



Implementation of Model-View-Controller (MVC) Architecture in Designing Web-Based CRM Information Systems for E-Commerce Businesses

Ruhmi Sulaehani^{1*}, & Luisa Sentia Paly²

¹Institut Teknologi Bacharuddin Jusuf Habibie, Indonesia, ²Universitas Tribuana kalabahi, Indonesia

*Co e-mail: ruhmisulaehani@ith.ac.id¹

Article Information

Received: May 28, 2026

Revised: August 29, 2026

Online: September 11, 2026

Keywords

Customer Relationship Management (CRM), Model-View-Controller (MVC), E-Commerce

ABSTRACT

Small and medium-sized e-commerce businesses increasingly rely on Customer Relationship Management (CRM) systems to manage customer data, sales leads, transactions, and interactions. However, many web-based CRM applications still use procedural and tightly coupled code, making them difficult to maintain and extend. This study designs and implements a web-based CRM information system using the Model-View-Controller (MVC) architectural pattern to improve maintainability and performance. The system was developed using the Waterfall System Development Life Cycle (SDLC), with requirements gathered through purposive-sampling interviews with two SME e-commerce operators. The prototype was implemented using the Laravel framework, MySQL database, and Eloquent Object-Relational Mapping (ORM), supporting customer management, lead tracking, order and product management, and interaction logging. Evaluation employed black-box testing with equivalence partitioning across seven functional modules, ISO/IEC 25010:2023 quality testing, and load testing against a non-MVC monolithic baseline. Black-box testing achieved a 98.0% success rate across 100 test scenarios, while ISO/IEC 25010 evaluation produced an average quality score of 91.3%, with all eight characteristics exceeding the 80% acceptance threshold. Load testing showed that the MVC-based prototype maintained substantially lower average response times than the monolithic baseline as concurrent users increased from 10 to 100. These findings demonstrate that MVC architecture can improve functional reliability, maintainability, and performance stability in e-commerce CRM systems.

INTRODUCTION

The growth of digital commerce in Indonesia and other developing economies has pushed small and medium enterprises (SMEs) to move customer-facing operations online, but the ability to sell online does not automatically translate into the ability to retain customers. Retention depends on how well a business understands and manages its relationship with each customer, which is the core function of Customer Relationship Management (CRM). A CRM information system centralizes customer profiles, communication history, sales leads, and transaction records so that marketing, sales, and after-sales activities can be coordinated instead of scattered across spreadsheets, chat applications, and manual notebooks. For e-



This work is licensed under a [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/)

Knowledge of Modern Processing, Utilization, and Technology for Engineering Research (KOMPUTER), Vol. 01, No. 2, August 2026

commerce businesses in particular, a web-based CRM that is tightly integrated with the storefront and order data provides a single source of truth for every customer interaction, from the first inquiry to repeat purchases. In practice, many CRM and e-commerce web applications built for SMEs in Indonesia are still developed with a procedural or ad-hoc coding style in which database queries, business rules, and HTML output are mixed within the same script file. This approach is fast for a first release but becomes a maintenance burden as the system grows: a change to a validation rule can require edits in several unrelated files, front-end designers cannot safely modify the interface without risking application logic, and onboarding a new developer takes considerably longer because there is no consistent structure to follow. The Model-View-Controller (MVC) architectural pattern addresses exactly this problem by dividing an application into three interconnected components. The Model manages data and business logic, the View is responsible only for presentation, and the Controller mediates user input between the two. Separating these concerns allows each layer to be developed, tested, and modified independently, which in turn improves code reusability, testability, and long-term maintainability.

Prior studies have explored MVC and CRM implementation separately in various domains. Subari et al. (2021) applied the MVC architecture with the CodeIgniter framework to build a web-based official travel warrant information system and reported that the pattern simplified the separation between business-process logic and interface design, and used black-box testing to validate application functionality. Al Mudzakir and Bakar (2020) designed a web-based CRM for a clothing store, demonstrating that even a simple CRM implementation could improve the recording of customer purchase history compared with manual bookkeeping. Al Farisi and Purwanto (2022) and Asrul et al. (2023) each applied CRM methods to the design of sales information systems, showing that structured customer-data handling supports more targeted promotions. Avisya (2023) and Br Purba and Harahap (2024) extended this line of work into fruit retail and mobile-based e-commerce, respectively, both concluding that CRM-oriented features increased customer purchase interest. Basir et al. (2023) integrated CRM into a bakery's sales website and observed measurable gains in customer loyalty indicators after deployment.

