www.ijarp.com ISSN 2456-9992 Page: 01-16

International Journal Advanced Research Publications

Case Study

Volume: 02
Issue: 07

DESIGN AND IMPLEMENTATION OF AN ELECTRONIC RECEIPT SYSTEM (CASE STUDY OF ESPAM FORMATION UNIVERSITY, PORTO-NOVO)

***Emmanuel Praise-God, Popoola Olusegun Victor, Owolola Lawrence, Olalere Sherifdeen Abiodun, Anoliefo Chukwuma Josemaria**

Department of Computer Science, ESPAM Formation University, Porto-Novo, Republic of Benin.

Article Received: 09 June 2026, Article Revised: 29 June 2026, Published on: 19 July 2026

***Corresponding Author: Emmanuel Praise-God**

Department of Computer Science, ESPAM Formation University, Porto-Novo, Republic of Benin.

Doi: https://doi-doi.org/101555/ijarp.3919

ABSTRACT

Manual, weakly integrated receipt processes cause delays, duplicate numbers, errors, reconciliation issues, document loss, and fraud risks. This study details a web-based electronic receipt system for ESPAM Formation University, Porto-Novo, developed iteratively through problem analysis, requirements modeling, database design, role-based interfaces, security, prototyping, and testing. The system links student records, confirmed payments, immutable receipt IDs, cryptographic hashes, QR codes, and audit logs within a three-tier architecture. Key controls include unique constraints, least privilege access, password hashing, secure sessions, input validation, transaction atomicity, and separation of receipt issuance from payment approval. A prototype with 5,000 synthetic records achieved transaction times of 0.278 ms, receipt lookup of 0.008 ms, and receipt-plus-QR generation of 6.454 ms. All 50 constraint tests rejected duplicates, showing the system's robustness. Results suggest a governed electronic receipt platform enhances traceability, retrieval, reconciliation, and anti-forgery. The paper presents a prototype and development roadmap needing stakeholder validation, payment integration, legal review, testing, backups, and phased acceptance before deployment.

KEYWORDS: Electronic Receipt, University Finance, Payment Verification, Qr Code, Audit Trail, Database Security, Porto-Novo.

1. INTRODUCTION

Universities receive tuition, acceptance, transcript, hostel, development levy, and other payments that must be acknowledged accurately. A receipt is not merely a printed slip; it is an accounting record connecting a payer, an obligation, a payment reference, an amount, an approval event, and a time of issuance. Where this chain is handled manually or across disconnected spreadsheets and paper files, the institution cannot reliably guarantee uniqueness, rapid retrieval, consistent reconciliation, or resistance to unauthorised modification. ESPAM Formation University describes itself as a bilingual private university established in 2007, with a Porto-Novo campus and programmes that include Computer Science and Business Administration [1]. This institutional setting motivates a receipt system that is simple enough for routine bursary work but sufficiently controlled for audit, student verification, and future integration with online payment channels. The present paper therefore designs and implements a prototype electronic receipt system tailored to a university environment. The central research question is: how can an electronic receipt platform be designed so that every confirmed payment yields one traceable, verifiable, and retrievable receipt without introducing avoidable complexity? The objective is not to digitise a weak manual process verbatim. It is to redesign the process with a focus on data integrity, accountability, security, and service quality.

The study is significant for three reasons. First, the issuance of receipts is one of the most visible interfaces between a university and its students; persistent delays or inconsistencies immediately damage trust. Second, receipt data feed registration, clearance, financial reporting, and audit, so errors propagate beyond the bursary. Third, a well-designed electronic receipt service can serve as a controlled integration point among the student-information system, banking channels, mobile payment services, and the general ledger. The value, therefore, lies less in eliminating paper than in creating a reliable institutional record. The remainder of the paper reviews relevant electronic-receipt and software-quality concepts, defines requirements and business rules, presents the architecture and database model, explains the implementation, reports controlled tests, and concludes with risks and a production-adoption roadmap. This sequence preserves traceability from the problem statement to the reported evidence.

