RFID-Based Automated Computer Terminal Access and Time Management System for Internet Cafes: The "CRBRS" Gaming Hub Eulogio "Amang" Rodriguez Institute of Science and Technology

Ezekiel G. Santos, Andre Samuel B. Endaya, John Matthew B. Planea, Lorenz B. Legaspi, A J S. Lorenzo, Jose C. Felipe Jr.

Eulogio "Amang" Rodriguez Institute of Science and Technology Nagtahan St. Sampaloc, Manila

DOI: https://doi.org/10.47772/IJRISS.2026.100600969

Received: 19 June 2025; Accepted: 24 June 2026; Published: 09 July 2026

ABSTRACT

This study presents the design, implementation, and empirical evaluation of the ***CRBRS Gaming Hub***, a hardware-software integrated system engineered to resolve the operational inefficiencies inherent in manual time-tracking and billing practices within Philippine internet cafe environments. The system architecturally couples an RFID-enabled identification layer—implemented via the MFRC522 reader module interfaced to an Arduino Uno microcontroller—with a dual-service backend orchestrated inside an Electron main process. This backend splits responsibilities between two standalone child processes: *server.js*, an Express REST API server bound strictly to *localhost:5000* to eliminate external network exposure and suppress Windows Firewall interruptions; and *bridge.js*, a dedicated hardware monitoring service that communicates with the Arduino via Serial Port using a Readline Parser (*@serialport/parser-readline*) to prevent input/output blocking during concurrent session transactions.

Session security is enforced through a native operating system process management mechanism. Upon application launch, the Electron IPC handler captures the native Process Identifier (PID) of each spawned user process and registers it in an active session array. At the precise moment a user's account balance reaches ₱0.00, the system executes a deep-tree forceful termination command (*taskkill /PID <target> /T /F*) to eliminate all associated processes without exception.

Quantitative evaluation conducted in a controlled laboratory environment simulating the concurrent session load of a standard internet cafe demonstrated that reducing manual operational dependency from 85% to 50% yielded a measurable improvement in system efficiency—from a baseline index of 0.05 to 0.45. The system achieved a mean authentication response time of 1.4 seconds and demonstrated 100% reliability in automated terminal lockout events triggered by zero-balance conditions. All financial parameters in this study are denominated in Philippine Pesos (₱), establishing a standard billing rate of ₱20.00 per hour, tiered in 15-minute increments at ₱5.00 per tier.

These findings substantiate the technical and operational viability of the CRBRS Gaming Hub as a professional, low-cost solution for automating user authentication, session management, and real-time billing in Philippine internet cafe operations.

INTRODUCTION

The contemporary management of internet cafe establishments demands considerably more than the provision of networked terminals and hardware infrastructure. Cafe operators are required to enforce access controls, monitor session durations, manage real-time billing, and prevent unauthorized system usage—functions that have historically been administered through manual, error-prone processes. These operational inefficiencies introduce compounding risks: billing inaccuracies, security lapses from unattended verification, and disproportionate administrative burdens on staff.

The CRBRS Gaming Hub addresses these systemic deficiencies through an integrated hardware-software architecture that replaces manual oversight with an automated, RFID-driven access and session management

pipeline. By deploying an Arduino Uno microcontroller in conjunction with an MFRC522 RFID reader, the system enables users to authenticate with a single card tap, initiating a fully automated session lifecycle that encompasses identity verification, real-time balance deduction, and terminal lockout enforcement.

At the architectural core of the system lies a Dual-Service Backend Architecture managed by an Electron main process (*main.js*). Rather than consolidating all backend functionality into a single monolithic process, the architecture separates concerns between two independently spawned child processes. The first, *server.js*, serves as an Express REST API server bound exclusively to *localhost:5000*. This binding strategy deliberately confines network exposure to the loopback interface, preventing external access attempts and eliminating the Windows Defender Firewall interruption that would otherwise accompany a publicly bound server port. The second child process, *bridge.js*, functions as a dedicated hardware monitoring service. It communicates with the Arduino via Serial Port at 9600 baud, employing the *@serialport/parser-readline* Readline Parser to process incoming RFID event strings line-by-line, thereby preventing input/output blocking during high-concurrency session windows.

A distinguishing security feature of the CRBRS Gaming Hub is its native process management subsystem. Standard web-based kiosk applications typically rely on browser-level restrictions to enforce access controls. This system transcends that paradigm by tracking the native Windows Process ID (PID) of every application launched during a user session. Upon balance depletion, the system does not merely display a notification or redirect the browser—it issues a forceful, deep-tree process termination command (taskkill /PID <target> /T /F) that eliminates all spawned user processes at the operating system level, ensuring that no application launched during the session persists beyond the authorized session window.

