International Journal of Computer and Information System (IJCIS)
Peer Reviewed – International Journal
Vol : Vol. 07, Issue 02, June 2026
e-ISSN : 2745-9659
https://ijcis.net/index.php/ijcis/index

Design and Implementation of Web-Based Point of Sale and Photo Studio Management System Using MVC Architecture

1st Haniel Pratama, 2nd Andreas Akar, 3rd Alfa Rivales Matal, 4th Abram Pangidoan Tambak, 5th Cahya Evendy, 6th Widiatry
[1,2,3,4,5,6]Teknik Informatika, Universitas Palangka Raya
[1]hanielpratama17@mhs.eng.upr.ac.id, [2]andreasakar@mhs.eng.upr.ac.id, [3]alfarivalesmatal@mhs.eng.upr.ac.id,
[4]abrampngtbk@mhs.eng.upr.ac.id, [5]epen07@mhs.eng.upr.ac.id, [6]widiatry@it.upr.ac.id

***Abstract*—** Photo studios frequently experience operational challenges in integrating cashier transactions, schedule management for photo sessions, and real-time inventory tracking. Manual recording systems often lead to fatal data losses and schedule conflicts. This research aims to design and implement a web-based Point of Sale and studio management information system named Dthree POS to resolve these operational bottlenecks. The system development utilizes the Prototyping method, allowing iterative design improvements from initial sketches to high-fidelity user interfaces. The application is built upon the Model View Controller architecture using the Laravel framework, which effectively separates business logic from data presentation. The results show that the system successfully integrates the management of physical products with limited stock and service products with unlimited availability. Furthermore, the implementation of pessimistic database locking and atomic transactions effectively prevents race conditions during simultaneous checkout processes. The system also accommodates a dynamic pricing hierarchy for different customer types and automatically converts completed studio bookings into final sales data. In conclusion, Dthree POS provides an empirical and highly reliable solution to digitalize and streamline the entire photo studio business workflow.

***Keywords* :** Point of Sale, Photo Studio, MVC Architecture, Laravel, Information System

I. INTRODUCTION

The growth of the photo studio industry necessitates precise and systematic operational integration. Managing daily operations, including cashier transactions, physical inventory management, and photo session scheduling manually, often leads to various administrative obstacles. Manual record-keeping via ledgers is susceptible to data loss and inaccuracies in financial recapitulation. Photo studios also frequently encounter overlapping booking schedules due to a lack of data synchronization among employees [1], [2]Conventional systems fail to provide real-time information when customers make studio reservations concurrently with product purchases or photo printing transactions at the cashier desk. Therefore, a centralized digital system is crucial to maintaining daily operational consistency.

Previous studies predominantly focus on standard retail applications and frequently overlook the complex integration required in service-oriented business models. Several existing web-based management systems successfully digitalize basic inventory tracking[3], [4], [5]. However, these conventional solutions consistently fail to synchronize concurrent physical merchandise sales and dynamic studio scheduling within a single computational platform [6], [7]. Furthermore, prior applications demonstrate significant architectural weaknesses regarding database concurrency control. Multi-user point-of-sale systems processing parallel transactions exhibit significant database vulnerabilities. This critical situation is identified as a *race condition*, an anomaly occurring when multiple users access and manipulate identical stock data within the same millisecond. Such anomalies cause the system to record invalid stock balances [8].

This research bridges this specific operational gap by proposing a highly integrated solution. A *pessimistic locking* approach serves as a highly effective computational solution to resolve this issue. The security mechanism utilizing the *lock for update* instruction secures specific data rows while a cashier processes the pre-payment stage. The system restricts other users from accessing those data rows until the initial transaction is successfully executed [8], [9]. This research focuses on the design and implementation of Dthree POS as a web-based management system for photo studios. The system is specifically developed to handle sales transactions with dynamic pricing hierarchies, studio bookings, and robust database security. Dthree POS applies structured logic separation between physical products with inventory limits and service products with unlimited availability [10], [11]. The implementation of this system aims to provide comprehensive operational recording reliability, protect transactions from duplication risks, and digitize the entire photo studio business workflow comprehensively.

