AbsensiQR: Design and Implementation of a Web-Based Lecture Attendance System Using Session-Bound QR Tokens and Layered Server-Side Validation

H. A. Mooduto[a,*]

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

***Abstract*— Recording lecture attendance on paper consumes class time, invites transcription errors, and leaves little evidence when a record is disputed. This study aims to design, implement, and evaluate AbsensiQR, a web-based attendance system for the Department of Information Technology, Politeknik Negeri Padang, that needs no hardware beyond the devices lecturers and students already carry. The system was built with PHP 8, MariaDB, and Bootstrap 5 as three role-based portals. A lecturer opens a session bound to a 256-bit random token rendered as a QR code; a student scans it in the phone browser, where camera frames are decoded locally, and the server applies seven ordered checks covering authentication, session state, date, time window, enrollment, and duplication before a record is written. The evaluation combined 31 black-box test cases, a decoding experiment on 1,900 synthetic camera frames, and a load test of 3,000 scan requests on a two-core testbed. An initial run exposed a legacy fallback that accepted a forged payload without a valid token; after its removal, all 31 cases behaved as specified. Symbols carrying the token were decoded without error from 2.0 pixels per module, up to a 40-degree viewing angle and 10% occlusion. Every load-test request was stored exactly once, with a 95th-percentile response time of at most 68 ms. These results indicate that a browser-only QR workflow is technically adequate for classroom attendance. Further research should add rotating tokens and location evidence to limit code relaying, and should measure usability with students and lecturers.**

***Keywords*— Attendance system; QR code; web application; server-side validation; black-box testing; higher education.**

I. INTRODUCTION

Class attendance is among the strongest known predictors of course grades in higher education. A meta-analysis covering more than eighty years of studies found that attendance relates to grades more strongly than standardized admission scores, study habits, or study skills [1]. Institutions therefore keep attendance records for every lecture and commonly make a minimum attendance rate a condition for sitting the final examination. The quality of those records depends on how they are collected. A circulated signature sheet interrupts the lecture, has to be recapitulated by hand at the end of the semester, and offers no protection against a student signing on behalf of an absent friend [2], [3].

Automated alternatives have been studied for more than a decade. Systems based on radio-frequency identification and smart cards identify a card rather than a person and require a reader in every room [4]–[6]. Fingerprint systems bind the record to the individual but need a sensor, an enrollment procedure, and physical contact with a shared surface [7]. Face recognition removes the contact and can register a whole class from one camera, and it has become the most active line of work [8]–[12], including in this journal [13]. These approaches share two costs: dedicated equipment must be bought and maintained, and biometric templates of students must be stored and protected. A systematic review reports that purchase and operating costs of biometric solutions are higher than those of the other technologies it compared [2].

The Quick Response (QR) code offers a different trade-off. The symbology is standardized in ISO/IEC 18004 [14], carries Reed–Solomon error correction [15], and can be read by the camera of any current smartphone. An attendance system can therefore follow a bring-your-own-device model in which the only equipment is the lecturer's screen and the students' phones. Early work showed that a code displayed in class and scanned by students can replace the roll call [16], and later systems added web portals, reporting, and usability evaluation for colleges, schools, and companies [17]–[20]. The known weakness of the approach is that a displayed code can be photographed and forwarded to someone outside the room. Countermeasures proposed in the literature include codes that change during the session, geolocation checks, and device binding [21], [22], and surveys of QR code security recommend that the code itself carry no trusted information and that all decisions be taken by the server [23].

Published reports of QR attendance systems concentrate on functional description and acceptance testing. Two questions that matter in a real classroom are reported less often: how small, blurred, or oblique a displayed code may be before a browser-based scanner fails, and how the recording endpoint behaves when a whole class scans within the same few seconds. The second question includes a correctness issue, because simultaneous requests must never produce two records for one student.

This paper presents AbsensiQR, a lecture attendance system developed for the Department of Information Technology, Politeknik Negeri Padang. Its contributions are as follows. First, it describes a complete browser-only design in which the QR code is generated and decoded on the client and carries nothing but an unguessable session token. Second, it specifies a layered server-side validation pipeline,

backed by a database uniqueness constraint, that decides whether a scan becomes an attendance record. Third, it reports an evaluation consisting of black-box functional tests, a controlled decoding experiment, and a load test, and it documents the defect and the limitations that the evaluation revealed.

II. Materials and Method

A. Research Stages

The work followed four stages that correspond to the generic activities of a software process [24]: requirements analysis, design, implementation, and evaluation. Requirements were expressed as use cases and the data model as an entity-relationship diagram, using the notation of the Unified Modeling Language where applicable [25]. The evaluation stage was planned so that every reported number can be reproduced from the source code and a synthetic data set, without access to personal data of students.

B. Functional Requirements

The system serves three roles, shown in Fig. 1. The administrator maintains users, courses, classes, and class enrollment, configures the system, creates database backups, and inspects the audit log. The lecturer opens an attendance session for one meeting of a class, displays its QR code, closes or reopens the session, corrects attendance manually when a student cannot scan, and exports reports. The student scans the code, sees the result immediately, and can review the attendance history and percentage for each course. Three business rules govern recording: a student may be recorded only for a class in which he or she is actively enrolled, only once per meeting, and only while the session is open.

