the cited study; “–” indicates it was not reported. The assessment is based strictly on the mechanisms and evaluation methods described in the respective publications, as summarized in Section I. As Table V shows, the reviewed Laravel-based RBAC studies each report at most a single access-control mechanism [10], [11], [12], [13], and the reviewed Laravel-based monitoring studies do not report security testing at all [7], [8]. This study presents a more complete combination of technical mechanisms (RBAC, role-based data masking, audit trail, rate limiting, UUID-based routing with policy authorization), each verified through its own testing scenario (TABLE 3), so that the security claim is supported by scenario-level functional evidence rather than resting only at the requirements or design level.

Against the digital-service effectiveness indicators formulated by [5] - such as service speed, data security, and user satisfaction - this study provides scenario-level evidence for the data-security indicator through the security-testing results in TABLE 3, and evidence for the user-satisfaction indicator through the SUS score of 93.6, understood within the usability-specific and single-site scope discussed above. The contribution of this study to the Information Systems/Software Engineering field therefore lies in offering a replicable pattern of RBAC, data masking, and audit-trail integration for other government information systems that manage high-risk personal data, rather than simply applying the Laravel framework as a development tool.

It should be emphasized that the 100% success rate in security testing cannot be interpreted as an absolute security guarantee. The testing performed was functional in nature - verifying that each security mechanism operates as designed for the defined scenarios in TABLE 3 - and did not include independent penetration testing by parties outside the research team. Likewise, the Repository and Service Pattern architecture was designed to support maintainability and future development, but the scalability claim has not been empirically validated through load testing or stress testing under data and concurrent-user volumes representative of a production environment.

IV. RESEARCH LIMITATIONS

This study has four limitations that should be stated explicitly. First, testing was conducted within the internal scope of the LPSK Medan Representative Office, without integration with the LPSK head-office system or external cross-agency systems (police, prosecutors, and related institutions), so system interoperability could not be evaluated. Second, the scalability claim is currently supported only by the architectural approach (Repository and Service Pattern) and has not been tested through load or stress testing under production-scale data volumes. Third, security testing was functional in nature and did not include independent penetration testing. Fourth, the UAT was conducted with a small, single-site sample ($n = 5$, LPSK Medan only); accordingly, the SUS score of 93.6 is reported strictly as evidence of perceived usability among these participants and not as a generalizable measure of user acceptance across LPSK's broader user population, and it should be corroborated by a larger, multi-site evaluation using an acceptance-specific instrument in future work.

V. CONCLUSION

This study successfully developed a Laravel-based monitoring system for witness and victim protection that integrates the application, review, service, and finance processes into a single platform, supported by layered security consisting of RBAC, role-based data masking, Bcrypt password hashing, CSRF protection, rate limiting, UUID-based routing combined with policy-based authorization, and audit trails. A requirements-to-feature mapping confirms that every identified requirement is addressed by a specific module and, where applicable, a specific security mechanism. Functional testing across 35 scenarios and security testing across 10 individually documented scenarios each achieved a 100% success rate, while user acceptance testing with 5 respondents at a single representative office produced a mean SUS score of 93.6 (Grade A, Excellent), interpreted here as perceived usability rather than generalized user acceptance. Compared with related Laravel-based and RBAC-based studies, this system is distinguished by its combination of RBAC, role-based data masking, audit trailing, and policy-enforced IDOR prevention, each independently verified. The main contribution of this study is a role-based security integration pattern combined with audit trailing, specifically designed to protect the confidentiality of identities within a highly sensitive government information system. Validating scalability through load testing, conducting independent third-party penetration testing, and extending the usability evaluation to a larger, multi-site sample are recommended as directions for future research.

