Design and Development of Sibita: A Web-Based Final Project Supervision Information System with Role-Based Workflow, Notifications, and an Audit Trail

H. A. Mooduto[a,*]

[a] *Department of Information Technology, Politeknik Negeri Padang, Limau Manis, Padang, 25164, Indonesia*
Corresponding author: [*] mooduto@pnp.ac.id

***Abstract*— Final project supervision in Indonesian vocational higher education is still largely coordinated through paper logbooks and personal messaging, which leaves progress difficult to monitor and decisions difficult to trace. This study designs, builds, and evaluates Sibita (Sistem Informasi Tugas Akhir), a web-based information system that covers the final project cycle, from supervisor assignment to the defense. The materials comprise the application source code (61 PHP files, about 24,700 lines), its MariaDB schema (26 tables, 33 foreign keys), and server-side records from twelve months of pilot operation. Development followed an iterative waterfall process in which a page-by-page code review against web security risk categories was inserted between implementation and deployment. The resulting system serves three roles, namely students, lecturers, and administrators, through 47 pages that integrate document review with an explicit state model, meeting logs, six-criterion assessment, defense scheduling with conflict detection, chat, notifications, and an audit trail. The review recorded 193 corrections on 38 pages; missing request-forgery protection affected 26 pages, and missing ownership checks affected 12. During the pilot, the authentication log registered 527 successful logins from 19 accounts and 26 failed attempts, while the database recorded 20 supervision pairs, 51 document submissions, 39 review comments, and 328 audited actions. These results indicate that a compact system can digitize supervision while keeping every decision attributable, and that systematic review is indispensable before such a system is exposed to users. Further research should measure usability and acceptance with standardized instruments and examine the effect of the system on completion time.**

***Keywords*— Final project information system; thesis supervision; role-based access control; web application security; audit trail; notification.**

I. INTRODUCTION

The final project (*tugas akhir*) is the capstone of diploma and applied-bachelor programs in Indonesian polytechnics. It is completed under the guidance of one or two supervisors over one or two semesters, and it ends with an oral defense before a panel of examiners. Throughout this period, the student and the supervisors exchange drafts, comments, revision requests, and meeting notes, while the department must know at any time who is supervising whom, how far each student has progressed, and who is ready to be examined. In many departments this coordination still relies on a printed logbook that is signed at each meeting, on drafts sent through personal messaging applications, and on spreadsheets kept by administrative staff.

The literature on research supervision explains why such arrangements are fragile. Reviews of supervision conducted at a distance report that students are more exposed to isolation and that irregular contact with the supervisor is among the recurring difficulties [1], [2]; good practice therefore depends on a channel that keeps the relationship visible and regular [3], and on academic support that is easy to reach online [4]. Written feedback on drafts is regarded as the central instrument of supervision [5], yet students often find that feedback arrives late or does not match what they expected [6], they engage with it unevenly [7], and their preferences differ across fields [8]. A logbook and a chat thread record neither the status of a draft nor the deadline attached to a revision, so the information that both parties need most is exactly what the manual process loses.

Web-based systems have been proposed to close this gap. Earlier work has designed thesis management systems that move title submission and document handling online [9], modeled the database for thesis data [10], monitored the activity of supervisors with attention to user experience [11], and evaluated the interface of an academic e-supervision application for vocational education [12]. Dashboards that summarize learner activity have also matured, and current guidance stresses that they should lead to action rather than merely display data [13], [14]. Two aspects nevertheless remain thinly covered in this body of work. First, most reports stop at functional testing and say little about security, although supervision systems store unpublished manuscripts, grades, and personal data, and although broken access control and injection continue to lead the list of web application risks [15], [16]. Second, few reports describe how administrative decisions made in the system can later be reconstructed.

This paper reports the design and development of Sibita (Sistem Informasi Tugas Akhir), a final project information system built in the Department of Information Technology, Politeknik Negeri Padang. The contribution is threefold: (1) an integrated workflow that connects supervisor assignment, document review, meeting logs, assessment, and defense scheduling for three roles; (2) a security and accountability layer, consisting of role-based access control, request-forgery protection, hardened file handling, and an audit trail, together with a quantified account of the defects that a structured code review removed before deployment; and (3) an evaluation based on server-side records from twelve months of pilot operation. The remainder of the paper describes the materials and method, presents and discusses the results, and closes with conclusions and directions for further research.

II. Materials and Method

A. Research Stages

The work followed a sequential life cycle in the sense of the waterfall model [17], [18], with one deliberate deviation. Comparative studies note that a plan-driven process suits projects whose requirements are stable and well understood, whereas iterative processes cope better with late findings [19], [20]. The supervision procedure of the department is stable, which favors the sequential model; however, experience with the first implementation showed that defects in authorization logic only surface when pages are read against one another. A review-and-hardening stage was therefore inserted between implementation and deployment and was allowed to loop back to implementation, as shown in Fig. 1.