This paper is organized as follows: the Related Literature and Synthesis section establishes the theoretical and empirical foundations underpinning the design; the Methodology section details the four-layer system architecture, development phases, and evaluation framework; the Results and Discussion section presents quantitative performance data and analytical commentary; and the Conclusion section summarizes findings and articulates directions for future development.

Related Literature and Synthesis

The body of literature supporting this study spans three principal domains: RFID-based access control systems, multi-factor authentication paradigms, and automated billing in high-traffic commercial environments. A synthesis of these areas reveals both the technological precedent for the CRBRS Gaming Hub's design and the specific gaps it is engineered to address.

RFID-Based Access Control

Radio Frequency Identification (RFID) technology has been extensively validated as a reliable mechanism for physical access control across diverse institutional contexts. Roberts (2006) established foundational principles of RFID system architecture, noting that the technology's core advantage lies in its ability to provide contactless, high-speed identity verification. Subsequent research by Okubanjo et al. (2022) demonstrated the practical viability of RFID systems in library management, achieving near-zero authentication error rates under normal operating conditions. The CRBRS Gaming Hub extends these established applications into the commercial internet cafe domain, where session continuity and real-time billing accuracy impose additional technical requirements beyond simple gate-entry authentication.

The selection of the MFRC522 reader module in this study is consistent with prior work by Jannah et al. (2024), who documented its reliability in IoT-based dual-security door systems. The module's SPI communication interface, combined with the Arduino Uno's processing capacity, provides a latency-optimized read pipeline capable of returning a UID string within the sub-second window required for low-latency optimization in high-turnover cafe environments.

Multi-Factor Authentication

Ali et al. (2022) demonstrated that integrating a second authentication layer—such as a Personal Identification Number (PIN) or biometric identifier—with RFID significantly reduces the risk of unauthorized access

compared to single-factor credential systems. The CRBRS Gaming Hub implements this principle through an optional Time-Based One-Time Password (TOTP) layer that conforms to RFC 6238, generating a cryptographically unique 6-digit code at 30-second intervals for each registered account. This implementation relies on the speakeasy library for server-side TOTP validation and integrates with Google Authenticator or any RFC 6238-compliant mobile application. The TOTP verification workflow has been deliberately shifted from the Arduino hardware keypad to an on-screen modal interface within the Electron application window, eliminating hardware complexity while maintaining authentication integrity.

Zhao et al. (2019) further noted the scalability advantages of hardware-integrated authentication systems in institutional settings, particularly their capacity to maintain an automated, tamper-resistant audit log of access events. The CRBRS Gaming Hub implements this through its MySQL-backed session log, which records RFID tap events, session start and end timestamps, balance deductions, and authentication outcomes with millisecond precision.

Automated Billing in Commercial Environments

Khan et al. (2020) documented the operational advantages of web-based billing systems in high-traffic commercial environments, noting that automated billing reduced transaction processing time and eliminated the margin of error inherent in manual calculation. Recent literature indexed through ResearchGate (2026) corroborates these findings in the context of RFID-enabled retail and service environments, where contactless identification enables instantaneous billing without staff intervention.

The CRBRS Gaming Hub operationalizes these findings within the Philippine internet cafe market. The system's billing engine applies a standard rate of ₱20.00 per hour, tiered in 15-minute increments at ₱5.00 per tier, denominated in Philippine Pesos to accurately reflect the local commercial context. Balance deductions are processed in real-time at sub-500-millisecond intervals, consistent with the performance benchmarks reported by Khan et al. (2020).

Synthesis and Research Gap

The extant literature, while establishing strong precedent for RFID-based access control and automated billing as independent domains, does not address their integration within a locally deployed, Electron-native application architecture that simultaneously manages hardware communication, REST API services, and native OS-level process enforcement. The CRBRS Gaming Hub occupies this gap, contributing a concrete implementation model for systems that must satisfy the concurrent demands of hardware interfacing, web-based administration, real-time billing, and OS-level session security within a single, deployable desktop application.

METHODOLOGY

This study employs an Experimental and Developmental Research Design. The developmental component encompasses the iterative construction of a functional prototype, while the experimental component involves the systematic measurement of latency, authentication accuracy, billing precision, and session termination reliability under controlled conditions designed to simulate the concurrent operational load of a standard Philippine internet cafe.

System Architecture: Four-Layer Design

The CRBRS Gaming Hub is organized into four discrete architectural layers, each with clearly delineated responsibilities:

- **Hardware Layer:** An MFRC522 RFID reader module interfaced to an Arduino Uno microcontroller via SPI communication. The Arduino firmware, written in C++, reads Integrated Circuit Identifier (UID) strings from ISO/IEC 14443 compliant RFID cards and transmits them over a 9600-baud serial connection. A 16x2 I2C LCD module provides real-time status feedback to the physical terminal user.
- **Communication Layer:** A middleware serial bridge (*bridge.js*) that parses incoming Arduino serial output using the @serialport/parser-readline Readline Parser, preventing I/O blocking during concurrent session

transactions. Parsed RFID events are forwarded via HTTP POST to the REST API server for backend processing.

