MOBILE AND FORENSICS

Vol. 8, No. 2, 2026, pp. 65-76 | DOI: 10.12928/mf.v10i1.15016

ISSN: 2714-6685 (Online)
ISSN: 2656-6257 (Print)

Development of Web Application for Certificate Automation using Cloudflare DNS API and ZeroSSL

Fredian Simanjuntak[a,1,*], Gary Happydinata[a,2], Herman[a,3]

[a] Computer Science, Universitas Internasional Batam, Batam, Indonesia

[1] fredian@uib.ac.id*; [2] 2231152.gary@uib.edu; [3] herman@uib.edu

*Corresponding Author: fredian@uib.ac.id

ARTICLE INFO

ARTICLE HISTORY

Received: Dec 10, 2025
Revised: Mar 6, 2026
Accepted: Mar 24, 2026

KEYWORDS

SSL Automation, Cloudflare DNS API, ZeroSSL, Agile Scrum, Certificate Lifecycle Management

ABSTRACT

Existing SSL/TLS automation tools were largely server-centric and required host-level configuration. These limitations increased complexity in distributed and multi-domain environments. This study proposed a centralized web-based SSL certificate automation system. The system integrated the Cloudflare DNS API and ZeroSSL REST API for certificate lifecycle management. The research adopted the Agile Scrum methodology during system development. An asynchronous queue-based architecture was implemented to support concurrent certificate issuance. The architecture reduced API rate-limit constraints. Automated Domain Validation (DV) was successfully performed through DNS integration. The system centralized certificate storage and monitoring. The interface simplified administrator operations. Configuration errors were reduced during certificate deployment. Operational efficiency was improved for distributed infrastructures. A usability evaluation was conducted with 25 technical practitioners. The evaluation produced a System Usability Scale mean score of 83.5 with a standard deviation of 6.7. The findings indicated excellent user acceptance and system usability. In conclusion, the proposed system effectively automated SSL certificate management through a centralized and usability-oriented approach. The system also minimized operational overhead and dependency on host-level configuration.

1. INTRODUCTION

Secure communication is a fundamental requirement of modern web systems, where SSL/TLS certificates are essential for ensuring data confidentiality, integrity, and trust between clients and servers [1], [2]. As web applications increasingly rely on multiple domains and distributed infrastructure, the task of managing SSL certificate lifecycles has become highly complex. Failures in certificate management, such as expired or misconfigured certificates, can lead to service disruptions and security risks [1]. Notably, while the recommended re-issuance period for standard

certificates is 55–60 days [3], [4], [5]. The complexity of managing short-lived certificates introduces significant operational overhead, particularly in distributed and resource-constrained environments where manual credential management is no longer scalable [6], [7]. Furthermore, research in usable security demonstrates that highly technical security interfaces often lead to user fatigue and misconfiguration, highlighting the critical need for automated, actionable, and user-friendly security dashboards [8], [9].

To address the inefficiencies of manual SSL management, numerous automation solutions have been developed. Widely used tools such as Certbot and acme.sh automate certificate issuance through protocol-level workflows [10]. While technically mature, these tools are primarily designed for server-level operation; they require local installation, command-line access, and per-host configuration [11]. For administrators managing multiple servers or distributed domains, this server-centric approach introduces fragmented certificate storage and repetitive setup tasks [12], [13].

In parallel, research has proposed automated certificate distribution frameworks to emphasize scalability. For example, the Automated SSL/TLS Certificate Distribution System (ASCEDS) centralized certificate generation via the ACME protocol, improving traceability but still requiring manual configuration for new clients [14], [15]. Further advancements have proposed cloud-based microservice architectures utilizing distributed agents [16]. However, these enterprise-grade platforms often introduce high deployment complexity, cost, and infrastructure overhead, making them impractical for small-to-medium-scale deployments or individual administrators.

Despite these advances, existing approaches exhibit a distinct gap: highly technical host-based scripts lack centralized visibility, while enterprise platforms introduce excessive architectural complexity. There remains a need for a lightweight, centralized orchestration approach that abstracts the complexities of Domain Validation (DV) [17] into an accessible interface without requiring host-level software installation.

To bridge this gap, this study proposes a web-based SSL certificate lifecycle automation architecture. By centralizing operations, the system reduces human error, bypasses host-level dependencies, and provides actionable workflow transparency. The main contributions of this research are:

1) Architectural Orchestration: The design and implementation of a centralized web application that integrates the Cloudflare DNS API for automated domain validation and the ZeroSSL REST API for certificate issuance and renewal.
2) Asynchronous Queue Management: The development of a queue-based workflow to handle concurrent bulk certificate requests, ensuring system stability and mitigating external API rate limitations.
3) Usability-Driven Security Engineering: An empirical evaluation of the system using the System Usability Scale (SUS) to demonstrate how centralized, API-driven interfaces reduce operational overhead for web administrators.