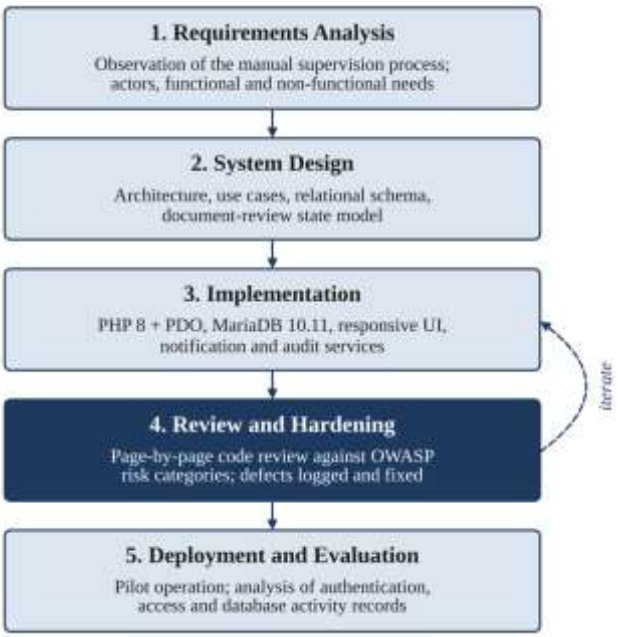


Fig. 1 Research stages; the review stage iterates with implementation

B. Materials

Three kinds of material were used. The first is the application itself: 61 PHP source files amounting to about 24,700 lines, of which 47 are page controllers and the rest are shared services for sessions, notifications, audit logging, and chat. The second is the relational schema, exported from MariaDB 10.11, which contains 26 base tables, one view, 33 foreign-key constraints, and three triggers. The third consists of records produced by the running system during the pilot: the authentication log (30 October 2025 to 12 November 2026), the daily counter of landing-page visits (17 August to 15 November 2026), the stored document files, and the identifier counters of the database as exported on 11 March 2026. No questionnaire or interview data were collected; the evaluation is confined to what these records can support.

C. Requirements

Requirements were derived from the supervision procedure of the department and grouped by actor. Students need to submit drafts to a specific supervisor, see the decision and the revision deadline, follow their own progress, and propose a defense once supervision is complete. Lecturers act in two capacities, as supervisors who review documents, record meetings, and assess students, and as examiners who enter defense scores. Administrators maintain master data, assign supervisors, validate final files, schedule defenses, and broadcast announcements. Non-functional requirements were stated with reference to the product quality model of ISO/IEC 25010 [21], with emphasis on security (confidentiality, integrity, accountability), functional correctness, and operability on mobile browsers. Table I summarizes the functional scope that resulted.

TABLE I
Functional Scope of Sibita by Role

Role	Main functions
Student	Upload documents and revisions; read feedback and deadlines; five-stage progress view; meeting history; chat; defense proposal; schedule; notifications
Lecturer	Review queue with accept or request-revision decisions; page-level comments; meeting log; six-criterion assessment; progress monitoring with CSV export; defense scores; calendar export
Administrator	Users, majors, study programs, academic periods; supervisor assignment; document oversight; final-file validation; defense scheduling; reports; announcements; audit trail

D. System Design

The system adopts a three-tier architecture (Fig. 2). The client tier is a responsive interface rendered in the browser. The application tier runs on PHP 8 and is organized as page controllers that pass through a common security layer before reaching the functional modules; all database access is performed through PDO prepared statements with native preparation enabled. The data tier consists of the MariaDB database, a file store for uploaded documents, and a scheduler that sends daily reminders and enforces log retention. Behavior was modeled with a use case diagram (Fig. 3).

