EduTask: Design and Development of a Secure Role-Based Web Application for Assignment Submission, Grading, and Learning Progress Visualization

H. A. Mooduto[a,*]

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

***Abstract*— Managing coursework through chat groups, e-mail, and cloud folders fragments the assignment cycle: deadlines are overlooked, files are misnamed or lost, and grades and feedback reach students late. This study designs, develops, and evaluates EduTask, a web-based assignment repository for the Department of Information Technology, Politeknik Negeri Padang, that unifies assignment publication, submission, grading, revision, notification, and reporting for three roles: administrator, lecturer, and student. The materials comprise the application source code (40 PHP 8 files; 30,760 lines), a 25-table MariaDB schema, and a staging dataset of 39 accounts, 12 assignments, and 150 submissions. The system was built through iterative prototyping on a three-tier architecture with Bootstrap 5 and Chart.js dashboards, and it embeds security controls mapped to the OWASP Top 10: role- and object-level authorization, prepared statements, CSRF tokens, bcrypt password hashing, account lockout, content-inspected file uploads, and an audit trail. Evaluation used 43 automated black-box scenarios covering authentication, assignment and grading workflows, submission rules, and report export. All 43 scenarios passed (100%), including rejection of SQL injection, forged requests, disguised executable uploads, stored cross-site scripting, and cross-lecturer data access; exports to PDF, XLSX, CSV, and iCalendar were structurally valid. Production schema counters indicate adoption of up to 210 accounts, 38 assignments, and 675 submissions. The results show that a framework-free PHP application can deliver a defensible assignment workflow whose on-screen analytics match its exported reports. Further research should measure usability and learning impact with end users, enforce idle-session expiry and multi-factor authentication, and add predictive early-warning analytics.**

***Keywords*— Assignment management; learning management system; role-based access control; web application security; black-box testing; learning analytics dashboard.**

I. INTRODUCTION

Learning management systems (LMSs) have become routine infrastructure in higher education, and reviews of their adoption explain continued use largely through perceived usefulness and ease of use [1], [2]. A steadily growing body of research documents how these platforms are used [3], and adoption has also been linked to the self-management habits of users [4]. In practice, however, many departments still run the assignment cycle through generic channels. Lecturers announce tasks in chat groups, students submit through e-mail or shared cloud links, and marks are returned in spreadsheets. Earlier development studies describe the consequences: students submit through cloud storage links supplied by each lecturer, and without an integrated application the process is time-consuming and inefficient [5]. File names follow no standard, and the status of each submission is hard to trace. Group-oriented tools were proposed for the same reason, namely that students lose track of who has completed which task and when it is due [6].

Three strands of evidence explain why this fragmentation matters. First, feedback is among the strongest influences on achievement, but only when it is specific and arrives while students can still act on it [7]. A pilot comparison in an online course found better grades when performance-gap feedback was combined with positive feedback [8], and analyses of personalized feedback messages show that students do engage with actionable messages delivered through the platform [9]. Second, submission timing is informative. A meta-analysis established a negative relationship between procrastination and academic performance [10], and large-scale studies based on LMS submission timestamps confirmed the association at the scale of whole institutions [11], [12]. Objective log indicators also predict late submission better than questionnaire data [13]. Third, visual summaries of these traces help both parties: reviews of learner-facing and teacher-facing dashboards report benefits for awareness and self-regulation, while warning that many dashboards remain exploratory, with limited evaluation of their impact [14], [15], [16]. Learning analytics studies likewise show that gradebook and activity data can flag students at risk early in a semester [17], [18], [19], [20].

A coursework system is also a security-sensitive application. It stores personal data and grades, accepts files from hundreds of users, and exposes forms that change academic records. The OWASP Top 10 places broken access control, cryptographic failures, and injection at the head of current web risks [21]. Reviews of SQL injection research agree that parameterized queries are the primary prevention technique [22], [23], [24]. Cross-site scripting remains one of the most frequently reported flaws more than two decades after its disclosure [25], [26], [27], and recent work continues to seek better detection [28]. Request forgery defenses are often incomplete even in mature frameworks, so their correct use depends on developer awareness [29], [30]. Unrestricted file upload is especially dangerous for PHP applications because an uploaded script may be executed by the server [31], [32], [33]. Finally, role-based access control (RBAC) is the accepted model for separating duties among user groups [34] and is widely applied in institutional web systems [35].

Existing reports on locally developed submission systems concentrate on functional delivery and user perception [5], [6]. They seldom document how the workflow rules are enforced on the server, how the security risks above are mitigated, or whether the figures shown on dashboards agree

with exported reports. General-purpose LMSs address many of these issues, but they bring a breadth of features and an administrative overhead that a single department does not always need.

This paper presents EduTask, a lightweight assignment repository built for the Department of Information Technology, Politeknik Negeri Padang. The study makes three contributions. First, it describes a complete role-based design for the assignment cycle, including explicit server-side rules for late submission, resubmission, and lecturer-initiated revision. Second, it documents a set of security controls mapped to the OWASP Top 10 and implemented without an application framework. Third, it reports an automated black-box evaluation of 43 functional and security scenarios, together with code metrics and the operational footprint of the deployed system. The remainder of the paper describes the materials and method, presents and discusses the results, and concludes with implications for further research.

II. Materials and Method

A. Research Design

The study follows a design-and-development approach organized as iterative prototyping [36], [37]. Fig. 1 summarizes the five stages. Requirements were derived from the way assignments are handled in the department. The design stage produced the architecture, the use-case model, the submission state model, and the relational schema. Implementation proceeded in successive releases: the source headers of the main modules record between two and six revisions, each of which lists the defects corrected after the preceding review. The evaluation stage therefore fed back into design and implementation, as indicated by the dashed path in Fig. 1.