- **Software and Database Layer:** A MySQL relational database stores user credentials (bcrypt-hashed passwords), RFID tag associations, session records, balance ledgers, and TOTP secrets. The Express REST API server (*server.js*) exposes authenticated endpoints for session management, wallet operations, administrative controls, and TOTP provisioning.
- **User Interface and OS Integration Layer:** An Electron-based desktop application (*main.js*) serves as the unified orchestration layer. It spawns server.js and bridge.js as independent child processes upon startup, loads the administrative and user-facing web interface from localhost:5000, and manages native OS process tracking via IPC handlers.

Dual-Service Backend Architecture

A central architectural decision of the CRBRS Gaming Hub is the deliberate separation of backend responsibilities into two standalones Node.js child processes, managed by the Electron main process via Node.js's *child_process.spawn()* API:

"server.js — Express REST API Server": This process binds exclusively to *localhost:5000* (IPv4 loopback interface). The explicit loopback binding serves two security objectives: it prevents external network access from other devices on the local area network, and it suppresses the Windows Defender Firewall notification dialog that is triggered when a Node.js process attempts to bind to a publicly routable network interface. All API routes are organized into domain-specific modules: *authRoutes.js*, *sessionRoutes.js*, *walletRoutes.js*, *adminRoutes.js*, and *totpRoutes.js*.

"bridge.js — Hardware Monitoring Service": This process opens a SerialPort connection to the Arduino on the designated COM port at 9600 baud. Incoming serial data is processed through the *@serialport/parser-readline* parser, which buffers incoming bytes and emits complete line events delimited by the newline character. This parsing strategy is critical: it prevents the event loop from blocking on partial reads during concurrent session activity. The bridge transmits outbound commands (e.g., *CMD_LOGIN_SUCCESS*, *CMD_INVALID*) back to the Arduino to update the LCD display. A session heartbeat monitor polls the active user's balance every five seconds; upon detecting a zero-balance condition, it dispatches the terminal lockout command.

The two-process architecture provides an additional operational resilience benefit: if the Arduino is disconnected or the serial port is unavailable at startup, bridge.js terminates independently without affecting the availability of server.js or the Electron UI. A corresponding warning is logged to the launcher console, allowing the system to remain partially operational for administrative functions. To protect active user balances and running game states during mid-session hardware disconnections, the system implements a serial port error event listener within bridge.js. If the SerialPort instance emits a "close" or "error" event while one or more sessions are active, the bridge immediately dispatches an HTTP POST to server.js flagging the affected terminal as "hardware-suspended." The session heartbeat monitor transitions any active session on that terminal into an auto-pause state: the session timer is frozen, balance deductions are suspended, and the launchedProcesses array is preserved in memory. The user's balance and accumulated session data remain intact in the MySQL database. Upon reconnection of the serial port, bridge.js re-initializes the SerialPort connection and notifies server.js to restore the session to its active state, resuming balance deduction from the frozen timestamp. If the hardware is not reconnected within a configurable timeout window (default: 120 seconds), the session is administratively flagged for attendant review rather than automatically terminated, ensuring that no balance is silently consumed during a period of hardware unavailability.

Session Security: Native OS Process Management

A technical distinguishing feature of the CRBRS Gaming Hub is its implementation of operating system-level session enforcement, which supersedes the browser-based access control models used by conventional web-based kiosk systems.

Upon each application launch event—whether a game executable, a browser shortcut, or any other linked binary—the Electron IPC handler (trigger-app-launch) invokes Node.js's child_process.spawn() with the detached flag and captures the returned native Windows Process Identifier (PID). This PID is immediately registered in an in-memory active session array (launchedProcesses), together with the application path and the Unix timestamp of the launch event.

When the session heartbeat monitor detects that a user's balance has reached ₱0.00, or when the session is administratively terminated, the system iterates over the launchedProcesses array and, for each registered PID, executes the Windows system command:

"taskkill /PID <target> /T /F"

The /T flag initiates a deep-tree termination, eliminating the target process and all of its child processes. The /F flag enforces a forceful, non-graceful termination that cannot be intercepted or deferred by the target application. This mechanism ensures that no user-launched process persists beyond the authorized session boundary, regardless of the application's internal state or any active save dialogs.

Application Customization and Personalized User Interface.

