Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

MarkSure: A Configurable University Result-Lifecycle System and a Measurement Study of What It Guarantees

Rohit Girase, Lokesh Patil, Aayush Gawali, Vaishnavi Sawale, and Mayur Mahajan
Department of Computer Science and Engineering, Sandip University, Nashik, India

***Abstract*—University results pass through many hands before a student sees them: internal assessment, script evaluation, computation under institution-specific ordinances, approval, publication, revaluation and third-party verification. We present MarkSure, a web platform that covers this lifecycle with an ordinance engine driven by policy data, a revaluation workflow with question-level requests, pre-publication checks, a hash-chained audit ledger, versioned Ed25519-signed grade cards and an early-warning model. We then test four guarantees against the implementation. Differential testing against an exact-arithmetic reference written from the specification on 120,000 randomized student-semesters found no disagreement beyond three unwritten conventions and six conventions fixed to match the code, and a credit-blind grace rule under which half a mark more can lower the SGPA. Exact pre-publication checks caught every injected totaling and absent-with-marks error, but the examiner-leniency test flagged a 10-point bias in only 22.7 % of sessions and the pass-rate rule raised 1.75 false alarms per clean session. Against an insider with database access, every direct edit of a current grade card was detected, but cards verified as valid after the attacker inserted a signing key, rolled back a revaluation, edited results between approval and publication or merely re-validated certain superseded cards; deleting the newest ledger entries, rewriting the chain and appending entries went unnoticed. At scale, two script-coding defects and a per-request aggregation limited set-up and result-day throughput. A three-feature early-warning model reached ROC-AUC 0.860–0.974 on public data, but 0.795 once already-disengaged students were excluded. We report a remedy for each failure.**

***Index Terms*—audit logs, differential testing, digital signatures, early warning, examination management, rules as code**

I. Introduction

At an Indian university a semester result combines internal marks entered by faculty, external answer scripts evaluated by examiners under dummy numbers, and a computation that applies the university's *ordinances* (its legal rules on passing minima, grace marks, grading and credit aggregation under the UGC credit framework [1]). After approval and publication, students may obtain scan copies of their scripts and ask for re-totaling or revaluation, and employers verify the resulting grade cards. At our own institution these steps are handled by separate tools and offices, so errors surface after publication and manual corrections leave little trace.

Published result-management systems automate result computation [2], examination administration [3] or post-release analysis [4]. On-screen moderation and question-wise marking [5], [6], blockchain credentials [7], [8] and early warning [9], [10] have been studied on their own. What is seldom reported is a test of whether such a system computes what its rules say, catches errors before release, and keeps its records tamper-evident.

We present MarkSure, a platform that covers the whole lifecycle, and measure four guarantees against its implementation (Table I): **G1** results are computed exactly as the configured policy specifies; **G2** entry and evaluation errors are caught before publication; **G3** changes to results, grade cards and their history are detectable; **G4** the system enforces its workflow and access rules and operates at university scale. Our contributions are (i) the system design (Section III); (ii) a reusable evaluation protocol: differential, metamorphic and brute-force testing of the rule engine, injected-error and power analysis of scrutiny rules, an insider attack catalog with capability levels, and scale and load tests (Section IV); and (iii) the measured strengths and failures of the implementation, each with a remedy (Sections V and VI). Several failures confirm known principles, for instance that an unanchored hash chain cannot reveal truncation [11], [12]; we state where this is the case and report which concrete defects of the implementation make them exploitable. Code, generators and raw outputs are available from the authors on request.

II. Related Work

Result and examination systems. A departmental system at the University of Benin computes scores and prints result sheets [2]; an examination-affairs system manages administrative workflow [3]; a dashboard analyzes results after release [4]; on-screen evaluation work moderates examiner variation [5] and shows students question-wise marked scripts [6]. Commercial systems make grading rules configurable, but we know of no published test of whether such configurations are computed correctly or of how revaluation supersedes issued documents.

Rules as code and their testing. Decision tables with a formal semantics allow gaps and overlaps in business rules to be found [13]. Catala formalizes legislation and reports that formalization exposes ambiguities in the legal text [14]; a re-implementation of the French income-tax engine was validated

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

TABLE I
GUARANTEES, MECHANISMS, EXPERIMENTS AND VERDICTS.

Guarantee	Mechanism	Exp.	Verdict
G1 exact computation	policy-driven engine	E1	holds up to three unwritten and six fixed conventions; grace not monotone in marks
G2 errors caught before release	exact and statistical checks	E2	exact checks hold; statistical checks weak or noisy
G3 tamper evidence	signed versioned cards; hash chain	E3	direct edits of current cards detected; fails for key insertion, revaluation rollback, post-approval edits, re-validated superseded cards, tail deletion and appends
G4 workflow, access, scale	state machines, RBAC; set-based processing	E4, E6	access rules hold; script coding and results page do not scale
(feature)	early warning	E5	competitive on school data; weaker without disengaged students; fairness caveats

against the official one [15]. Our E1 applies this idea to an ordinance engine, combining differential testing [16], random property-based testing [17] and metamorphic relations [18]; the control flow follows standard workflow patterns [19].

Tamper evidence. Hash-linking records dates back to digital time-stamping [20]. Secure audit logs add forward security [11], DBMS audit logs were made tamper-evident by hashing transactions and notarizing the result externally [12], and forensic analysis locates when and where data were altered [21]. Ledger databases export digests so that administrators cannot rewrite history unnoticed [22], [23], and Certificate Transparency publishes signed tree heads for the same purpose [24]. These works assume an external, trusted anchor; our results confirm what happens without one and identify further gaps specific to an application-level design.

Academic credentials. Blockchain-based academic records have produced many prototypes [7], [8]; W3C Verifiable Credentials standardize holder-presentable signed credentials [25], and India's National Academic Depository holds institutions' awards [26]. MarkSure's cards are verified online against the issuer.

