Design and Development of a Secure Web-Based E-Book Repository for Vocational Higher Education Using a Lightweight MVC Architecture

H. A. Mooduto[a,*]

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

Abstract*— Lecturers in vocational higher education increasingly produce teaching books and practicum modules in digital form, yet these files are often scattered across personal storage and messaging groups, which hampers discovery, version control, and usage monitoring. This study aims to design, build, and verify a web-based e-book repository for the Department of Information Technology, Politeknik Negeri Padang, that is light enough to run on modest infrastructure and secure by default. The system was developed through five stages (requirements analysis, design, implementation, verification, and deployment) using PHP 8 with a purpose-built model-view-controller core, MariaDB 10.11, and Tailwind CSS. It provides role-based services for guests, members, librarians, and administrators; a three-stage search strategy that combines ISBN prefix matching, FULLTEXT boolean retrieval, and an escaped LIKE fallback; content-based PDF validation with SHA-256 duplicate detection; Argon2id password hashing; CSRF tokens; atomic rate limiting; a strict Content Security Policy; and audit logging. The materials comprise 27 departmental e-books (175.8 MB, 6,023 pages) and a synthetic catalogue of up to 100,000 records. All 34 functional and 11 security test cases passed. Median server response time was below 5 ms for every main page, and FULLTEXT retrieval was 2.1 to 99 times faster than LIKE scanning at 100,000 records while returning relevance-ranked results. The findings show that a dependency-light architecture can deliver a secure and responsive repository for a small academic unit. Further research should measure usability with students and lecturers, add metadata interoperability such as OAI-PMH, and evaluate behaviour under concurrent production load.

***Keywords*— E-book repository; digital library; model-view-controller; web application security; full-text search; vocational higher education.**

I. INTRODUCTION

Electronic books have become a routine part of study in higher education. Surveys of students and academic staff report that convenience of access, searchability, and availability at any time are the main reasons for choosing e-books over print [1], and that e-book use is associated with better academic achievement when institutions make collections easy to reach [2]. Adoption, however, still depends on infrastructure, awareness, and the ease with which titles can be found [3]. For vocational institutions the issue is sharper, because much of the reading material is produced in-house: lecturers write teaching books, practicum modules, and job sheets that follow the local curriculum, and these electronic modules have been shown to support independent, practice-oriented learning [4], [5].

In practice, such materials are frequently distributed through messaging groups, personal cloud folders, or learning-management-system attachments tied to a single class. Students cannot search across courses, lecturers cannot tell which version is current, and the department has no record of how often a title is read. Reviews of open educational resources note the same gap: research concentrates on the creation of content, while the usability and upkeep of the repositories that hold it receive much less attention [6], [7].

Institutional repositories were proposed two decades ago as essential infrastructure for managing and disseminating the digital output of a university [8]. A systematic review of 115 studies found that their main benefits are visibility and support for teaching and research, whereas inadequate information technology infrastructure and limited awareness are the dominant obstacles [9]. In Indonesia, repositories are widespread but are run mostly at university level and are oriented toward theses and published articles [10]. Full-featured platforms require server resources and specialist administration that a single department rarely has, and they are not designed around teaching materials that are revised every semester.

A department-level repository is therefore attractive, but it brings its own risk. A small in-house web application that accepts file uploads and stores user accounts is exposed to the same threats as any larger system. Broken access control, cryptographic failures, and injection lead the OWASP Top 10 [11]; SQL injection and cross-site scripting remain common despite years of research on detection and mitigation [12], [13], [14]; and file upload handlers are a recurring source of critical vulnerabilities [15]. Mainstream PHP frameworks address many of these concerns, but they add a large dependency tree and measurable request overhead [16], [17], and their defences still depend on developers enabling and using them correctly [18].

This paper reports the design and development of the E-Book Repository of the Department of Information Technology, Politeknik Negeri Padang. The work makes three contributions. First, it describes a dependency-light model-view-controller (MVC) architecture, with one third-party runtime package, in which security controls are part of the request pipeline and cannot be bypassed by an individual controller. Second, it presents a three-stage catalogue search that combines ISBN prefix matching, FULLTEXT boolean retrieval, and a safe pattern-matching fallback, together with content-hash duplicate detection for uploaded files. Third, it provides a reproducible verification of the system: 45 scripted functional and security test cases and performance measurements on the real departmental collection and on synthetic catalogues of up to 100,000 records. The remainder of the paper describes the materials and method, presents

and discusses the results, and closes with conclusions and directions for further work.

II. Materials and Method

A. Research Stages

Development followed the five stages shown in Fig. 1, adapted from the generic communication, planning, modelling, construction, and deployment framework of software engineering [19]. Defects found during verification were fed back into implementation until every test case passed. Requirements were derived from the way teaching materials are currently shared in the department and from an inspection of the existing e-book collection, which determined the file sizes and formats the system must handle.