1.1 Problem Statement

A paper-dominant receipt process typically depends on handwritten entries, carbon copies, manual serial numbering, physical filing, and later spreadsheet reconciliation. Such a process is slow during registration peaks and structurally vulnerable. A duplicated serial number may

go unnoticed; an amount may be altered after issue; a receipt may be misplaced; and an auditor may be unable to reconstruct which staff member created or reprinted a document. Even where a computer is used to type receipts, the absence of a unified database and access-control model leaves the underlying weaknesses intact.

1.2 Aim and Objectives

The aim is to design and implement a secure, database-driven electronic receipt system for the Porto-Novo case-study environment. The specific objectives are to:

- model students, payments, receipts, users, and audit events in a normalised relational database;
- generate unique receipt numbers only after a payment has been confirmed;
- embed a cryptographic verification value and QR code in each receipt;
- provide role-based functions for finance officers, supervisors, administrators, auditors, and students;
- support search, reprint, summary reporting, and export without changing the original transaction; and
- Evaluate functional correctness, integrity controls, and prototype response time using synthetic records.

1.3 Contribution and Scope

The contribution is a complete, reproducible design that joins process controls and software controls. Many student projects stop at interface screenshots. That is inadequate. A credible receipt system must specify transaction boundaries, uniqueness rules, security roles, exception handling, auditability, and deployment controls. The proposed prototype covers payment recording, supervisory confirmation, receipt issuance, verification, reprint, reporting, and audit logging. It excludes live card processing, bank settlement, biometric identity proofing, and full integration with an existing student-information system. Those are production-stage extensions.

2. RELATED WORK AND CONCEPTUAL BASIS

Electronic receipts replace or complement paper evidence with machine-readable records that can be stored, searched, and transmitted. Wadsworth et al. [2] demonstrated an electronic receipt management concept aimed at reducing dependence on paper and improving receipt organisation. In educational settings, recent receipt-generation studies have emphasised OTP

or QR verification, although many provide limited evidence on database integrity, auditability, and operational governance [3]. The relevant lesson is that a QR code is not secure on its own: it is merely a carrier. The underlying verification endpoint, unique identifier, and protected database determine whether the receipt can be trusted.

The system is grounded in standard software-engineering principles. Requirements must be traceable to design decisions; the database must enforce invariants that the interface alone cannot guarantee; and quality must be assessed beyond cosmetic usability [4], [5]. ISO/IEC 25010 defines software quality in terms of characteristics such as functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability [6]. For this application, functional correctness, integrity, security, usability, and recoverability are the dominant considerations.

The DeLone–McLean information-systems success perspective further distinguishes system quality, information quality, service quality, use, user satisfaction, and net benefits [13]. Applied here, a receipt is successful only when the platform is dependable, the displayed financial information is accurate, users can complete tasks without avoidable support, and the resulting record improves institutional operations. A technically correct database with confusing forms would fail; equally, an attractive interface without strong integrity constraints would be unsafe. The design therefore balances human workflow and enforceable controls.

Table 1. Mapping of receipt-process weaknesses to proposed controls.

Manual-process weakness	System response
Handwritten or duplicated serial numbers	Database-generated unique receipt number with a UNIQUE constraint
Lost paper copies	Searchable central record and controlled reprint
Altered amount or payer details	Hash-based verification and an immutable issued receipt
Unknown issuing officer	User identity and timestamp in the audit log
Slow reconciliation	Daily, fee-type, and student-level reports
Unauthorised access	Role-based permissions, secure sessions, and account controls

3. METHODOLOGY AND REQUIREMENTS ANALYSIS

An iterative prototyping method was adopted. The stages were: (i) define the existing process risks; (ii) identify actors and use cases; (iii) specify functional and non-functional requirements; (iv) design the data model and architecture; (v) implement the prototype; and

(vi) test each requirement with controlled data. This approach is appropriate because finance users can judge forms and reports early, while developers can progressively harden data rules and security controls [4], [5].

3.1 Actors and Use Cases

Five actor classes are defined. A finance officer captures or imports a payment and requests confirmation. A finance supervisor confirms the validity of payments and authorises the issuance of receipts. An administrator manages users, fee types, academic sessions, and configuration. An auditor has read-only access to transactions and logs. A student or external verifier can view only the public verification result for a supplied receipt number or QR token. Separating capture and confirmation reduces the risk that a single user can create an unsupported payment and immediately issue an apparently valid receipt.

