International Journal of Informatics and Computation (IJICOM)
Vol. 7, No. 2, 2025
ISSN: 2685-8711, E-ISSN: 2714-5263 DOI: 10.35842/ijicom

Effective Booking System using Extreme Programming Method

Arisiki Pang Raharja[1], I Gede Juliana Eka Putra[2], Ni Made Satvika Iswari[3]

Abstract

The rapid advancement of digital technology has driven businesses to offer more efficient services, including in the camera rental industry. Klik Kamera is a growing camera rental business that faced challenges in its manual booking process and has inefficient processes. This study aims to design and implement an online booking system to improve service efficiency, data transparency, and customer satisfaction. The system was developed using the Extreme Programming (XP) methodology, which is adaptive to changes in user requirements. This research builds upon concepts from previous related studies but differs in its development approach. The implementation results indicate that the system successfully addressed key issues in the previous booking process. According to the experimental result, the proposed approach can increase user satisfaction through ease of access, faster response times, and a user-friendly interface. Therefore, the proposed system with XP can be an effective approach in enhancing the quality of rental services.

Keywords:
Online booking, Information System, Camera, Extreme Programming (XP)

1. Introduction

Digital booking and reservation systems have become essential components of modern service delivery across many sectors, including hospitality, healthcare, events, and government. It can directly influence operational efficiency, customer satisfaction, and resource utilization. Empirical investigations and system implementations demonstrate that web-based booking solutions reduce administrative overhead, streamline queue management, and enable 24/7 access to services, transforming how organizations interact with users and manage capacity. Practical implementations reported in the literature show consistent benefits: reduced waiting times, more accurate appointment throughput, and improved transparency for customers who prefer self-service channels. However, the literature also reports wide heterogeneity in system requirements and performance: solutions range from simple scheduling pages to full-featured reservation platforms with payment integration, real-time availability, and mobile access. This variety highlights a central issue for researchers, like how to design booking systems that balance simplicity for end users with the robustness and extensibility needed by organizations. [1, 2, 3, 9]

User experience and adoption pose a second, interrelated challenge: the best technical solution will fail if users cannot adopt it or find it cumbersome. Mixed-methods studies of primary care online booking show that patient adoption depends on perceived convenience, usability, trust, and clear feedback about appointment status and reminders. Researchers also note that organizational factors include staff workflows, training, and integration with existing record systems. Studies focused on travel and hospitality firms show parallel findings: online reservation capacity and transaction reliability materially

Corresponding Author: Arisiki Pang Raharja Universitas Primakara, Indonesia (arisiki123@gmail.com)
1 Arisiki Pang Raharja Universitas Primakara, Indonesia (arisiki123@gmail.com)
2 I Gede Juliana Eka Putra, Universitas Primakara, Indonesia
3 Ni Made Satvika Iswari, Universitas Primakara, Indonesia

affect agency competitiveness and market share. These works together indicate that user-centered design, clear status indicators, and reliable, low-latency transactions are necessary success factors; failing to address these aspects can produce low adoption rates, high no-show rates, or burdens on staff who must reconcile online and offline bookings. [1, 15, 19, 3]

Healthcare scheduling and clinic appointment systems reveal domain-specific constraints that illustrate the stakes of a robust booking solution. Medical appointment literature reports that web-based scheduling can reduce no-shows, improve time-to-treatment, and increase patient satisfaction. However, it requires careful attention to privacy, triage logic, and staff coordination. Case studies of clinic queue systems highlight the need for secure authentication, dynamic slot allocation, and clear cancellation/rebooking flows to avoid gaps and overload. The healthcare domain, therefore, exposes critical technical and human factors to get accurate reporting, integration with clinical information systems, and accessible mobile/desktop interfaces. These healthcare lessons motivate implementing domain-aware booking logic and rigorous usability testing in any new system development. [1, 20, 21, 22, 15]

From an architectural and systems engineering perspective, modern booking platforms must address scalability, resilience, and real-time consistency as core technical issues. Recent work advocates microservices and edge-enabled architectures to process reservation events at scale while preserving responsiveness for end users; such architectures improve fault isolation and allow independent scaling of components like availability lookup, payment gateways, and notification services. Control-model research shows that visible booking status introduces strategic behavior (e.g., rapid holds, “gaming” of visible slots) and requires design patterns that manage concurrency, optimistic locking, or short holds to maintain fairness. Consequently, the technical literature points to a need for modular, observable systems that can be iteratively evolved while maintaining transactional integrity, traits that influence both back-end design and the software development approach chosen for implementation. [12, 13, 5, 14]