2. METHOD

This study adopts the Design Science Research (DSR) paradigm, focusing on the creation and evaluation of an innovative IT artifact to solve practical limitations in SSL certificate lifecycle management. As illustrated in Fig 1, the overarching research flow consists of four primary phases:

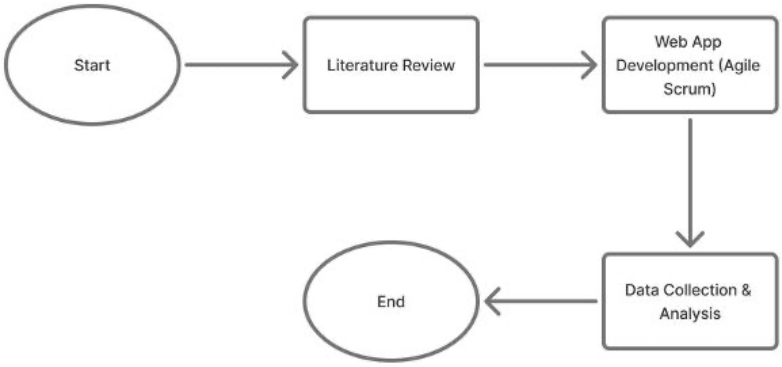

Fig. 1. Research Flow

The proposed methodology for application development is the Agile Scrum framework, which facilitates the structuring and organization of features, enhancements, and modifications. This approach is designed to optimize application performance while preserving usability [18], [19]. Employing this, we plan our sprints and project goals to ensure usability and facilitate further development. The development team consist of Fredian (Product Owner), Gary (Scrum Master and Fullstack Developer).

The framework starts with user interviews, product research, obtaining project requirements, and expectations. With this we can create Product Backlogs that functions as a prioritized compilation of critical features, including domain synchronization, automated certificate generation, bulk issuance, and real-time progress tracking. During the development phase, tasks are selected from the backlog based on priority and feasibility. Each sprint defines a clear and measurable goal, such as completing automated certificate issuance or refining the bulk issuance feature, to maintain focus and consistent progress. A Daily Scrum is conducted to evaluate work progress and identify obstacles, allowing the developer to remain aligned with the sprint objectives and quickly address emerging issues.

The Definition of Done (DoD) ensures that each implemented feature meets specific criteria for completeness and stability. A feature is considered complete when the application can successfully retrieve domains from Cloudflare, issue certificates using ZeroSSL, and handle multiple concurrent issuance tasks without errors. After each sprint, a Sprint Review is conducted to verify that the completed work functions as intended and integrates seamlessly with existing components.

Finally, the Sprint Retrospective evaluates the development process, identifies challenges, and proposes improvements for future iterations. This continuous refinement helps sustain quality and maintain development efficiency in the model. Through this structured and iterative approach, the project ensured that the resulting application was functional, reliable, and adaptable to future enhancements.

2.1. Research Method

This study adopted a Design Science Research (DSR) approach, focusing on the design, implementation, and evaluation of an information system artifact intended to address practical limitations in SSL certificate lifecycle management. The choice of DSR was motivated by the research objective, which aimed not only to develop a functional web application but also to evaluate its effectiveness and usability in solving a real-world operational problem [20]. DSR emphasizes the creation of purposeful artifacts and their evaluation based on utility and performance, making it suitable for research involving system automation and security engineering.

2.2. Research Model

Based on the Design Science Research paradigm, a conceptual research model was developed to structure the relationship between the proposed artifact and its evaluated outcomes. The central construct of this study is a web-based SSL certificate lifecycle automation architecture, designed to centralize certificate issuance, validation, and storage through API-driven orchestration. The artifact incorporates specific design components, including DNS synchronization [21], [22], bulk certificate issuance, asynchronous queue processing, and centralized certificate management. The conceptual research model of the proposed architecture is presented in Fig. 2.

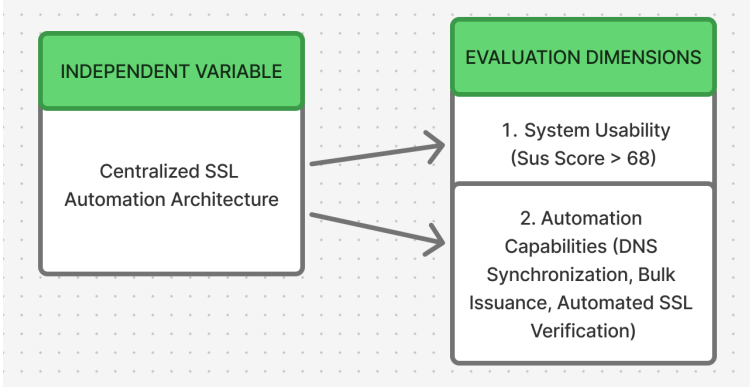

Fig. 2. Conceptual Research Model

The model establishes the Centralized SSL Automation Architecture as the Independent Variable. The evaluation metrics associated with the operational outcomes are summarized in Table 1.

Table 1. Research Variables and Evaluation Metrics

Variable	Metric
Independent Variable	Centralized SSL Automation Architecture implemented onto a web application.
Evaluation Dimensions	System Usability: Measured quantitatively using the System Usability Scale (SUS) to determine user acceptance. The target benchmark is a score > 68.
	Automation Capabilities: Measured by effectiveness of each features including, DNS Synchronization, Bulk Issuance, and Automated SSL Verification.

2.3. System Architecture

To implement the proposed automation capabilities, the system architecture was designed using three main layers:

1) Web Interface Layer: Developed in Vue.js to provide a reactive, cross-device dashboard for domain and lifecycle monitoring.
2) Application and Orchestration Layer: Built on Laravel 10 (PHP 8.2), this backend generates Certificate Signing Requests (CSRs), manages database records, and dispatches asynchronous background jobs.
3) External Service Integration Layer: Interfaces with the Cloudflare DNS API for automated Domain Validation and the ZeroSSL REST API for certificate retrieval.