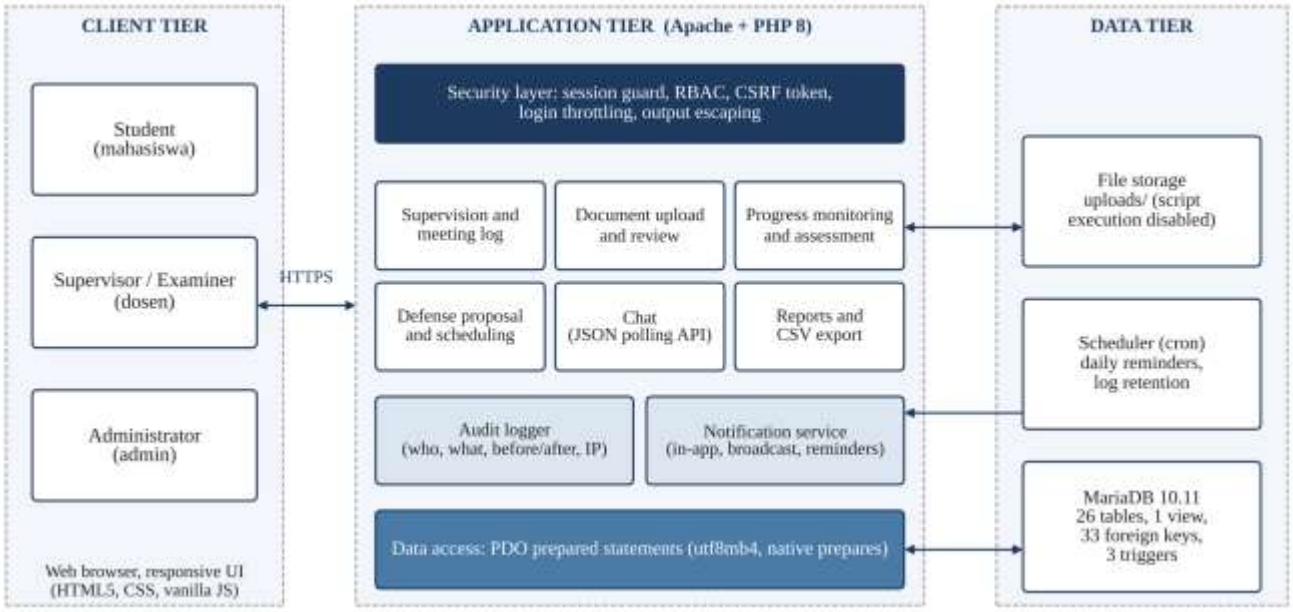


Fig. 2 Three-tier architecture of Sibita

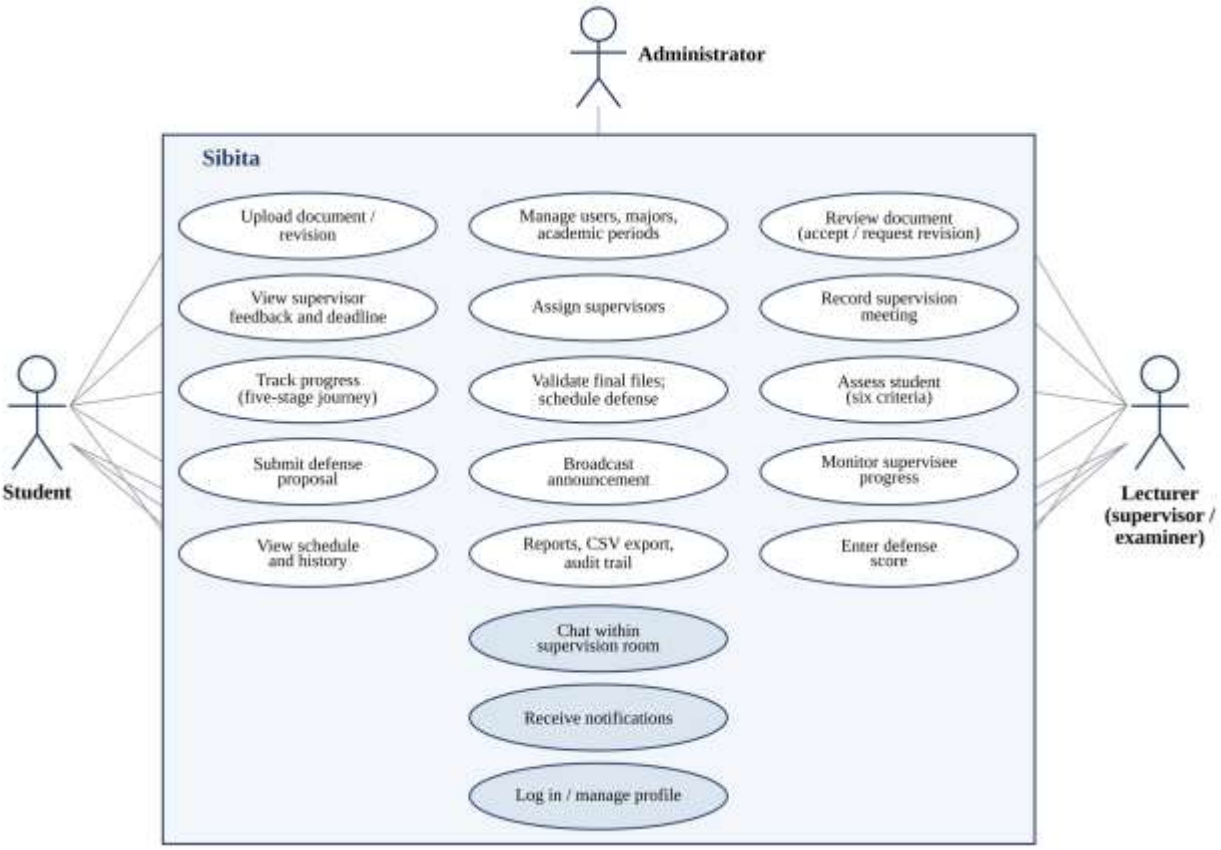


Fig. 3 Use case diagram; shaded use cases are shared by all actors

Access control follows the role-based model, in which permissions are attached to roles and users acquire them through role membership [22]; the model is widely applied in multi-user web information systems because it keeps authorization decisions in one place [23]. Sibita defines three roles and, in addition, checks ownership at the level of each record: a supervisor may act only on supervision records in which he or she is named, a student only on his or her own documents, and only the first supervisor may close a supervision record.

The review of a document is modeled as a finite state machine with four states (Fig. 4). A newly uploaded document awaits review; the supervisor to whom it is addressed either accepts it or requests a revision, in which case a note and a deadline that is not in the past are mandatory. When the student uploads the revision, the

document enters a distinct state so that it reappears at the top of the supervisor's queue. Every transition is submitted as a token-protected request, generates a notification for the other party, and is written to the audit log.

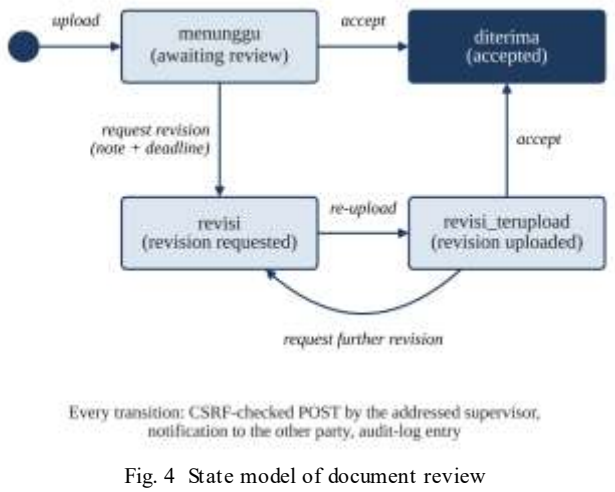


Fig. 4 State model of document review

Two computations in the assessment module are fixed by design. After a meeting, the supervisor rates six criteria (mastery of the subject, progress, document quality, independence, punctuality, and overall impression) on a scale from 1 to 10 in steps of 0.5, and the meeting score is their arithmetic mean,

$$r = \frac{1}{6}\sum_{i=1}^{6} c_i \tag{1}$$

where c_i is the rating of criterion i; one assessment is allowed per meeting and lecturer. The defense score is the mean of the scores (0 to 100) entered by the examiners who are actually assigned,

$$S = \frac{1}{n}\sum_{j=1}^{n} s_j, \quad 1 \le n \le 3 \tag{2}$$

so that a panel of two examiners is not penalized by an empty third slot. A defense is marked as completed automatically when all assigned examiners have entered their scores.