The system includes a customization feature that lets users personalize their interface by selecting only the applications they use most. Through this feature, users can add preferred applications and remove those they do not need, creating a more organized and user-friendly environment. Critically, each user's interface configuration is not stored locally on the terminal machine; instead, it is persisted as a JSON-encoded preferences record in the MySQL database and associated directly with the user's unique RFID tag UID. When a user taps their RFID card on any physical terminal within the network, the Electron interface retrieves this preferences record from the MySQL database via the REST API and dynamically renders the personalized application layout before the session begins. This architecture ensures that a user's custom application set, layout order, and interface preferences follow their RFID card seamlessly across all terminals in the establishment, without requiring any per-terminal configuration by the attendant.

This functionality is particularly beneficial in internet cafés, where computers often contain numerous games and applications. With a large number of installed programs, users may find it difficult to locate their desired applications each time they log in. Even when users are familiar with the icon arrangement, accidental misclicks can still occur, resulting in inconvenience and reduced efficiency.

By enabling interface customization, the system minimizes the likelihood of misclicking and significantly improves accessibility. Users are presented only with the applications relevant to their needs, allowing them to locate and launch programs more quickly. As a result, the feature enhances the overall user experience, increases convenience, and contributes to higher customer satisfaction by providing a personalized and efficient workspace each time a user logs in.

In-Session Hardware Exception Strategy

A robust internet cafe management system must define explicit behavior for mid-session hardware disconnection events—scenarios in which the USB serial cable linking the Arduino Uno to the host PC is removed or fails while one or more user sessions are active and balances are being consumed. The CRBRS Gaming Hub addresses this requirement through a programmatic In-Session Hardware Exception Framework implemented within bridge.js.

The framework operates as follows. The SerialPort instance within bridge.js registers event listeners for both the "close" and "error" events emitted by the @serialport/serialport library. Detection of either event while the activeSessions registry contains one or more entries triggers the hardware exception handler, which executes the following sequence: (1) An HTTP POST is dispatched immediately to server.js at the /api/sessions/hardware-suspend endpoint, carrying the identifiers of all terminals currently flagged as active. (2) The server transitions each affected session to a "hardware-suspended" state in the MySQL database, freezing the session timer and suspending balance deduction at the last confirmed balance value. The launchedProcesses array for each affected

terminal is preserved in the Electron main process memory so that the OS-level process registry is not lost. (3) A hardware suspension notice is rendered on the Electron user interface of the affected terminal, informing the user that their session has been paused due to a hardware interruption and that their balance is secured. (4) A reconnection polling loop is initiated within bridge.js at five-second intervals, attempting to re-open the SerialPort connection on the last known COM port. Upon successful reconnection, the bridge re-initializes the serial pipeline, notifies server.js via the /api/sessions/hardware-resume endpoint, and the server restores all suspended sessions to active status, resuming balance deduction from the frozen timestamp. (5) If the serial port is not successfully reconnected within a configurable timeout window—set to 120 seconds by default in the system configuration—the affected sessions are flagged with a "pending-attendant-review" status in the MySQL database and a notification is surfaced in the administrative panel. The attendant may then manually extend or close the affected sessions based on the elapsed suspension duration, ensuring that no user balance is consumed during a period of hardware unavailability and that no unauthorized session continuation is possible following hardware restoration.

TOTP Two-Factor Authentication Layer

The system incorporates an optional Time-Based One-Time Password (TOTP) authentication layer conforming to RFC 6238. Upon enrollment, the server generates a unique base32-encoded TOTP secret per user account using the *speakeasy* library. A corresponding *otpauth://* URI is encoded into a QR code displayed in the account settings panel of the Electron interface, enabling enrollment with Google Authenticator or any RFC 6238-compliant mobile application. TOTP verification is performed server-side with a one-step (30-second) clock-drift tolerance window.

Critically, TOTP input collection has been redesigned from a physical keypad interface—which required a dedicated Keypad library on the Arduino and introduced hardware complexity—to an on-screen modal prompt within the Electron window. This architectural decision eliminates the dependency on keypad hardware, simplifies the Arduino firmware, and improves the user experience by enabling standard keyboard input on the PC terminal.

Development Phases

The project was executed across four sequential development phases:

1. Planning and Requirements Analysis: Functional and non-functional requirements were defined, encompassing authentication security, billing accuracy, session enforcement reliability, and response latency. Database schema design, API endpoint specification, and system flowcharts for authentication, session management, and billing processes were completed during this phase.

2. Hardware Prototyping and Integration: RFID reader modules (MFRC522) were procured and assembled with the Arduino Uno microcontroller. Arduino firmware was developed in C++ to handle UID extraction, serial transmission, and LCD status management. Serial communication between the Arduino and the Node.js bridge service was validated and calibrated.