Table 2. Condensed functional and non-functional requirements.

ID	Requirement	Acceptance condition
FR1	Create and maintain student and fee records.	Required fields are validated, and duplicate student identifiers are rejected.
FR2	Record a payment against a student.	Amount is positive; reference is unique; status begins as pending or imported.
FR3	Confirm or reject a payment	Only an authorised supervisor can change approval status.
FR4	Generate one receipt for one confirmed payment	A confirmed payment has at most one receipt; receipt numbers are unique.
FR5	Verify, search, view and reprint	Receipt is retrieved by number, student, reference, date, or QR token.
FR6	Produce operational and audit reports	Totals reconcile to confirmed payments and preserve filter criteria.
NFR1	Security	Least privilege, hashed passwords, protected sessions, validated input, audit logs.
NFR2	Performance	Routine search and issuance are complete within an acceptable interactive delay.
NFR3	Reliability	Atomic transactions, backup, restore procedure, and controlled error handling.
NFR4	Usability	Consistent labels, minimal data entry, visible status, and printable output.

3.2 Business Rules

- A receipt shall not exist without a confirmed payment.
- One confirmed payment shall produce at most one active receipt.

- Payment reference, receipt number, and verification hash shall each be unique.
- Issued receipt values shall not be edited; corrections use reversal and reissue procedures.
- Every create, confirm, reject, reprint, reverse, export, and administrative action shall be logged.
- A public verification response shall expose only the minimum data needed to establish validity.
-

3.3 Receipt Number and Integrity Model

Let Y denote the four-digit year and n the next sequence value generated within a protected database transaction. The receipt number is defined as:

$R = \text{“EFU-RCP”} \,||\, Y \,||\, \text{zero-pad}(n, 7).$

For receipt number R, payment reference P, amount A, student identifier S, and server-side secret K, the prototype computes:

$H = \text{SHA-256}(R \,||\, P \,||\, \text{canonical}(A) \,||\, S \,||\, K).$

The QR payload carries a verification URL containing R and a truncated lookup token. The server recomputes or securely compares the stored value. The secret is not printed. In production, key rotation, a keyed hash such as HMAC, and a formal revocation procedure should replace a fixed prototype salt.

4. SYSTEM DESIGN

4.1 Architecture

The system uses a three-tier web architecture. The presentation layer provides responsive pages for data entry, approval, reporting, student viewing, and public verification. The application layer applies authentication, authorisation, business rules, receipt generation, notification, and audit logic. The data layer stores normalised records and enforces referential integrity, uniqueness, and transaction consistency. This separation improves maintainability and permits the database, web application, and document-generation service to be scaled or secured independently.

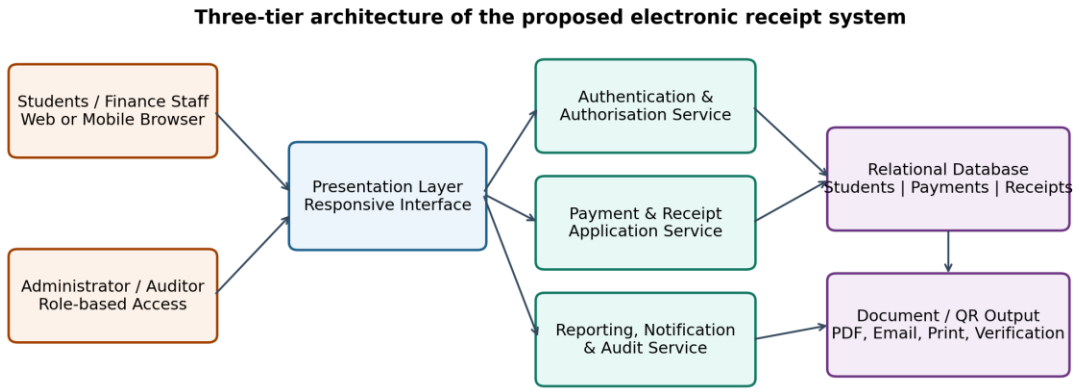

Figure 1. Proposed a three-tier architecture.