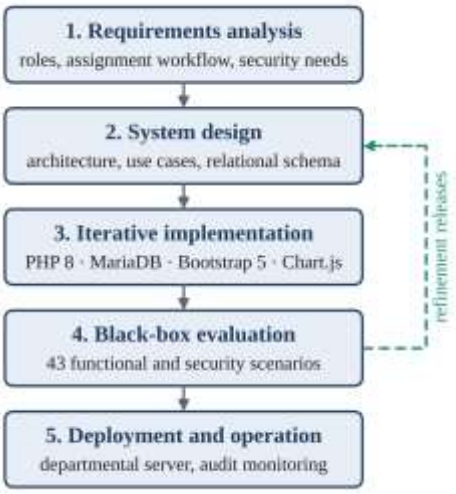


Fig. 1 Research stages of the EduTask development

The materials consist of three artifacts: (i) the application source code, comprising 40 PHP files and 30,760 lines excluding third-party libraries; (ii) the production database schema, exported from MariaDB 10.11 on 14 November 2026 and containing 25 tables; and (iii) a staging dataset generated for testing. No personal data from the production system were used in this study.

B. Requirements and Actors

Three actors interact with the system. Table I lists the principal functions available to each role. Accounts are created by the administrator; self-registration is disabled, and class membership is managed by the lecturer who teaches the class, either one student at a time or by importing a list of student numbers.

TABLE I
Principal Functions per Role

Role	Principal functions
Administrator	Manage users, courses, classes, and enrolment; global search; six system reports with period comparison; audit trail with filters; export to PDF, XLSX, CSV, and JSON
Lecturer	Create, duplicate, edit, and delete assignments with attachment, deadline, maximum score, and late/resubmission options; notify class; grading queue with quick and bulk grading; feedback; revision request; student notes and reminders; class and assignment reports
Student	Personal dashboard; prioritized assignment list; text and file submission; revision; submission history; grades with trend chart; deadline calendar with iCalendar export; notifications; profile and security settings

C. System Architecture

EduTask uses a three-tier architecture, shown in Fig. 2. The presentation tier is rendered on the server and styled with Bootstrap 5 and Bootstrap Icons; Chart.js draws the charts and Cropper.js supports profile-photo cropping. The layout is responsive, with an off-canvas sidebar on narrow screens. The application tier is written in PHP 8 without an application framework. It is divided into a public and authentication area and three role modules, which share a small set of libraries: common helper functions, a report builder for the administrator, a class-analytics builder for the lecturer, a native XLSX writer, and the mPDF 8.2 library for PDF output. The data tier consists of a MariaDB database on the InnoDB engine and a file store for submissions, assignment attachments, and profile photos. Every database connection sets the UTF-8 character set and aligns the session time zone of the database with that of PHP, so that deadlines are compared on a single clock.

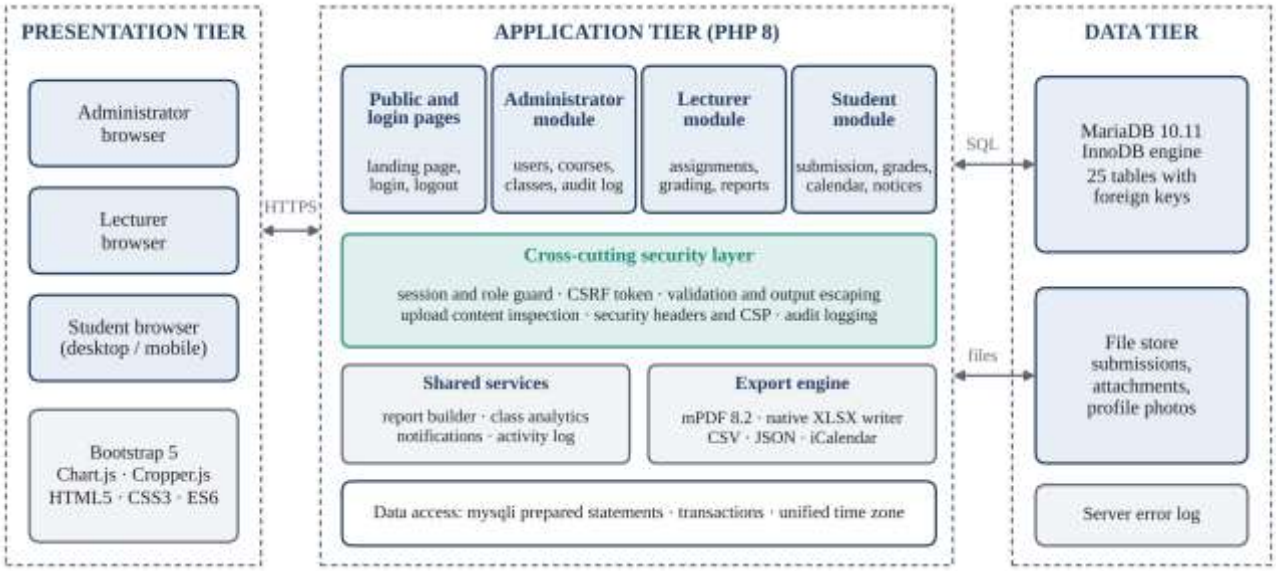


Fig. 2 Three-tier architecture of EduTask with the cross-cutting security layer

D. Functional Modelling

Fig. 3 presents the use-case model. All use cases include authentication. The central business object is the submission, whose life cycle is modelled as the state machine in Fig. 4. A student may submit before the deadline, or after it only when the lecturer has enabled late submission; a late submission is accepted and flagged. An ungraded submission may be replaced before the deadline if resubmission is enabled. Once a grade is recorded the submission is locked. The lecturer can reopen it by requesting a revision with a mandatory note; the previous score and feedback are then archived as a draft and shown again when the revised work is marked. A revision keeps the original submission time, so work revised at the request of the lecturer is not counted as late. These rules are evaluated on the server from stored data by a single decision function that is used both to render the form and to validate the request, which prevents the rules from being bypassed through crafted parameters.