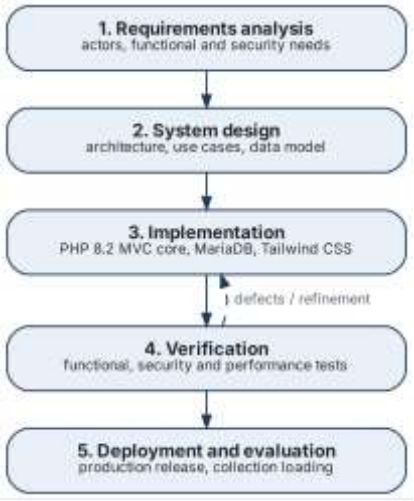


Fig. 1 Stages of the research

B. Materials

The application was written in PHP (version 8.2 or later is required; version 8.3.6 was used for testing) and uses MariaDB 10.11 with the InnoDB engine and Tailwind CSS 3.4 for the interface. Its only third-party runtime package is a loader for environment configuration; routing, session handling, access control, file handling, and data access were written for the project. The code base comprises 5,159 lines of PHP in 17 controllers, 8 models, 5 middleware classes, 11 core classes, and 31 view templates, and it exposes 44 routes.

Two data sets were used. The first is the working collection of the department: 27 e-books in PDF [20], consisting of teaching books, practicum modules, reference books, and final-project reports, with a total size of 175.8 MB and 6,023 pages (1.6 MB to 14.4 MB per file). The second is a synthetic catalogue for search benchmarking. Its vocabulary was extracted from pages 5 to 40 of the 27 real books; titles of four to eight words and descriptions of 40 to 60 words were sampled from that vocabulary with a fixed random seed, and the table was grown to 1,000, 10,000, 50,000, and 100,000 records with the same schema and indexes as the production table.

C. Requirements and Actors

Four actors were identified, as shown in Fig. 2. A guest may browse and search the catalogue and read book details. A member, after registration, may preview and download files, bookmark titles, rate and review them, and see a personal download history. A librarian manages books, categories, and reviews and can export reports. An administrator additionally manages user accounts and inspects the audit log. Each role inherits the permissions of the role above it in the figure, following the hierarchical role-based access control model [21]. Table I summarises the resulting permission matrix. Non-functional requirements were derived from the security, performance efficiency, and functional suitability characteristics of the ISO/IEC 25010 product quality model [22].

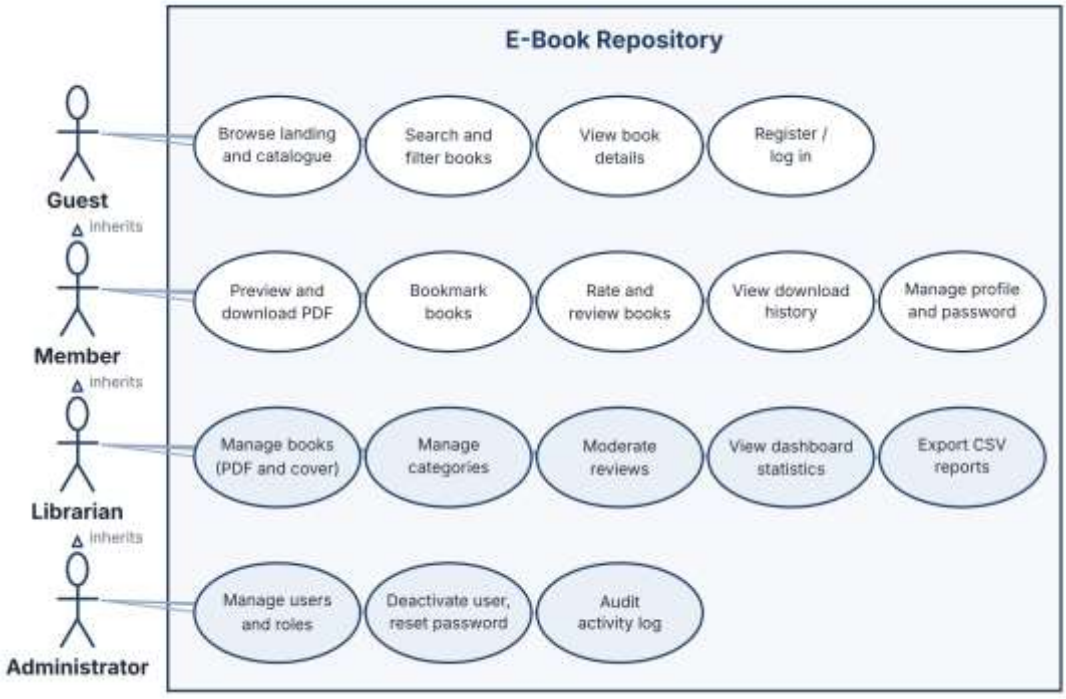


Fig. 2 Use case diagram of the E-Book Repository

TABLE I
PERMISSION MATRIX BY ROLE

Function	G	M	L	A
Browse, search, and view book details	✓	✓	✓	✓
Preview and download PDF	–	✓	✓	✓
Bookmark, rate, review, download history	–	✓	✓	✓
Manage books and categories	–	–	✓	✓
Moderate reviews, view statistics, export CSV	–	–	✓	✓
Manage users, roles, and account status	–	–	–	✓
Inspect activity (audit) log	–	–	–	✓

G = guest, M = member, L = librarian, A = administrator.