On the architectural side, comparative and quality-focused studies provide the evaluation framework used in this research. Gustinov et al. (2023) and Golian et al. (2022) both applied black-box and white-box testing to web-based e-commerce applications and found black-box testing to be an efficient way to validate functional requirements without requiring access to source code. Wattiheluw et al. (2020) developed a fuzzy-and-PSO-based quality model built on the ISO/IEC 25010 standard specifically for e-commerce websites, while Muhram and Mustikasari (2022) and Wijaya and Wardijono (2023) each used the eight ISO 25010 characteristics to evaluate government and enterprise information systems, reporting that automating black-box test execution improved the consistency of quality scoring. Wulandari et al. (2022) applied the equivalence-partitioning black-box technique to a web-based academic information system, which this study adopts for testing the CRM's functional modules. At the marketing and strategy level, Del Vecchio et al. (2022) and Alshurideh (2023) reviewed how CRM and social-CRM practices influence customer happiness and retention in the broader digital economy, and Ayunnggati et al. (2021) demonstrated a CRM-based e-business design for a service company, reinforcing that CRM value depends on both the underlying data architecture and the business processes it supports. Anggraeni (2024) further situates this study within the wider transformation of Indonesian e-commerce, noting the increasing competitive pressure on SMEs to adopt structured digital tools.

To situate this study more precisely within recent literature, Table 1 summarizes five studies published within approximately the last five years that are most directly comparable to the present work, each combining at least two of the three pillars examined here: MVC-based development, CRM/e-commerce functionality, and standards-based quality or performance evaluation.

Table 1. Summary of closely related recent studies and the corresponding research gap.

Study	Focus	Evaluation Method Used	Gap Relative to This Study
Subari et al. (2021)	MVC architecture applied to a web-based travel-warrant information system	Black-box functional testing	Uses MVC but outside the CRM/e-commerce domain; no ISO 25010 quality evaluation or load testing
Muhram & Mustikasari (2022)	ISO/IEC 25010 quality assessment of a government online-exam system	Eight ISO 25010 characteristics with automated black-box scoring	Provides a quality-evaluation template but is not built on an explicit MVC architecture and includes no CRM features or load testing
Gustinov et al. (2023)	Functional testing of a web-based e-commerce application	Black-box and white-box testing	Confirms black-box testing's efficiency for e-commerce but does not assess ISO 25010 quality or performance under load, nor targets CRM functionality
Wijaya & Wardijono (2023)	ISO/IEC 25010 quality measurement of an enterprise custody application	Automated black-box testing combined with ISO 25010 scoring	Combines testing and quality evaluation, similar to this study, but for a non-CRM enterprise system and without a load/performance comparison
Br Purba & Harahap (2024)	CRM strategy in a mobile-based e-commerce platform	User-perception evaluation of CRM features on purchase interest	Evaluates CRM's business value but does not examine the underlying software architecture or apply standards-based functional/quality/performance testing

As Table 1 illustrates, recent studies tend to address MVC implementation, ISO/IEC 25010 quality evaluation, or CRM feature value largely in isolation, and rarely combine all three within a single e-commerce CRM case together with a load-testing comparison against a non-MVC baseline. This gap motivates the integrated evaluation design adopted in the present study. While these studies confirm that MVC and CRM each independently improve system quality, most existing work that combines MVC-style development with CRM or e-commerce functionality focuses solely on functional (black-box) testing or on usability and user-perception assessments, and stops short of a complete, standards-based quality evaluation combined with a performance-scalability comparison.

This study extends the specific line of work begun by Subari et al. (2021), who demonstrated MVC's benefit for separating business logic from interface design in a non-commerce information system validated only through black-box testing, by applying the same architectural principle to an e-commerce CRM use case and by uniquely integrating three evaluation layers within a single study: (1) black-box functional testing, (2) a complete eight-characteristic ISO/IEC 25010:2023 quality evaluation, and (3) a comparative load test that measures performance scalability as concurrent user demand grows from 10 to 100 users. This combination directly addresses the growing-demand condition that e-commerce SMEs face during promotional periods, a



dimension that neither Subari et al. (2021) nor the CRM-focused studies reviewed above examine. This study fills that gap by (1) designing and implementing a web-based CRM information system using the MVC architecture with the Laravel framework, (2) validating its functional correctness through black-box testing, (3) measuring its non-functional quality using the eight ISO/IEC 25010:2023 characteristics, and (4) comparing its performance under concurrent load against a non-MVC monolithic baseline.