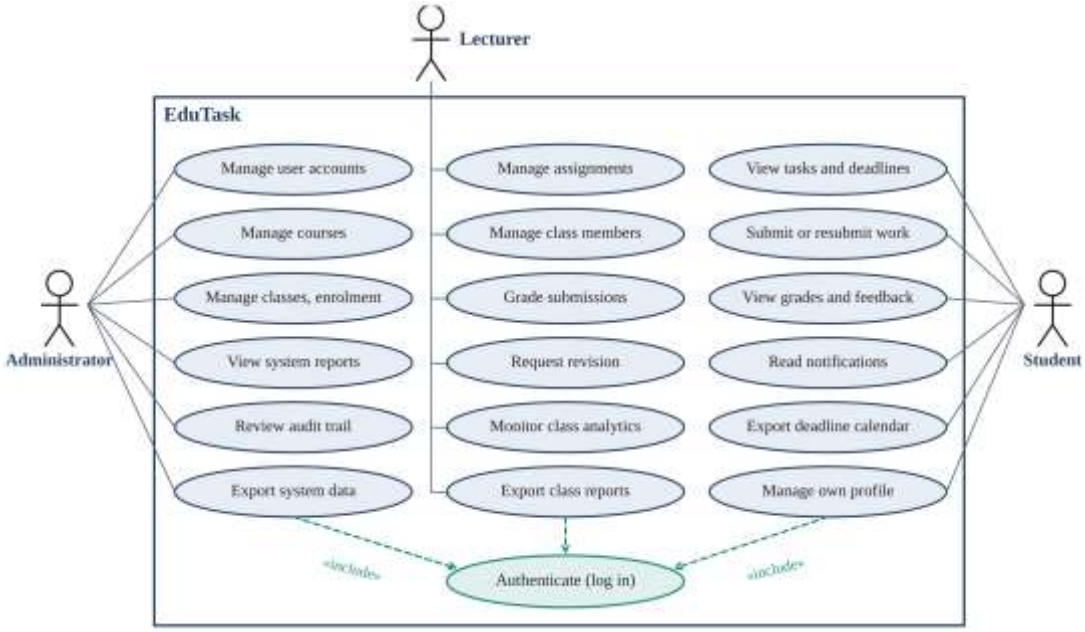


Fig. 3 Use-case diagram of EduTask

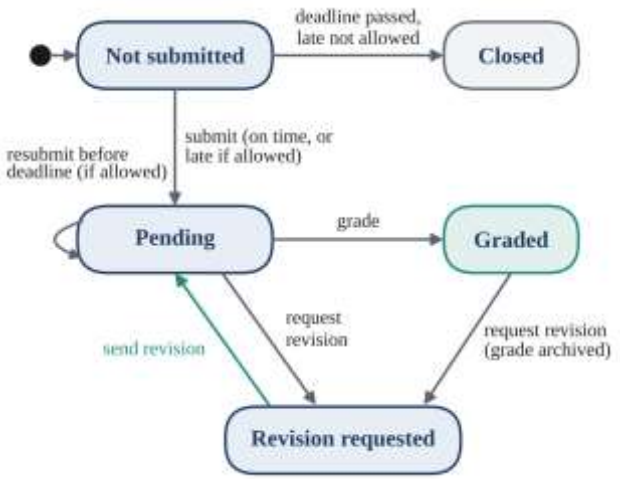


Fig. 4 State diagram of a submission

E. Data Model

The production schema contains 25 tables. Fig. 5 shows the eleven entities that carry the assignment workflow. A course is offered as one or more classes; each class has one lecturer and many students through the associative entity class_students. An assignment belongs to a class, and a unique key on the pair of assignment and student guarantees at most one submission per student per assignment. A submission has at most one grade, enforced by a unique key, and may have an archived draft grade. Foreign keys cascade deletions along the ownership chain and set authorship references to null when an account is removed. Notifications, the activity log, and the login history reference the user and support the notification center and the audit trail.