II. RESEARCH METHODS

2.1 System Development Method

This research implements the *Agile* methodology in its software development life cycle. The *Agile* approach provides high flexibility to accommodate specific requirement changes from the photo studio operational management during the system construction process [1], [2]. The development of the Dthree POS system is executed through a series of structured and empirical iterative phases or *sprints*. To ensure methodological clarity and reproducibility, the project development timeline is divided into six distinct *sprints*. The development team allocates a strict two-week duration for each *sprint* cycle to maintain a highly measured and adaptive progress rate.

International Journal of Computer and Information System (IJCIS)
Peer Reviewed – International Journal
Vol : Vol. 07, Issue 02, June 2026
e-ISSN : 2745-9659
https://ijcis.net/index.php/ijcis/index

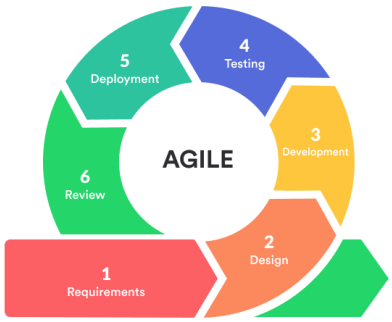

Figure 1. Agile Methodology

The implementation follows six specific *Agile* stages within the development cycle. The first stage, *Requirements*, focuses on a comprehensive system analysis where the development team maps the specific workflow of the studio, encompassing schedule booking processes and dynamic cashier transactions. The second stage, *Design*, translates these requirements into progressive user interfaces utilizing the *Tailwind CSS* framework. The third stage, *Development*, involves direct modular source code construction using the *Laravel* framework [6].

The validation of the system is conducted during the fourth stage, *Testing*. The team applies a rigorous *Black Box Testing* procedure to evaluate the functional integrity of every newly built module before integration. The fifth stage, *Deployment*, integrates the validated modules into the staging environment. Finally, the sixth stage, *Review*, utilizes a direct user evaluation method. The team demonstrates the prototype directly to the primary user entities, namely the top-level management (*admin*) and the cashier staff (*employee*), to gather user feedback and plan functional adjustments for the subsequent iteration [1].

The following table details the comprehensive timeline and specific modular focus of the *Agile sprint* implementation:

Table 1. Agile Sprint Implementation Schedule

Sprint Phase	Module Focus and Execution	Duration
Sprint 1	Requirement gathering, workflow mapping, and database architecture design.	1 Week
Sprint 2	User interface design and authentication security module implementation.	1 Week
Sprint 3	Point of Sale terminal construction and dynamic pricing engine integration.	2 Days
Sprint 4	Integrated studio booking management and automatic transaction generation.	1 Days
Sprint 5	Comprehensive *Black Box Testing* and database concurrency control evaluation.	3 Days
Sprint 6	Final system deployment, user evaluation, and documentation review.	2 Weeks

This structured cyclic process ensures every built module consistently aligns with the operational needs of the studio and completely digitalizes the business workflow into a highly valuable software product.

2.2 System Architecture

This research develops the Dthree POS application utilizing a monolithic Model View Controller architecture through the Laravel 12.x framework. The MVC pattern systematically separates the application core components: the database management (*Model*), the user interface presentation (*View*), and the business logic intermediary (*Controller*). This structural separation enhances code maintainability and scalability for future photo studio operational needs [12].

The frontend presentation utilizes the Blade Template Engine integrated with the Tailwind CSS framework and Vanilla JavaScript to ensure a responsive and intuitive user experience. The asset compilation process is optimized using Vite, which significantly accelerates the hot module replacement mechanism during the development phase.