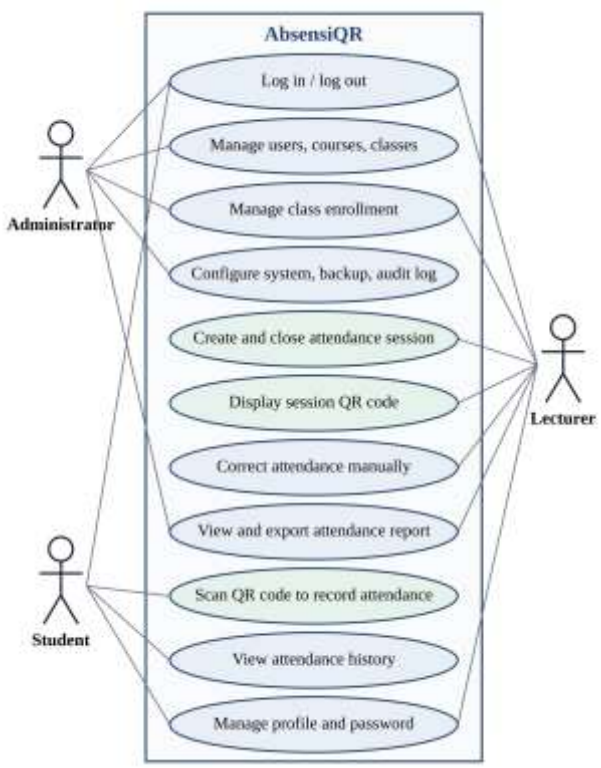


Fig. 1 Use case diagram of AbsensiQR

C. System Architecture

AbsensiQR is a three-tier web application (Fig. 2). The client tier is an ordinary web browser; no application has to be installed. The lecturer's browser draws the QR code on an HTML canvas with the qrcode-generator library [26], and the student's browser captures camera frames through the Media Capture and Streams interface [27] and decodes them with jsQR [28]. Images therefore never leave the phone; only the decoded text is sent to the server. The application tier is written in PHP 8 and organized as three portals that share an authentication module and a JSON endpoint for scan processing. The data tier is a MariaDB database accessed through PHP Data Objects with prepared statements. Table I lists the components.

TABLE I
Technology Stack of AbsensiQR

Component	Technology
Server language	PHP 8 with PDO (evaluated on 8.3.6)
Database	MariaDB 10.11, InnoDB, utf8mb4
User interface	Bootstrap 5.3.2, Bootstrap Icons 1.11.3
QR generation	qrcode-generator 1.4.4, client side
QR decoding	jsQR 1.4.0, client side
Charts, calendar	Chart.js 4.4.0, FullCalendar 5.11
Report export	CSV, spreadsheet, PDF (Dompdf 3.1)
Web server	Apache 2.4 with access rules