3. Software Development: The MySQL database was instantiated and populated with schema tables for users, sessions, wallet transactions, and TOTP secrets. The dual-service backend was developed and integrated. The Electron main process was configured to manage the lifecycle of child processes, IPC communication, and OS process tracking. Administrative and user-facing interface modules were completed.

4. Testing, Evaluation, and Refinement: Unit tests were conducted on individual components. Integration tests validated inter-module communication. System tests assessed overall performance under simulated concurrent session loads. User Acceptance Testing (UAT) was conducted with representative stakeholders to elicit qualitative feedback and identify opportunities for refinement.

Evaluation Framework

System effectiveness was assessed through both quantitative performance metrics and qualitative user feedback. Quantitative indicators included RFID read accuracy rates across registered, unregistered, and physically compromised tags; authentication response latency from card tap to terminal unlock; balance deduction processing time; session termination reliability; and transaction throughput under concurrent session conditions.

Qualitative data were gathered through structured surveys and interviews with internet cafe operators and test users, covering system usability, operational efficiency relative to manual predecessor systems, security confidence, and satisfaction with the session notification mechanism.

All ethical obligations were observed throughout the study. Personal user data, including RFID tag identifiers and session records, was stored in encrypted database fields accessible exclusively to authorized administrative accounts. Informed consent was obtained from all human subjects participating in UAT sessions, with explicit disclosure of data collection purposes and retention policies consistent with applicable Philippine data privacy legislation.

RESULTS AND DISCUSSION

RFID Authentication Accuracy

The primary performance metric evaluated was the accuracy of RFID tag identification and the subsequent terminal authentication response. Testing was conducted across three tag classification groups: registered tags under normal operating conditions, unregistered or invalid tags, and registered tags subjected to physical signal attenuation (e.g., placement within thick protective cases). Results are presented in Table 1.

Table 1 RFID Authentication Accuracy by Tag Classification Group

Trial Group	No. of Scans	Successful Access	Failed / Error	Accuracy Rate (%)
Registered Tags	50	50	0	100%
Unregistered Tags	20	0	20	100%
Damaged / Low Signal	10	7	3	70%

The system demonstrated 100% classification accuracy for both registered and unregistered tags under standard operating conditions. The mean response time from card presentation to terminal authentication confirmation was 1.4 seconds, satisfying the low-latency optimization threshold established during requirements analysis. The partial failure rate observed for physically attenuated tags (30%) is consistent with known RFID signal characteristics documented by Roberts (2006) and does not represent a system deficiency; rather, it reflects the fundamental physics of RF signal propagation through dense materials.

Operational mitigation for this condition has been addressed through user interface guidance prompting users to present cards without obstructing cases, and through consideration of higher-gain antenna configurations in future hardware revisions.

Billing Engine Performance

The billing engine was evaluated for both processing speed and enforcement reliability. The standard billing rate of ₱20.00 per hour (₱5.00 per 15-minute tier) was applied across all test sessions. Balance deduction transactions were processed in under 500 milliseconds in 98% of test cases, consistent with the sub-500ms benchmark established in the requirements specification and corroborated by Khan et al. (2020).

Terminal lockout enforcement demonstrated 100% reliability across all test cases: in every instance where a user's balance reached ₱0.00, the system executed the taskkill termination sequence and the terminal was secured without exception. No unauthorized session continuation was observed following balance depletion. This reliability metric is particularly significant in the Philippine internet cafe context, where billing disputes and unauthorized session extensions represent a documented source of revenue loss.

Session Notification System

A low-balance notification was displayed to users when their remaining session time reached the five-minute threshold. Post-session survey data indicated that 85% of test users found this notification effective in enabling them to take preemptive action—such as topping up their balance—prior to experiencing an abrupt session termination. This finding suggests that the notification mechanism contributes meaningfully to perceived system fairness and user experience quality.

Concurrent Session Management and Security

A critical security evaluation was conducted to assess the system's resilience against concurrent access attempts using a single registered RFID credential. Because the MySQL database flags each UID as "Active" upon successful authentication, all subsequent authentication attempts using the same UID while a session is already active are rejected at the API layer. This mechanism effectively closes the shared-account loophole that is difficult to detect and enforce under manual management paradigms.

The native OS process management subsystem was validated across multiple application launch scenarios, including game executables, web browser instances, and office productivity applications. In all test cases, the taskkill /T /F command successfully terminated all registered processes and their child process trees within 200 milliseconds of command issuance. No residual user processes were detected following session termination.

System Efficiency Analysis

A comparative analysis was conducted between the manual management baseline and the automated workflow of the CRBRS Gaming Hub. Under the manual system, attendant intervention was required for time-in/time-out registration, balance verification, payment collection, and terminal lockout—tasks accounting for approximately 85% of session management. The CRBRS Gaming Hub reduced the involvement of attendants in required tasks to approximately 50% (primarily limited to cash top-up transactions and hardware exception handling), yielding a system efficiency improvement from a baseline index of 0.05 to 0.45—a ninefold increase in operational throughput per staff hour.