Data were modeled with the entity-relationship approach [24] and then mapped to tables. Fig. 5 shows the core entities. The supervision record (*bimbingan*) is the hub of the model: it links one student to a first and an optional second supervisor, and documents, meetings, assessments, and chat rooms all depend on it, whereas defenses refer to the student and to as many as three examiners.

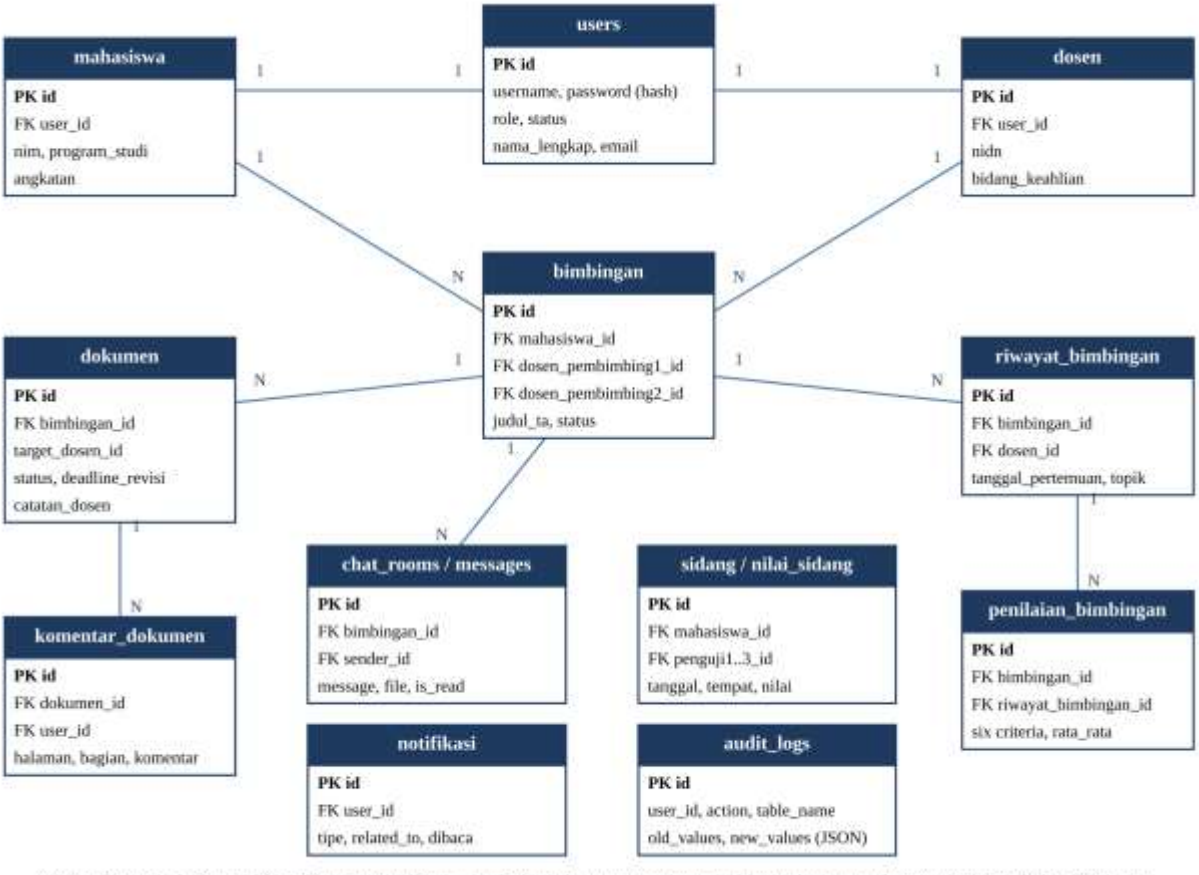


Fig. 5 Core entities and relationships of the Sibita database (attributes abridged)

E. Security Design

Controls were selected against the risk categories of the OWASP Top 10 [15]. Injection is prevented by parameterized queries, the countermeasure on which classifications and reviews of SQL injection agree [25], [26], [27]. Cross-site scripting, which surveys still describe as one of the most prevalent web flaws [28], [29], is addressed by

escaping all output through a single helper. Cross-site request forgery is addressed by a per-session synchronizer token on every state-changing request [30] and, as a second layer, by cookies marked SameSite=Strict; this combination is deliberate, because studies of web frameworks and of the SameSite attribute show that neither defense is reliable alone [31], [32], [33], [34].

File upload received particular attention, since unrestricted upload of executable files is a well-documented path to remote code execution [35], [36], [37]. Passwords are stored with the adaptive hashing interface of PHP, which uses bcrypt by default [38], [39], and failed logins are throttled per account and per address in line with authentication guidance [40]. Finally, all changes to data are recorded in an audit log, because logging is a precondition for accountability and for the investigation of incidents [41].

F. Review and Evaluation Procedure

In the review stage, each page controller was read in full against a fixed checklist: authentication and role check, ownership check on every identifier received from the client, request-forgery token on every state-changing request, input validation, output escaping, query construction, file handling, error handling, and consistency of the figures shown to the user. Each defect was corrected and described in a change note placed in the header of the file. For this paper, the 193 notes found in 38 files were classified by keyword into nine categories, and the number of pages affected by each category was counted; a page is counted once per category.

Operational evaluation used descriptive statistics only. Successful and failed logins were counted per month from the authentication log; the number of records ever created in each table was read from the auto-increment counters of the database export, which gives an upper bound on the rows currently stored; and visits to the public landing page were aggregated per week. Perceived usability was not measured in this study. The System Usability Scale [42], [43], which is the instrument most often applied to educational technology [44], [45], is planned for the next phase and is discussed under limitations.