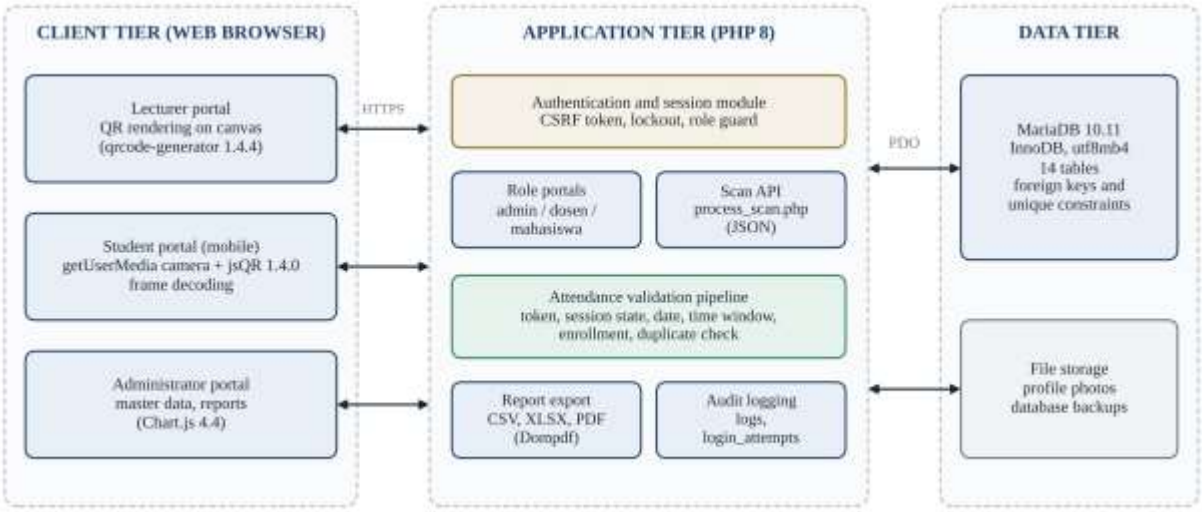


Fig. 2 Three-tier architecture of AbsensiQR

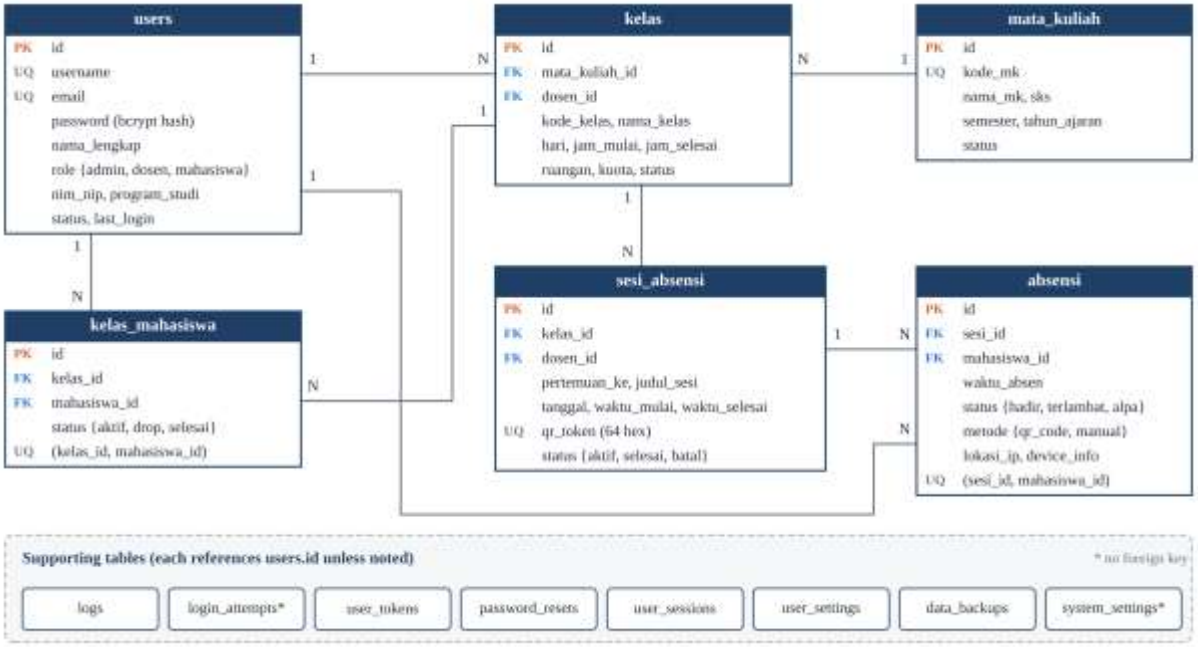


Fig. 3 Entity-relationship diagram of the AbsensiQR database (PK: primary key, FK: foreign key, UQ: unique constraint)

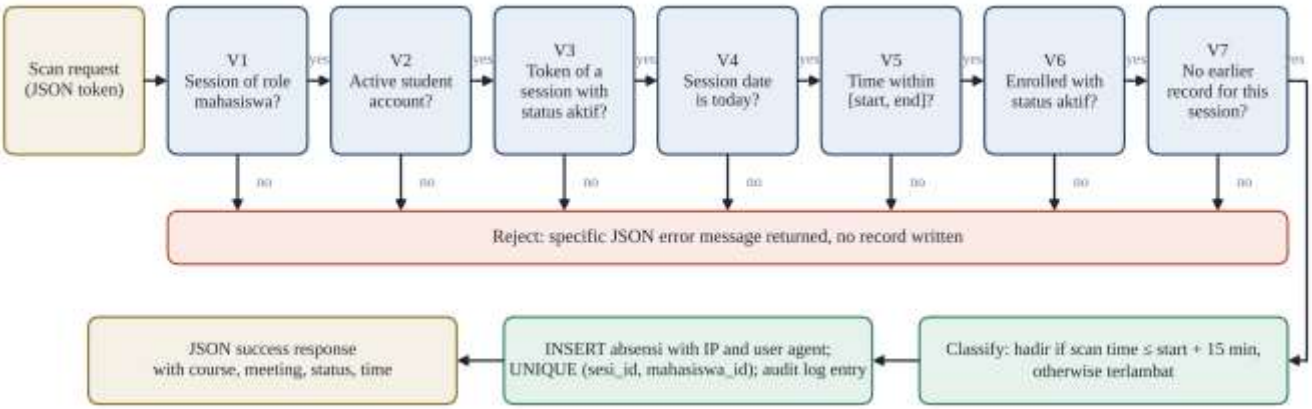


Fig. 4 Server-side validation of a scan request: checks V1 to V7 are applied in order, and only a request that passes all of them is classified and stored

D. Data Model

The database contains 14 tables (Fig. 3). Six of them carry the attendance logic. Table `users` stores all three roles and is distinguished by the `role` column. A course in `mata_kuliah` is offered as one or more classes in `kelas`, each taught by one lecturer, and `kelas_mahasiswa` records enrollment with a status of active, dropped, or completed. Each meeting is a row of `sesi_absensi` that holds the date, the time window, the session state, and the QR token; each scan becomes a row of `absensi` with the scan time, the status, the method, the client address, and the browser identification string. Three uniqueness constraints protect integrity at the database level: one enrollment per student and class, one token per session, and one attendance record per student and session. Foreign keys cascade on deletion so that removing a class removes its sessions and records. The remaining eight tables support login throttling, persistent login, password reset, user preferences, system settings, backups, and the audit log.

E. Session Token and QR Code

When a lecturer creates a session, the server verifies that the class belongs to that lecturer and that the meeting number is not yet used, then draws 32 bytes from the operating system's cryptographically secure generator and stores them as 64 hexadecimal characters. Such a source is required for values that an adversary must not predict [29]. The token is the only content of the QR code. For a token space of 256 bits, the probability that an attacker finds a valid token in n online attempts is

$$P_{\text{guess}} = n / 2^{256} \tag{1}$$

which is negligible for any feasible n. Encoded in byte mode at error correction level M, the token yields a version 5 symbol of 37 × 37 modules. Level M allows recovery from damage to about 15% of the codewords [14]. The symbol is drawn at 300 pixels on the lecturer's screen together with a countdown to the end of the session.

On the student side, the scanner requests the rear camera, scales each video frame so that its longer side does not exceed 640 pixels, and passes the frame to the decoder, which also tries the inverted image. When the device supports it, a control switches the camera light on. Two alternatives are offered when the camera cannot be used: the student may upload a photograph of the code or type the token.