Scrutiny and early warning. Rater severity is classically analyzed with many-facet Rasch models [27], and marking reliability studies show substantial variation between markers [28]. Early warning has run inside operational platforms [9], prediction of failure and dropout is well studied [10], attendance is a robust correlate of grades [29], and subgroup audits of such models are recommended [30], [31].

III. SYSTEM DESIGN

Terms. A course has *internal* components marked by faculty and *external* components (the end-semester examination) marked by examiners on scripts identified only by *dummy numbers*. *Grace marks* are a bounded number of marks an ordinance allows to be added so that a narrowly failed course is passed. *SGPA* and *CGPA* are the credit-weighted semester and cumulative grade-point averages on a 10-point scale; *ATKT* ("allowed to keep terms") is promotion with a limited number of failed courses; a *reserved* result is withheld because a mark is missing. *Re-totaling* re-adds the marks of a script, *revaluation* re-marks it.

MarkSure is a React client, an Express/Node.js API and a PostgreSQL database whose schema is migrated automatically. The rule engine, the scrutiny checks and the early-warning model are pure functions without database access, which lets us test them in isolation. We describe the evaluated code (commit `ca0a847`).

A. Lifecycle and workflow

Nine phases, from set-up to verification, are linked by two loops: scrutiny can return a session to evaluation, and an accepted revaluation re-enters processing for one student. Three finite-state machines govern hand-overs. An internal-marks sheet moves from draft to submitted to locked, and only the Controller of Examinations (CoE) may reopen it, with a reason. A session moves from marks entry to processed, approved and published; approval is refused while a blocking or unreviewed warning flag is open, and publication is refused if a SHA-256 snapshot of the results differs from the one stored at approval. A post-result request moves through payment, blind assignment to an examiner other than the original one, an optional third or senior examiner, decision, result update, re-issue and refund. Each accepted transition appends a ledger entry in the same transaction as the change.

B. Ordinance as policy data

Each scheme points to a versioned JSON policy: grade bands, absolute or relative grading (bands at $\mu + z\sigma$ of passed students' percentages, with a minimum class size), component and aggregate pass percentages, grace limits, SGPA/CGPA options, ATKT limit, class awards, service fees and deadlines, the revaluation rule and the scrutiny thresholds. A draft policy can be simulated on a session before activation. A course is evaluated in four steps. (1) A missing mark reserves the course; absence in an external component makes it absent; otherwise it fails if a component or the total is below its minimum. If every failing component is grace-eligible, the grace needed is $g = \max(\sum_c d_c,\, d_{\mathrm{tot}})$, the smallest amount that lifts each failing component and the total to their minima. (2) Needs within the per-course cap are sorted in ascending order and granted while the semester budget and the maximum number of graced courses allow; for the ordinance objective "clear as many courses as possible, then use the least grace" this greedy rule is optimal, since the k smallest needs form the cheapest feasible set of size k. (3) The final percentage selects the grade; a course passed only through grace gets the lowest pass grade. (4) $\mathrm{SGPA} = \sum_i C_i G_i / \sum_i C_i$, CGPA is the credit-weighted mean of SGPAs, and status and class award follow from thresholds.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Revaluation. With original marks O, revaluers' marks R_1, R_2, maximum M and $\pi(x) = 100|x|/M$: in single mode an increase with $\pi(R_1 - O) > T_3$ goes to a senior examiner, whose marks become the candidate, and otherwise R_1 is the candidate; in double mode $\pi(R_1 - R_2) > T_2$ calls a third examiner and the candidate is the mean of the two closest marks, otherwise the mean of R_1 and R_2. The candidate, rounded to half marks, replaces O only if $\pi(\cdot - O) > T_1$ and, for a decrease, only if the policy allows decreases. The fee is refunded above a change threshold or when the pass/fail status changes.

C. Pre-publication scrutiny

Blocking checks are exact: the question-wise total of an on-screen-marked script differs from the recorded external marks; a student recorded absent has marks; an SGPA differs from an independent SQL recomputation. *Warning* checks are statistical: an examiner whose mean deviates from the course mean, $z_e = (\bar{x}_e - \bar{x}_c)/(s_c/\sqrt{n_e})$ with $|z_e| > 3$ and $n_e \geq 5$ (the course mean includes the examiner's own scripts); an examiner whose spread is below 20 % of the course spread; a course pass rate outside $\pm 2\sigma$ of its earlier sessions; and internal marks $\geq 85\,\%$ with external marks $\leq 25\,\%$. The CoE must resolve every warning before approval.

D. Integrity

Ledger. Entry i stores $h_i = \mathrm{SHA256}(h_{i-1}|t|u|e|k|a|b|f|r)$, the "|"-joined previous hash, time, actor, entity, entity key, action, sorted-key JSON before and after images, and reason; the stored actor role and client address are not hashed, and no key is involved. A row-level trigger rejects `UPDATE` and `DELETE`, and a verifier recomputes the chain. The design specification calls for an external anchor of the chain head; the code does not implement one.

Grade cards. At publication each semester result becomes a JSON body (identity, courses, grades, SGPA/CGPA, versions, document number, issue time) serialized with recursively sorted keys (in the spirit of, not identical to, RFC 8785 [32]), hashed with SHA-256 and signed with Ed25519 [33], [34]. The private key stays on the application server; public keys are stored in a database table by key identifier, and the verifier trusts any key found there. A revaluation that changes a result issues version $v + 1$ and marks version v superseded in the same transaction. The public verification page (reached through a QR code holding the document URL) reports valid, superseded, revoked, tampered or not found. A card is tampered if its hash or signature fails or, for a current card, its body differs from the record re-derived from the live tables; the status and the superseded-by link are database columns, not signed.