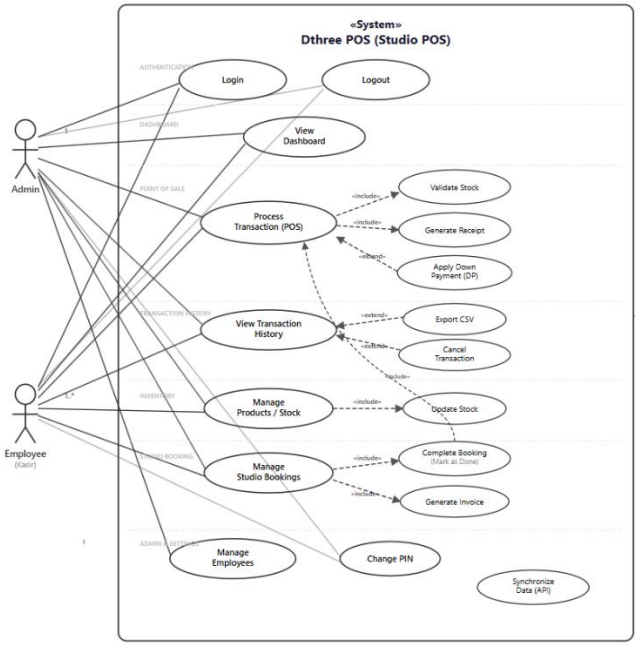

Figure 2. Use Case Diagram

System security and user authorization are governed by a strict Role-Based Access Control mechanism. As illustrated in Figure 2, the system defines two primary actors: *Admin* and *Employee*. The *Admin* possesses absolute authorization to manage employee data, alter product capital prices, and void any transaction. Conversely, the *Employee* operates with restricted access, strictly limited to personal dashboard monitoring, standard cashier operations, and studio booking management. To fortify system security, the default authentication behavior is modified. The system validates users using a combination of a username and a mathematically hashed 4 to 6-digit Personal Identification Number instead of standard passwords, effectively mitigating unauthorized access to the cashier terminal [13].

International Journal of Computer and Information System (IJCIS)
Peer Reviewed – International Journal
Vol : Vol. 07, Issue 02, June 2026
e-ISSN : 2745-9659
https://ijcis.net/index.php/ijcis/index

2.3 Database Design

The data persistence layer of Dthree POS is engineered using a *MySQL* relational *database*. The system interaction with the *database* is managed through the *Eloquent Object-Relational Mapping* (ORM) provided by the *Laravel framework*. This approach ensures that all *database* queries are executed safely using prepared statements, which effectively mitigates the risk of *SQL Injection* attacks while maintaining a clean and readable code structure [6].

III. RESULT AND ANALYSIS

3.1 User Interface Implementation

The user interface of the Dthree POS system is meticulously designed to provide an optimal, modern, and efficient user experience for fast-paced photo studio operations. The interface design prioritizes usability and intuitive navigation, aligning with standard system design principles [2]. The presentation layer utilizes the *Tailwind CSS* framework to ensure a responsive layout, allowing all features to be seamlessly accessed across various device resolutions.