F. Validation Pipeline

The decoded text is posted to the scan endpoint, which applies the ordered checks V1 to V7 of Fig. 4. A request that fails any check receives a specific error message and leaves the database unchanged. A request that passes all checks is classified by its arrival time: it is recorded as present (*hadir*) when it arrives within 15 minutes of the scheduled start and as late (*terlambat*) otherwise. The record is then inserted together with the client address and browser identification, and the event is written to the audit log. Check V7 is an application-level test; the unique constraint on the pair of session and student enforces the same rule inside the database and therefore also covers requests that arrive at the same instant.

A lecturer may add or change a record by hand, for example for a student whose phone is unavailable; such records are marked with the method *manual* so that they remain distinguishable in reports. For a student and a course, the attendance percentage shown on the dashboard is

$$A = (H + T) / S \times 100\% \tag{2}$$

where H and T are the numbers of meetings recorded as present and late, and S is the number of completed sessions of the class. The dashboard marks a course as safe when A is at least 75%.

G. Security Controls

Security requirements were derived from the OWASP Top 10 and the Application Security Verification Standard [30], [31]. Table II maps the threats considered to the controls implemented. Passwords are stored as bcrypt hashes [32] and are rehashed transparently when the default cost changes. Failed logins are counted per account and per client address, and further attempts are refused for the rest of a 15-minute window once a limit is reached, in line with the throttling recommended for online guessing attacks [33]. The response to a failed login is the same whether or not the account exists, and the password verification routine runs in both cases so that response time does not reveal which is true. The login page is served with a Content Security Policy [34] that limits scripts to the application's own origin, one content delivery network, and inline code carrying a per-request nonce.

TABLE II
THREATS CONSIDERED AND CONTROLS IMPLEMENTED

Threat	Control
Credential theft	bcrypt password hashes with automatic rehash
Online guessing	Lockout after 5 failures per account or 20 per address in 15 min; uniform error message
Request forgery	Per-session 256-bit token on the login form, compared in constant time
Session hijacking	Identifier regenerated at login; HttpOnly and SameSite cookies, Secure under HTTPS
Script injection	Content Security Policy with nonce; integrity hashes on third-party scripts; output escaping
SQL injection	Prepared statements for every query with user input
Stolen persistent login	Random token stored only as SHA-256 digest, rotated on use, 30 days, five devices
Privilege escalation	Role guard on portal pages; ownership check on classes and sessions
Repudiation	Audit log and attendance rows keep address and browser identification
Information exposure	Directory listing disabled; configuration, library, and backup paths denied

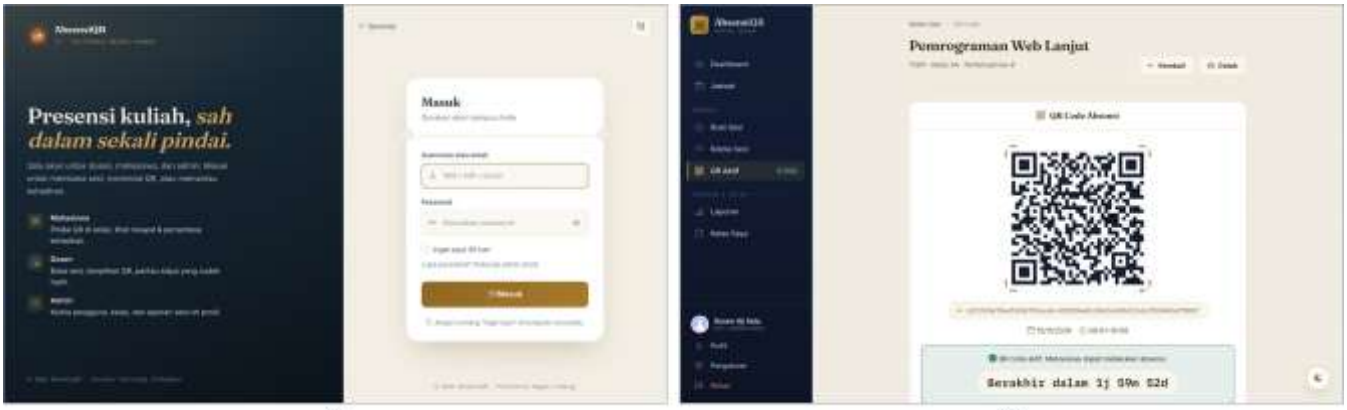
H. Evaluation Method

The evaluation addressed functional suitability and performance efficiency, two characteristics of the ISO/IEC 25010 product quality model [35]. All experiments ran on one testbed: a virtual machine with two cores of an Intel Xeon processor at 2.30 GHz and 7 GB of memory, running PHP 8.3.6 with four worker processes and MariaDB 10.11.14. The database was filled with synthetic data only: one administrator, three lecturers, 200 students, four classes, and 24 past sessions.

1) Functional testing: Thirty-one black-box test cases [36] were derived from the requirements of Section II-B, the checks of Fig. 4, and the controls of Table II. Each case was executed as real HTTP requests against the running application, including login with the anti-forgery token, and the verdict was taken from the response and from the resulting database state.