4.2 Operational Workflow

The critical path begins with authentication and ends with an audit event. Payment validation occurs before numbering because reserving receipt numbers for invalid transactions creates gaps and reconciliation disputes. The receipt and its audit record are written within the same transaction. Notification failure does not invalidate the financial record; it creates a retryable delivery event.

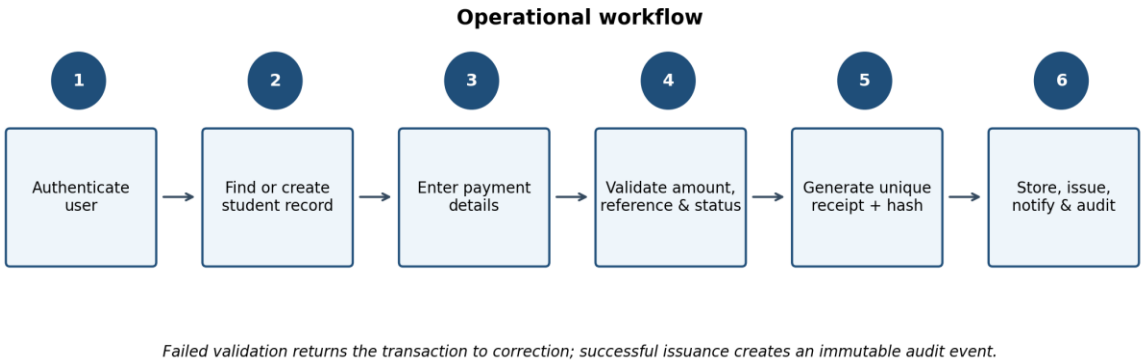

Figure 2. Receipt issuance workflow.

4.3 Security Design

Security is treated as a system property, not an afterthought. OWASP recommends explicit verification of authentication, access control, session management, validation, and other application controls [7]. Session identifiers should be renewed after authentication or changes in privileges and protected with secure cookie attributes [8]. NIST digital-identity guidance similarly emphasises authenticated sessions, secure recovery, and risk-appropriate controls

[9]. Accordingly, the design applies Argon2id password hashing [10], HTTPS, server-side authorisation checks, CSRF protection, rate limiting, generic login errors, account lockout thresholds, inactivity timeout, secure logout, parameterised queries, output encoding, and backup encryption.

Roles are not cosmetic menu filters. Every protected endpoint rechecks permission on the server. Finance officers cannot change user privileges; auditors cannot alter transactions; students cannot enumerate other students' records; and public verification reveals only receipt status, masked payer identity, fee description, amount, and issue date. Logs are append-oriented and excluded from ordinary update forms.

4.4 Database Design

The relational model separates identity, payment, receipt, user, and audit data. This avoids storing duplicate student details in every receipt and allows a student to make multiple payments, with each payment producing at most one receipt. Foreign keys preserve valid relationships, while unique indexes prevent duplicate payment references and receipt numbers. Database constraints are deliberately redundant with interface validation: the browser improves usability, but the database is the final guardian of integrity.

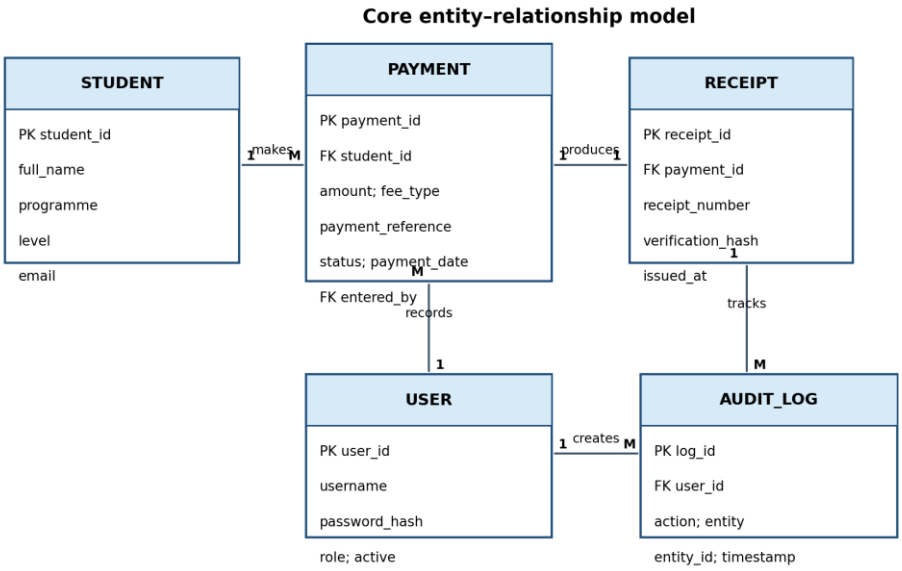

Figure 3. Core entity–relationship model.