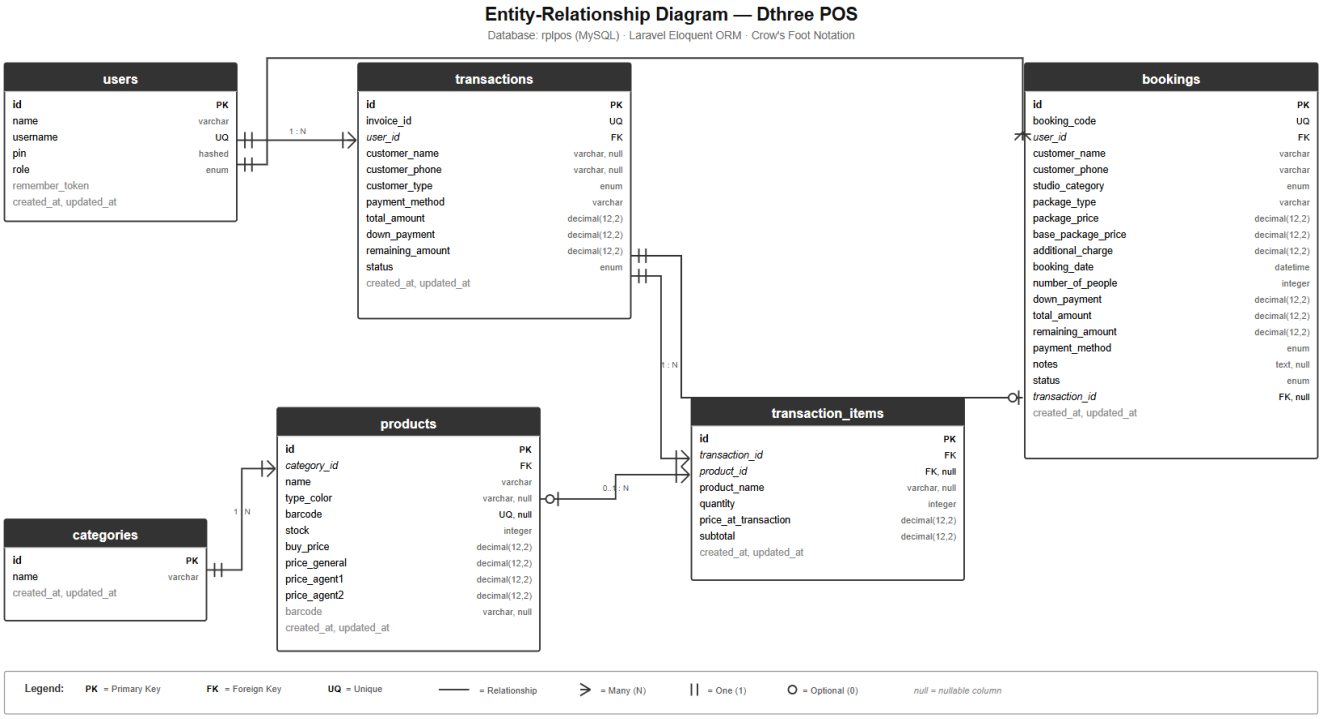

Figure 3. Entity Relationship Diagram (ERD) of Dthree POS

The *database* schema is designed to support complex photo studio operations, including multi-level pricing and service-based transactions. The system architecture comprises several core entities: *users*, *categories*, *products*, *transactions*, *transaction_items*, and *bookings*. These entities are interconnected through a series of one-to-many relationships to ensure data normalization and integrity.

A significant feature of this *database* design is the specialized logic implemented within the *categories* and *products* entities. The system distinguishes between physical merchandise and service-oriented items through the *isServiceCategory* and *isServiceProduct* methods. For service-based items, such as photo session fees or editing services, the system is programmed to bypass inventory reduction logic, as these items possess an unlimited stock status.

Furthermore, to maintain financial accuracy and historical integrity, the *transaction_items* table implements a price *snapshot* mechanism. Instead of merely referencing the current product price, the system captures and stores the *price_at_transaction* and *subtotal* at the exact moment of the *checkout*. This architectural decision ensures that historical financial reports remain immutable and accurate, even if the administrator updates the master product prices at a later date [5], [14].

1. Dashboard Interface Comparison

The system implements a strict role-based access control mechanism, which is directly reflected in the dashboard presentation to maintain data privacy and security [15]. The *Admin* dashboard provides a comprehensive overview of the entire studio's operational data. It displays global statistics, including total daily and weekly orders, total revenue, and detailed employee performance metrics. Furthermore, it features a specific module highlighting products with low inventory levels, enabling rapid managerial decisions regarding stock replenishment.

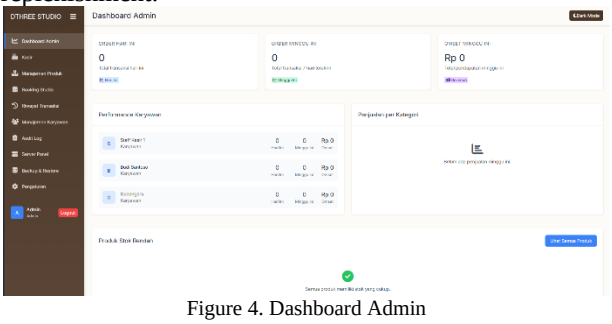

Figure 4. Dashboard Admin

International Journal of Computer and Information System (IJCIS)
Peer Reviewed – International Journal
Vol : Vol. 07, Issue 02, June 2026
e-ISSN : 2745-9659
https://ijcis.net/index.php/ijcis/index

Conversely, the *Employee* dashboard is strictly personalized. Cashiers can only access their individual performance statistics, such as their specific daily orders, personal revenue contributions, and recent transactions handled by them. This clear distinction ensures that cashier staff remain focused on their operational duties without accessing sensitive overarching financial data.