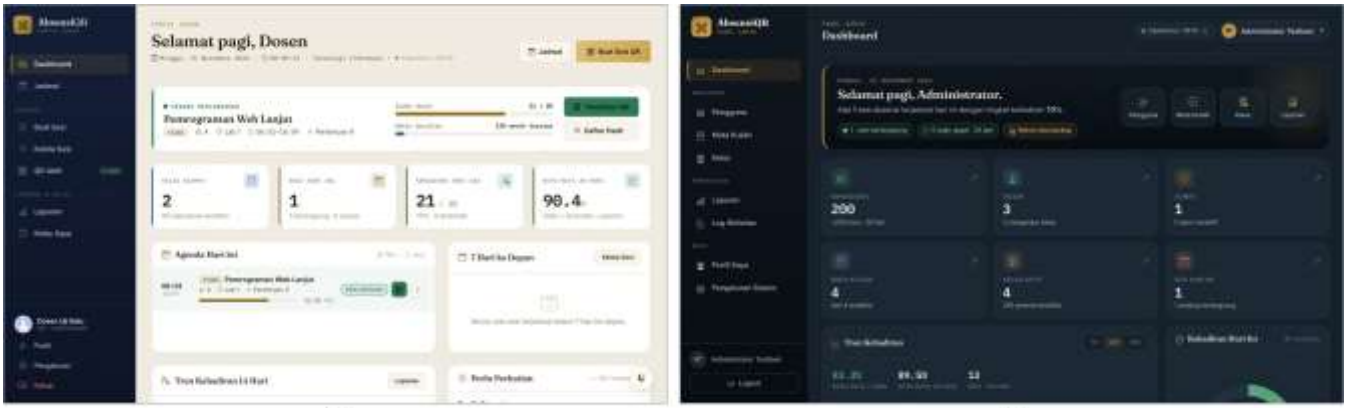
2) QR decoding experiment: Fifty random tokens were encoded with the library and settings used by the application. Each symbol was composited into a 640 × 480 frame, the size delivered to the decoder, under 38 conditions that varied one factor at a time: apparent size, Gaussian blur, in-plane rotation, viewing angle, and occlusion of a central square. The resulting 1,900 frames were decoded with the same jsQR version, and a frame counted as a success only when the decoded text equalled the token.

3) Load testing: Two hundred authenticated student sessions were prepared, and in each run every student submitted one valid scan for a newly created session. Runs were made at 1, 10, 25, 50, and 100 concurrent requests and repeated three times, giving 3,000 requests. Response time was measured at the client, and after each run the number of stored records was compared with the number of requests. Client and server shared one host, so the figures exclude network delay. They are interpreted against the 0.1 s limit below which a response is perceived as instantaneous [37].



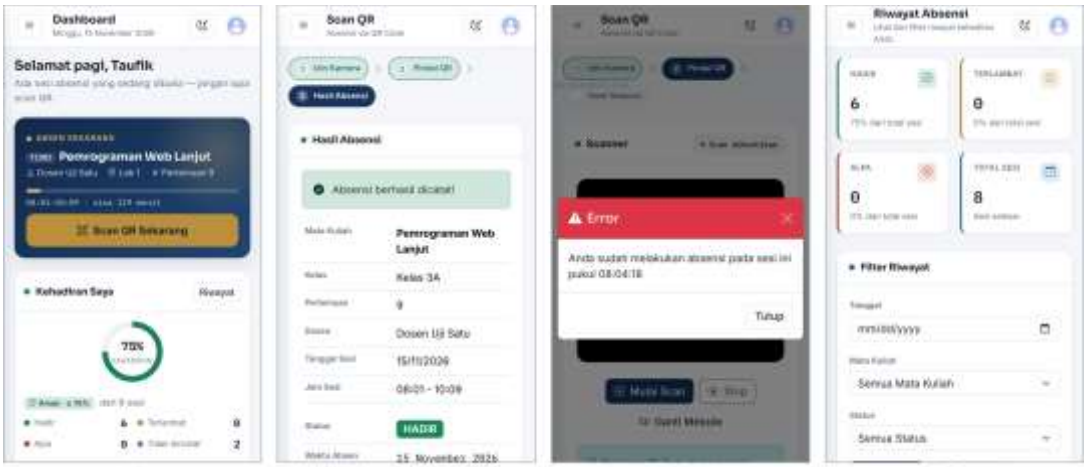
(a) (b)

Fig. 5 Desktop interface: (a) login page; (b) QR code display of an open session, shown with synthetic test data



(a) (b)

Fig. 6 Dashboards: (a) lecturer; (b) administrator, shown with synthetic test data



(a) (b) (c) (d)

Fig. 7 Student interface on a phone viewport: (a) dashboard with an open session; (b) result of a successful scan; (c) rejection of a repeated scan; (d) attendance history

The interface was additionally exercised in the Chrome browser with a desktop viewport for lecturers and an emulated phone viewport for students. For the student scan, the browser camera was replaced by a synthetic video stream showing the code of an open session, so that the complete path from camera frame to stored record was executed.

III. Results and Discussion

A. Implemented System

The implementation comprises about 22,000 lines of PHP in three portals. Fig. 5 shows the login page and the lecturer's QR display, which presents the code, the token, the session window, and a countdown. The lecturer and administrator dashboards (Fig. 6) summarize the running session, the attendance rate of the day, and trends over recent weeks.

The student portal is laid out for a phone screen (Fig. 7). The dashboard announces an open session of an enrolled class and leads directly to the scanner. In the end-to-end run with the synthetic camera stream, the code was decoded in the browser, the server accepted the scan, and the result page reported the course, meeting, status, and time; the stored row carried the method *qr_code* and the browser identification of the emulated phone. A second scan of the same session was refused with the time of the first record.

B. Functional Testing

Table III summarizes the functional test. In the first run, 30 of the 31 cases produced the specified behavior; after the correction described below, the complete suite was repeated and all 31 cases conformed. All rejection paths of the pipeline returned their own message and wrote nothing, an injection string submitted as a token was handled as an unknown token, and both classification outcomes were recorded correctly. In the duplication group, twenty identical requests sent simultaneously by one student produced one success response, nineteen rejections, and exactly one row, which confirms that the uniqueness constraint closes the gap between check and insertion.