4.5 Transaction Logic

Receipt issuance is executed as one atomic transaction. The application locks or rechecks the payment row, verifies that the status is CONFIRMED and no receipt already exists, obtains

the next protected sequence, computes the verification value, inserts the receipt, and writes the audit event. If any step fails, the transaction rolls back. In simplified pseudocode:

```
BEGIN TRANSACTION
  payment ← SELECT … FOR UPDATE
  IF payment. status ≠ CONFIRMED THEN ABORT
  IF receipt exists for payment THEN RETURN existing receipt.
  R ← next protected receipt number
  H ← keyed_hash(R, reference, amount, student_id)
  INSERT receipt(R, payment_id, H, issued_at)
  INSERT audit_log(user, 'ISSUE_RECEIPT', R, timestamp)
COMMIT; generate PDF/QR; queue notification
```

5. IMPLEMENTATION

The demonstrator was implemented as a server-rendered web application using Python, Flask-style routing, a relational schema, HTML/CSS forms, and a QR code generation library. SQLite was used for reproducible local tests; a production deployment should use a managed PostgreSQL or MySQL database with documented backup, replication, and monitoring. The prototype modules were authentication, student registry, fee configuration, payment capture, approval, receipt generation, verification, reporting, and audit review. The interface follows a task-oriented design. The finance dashboard shows pending confirmations, daily totals, and shortcuts. Receipt entry uses controlled fee lists and automatic student lookup. The generated document contains the institutional identity, receipt number, payer, fee, amount in figures, payment reference, status, issue date, issuing unit, and QR verification block. Reprints are visibly marked as copies while retaining the original receipt number and issue timestamp.

5.1 Interface and Output Design

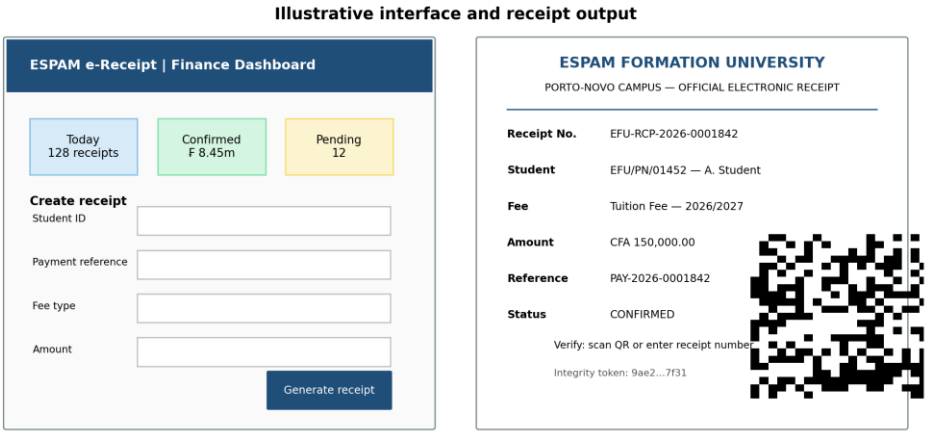

Figure 4. Illustrative finance dashboard and electronic receipt.

The design minimises free-text entry. Student identity is selected from a validated record; fee type is drawn from an administrator-maintained catalogue; amount is either predefined or checked against an approved range; and the payment reference must be unique. Confirmation screens show a complete transaction summary so that the supervisor does not approve from an ambiguous list. Error messages identify the field to correct without exposing server details.

5.2 Reporting and Reconciliation

The Operational Reports Group confirmed payments and issued receipts by day, academic session, fee type, programme, level, cashier, and payment channel. Reconciliation is based on the invariant that the sum of active receipts for a filter equals the sum of confirmed, receipted payments for the same filter. Differences are not silently adjusted; they are listed as exceptions such as confirmed-without-receipt, reversed receipt, duplicate external reference, or notification failure.