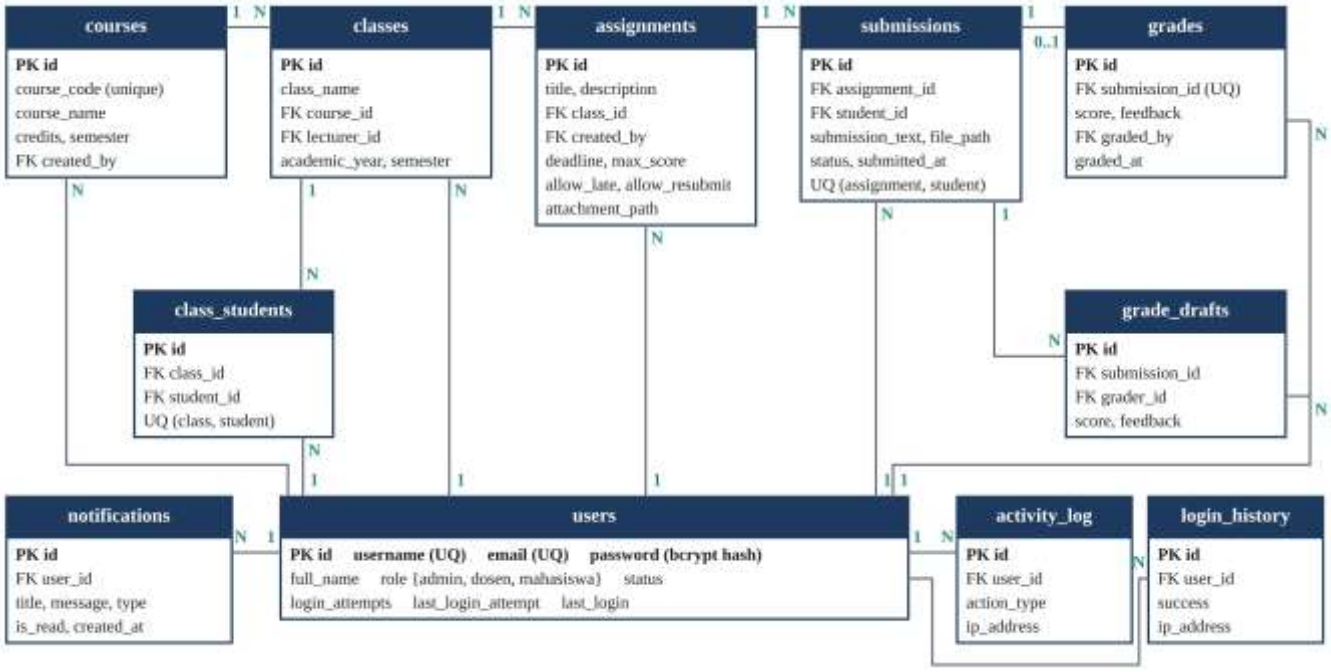


Fig. 5 Entity-relationship diagram of the core workflow tables (PK = primary key, FK = foreign key, UQ = unique key)

F. Security Design

Security requirements were derived from the OWASP Top 10 [21] and from the authentication guidance of NIST SP 800-63B [38]. Table II maps the implemented controls to the risk categories.

1) Authentication: Passwords are stored with the bcrypt adaptive hash [39] through the native PHP password interface. The login form returns one generic message for an unknown user name and for a wrong password, and it performs a dummy hash computation for unknown names so that response time does not reveal valid accounts. After five consecutive failures the account is locked for 15 minutes, in line with the throttling recommended in [38]. The session identifier is regenerated after a successful login, and the session cookie is flagged HttpOnly, SameSite, and Secure when the site is served over HTTPS.

2) Authorization: Access is decided at two levels. A role guard at the top of every page redirects users whose role does not match the module, following the RBAC model [34]. In addition, every query that reads or changes a class, assignment, submission, or grade is constrained to the objects owned by the current user: a lecturer can act only on classes that he or she teaches or on assignments that he or she created, and a student only on classes in which he or she is enrolled.

3) Input, output, and request integrity: Statements that carry user input are executed as prepared statements. Values are escaped on output, and free-text answers are rendered on the server as inert text. Every state-changing form carries a random 256-bit token that is compared in constant time, and forms follow the Post-Redirect-Get pattern. The login page sends a nonce-based Content Security Policy, a mechanism whose effectiveness depends on avoiding unsafe inline allowances [40].

4) File upload: Answer files are limited to 10 MB and to an allow-list of thirteen document, archive, and image extensions. The declared extension must agree with the

content type detected from the file itself. Images are decoded and re-encoded, which discards any embedded payload, and stored files receive a random prefix. Files are never linked directly; they are served by a script that first verifies ownership and then forces a safe content type.

TABLE II
SECURITY CONTROLS MAPPED TO OWASP TOP 10:2021

Risk category	Control implemented in EduTask
A01 Broken access control	Role guard on every page; ownership constraint in every object query; files served only after an access check
A02 Cryptographic failures	Bcrypt password hashing; Secure, HttpOnly, SameSite session cookie; HSTS header under HTTPS
A03 Injection	Prepared statements; output escaping; server-side rendering of user text; CSV cells neutralized against formula injection
A04 Insecure design	Single server-side decision function for submission rules; transactions for grading and revision
A05 Security misconfiguration	Error display disabled; security headers (frame, sniffing, referrer); nonce-based CSP on the login page
A07 Identification and authentication failures	Generic login message; lockout after five failures for 15 minutes; session identifier regeneration; password change throttling
A08 Software and data integrity failures	CSRF token on every state-changing form; upload allow-list, content inspection, and image re-encoding
A09 Logging and monitoring failures	Activity log and login history with address and user agent; administrator audit view and export

G. Learning Progress Indicators

To keep assignments with different maximum scores comparable, every score is normalized to a percentage. For a score s on an assignment with maximum score $s_{\max}$, the normalized score is

$$p = \frac{s}{s_{max}} \times 100 \quad (1)$$

Letter grades are derived from p with fixed bands: A for $p \geq 85$, B for 70–84, C for 55–69, D for 40–54, and E below 40. Class indicators consider only active students who are still enrolled and only assignments whose deadline has passed. With N students, T ended assignments, and n_{on} and n_{late} on-time and late submissions, the submission rate and the punctuality rate are

$$R = \frac{n_{on} + n_{late}}{N \times T} \times 100 \quad (2)$$

$$P = \frac{n_{on}}{n_{on} + n_{late}} \times 100 \quad (3)$$

A student is flagged as needing attention when the mean normalized score is below 55% or when two or more ended assignments are missing. The same computation feeds the on-screen report and every export format, so that exported figures always equal the displayed ones.

H. Evaluation Procedure

The functional suitability and security characteristics of ISO/IEC 25010 [41] were evaluated by black-box testing. Test cases were designed with equivalence partitioning, boundary value analysis, and state-transition techniques as defined in ISO/IEC/IEEE 29119-4 [42], and were grouped into four areas: (A) authentication and access control, (B) assignment and grading by the lecturer, (C) submission by the student, and (D) reporting, export, and audit. Negative cases were derived from the risks in Table II.

The tests were executed in an isolated staging environment consisting of PHP 8.3.6, MariaDB 10.11.14, and the Chromium browser automated with Playwright 1.63. The production schema was loaded unchanged and populated with a synthetic dataset of 39 accounts (one administrator, two lecturers, and 36 students, one of them suspended), three courses, three classes with 52 enrolments, 12 assignments, 150 submissions, and 106 grades. Each scenario was scripted as HTTP requests or browser actions, and its outcome was verified both from the response and from the resulting database state. In addition, static metrics were collected from the source code, and the auto-increment counters in the production schema export were read as upper bounds on the number of records created during operation.