D. System Architecture

Fig. 3 shows the architecture. Every request enters through a single front controller [23], which loads configuration, applies two global middleware classes (HTTPS enforcement in production and security response headers), starts a hardened session, and hands the request to the router. The router matches the HTTP method and path against the route table and runs the middleware declared for that route (authentication, role check, and CSRF verification) before the controller is instantiated. Controllers coordinate models, which access the database only through prepared statements, and views, which escape every dynamic value on output. Because protection is declared in the route table, the access policy of the whole application can be reviewed in one file, and a controller cannot be reached without passing the checks attached to its route. Uploaded e-books are stored outside the reach of direct URLs: the web server denies requests for PDF files, and downloads and previews are streamed by a controller after the session has been checked.



Fig. 3 System architecture and request pipeline

E. Data Model

The database consists of seven tables (Fig. 4). The books table holds bibliographic metadata, the stored file name, the file size, and the SHA-256 digest of the file; a unique index on the digest prevents the same file from being registered twice. Bookmarks and ratings carry a composite unique key on user and book, so that a user holds at most one bookmark and one rating per title, and a CHECK constraint restricts scores to 1 to 5. Deleting a book cascades to its bookmarks, ratings, and download log, whereas deleting a category or a user sets the corresponding reference to null so that history is preserved. A FULLTEXT index covers title, author, and description.

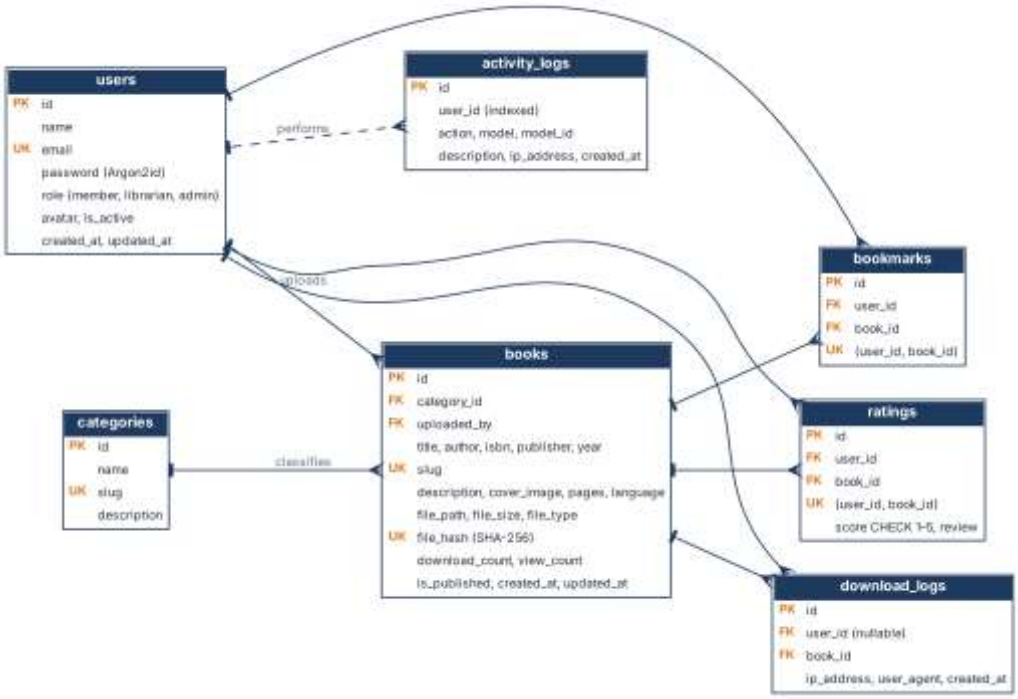


Fig. 4 Entity relationship diagram of the database

TABLE II
SECURITY CONTROLS MAPPED TO OWASP TOP 10:2021 CATEGORIES

Risk category	Control	Implementation
A01 Broken access control	Role-based access control declared per route; server-side streaming of files	AuthMiddleware and RoleMiddleware in the route table; role re-validated against the database every 300 s; PDF files not directly addressable
A02 Cryptographic failures	Memory-hard password hashing; transport security	Argon2id with automatic rehash; HTTPS redirect and HSTS in production; Secure cookie flag
A03 Injection	Parameterised queries; output encoding; content policy	PDO prepared statements with emulation disabled; escaped LIKE wildcards; HTML entity encoding in views; CSP script-src 'self'
A04 Insecure design	Rate limiting; content-based upload validation; duplicate detection	File-locked counters for login, registration, and rating; MIME and magic-byte checks; unique SHA-256 digest; random stored file names
A05 Security misconfiguration	Hardened defaults	Errors logged, not displayed, in production; X-Frame-Options, X-Content-Type-Options, Referrer-Policy, Permissions-Policy headers
A07 Identification and authentication failures	Session hardening; anti-forgery tokens	Session ID regenerated at login and every 30 min; HttpOnly and SameSite=Lax cookie; 256-bit CSRF token on all POST routes
A09 Logging and monitoring failures	Audit trail	activity_logs and download_logs tables with user, action, object, and network address; health-check endpoint

F. Search Strategy

Catalogue search must serve three kinds of query: an ISBN typed or scanned by a librarian, ordinary keywords, and very short terms such as "AI" or "C#". A single mechanism does not handle all three. InnoDB full-text indexes ignore tokens shorter than three characters by default [24], and combining a full-text predicate with a pattern predicate through OR prevents the index from being used. The search was therefore designed as the three-stage cascade in Fig. 5. A query that looks like an ISBN is matched as a prefix of the ISBN column. Otherwise, if every token has at least three characters, the query is rewritten in boolean mode so that each token is required and treated as a prefix (for example, +jaringan* +komputer*), and results are ordered by the relevance score of the inverted index [25]. If a stage returns no rows, or a token is too short, the search falls back to substring matching on title, author, ISBN, and description, with wildcard characters in the user input escaped. All three stages use bound parameters.