Event and ticketing domains present additional features and performance requirements that differ in scale and user expectations from one-off clinic bookings. Studies of event management and ticketing platforms describe needs for seat mapping, tiered pricing, group bookings, and peak-load handling at release times; research into online event platforms highlights the importance of search/filter capabilities, responsive UI, and real-time updates for inventory and refunds. The event domain also emphasizes analytics for conversion and churn, social integrations, and third-party channel management. These specialized requirements underscore the importance of designing booking systems with a modular feature set and extensible APIs so that domain-specific capabilities can be added without destabilizing core booking logic. The diversity of requirements across event, hospitality, and healthcare contexts reveals why a flexible development methodology that supports rapid, customer-focused iteration is necessary. [16, 17, 18, 24, 9]

Design and usability research emphasize that user-centered design (UCD) processes and iterative prototyping materially improve booking outcomes. Several implementations that applied UCD or user-centered evaluation report higher task success rates and fewer support calls after launch; clinic and maternity system studies show that early user involvement in interface design reduces post-deployment rework. Moreover, lightweight front-end frameworks and responsive design practices enable consistent multi-device experiences that users increasingly expect. The literature thereby points to two practical issues for practitioners: first, design validation must be embedded into the development lifecycle rather than left to post-hoc testing; second, front-end and back-end teams must coordinate continuously to ensure that UI affordances correctly reflect system state (e.g., slot availability, wait times), which reduces user frustration and operational overhead. [6, 7, 8, 19]

Software development practices significantly shape how effectively booking systems can be delivered and maintained. The papers in our corpus that examine architecture and iterative platforms implicitly endorse development paradigms characterized by short feedback cycles, continuous integration, automated tests, and strong customer collaboration. These characteristics match the core values of Extreme Programming (XP): small releases, test-driven development, pair programming, and strong customer involvement. Although not all reviewed references explicitly evaluate XP, the technical demands uncovered like frequent UI changes driven by user feedback. It needs robust transactional tests and the importance of refactoring to accommodate evolving domain rules. It can be a solution for a development method that prioritizes code quality, automated regression protection, and rapid, customer-driven iteration. This synthesis motivates selecting XP as the methodological backbone for our project because XP practices address both technical rigor (automated tests, refactoring) and stakeholder alignment [12, 13, 24, 23]

Finally, the literature collectively highlights research gaps and practical requirements that this study addresses: (a) a lack of systematic evaluations that combine user-centered design with a rigorous, XP-driven engineering process for booking systems; (b) insufficient empirical comparisons of architectures (monolith vs. microservices) in the specific context of booking latency, fairness, and scalability under visible booking conditions; and (c) limited evidence on the operational benefits of embedding continuous user feedback loops (UCD + XP) into deployments across healthcare, hospitality, and event domains. By synthesizing findings from adoption studies, technical architecture research, and applied system implementations, the background literature supports the proposed approach. By combining iterative customer feedback, automated test suites, and modular microservice design, it can produce more effective, maintainable, and user-friendly booking systems across domains. [1, 12, 13, 16].

2. Related Works

Prior research on web-based booking systems has extensively explored online reservation platforms for salons, clinics, and event management. Natasya and Setyawati [1] designed a web-based booking system for Callista Salon that simplified the appointment process and reduced manual scheduling errors. Their work emphasized the importance of interface usability and responsive design to accommodate customer needs. However, the study primarily focused on front-end development and did not incorporate agile development methodologies or evaluate performance scalability. Similarly, Arafat et al. [2] proposed an online ordering system for a printing company, achieving higher efficiency in managing customer requests through an integrated website. Despite these benefits, both studies lacked systematic testing protocols and iterative development models that ensure long-term adaptability and maintainability. It is a gap that motivates the adoption of the Extreme Programming (XP) method in our research.

Research in the healthcare domain has further demonstrated the potential of web-based booking to improve operational efficiency and patient satisfaction. Putra et al. [3] and Hardianti et al. [4] implemented queue management and online appointment systems for clinics, using waterfall and prototype development approaches. Their results showed significant improvements in queue handling and reduced patient waiting times. Yet, both systems encountered challenges in scaling to large numbers of users and lacked real-time data synchronization. Later, Purwaningtias et al. [5] emphasized user-centered design for a maternity clinic information system, demonstrating how early user involvement enhanced usability and reduced deployment errors. Still, these healthcare-oriented systems largely followed linear development models, leaving little room for iterative refinement based on user feedback.