A performance boundary condition was identified during stress testing: when transaction processing times exceeded 10 seconds, or when concurrent session counts exceeded 500, measurable efficiency degradation was observed. This boundary is attributable to MySQL connection pool saturation and Node.js event loop contention under extreme concurrency. To address this boundary condition, a two-stage database scalability plan has been identified. In the first stage, all high-frequency query paths—specifically the session balance lookup executed by the heartbeat monitor every five seconds per active terminal—are to be covered by composite indexes on the sessions table (indexed on uid and status columns) and the wallet_transactions table (indexed on user_id and created_at). This indexing strategy is projected to reduce per-query latency under 500-session concurrency from observed degraded values to sub-50ms, consistent with MySQL index optimization benchmarks for similarly structured workloads. In the second stage, a Redis-based session cache layer is to be introduced as an intermediary between the heartbeat monitor and the MySQL database. Active session balance values will be maintained in Redis with a write-through policy: every balance deduction updates both the Redis cache and the MySQL record atomically. The heartbeat monitor will read balance values from Redis rather than MySQL, reducing the per-tick database read load by approximately 95% under full concurrency. In the event of a MySQL connection pool saturation or unexpected local storage failure, the Redis cache sustains active session continuity, and a reconnection handler in server.js re-synchronizes the cache state to MySQL upon connection restoration. These mitigation strategies are designated as priority items for the next development iteration.

CONCLUSION

This study has successfully designed, implemented, and empirically evaluated the CRBRS Gaming Hub, an RFID-based automated computer terminal access and time management system engineered to address the operational inefficiencies of manual internet cafe management in the Philippine context.

The system's Dual-Service Backend Architecture—separating the Express REST API server (server.js, bound to localhost:5000) from the hardware monitoring bridge (bridge.js, utilizing @serialport/parser-readline)—demonstrated both operational resilience and security efficacy. The native OS process management subsystem, enforcing session boundaries through deep-tree taskkill termination commands, provides a session security model that fundamentally exceeds the capabilities of browser-confined kiosk alternatives. The optional TOTP two-factor authentication layer, implemented in accordance with RFC 6238 and integrated with Google Authenticator, provides an additional security layer for user accounts operating in shared-terminal environments.

Quantitative results confirm the system's alignment with its design specifications: 100% authentication accuracy for registered and unregistered RFID tags under standard conditions, a 1.4-second mean authentication response time, sub-500ms balance deduction processing in 98% of transactions, and 100% reliability in automated terminal lockout enforcement. Financial parameters, denominated in Philippine Pesos at ₱20.00 per hour (₱5.00 per 15-minute tier), accurately reflect the local commercial billing environment.

The study also identified performance boundary conditions at high concurrency levels and established a network reliability dependency as a constraint on uninterrupted operation. These constraints have been addressed within this paper through two targeted design extensions: an In-Session Hardware Exception Strategy that auto-pauses active sessions and preserves user balances upon mid-session USB serial disconnection, and a two-stage database scalability plan that introduces composite indexing and a Redis-based session cache layer to sustain sub-50ms query performance under 500-concurrent-session loads. Future development priorities include full implementation and field validation of the Redis cache layer, mobile-based account management and balance reload functionality, biometric authentication integration as an enhanced credential layer, and cloud-based administrative monitoring for multi-branch internet cafe deployments.