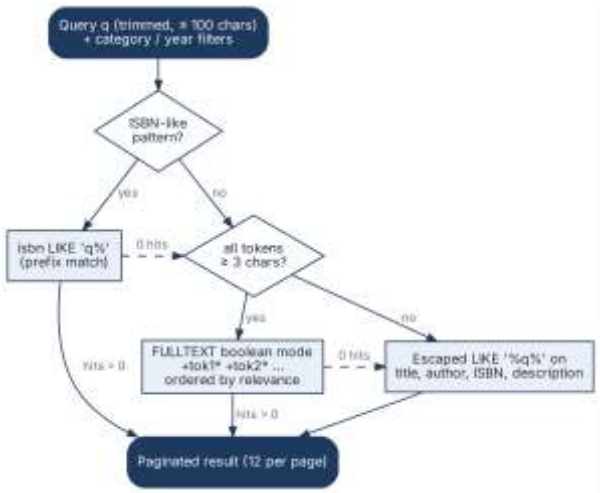


Fig. 5 Three-stage catalogue search

G. Security Design

Security requirements were taken from the OWASP Top 10 [11] and the Application Security Verification Standard [26]. Table II maps each risk category to the control implemented. The main design decisions are as follows.

1) Authentication and session: Passwords are hashed with Argon2id [27], a memory-hard function, and are transparently rehashed at login if parameters change. In line with current guidance on throttling online guessing [28], failed logins are limited per network address (eight attempts in five minutes) and per account (five attempts in ten minutes); registration and rating submissions are limited as well. Counters are kept in files that are updated under an exclusive lock, so that simultaneous requests cannot both pass the limit. The session identifier is regenerated at login and every 30 minutes, the cookie is flagged HttpOnly and SameSite=Lax [29], [30], and the account status and role stored in the session are re-read from the database at most every 300 seconds so that deactivation by an administrator takes effect without waiting for logout.

2) Request forgery and script injection: Every state-changing route requires a synchroniser token [31] of 32 random bytes that is compared in constant time; a missing or wrong token ends the request with status 419. Output is escaped with HTML entity encoding, and a Content Security Policy restricts scripts to the application's own origin with no inline code, which limits the effect of any injection that escaping might miss [32].

3) File upload: An uploaded e-book is accepted only if its size is within the configured limit (50 MB), its content type as detected from the file content is PDF, and its first bytes carry the PDF signature; the client-supplied name and extension are ignored. The SHA-256 digest [33] of the content is computed before the file is moved, and the upload is rejected if the digest already exists, which is whole-file deduplication [34]. Accepted files are stored under a random 128-bit name. These checks address the upload weaknesses catalogued by Lee et al. [15].

H. Verification Procedure

Verification was scripted so that it can be repeated after every change. The application was deployed on a test server (two virtual CPU cores of an Intel Xeon 2.1 GHz processor, 7 GB of memory, Ubuntu 24.04, PHP 8.3.6, MariaDB 10.11.14) and loaded with the 27 real e-books and accounts for each role. Functional test cases were designed with black-box techniques, namely equivalence partitioning of valid and invalid inputs and use-case testing [35], and were executed over HTTP by a script that also inspected the database and the file store after each action. Security test cases reproduced common attacks from the OWASP Top 10: requests without or with forged tokens, password guessing, injection payloads, stored script injection, and privilege escalation.

Response time was measured at the client for seven representative pages; each page was requested 100 times in sequence after five warm-up requests, and the median and 95th percentile were recorded. Search was benchmarked by executing, for each catalogue size and query, the same pair of statements the application issues (a count and a page of 12 rows) 30 times after two warm-up runs. The speed-up of full-text retrieval over pattern matching is expressed as

$$S = \frac{t_{LIKE}}{t_{FT}} \tag{1}$$

where t_{LIKE} and t_{FT} are the median query times of pattern matching and full-text retrieval, respectively. Three queries were used: a rare term, a two-term query, and a frequent term. The cost of password hashing and of file digest computation was measured in the same environment.

III. Results and Discussion

A. Implementation

Fig. 6 shows the main screens of the system. The landing page presents the newest titles and collection statistics; the catalogue offers keyword search, category filters, and pagination of 12 titles per page; the detail page shows metadata, counters, preview and download buttons, and the rating and review section; and the dashboard gives librarians and administrators totals, downloads over the last seven days, the most popular categories and titles, and recent activity. Reports of books, downloads, and users can be exported as CSV files. Registration can be configured without code changes as open, restricted to institutional e-mail domains, subject to administrator approval, or closed. The interface is responsive and was built without client-side frameworks: the application ships two small scripts and no inline code, which is what allows the strict script policy described above.