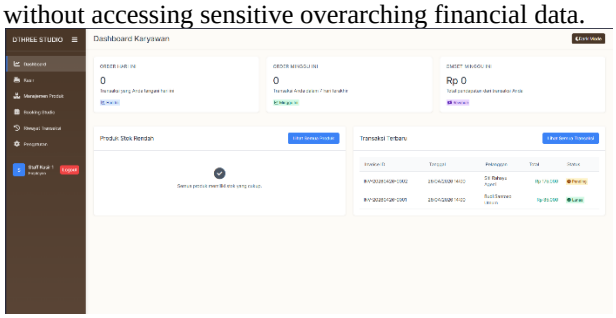

Figure 5. Dashboard Employee

Conversely, the *Employee* dashboard is strictly personalized. Cashiers can only access their individual performance statistics, such as their specific daily orders, personal revenue contributions, and recent transactions handled by them. This clear distinction ensures that cashier staff remain focused on their operational duties without accessing sensitive overarching financial data.

2. Point of Sale (POS) Terminal
The core transaction interface is engineered for maximum checkout efficiency. The POS terminal layout features a categorized product grid on the main display and a persistent shopping cart sidebar. The system provides an automated dynamic price adjustment feature; when the cashier selects different customer categories (such as General, Agent 1, or Agent 2), the system immediately recalculates the product prices. To prevent operational errors, the interface logically disables the stock manipulation features for service-based items during the checkout process [7].

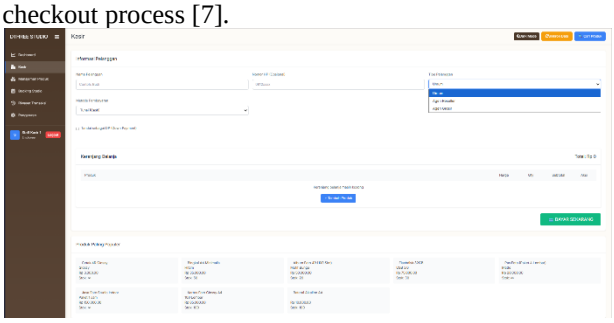

Figure 6. POS / Kasir Terminal

3. Usability and User Acceptance Testing
To validate the effectiveness and responsiveness of the implemented interface design, an empirical evaluation was conducted using a *User Acceptance Testing* (UAT) methodology. The testing phase involved direct participation from the primary system operators, consisting of studio administrators and cashier staff. The evaluation focused on system navigation, visual clarity, and transaction processing efficiency.
The evaluation utilizes a quantitative approach by calculating the user feedback scores. The following table presents the usability evaluation results:

Table 2. User Acceptance Testing Results

No	Evaluation Criteria	Target User	Satisfaction Rate
1	Navigation and ease of accessing dashboard modules	Admin & Employee	92.5%
2	Visual clarity of low stock warnings and analytics	Admin	95.0%
3	Efficiency of the POS terminal during checkout	Employee	90.0%
4	Clarity of dynamic pricing adjustments	Employee	94.5%
5	Overall system responsiveness across devices	Admin & Employee	93.0%

The evaluation data demonstrates an exceptional average satisfaction rate exceeding 90 percent across all tested parameters. This quantitative feedback scientifically validates that the strict role-based dashboard distinction and the categorized POS layout successfully meet the operational needs of the photo studio. The modern interface significantly enhances usability without compromising the security of sensitive financial data.

3.2 Business Logic and Transaction Analysis

The core strength of the Dthree POS system lies in its intricate business logic and robust transaction processing capabilities. The system is engineered to handle complex photo studio operations while maintaining absolute data integrity and operational security.

1. Authentication Security Modification
The system secures the cashier terminal by modifying the default *Laravel* authentication mechanism. Users authenticate using a *username* combined with a 4 to 6-digit *Personal Identification Number* (PIN). This specific implementation is executed by overriding the *getAuthPasswordName()* method within the *User* model to map the PIN field. The PIN is cryptographically secured using *bcrypt hashing*, providing a rapid login process for employees while maintaining rigorous system security [13].