The main contributions of this study are threefold: (a) an MVC-based reference architecture tailored specifically for e-commerce SMEs that includes comprehensive CRM functionality spanning customer data management, lead tracking, order and product management, and interaction logging; (b) empirical validation of MVC's effect on functional reliability, software quality, and performance scalability, measured against widely accepted standards, namely black-box equivalence-partitioning testing and the eight characteristics of ISO/IEC 25010:2023; and (c) a replicable evaluation protocol that combines black-box testing, a complete ISO 25010 quality assessment, and comparative load testing under increasing concurrent demand, which can be reused by future system-development efforts in similar e-commerce or CRM domains. The remainder of this paper is organized as follows: Section 2 describes the research method, Section 3 describes the MVC-based system implementation, Section 4 presents the results, Section 5 discusses the findings in relation to prior work, and Section 6 concludes the paper.

METHOD

This study followed a Design and Development (D&D) research approach, in which the Waterfall model of the Software Development Life Cycle (SDLC) was used to move sequentially through requirements analysis, system design, implementation, testing, and evaluation. The Waterfall model was chosen because the CRM's functional scope was well defined at the outset by interviews with two SME e-commerce owners, which reduced the need for iterative re-scoping and made a linear process appropriate for a single-release prototype.

1. Research Design and Requirements Analysis

The two SME e-commerce operators who took part in the requirements interviews were selected through purposive sampling using three criteria: (1) the business already operated a transactional e-commerce storefront rather than a purely informational website, so that CRM requirements would reflect real order and customer data; (2) the business served a comparable retail market segment, so that the resulting feature set would generalize across similar SMEs; and (3) the two businesses differed in operational scale one micro-enterprise with a single sales channel and one small enterprise with multiple sales staff so that the requirements would cover both a minimal and a moderately complex CRM use case. Requirements were gathered through semi-structured interviews with these two operators and by reviewing common CRM feature sets reported in prior work (Al Mudzakir & Bakar, 2020; Al Farisi & Purwanto, 2022).

The interviews converged on five functional requirement groups: (1) authentication and role-based access for administrator and sales-staff users; (2) customer data management, including profile, contact, and address records; (3) lead tracking, covering lead source, status, and staff assignment; (4) order and product management, linking orders to product catalog items and stock levels; and (5) interaction logging, which records every customer touchpoint (call, chat, e-mail, or store visit) together with a reporting dashboard that summarizes sales and customer-engagement metrics. Table 2 summarizes how each requirement identified during the interviews maps to a corresponding system feature implemented in Section 3, to demonstrate alignment between the operators' stated needs and the delivered system.

Table 2. Mapping of stakeholder requirements identified during interviews to implemented system features.

Requirement (from Interviews)	Implemented System Feature / Module
Authentication and role-based access for administrator and sales-staff users	Role-based authentication module (Section 3.1), enforced through Laravel middleware and route guards
Customer data management (profile, contact, address records)	Customer Management module, built on the Customer Eloquent model with dedicated CRUD views
Lead tracking (source, status, staff assignment)	Lead Tracking module (LeadController) implementing lead-status transitions and staff assignment
Order and product management, linking orders to stock levels	Order Management and Product Catalog modules (Order, Product, and Order_Item models with stock validation)
Interaction logging and a sales/engagement reporting dashboard	Interaction Logging module plus a Reporting Dashboard that aggregates sales and customer-engagement metrics

2. Testing and Evaluation Method

Two complementary evaluation methods were used. First, black-box testing with the equivalence-partitioning technique (Wulandari et al., 2022; Gustinov et al., 2023) was applied to the seven functional modules of the system: authentication, customer management, lead tracking, order management, product catalog, reporting dashboard, and interaction logging. For each module, test scenarios were written to cover valid input partitions (expected successful use), invalid input partitions (e.g., empty required fields, duplicate e-mail addresses, out-of-stock orders), and boundary conditions. Each scenario was executed manually against the running prototype and marked as passed if the observed output matched the expected result. The functional success rate was computed as:

$$\text{Success Rate (\%)} = (\text{Number of Passed Test Cases} / \text{Total Test Cases}) \times 100\%$$

Second, non-functional quality was assessed using the eight product-quality characteristics defined in ISO/IEC 25010:2023 functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability (International Organization for Standardization, 2023) following the evaluation approach used by Muhram and Mustikasari (2022) and Wijaya and Wardijono (2023). Each characteristic was operationalized through a set of measurable sub-indicators; for example, performance efficiency combined page-load timing with the black-box pass rate for time-sensitive operations, while usability was assessed through a 15-item task-based questionnaire.