III. Results and Discussion

A. Implemented System

The implemented system consists of 47 pages: 13 for administrators, 13 for students, and the remainder for lecturers and for functions shared by all roles, such as login, profile, chat, and notifications. A single login form serves all users; the role and the corresponding dashboard are determined from the account. The interface is written without a client-side framework, and charts are drawn as inline graphics, so that pages remain usable on the mobile connections that students commonly rely on.

The student dashboard replaces a generic statistics panel with a list of next actions (for example, a revision that is due) and a five-stage view of the journey from supervisor assignment to the defense. This follows the recommendation that dashboards should point to what the user ought to do next [13]. The lecturer dashboard opens with the review queue, in which re-uploaded revisions are listed first, and the monitoring page shows the stage reached by each supervisee together with a weekly activity map that can be exported.

Communication is supported in two ways. Each supervision record has a chat room restricted to the student and the supervisors; the client polls a JSON endpoint every four seconds, falls back to fifteen seconds after three consecutive failures, and stops polling while the browser tab is hidden. Notifications are created in the same transaction as the event that causes them, for example an upload, a decision, a comment, or a schedule, while a scheduled task sends reminders one day before a meeting and before a revision deadline and is guarded so that it runs once per day. Administrators can broadcast announcements to a role; these are delivered only to active accounts.

Defense administration is the last link of the workflow. A student whose supervision has been closed and whose required files are complete submits a proposal; the administrator either rejects it with a note or accepts it by scheduling the defense. On scheduling, the system refuses a slot in which the same examiner or the same room is already booked within ninety minutes. Examiners enter their scores independently, and the final score follows (2).

B. Database Implementation

The schema was implemented with the InnoDB engine and declares 33 foreign keys, most of them with cascading deletion from the supervision record to its documents, meetings, comments, assessments, and chat rooms. Deletion of a supervision record that already has such dependents is nevertheless blocked in the application, so that history cannot be removed by accident. A view lists the students who are ready for the defense, and three triggers write an event to a queue table whenever a document, a comment, or a meeting is inserted. Table II lists the core tables with the number of records created up to the export date.

TABLE II
CORE TABLES AND RECORDS CREATED UP TO 11 MARCH 2026

Table	Content	Records
users	Accounts of all roles	54
dosen / mahasiswa	Lecturer and student profiles	26 / 23
bimbingan	Supervision pairs and project titles	20
dokumen	Document submissions	51
komentar, komentar_dokumen	General and page-level comments	39
riwayat_bimbingan	Supervision meetings	15
chat_messages	Messages in supervision rooms	15
notifikasi	In-app notifications	75
audit_logs	Audited actions	328

Records are read from auto-increment counters and therefore include rows that were later deleted.

C. Security Controls

Table III maps the main risks to the controls that are present in the deployed code. Three of them deserve comment. First, the upload routine accepts a file only if its extension is on a whitelist of document, archive, and image types and if the type detected from the file content matches that extension; the file is then stored under a random name, and the upload directory is configured so that the web server never executes scripts from it. A file that merely begins with

a PDF signature but carries a script extension, which passed the first implementation, is rejected. Second, documents are no longer served through direct links; every download passes through a controller that verifies the requester's relation to the supervision record and writes an audit entry. Third, the login procedure always performs a hash verification, against a dummy hash when the account does not exist, so that response time does not reveal which user names are valid.

TABLE III
RISKS AND IMPLEMENTED CONTROLS

Risk	Control in Sibita
SQL injection	PDO prepared statements with native preparation; 198 prepared queries; whitelisted column names
Cross-site scripting	Central output-escaping helper on all dynamic output
Request forgery	Per-session token compared in constant time on 31 pages; SameSite=Strict, HttpOnly session cookie
Broken access control	Role check and record-ownership check on every page; mediated downloads
Malicious upload	Extension whitelist, content-type match, 50 MB limit, random file name, execution disabled in upload directory
Credential attacks	Adaptive password hashing; lockout for 15 minutes after 5 failures per account and address or 20 per address; uniform response time
Session fixation	Session identifier regenerated at login and every 30 minutes
Repudiation	Audit log with actor, action, record, values before and after, address, and device; exports are themselves logged

D. Defects Removed by the Review

The review stage produced 193 change notes in 38 of the 47 pages. Fig. 6 shows how many pages were affected by each category of defect. The most widespread defect was the absence of a request-forgery token on state-changing requests, found on 26 pages. Missing ownership or authorization checks affected 12 pages; typical cases were a student who could attach a document to another student's supervision record because the request was processed before ownership was verified, and a second supervisor who could close a supervision record. Unsafe file handling affected eight pages, and one page built a column name from user input, which made it open to SQL injection.

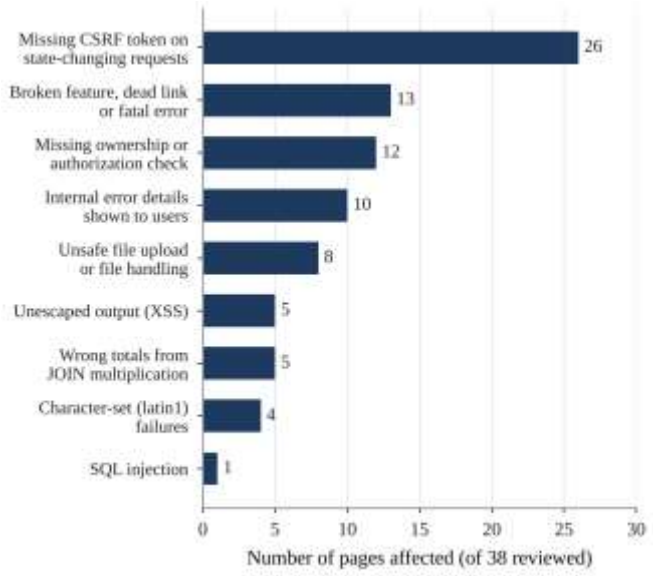


Fig. 6 Pages affected by each defect category found in the code review

Not all defects concerned security. On 13 pages a feature did not work at all, for instance because of a link to a page that did not exist or a missing class that stopped execution after the data had been saved. On five pages the totals shown to users were inflated because documents and meetings were joined in a single query, so that two documents and four meetings were reported as eight of each. Such errors do not appear in a functional test that only checks whether a page loads and saves, which is the form of black-box testing reported in most comparable studies [9], [11].