III. RESULTS AND DISCUSSION

A. Implementation Results

The implemented system comprises 33 pages in the three role modules and the authentication area, supported by six shared library and configuration files and a public landing page. Table III gives the size of each part. The lecturer and administrator modules account for three quarters of the code, which reflects the weight of grading, analytics, and reporting in the workflow. Static pattern matching identified 328 parameterized-query call sites, counting direct prepared statements and the helper functions that wrap them.

TABLE III
SIZE OF THE EDUTASK SOURCE CODE

Part	Files	Lines	Parameterized queries
Administrator module	9	11,956	89
Lecturer module	11	11,371	139
Student module	11	4,951	50
Authentication	2	424	6
Shared libraries and configuration	6	1,720	43
Public landing page	1	338	1
Total	**40**	**30,760**	**328**

Fig. 6 shows four representative screens rendered with the staging dataset. The lecturer dashboard (Fig. 6a) opens with the grading queue ordered by waiting time, the deadlines of the next 14 days, and the submission rate of the current semester. The grading workspace (Fig. 6b) places the answer beside the scoring panel, offers score shortcuts and reusable feedback phrases, and moves to the next ungraded submission after saving. The student submission page (Fig. 6c) states the remaining time, whether late submission and resubmission are permitted, and the accepted formats before the student writes or uploads an answer. The class report (Fig. 6d) summarizes the submission rate, punctuality, mean score, and the number of students needing attention, followed by a trend chart per assignment and the grade distribution.

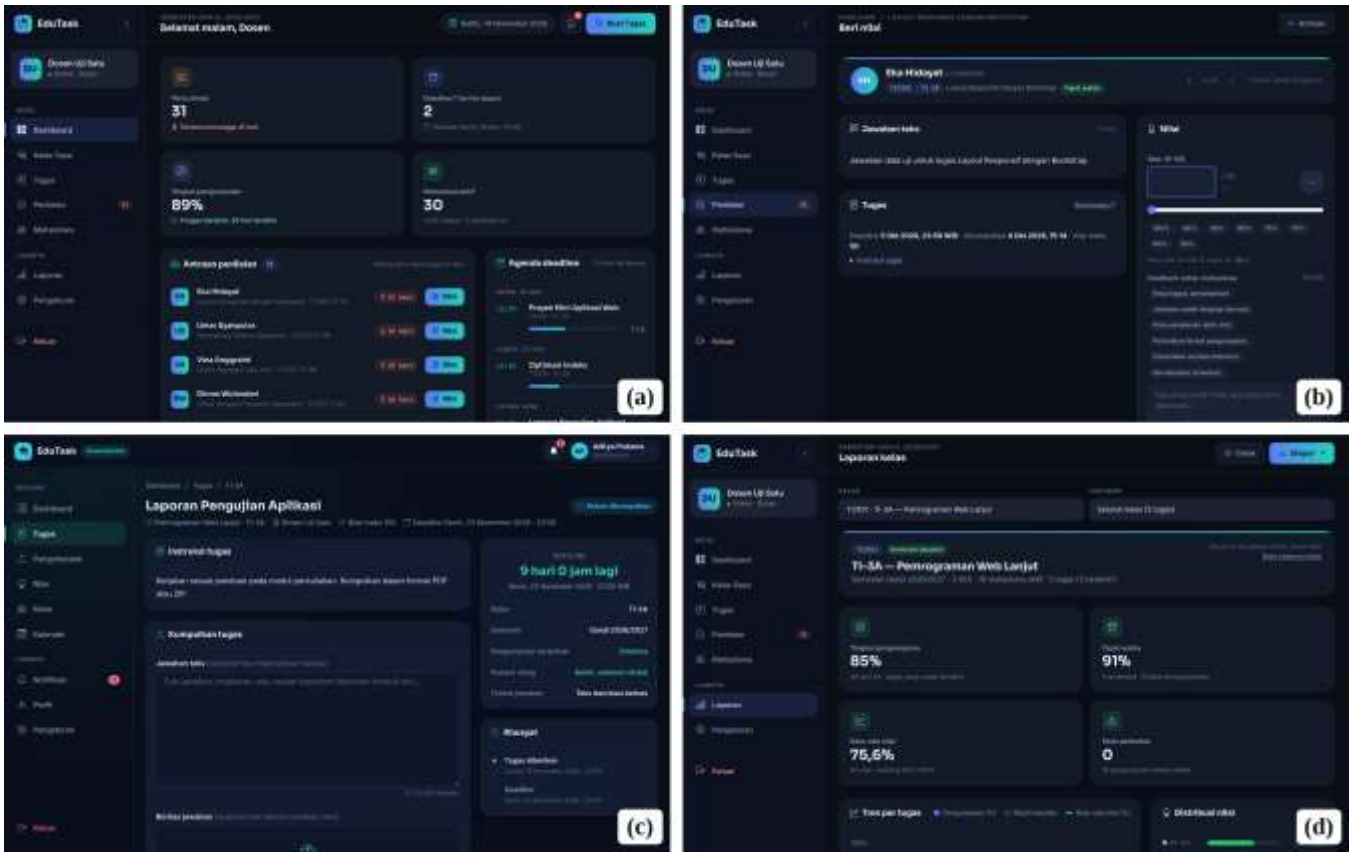


Fig. 6 User interface of EduTask with staging data: (a) lecturer dashboard, (b) grading workspace, (c) student submission page, (d) class report

B. *Functional and Security Testing*

All 43 scenarios produced the expected result, giving a pass rate of 100%. Fig. 7 shows the distribution of scenarios over the four areas, and Table IV lists twenty representative cases with the observed behavior.