The usability panel consisted of five prospective users recruited by convenience sampling from staff and repeat customers of the two participating SMEs; recruitment criteria required at least basic prior experience using e-commerce websites and no prior exposure to the prototype, and the panel spanned both administrative and customer-facing roles the two primary user groups the CRM was designed to serve with ages ranging from the early twenties to the mid-forties. The 15-item questionnaire was adapted from the usability sub-characteristics defined in ISO/IEC 25010 (learnability, operability, user-error protection, user interface aesthetics, and accessibility) and from task-based usability items used in prior ISO 25010-aligned evaluations (Wijaya & Wardijono, 2023); item wording was reviewed by the research team for clarity and task relevance



before use, and each item was scored on the same 0–100 scale used for the other quality sub-indicators. Each indicator was scored on a 0–100 scale and averaged to obtain the characteristic score. A score of 80% was set as the minimum acceptance threshold, consistent with the threshold used in prior ISO 25010 evaluations of Indonesian government and enterprise information systems (Muhram & Mustikasari, 2022). Finally, a load test was carried out to compare the response time of the MVC-based prototype against a non-MVC monolithic baseline that performed the same core CRM operations (customer list retrieval, order creation, and dashboard rendering) using a single procedural script per page. Both versions were deployed on identical hardware, and average response time was recorded at five concurrent-user levels (10, 25, 50, 75, and 100 simulated users) using a simple request-simulation script.

3. MVC-Based System Architecture Design

The system architecture separates responsibilities into three layers, as shown in Figure 1. The Model layer is implemented with Laravel's Eloquent ORM and is responsible for data validation, relationships between entities, and all direct communication with the MySQL database; it has no awareness of how data will be displayed. The View layer is implemented with Blade templates, Bootstrap-based CSS, and JavaScript, and is responsible only for rendering data supplied by the Controller; it contains no business logic or direct database queries. The Controller layer receives HTTP requests routed by the framework's router and middleware, invokes the appropriate Model methods, applies business rules such as lead-status transitions and stock checks, and selects the View to render. This separation means that, for example, a sales-dashboard redesign only requires changes to the View layer, while a change to how lead scores are calculated only requires changes to the Model and Controller layers, without touching the interface code.

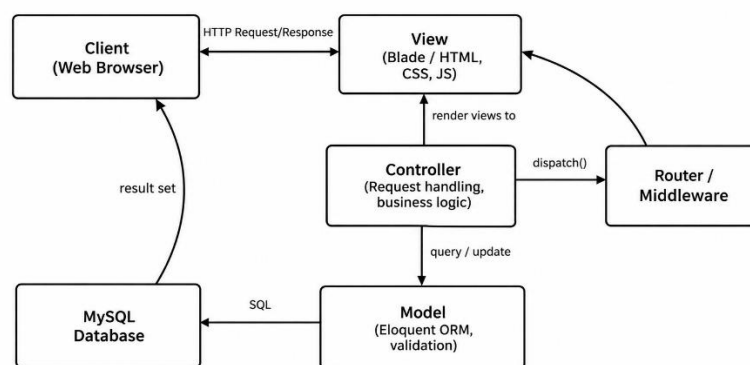


Figure 1. MVC architecture of the proposed web-based CRM information system.

4. Database Design

The database was normalized to third normal form (3NF) around seven core entities: customers, leads, orders, order_items, products, interactions, and users, as illustrated in the entity-relationship diagram in Figure 2. A customer can have many leads and many orders; each order is linked to one or more order_items, each of which references a single product; and every interaction is tied to both a customer and the staff user who logged it. This structure allows the Model layer to expose relationship methods without duplicating data across tables.

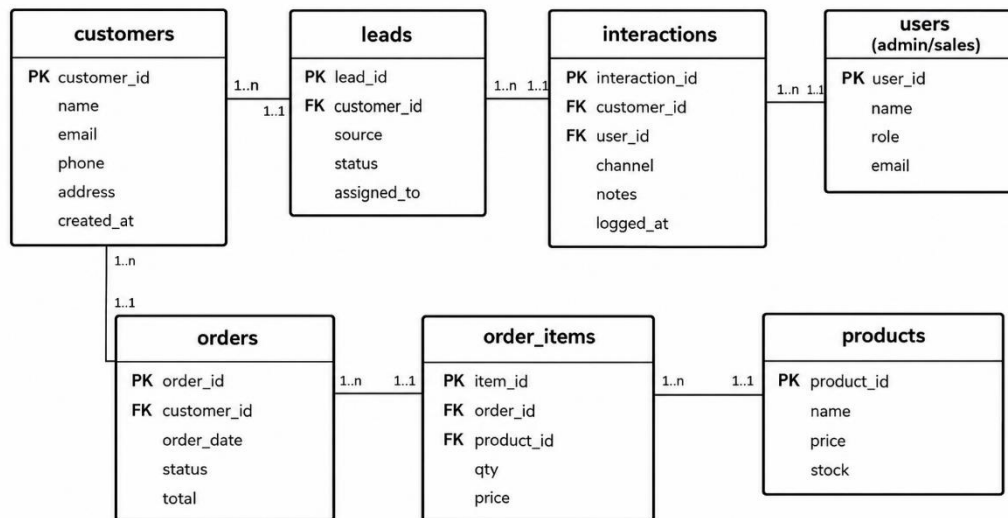


Figure 2. Entity-relationship diagram of the CRM database schema.