The distribution in Fig. 6 is consistent with the wider evidence. Access-control failures head the OWASP ranking [15], and here the two categories that concern who may perform an action, namely request forgery and ownership, together account for the largest share of affected pages. The pattern also agrees with the observation that token-based defenses protect only where developers remember to apply them [31]: in an application written without a framework, nothing applies the token by default. The practical lesson is that a small academic system built by a small team should not rely on functional testing alone; a checklist-driven reading of every page, repeated after each change, was what revealed the defects that mattered most.

E. Operational Use

Between 30 October 2025 and 12 November 2026 the authentication log recorded 527 successful logins by 19 distinct accounts, 26 failed attempts, and 399 explicit logouts. The share of failed attempts among all attempts is given by (3).

$$f = \frac{26}{527 + 26} = 4.7\% \tag{3}$$

About three quarters of the sessions (399 of 527) were closed by an explicit logout rather than by expiry. Fig. 7 shows the monthly distribution. The two highest months, October and November 2025, coincide with commissioning, when administrator and test accounts were used intensively; activity then settled at 23 to 31 logins per month from March to July 2026. No successful login was written to this log from August to October 2026, although the landing page continued to be visited (Fig. 8); the gap may reflect the

semester break or a change in logging during the redesign of the login page, and it should not be read as evidence of zero use. Logins reappear in November 2026.

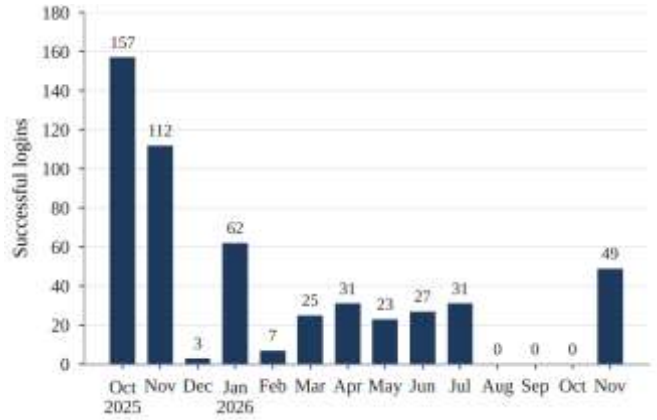


Fig. 7 Successful logins per month, October 2025 to November 2026

By the export date, 20 supervision pairs had been registered and had produced 51 document submissions, 39 comments, 15 recorded meetings, and 75 notifications (Table II). At the end of the observation period the file store held 31 documents, 28 in PDF and three in DOCX format, occupying about 65 MB. The audit log had recorded 328 actions. The public landing page was visited 1,044 times in the 90 days for which the counter is available, with a median of 10 and a mean of 11.6 visits per day; the weekly series in Fig. 8 fluctuates between 32 and 132 visits without a clear trend.

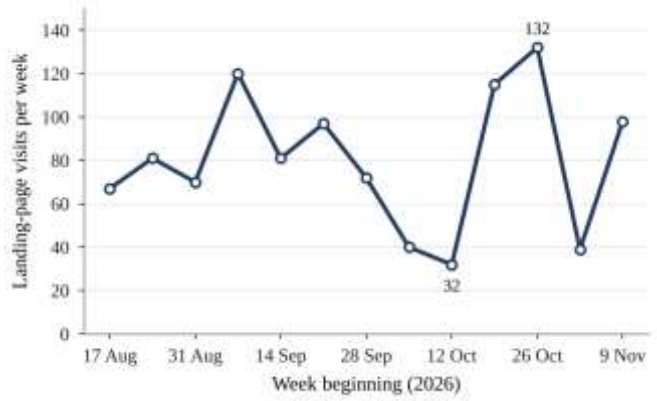


Fig. 8 Visits to the public landing page per week, 17 August to 15 November 2026

These figures describe a pilot of modest size, but they show the workflow in use from end to end: documents were submitted, reviewed, and commented on, meetings were logged, and each of these steps left an attributable trace. On average, each supervision pair produced about 2.6 submissions and two comments, which suggests that drafts and feedback did move through the system rather than around it. The ratio of comments to submissions (39 to 51) is of particular interest in the light of the feedback literature [5], [6]: with the manual process, neither number is known to the department at all.

F. Limitations

The evaluation has four limitations. First, it relies on system records and does not include users' judgments; usability and acceptance remain to be measured, and the usability of educational technology is known to vary considerably between systems [44]. Second, the pilot involved 19 active accounts in one department, so the load and the diversity of use are limited. Third, defect categories were assigned by keyword from the developers' own change notes, which may undercount defects that were fixed without a note; an independent penetration test would give a stronger assurance. Fourth, some design debts remain: tables use a single-byte character set, so text outside that set is rejected with a message instead of being stored; two comment tables overlap in purpose; and the event queue filled by triggers has become redundant since pages create notifications directly.