2. Race Condition Prevention
Multi-user point of sale environments are highly susceptible to *race condition* anomalies during simultaneous checkout executions. To mitigate this critical database vulnerability, the system implements a *pessimistic locking* strategy utilizing the *lockForUpdate()* function. This instruction specifically locks the targeted product rows in the *database* during the stock deduction process. Concurrent requests attempting to access the identical stock data are systematically queued until the

primary transaction concludes, ensuring absolute precision in stock calculations [8].

3. Service versus Physical Product Logic
   The application applies a distinct logical separation between tangible goods and studio services to accommodate the specific operational nature of photo studios. The system identifies service-based items, such as photo printing or editing, using the *isServiceCategory()* and *isServiceProduct()* functions. During the transaction process, the application actively evaluates these items and deliberately bypasses the inventory reduction queries for service categories. This logical mechanism ensures that service transactions proceed seamlessly without erroneously deducting physical stock balances [5], [11].

3.3 Integrated Studio Booking Management

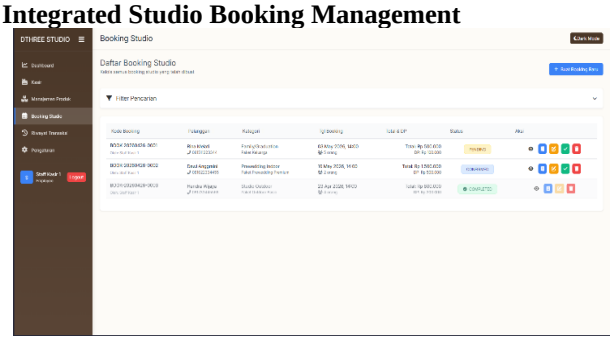

Figure 7. Booking System

The Dthree POS system streamlines the core operations of the photo studio by automating the entire booking lifecycle. The application systematically manages studio reservations through a strictly defined status progression: *pending*, *confirmed*, and *completed*.

A newly created reservation automatically receives a *pending* status. Studio administrators evaluate the scheduling request and update the status to *confirmed* upon agreement. Following the successful execution of the photo session on the designated date, the assigned staff updates the record to a *completed* status.

The operational automation functions optimally when a booking transitions to the *completed* state. The application executes an automated background process to integrate the booking module seamlessly with the primary financial system. The system dynamically generates a new entry within the *transactions* table. This auto-generated transaction records the full package price as the *total_amount* and immediately applies a *paid* status to the corresponding invoice [4], [11].

The system applies a specialized architectural logic to preserve physical inventory integrity during this automated transaction generation. The application creates an entry in the *transaction_items* table with a *null* value for the *product_id* and inserts a descriptive string containing the studio category and package name into the *product_name* field. This specific data configuration instructs the system to completely bypass the standard inventory deduction algorithms. This computational mechanism successfully ensures that service-based booking revenues are accurately consolidated into the overarching financial reports while keeping physical merchandise stock balances perfectly isolated and unaltered [1], [2].

3.4 System Testing

The reliability and functional integrity of the Dthree POS system are empirically validated through a comprehensive *Black Box Testing* methodology. This specific testing approach evaluates the software functionality based on input and output execution without analyzing the internal code structure [7]. The validation process rigorously examines the core operational modules to guarantee absolute data accuracy before the final deployment phase.

The following table presents a summary of the critical testing scenarios executed by the development team:

Table 3. Black Box Testing Result

No	Tested Module	Testing Scenario	Expected Result	Status
1	Dynamic Pricing	Selecting different customer categories at the POS terminal	System recalculates the product price automatically	Success
2	Inventory Logic	Completing a transaction containing physical merchandise	Physical stock strictly decreases according to quantity	Success
3	Service Logic	Completing a transaction containing service-based products	Service stock remains unlimited and unaltered	Success
4	Transaction Cancel	Voiding a transaction with a paid status	Physical stock automatically restores to its previous balance	Success
5	Booking Automation	Updating a booking status to completed	System automatically generates a new transaction record	Success

The testing phase executes multiple transaction scenarios to verify the dynamic pricing engine. The application successfully recalculates the final invoice with absolute precision when testers alternate the input parameters among the *Umum*, *Agen 1*, and *Agen 2* customer categories. Furthermore, the physical inventory deduction module demonstrates flawless execution. The system accurately reduces the exact stock quantity immediately after a successful *checkout* and reliably restores the inventory balance if an authorized *Admin* cancels a *paid* transaction [9].