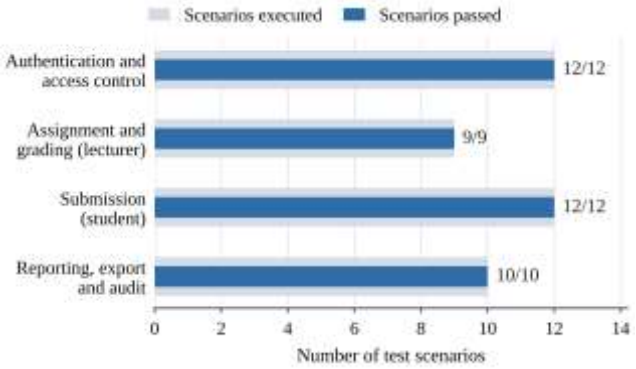


Fig. 7 Black-box test scenarios executed and passed per area

In area A, valid credentials led each role to its own dashboard, whereas requests for pages of another role and unauthenticated requests were redirected. The sixth attempt on a locked account was refused even with the correct password, and a suspended account could not log in. In area B, the notification option created exactly one notification for each of the 18 active members of the class. Server-side validation rejected a deadline in the past, a score above the maximum, and a request without a token. A second lecturer could neither grade nor read a submission that belonged to the first lecturer, which confirms object-level authorization rather than role checking alone.

In area C, the upload filter rejected a script file, a script disguised with an image extension and header, an oversized file, and an empty submission, while a valid document was stored under a randomized name and could be downloaded only by its owner. The deadline rules behaved as modelled in Fig. 4: a late submission was refused when late submission was disabled, accepted and marked late when enabled, and a graded answer could not be altered. A script payload placed in the answer text was displayed literally in the lecturer browser and did not execute. In area D, the Excel, PDF, CSV, and iCalendar outputs were structurally valid, a cell beginning with an equals sign was neutralized in CSV, and export endpoints refused requests from other roles and for classes of other lecturers. The audit trail recorded the successful and failed logins, grading, revision, assignment creation, and export actions generated during the session.

TABLE IV
REPRESENTATIVE BLACK-BOX TEST SCENARIOS AND RESULTS

ID	Scenario	Expected result	Observed result	Status
A5	Login with an unregistered user name	Same generic message as for a wrong password	Identical message; no account disclosure	Pass
A6	SQL injection payload in the login form	Rejected; no session created	Rejected with the generic message	Pass

ID	Scenario	Expected result	Observed result	Status
A7	Login request without CSRF token	Rejected	Form-expired message; no session	Pass
A8	Correct password after five failed attempts	Account locked for 15 minutes	Refused with remaining lock time	Pass
A11	Student requests administrator and lecturer pages	Access denied	Redirected to the student dashboard	Pass
B1	Create assignment with the notify option	Stored; all active class members notified	Stored; 18 notifications for 18 students	Pass
B4	Create assignment for a class of another lecturer	Rejected	Validation message; nothing stored	Pass
B5	Score above the maximum (55 of 50)	Rejected	Rejected; no grade stored	Pass
B7	Request revision of a graded submission	Status changed; previous grade archived	Status "revised"; one archived draft	Pass
B8	Lecturer grades a submission of another lecturer	Rejected	Not-found/no-access message; no grade	Pass
C2	Upload a .php file as an answer	Rejected by extension allow-list	Rejected; nothing stored	Pass
C3	Upload PHP code named foto.jpg with a GIF header	Rejected by content inspection	Content does not match extension	Pass
C4	Upload an 11 MB PDF (limit 10 MB)	Rejected by size limit	Rejected with the limit stated	Pass
C8	Submit after the deadline, late not allowed	Rejected	Rejected; nothing stored	Pass
C9	Submit after the deadline, late allowed	Accepted and recorded as late	Stored; submission time after deadline	Pass
C10	Submit to an assignment of a class not enrolled in	Denied	Redirected; nothing stored	Pass
C12	Script payload in the answer, viewed by the lecturer	Shown as text; not executed	Displayed literally; no execution	Pass
D1	Export class report to Excel	Valid XLSX workbook	ZIP container with workbook part	Pass
D3	CSV export of a name beginning with "="	Cell neutralized	Value prefixed with an apostrophe	Pass
D5	Export student deadlines to iCalendar	Valid calendar file	VCALENDAR with nine events	Pass

C. Reporting and Visualization

Reports are produced at three levels. The administrator obtains six reports (system summary, users, courses, assignments, grades, and activity) with indicators compared against the preceding period and time series whose scale adapts to the selected range. The lecturer obtains a class report and an assignment report; the latter adds the distribution of submission timing in four intervals, from more than three days before the deadline to late. The student sees the trend of normalized scores, the distribution of letter grades, and progress per course. Reports can be exported to PDF, XLSX, CSV, and, for the administrator, JSON; deadlines can be exported in the iCalendar format [43] for use in external calendar applications. Because the screen and the export share one computation, the exported figures matched the displayed figures in every test.

D. Operational Footprint

EduTask has been deployed on a departmental server. Table V reports the auto-increment counters contained in the production schema export of 14 November 2026. A counter gives the number of identifiers issued and is therefore an upper bound on the number of records created, since deleted or rolled-back rows also consume identifiers. The counters indicate use at the scale of a department: up to 210 accounts, 19 classes of 12 courses, 38 assignments, 675 submissions, and 575 recorded grades, the latter corresponding to about 85% of the submission identifiers.

TABLE V
IDENTIFIER COUNTERS IN THE PRODUCTION SCHEMA (UPPER BOUNDS)

Table	Counter	Meaning
users	210	Accounts created
courses	12	Courses registered
classes	19	Classes opened
class_students	393	Enrolments recorded
assignments	38	Assignments published
submissions	675	Submissions received
grades	575	Grades recorded
notifications	537	Notifications generated
login_history	280	Login attempts logged
activity_log	1,411	Audit entries written

E. Discussion

The results address the three problems identified in the introduction. The first is traceability. Earlier systems replaced ad hoc file links with a standard submission point and reported that students and lecturers found the process simpler [5]; EduTask adds explicit, server-enforced rules for lateness, resubmission, and revision, so that the state of every submission is unambiguous and auditable. The second is timeliness of feedback. The grading queue, the save-and-continue workflow, and the automatic grade notification are intended to shorten the interval between submission and feedback, which the literature identifies as a condition for feedback to be useful [7], [9]. The system now records the timestamps needed to measure that interval, although this study did not measure it. The third is visibility of progress. The class report exposes submission timing and missing work, the behavioral signals that large-scale studies associate with performance [11], [12], and it flags students with a transparent rule. Published early-warning models reach useful accuracy from gradebook and activity data within the first weeks of a course [18], [19]; the data collected by EduTask would allow such models to be trained locally and compared with the present rule.