To support scalability, certificate requests are processed through a queue-based asynchronous mechanism. Each batch (domain(s) submitted on 1 request by multi select, or bulk generate form) is treated as an independent job, enabling concurrent processing while reducing blocking behavior and mitigating external API rate limitations [23]. This design allows bulk certificate issuance and real-time progress tracking.

Private keys and certificates are stored within the application database to enable centralized lifecycle management and cross-device accessibility. While centralized storage improves operational efficiency, it introduces security considerations related to key protection. Access to cryptographic material is restricted through authenticated application control, and future improvements may incorporate encrypted storage or dedicated key management services [24].

2.4. Evaluation Design

This study employed an artifact-centered evaluation design consistent with Design Science Research principles. The proposed SSL certificate automation system was evaluated through functional validation and usability assessment rather than comparative experimental testing. The evaluation focused on two primary dimensions:

1) Functional correctness: successful certificate issuance and lifecycle operations.
2) Usability: measured using the System Usability Scale (SUS).

Participants interacted with the system by performing predefined certificate issuance tasks, including domain synchronization, certificate generation, and status monitoring. No control condition was implemented, as the objective of the study was to assess the artifact's operational feasibility and user acceptance rather than to conduct comparative performance benchmarking.

2.5. Research Instrument and Sampling

This study employed a purposive non-probability sampling technique to recruit participants with technical expertise relevant to web development and infrastructure management. The target population consisted of individuals who had prior experience with domain configuration, hosting environments, or SSL/TLS certificate management, ensuring that respondents possessed sufficient contextual understanding to evaluate the proposed system effectively.

A total of 25 participants (N = 25) were involved in the usability evaluation. The sample included 8 DevOps professionals (32%), 10 software developers (40%) specializing in frontend, backend, and full-stack development, and 7 Information Systems students (28%) with academic exposure to web

technologies. This composition represents a practitioner-oriented sample while still incorporating perspectives from pre-professional users.

The inclusion criteria required participants to: (1) have experience in web development or infrastructure-related activities, (2) possess basic knowledge of domain or SSL/TLS certificate configuration processes, and (3) be capable of interacting with web-based administrative platforms. These criteria ensured that participant responses were based on relevant technical understanding rather than general end-user perceptions.

2.6. Data Collection and Analysis

Data collection was conducted in two separate phases. The first phase focused on system performance testing as a form of technical verification, while the second phase involved user acceptance testing to provide empirical validation of the proposed system.

2.7. Automation Capability Testing

To evaluate the functional reliability and operational efficiency of the proposed system, an artifact-centered performance profiling approach was adopted instead of comparative experimental testing. The system was tested under varying workloads by generating live SSL certificates for actual domains. The testing scenarios involved batch submissions ranging from small-scale requests (1–5 domains) to the maximum supported concurrent workload of 100 domains per request.

During the testing process, the system's background queue workers were monitored to assess performance based on the three operational metrics defined in the conceptual framework:

1) DNS Synchronization: The capability of the backend system to automatically interact with the Cloudflare API for creating and removing CNAME records without manual intervention.
2) Automated SSL Verification: The ability of the system to successfully complete ZeroSSL domain validation procedures, including the effectiveness of the implemented three-retry queue mechanism in handling delayed API responses.
3) Bulk Issuance: The capacity of the system to process up to 100 concurrent asynchronous jobs without encountering server timeouts or external API rate-limit errors.

2.8. User Acceptance and Usability Testing

Application testing and user feedback was conducted using a usability testing instrument. Respondents were asked to assess the proposed system by answering ten usability statements using a five-point Likert scale ranging from 1 (strongly disagree) to 5 (strongly agree), as presented in Table 2.

Table 2. Usability Evaluation Questionnaire Items

No	Questions	Disagree		Neutral		Agree
		1	2	3	4	5
1	I think that I would like to use this system frequently.					
2	I found the system unnecessarily complex.					
3	I thought the system was easy to use.					
4	I think that I would need the support of a technical person to be able to use this system.					
5	I found the various functions in this system were well integrated.					
6	I thought there was too much inconsistency in this system.					
7	I would imagine that most people would learn to use this system very quickly.					
8	I found the system very cumbersome to use.					
9	I felt very confident using the system.					
10	I needed to learn a lot of things before I could get going with this system.					

Statistical analyses for the quantitative usability evaluation were performed using Microsoft Excel. Prior to analysis, the collected responses underwent a data cleaning process to identify and remove anomalous response patterns, including straight-lining behavior across alternating positive and negative questionnaire items. The validity of the ten questionnaire items was examined using Pearson's Product-Moment Correlation by comparing each item score with the overall questionnaire score. In addition, the internal consistency of the SUS instrument was evaluated using Cronbach's Alpha reliability analysis. A minimum Cronbach's Alpha coefficient of 0.70 was used as the acceptance threshold to indicate satisfactory reliability of the usability measurement instrument.

$$\text{SUS} = \frac{1}{n} \sum_{j=1}^{n} [(\Sigma^{i=1,3,5,7,9} (x_i - 1) + \Sigma^{i=2,4,6,8,10} (5 - x_i)) * 2.5]$$

With:

- n = total number of respondents
- $\frac{1}{n}\sum_{j=1}^{n}$ = average score across all respondents.
- x_i = score assigned by respondents to each SUS item, ranging from 1 to 5.
- $\sum i=^{1,3,5,7,9} (x_i - 1)$: score adjustment for positive statements.
- $\sum i=^{2,4,6,8,10} (5 - x_i)$: score adjustment for negative statements.
- 2.5: conversion factor used to scale the SUS score to a range of 0–100.