TABLE III
Summary of Black-Box Functional Testing

Test group	Cases	Conforming
Authentication and role-based access	5	5
Session management by the lecturer	5	5
Token validity (empty, unknown, injection, closed)	4	4
Date and time window	3	3
Enrollment and enrollment status	2	2
Recording and present/late classification	4	4
Duplicate prevention, including concurrent requests	2	2
Login hardening and response headers	5	5
Forged payload without a valid token	1	1*
Total	**31**	**31**

Non-conforming in the first run; conforming after the fallback lookup was removed

The case that failed in the first run was a security defect. The scan endpoint contained a fallback from an earlier payload format: when the submitted text was a JSON object and its token matched no session, the endpoint looked up the most recent open session by the class and lecturer identifiers contained in the same object. A request that supplied these two small integers and an arbitrary token was accepted. The other checks still applied, so the request had to come from a logged-in student who is enrolled in the class, during the session window, and it could be made only once. Within those bounds, however, the fallback removed the need to see the code at all. The fallback was deleted, so that the token is now the only key to a session and any other field of the payload is ignored, and the case was kept as a regression test. The finding illustrates the value of deriving negative tests from the trust boundary rather than from the user interface, which never produces such a payload.

C. QR Decoding Robustness

Fig. 8 shows the decoding success rate under four of the five factors. No frame was decoded to a wrong text in any condition, so every failure was a refusal. Decoding was reliable from 2.0 pixels per module, which corresponds to a symbol, including its quiet zone, that spans 90 pixels or about 14% of the frame width. Between 1.7 and 1.9 pixels per module the outcome was unstable (8%, 100%, and 12%

at three adjacent sizes), an effect of resampling at non-integer module sizes, and below 1.5 pixels per module nothing was decoded. Blur was tolerated up to a standard deviation of 0.34 modules; the rate fell to 38% at 0.45 modules and to zero beyond. In-plane rotation had no effect at any tested angle from 0 to 45 degrees. The viewing angle was tolerated fully up to 40 degrees, with 96% at 50 degrees, 76% at 60 degrees, and failure at 70 degrees. A grey square covering up to 10% of the symbol area was corrected in all frames, whereas 12% or more always failed. This is below the nominal 15% of level M, which refers to codewords rather than area and assumes that function patterns remain intact.

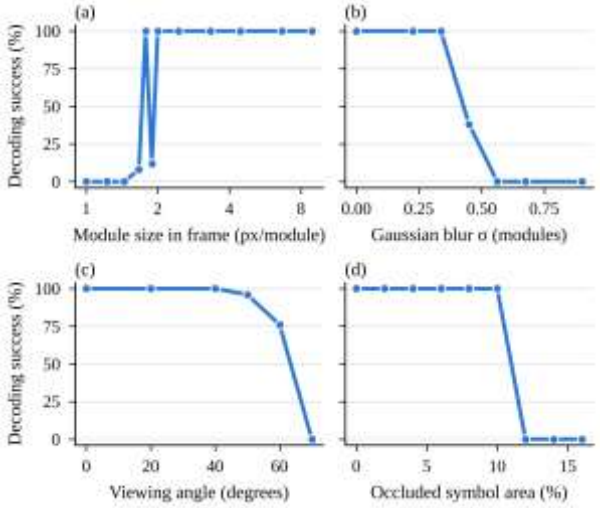


Fig. 8 Decoding success of the browser decoder for 50 tokens per condition: (a) apparent module size; (b) blur; (c) viewing angle; (d) occluded area

For classroom use these limits translate into simple guidance. The code should fill at least about one seventh of the phone's frame, which is easily met when it is projected, but may require students at the back to zoom or move closer when it is shown on a laptop screen. Seats at steep angles to the screen remain usable up to roughly 40 degrees. The decoder needed between 10 and 18 ms per frame on the testbed, so decoding speed is not a constraint.

D. Performance under Concurrent Scans

Table IV and Fig. 9 report the load test, which was run on the corrected endpoint. All 3,000 requests succeeded, and after every run the number of stored records equalled the number of requests. A single scan was processed in a median of 2.2 ms. With 100 requests in flight, the median rose to 47.4 ms and the 95th percentile to 67.8 ms; the largest single response time observed was 102 ms. Throughput settled between 835 and 889 requests per second from ten concurrent requests upward, which indicates that the two cores were saturated and that additional concurrency only lengthened the queue. Login is deliberately slower because of password hashing and took a median of 54 ms.

TABLE IV
RESPONSE TIME AND THROUGHPUT OF THE SCAN ENDPOINT

Concurrent requests	Median (ms)	95th pct. (ms)	Max. (ms)	Throughput (req/s)
1	2.2	2.7	5.7	430
10	10.9	20.9	33.5	835
25	21.7	39.0	71.9	854
50	27.5	50.2	77.6	844
100	47.4	67.8	101.5	889

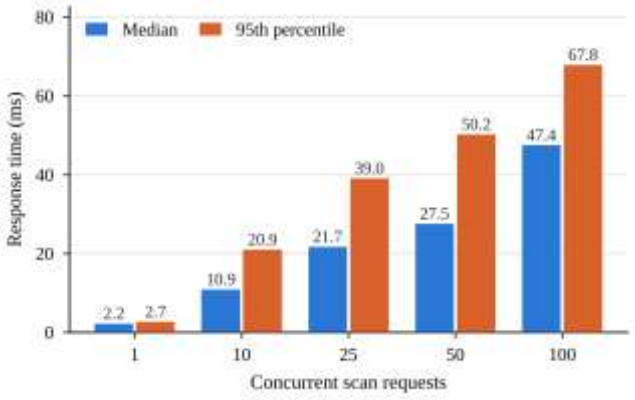


Fig. 9 Response time of the scan endpoint (mean of three runs of 200 requests)

At every load level the 95th percentile is below the 0.1 s limit for an instantaneous reaction [37], and only the single slowest of the 3,000 requests reached it. The figures describe server processing only. In a lecture hall the perceived time will be dominated by the wireless network and by the handshake of the secure connection, and a production web server will schedule requests differently from the testbed. The result nevertheless shows that the validation pipeline, with about six indexed queries per scan, is not a bottleneck for a class or even for several classes scanning at once.