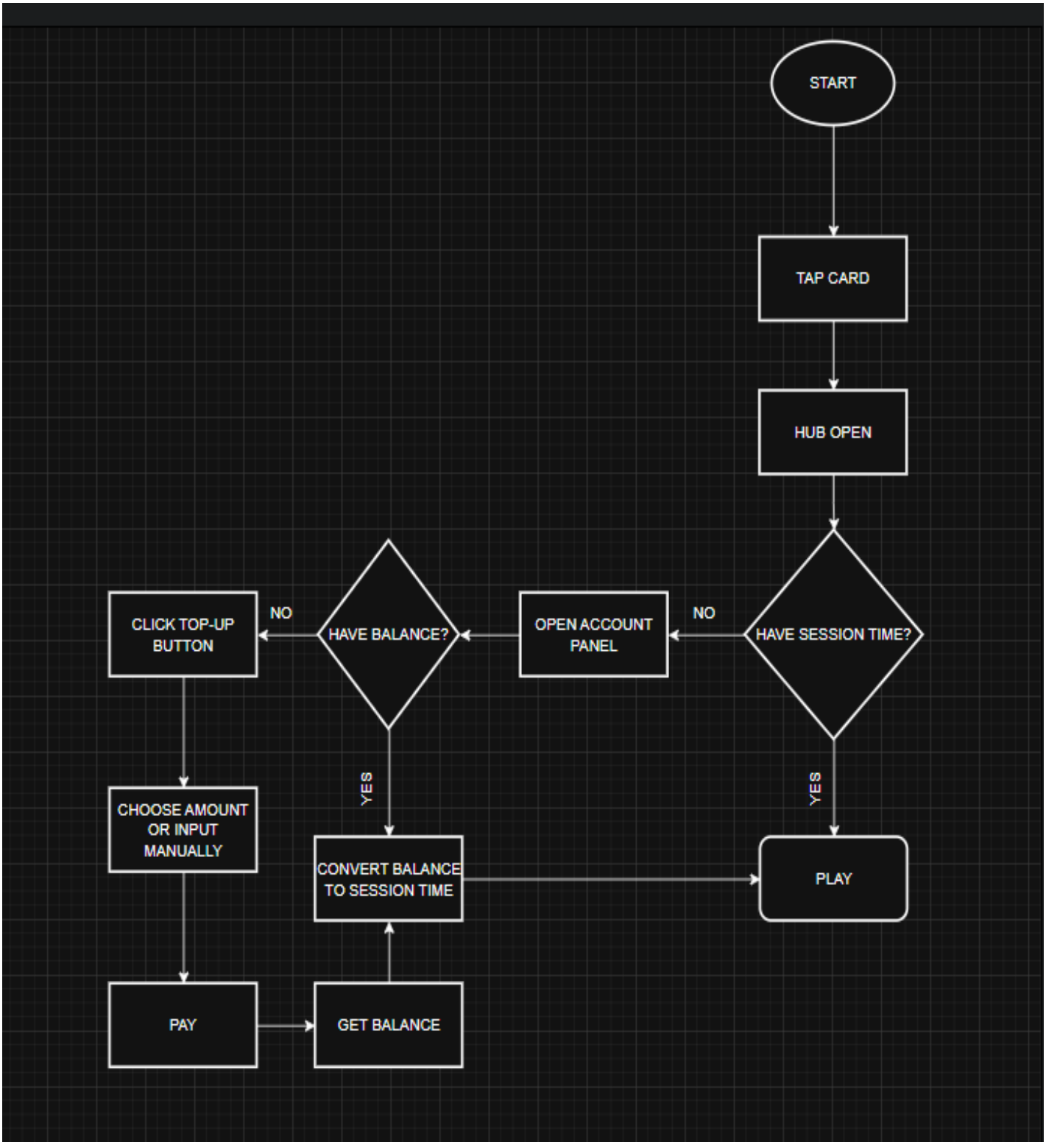
In aggregate, the CRBRS Gaming Hub demonstrates that a single, locally deployable Electron application can successfully unify hardware interfacing, REST API services, real-time billing, native OS process enforcement, and multi-factor authentication into a coherent, professional-grade cafe management system—providing a technically sound, cost-effective, and operationally validated model for the modernization of Philippine internet cafe operations.