2.9. Data Validity and Reliability Analysis

Statistical analyses for the quantitative usability evaluation were performed using Microsoft Excel. Prior to analysis, the collected responses underwent a data cleaning process to identify and remove anomalous response patterns, including straight-lining behavior across alternating positive and negative questionnaire items. The validity of the ten questionnaire items was examined using Pearson's Product-Moment Correlation by comparing each item score with the overall questionnaire score. In addition, the internal consistency of the SUS instrument was evaluated using Cronbach's Alpha reliability analysis. A minimum Cronbach's Alpha coefficient of 0.70 was used as the acceptance threshold to indicate satisfactory reliability of the usability measurement instrument.

3. RESULTS AND DISCUSSION

This chapter presents the outcomes of the development process and evaluates the system's performance in enhancing SSL certificate lifecycle management through centralized API orchestration. The implementation stage represents the realization of the system architecture described in the previous chapter, where the conceptual model is transformed into a functioning web application designed to reduce manual configuration errors and operational bottlenecks.

3.1. Client-Side Interface (User Interaction)

The frontend was developed using Vue.js to ensure a reactive and accessible Single Page Application (SPA) experience. The interface design prioritizes the reduction of technical complexity by providing a unified dashboard. As illustrated in Fig. 3, Fig. 4, and Fig. 5, the system provides integrated interfaces for domain management, SSL queue monitoring, and certificate administration. Features such as "Bulk Generate" and "SSL Queue" explicitly display domain validation status and real-time certificate issuance progress. This interface design improves operational efficiency by eliminating the need for administrators to perform SSL management through command-line interactions.

3.2. Server-Side Infrastructure (Data Synchronization)

To support the automation requirements, the backend was developed using Laravel 10 (PHP 8.2) with MySQL as the database management system. The API architecture provided endpoints optimized for asynchronous job processing. Integration with ZeroSSL and Cloudflare was implemented through REST APIs to replace manual certificate configuration procedures. Background processes, including CNAME record creation, ZeroSSL verification triggering, and DNS record deletion after verification, were executed asynchronously using Laravel Queue workers and the Scheduler feature. This approach ensured that the main application process remained responsive during large-scale certificate operations. [25].