Reconciliation difference $D = \Sigma$ *confirmed receipted payments* $- \Sigma$ *active receipt amounts.*

For a correctly reconciled period, $D = 0$. The report also displays count-based checks because equal totals can conceal offsetting errors. Export files include the filter criteria, generation time, requesting user, and, in a production environment, a checksum or digital signature.

5.3 Exception Handling

Table 3. Exception-handling policy.

Exception	System behaviour	Required authority
Incorrect pending payment	Edit before confirmation; preserve change log	Finance officer
Confirmed payment was later found invalid.	Reject reversal request; supervisor decides	Supervisor
The issued receipt contains the wrong source data.	Reverse original; correct payment; issue linked replacement	Supervisor + audit trail
Lost receipt	Reprint the same number and mark “COPY”	Finance officer
Notification failed	Keep a valid receipt; retry email/SMS separately	Automated queue/officer
Compromised user account	Disable account, revoke sessions, review logs	Administrator/security officer

5.4 Deployment Configuration

A production topology should place the application behind a reverse proxy with TLS, host the database on a restricted network segment, store secrets outside source code, and centralise logs. Automated nightly backups are insufficient unless restoration is tested. The recommended minimum is daily encrypted backups, weekly offline or immutable copies, quarterly restore drills, and documented recovery-time and recovery-point objectives. System clocks must be synchronised because timestamps are evidential records.

6. TESTING AND RESULTS

Testing combined requirement-based functional tests, database-integrity tests, security-oriented checks, and controlled performance benchmarks. The data set contained 2,000 synthetic student records and 5,000 synthetic payments. No real student, banking, or institutional financial data were used. The environment was a local single-process prototype; therefore, the results establish the feasibility and correctness of the implementation logic, not production capacity.

6.1 Functional and Integrity Tests

Table 4. Controlled functional and integrity test results.

Test category	Cases	Passed	Result
Authentication and role restrictions	8	8	Authorised pages and actions were correctly restricted.
Student and payment validation	10	10	Required fields, positive amounts, and unique references are enforced.
Confirmation and receipt issuance	9	9	Only confirmed payments produced receipts; the one-to-one rule held.
Search, verification, reprint and reports	9	9	Records were consistently retrieved using all tested keys.
Audit and exception handling	6	6	Relevant actions produced traceable events and controlled outcomes.
Deliberate duplicate receipt insertions	50	50	UNIQUE constraints rejected all attempts.

6.2 Performance Benchmark

The benchmark measured database insertion, indexed receipt lookup, and complete in-memory generation of a verification hash, QR image, and receipt template. Across 5,000 receipt transactions, the mean database insert-and-commit time was 0.278 ms, and the 95th percentile was 0.398 ms, corresponding to approximately 3591 local transactions per second.

Across 2,000 indexed lookups, mean latency was 0.008 ms. For 1,000 complete receipt-plus-QR generations, mean latency was 6.454 ms, and the 95th percentile was 6.763 ms. These low values reflect a local test without network delay, PDF rendering, email delivery, or concurrent users; they must not be misrepresented as evidence of production service levels.