VI. REFERENCES

- [1] P. R. Indonesia, *Undang-Undang Republik Indonesia Nomor 13 Tahun 2006 tentang Perlindungan Saksi dan Korban*. 2006.
- [2] P. R. Indonesia, *Undang-Undang Republik Indonesia Nomor 31 Tahun 2014 tentang Perubahan atas Undang-Undang Nomor 13 Tahun 2006 tentang Perlindungan Saksi dan Korban*. 2014.
- [3] P. R. Indonesia, *Undang-Undang Republik Indonesia Nomor 27 Tahun 2022 tentang Pelindungan Data Pribadi*. 2022.
- [4] E. Lase and H. Lase, "Implementasi sistem monitoring dalam meningkatkan pelayanan publik pada Dinas SPMDP2A Kabupaten Nias," *JMPIS*, vol. 5, no. 4, 2024, doi: 10.38035/jmpis.v5i4.
- [5] G. V. Pedrosa, R. A. D. Kosloski, V. G. de Menezes, G. Y. Iwama, W. C. M. P. da Silva, and R. M. da C. Figueiredo, "A systematic review of indicators for evaluating the effectiveness of digital public services," *Information*, vol. 11, no. 10, pp. 1–14, 2020, doi: 10.3390/info11100472.
- [6] Syamsiar, "The role of digitalization in enhancing public service effectiveness in Indonesia," *J. Ilm. Ilmu Adm. Publik J. Pemikir. dan Penelit. Adm. Publik*, vol. 13, pp. 465–472, 2024.
- [7] R. A. Prasetyo and C. Cuhenda, "Perancangan sistem monitoring dan evaluasi proyek berbasis Laravel untuk optimalisasi manajemen proyek PT. Inkaba (Perseroda)," *J. Inform. dan Teknol. Pendidik.*, vol. 5, no. 1, pp. 44–53, 2025, doi: 10.59395/jitp.v5i1.126.
- [8] G. A. Supriatmaja, K. Mahendra, I. G. B. Widi Atmaja, D. M. J. Putra, N. K. A. T. Wahyuni, and P. Y. Prawati, "Pengembangan sistem informasi monitoring dan pengarsipan administrasi keuangan menggunakan framework Laravel pada BPS Buleleng," *Infomatek*, vol. 26, no. 2, pp. 239–252, 2024, doi: 10.23969/infomatek.v26i2.19231.
- [9] R. Y. Endra, Y. Aprilinda, Y. Y. Dharmawan, and W. Ramadhan, "Analisis perbandingan bahasa pemrograman PHP Laravel dengan PHP native pada pengembangan website," *Expert J. Manaj. Sist. Inf. dan Teknol.*, vol. 11, no. 1, p. 48, 2021, doi: 10.36448/expert.v11i1.2012.
- [10] 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.
- [11] Y. Yuricha and I. K. Phan, "Penerapan Role Based Access Control dalam sistem supply chain management berbasis cloud," *MALCOM Indones. J. Mach. Learn. Comput. Sci.*, 2023.
- [12] Y. A. Prasetya and D. Manongga, "Role-based access control (RBAC) untuk sistem otorisasi terpusat berbasis Flask studi kasus PT. XYZ," *JUPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.)*, vol. 9, no. 4, pp. 1768–1778, 2024.
- [13] M. Sahyudi and E. R. Susanto, "Analisis implementasi sistem keamanan basis data berbasis Role-Based Access Control (RBAC) pada aplikasi Enterprise Resource Planning," *SATESI J. Sains Teknol. dan Sist. Inf.*, vol. 5, no. 1, pp. 105–116, 2025, doi: 10.54259/satesi.v5i1.3997.
- [14] S. Thiagarajan, D. S. Semmel, and M. I. Semmel, *Instructional development for training teachers of exceptional children: A sourcebook*. Indiana University, Bloomington: Center for Innovation in Teaching the Handicapped, 1974.
- [15] A. Supriyatna, "Analisis kerentanan aplikasi web e-commerce berdasarkan standar OWASP Top 10: Studi kasus pada situs Kopi Lampung Nusantara," *Expert J. Manaj. Sist. Inf. dan Teknol.*, vol. 14, no. 2, 2024, doi: 10.36448/expert.v14i2.4034.
- [16] R. A. Putra, I. A. Kautsar, H. Hindarto, and S. Sumarno, "Detection and prevention of Insecure Direct Object References (IDOR) in website-based applications," *Procedia Eng. Life Sci.*, vol. 4, 2023.
- [17] J. Brooke, *SUS: A Quick and Dirty Usability Scale*. 1996.
- [18] A. Bangor, P. Kortum, and J. Miller, "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale," *J. Usability Stud.*, vol. 4, no. 3, pp. 114–123, 2009.

This page is intentionally left blank