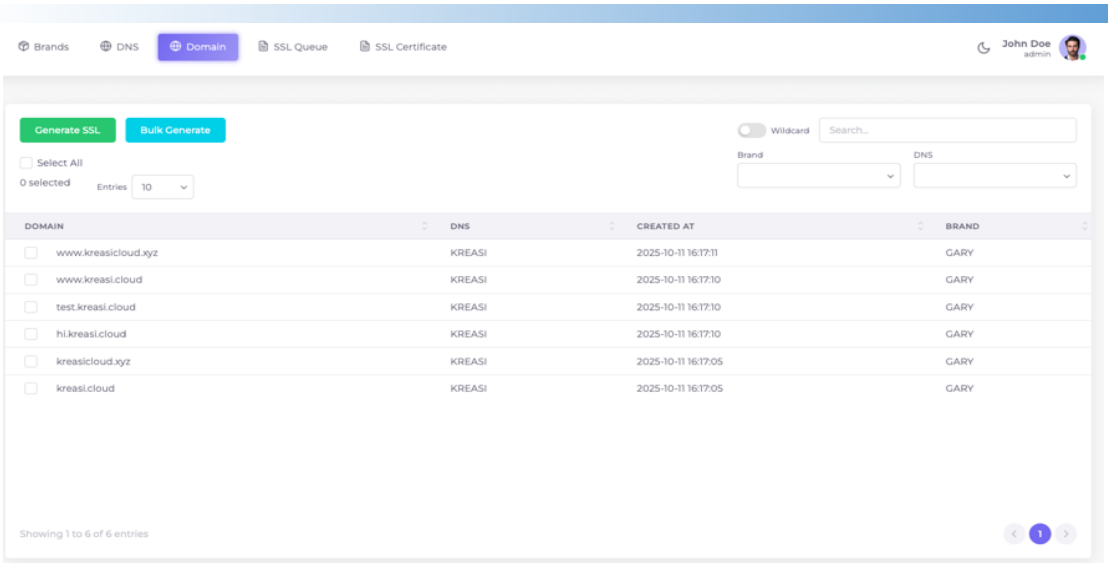

Fig. 3. Domain List Interface

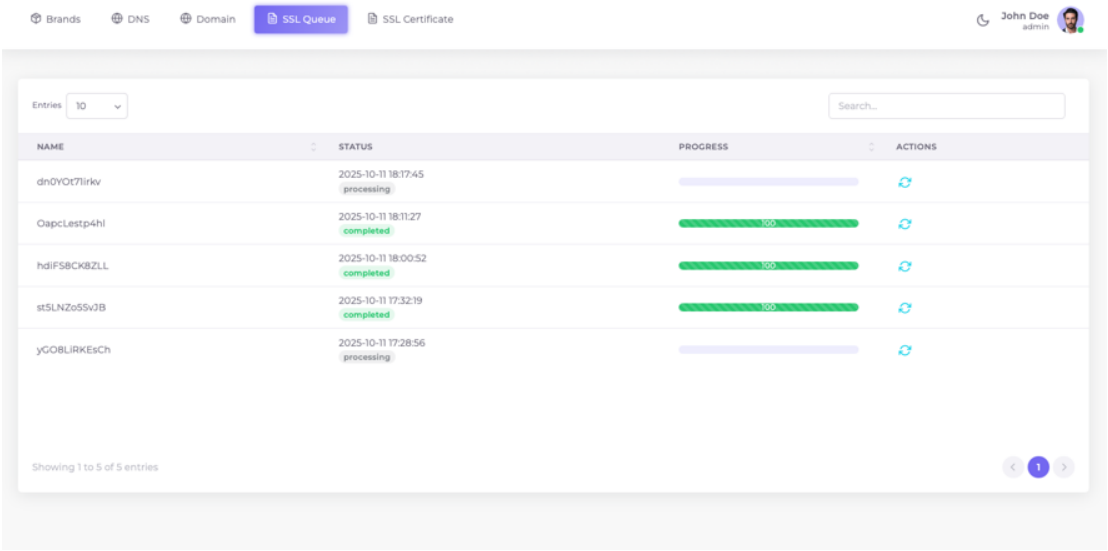

Fig. 4. SSL Queue Interface

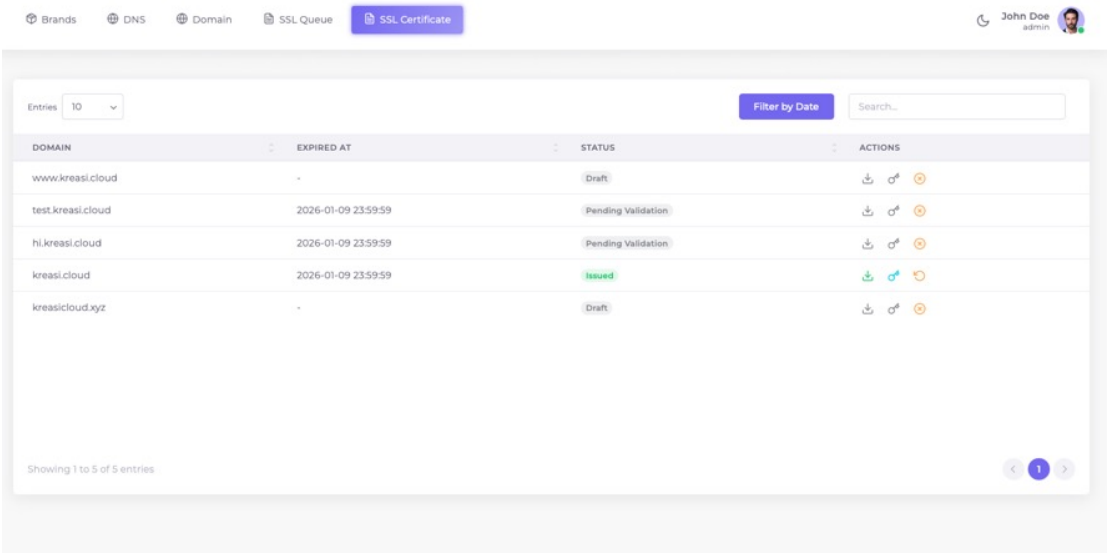

Fig. 5. SSL Certificate List

3.3. Data Validity

A Pearson Product-Moment Correlation analysis was conducted to assess the validity of the 10 System Usability Scale (SUS) questionnaire items. The correlation between each individual item score and the total score was calculated. For a sample size of N=25, the critical r-table value at a 10% significance level (α=0.10) is 0.336. The calculated correlation coefficients (r_{count}) for all ten items ranged from a minimum of 0.342 to a maximum of 0.894, successfully exceeding the critical threshold. Consequently, all questionnaire items were deemed valid constructs for measuring usability within this testing context.

3.4. Data Reliability

The internal consistency of the instrument was evaluated using Cronbach's Alpha to determine if the questionnaire yielded stable results across the technical respondent group. The mathematical analysis produced a total score variance of 30.65 and an accumulated item variance of 6.35, resulting in a Cronbach's Alpha coefficient of 0.881. This value significantly surpasses the recommended minimum threshold of $\alpha > 0.70$, confirming that the instrument demonstrates excellent internal reliability and that the collected data is robust for hypothesis testing.

3.5. System Performance and Automation Capabilities (H1)

System performance was assessed to validate the Automation Capabilities construct. Performance profiling was conducted on the asynchronous queueing mechanism using live domain tests. The system was subjected to batch requests ranging from 1 to 100 concurrent domains.

The benchmarks demonstrated high internal system efficiency. For small-scale bulk requests (1 to 5 domains), the end-to-end issuance latency from user submission to certificate availability ranged between 5 to 15 seconds. Under maximum-capacity stress tests (100 domains), the issuance completion time scaled to approximately 2 to 3 minutes. The observed variance is primarily attributable to external dependencies (ZeroSSL cryptographic forging speed and Cloudflare DNS propagation). Despite these external bottlenecks, the application's queue worker successfully prevented server timeouts. The 3-retry exponential backoff mechanism successfully managed API delays, proving robust fault tolerance. Therefore, Hypothesis 1 is supported.