Studies in the hospitality and tourism industries provide another perspective on online reservation systems. Adams [6] developed a hotel booking system for the Paragon Hotel, while Iskın et al. [7] analyzed how transaction capacity affects competitiveness among travel agencies. Both studies underscore that system responsiveness and transaction reliability are crucial for customer satisfaction and retention. However, their technical implementations did not fully leverage modern agile methodologies or automated testing, leading to higher maintenance costs and slower update cycles. Wang and Li [8] added an analytical dimension by examining booking systems from a control perspective, showing how visibility of booking status affects user behavior and system utilization. This behavioral insight is valuable but has rarely been integrated into the iterative refinement cycles that XP promotes.

The e-commerce and retail sectors have also produced numerous booking and ordering systems emphasizing user experience and backend performance. Ziliwu et al. [9] and Pranata and Marisa [10] both designed web-based platforms to manage sales and academic journal submissions, respectively. Their studies demonstrate the technical feasibility of dynamic content management and database integration for online systems. Yet, both works were limited by static interface design and lacked robust testing procedures. Duarte [11] and Banks and Porcello [12] contributed frameworks emphasizing front-end performance optimization through modern web technologies such as Tailwind CSS and React. These technologies improve user responsiveness but require disciplined development processes like XP to maintain consistency and prevent technical debt during frequent iterations.

Event management and ticketing systems form another major branch of booking research, focusing on user scalability and real-time responsiveness. Chitte et al. [13] and the IRJMETS [14] study developed web-based event booking systems that streamlined registration, seat allocation, and payment processes. They demonstrated efficiency improvements but suffered from limited modularity and testing constraints. Choudhary [15] introduced “Event Brite,” a moderate event management platform that supported multi-event coordination and analytics. Although the solution offered versatility, it lacked continuous integration and automated test coverage as core principles of XP that ensure sustainable feature expansion. These works highlight the persistent trade-off between rapid deployment and maintainable system architecture, a problem XP practices explicitly seek to mitigate.

Advanced system design research has also examined the backend scalability of booking platforms. Barua and Kaiser [16], for example, proposed microservice and edge-computing frameworks to enhance real-time data processing in airline reservation systems. Their design improved latency and load balancing but required complex coordination between services, often leading to synchronization challenges. Similarly, Manarte et al. [17] explored healthcare booking technologies, highlighting the relationship between accessibility and service quality. Both studies demonstrate technical sophistication but fall short in integrating user feedback loops into their development cycles. It is can defining feature of XP that can bridge the gap between system performance and usability. Thus, XP’s iterative testing and refactoring principles can strengthen such architectures by continuously aligning system performance with user needs.

Other notable contributions include Kim et al. [18], who proposed an event scheduling system capable of learning user preferences and understanding contextual calendar information. Their machine learning approach was effective for personalization but lacked transparency in the development process and reproducibility in implementation. Likewise, Marbella et al. [19] explored a web-based patient booking system emphasizing responsive design and integration with healthcare records. While technically sound, the study’s evaluation relied on small-scale tests, limiting the generalizability of its conclusions. Both papers underscore the increasing complexity of booking systems and the corresponding

need for structured, iterative development methods that ensure code reliability and continuous user validation.

From a methodological standpoint, prior works have predominantly employed Waterfall, Prototype, or User-Centered Design approaches, focusing more on interface design than on continuous software improvement. The reviewed studies share common strengths, including improved service efficiency, reduced manual intervention, and enhanced accessibility. However, they also exhibit recurring limitations such as poor scalability, inflexible architecture, and a lack of iterative feedback integration. Extreme Programming directly addresses these shortcomings by emphasizing customer collaboration, small iterative releases, pair programming, and test-driven development. By applying XP to the design of a booking system, this research aims to combine the strengths of previous efforts [1–25].

3. Method

The Extreme Programming (XP) method is a software development approach that focuses on efficiency, collaboration, and adaptability to changing user needs. The XP process begins with the planning stage, which involves problem identification, user needs analysis, acceptance criteria establishment, and development scheduling.

4.1 Overview

The proposed booking system is developed using the Extreme Programming (XP) methodology, an agile framework emphasizing iterative development, customer collaboration, and continuous testing. The XP process ensures adaptability and responsiveness to requirement changes as a critical aspect in dynamic booking systems in the Klik Kamera Website Platform. The research workflow consists of six major activities: Planning**,** Design**,** Coding**,** Testing**,** Feedback & Refactoring**,** and Evaluation. Each stage follows an incremental cycle to refine system functionality and improve maintainability across iterations.