With respect to security, the tests confirm that the principal controls operate as designed against the attack classes that dominate the literature: injection [22], [24], cross-site scripting [25], request forgery [29], and

unrestricted file upload [31], [33]. Two observations deserve emphasis. Object-level authorization, tested by letting one lecturer address the objects of another, is a property that role checks alone do not provide and that corresponds to the highest-ranked OWASP category [21]. Moreover, several controls were introduced during the refinement releases after review had revealed weaknesses in earlier versions, such as stored script injection through answer text and profile uploads saved under their submitted extension. This supports the view that framework-free applications can be made defensible, but only with deliberate review, because nothing is provided by default [29], [30].

The study has limitations. The evaluation was functional and was run on synthetic data in a staging environment; a pass rate of 100% shows conformance to the specified scenarios and does not prove the absence of other defects. Usability was not measured. A systematic review of 104 studies reported a mean System Usability Scale score of about 70 for educational technology [44], and a comparable system reported 70.44 [6]; these values provide a benchmark for a planned survey of lecturers and students. Code inspection also identified items for the next release: the idle-session limit defined in the configuration is not yet enforced by the role modules; columns for two-factor authentication exist but the feature is not implemented; explicit security headers are sent by 19 of the 33 pages; seven tables are not referenced by the current code; the core tables use a single-byte character set, so that characters outside it are rejected at input; and the interface depends on libraries loaded from a public content delivery network.

IV. Conclusion

This study designed, developed, and evaluated EduTask, a role-based web application that brings assignment publication, submission, grading, revision, notification, and reporting into one workflow for a polytechnic department. The system is implemented in PHP 8 and MariaDB on a three-tier architecture, enforces its submission rules on the server, and applies controls mapped to the OWASP Top 10. In an automated black-box evaluation, all 43 functional and security scenarios passed, including the rejection of injection, forged requests, disguised executable uploads, stored scripts, and access to the data of another lecturer, and all export formats were valid. Schema counters from the deployed system indicate use by up to 210 accounts with 675 submissions.

The findings imply that a small, framework-free application can provide a traceable and defensible assignment cycle, provided that authorization is checked at the level of individual objects and that one computation serves both the dashboards and the exports. Further research should (i) evaluate usability and acceptance with lecturers and students, (ii) use the recorded timestamps to measure whether feedback time and on-time submission improve, (iii) complete idle-session expiry, multi-factor authentication, and uniform security headers, followed by penetration testing, and (iv) extend the rule-based attention flag with predictive early-warning analytics.

Acknowledgment

The authors thank the Department of Information Technology, Politeknik Negeri Padang, for providing the server infrastructure, and the lecturers and students who used EduTask in their classes.

References

[1] N. Cavus, B. Omonayajo, and M. R. Mutizwa, "Technology acceptance model and learning management systems: Systematic literature review," *Int. J. Interact. Mob. Technol.*, vol. 16, no. 23, pp. 109–124, Dec. 2022, doi: 10.3991/ijim.v16i23.36223.

[2] A. F. Maluleke and G. Maake, "Examining the impact of learning management systems in African higher education: A systematic review," *Afr. J. Inter/Multidiscip. Stud.*, vol. 7, no. 1, pp. 1–15, Mar. 2025, doi: 10.51415/ajims.v7i1.1550.

[3] H. Altinpulluk and M. Kesim, "A systematic review of the tendencies in the use of learning management systems," *Turkish Online J. Distance Educ.*, vol. 22, no. 3, pp. 40–54, 2021, doi: 10.17718/tojde.961812.

[4] S. Balkaya and U. Akkucuk, "Adoption and use of learning management systems in education: The role of playfulness and self-management," *Sustainability*, vol. 13, no. 3, 2021, Art. no. 1127, doi: 10.3390/su13031127.

[5] S. B. Mahlan, J. Othman, M. Shamsuddin, S. A. Mohamed Yusoff, and Z. Othman, "A web-based system for managing student attendance and assessment submissions: An SDLC waterfall model approach," *J. Comput. Res. Innov.*, vol. 10, no. 2, pp. 130–143, 2025.

[6] M. L. H. Akram, R. Aminuddin, Z. Abu, and N. S. Muhammat Pazil, "Group assignment management system (GAMS) for university students," *Int. J. Acad. Res. Progress. Educ. Dev.*, vol. 13, no. 1, Jan. 2024, doi: 10.6007/IJARPED/v13-i1/19125.

[7] J. Hattie and H. Timperley, "The power of feedback," *Rev. Educ. Res.*, vol. 77, no. 1, pp. 81–112, Mar. 2007, doi: 10.3102/003465430298487.

[8] E. Faulconer, J. Griffith, and A. Gruss, "The impact of positive feedback on student outcomes and perceptions," *Assess. Eval. High. Educ.*, Apr. 2021, doi: 10.1080/02602938.2021.1910140.

[9] H. Iraj *et al.*, "Narrowing the feedback gap: Examining student engagement with personalized and actionable feedback messages," *J. Learn. Anal.*, vol. 8, no. 3, pp. 101–116, Nov. 2021, doi: 10.18608/jla.2021.7184.

[10] K. R. Kim and E. H. Seo, "The relationship between procrastination and academic performance: A meta-analysis," *Pers. Individ. Differ.*, vol. 82, pp. 26–33, Aug. 2015, doi: 10.1016/j.paid.2015.02.038.

[11] S. H. Cormack, L. A. Eagle, and M. S. Davies, "A large-scale test of the relationship between procrastination and performance using learning analytics," *Assess. Eval. High. Educ.*, vol. 45, no. 7, pp. 1046–1059, 2020.