E. Early warning

Before the end-semester examination, every enrolment is scored from attendance, the course's internal percentage and the student's mean internal percentage, using a class-weighted L2 logistic regression trained on earlier sessions (label: course failed). For a linear model on standardized features, $w_i z_i$ is the exact Shapley value relative to the training mean under feature independence [35], so the displayed explanation is exact for this model. Scores are meant to trigger support only.

IV. Evaluation Methodology

All experiments run the evaluated code unchanged. Pure engines are called through a Node.js bridge that feeds them JSON lines; database experiments run the full server on a dedicated PostgreSQL 18.4 database. Generators are seeded; values drawn inside the database (script allocation, dummy and document numbers, signing keys) are not. Every per-case output is stored and all reported numbers are recomputed from those files by one script. Hardware: one desktop PC (12 logical cores, Windows 11, Node.js 26.1.0, Python 3.11.9), with load generator and server on the same machine.

A. E1: rule-engine correctness (G1)

We wrote a reference implementation of the result semantics in Python with exact rational arithmetic and HALF_UP rounding, from the project's written specification. Its author had read the JavaScript beforehand, so the reference is independent in arithmetic and code but not blind to the implementation. Three switches, fixed before the first comparison, capture points where the literal specification and a plausible implementation differ: (a) pass minimum real-valued ($m \geq pM/100$) or integer ($\lceil pM/100 \rceil$); (b) rounding of exact or of binary floating-point values (the latter mirrors JavaScript's arithmetic); (c) relative cut-offs compared exactly or after rounding to two decimals. Six conventions on which the specification is silent were fixed to match the code: internal absence counts as zero; the grace need is the smallest sufficient amount; relative grading uses the population SD; absent courses count towards ATKT; a grace pass receives the lowest pass grade; and the revaluation tie-break takes the first closest pair. Each disagreement is attributed to the smallest set of switches that removes it; a disagreement no switch removes would be a defect. We compare all 15 course and semester fields printed on a grade card on 120,000 random student-semesters (4–8 courses from six templates including integrated theory–practical courses; boundary marks in half the courses; absences and missing marks; 0–5 earlier semesters) under six policy variants. We also check invariants, compare the greedy grace allocation with a brute-force optimum, apply a metamorphic relation (half a mark more should not lower the SGPA), and evaluate the revaluation rule on every half-mark combination for $M \in \{25, 60\}$, both modes and both decrease options (strided in double mode), calling a third examiner only when the rule asks for one.

B. E2: pre-publication scrutiny (G2)

No public data contain question-wise university marks with examiner identities, so we use a generator: six courses with persistent difficulty plus between-session variation (SD 2 points), 120 students, 4 examiners per course with random allocation, six 10-mark questions, ability-linked internal marks,

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

and a pass-rate history of three prior sessions (computed without grace, unlike the service). Each session receives a single error whose target we record; 300 sessions per scenario plus as many clean sessions are run, and we report the share of sessions in which the injected entity is flagged, with Clopper–Pearson intervals. The exact checks are functional tests (the injection creates the inconsistency they compare); only the statistical checks are estimated. For the examiner test we also run 1,000 sessions per point of a grid over shift and scripts per examiner, compare with the analytic power of the shipped statistic (whose variance is $(1-n_e/N)$ under the null because the examiner is part of the course mean), and evaluate a leave-examiner-out statistic on the same data. For the pass-rate rule we vary the between-session SD (2, 4, 6) and compare the shipped rule with a binomial rule ($|z| > 3$ against the pooled history).

C. E3: tamper evidence (G3)

Threat model. An insider can run SQL in the application's database, the system's realistic weak point because it connects as the database superuser. We distinguish three capabilities: **W**, write access to result, document and ledger tables (any role able to run the application's own statements); **K**, W plus write access to the key and settings tables; **O**, table owner or superuser (disable triggers, `TRUNCATE`). Since the application itself writes the key and settings tables, K equals W in the shipped deployment. The adversary does not hold the Ed25519 private key, which stays on the application server. Out of scope: compromise of the application server or key, network attackers and denial of service. Following forensic studies of database tampering [21], but without their external notariser, we execute each attack inside a transaction on the live instance, run the shipped verifiers in that transaction and roll back. Before the attacks we apply 25 revaluations through the shipped re-issue path, so that superseded cards exist. For ledger attacks we also check a *fresh* anchor (head hash and count captured just before the attack) and a *stale* anchor captured 20 entries earlier, the situation between periodic anchors. We further apply five legitimate administrative operations to measure false tamper verdicts, and test token minting and rate limits over HTTP.

D. E4: scale and load (G4)

We process, approve and publish synthetic cohorts of 100–10,000 students (six courses each) one after another in one growing database, so set-up times depend on run order; to isolate the script-coding statement we time it alone on a fresh database for different numbers of existing accounts, with and without materializing its examiner list. Load is measured alone on the large database (32,920 students), 3×15 s per point, with closed-loop clients without think time on the results and dashboard endpoints, plus login throughput and login latency during a publication.

E. E5: early warning on public data

MarkSure's demo data are synthetic, so we use two public datasets as proxies. UCI Student Performance [36] (Portuguese $n = 649$, Mathematics $n = 395$; label final grade $< 10/20$) supplies period grades and absences: attendance $\mapsto -$absences, internal percentage $\mapsto$ first-period grade (*early*; the two internal features then coincide) or second-period grade (*late*). UCI Dropout and Academic Success [37], [38] ($n = 4{,}244$ with at least one first-semester unit; label dropout) supplies first-semester grade, approved and evaluated units. We also report the subset with at least one approved unit, because students without one have often already left. The task differs from MarkSure's (student-level outcome instead of per-course failure). We compare the shipped model with scikit-learn [39] logistic regression, random forest [40] and XGBoost [41] on the same three features, and XGBoost on all pre-outcome features, with 5×5-fold stratified cross-validation and the shipped threshold of 0.5. We report ROC-AUC, at-risk recall, expected calibration error [42] with 10 equal-width bins, explanation additivity and recall by sex [31].