E. Operational Footprint

The schema snapshot of the deployed database, taken on 2 October 2026, gives an indication of the scale of use. Table V lists, for the main tables, the number of identifiers issued up to that date as read from the auto-increment counters. These are upper bounds on the number of rows, since deleted rows are not subtracted, and they were not used in any of the experiments above.

TABLE V
IDENTIFIERS ISSUED IN THE DEPLOYED DATABASE (2 OCTOBER 2026)

Entity (table)	Identifiers issued
User accounts (users)	195
Courses (mata_kuliah)	16
Classes (kelas)	30
Enrollments (kelas_mahasiswa)	480
Attendance sessions (sesi_absensi)	71
Attendance records (absensi)	832
Audit log entries (logs)	1,099

F. Discussion and Limitations

The results support the premise that attendance can be recorded reliably with no equipment other than a screen and the students' own phones, which is the main practical

advantage of QR-based systems over card and biometric systems [2], [4]. Compared with earlier QR attendance reports [16]–[19], the present work makes the decision logic explicit, places it entirely on the server as recommended for QR applications [23], and quantifies both the reading limits of the browser decoder and the behavior of the endpoint under simultaneous scans.

Several limitations must be stated. First, the token is fixed for the duration of a session. A student in the room can photograph the code and send it to an enrolled classmate elsewhere, who can then scan it within the time window. The system limits the damage, since each account can be recorded only once and the address and browser identification of each scan are kept for inspection, but it does not prevent the relay. Tokens that rotate every few seconds and checks of location or network, as proposed in [21], [22], are the appropriate next step. Second, the lateness threshold of 15 minutes is fixed in the scan endpoint although the administrator's settings page already stores a configurable value; the two should be connected. Third, camera access in browsers is available only in a secure context [27], so the system must be served over HTTPS. Fourth, the performance figures come from a small testbed with synthetic data and without network effects. Finally, no usability study has been carried out; perceived ease of use by students and lecturers remains to be measured, for instance with the System Usability Scale [38].

IV. Conclusion

AbsensiQR records lecture attendance through a QR code that is generated and decoded in the browser and that carries only a 256-bit session token. All decisions are taken by a server-side pipeline of seven checks, and a database constraint guarantees one record per student and session. In testing, a negative test exposed a legacy fallback that weakened the token requirement; after it was removed, all 31 functional cases conformed to the specification. The browser decoder read the session code without error from 2.0 pixels per module, up to a viewing angle of 40 degrees and with up to 10% of the symbol covered. The scan endpoint stored each of 3,000 requests exactly once and answered within 68 ms at the 95th percentile on a two-core server.

For the department, the system replaces signature sheets with records that are available to lecturers and students immediately and that carry evidence of when and from which device they were made. Further research will introduce rotating tokens and location or network evidence to counter code relaying, connect the lateness threshold to the system settings, repeat the load test on the production infrastructure, and evaluate usability and acceptance with students and lecturers over a full semester.

Acknowledgment

The authors thank the Department of Information Technology, Politeknik Negeri Padang, for supporting the deployment of the system.

References

[1] M. Credé, S. G. Roch, and U. M. Kieszczynka, "Class attendance in college: A meta-analytic review of the relationship of class attendance with grades and student characteristics," *Rev. Educ. Res.*, vol. 80, no. 2, pp. 272–295, Jun. 2010, doi: 10.3102/0034654310362998.

[2] N. S. Ali, A. H. Alhilali, H. D. Rjeib, H. Alsharqi, and B. Al-Sadawi, "Automated attendance management systems: Systematic literature review," *Int. J. Technol. Enhanc. Learn.*, vol. 14, no. 1, pp. 37–65, 2022, doi: 10.1504/IJTEL.2022.120559.

[3] S. H. Falih and Y. H. Ismael, "Student attendance and evaluation system: A review," *Al-Rafidain J. Comput. Sci. Math.*, vol. 19, no. 1, pp. 48–52, 2025, doi: 10.33899/csmj.2025.155461.1158.

[4] A. K. Muslim, N. Ahmad, D. Z. M. Dzulkifli, V. G. Raj, and M. I. Ishak, "Smart card attendance systems in educational institutes: A systematic review," *J. Technol. Manag. Technopreneurship*, vol. 12, no. 2, 2024.

[5] H. A. Abdulghani, N. A. Nijdam, and D. Konstantas, "Analysis on security and privacy guidelines: RFID-based IoT applications," *IEEE Access*, vol. 10, pp. 131528–131554, 2022, doi: 10.1109/ACCESS.2022.3227449.

[6] S. Meivel, C. Praghadeesh, A. Ravinder, and D. Sibisaran, "Hybrid student authentication system using RFID reader and face biometrics using deep learning techniques," in *Proc. Int. Conf. Appl. Artif. Intell. Comput. (ICAAIC)*, 2022, pp. 550–556.

[7] A. Olasupo *et al.*, "A fingerprint based students attendance management system for Olabisi Onabanjo University," *FUDMA J. Sci.*, vol. 6, no. 1, pp. 253–265, Mar. 2022, doi: 10.33003/fjs-2022-0601-896.

[8] O. Alniemi and H. F. Mahmood, "Class attendance system based on face recognition," *Rev. Intell. Artif.*, vol. 37, no. 5, pp. 1245–1253, Oct. 2023, doi: 10.18280/ria.370517.