3.6. Operational Efficiency Comparison

To illustrate the reduction in manual effort, the experimental results were evaluated in comparison with the Baseline Model, which represents a manual server-by-server SSL configuration approach.

3.7. Comparison of Baseline vs. Experimental Model

As shown in Table 3, the proposed system significantly reduces both temporal and technical overhead associated with SSL certificate management, while also alleviating the cognitive burden on administrators in multi-domain environments.

Table 3. Comparison of Baseline vs. Experimental Model

Metric	Baseline Model (Manual/Certbot per host)	Experimental Model (Proposed Web App)	Improvement
Execution Method	10-15 seconds (Certbot CLI / Manual Issuance via third party web)	5-15 seconds (Centralized Web Dashboard)	33% Reduction
DNS Verification	>30 seconds (Manual DNS Record Creation)	3-5 seconds (Automation API for CNAME Record Creation/Deletion)	90% faster, Automated Record Creation
Concurrency	>20 minutes (Sequential per-domain issuance)	>2 minutes (Asynchronous Bulk Issuance - 100 domains)	90%, Massive reduction in time-on-task
Visibility	Fragmented across server logs	Unified real-time tracking interface	Enhanced administrative transparency

3.8. Usability Analysis (H2)

User acceptance testing was conducted using the System Usability Scale (SUS) with 25 technical practitioners, including DevOps professionals, software developers, and IT students. The resulting SUS scores are presented in Table 4, while the overall score distribution is illustrated in Fig. 6.

Table 4. System Usability Scale (SUS) Questionnaire Results

Calculation Score										SUS
Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
4	2	4	1	5	2	5	1	4	2	82.5
5	1	5	1	4	1	5	1	4	1	90.0
4	1	4	2	4	2	4	2	5	2	82.5
4	2	5	1	5	1	4	2	5	1	87.5
3	2	4	2	4	1	4	1	4	2	77.5
5	1	5	1	5	2	5	1	5	1	90.0
4	2	4	1	5	2	5	1	4	2	82.5
5	1	5	1	5	1	5	1	5	1	95.0
5	1	4	1	5	1	5	1	5	2	90.0
4	1	5	1	5	2	5	2	4	1	85.0
5	1	5	2	5	2	4	1	5	2	87.5
4	2	5	2	5	2	4	2	4	2	80.0
5	1	4	1	5	1	5	1	5	1	95.0
4	2	4	1	5	2	4	1	5	1	82.5
5	1	5	2	5	1	5	1	5	1	90.0
4	2	5	1	5	2	4	2	5	2	82.5
4	1	4	2	4	1	4	1	5	2	82.5
5	2	5	1	5	1	5	1	5	2	90.0
4	2	4	1	4	1	4	1	5	2	82.5
5	1	5	1	5	2	5	1	5	1	90.0
4	2	4	2	5	1	5	1	5	2	82.5
5	1	5	1	5	1	5	1	4	1	90.0
5	1	5	1	5	1	5	1	5	1	95.0
4	2	5	1	5	1	5	2	5	1	85.0
5	1	5	1	5	1	5	1	5	1	95.0

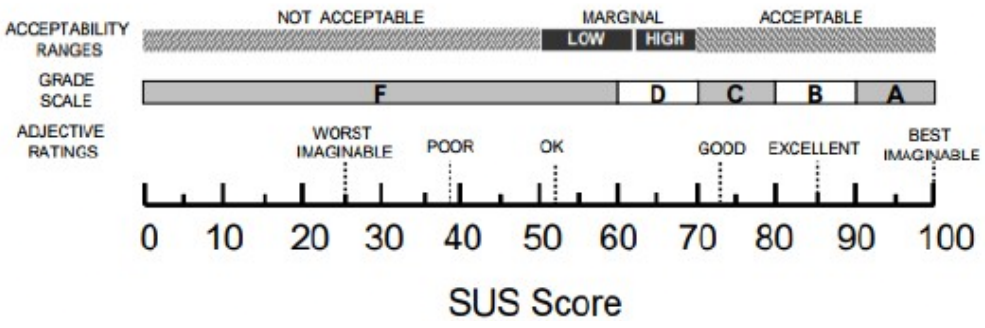

Fig. 6. System Usability Scale (SUS) Score Distribution

The evaluation yielded a mean System Usability Scale (SUS) score of 83.5 (standard deviation = 6.7), with individual scores ranging from 75.0 to 95.0. Based on established SUS interpretation benchmarks, scores above 80 are classified as indicating excellent usability. This result indicates that, despite the inherent complexity of cryptographic certificate issuance and DNS synchronization

processes, the proposed interface effectively abstracts system complexity and provides a user-friendly experience for practitioners. Accordingly, Hypothesis 2 is supported.

This study addressed the operational complexity and scalability challenges associated with host-centric SSL certificate management. The results indicate that the proposed web-based system reduces the gap between manual certificate administration and enterprise-level microservice-based platforms. In contrast to conventional tools such as Certbot, which require local installation and host-specific configuration, the proposed system centralizes the Domain Validation (DV) process within a unified interface.

The system automatically retrieves validation data from ZeroSSL and integrates it with Cloudflare DNS management, thereby reducing the need for manual DNS configuration. In addition, the implementation of an asynchronous Laravel Queue mechanism enables efficient bulk certificate issuance without causing external API rate limit violations or synchronous timeout issues. This architecture also helps mitigate the impact of latency from external services on user operations.