F. E6: workflow and access control (G4)

We run the project's unit and end-to-end tests and 15 functional negative tests from the OWASP Top 10 categories [43]; they are functional checks, not a security audit.

V. Results

A. E1: exact up to three unwritten conventions

Under the implementation's conventions the engine agreed with the reference on 100.00 % of the 120,000 student-semesters, for every printed field. Read literally, the specification differed on 5,137 cases (agreement 93.85–99.05 % across the six policies); every disagreement was removed by one switch or a combination (0 unexplained): 4,925 by integer pass minima (with a 35 % minimum on a 70-mark paper, 24.5 marks fail in the code but pass under the literal rule), 169 by binary rounding (a CGPA whose exact value ends in 5 at the third decimal can be rounded down because its binary value lies just below the tie), 35 by rounded relative cut-offs and 8 by combinations. None is wrong in itself, but each is a decision an ordinance must state and the policy schema cannot express. No invariant was violated. The greedy grace allocation matched the brute-force optimum on all 95,622 instances with candidates, including the 3,137 in which the budget or course cap forced a choice (3,137 optimal). The revaluation rule agreed on all 10,910,198 decisions (0 mismatches).

Non-monotone grace. In 7 of 20,000 default-policy cases (and 1 of 20,000 with generous grace; 0 without grace), half a mark more produced a lower SGPA: the smaller need reorders the candidates and the budget moves to a course with fewer credits, so a course with more credits fails. The allocation is optimal for the ordinance's objective, but that objective ignores credits.

B. E2: exact checks reliable, statistical checks weak or noisy

Table II lists detection per injected error. The exact checks found every transfer error that changed the recorded total (the one miss is an injection on a zero-mark script, which left the total unchanged), every absent student with marks, a functional

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

TABLE II
E2: SHARE OF 300 SYNTHETIC SESSIONS IN WHICH THE INJECTED ENTITY WAS FLAGGED. UPPER BLOCK: EXACT CHECKS (FUNCTIONAL); LOWER BLOCK: STATISTICAL CHECKS. *IN PARENTHESES: INFLATED STUDENTS FLAGGED.

Injected error	Check	Sessions detected % [95% CI]
Total transfer error	question sum	99.7 [98.2, 100.0]
Absent but marks recorded	absent check	100.0 [98.8, 100.0]
Flat marking by one examiner	low spread	100.0 [98.8, 100.0]
Lenient examiner, +5 pts	examiner z	1.0 [0.2, 2.9]
Lenient examiner, +10 pts	examiner z	22.7 [18.1, 27.8]
Strict examiner, -5 pts	examiner z	1.7 [0.5, 3.8]
Strict examiner, -10 pts	examiner z	24.3 [19.6, 29.6]
Internals inflated (5 students)*	int.–ext. gap	98.3 [96.2, 99.5] (87)
Course-wide scaling $\times 0.8$	pass-rate shift	94.0 [90.7, 96.4]
Digit swap in an internal mark	int.–ext. gap	2.3 [0.9, 4.7]

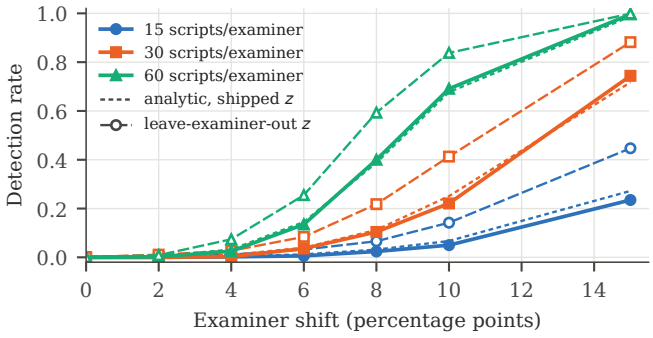


Fig. 1. E2: Detection of one shifted examiner (four examiners per course). Solid: shipped statistic; dotted: its analytic power; dashed with hollow markers: leave-examiner-out statistic (1,000 sessions per point).

confirmation; after we fixed a generator artifact, clean sessions raised 0 blocking flags. The low-spread rule flagged every flat-marking examiner, the internal–external gap rule flagged at least one of five inflated students in 98.3 % of sessions (86.7 % individually), the pass-rate rule caught a course-wide scaling error in 94.0 %, and a digit swap in an internal mark was caught in only 2.3 %, since faculty-entered marks have no question-wise record.

Examiner test. With 30 scripts per examiner, a 10-point lenient or strict shift was flagged in 22.7 and 24.3 % of sessions, a 5-point shift in 1.0 and 1.7 %. Fig. 1 shows that the empirical rate follows the analytic power of the shipped statistic (largest deviation 3.7 points). Because the examiner is part of the course mean, the statistic's null variance is $1-n_e/N$, so "$|z| > 3$" acts as a stricter test. A leave-examiner-out statistic raised detection at a 10-point shift from 5/22/69 % to 14/41/84 % for 15/30/60 scripts, but flagged 0.004 innocent examiners per session without a shift (shipped statistic: 0.000).

Pass-rate rule. Clean sessions produced 1.75 pass-rate warnings each. This needs no simulation: with three earlier sessions and the population SD, a stable course is flagged with probability $P(|t_2| > \sqrt{2}) = 0.293$, i.e. 1.76 per six-course session. The binomial rule reduced false alarms to 0.19 per session at a between-session SD of 2 points (recall 83 %), but to only 0.89 at SD 4 and 1.66 at SD 6, where the shipped rule gave 1.75 and 1.67; its advantage depends on how stable real pass rates are.