[9] A. S. Lateef and M. Y. Kamil, "Face recognition-based automatic attendance system in a smart classroom," *Iraqi J. Electr. Electron. Eng.*, vol. 20, no. 1, Jun. 2024, doi: 10.37917/ijeee.20.1.4.

[10] A. S. Lateef and M. Y. Kamil, "Facial recognition technology-based attendance management system application in smart classroom," *Iraqi J. Comput. Sci. Math.*, vol. 4, no. 3, Aug. 2023, doi: 10.52866/ijcsm.2023.02.03.012.

[11] N. A. Ismail *et al.*, "Web-based university classroom attendance system based on deep learning face recognition," *KSII Trans. Internet Inf. Syst.*, vol. 16, no. 2, 2022.

[12] V. Seelam *et al.*, "Smart attendance using deep learning and computer vision," *Mater. Today Proc.*, vol. 46, pp. 4091–4094, 2021, doi: 10.1016/j.matpr.2021.02.625.

[13] H. Pranoto and O. Kusumawardani, "Real-time triplet loss embedding face recognition for authentication student attendance records system framework," *JOIV : Int. J. Inform. Visualization*, vol. 5, no. 2, pp. 150–155, Jun. 2021, doi: 10.30630/joiv.5.2.480.

[14] *Information Technology—Automatic Identification and Data Capture Techniques—QR Code Bar Code Symbology Specification*, ISO/IEC 18004:2015, 2015.

[15] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, Jun. 1960, doi: 10.1137/0108018.

[16] F. Masalha and N. Hirzallah, "A students attendance system using QR code," *Int. J. Adv. Comput. Sci. Appl.*, vol. 5, no. 3, pp. 75–79, 2014, doi: 10.14569/IJACSA.2014.050310.

[17] M. N. Saad, T. Z. T. Muda, N. Oulaganathan, and R. Kaliappan, "Online student attendance system using QR-code for Kolej Komuniti Segamat," in *Proc. Int. Conf. Commun. Media (i-COME 2022)*, 2023, pp. 364–375, doi: 10.2991/978-2-38476-098-5_32.

[18] Vitriani, G. Ali, W. N. Rohman, and M. Novalia, "Design of a web-based student attendance information system using QR code" (in Indonesian), *KLIK Kaji. Ilm. Inform. Komput.*, vol. 3, no. 5, 2023, doi: 10.30865/klik.v3i5.752.

[19] A. R. Djamarullah, I. Nuryasin, and H. Wibowo, "Designing a QR code attendance system using BYOD (bring your own device)," *Ultimatics J. Tek. Inform.*, vol. 16, no. 1, Jun. 2024.

[20] A. Ayuliana, F. V. Nurfitriana, and F. Dirayati, "Analysis and design of employee attendance system using QR code with webcam (case study: PT. Adhicon Perkasa)," *Eng. Math. Comput. Sci. J.*, vol. 7, no. 1, pp. 127–133, 2025.

[21] S. W. Taju and Y. P. Mamahit, "Implementing QR code and geolocation technologies for the student attendance system," *CogITo Smart J.*, vol. 10, no. 1, 2024.

[22] A. Nwabuwe, B. Sanghera, T. Alade, and F. Olajide, "Fraud mitigation in attendance monitoring systems using dynamic QR code, geofencing and IMEI technologies," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 4, pp. 938–945, 2023, doi: 10.14569/IJACSA.2023.01404104.
[23] H. A. M. Wahsheh and F. L. Luccio, "Security and privacy of QR code applications: A comprehensive study, general guidelines and solutions," *Information*, vol. 11, no. 4, Apr. 2020, Art. no. 217, doi: 10.3390/info11040217.
[24] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2020.
[25] *OMG Unified Modeling Language (OMG UML)*, version 2.5.1, Object Management Group, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1
[26] K. Arase, *QR Code Generator (qrcode-generator)*, version 1.4.4, 2019. [Online]. Available: https://github.com/kazuhikoarase/qrcode-generator
[27] C. Jennings *et al.*, "Media capture and streams," World Wide Web Consortium, W3C Candidate Recommendation Draft, 2025. [Online]. Available: https://www.w3.org/TR/mediacapture-streams/
[28] *jsQR: A Pure JavaScript QR Code Reading Library*, version 1.4.0, Cozmo, 2021. [Online]. Available: https://github.com/cozmo/jsQR
[29] D. Eastlake 3rd, J. Schiller, and S. Crocker, "Randomness requirements for security," Internet Eng. Task Force, RFC 4086, Jun. 2005, doi: 10.17487/RFC4086.
[30] *OWASP Top 10:2021*, OWASP Foundation, 2021. [Online]. Available: https://owasp.org/Top10/
[31] *Application Security Verification Standard*, version 4.0.3, OWASP Foundation, Oct. 2021. [Online]. Available: https://owasp.org/www-project-application-security-verification-standard/
[32] 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.
[33] P. A. Grassi *et al.*, "Digital identity guidelines: Authentication and lifecycle management," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST Spec. Publ. 800-63B, Jun. 2017, doi: 10.6028/NIST.SP.800-63b.
[34] M. West and A. Sartori, Eds., "Content security policy level 3," World Wide Web Consortium, W3C Working Draft, 2025. [Online]. Available: https://www.w3.org/TR/CSP3/
[35] *Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—Product Quality Model*, ISO/IEC 25010:2023, 2023.
[36] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, 3rd ed. Hoboken, NJ, USA: Wiley, 2011.
[37] J. Nielsen, *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann, 1993.
[38] 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.