IV. CONCLUSION

This study designed and built Sibita, a web-based information system that carries the final project process of a polytechnic department from supervisor assignment through document review, meeting logs, and assessment to the scheduling and scoring of the defense. The system serves students, lecturers, and administrators through 47 pages on a three-tier architecture, with a database of 26 tables, a document-review state model, in-room chat, notifications, and an audit trail.

Two findings stand out. A checklist-driven code review inserted before deployment recorded 193 corrections on 38 pages, dominated by missing request-forgery tokens (26 pages) and missing ownership checks (12 pages), none of which would have been caught by testing whether pages load and save. In twelve months of pilot operation the system registered 527 successful logins from 19 accounts, 51 document submissions, 39 comments, and 328 audited actions, which shows that the workflow is usable end to end and that each decision can be traced to an actor and a time.

Further research will proceed in three directions: measuring usability and acceptance with standardized instruments such as the System Usability Scale across all three roles; extending the pilot to other departments and comparing completion time and feedback turnaround before and after adoption; and strengthening the platform through an independent penetration test, migration of the database to a full Unicode character set, and integration with the institution's academic information system and single sign-on.

ACKNOWLEDGMENT

The authors thank the Department of Information Technology, Politeknik Negeri Padang, and the lecturers and students who took part in the pilot.

REFERENCES

[1] E. Melián, J. I. Reyes, and J. Meneses, "The online PhD experience: A qualitative systematic review," *Int. Rev. Res. Open Distrib. Learn.*, vol. 24, no. 1, pp. 137–158, Feb. 2023, doi: 10.19173/irrodl.v24i1.6780.

[2] S. Miller, F. Miller, K. Baddock, and S. Baddock, "Research supervision at distance in higher education: A review," *Scope: Contemp. Res. Topics (Learn. Teach.)*, no. 12, p. 71, Nov. 2023, doi: 10.34074/scop.4012012.

[3] M. A. Gray and L. Crosta, "New perspectives in online doctoral supervision: A systematic literature review," *Stud. Contin. Educ.*, vol. 41, no. 2, pp. 173–190, 2019, doi: 10.1080/0158037X.2018.1532405.

[4] C. Walsh, L. A. Bragg, M. Heyeres, A. Yap, and M. Ratcliff, "A systematic literature review of online academic student support in higher education," *Online Learn.*, vol. 28, no. 2, pp. 1–24, 2024, doi: 10.24059/olj.v28i2.3954.

[5] R. Chugh, S. Macht, and B. Harreveld, "Supervisory feedback to postgraduate research students: A literature review," *Assess. Eval. High. Educ.*, vol. 47, no. 5, pp. 683–697, 2022, doi: 10.1080/02602938.2021.1955241.

[6] M. Neupane Bastola and G. Hu, "Supervisory feedback across disciplines: Does it meet students' expectations?," *Assess. Eval. High. Educ.*, vol. 46, no. 3, 2021, doi: 10.1080/02602938.2020.1780562.

[7] Y. Zheng, S. Yu, B. Wang, and Y. Zhang, "Exploring student engagement with supervisor feedback on master's thesis: Insights from a case study," *Innov. Educ. Teach. Int.*, vol. 57, no. 2, pp. 186–197, 2020, doi: 10.1080/14703297.2019.1617181.

[8] Y. Nurie, "Graduate students' perceived needs and preferences for supervisor written feedback for thesis writing," *J. Lang. Educ.*, vol. 6, no. 4, pp. 153–170, Dec. 2020, doi: 10.17323/jle.2020.10340.

[9] Y. Nugraha, "Web-based thesis management information system design," *Sinkron: J. Penelit. Tek. Inform.*, vol. 7, no. 4, pp. 2602–2612, 2022, doi: 10.33395/sinkron.v7i4.12244.

[10] S. Suraya and M. Sholeh, "Designing and implementing a database for thesis data management by using the Python Flask framework," *Int. J. Eng. Sci. Inf. Technol.*, vol. 2, no. 1, 2022.

[11] F. Asrin and Y. Sholva, "User satisfaction assessment of final project supervisor monitoring information system using user experience design method," *J. Inf. Syst. Informatics*, vol. 4, no. 4, pp. 975–991, Nov. 2022, doi: 10.51519/journalisi.v4i4.368.

[12] Syukhri and Ganefri, "Expert analysis on the user interface of an academic e-supervision application based on vocational education character," *JOIV: Int. J. Informatics Vis.*, vol. 8, no. 2, pp. 598–605, May 2024.

[13] R. Kaliisa, I. Jivet, and P. Prinsloo, "A checklist to guide the planning, designing, implementation, and evaluation of learning analytics dashboards," *Int. J. Educ. Technol. High. Educ.*, vol. 20, 2023, Art. no. 28, doi: 10.1186/s41239-023-00394-6.

[14] G. Ramaswami, T. Susnjak, A. Mathrani, and R. Umer, "Use of predictive analytics within learning analytics dashboards: A review of case studies," *Technol. Knowl. Learn.*, vol. 28, no. 3, 2023, doi: 10.1007/s10758-022-09613-x.

[15] OWASP Foundation, "OWASP Top 10:2021," 2021. [Online]. Available: https://owasp.org/Top10/

[16] M. Noman *et al.*, "A survey on detection and prevention of web vulnerabilities," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 6, p. 521, 2020.

[17] W. W. Royce, "Managing the development of large software systems," in *Proc. IEEE WESCON*, Los Angeles, CA, USA, 1970, pp. 1–9.

[18] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2020.

[19] A. Demirağ, E. N. Demirkol Öztürk, and C. Ünal, "Analysis and comparison of waterfall model and agile approach in software projects," *Online Acad. J. Inf. Technol.*, vol. 14, no. 54, pp. 183–203, 2023, doi: 10.5824/ajite.2023.03.002.x.