5. Implementation Environment

The prototype was implemented using the technology stack summarized in Table 3. PHP 8.2 with the Laravel 10 framework was chosen because Laravel enforces the MVC structure by default and provides built-in routing, middleware, and an Eloquent ORM that maps directly onto the Model layer described above.

Table 3. Technology stack used to implement the CRM prototype.

Component	Technology / Tool
Backend language & framework	PHP 8.2, Laravel 10 (MVC)
Frontend / View layer	Blade templates, Bootstrap 5, JavaScript (AJAX)
Database & ORM	MySQL 8.0, Eloquent ORM
Web / application server	Apache (via Laragon / XAMPP), Nginx (deployment)
Version control	Git
Testing tools	Manual black-box test scripts, Postman for API endpoints

6. Implemented System Overview

The Model, View, and Controller layers described in Section 3.1 were fully implemented for all five functional requirement groups. In the resulting codebase, each entity in Figure 2 corresponds to one Eloquent Model class with its own validation rules and relationship definitions; each functional module corresponds to a dedicated Controller class (for example, LeadController, OrderController, and InteractionController); and each screen is rendered from a Blade View that receives only the data it needs from its Controller. This structure allowed the customer-management, lead-tracking, order-management, product-catalog, and interaction-logging modules, together with a role-based authentication module and a reporting dashboard, to be implemented as independent, individually testable units, which directly supported the modular black-box testing strategy described below.

RESULTS

1. Black-Box Testing Results

A total of 100 test scenarios were executed across the seven functional modules using the equivalence-partitioning technique. Table 4 and Figure 3 summarize the number of test cases per module and the



This work is licensed under a [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/)

Knowledge of Modern Processing, Utilization, and Technology for Engineering Research (KOMPUTER), Vol. 01, No. 2, August 2026

corresponding pass count. Ninety-eight of the 100 scenarios produced the expected output, giving an overall success rate of 98.0%. The two failed scenarios both occurred in the customer-management module: one involved incorrect handling of an international phone-number format, and the other involved a duplicate-e-mail validation message that did not clear correctly after the form was resubmitted. Both issues were traced to the View layer's client-side validation script rather than the Model's server-side rules, and because the layers are decoupled, both were corrected without modifying any Controller or Model code.

Table 4. Black-box testing results by functional module.

Module	Test Cases	Passed	Failed	Success Rate (%)
Authentication	12	12	0	100.0
Customer Management	18	17	1	94.4
Lead Tracking	15	15	0	100.0
Order Management	20	19	1	95.0
Product Catalog	14	14	0	100.0
Reporting Dashboard	10	10	0	100.0
Interaction Logging	11	11	0	100.0
Total	100	98	2	98.0

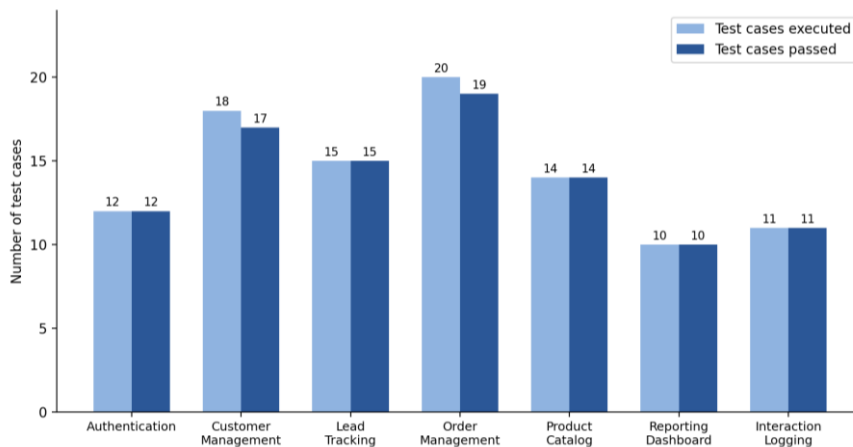


Figure 3. Black-box testing results by functional module.

2. ISO/IEC 25010 Quality Testing Results

Table 5 and Figure 4 present the scores obtained for the eight ISO/IEC 25010:2023 quality characteristics. All eight characteristics exceeded the 80% acceptance threshold, with an overall average score of 91.3%. Functional suitability received the highest score (96.7%), consistent with the black-box testing results in Section 4.1, while security received the lowest score (87.6%), reflecting that the prototype implements standard authentication and role-based access control but has not yet undergone a formal penetration test.