C. E3: direct edits detected, trust root and metadata exposed

Table III summarizes the attacks; outcomes are deterministic, so "Trials" shows coverage (distinct targets in parentheses), not statistical replication. All 121 cards verified with their recorded status beforehand (median verification 2.0 ms). Every direct edit of a current card was reported as tampered, whether it changed the result tables (caught by live re-derivation), the signed body (hash) or body and hash together (signature). Three attacks produced cards that verified as *valid*: an attacker with capability K inserted an own public key and re-signed an edited body (25 of 25); an attacker with W undid an accepted revaluation by restoring the old card, the old result versions and the current-version flags (25 of 25, on 20 cards); and edits between approval and publication were blocked by the snapshot check unless the attacker also recomputed the stored snapshot, in which case publication signed the edited result. Even a one-statement status flip revived an old card in 5 of 25 trials (4 of 17 cards): all were cards re-issued after a revaluation in an earlier semester, which the verifier re-derives from their own, no longer current, semester record; 3 of the 4 carried an outdated CGPA. Unsigned metadata also allowed a valid body under a superseded number with another student's name, silent revocation and deletion of a card. Over HTTP, a token signed with the secret read from the settings table was accepted as the CoE (HTTP 200), and with a spoofed `X-Forwarded-For` per request the public rate limit never engaged in 90 requests; the verification log recorded the spoofed address.

The append-only trigger rejected every in-place update (50/50), and edits of hashed content or deletions inside the chain were located. Deleting the newest entries, editing the unhashed role and address columns, rewriting all later hashes, appending forged entries and `TRUNCATE` (which bypasses row-level triggers and leaves an empty chain that verifies) went undetected. A fresh anchor caught all but the unhashed-column edits and the forged appends; a stale anchor, 20 entries old, missed deletions and rewrites confined to those entries, which is the realistic case between periodic anchors. Because the hash joins fields with "|", two different entries can share a hash; we constructed such a pair. Verifying 1,000,000 entries took 10.7 s and 1,457 MB, as the verifier loads the whole table.

False tamper verdicts. Legitimate administration turned genuine cards into "tampered": renaming a published session (the update the API's rename endpoint performs) affected 60/60 valid cards of that session, correcting student names 60/60, recording ABC IDs 60/60, renaming the program 60/60 and correcting course titles 60/60, because identity fields are part of the re-derived body.

D. E4: processing scales, set-up and the results page do not

Table IV and Fig. 2 report the lifecycle at scale. The largest cohort (10,000 students) was processed in 12.3 s and

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

TABLE III
E3: ATTACKS BY AN INSIDER WITHOUT THE SIGNING KEY. CAP.: W WRITE DATA TABLES, K ALSO KEY/SETTINGS TABLES, O OWNER OR SUPERUSER. FRESH/STALE: DETECTED WITH A HEAD ANCHOR TAKEN JUST BEFORE / 20 ENTRIES BEFORE THE ATTACK (SIMULATED; FOR THE FIRST BLOCK AND THE UPPER LEDGER ROWS, RUN ON THE 121-CARD DATABASE BEFORE THE REVALUATIONS, THE STALE VALUE FOLLOWS FROM THE VERIFIER OUTCOME). ‡FORGERY VERIFIED AS VALID.

Attack	Cap.	Trials	Shipped verifier outcome	Fresh	Stale
Grade edited in result tables	W	50 (43)	tampered	n/a	n/a
Signed card body edited	W	50 (43)	tampered	n/a	n/a
Body edited, hash recomputed	W	50 (44)	tampered	n/a	n/a
Other card's signed body, current no.	W	50 (40)	tampered	n/a	n/a
Superseded card set to valid (naive)‡	W	25 (17)	tampered/valid	n/a	n/a
Attacker key inserted, card re-signed‡	K	25	valid	n/a	n/a
Revaluation undone consistently‡	W	25 (20)	valid	n/a	n/a
Edit after approval	W	3	publication refused	n/a	n/a
Edit after approval, snapshot recomputed‡	W	3	published, valid	n/a	n/a
Other card's signed body, superseded no.	W	25 (19)	superseded	n/a	n/a
Valid card set to revoked	W	50 (42)	revoked	n/a	n/a
Card row deleted	W	25	not found	n/a	n/a
Ledger UPDATE, trigger active	W	50 (40)	rejected	–	–
Ledger entry edited	O	50 (37)	break located	yes	yes
Middle entry deleted	O	50 (41)	break located	yes	yes
Actor role / IP edited	O	50 (44)	chain verifies	no	no
Forged entry appended	W	10 (6)	chain verifies	no	no
Newest 1–20 entries deleted	O	10 (7)	chain verifies	yes	no
Edit + later hashes recomputed	O	10	chain verifies	yes	9/10
TRUNCATE	O	10 (1)	chain verifies	yes	yes

TABLE IV
E4: MEDIAN TIME PER SYNTHETIC COHORT (SIX COURSES PER STUDENT; RUNS SHARE ONE GROWING DATABASE).

Students	Runs	Set-up (s)	Process (s)	Publish (s)	Collisions
100	3	0.1	0.10	0.31	0
500	3	0.6	0.52	1.56	0
1,000	3	1.8	0.45	3.14	0
2,000	3	17.9	2.78	8.51	0
5,000	2	177.7	6.23	28.04	3
10,000	1	886.7	12.27	35.72	3

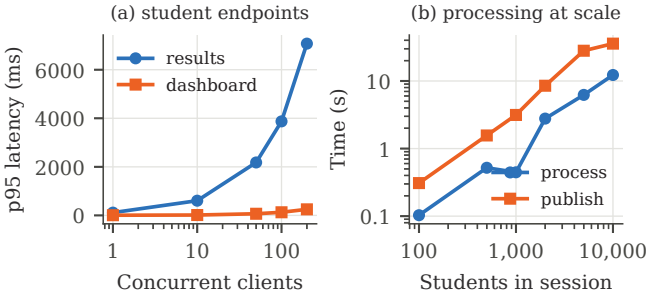


Fig. 2. E4: (a) 95th-percentile latency on the large database (median of 3 runs); (b) processing and publication time versus cohort size.