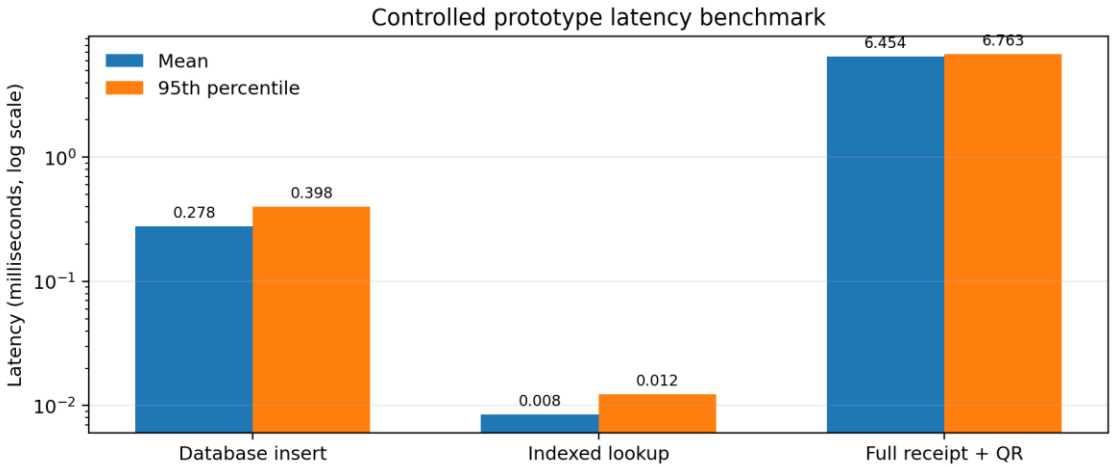

Figure 5. Controlled prototype latency benchmark; vertical scale is logarithmic.

6.3 Quality Assessment

Against the ISO/IEC 25010 perspective [6], the prototype demonstrated functional suitability through passed use cases, performance efficiency through low local latency, security through layered controls and enforced uniqueness, usability through task-focused forms, maintainability through modular separation, and reliability through atomic transactions. Compatibility, accessibility, high availability, disaster recovery, and large-scale concurrency remain to be validated in a production-like environment.

7. DISCUSSION

The principal result is not the speed of generating a PDF; that is trivial. The important result is the preservation of accounting meaning across the complete transaction. A receipt becomes trustworthy when the system can demonstrate who captured the payment, who confirmed it, which payment reference supported it, whether another receipt already existed, what data were printed, and whether later actions changed its status. The proposed design makes these questions answerable from structured records rather than memory or paper files.

Database-enforced uniqueness is especially significant. Interface checks alone fail under concurrent requests or direct database access. The successful rejection of all deliberate duplicate insertions confirms that the one-payment–one-receipt rule is encoded at the data

layer. Similarly, QR verification improves convenience but only when paired with a protected server-side record and a carefully limited public response. Printing the full hash or embedding sensitive student information in the QR code would be poor practice.

7.1 Expected Institutional Benefits

- Faster receipt retrieval for registration, transcript, clearance, and audit enquiries.
- Reduced disputes arising from illegible handwriting, missing copies, or duplicated numbers.
- Improved daily reconciliation through consistent fee categories and payment references.
- Greater accountability because approval, issuance, reprint, reversal, and export actions are attributable.
- Lower paper dependence while preserving printable output for users who require it.
- A foundation for future integration with student portals, bank APIs, mobile money, and accounting software.

7.2 Risks and Mitigations

Table 5. Principal implementation risks and mitigations.

Risk	Consequence	Mitigation
Poor master data	Receipts contain incorrect identity or fee details	Clean student and fee records; validate ownership and change authority.
Shared staff accounts	Actions cannot be attributed	Individual accounts, MFA for privileged users, periodic access review.
Weak deployment security	Credential theft or data exposure	TLS, hardened server, patching, secrets management, penetration testing.
Unreliable connectivity	Interrupted issuance during peak periods	Resilient network, clear retry logic, transaction idempotency, service monitoring.
Backup failure	Loss of financial evidence	Encrypted backups, offline copy, restore drills, retention policy.
Resistance to change	Parallel unofficial processes persist	Training, phased rollout, documented SOPs, management enforcement.

7.3 Limitations

This study has four limitations. First, the institution's current bursary workflow was not independently audited; requirements must therefore be validated with actual finance, registry, audit, IT, and student representatives. Second, all records were synthetic. Third, the benchmark was local and single-node, without concurrent browser sessions, production PDF

rendering, payment-gateway delay, or network variability. Fourth, legal compliance, retention periods, privacy notices, and cross-border hosting implications require professional review in accordance with applicable Beninese law and university policy.

7.4 Production Adoption Roadmap

A defensible rollout should proceed through: requirements validation; data cleaning and migration rehearsal; security architecture review; pilot deployment in one fee category; user acceptance testing; parallel reconciliation against the existing process; penetration testing; backup and restore exercise; staff training; controlled cutover; and post-implementation audit. Direct institution-wide launch without these gates would be reckless.