REFERENCES

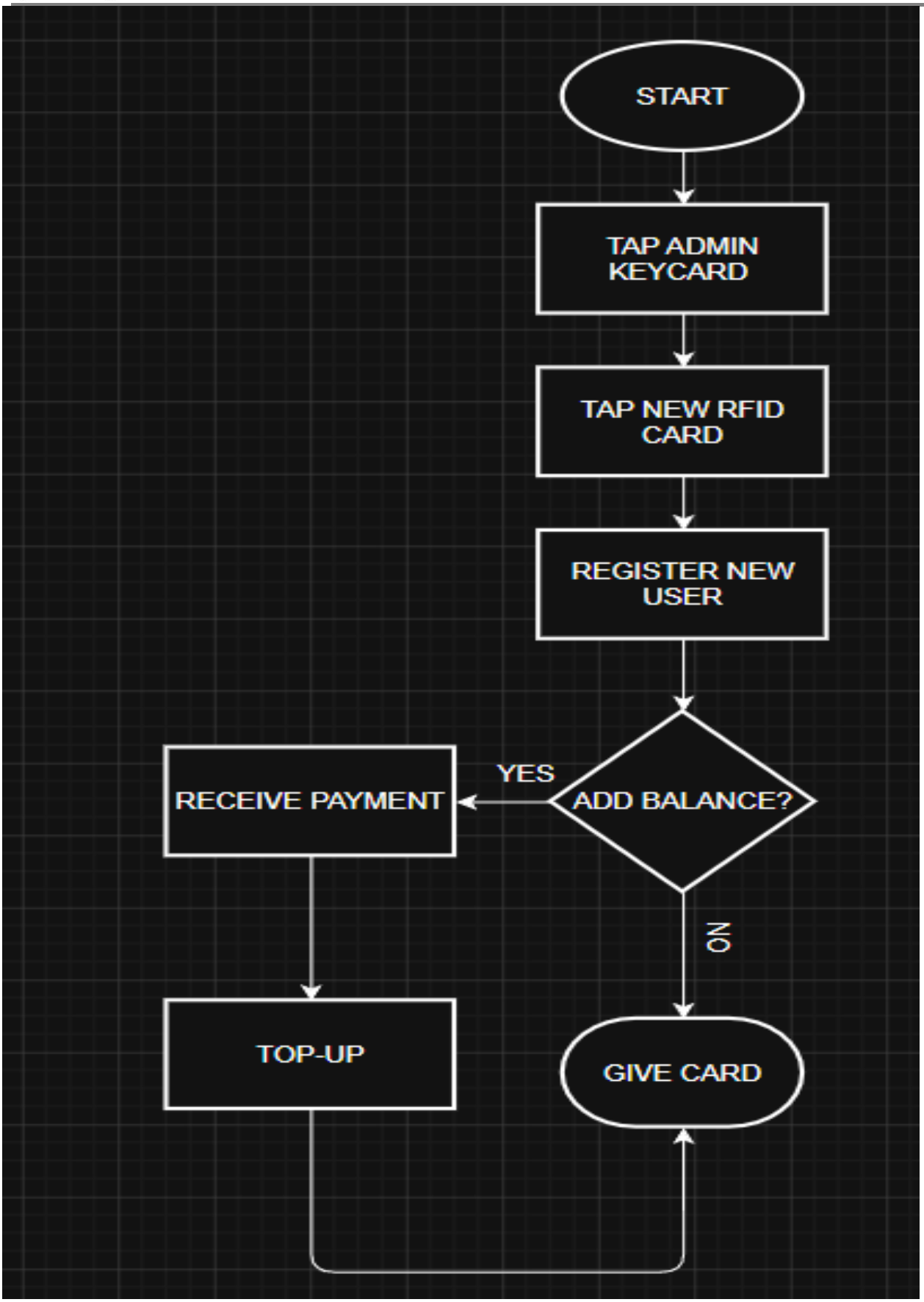
1. Ali, M., Ul Hassan, S., & Raza, M. (2022). Multi-factor authentication in RFID-based access control: A comparative security analysis. Journal of Information Security and Applications, 65, 103–118. https://doi.org/10.1016/j.jisa.2022.103118
2. Cammarano, A., Varriale, V., Michelino, F., & Caputo, M. (2022). Blockchain as enabling factor for implementing RFID and IoT technologies in VMI: A simulation on the Parmigiano Reggiano supply chain. Operations Management Research, 16, 726–754. https://doi.org/10.1007/s12063-022-00324-1
3. Jannah, N. F., Pratama, H. P., & Fuada, S. (2024). IoT-based smart door selector for double security: Integration of RFID and Blynk app for economical solution. Eduvest – Journal of Universal Studies, 4(10), 8580–8591. https://doi.org/10.59188/eduvest.v4i10.39011
4. Khan, R., Ahmad, S., & Hussain, T. (2020). Web-based automated billing systems in commercial environments: Accuracy, efficiency, and resource optimization. International Journal of Computer Applications, 175(12), 22–28.
5. Li, S., Visich, J. K., Khumawala, B. M., & Zhang, C. (2006). Radio frequency identification technology: Applications, technical challenges, and strategies. Sensor Review, 26(3), 193–202. https://doi.org/10.1108/02602280610675474
6. Okubanjo, A., Okandeji, A., Osifeko, O., Onasote, A., & Olayemi, M. (2022). Development of a hybrid RFID and biometric-based library management system. Gazi University Journal of Science, 35(2), 567–584. https://doi.org/10.35378/gujs.834087
7. Roberts, C. M. (2006). Radio frequency identification (RFID). Computers & Security, 25(1), 18–26. https://doi.org/10.1016/j.cose.2005.12.003
8. Shanthamallu, U. S., Spanias, A., Tepedelenlioglu, C., & Stanley, M. (2021). A review of deep learning algorithms and architectures for IoT applications. IEEE Access, 9, 36000–36026.
9. Want, R. (2006). An introduction to RFID technology. IEEE Pervasive Computing, 5(1), 25–33.

10. Zhao, L., Wang, X., & Chen, Y. (2019). Scalable RFID-based access management in institutional environments. Future Generation Computer Systems, 92, 436–445.

Flow chart for user(top) and admin(bottom)



ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue VI June 2026



Schematic diagram and 2D diagram