published, with one signed card, ledger entry and notification per student, in 35.7 s. Session set-up took 1.8 s at 1,000 students but 887 s at 10,000. Timed alone on a fresh database, the script-coding statement for 2,000 students (10,000 scripts) took 1.55 s with 2,070 accounts and 157.1 s with 42,070; materializing the examiner list reduced this to 0.23 and 1.14 s. The cause is the query plan: the examiner list, a once-referenced common table expression, is inlined into a sub-plan that scans all accounts once per script. Dummy numbers are eight random hexadecimal digits under a global uniqueness constraint, and one collision aborts script coding: our first 5,000-student run failed this way, and with a retry 6 collisions occurred in 2 runs. By the birthday bound a 10,000-student session collides with itself with probability 25 % and with 50,000 existing scripts 58 %; document numbers share the defect (1.2 % per 10,000-card publication).

On the large database the results page sustained only 31 requests/s (single-client median 105.8 ms; 95th percentile 7075 ms at 200 clients), against 888 for the dashboard, because each request recomputes class averages over every result of the student's sessions. Login peaked at 19.0 requests/s (password hashing in the event loop), and while 2,000 cards were published (11.3 s) the global ledger lock delayed logins by up to 11.2 s. No request failed; single-machine figures bound rather than predict a deployment.

E. E5: competitive model, proxy and fairness caveats

The shipped model's ROC-AUC ranged from 0.860 to 0.974 (Table V), within 0.001 of scikit-learn's logistic regression on the same features; on the school data it matched or exceeded the all-feature model. On the dropout data, XGBoost with all features gained 0.051. Excluding the 538 students without an approved unit lowered the shipped model's AUC from 0.860 to 0.795 and its recall from 0.72 to 0.66 when retrained on the subset (0.56 for the full-data model on the same students), against 0.869 for the all-feature model. The attendance proxy received a positive weight (0.052), so its explanation would say that more evaluated units raise risk. Explanations reproduced the logit to within 0.0015 (display rounding). Calibration was moderate (ECE 0.097–0.143). On the dropout data recall was 0.66 for women and 0.78 for men, with false-positive rates 0.11 and 0.17, although sex is not an input.

F. E6: workflow and access control

All 16 unit tests, all 21 end-to-end lifecycle checks and 15 of 15 negative tests passed: role violations, forged or altered tokens, re-processing of published sessions and SQL meta-characters were rejected, identifiers were masked, and lock-out and rate limits engaged for a single client address.

VI. DISCUSSION

Rules as data need explicit conventions. The differential test surfaced decisions nobody had written down, as formalizing legislation does [14]. Threshold rounding, decimal (not binary [44]) arithmetic for grade points, cut-off precision and the grace objective should be policy fields; a credit-aware grace objective removes the non-monotonicity.

Statistical scrutiny should be redesigned, not tuned. Comparing an examiner with the rest of the course fixes the mis-standardized statistic and raises power 1.2–2.8 times, at the price of more false flags, but moderate leniency remains hard to see at typical script counts, and allocation by center or college would confound examiners with cohorts; rater models [27] and pooling across courses are better foundations. Pass-rate rules need calibration on the institution's own history.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

TABLE V
E5: ROC-AUC (MEAN OF 5×5-FOLD CV). MS: SHIPPED MODEL (REC.: AT-RISK RECALL AT 0.5); "-3": SAME THREE FEATURES; "-ALL": ALL PRE-OUTCOME FEATURES.

Data	n	Risk %	MS AUC	MS rec.	LR-3	RF-3	XGB-3	XGB-all
Por., early	649	15	0.944	0.88	0.943	0.933	0.936	0.931
Por., late	649	15	0.972	0.94	0.971	0.970	0.959	0.959
Mat., early	395	33	0.923	0.82	0.923	0.928	0.925	0.913
Mat., late	395	33	0.974	0.94	0.975	0.975	0.968	0.974
Dropout	4,244	32	0.860	0.72	0.860	0.863	0.861	0.911
Dropout, >0 approved	3,706	23	0.795	0.66	0.795	0.799	0.795	0.869

Because marking is noisy [28], a flag should start a review, not imply misconduct.

Integrity needs a trust root outside the database. Re-deriving a card from live data detected every direct edit of a current card, but E3 shows that the guarantee rests on assumptions the code does not enforce. Remedies, ordered by the attacks they stop: (i) pin the verification keys in the verifier's configuration or code, with key validity periods and revocation, instead of trusting a writable table, and let relying parties check signatures offline; (ii) sign state changes (supersession, revocation) as new signed statements rather than editing status columns, and bind the document number into the checked body; (iii) publish the signed ledger head and count after every session and make the verifier compare against the latest published head [22], [24]; since entries written after the last anchor remain unprotected, anchor at every approval and publication; (iv) authenticate entries (a MAC or signature by the writing service) so that forged appends are attributable, and encode fields unambiguously (length-prefixed or a JSON array); (v) bind the approval snapshot to the CoE's signature so that publication cannot follow a silent edit; (vi) keep the secret that signs login tokens out of the database; (vii) re-derive only result fields, or re-issue a card when identity data change, to avoid false tamper verdicts; and (viii) run the application as a non-owner role without `TRUNCATE` or trigger privileges. Item (viii) alone does not stop key insertion, token minting or forged appends. A blockchain is unnecessary for a single issuer [23]; what matters is independent custody of keys and anchors.

Scale defects appear only at scale. They are invisible in a 60-student demonstration. Materializing the examiner list, sequence-based or 128-bit numbers, class averages computed once per publication, a hashing worker pool and per-session ledger appends remove them.