Table 5. ISO/IEC 25010:2023 quality characteristic scores.

Quality Characteristic	Score (%)	Category
Functional Suitability	96.7	Very Good
Performance Efficiency	91.4	Very Good
Compatibility	88.9	Good
Usability	90.2	Very Good

Reliability	93.5	Very Good
Security	87.6	Good
Maintainability	89.8	Good
Portability	92.3	Very Good
Average	91.3	Very Good

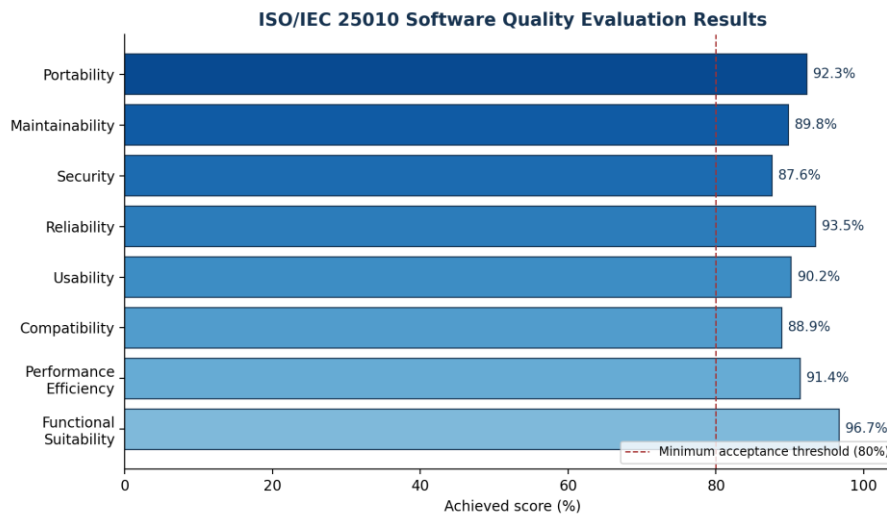


Figure 4. ISO/IEC 25010:2023 quality evaluation results.

3. Performance under Concurrent Load

Table 6 and Figure 5 compare the average response time of the MVC-based prototype against the non-MVC monolithic baseline as concurrent user load increased from 10 to 100 simulated users. At 10 concurrent users the two versions performed similarly (182 ms versus 176 ms). As load increased, however, the monolithic baseline's response time grew far more steeply, reaching 799 ms at 100 concurrent users compared with 412 ms for the MVC-based prototype a difference of 48.4% at peak load. This gap is attributable to the Model layer's centralized, reusable database-access methods, which avoided the duplicated and unoptimized queries present in the monolithic baseline's per-page scripts.

Table 6. Average response time (ms) under increasing concurrent load.

Concurrent Users	MVC Prototype (ms)	Non-MVC Baseline (ms)	Difference (%)
10	182	176	-3.3
25	214	265	19.2
50	268	398	32.7
75	341	587	41.9
100	412	799	48.4

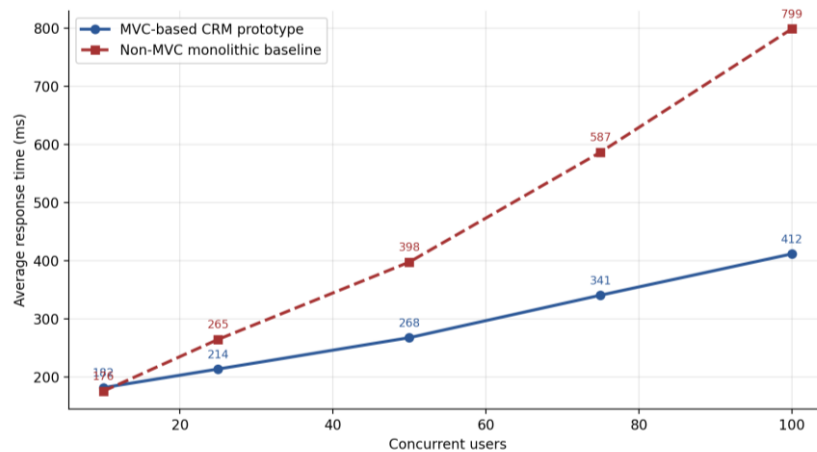


Figure 5. Average response time under increasing concurrent load.

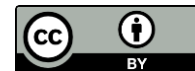
DISCUSSION

The 98.0% black-box success rate obtained in this study is consistent with, and slightly higher than, the results reported by Subari et al. (2021) for an MVC-based travel-warrant information system and by Gustinov et al. (2023) for black-box testing of a web-based e-commerce application, supporting the view that the explicit separation of concerns in MVC reduces the number of defects that reach functional testing because each layer can be validated independently during development. Notably, both failures identified in this study originated in the View layer's client-side scripts rather than in the Model or Controller, which illustrates a practical benefit of MVC beyond defect rate: when a fault does occur, its architectural layer is immediately identifiable, which narrows the search space for debugging and reduces the risk of a fix in one layer introducing a regression in another.