Nevertheless, several limitations were identified. The usability evaluation was conducted with a relatively small sample of technical participants (N = 25), which may limit the generalizability of the findings to non-technical administrators. Furthermore, the current implementation relies on a single Certificate Authority provider (ZeroSSL), introducing a potential single point of failure in the event of upstream service disruptions.

4. CONCLUSION

This study successfully designed, implemented, and evaluated a centralized web-based SSL certificate automation system to address the operational complexities of host-centric certificate lifecycle management. Guided by the Design Science Research (DSR) approach, the development process applied Agile Scrum methodology to produce a reactive Vue.js interface and a Laravel-based backend orchestration system. Through native integration with the Cloudflare DNS API and ZeroSSL REST API, the system effectively abstracted Domain Validation (DV) processes from local server environments and reduced the need for manual cryptographic configuration.

The evaluation results confirmed that the system met the intended research objectives outlined in the introduction. The asynchronous queue-based architecture demonstrated strong automation performance by handling concurrent bulk certificate issuance while mitigating external API rate limits and avoiding synchronous server timeouts. In addition, the usability evaluation produced a System Usability Scale (SUS) score of 83.5, indicating excellent usability and confirming that the centralized dashboard significantly reduced administrative effort and cognitive workload. These findings demonstrate that abstracting SSL operations into an API-driven orchestration layer improves efficiency, reduces configuration errors, and supports scalable security management practices in distributed environments.

Despite these positive outcomes, several limitations were identified. The usability testing involved a relatively small and highly technical sample (N = 25), which may not fully represent non-technical users. The system was also not evaluated under extreme enterprise-level load conditions, limiting insights into large-scale stress performance. Furthermore, the architecture depends on a single Certificate Authority (ZeroSSL), which introduces a potential single point of failure in cases of service disruption. Security validation of backend cryptographic storage was also outside the scope of this study.

To address these limitations, future research should focus on enhancing system robustness and scalability. Recommended improvements include multi-CA integration (such as Let’s Encrypt and Google Trust Services) to improve redundancy and availability. Future work should also incorporate load-balanced queue architectures for large-scale deployment testing and conduct comprehensive stress and penetration testing. Additionally, stronger security mechanisms such as Hardware Security Modules (HSM) or advanced encryption techniques for private key storage should be explored to improve enterprise-grade security assurance.

REFERENCES

[1] Y. Wang *et al.*, "Identifying vulnerabilities of SSL/TLS certificate verification in Android apps with static and dynamic analysis," *J. Syst. Softw.*, vol. 167, p. 110609, Sep. 2020, doi: 10.1016/j.jss.2020.110609.

[2] Y. Zhu, E. Haihong, and M. Song, "A Scheduling System for Big Data Hybrid Computing Workflow," in *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China: IEEE, Oct. 2020, pp. 102–106. doi: 10.1109/ICSESS49938.2020.9237729.

[3] J. Aas *et al.*, "Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web," in Proceedings *of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London United Kingdom: ACM, Nov. 2019, pp. 2473–2487. doi: 10.1145/3319535.3363192.

[4] T. V. Doan, I. Tsareva, and V. Bajpai, "Measuring DNS over TLS from the Edge: Adoption, Reliability, and Response Times," in *Passive and Active Measurement*, vol. 12671, O. Hohlfeld, A. Lutu, and D. Levin, Eds., in Lecture Notes in Computer Science, vol. 12671. , Cham: Springer International Publishing, 2021, pp. 192–209. doi: 10.1007/978-3-030-72582-2_12.

[5] H. Lee, D. Kim, and Y. Kwon, "TLS 1.3 in Practice: How TLS 1.3 Contributes to the Internet," in *Proceedings of the Web Conference 2021*, Ljubljana Slovenia: ACM, Apr. 2021, pp. 70–79. doi: 10.1145/3442381.3450057.

[6] J. Göppert, A. Walz, and A. Sikora, "A Generic Credential Management Model for Secure Field-Level Communication in IIoT Networks," *IEEE Access*, vol. 14, pp. 18799–18811, 2026, doi: 10.1109/ACCESS.2026.3659483.

[7] A. Liu, A. Alqazzaz, H. Ming, and B. Dharmalingam, "Iotverif: Automatic Verification of SSL/TLS Certificate for IoT Applications," *IEEE Access*, vol. 9, pp. 27038–27050, 2021, doi: 10.1109/ACCESS.2019.2961918.

[8] J. Astorga, M. Barcelo, A. Urbieta, and E. Jacob, "How to Survive Identity Management in the Industry 4.0 Era," *IEEE Access*, vol. 9, pp. 93137–93151, 2021, doi: 10.1109/ACCESS.2021.3092203.

[9] A. L. Rotthaler, H. S. Ramulu, L. Simko, S. Fahl, and Y. Acar, "'It's Time. Time for Digital Security.': An End User Study on Actionable Security and Privacy Advice," in *2025 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA: IEEE, May 2025, pp. 2228–2245. doi: 10.1109/SP61157.2025.00100.

[10] R. Matsumoto, K. Rikitake, and K. Kuribayashi, "Large-scale Certificate Management on Multi-tenant Web Servers," *J. Inf. Process.*, vol. 27, no. 0, pp. 650–657, 2019, doi: 10.2197/ipsjjip.27.650.