[12] S. Sabnis, R. Yu, and R. F. Kizilcec, "Large-scale student data reveal sociodemographic gaps in procrastination behavior," in *Proc. 9th ACM Conf. Learn. @ Scale (L@S '22)*, New York, NY, USA, Jun. 2022, doi: 10.1145/3491140.3528285.

[13] C. Imhof, I.-S. Comsa, M. Hlosta, B. Parsaeifard, I. Moser, and P. Bergamin, "Prediction of dilatory behavior in eLearning: A comparison of multiple machine learning models," *IEEE Trans. Learn. Technol.*, vol. 16, no. 5, pp. 648–663, Oct. 2023, doi: 10.1109/TLT.2022.3221495.

[14] N. Valle, P. Antonenko, K. Dawson, and A. C. Huggins-Manley, "Staying on target: A systematic literature review on learner-facing learning analytics dashboards," *Br. J. Educ. Technol.*, vol. 52, no. 4, pp. 1724–1748, Jul. 2021, doi: 10.1111/bjet.13089.

[15] 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, no. 1, May 2023, Art. no. 28, doi: 10.1186/s41239-023-00394-6.

[16] R. Bodily and K. Verbert, "Review of research on student-facing learning analytics dashboards and educational recommender systems," *IEEE Trans. Learn. Technol.*, vol. 10, no. 4, pp. 405–418, Oct.–Dec. 2017, doi: 10.1109/TLT.2017.2740172.

[17] D. Ifenthaler and J. Y.-K. Yau, "Utilising learning analytics to support study success in higher education: A systematic review," *Educ.*

Technol. Res. Dev., vol. 68, no. 4, pp. 1961–1990, Aug. 2020, doi: 10.1007/s11423-020-09788-z.

[18] J. B. Osborne and A. S. I. D. Lang, "Predictive identification of at-risk students: Using learning management system data," *J. Postsecond. Student Success*, vol. 2, no. 4, 2023, doi: 10.33009/fsop_jpss132082.

[19] R. M. Santos and R. Henriques, "Accurate, timely, and portable: Course-agnostic early prediction of student performance from LMS logs," *Comput. Educ. Artif. Intell.*, vol. 5, 2023.

[20] Y.-H. Hu, "Using few-shot learning materials of multiple SPOCs to develop early warning systems to detect students at risk," *Int. Rev. Res. Open Distrib. Learn.*, vol. 23, no. 1, pp. 1–20, 2022, doi: 10.19173/irrodl.v22i4.5397.

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

[22] M. Hallo and G. Suntaxi, "A survey on SQL injection attacks, detection and prevention techniques – a tertiary study," *Int. J. Secur. Netw.*, vol. 17, no. 3, 2022.

[23] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Detection of SQL injection attack using machine learning techniques: A systematic literature review," *J. Cybersecur. Priv.*, vol. 2, no. 4, pp. 764–777, Sep. 2022, doi: 10.3390/jcp2040039.

[24] 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. Glob. Perspect.*, Oct. 2021, doi: 10.1080/19393555.2021.1995537.

[25] 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.

[26] D. Korać, B. Damjanović, D. Simić, and K.-K. R. Choo, "A hybrid XSS attack (HYXSSA) based on fusion approach: Challenges, threats and implications in cybersecurity," *J. King Saud Univ. – Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9284–9300, Nov. 2022, doi: 10.1016/j.jksuci.2022.09.008.

[27] S. J. Y. Weamie, "Cross-site scripting attacks and defensive techniques: A comprehensive survey," *Int. J. Commun. Netw. Syst. Sci.*, vol. 15, no. 8, pp. 126–148, 2022, doi: 10.4236/ijcns.2022.158010.

[28] K. H. Hamzah, M. Z. Osman, T. Anthony, M. A. Ismail, Z. Abdullah, and A. Alanda, "Comparative analysis of machine learning algorithms for cross-site scripting (XSS) attack detection," *JOIV: Int. J. Inform. Visualization*, vol. 8, no. 3-2, pp. 1678–1685, Nov. 2024, doi: 10.62527/joiv.8.3-2.3451.

[29] 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 '21)*, San Sebastian, Spain, Oct. 2021.

[30] K. Peguero and X. Cheng, "CSRF protection in JavaScript frameworks and the security of JavaScript applications," *High-Confid. Comput.*, vol. 1, no. 2, Dec. 2021, Art. no. 100035, doi: 10.1016/j.hcc.2021.100035.

[31] 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.

[32] J. Huang, Y. Li, J. Zhang, and R. Dai, "UChecker: Automatically detecting PHP-based unrestricted file upload vulnerabilities," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2019, pp. 581–592, doi: 10.1109/DSN.2019.00064.

[33] S. Neef and M. Oudeh, "Bringing UFUs back into the air with FUEL: A framework for evaluating the effectiveness of unrestricted file upload vulnerability scanners," 2024, *arXiv:2405.16619*.

[34] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, Aug. 2001, doi: 10.1145/501978.501980.

[35] 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.

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

[37] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA, USA: Pearson, 2016.

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

[39] N. Provos and D. Mazières, "A future-adaptable password scheme," in *Proc. USENIX Annu. Tech. Conf. (FREENIX Track)*, Monterey, CA, USA, 1999, pp. 81–91.

[40] S. Roth, T. Barron, S. Calzavara, N. Nikiforakis, and B. Stock, "Complex security policy? A longitudinal analysis of deployed content security policies," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, 2020.

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

[42] *Software and Systems Engineering—Software Testing—Part 4: Test Techniques*, ISO/IEC/IEEE Standard 29119-4:2021, 2021.

[43] B. Desruisseaux, Ed., "Internet calendaring and scheduling core object specification (iCalendar)," Internet Eng. Task Force, RFC 5545, Sep. 2009, doi: 10.17487/RFC5545.

[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.