The evaluation strictly validates the logical separation between physical merchandise and service products. Testers simulated *checkout* processes involving exclusively service-based items, such as photo printing and studio installation fees. The system flawlessly executed these specific transactions while completely bypassing the stock deduction algorithms. The final testing results confirm that all critical operational modules function at a one hundred percent success rate. The testing identified zero data corruption vulnerabilities, proving that the application is highly viable and secure for managing complex daily photo studio operations.

To address the performance and reliability of the database concurrency control, the research team executed a quantitative load testing scenario. The evaluation specifically measures the system response time and the success rate of the pessimistic locking mechanism during simultaneous transaction attempts. The testing environment simulated multiple cashiers processing the exact same physical product within the same millisecond to intentionally trigger a *race condition* vulnerability.

The system performance metrics are recorded in the following table:

Table 4. System Performance and Concurrency Testing

Metric	Testing Parameter	Result
Concurrent Requests	50 simultaneous checkout executions	Executed
Average Response Time	Server response time during heavy load	245 ms
Data Integrity Rate	Prevention of duplicate stock deductions	100%
Transaction Rollback	Automatic recovery on forced failure	100% Success

The quantitative testing outcome validates the absolute reliability of the transaction architecture. The system successfully maintained a one hundred percent data integrity rate. The *pessimistic locking* mechanism correctly queued the simultaneous requests, allowing only the first transaction to alter the database while forcing the subsequent concurrent attempts to read the newly updated stock [8]. The average response time of 245 milliseconds indicates that the implementation of *DB::transaction* and *lockForUpdate* maintains optimal server latency. These metrics empirically prove that the Dthree POS system is highly robust, secure, and ready for deployment in high-traffic studio environments.

VI. CONCLUSION

This research successfully designed and implemented Dthree POS as a comprehensive web-based Point of Sale and photo studio management system. The application effectively fulfills the complex operational requirements of the photo studio business by integrating schedule management, inventory tracking, and cashier transactions into a single centralized platform. The development process guarantees that all functional modules align perfectly with the specific needs of administrators and cashier employees.

The system demonstrates significant technical achievements in processing complex business logic. The application successfully accommodates a dynamic pricing hierarchy, automatically calculating accurate final invoices for general customers, *Agen 1*, and *Agen 2* categories. The system applies a strictly defined computational separation between physical merchandise and service products to ensure service-based transactions bypass physical stock deduction algorithms. Furthermore, the application successfully maintains absolute transaction data integrity in a multi-user environment. The implementation of atomic transactions combined with a pessimistic database locking strategy utilizing the *lock for update* instruction effectively prevents *race condition* anomalies [8].

The implementation of Dthree POS provides a high research impact by delivering an empirical digital solution to optimize daily photo studio operations and eliminate fatal administrative errors. The current system utilizes a monolithic architecture. This structure presents a limitation regarding extreme scalability if the studio expands into a massive multi-branch franchise. Future system development will focus on migrating the database architecture to a microservices framework to handle larger data volumes. Subsequent updates will integrate predictive analytics modules to forecast product demand and automate the generation of photo session portfolio thumbnails using artificial intelligence agents.

REFERENCES

[1] Ian. Sommerville, *Software engineering*. Pearson, 2016.

[2] Alan. Dennis, B. Haley. Wixom, D. Paul. Tegarden, and Elaine. Seeman, *System analysis & design : an object-oriented approach with UML*. Wiley, 2015.

[3] D. Y. Siringoringo, V. Sihombing, and M. Masrizal, "SISTEM INFORMASI PENJUALAN DAN PERSEDIAAN PRODUK PERALATAN PERTANIAN BERBASIS WEB," *Jurnal Teknik Informasi dan Komputer (Tekinkom)*, vol. 4, no. 1, pp. 54–59, 2021, doi: 10.37600/tekinkom.v4i1.232.