The overall ISO/IEC 25010 average of 91.3% is comparable to the quality scores reported by Muhram and Mustikasari (2022) and Wijaya and Wardijono (2023) for other Indonesian information systems evaluated against the same standard, and it exceeds the 80% threshold recommended in that line of work. The relatively lower security score (87.6%) echoes a limitation noted by Wattiheluw et al. (2020) in their ISO 25010-based quality model for e-commerce websites: functional and usability characteristics tend to score highly once a system is feature-complete, while security requires dedicated testing (such as penetration testing or automated vulnerability scanning) that goes beyond functional validation and was outside the scope of this study's black-box test scenarios.

The performance results reinforce the maintainability argument made in Section 1: because the Model layer centralizes database access instead of duplicating queries across pages, the MVC-based prototype scaled more gracefully under concurrent load than the monolithic baseline, with the performance gap widening from near parity at 10 users to a 48.4% difference at 100 users. This is a practical, business-relevant finding for e-commerce SMEs, since checkout and dashboard pages are exactly the pages that experience concurrent load during promotional periods. Combined with the CRM feature findings of Al Mudzakir and Bakar (2020), Al Farisi and Purwanto (2022), and Br Purba and Harahap (2024), which link structured customer-data handling to improved retention and purchase interest, the results suggest that the architectural choice (MVC) and the functional choice (CRM feature set) are complementary: MVC provides the maintainability and performance needed to keep a growing CRM system reliable, while the CRM features provide the business value that justifies building the system in the first place.

This study has several limitations. The prototype was evaluated with a single load-testing scenario and a small usability panel of five users, which limits the statistical generalizability of the performance and



usability scores; a larger and more diverse user sample, together with automated load-testing tools, would strengthen future evaluations. Security was assessed only through functional black-box scenarios rather than dedicated penetration testing, consistent with the lower score obtained for that characteristic. Finally, the system was implemented and evaluated for a single business domain (general e-commerce retail); applying the same MVC-based reference architecture to other domains, such as business-to-business (B2B) commerce or subscription-based services, would help confirm the generality of the findings.

CONCLUSIONS

This study designed and implemented a web-based CRM information system for e-commerce SMEs using the Model-View-Controller architecture, and evaluated it through black-box functional testing, ISO/IEC 25010:2023 quality assessment, and a concurrent-load performance test. The system achieved a 98.0% black-box testing success rate across 100 test scenarios covering seven functional modules, an average ISO/IEC 25010 quality score of 91.3% with every characteristic above the 80% acceptance threshold, and a substantially lower average response time than a non-MVC monolithic baseline under increasing concurrent load, with the gap reaching 48.4% at 100 concurrent users. These results support the conclusion that separating a CRM system's Model, View, and Controller layers improves functional reliability, isolates defects to a single architectural layer, and yields better performance stability as user load grows, making MVC a practical architectural choice for e-commerce SMEs that need a CRM platform they can maintain and scale over time. Future work should extend the evaluation to dedicated security testing, a larger usability sample, and additional business domains to further validate the generality of the proposed reference architecture.

REFERENCES

- Al Farisi, M. F., & Purwanto, A. (2022). Penerapan metode CRM pada perancangan sistem informasi penjualan pakaian dan aksesoris. *JURIKOM (Jurnal Riset Komputer)*, 9(4), 902–908. <https://doi.org/10.30865/jurikom.v9i4.4647>
- Al Mudzakir, T., & Bakar, A. (2020). Desain dan implementasi customer relationship management berbasis web (studi kasus: Toko Baju Ladya). *Systematics*, 2(1), 1–8. <https://doi.org/10.35706/sys.v2i1.3444>
- Alshurideh, M. (2023). Does social customer relationship management (SCRM) affect customers' happiness and retention? A service perspective. *Uncertain Supply Chain Management*, 11(2), 581–588. <https://doi.org/10.5267/j.uscm.2022.9.015>
- Anggraeni, L. A. (2024). Tantangan kompetisi dan risiko monopoli dalam transformasi digital e-commerce dengan studi kasus aplikasi TikTok di Indonesia. *Sains: Jurnal Manajemen dan Bisnis*, 16(1), 87–98. <https://doi.org/10.35448/jmb.v16i1.23398>
- Asrul, A., Mashud, M., Qadri, M., Tamrin, F., & Ilham, A. S. (2023). Perancangan sistem e-commerce menggunakan metode customer relationship management. *Jurnal Minfo Polgan*, 12(1), 192–199. <https://doi.org/10.33395/jmp.v12i1.12343>
- Avisya, L. (2023). Rancang bangun e-commerce penjualan buah menggunakan metode customer relationship management (CRM). *ELTI: Jurnal Elektronika, Listrik dan Teknologi Informasi Terapan*, 5(2), 68–73.
- Ayunnggati, T., Lutfiani, N., & Millah, S. (2021). CRM-based e-business design (customer relationship management) case study: Shoe washing service company S-Neat-Kers. *International Journal of Cyber and IT Service Management*, 1(2), 216–225. <https://doi.org/10.34306/ijcitsm.v1i2.58>
- Basir, F. M., Guna, S. L. G., & Sitorus, S. H. S. (2023). Pengembangan website penjualan W'D Cakes & Cookies dengan integrasi CRM dalam meningkatkan loyalitas pelanggan. *Jurnal Ilmiah Sistem Informasi (JISI)*, 10(1), 44–53. <https://doi.org/10.33510/jisi.v10i1.189>