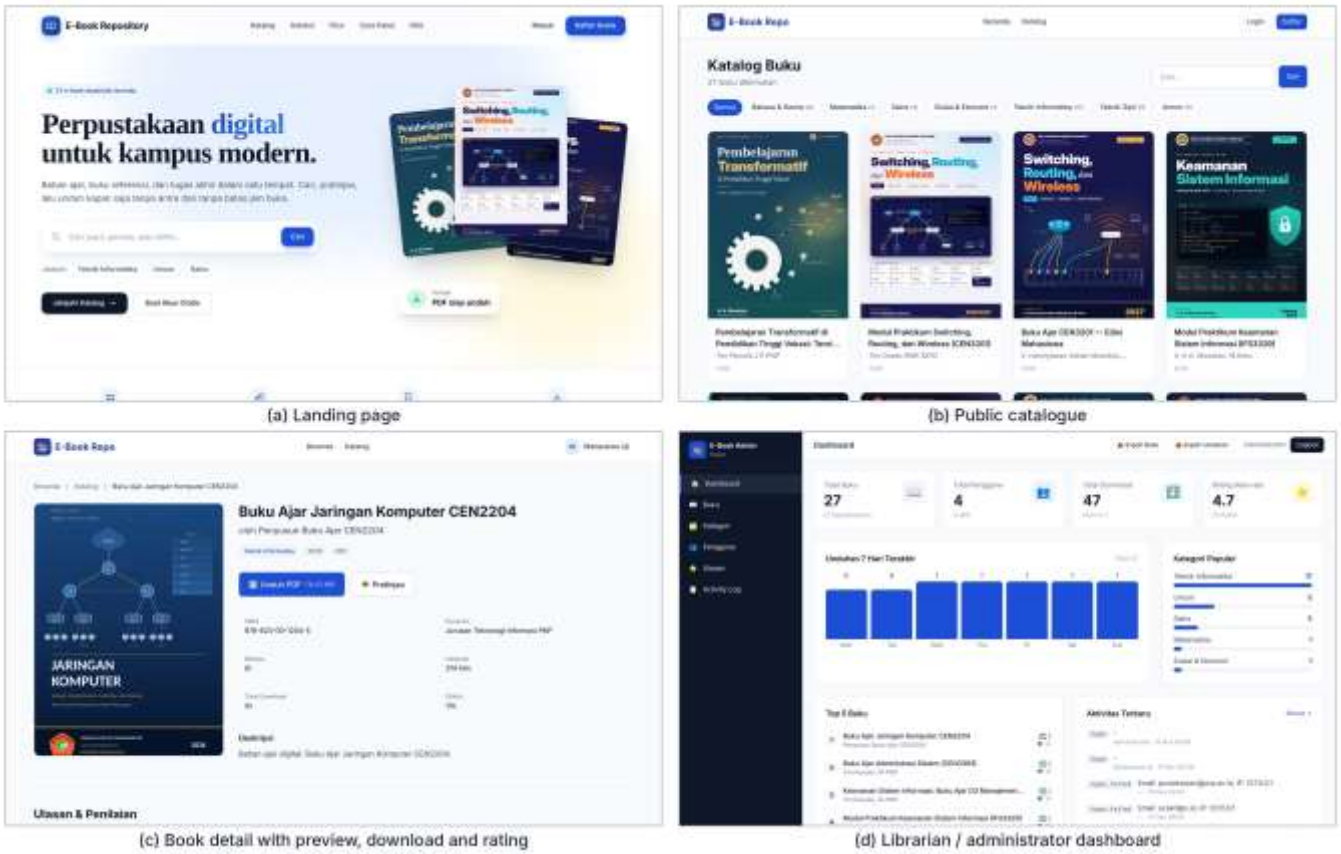


Fig. 6 Implemented interface: (a) landing page, (b) public catalogue, (c) book detail, and (d) dashboard for librarians and administrators

B. Functional Testing

All 34 functional test cases produced the expected result (Table III). Beyond checking the page returned, the test script verified side effects. A member download returned the complete 14.4 MB file as an attachment, incremented the download counter, and added exactly one row to the download log, whereas the same request from a guest was redirected to the login page. Submitting a second rating for the same title updated the existing row instead of creating another. Uploading a valid PDF stored the file under a random name with a 64-character digest; uploading the same file again under a different title was rejected with a message naming the existing book; and a script file renamed with a .pdf extension was rejected by content inspection. Deleting a book removed both the database row and the stored file. The librarian account could open book management but received status 403 for user management and the activity log, which confirms the boundary between the two staff roles.