Early warning. Three features ranked students about as well as richer models on the school data, but on the dropout data the advantage came largely from already-disengaged students, the attendance proxy got an implausible sign, and recall differed by sex; predictions should only trigger reviewed support.

Limitations. Scrutiny and scale data are synthetic, generated by our own scripts, and their realism is an assumption. The early-warning datasets are proxies from Portuguese education. One code snapshot was tested on one desktop, with client and server on the same machine; scale runs shared one growing database. The reference implementation was written by someone who had read the code. We did not encode real university ordinances, run a usability study, compare with manual processing or measure real revaluation turnaround. Payments, SMS/e-mail and the national depository interface are simulated or absent.

VII. CONCLUSION

Measuring MarkSure against four guarantees shows that its engine is exact relative to conventions that should become policy options, its exact checks are reliable while the statistical ones need redesign, its signed grade cards reveal direct edits of current cards but not an insider who can write the key table, roll back a revaluation or edit results before publication, and its ledger proves edits, not completeness, until its head is anchored externally. We hope the evaluation protocol is useful for other academic record systems.

ACKNOWLEDGMENT

An AI system (Claude Opus 5.5, Anthropic) assisted with the evaluation scripts, the generation of tables and figures from saved outputs, the bibliography script, drafting and editing of all sections, and simulated reviews. MarkSure itself was developed by the authors, who checked the numbers against the raw outputs and the citations against their DOI or arXiv records and take full responsibility for the paper.

REFERENCES

[1] University Grants Commission, "UGC guidelines on adoption of choice based credit system," University Grants Commission, New Delhi, 2015, accessed 2026-11-08. [Online]. Available: https://www.ugc.gov.in/pdfnews/8023719_Guidelines-for-CBCS.pdf

[2] K. C. Ukaoha and I. F. Amadin, "Computerized result processing system: A case study of the department of computer science, University of Benin," in *2014 IEEE 6th International Conference on Adaptive Science & Technology (ICAST)*. IEEE, Oct. 2014, pp. 1–4. [Online]. Available: https://doi.org/10.1109/icastech.2014.7068148

[3] H. Xu, R. Guo, L. Lei, and R. Fan, "Design and implementation of examination affairs management system," in *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*. IEEE, Jul. 2016, pp. 294–296. [Online]. Available: https://doi.org/10.1109/icisce.2016.72

[4] S. B. Akki and M. Vijayalakshmi, "Design of dashboard for University examination result analysis system," in *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*. IEEE, Dec. 2018, pp. 313–316. [Online]. Available: https://doi.org/10.1109/csitss.2018.8768498

[5] V. V. Kamat and K. G. Dessai, "E-Moderation of Answer-Scripts evaluation for controlling Intra/Inter examiner heterogeneity," in *2018 IEEE Tenth International Conference on Technology for Education (T4E)*. IEEE, Dec. 2018, pp. 130–133. [Online]. Available: https://doi.org/10.1109/t4e.2018.00035

[6] P. S. Nayak, T. M.R., and N. Kumar, "Flexible question wise evaluation of digitized answer scripts," in *2018 IEEE Tenth International Conference on Technology for Education (T4E)*. IEEE, Dec. 2018, pp. 216–217. [Online]. Available: https://doi.org/10.1109/t4e.2018.00058

[7] M. Turkanovic, M. Holbl, K. Kosic, M. Hericko, and A. Kamisalic, "EduCTX: A Blockchain-Based higher education credit platform," *IEEE Access*, vol. 6, pp. 5112–5127, 2018. [Online]. Available: https://doi.org/10.1109/access.2018.2789929

[8] A. Rustemi, F. Dalipi, V. Atanasovski, and A. Risteski, "A systematic literature review on Blockchain-Based systems for academic certificate verification," *IEEE Access*, vol. 11, pp. 64 679–64 696, 2023. [Online]. Available: https://doi.org/10.1109/access.2023.3289598

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

[9] K. E. Arnold and M. D. Pistilli, "Course signals at purdue: using learning analytics to increase student success," in *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, ser. LAK 2012. ACM, Apr. 2012, pp. 267–270. [Online]. Available: https://doi.org/10.1145/2330601.2330666

[10] C. Romero and S. Ventura, "Educational data mining and learning analytics: An updated survey," *WIREs Data Mining and Knowledge Discovery*, vol. 10, no. 3, Jan. 2020. [Online]. Available: https://doi.org/10.1002/widm.1355

[11] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," *ACM Transactions on Information and System Security*, vol. 2, no. 2, pp. 159–176, May 1999. [Online]. Available: https://doi.org/10.1145/317087.317089

[12] R. T. Snodgrass, S. S. Yao, and C. Collberg, "Tamper detection in audit logs," in *Proceedings 2004 VLDB Conference*. Elsevier, 2004, pp. 504–515. [Online]. Available: https://doi.org/10.1016/b978-012088469-8.50046-2

[13] D. Calvanese, M. Dumas, Ü. Laurson, F. M. Maggi, M. Montali, and I. Teinemaa, "Semantics, analysis and simplification of DMN decision tables," *Information Systems*, vol. 78, pp. 112–125, Nov. 2018. [Online]. Available: https://doi.org/10.1016/j.is.2018.01.010

[14] D. Merigoux, N. Chataing, and J. Protzenko, "Catala: a programming language for the law," *Proceedings of the ACM on Programming Languages*, vol. 5, no. ICFP, pp. 1–29, Aug. 2021. [Online]. Available: https://doi.org/10.1145/3473582

[15] D. Merigoux, R. Monat, and J. Protzenko, "A modern compiler for the french tax code," in *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction*, ser. CC '21. ACM, Feb. 2021, pp. 71–82. [Online]. Available: https://doi.org/10.1145/3446804.3446850