7.5 Governance, Change Management and Sustainability

The platform requires an owner, not merely a developer. The finance director should own the receipt policy; the IT unit should own availability, patching, backups, and access administration; internal audit should review exception reports and privileged actions; and the registry should govern authoritative student identity. A change-control committee should approve new fee types, integrations, report definitions, and data-retention rules. Without explicit ownership, configuration changes will quietly erode reconciliation and security. Adoption should be measured using operational indicators rather than anecdotes: percentage of confirmed payments receipted within five minutes, number of unreconciled items, duplicate-reference rejection rate, reprint and reversal rate, failed verification attempts, system availability, backup-restore success, help desk volume, and median task completion time. Targets should be agreed before the pilot launch and reviewed monthly. Sustainability also requires documented source code, database migrations, deployment scripts, administrator manuals, user procedures, and a support arrangement that does not rely on a single graduating student.

8. CONCLUSION AND RECOMMENDATIONS

This paper presents a prototype electronic receipt system designed and implemented for the ESPAM Formation University case-study environment in Porto-Novo. The design replaces isolated receipt production with an auditable transaction model connecting students, payments, approvals, receipts, verification values, users, and logs. Unique constraints and atomic transactions prevent duplicate or unsupported issuance; role separation improves accountability; QR-based verification assists students and auditors; and structured reports support reconciliation. Controlled testing with synthetic records showed correct enforcement of the defined business rules and low prototype latency. These results are encouraging but

deliberately limited. They do not substitute for live stakeholder validation, production concurrency testing, legal assessment, or independent security testing. The recommended next step is a supervised pilot integrated with the university's authoritative student and payment data, followed by user acceptance and audit reconciliation before general deployment.

The following recommendations are decisive: (1) retain separation between payment capture and confirmation; (2) enforce uniqueness and referential integrity in the database, not merely in the interface; (3) use HMAC or digital signatures with managed keys for production verification; (4) require MFA for administrators and supervisors; (5) establish reversal rather than silent editing; (6) test backup restoration; (7) monitor anomalous reprints, reversals, and failed logins; and (8) review security, privacy, retention, and accessibility annually. A receipt system that cannot withstand an audit is not an electronic control—it is only a faster printer.

REFERENCES

1. ESPAM Formation University, "About ESPAM Formation University" and "Porto-Novo Campus," official institutional website, accessed 14 July 2026.
2. K. T. Wadsworth, M. T. Guido, J. F. Griffin, and A. Mandil, "An innovation in paper receipts: The electronic receipt management system," 2010 IEEE Systems and Information Engineering Design Symposium, pp. 88–93, 2010, doi: 10.1109/SIEDS.2010.5469674.
3. O. Augustine, "Development of a secure web application for digital payment and receipt management," FNAS Journal of Computing and Applications, 2024.
4. I. Somerville, Software Engineering, 10th ed. Boston, MA: Pearson, 2016.
5. R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 9th ed. New York: McGraw-Hill, 2020.
6. ISO/IEC 25010:2023, Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—Product Quality Model, International Organisation for Standardisation, 2023.
7. OWASP Foundation, Application Security Verification Standard (ASVS), Version 5.0, 2025.
8. OWASP Foundation, "Session Management Cheat Sheet," OWASP Cheat Sheet Series, accessed 14 July 2026.
9. D. Temoshok et al., Digital Identity Guidelines: Authentication and Authenticator Management, NIST SP 800-63B-4, July 2025.

10. A. Biryukov, D. Dinu, D. Khovratovich, and S. Josefsson, “Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications,” RFC 9106, Sept. 2021, doi: 10.17487/RFC9106.
11. ISO/IEC 18004:2024, Information Technology—Automatic Identification and Data Capture Techniques—QR Code Bar Code Symbology Specification, ISO, 2024.
12. R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed. Pearson, 2016.
13. W. H. DeLone and E. R. McLean, “The DeLone and McLean model of information systems success: A ten-year update,” Journal of Management Information Systems, vol. 19, no. 4, pp. 9–30, 2003.
14. J. Nielsen, Usability Engineering. San Francisco, CA: Morgan Kaufmann, 1994.
15. ISO/IEC 27001:2022, Information Security, Cybersecurity and Privacy Protection—Information Security Management Systems—Requirements, ISO, 2022.