TABLE III
RESULTS OF FUNCTIONAL (BLACK-BOX) TESTING

Module	Cases	Passed
Public catalogue and search (F01–F09)	9	9
Registration and authentication (F10–F14, F21, F34)	7	7
Member services: download, preview, bookmark, rating, history (F15–F20)	6	6
Role-based access (F22–F24)	3	3
Collection management: category, upload, edit, delete (F25–F30)	6	6
User administration, export, and audit (F31–F33)	3	3
Total	**34**	**34 (100%)**

C. Security Testing

Table IV lists the security test cases and the behaviour observed; all 11 passed. Two observations deserve comment. First, the two rate limits act independently: guessing passwords for one account was stopped at the sixth attempt, and trying a different account on each attempt from one address was stopped at the ninth. During the lock-out, the correct password was refused as well, which denies an attacker confirmation of a successful guess; the cost is that a legitimate user is locked out for up to ten minutes, a deliberate trade-off for a service that holds no time-critical function. Second, injection payloads submitted through the search box were handled as ordinary text: the pages returned

normally, a time-delay payload caused no delay, and the catalogue table was unchanged.

TABLE IV
RESULTS OF SECURITY TESTING

ID	Attack scenario	Expected behaviour	Observed behaviour	Result
S01	POST request without CSRF token	Rejected	HTTP 419	Pass
S02	POST request with forged CSRF token	Rejected	HTTP 419	Pass
S03	Password guessing against one account	Locked after 5 failed attempts	Attempts 1–5 refused as wrong; attempts 6–7 locked	Pass
S04	Correct password submitted during lock-out	Still refused	Refused with temporary-lock message	Pass
S05	Credential stuffing: 10 accounts from one address	Blocked after 8 attempts	Attempts 1–8 refused as wrong; attempts 9–10 blocked	Pass
S06	Four SQL injection payloads in search (tautology, UNION, time delay, stacked DROP)	No error, no delay, data intact	HTTP 200 for all; no measurable delay; 27 books remain	Pass
S07	Stored XSS payload in a review	Rendered as text	Payload returned entity-encoded	Pass
S08	Security response headers	Five headers present	X-Frame-Options, X-Content-Type-Options, Referrer-Policy, Permissions-Policy, Content-Security-Policy	Pass
S09	Session cookie attributes	HttpOnly and SameSite	HttpOnly; SameSite=Lax	Pass
S10	Unauthenticated export of user data	Denied	HTTP 302 to login page	Pass
S11	Member sends admin delete request (vertical privilege escalation)	Denied; book intact	HTTP 403; book intact	Pass

Argon2id hashing with the default parameters of the runtime (64 MiB of memory, four iterations) took a median of 202 ms per password, against 58 ms for bcrypt at its default cost. The delay is paid once per login and is not noticeable to a user, while it raises the cost of offline guessing should the user table ever leak [27]. Computing SHA-256 digests for the whole collection (175.8 MB) took 0.78 s, about 225 MB/s, so duplicate detection adds well under a tenth of a second to the upload of the largest file in the collection.

D. Performance

Table V reports server response time. The median was between 1.5 ms and 4.3 ms for every page, and the 95th percentile did not exceed 5.3 ms, including the dashboard, which runs more than a dozen aggregate queries. These figures were obtained on the loopback interface with one client, so they represent the processing cost of the application and not the latency a remote user would experience. They are nevertheless informative when set against published measurements of full-stack PHP frameworks, where simple create-read-update-delete requests take tens of milliseconds [16], [17]: a request pipeline that loads only the classes it needs leaves ample headroom on modest hardware.

TABLE V
SERVER RESPONSE TIME OVER 100 REQUESTS PER PAGE

Page	Size (kB)	Median (ms)	P95 (ms)
Landing page	63.5	1.75	2.08
Catalogue (page 1)	16.5	3.34	3.90
Search, two keywords	6.1	3.47	4.34
Book detail	11.9	3.86	4.43
Login form	3.2	1.45	1.99
Admin dashboard	22.1	4.33	5.28
Health check	0.1	1.97	2.39

Fig. 7 compares the two search mechanisms as the catalogue grows. Pattern matching scans every row, so its time rises in proportion to the number of records, from between 4 ms and 8 ms at 1,000 records to between 363 ms and 478 ms at 100,000. Full-text retrieval depends mainly on the number of matching documents. For the rare term it stayed almost flat (1.0 ms to 4.3 ms), a speed-up of 99 at 100,000 records by (1); for the two-term query the speed-up was 8.4 (57 ms against 478 ms); and for the frequent term, which matches about one record in five, it was 2.1 (170 ms against 363 ms), because counting and ranking more than 21,000 hits dominates the cost. The methods also differ in what they return. For the two-term query, substring matching found only the 48 records in which the two words are adjacent, whereas the boolean query found 2,541 records containing both words anywhere and ordered them by relevance. Full-text retrieval is thus both faster and more useful, and the fallback is reached only for short tokens or when the index finds nothing.