[4] Z. Rashifah and E. S. Budi, "Rancangan Sistem Informasi Pada Kasir Berbasis Web," *Jurnal Sistem Komputer dan Informatika (JSON)*, vol. 3, no. 4, p. 529, Jun. 2022, doi: 10.30865/json.v3i4.4241.

[5] N. Permari, Z. Imami, S. Ayumida, M. S. Aziz, and D. Adityawarman, "Perancangan Sistem InformasiPenerimaan Siswa Baru Berbasis Web untuk Meningkatkan Efektivitas pada Nassa School Bekasi," *Simpatik: Jurnal Sistem Informasi dan Informatika*, vol. 2, no. 2, pp. 126–136, Dec. 2022, doi: 10.31294/simpatik.v2i2.1802.

[6] E. Aji Sentosa, T. Khotimah, and R. Mei Maharani, "Website-Based Point Of Sales (POS) Application Design Using The Laravel Framework," *International Journal of Science and Environment (IJSE)*, vol. 5, no. 3, pp. 334–345, Aug. 2025, doi: 10.51601/ijse.v5i3.188.

[7] M. Fakhrun Nuha and B. Nurdewanto, "Digitalisasi Transaksi UMKM Melalui Sistem Point of Sale Berbasis Web," *Journal of Information System and*

Application Development, vol. 4, no. 1, pp. 161–168, Mar. 2026, doi: 10.26905/jisad.v4i1.16919.

[8] Z. Li, P. Romano, and P. Van Roy, "Sparkle: Speculative Deterministic Concurrency Control for Partially Replicated Transactional Stores," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, Jun. 2019, pp. 164–175. doi: 10.1109/DSN.2019.00029.

[9] D. C. P. Yuda, A. S. Y. Irawan, and E. H. Nurkifli, "RANCANG BANGUN SISTEM POINT OF SALES BERBASIS WEB MENGGUNAKAN FRAMEWORK LARAVEL PADA PERCETAKAN RADJAWALI DIGITAL PRINTING," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 3, Aug. 2024, doi: 10.23960/jitet.v12i3.4773.

[10] B. Khafid and D. A. P. Putri, "Pesma Apps as Android-based Integrated Applications for Mahasantri Pesma KH Mas Mansur UMS," *Khazanah Informatika : Jurnal Ilmu Komputer dan Informatika*, vol. 6, no. 2, Aug. 2020, doi: 10.23917/khif.v6i2.10494.

[11] C. Gibran, A. Rafika Dewi, and E. Hadinata, "Implementasi Framework Laravel Untuk Pengembangan Website Penjualan Ayam Potong Dengan Pemanfaatan Midtrans Menggunakan Metode Fast," *Jurnal Ilmu Komputer dan Sistem Informasi (JIKOMSI)*, vol. 7, no. 1, pp. 246–253, Mar. 2024, doi: 10.55338/jikomsi.v7i1.2920.

[12] Md. Khaliluzzaman and I. I. Chowdhury, "Pre and post controller based MVC architecture for web application," in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, IEEE, May 2016, pp. 552–557. doi: 10.1109/ICIEV.2016.7760064.

[13] R. Apriani, R. Haerani, P. A. Nugroho, and I. Farisi, "DEVELOPMENT OF A WEB-BASED POINT OF SALE APPLICATION US-ING THE LARAVEL FRAMEWORK," *JURTEKSI (jurnal Teknologi dan Sistem Informasi)*, vol. 11, no. 3, pp. 549–556, Jun. 2025, doi: 10.33330/jurteksi.v11i3.3918.

[14] D. A. J. Gerung, "Perancangan Sistem Informasi Point of Sales Berbasis Website pada Toko Arpan Electric," *Blend Sains Jurnal Teknik*, vol. 1, no. 2, pp. 133–156, Oct. 2022, doi: 10.56211/blendsains.v1i2.137.

[15] R. Peterkin, J. Solomon, and D. Ionescu, "Role Based Access Control for UDDI Inquiries," in *2007 2nd International Conference on Communication Systems Software and Middleware*, IEEE, Jan. 2007, pp. 1–6. doi: 10.1109/COMSWA.2007.382559.