[11] Firkhan Ali Hamid Ali, Mohd Khairul Amin Mohd Sukri, Mohd Zalisham Jali, Muhammad Al-Fatih, and Mohd Azhari Mohd Yusof, "Web-Based Reporting Vulnerabilities System for Cyber Security Maintenance," *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 29, no. 3, pp. 198–205, Feb. 2023, doi: 10.37934/araset.29.3.198205.

[12] Y. Fidaner, A. Coskun, and T. Ergun, "S/MIME Certificate Test Suite," *Eurasia Proc. Sci. Technol. Eng. Math.*, vol. 24, pp. 83–88, Dec. 2023, doi: 10.55549/epstem.1406237.

[13] D. Aryachandra, I. Fikri Yanto, M. Miftahul Khair, and M. Reza Sah Pahlevi, "Menyembunyikan Alamat IP Webserver dengan Proxy Dns Records Cloudflare," *J. Sos. Teknol.*, vol. 4, no. 4, pp. 218–226, Apr. 2024, doi: 10.59188/jurnalsostech.v4i4.1221.

[14] F. B. Manolache and O. Rusu, "Automated SSL/TLS Certificate Distribution System," in *2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, Iasi, Romania: IEEE, Nov. 2021, pp. 1–6. doi: 10.1109/RoEduNet54112.2021.9637722.

[15] A. Aksoy, L. Valle, and G. Kar, "Automated Network Incident Identification through Genetic Algorithm-Driven Feature Selection," *Electronics*, vol. 13, no. 2, p. 293, Jan. 2024, doi: 10.3390/electronics13020293.

[16] P. S. Yadav, "Automation of Digital Certificate Lifecycle: Improving Efficiency and Security in IT Systems," *J. Math. Comput. Appl.*, pp. 1–4, Dec. 2022, doi: 10.47363/JMCA/2023(2)E107.

[17] K. E. CiBalik and C. Koçak, "Detection of SSL/TLS Implementation Errors in Android Applications," *Gazi Üniversitesi Fen Bilim. Derg. Part C Tasar. Ve Teknol.*, vol. 9, no. 2, pp. 211–219, Jun. 2021, doi: 10.29109/gujsc.878053.

[18] P. Gregory, D. E. Strode, H. Sharp, and L. Barroca, "An onboarding model for integrating newcomers into agile project teams," *Inf. Softw. Technol.*, vol. 143, p. 106792, Mar. 2022, doi: 10.1016/j.infsof.2021.106792.

[19] T. Tuunanen, R. Winter, and J. V. Brocke, "Dealing with Complexity in Design Science Research: A Methodology Using Design Echelons," *MIS Q.*, vol. 48, no. 2, pp. 427–458, Jun. 2024, doi: 10.25300/MISQ/2023/16700.

[20] M. Göktürk, "An Interface Evaluation Model for Usability and Perceived Security," *IEEE Access*, vol. 13, pp. 91989–92007, 2025, doi: 10.1109/ACCESS.2025.3572108.

[21] C. Mou, "DNS is the Internet Pivotal Basics and Fundamental," *Int. J. Adv. Netw. Monit. Controls*, vol. 7, no. 2, pp. 11–23, Jan. 2022, doi: 10.2478/ijanmc-2022-0012.

[22] H. Park, K. Lim, D. Kim, D. Yu, and H. Koo, "Demystifying the Regional Phishing Landscape in South Korea," *IEEE Access*, vol. 11, pp. 130131–130143, 2023, doi: 10.1109/ACCESS.2023.3333883.

[23] R. Halder, D. Das Roy, and D. Shin, "A Blockchain-Based Decentralized Public Key Infrastructure Using the Web of Trust," *J. Cybersecurity Priv.*, vol. 4, no. 2, pp. 196–222, Mar. 2024, doi: 10.3390/jcp4020010.

[24] I. F. Ashari, A. J. Aryani, and A. M. Ardhi, "Design and build inventory management information system using the Scrum method," *JSiI J. Sist. Inf.*, vol. 9, no. 1, pp. 27–35, Mar. 2022, doi: 10.30656/jsii.v9i1.4050.

[25] A. Hounsel, K. Borgolte, P. Schmitt, J. Holland, and N. Feamster, "Comparing the Effects of DNS, DoT, and DoH on Web Performance," in *Proceedings of The Web Conference 2020*, Taipei Taiwan: ACM, Apr. 2020, pp. 562–572. doi: 10.1145/3366423.3380139.

AUTHOR BIOGRAPHY

Fredian Simanjuntak was born in Indonesia. He received his Master's degree in Computer Science from Bina Nusantara (BINUS) University. He is currently a lecturer in the Faculty of Information Systems at Universitas Internasional Batam. His academic interests include software engineering, information systems, and technology-driven innovation.

Gary Happydinata was born in Singapore and is currently pursuing a Bachelor's degree in Information Systems at the Faculty of Computer Science, Universitas Internasional Batam. His research interests include web application engineering and automation technologies supporting modern DevOps practices.

Herman was born in Indonesia. He received his Master's degree in Computer Science from Bina Nusantara (BINUS) University. He currently serves as a lecturer in the Faculty of Information Systems, Universitas Internasional Batam. His academic interests include software engineering, information systems, and technology innovation.