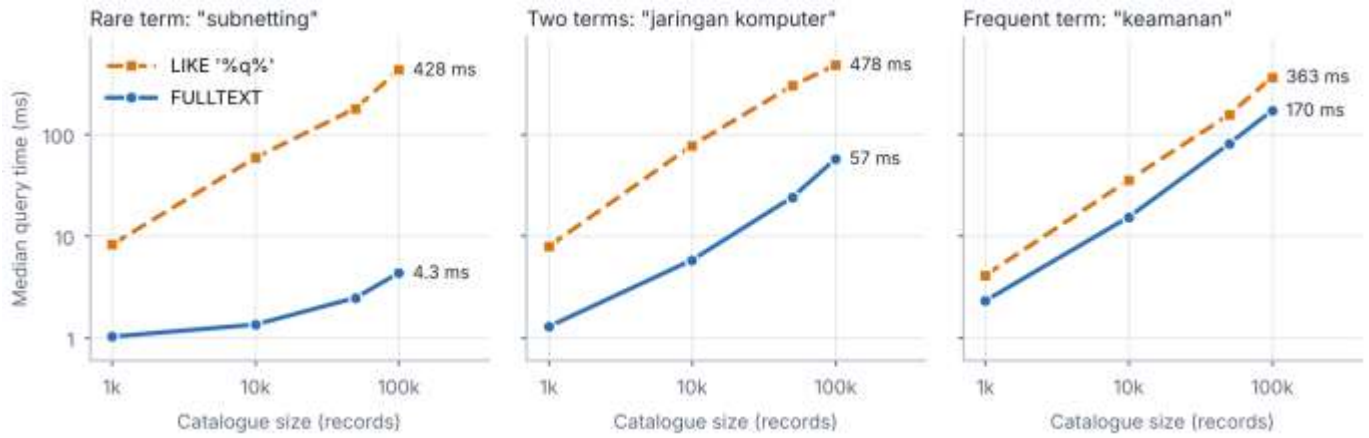


Fig. 7 Median query time of FULLTEXT boolean retrieval and LIKE pattern matching for three query types as the catalogue grows (logarithmic axes)

E. Discussion

The results support the premise that a small academic unit can run its own repository without adopting a heavyweight platform, provided that security is designed into the request pipeline. Earlier reviews identified infrastructure and technical capacity as the main barriers to repository adoption [9]; the system described here needs only a standard PHP and MariaDB host and one configuration file. At the same time, the study answers the concern that in-house applications trade safety for simplicity. Analyses of web frameworks have shown that built-in defences are often left disabled or misused [18], and that new variants of request forgery keep appearing on the client side [36]. Declaring authentication, role, and token checks in the route table makes an omission visible in review, and the scripted test suite turns each control in Table II into a regression test.

The search design reflects the nature of a teaching collection. Titles contain course codes, acronyms, and short technical terms, and librarians search by ISBN. A cascade that tries the most specific interpretation first and degrades gracefully gave correct results for all three query types in testing, and the benchmark indicates that the catalogue could grow by three orders of magnitude before search time becomes perceptible. Content-hash deduplication suits such a collection as well: a file that is submitted again under a different title is caught regardless of its name, while a revised edition, whose content differs, is accepted as a new item.

The study has limitations. Tests ran against a single-node deployment with one client; behaviour under concurrent load and on the production web server was not measured, and the web-server rule that blocks direct access to PDF files was not part of the scripted tests. The rate limiter stores counters in local files and would need a shared store if the application were scaled to several nodes. The repository does not yet expose standard metadata for harvesting, and files are served whole, without byte-range support for resuming large downloads. Most importantly, the evaluation was technical: acceptance and usability among students and lecturers, which the literature identifies as decisive for sustained e-book use [1], [3], remain to be measured.

IV. CONCLUSION

This study designed, built, and verified a web-based e-book repository for a vocational higher-education department. The system implements role-based services for guests, members, librarians, and administrators on a purpose-built MVC core with a single third-party runtime dependency. Security controls covering access control, password storage, injection, request forgery, upload validation, rate limiting, and audit logging are enforced in the request pipeline. All 34 functional and 11 security test cases passed on a deployment loaded with the department's 27 e-books. Median server response time stayed below 5 ms for every main page, and the three-stage search answered queries on a 100,000-record catalogue 2.1 to 99 times faster through the full-text index than through pattern matching, while returning relevance-ranked results.

The implication is that departments can manage their own teaching materials securely on modest infrastructure, complementing university-level repositories that focus on research output. Further research should (i) evaluate usability and acceptance with students and lecturers using standard instruments, (ii) add Dublin Core metadata and an OAI-PMH interface so that the collection can be harvested by institutional and national aggregators, (iii) measure performance under concurrent load in production and move rate-limit counters to a shared store, and (iv) extend retrieval from catalogue metadata to the full text of the e-books.

ACKNOWLEDGMENT

The authors thank the Department of Information Technology, Politeknik Negeri Padang, for supporting this study and for the teaching materials used as test data.

REFERENCES

[1] T. Hackman et al., "E-book perceptions and use: A longitudinal follow-up study," *portal: Libr. Acad.*, vol. 22, no. 3, pp. 681–723, Jul. 2022, doi: 10.1353/pla.2022.0036.

[2] E. S. Gbesoevi, M. L. Oyelami, B. A. Ola, and O. G. Jinodu, "E-books utilisation and students' academic achievement in universities in Lagos State, Nigeria: Implication for educational planners," *Int. J. Educ. Best Pract.*, vol. 6, no. 2, pp. 134–143, Oct. 2022, doi: 10.32851/ijebp.v6n2.p134-143.

[3] M. Singh and S. Kumar, "Narrowing the digital gap: Factors affecting e-book adoption in Delhi's higher education institutions," *Ann. Libr.*

Inf. Stud., vol. 72, no. 1, pp. 78–87, Mar. 2025, doi: 10.56042/alis.v72i1.13749.