[16] X. Yang, Y. Chen, E. Eide, and J. Regehr, "Finding and understanding bugs in C compilers," in *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '11. ACM, Jun. 2011, pp. 283–294. [Online]. Available: https://doi.org/10.1145/1993498.1993532

[17] K. Claessen and J. Hughes, "QuickCheck: a lightweight tool for random testing of Haskell programs," in *Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, ser. ICFP00. ACM, Sep. 2000, pp. 268–279. [Online]. Available: https://doi.org/10.1145/351240.351266

[18] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys*, vol. 51, no. 1, pp. 1–27, Jan. 2018. [Online]. Available: https://doi.org/10.1145/3143561

[19] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, Jul. 2003. [Online]. Available: https://doi.org/10.1023/a:1022883727209

[20] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *Journal of Cryptology*, vol. 3, no. 2, pp. 99–111, Jan. 1991. [Online]. Available: https://doi.org/10.1007/bf00196791

[21] K. E. Pavlou and R. T. Snodgrass, "Forensic analysis of database tampering," *ACM Transactions on Database Systems*, vol. 33, no. 4, pp. 1–47, Nov. 2008. [Online]. Available: https://doi.org/10.1145/1412331.1412342

[22] P. Antonopoulos, R. Kaushik, H. Kodavalla, S. Rosales Aceves, R. Wong, J. Anderson, and J. Szymaszek, "SQL ledger: Cryptographically verifiable data in Azure SQL database," in *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD/PODS '21. ACM, Jun. 2021, pp. 2437–2449. [Online]. Available: https://doi.org/10.1145/3448016.3457558

[23] X. Yang, Y. Zhang, S. Wang, B. Yu, F. Li, Y. Li, and W. Yan, "LedgerDB: a centralized ledger database for universal audit and verification," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3138–3151, Aug. 2020. [Online]. Available: https://doi.org/10.14778/3415478.3415540

[24] B. Laurie, A. Langley, and E. Kasper, *Certificate Transparency*, Jun. 2013. [Online]. Available: https://doi.org/10.17487/rfc6962

[25] M. Sporny, T. T. Jr, I. Herman, M. B. Jones, and G. Cohen, "Verifiable credentials data model v2.0," W3C Recommendation, May 2025, accessed 2026-11-08. [Online]. Available: https://www.w3.org/TR/vc-data-model-2.0/

[26] Ministry of Education, Government of India, "National academic depository (NAD)," Online depository of academic awards, operated through DigiLocker, 2026, accessed 2026-11-08. [Online]. Available: https://nad.digilocker.gov.in/

[27] G. Engelhard, "Examining rater errors in the assessment of written composition with a Many-Faceted Rasch model," *Journal of Educational Measurement*, vol. 31, no. 2, pp. 93–112, Jun. 1994. [Online]. Available: https://doi.org/10.1111/j.1745-3984.1994.tb00436.x

[28] J.-A. Baird, J. Greatorex, and J. F. Bell, "What makes marking reliable? experiments with UK examinations," *Assessment in Education: Principles, Policy & Practice*, vol. 11, no. 3, pp. 331–348, Sep. 2004. [Online]. Available: https://doi.org/10.1080/0969594042000304627

[29] 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," *Review of Educational Research*, vol. 80, no. 2, pp. 272–295, Jun. 2010. [Online]. Available: https://doi.org/10.3102/0034654310362998

[30] R. S. Baker and A. Hawn, "Algorithmic bias in education," *International Journal of Artificial Intelligence in Education*, vol. 32, no. 4, pp. 1052–1092, Dec. 2022. [Online]. Available: https://doi.org/10.1007/s40593-021-00285-9

[31] J. Gardner, C. Brooks, and R. Baker, "Evaluating the fairness of predictive student models through slicing analysis," in *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*, ser. LAK19. ACM, Mar. 2019, pp. 225–234. [Online]. Available: https://doi.org/10.1145/3303772.3303791

[32] A. Rundgren, B. Jordan, and S. Erdtman, "JSON canonicalization scheme (JCS)," RFC Editor, Request for Comments RFC 8785, Jun. 2020. [Online]. Available: https://doi.org/10.17487/rfc8785

[33] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, Aug. 2012. [Online]. Available: https://doi.org/10.1007/s13389-012-0027-1

[34] S. Josefsson and I. Liusvaara, "Edwards-Curve digital signature algorithm (EdDSA)," RFC Editor, Request for Comments RFC 8032, Jan. 2017. [Online]. Available: https://doi.org/10.17487/rfc8032

[35] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," arXiv:1705.07874, 2017. [Online]. Available: https://arxiv.org/abs/1705.07874

[36] P. Cortez, "Student performance," 2008. [Online]. Available: https://doi.org/10.24432/C5TG7T

[37] V. Realinho, J. Machado, L. Baptista, and M. V. Martins, "Predicting student dropout and academic success," *Data*, vol. 7, no. 11, p. 146, Oct. 2022. [Online]. Available: https://doi.org/10.3390/data7110146

[38] V. Realinho, M. Vieira Martins, J. Machado, and L. Baptista, "Predict students' dropout and academic success," 2021. [Online]. Available: https://doi.org/10.24432/C5MC89

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," arXiv:1201.0490, 2012. [Online]. Available: https://arxiv.org/abs/1201.0490

[40] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/a:1010933404324

[41] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. ACM, Aug. 2016, pp. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[42] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," arXiv:1706.04599, 2017. [Online]. Available: https://arxiv.org/abs/1706.04599

[43] OWASP Foundation, "OWASP top 10:2021," Open Worldwide Application Security Project, 2021, accessed 2026-11-08. [Online]. Available: https://owasp.org/Top10/2021/

[44] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, Mar. 1991. [Online]. Available: https://doi.org/10.1145/103162.103163

(This work is licensed under a Creative Commons Attribution 4.0 International License.)