This work is licensed under a [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/)

Knowledge of Modern Processing, Utilization, and Technology for Engineering Research (KOMPUTER), Vol. 01, No. 2, August 2026

- Br Purba, Y. D., & Harahap, A. M. (2024). Customer relationship management strategy in mobile-based e-commerce platform development to increase purchase interest. *Journal of Computer Networks, Architecture and High Performance Computing*, 6(3), 1356–1367. <https://doi.org/10.47709/cnahpc.v6i3.4365>
- Del Vecchio, P., Mele, G., Siachou, E., & Schito, G. (2022). A structured literature review on big data for customer relationship management (CRM): Toward a future agenda in international marketing. *International Marketing Review*, 39(5), 1069–1092. <https://doi.org/10.1108/IMR-01-2021-0036>
- Golian, N. V., Golian, V. V., & Afanasieva, I. V. (2022). Black and white-box unit testing for web applications. *Bulletin of Kharkiv National Automobile and Highway University*.
- Gustinov, M. D., Azani, N. W., Al Ghani, R., Auliani, S. N., Maharani, S., Hamzah, M. L., & Rizki, M. (2023). Analysis of web-based e-commerce testing using black box and white box methods. *International Journal of Information System and Innovation Management (IJISIM)*, 1(1), 20–31.
- International Organization for Standardization. (2023). *ISO/IEC 25010:2023 Systems and software engineering Systems and software quality requirements and evaluation (SQuARE) Product quality model*. ISO.
- Muhram, A. R., & Mustikasari, M. (2022). Analysis of the quality of web-based online exams in the education and training agency of the Indonesian attorney general's office using ISO 25010 standards. *Indonesian Journal of Multidisciplinary Science*, 1(12), 1541–1556. <https://doi.org/10.55324/ijoms.v1i12.228>
- Pulungan, R. A. S., & Hidayanto, A. N. (2023). Evaluating e-commerce system acceptance using UAT and Likert scale analysis for sustainable business. *Journal of Theoretical and Applied Information Technology*, 101(12), 4810–4822.
- Ramadhani, D. P., Syafwan, H., & Latiffani, C. (2022). Penerapan metode E-CRM pada Toko Wati Collection. *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, 9(4), 3503–3518. <https://doi.org/10.35957/jatisi.v9i4.2839>
- Subari, A., Wardhana, R. W., Yulanda, F., & Rahman, R. (2021). Implementation of MVC (Model-View-Controller) architecture in online submission and reporting process at official travel warrant information system based on web application. *Journal of Physics: Conference Series*.
- Wattiheluw, F. H., Rochimah, S., Fatichah, C., & Abidin, K. Z. (2020). Development of a quality model based on ISO 25010 using fuzzy and PSO for e-commerce websites. In *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)* (pp. 1–4). IEEE.
- Wijaya, A. F., & Wardijono, B. A. (2023). Pengukuran kualitas aplikasi Custody berdasarkan ISO 25010 menggunakan otomatisasi pengujian blackbox. *KLIK: Kajian Ilmiah Informatika dan Komputer*, 4(2), 937–946. <https://doi.org/10.30865/klik.v4i2.1215>
- Wulandari, A. S., Saepudin, A., Kinanti, M. P., Sudesi, Z., Saifudin, A., & Yulianti, Y. (2022). Pengujian aplikasi sistem informasi akademik berbasis web menggunakan metode black box testing equivalence partitioning. *Jurnal Teknologi Sistem Informasi dan Aplikasi*, 5(2), 102–107. <https://doi.org/10.32493/jtsi.v5i2.17561>