[4] N. Marlena, F. D. Patrikha, and R. Dwijayanti, "Electronic modules in an Indonesian higher education: Conceptualisation, development and application," *AL-ISHLAH: J. Pendidik.*, vol. 14, no. 3, pp. 3943–3954, 2022, doi: 10.35445/alishlah.v14i3.1473.

[5] Mahzuardi, Refdinal, Ambiyar, and H. Maksum, "Development of e-module based teaching materials in the subject of ship main procovery machinery," *J. Penelit. Pendidik. IPA*, vol. 10, no. 10, pp. 8030–8041, Oct. 2024, doi: 10.29303/jppipa.v10i10.8644.

[6] D. Otto, N. Schroeder, D. Diekmann, and P. Sander, "Trends and gaps in empirical research on open educational resources (OER): A systematic mapping of the literature from 2015 to 2019," *Contemp. Educ. Technol.*, vol. 13, no. 4, 2021, Art. no. ep325, doi: 10.30935/cedtech/11145.

[7] S. Mishra, "A review of reviews on open educational resources," *Open Praxis*, vol. 17, no. 2, pp. 305–325, 2025, doi: 10.55982/openpraxis.17.2.851.

[8] C. A. Lynch, "Institutional repositories: Essential infrastructure for scholarship in the digital age," *portal: Libr. Acad.*, vol. 3, no. 2, pp. 327–336, Apr. 2003.

[9] S. Asadi, R. Abdullah, Y. Yah, and S. Nazir, "Understanding institutional repository in higher learning institutions: A systematic literature review and directions for future research," *IEEE Access*, vol. 7, pp. 35242–35263, 2019, doi: 10.1109/ACCESS.2019.2897729.

[10] D. E. Irawan, J. Abraham, R. A. Zein, I. A. Ridlo, and E. K. Aribowo, "Open access in Indonesia," *Develop. Change*, vol. 52, no. 3, pp. 651–660, 2021, doi: 10.1111/dech.12637.

[11] *OWASP Top 10:2021*, OWASP Foundation, 2021. [Online]. Available: https://owasp.org/Top10/. Accessed on: Nov. 15, 2026.

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

[13] G. E. Rodríguez, J. G. Torres, P. Flores, and D. E. Benavides, "Cross-site scripting (XSS) attacks and mitigation: A survey," *Comput. Netw.*, vol. 166, Jan. 2020, Art. no. 106960, doi: 10.1016/j.comnet.2019.106960.

[14] 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.

[15] 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.

[16] P. Gajewski and M. Dzieńkowski, "Comparison of Laravel and Symfony - the most popular PHP frameworks, based on a simple CRUD application," *J. Comput. Sci. Inst.*, vol. 31, pp. 132–137, Jun. 2024.

[17] M. K. Ahmed, A. H. Bello, S. S. Jauro, and M. Dawaki, "A comparative analysis of performance optimization techniques for benchmarking PHP frameworks: Laravel and CodeIgniter," *Dutse J. Pure Appl. Sci.*, vol. 10, no. 3c, Sep. 2024, doi: 10.4314/dujopas.v10i3c.27.

[18] 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, Oct. 2021.

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

[20] *Document Management - Portable Document Format - Part 2: PDF 2.0*, ISO 32000-2:2020, 2020.

[21] 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.

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

[23] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley, 2002.

[24] "Full-text index overview," MariaDB Knowledge Base. [Online]. Available: https://mariadb.com/kb/en/full-text-index-overview/. Accessed on: Nov. 15, 2026.

[25] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[26] *Application Security Verification Standard 4.0.3*, OWASP Foundation, Oct. 2021. [Online]. Available: https://owasp.org/www-project-application-security-verification-standard/. Accessed on: Nov. 15, 2026.

[27] A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson, "Argon2 memory-hard function for password hashing and proof-of-work applications," RFC 9106, Internet Res. Task Force, Sep. 2021, doi: 10.17487/RFC9106.

[28] D. Temoshok et al., "Digital identity guidelines: Authentication and authenticator management," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST SP 800-63B-4, Jul. 2025, doi: 10.6028/NIST.SP.800-63B-4.

[29] S. Calzavara, R. Focardi, M. Squarcina, and M. Tempesta, "Surviving the web: A journey into web session security," *ACM Comput. Surv.*, vol. 50, no. 1, 2017, Art. no. 13, doi: 10.1145/3038923.

[30] 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.

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

[32] 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, doi: 10.14722/ndss.2020.23046.

[33] *Secure Hash Standard (SHS)*, FIPS PUB 180-4, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Aug. 2015, doi: 10.6028/NIST.FIPS.180-4.

[34] W. Xia et al., "A comprehensive study of the past, present, and future of data deduplication," *Proc. IEEE*, vol. 104, no. 9, pp. 1681–1710, Sep. 2016, doi: 10.1109/JPROC.2016.2571298.

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

[36] S. Khodayari and G. Pellegrino, "JAW: Studying client-side CSRF with hybrid property graphs and declarative traversals," in *Proc. 30th USENIX Secur. Symp.*, Aug. 2021, pp. 2525–2542.