[20] A. Mishra and Y. I. Alzoubi, "Structured software development versus agile software development: A comparative analysis," *Int. J. Syst. Assur. Eng. Manag.*, 2023, doi: 10.1007/s13198-023-01958-5.

[21] *Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—Product Quality Model*, ISO/IEC Standard 25010, 2023.

[22] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, Feb. 1996, doi: 10.1109/2.485845.

[23] Yuricha and I. K. Phan, "The implementation of role based access control in a cloud-based supply chain management system," *MALCOM: Indones. J. Mach. Learn. Comput. Sci.*, vol. 3, no. 2, pp. 339–348, Oct. 2023, doi: 10.57152/malcom.v3i2.1259.

[24] P. P.-S. Chen, "The entity-relationship model—Toward a unified view of data," *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, Mar. 1976, doi: 10.1145/320434.320440.

[25] W. G. J. Halfond, J. Viegas, and A. Orso, "A classification of SQL-injection attacks and countermeasures," in *Proc. IEEE Int. Symp. Secure Softw. Eng.*, Arlington, VA, USA, 2006, pp. 13–15.

[26] M. Nasereddin, A. ALKhamaiseh, M. Qasaimeh, and R. Al-Qassas, "A systematic review of detection and prevention techniques of SQL injection attacks," *Inf. Secur. J.: A Global Perspective*, vol. 32, no. 4, pp. 252–265, 2023, doi: 10.1080/19393555.2021.1995537.

[27] V. Abdullayev and A. S. Chauhan, "SQL injection attack: Quick view," *Mesopotamian J. CyberSecurity*, vol. 3, no. 1, 2023, doi: 10.58496/MJCS/2023/006.

[28] A. Hannousse, S. Yahiouche, and M. C. Nait-Hamoud, "Twenty-two years since revealing cross-site scripting attacks: A systematic mapping and a comprehensive survey," *Comput. Sci. Rev.*, vol. 52, 2024, Art. no. 100634, doi: 10.1016/j.cosrev.2024.100634.

[29] J. Kaur, U. Garg, and G. Bathla, "Detection of cross-site scripting (XSS) attacks using machine learning techniques: A review," *Artif. Intell. Rev.*, vol. 56, 2023, doi: 10.1007/s10462-023-10433-3.

[30] A. Barth, C. Jackson, and J. C. Mitchell, "Robust defenses for cross-site request forgery," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, 2008, pp. 75–88, doi: 10.1145/1455770.1455782.

[31] X. Likaj, S. Khodayari, and G. Pellegrino, "Where we stand (or fall): An analysis of CSRF defenses in web frameworks," in *Proc. 24th Int. Symp. Res. Attacks, Intrusions Defenses (RAID)*, San Sebastian, Spain, 2021.

[32] S. Khodayari and G. Pellegrino, "The state of the SameSite: Studying the usage, effectiveness, and adequacy of SameSite cookies," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, 2022.

[33] L. Compagna, H. Jonker, J. Krochewski, B. Krumnow, and M. Sahin, "A preliminary study on the adoption and effectiveness of SameSite cookies as a CSRF defence," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops*, Vienna, Austria, 2021.

[34] M. Squarcina, P. Adão, L. Veronese, and M. Maffei, "Cookie crumbles: Breaking and fixing web session integrity," in *Proc. 32nd USENIX Secur. Symp.*, Anaheim, CA, USA, 2023, pp. 5539–5556.

[35] T. Lee, S. Wi, S. Lee, and S. Son, "FUSE: Finding file upload bugs via penetration testing," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, 2020, doi: 10.14722/ndss.2020.23126.

[36] Y. Chen, Y. Li, Z. Pan, Y. Lu, J. Chen, and S. Ji, "URadar: Discovering unrestricted file upload vulnerabilities via adaptive dynamic testing," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, p. 1251, 2024.

[37] H. Oz *et al.*, "(In)Security of file uploads in Node.js," in *Proc. ACM Web Conf.*, Singapore, 2024, doi: 10.1145/3589334.3645342.

[38] N. Provos and D. Mazières, "A future-adaptable password scheme," in *Proc. USENIX Annu. Tech. Conf.*, Monterey, CA, USA, 1999, pp. 81–90.

[39] E. Ulutas and B. Celiktas, "Evaluation of password hashing competition finalists: Performance, security, compliance mapping, and post-quantum readiness," *Black Sea J. Eng. Sci.*, vol. 8, no. 6, pp. 1841–1855, 2025, doi: 10.34248/bsengineering.1670109.

[40] P. A. Grassi *et al.*, "Digital identity guidelines: Authentication and lifecycle management," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Spec. Publ. 800-63B, Jun. 2017, doi: 10.6028/NIST.SP.800-63b.

[41] A. Ali, M. Ahmed, and A. Khan, "Audit logs management and security: A survey," *Kuwait J. Sci.*, vol. 48, no. 3, 2021, doi: 10.48129/kjs.v48i3.10624.

[42] J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, Eds. London, U.K.: Taylor & Francis, 1996, pp. 189–194.

[43] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the System Usability Scale," *Int. J. Hum.-Comput. Interact.*, vol. 24, no. 6, pp. 574–594, 2008, doi: 10.1080/10447310802205776.

[44] P. Vlachogianni and N. Tselios, "Perceived usability evaluation of educational technology using the System Usability Scale (SUS): A systematic review," *J. Res. Technol. Educ.*, vol. 54, no. 3, pp. 392–409, 2022, doi: 10.1080/15391523.2020.1867938.

[45] P. Vlachogianni and N. Tselios, "Perceived usability evaluation of educational technology using the Post-Study System Usability Questionnaire (PSSUQ): A systematic review," *Sustainability*, vol. 15, no. 17, 2023, Art. no. 12954, doi: 10.3390/su151712954.