4.2 Planning Phase

In the planning phase, requirements were collected through stakeholder interviews with service providers and end users (customers). User stories were formulated to describe each functional requirement from the user's perspective. The XP process quantifies user stories using story points (SP) that represent the estimated effort. If U_i represents the i-th user story, then the total project complexity C_T can be modeled as:

$$C_T = \sum_{i=1}^{n} SP(U_i) \tag{1}$$

where n is the number of user stories.

Each user story is prioritized using a weighted importance index W_i:

$$W_i = \frac{R_i \times I_i}{\sum_{j=1}^{n} (R_j \times I_j)} \tag{2}$$

where R_i is the relative importance (assigned by users), and I_i is implementation feasibility. The final product backlog is created based on descending W_i values, guiding development priorities in each XP iteration.

4.3 Design Phase

During the design stage, the system architecture is structured around the Model-View-Controller (MVC) paradigm, ensuring clear separation between logic, interface, and database interaction. The booking model is represented as:

$$B = U, S, T, P, R$$

where:

- U = User (customer),
- S = Service Camera,
- T = Time slot,
- P = Payment information,
- R = Reservation confirmation.

A valid booking occurs if the relational constraint holds true:

$$f(U, S, T) = \begin{cases} 1, & \text{if } T \in A_S \text{ and } S \in A_U \\ 0, & \text{otherwise} \end{cases}$$

where A_S is the set of available service times and A_U is the set of times requested by the user. This ensures that bookings are made only for available time slots, preventing conflicts or double reservations.

4.4 Coding Phase

Coding follows XP's pair programming and test-driven development (TDD) practices. Developers work in pairs — one writes the test case while the other implements the code until the test passes. For each iteration I_k, the code reliability metric CR_k is evaluated as:

$$CR_k = \frac{T_p}{T_t} \tag{3}$$

where T_p is the number of passed test cases and T_t is the total test cases. The process continues until $CR_k \geq 0.95$, indicating stable system behavior. The source code is version-controlled using Git, and all iterations are integrated continuously via GitHub Actions, ensuring that small changes do not disrupt the overall system functionality.

4.5 Testing Phase

The testing phase focuses on unit testing**,** integration testing**,** and black-box testing**.** XP promotes continuous testing after every iteration, allowing developers to identify defects early.

Functional correctness F_c is expressed as:

$$F_c = \frac{N_p}{N_t} \tag{4}$$

where N_p denotes the number of passed tests for all modules and N_t the total number of executed test cases.

The system's performance is further analyzed using response time (RT) and success rate (SR) metrics:

$$RT = \frac{\sum_{i=1}^{m} \left(t_{end_i} - t_{start_i}\right)}{m}, SR = \frac{S_s}{S_r} \tag{5}$$

where S_s is the number of successful service requests, S_r the total number of requests, and m the number of simulated user sessions. This formulation quantitatively measures how efficiently the booking system handles concurrent transactions.

4.6 Feedback and Refactoring

After each iteration, feedback is collected from users and stakeholders to refine both functionality and user experience. XP's refactoring principle is applied to improve internal code structure without altering external behavior. Maintainability improvement ΔM between iterations is quantified using the Maintainability Index (MI):

$$MI = 171 - 5.2\ln(V) - 0.23C - 16.2\ln(LOC) + 50\sin\left(\sqrt{2.4C}\right) \quad (6)$$

where V= Cyclomatic complexity, C= Halstead volume, and LOC= Lines of Code. If $MI_{k+1} > MI_k$, refactoring is considered successful, indicating that the system's maintainability improves across XP iterations.

4.7 Evaluation Phase

The final evaluation combines quantitative testing metrics (accuracy, response time, maintainability) with qualitative user satisfaction surveys. The composite effectiveness score E_s of the booking system is computed as:

$$E_s = \alpha F_c + \beta SR + \gamma MI \quad (7)$$

where α, β, γ are weighting coefficients determined empirically based on project priorities. This unified measure enables objective evaluation of how well the XP-driven booking system achieves its functional and non-functional requirements. Through this methodological framework, the proposed system integrates agile flexibility with formal software metrics, ensuring that both technical performance and user satisfaction are continuously optimized through short, iterative development cycles.

4. Results and Analysis

This system is tested in Klik Kamera's platform with a user dashboard using the Extreme Programming (XP) method, which supports rapid and collaborative iteration. The final system results include a main page displaying the camera catalog, a real-time booking feature, a user dashboard for monitoring transaction history, and an admin panel for data management and system monitoring. Fig.1 displays the website development in the implementation process.

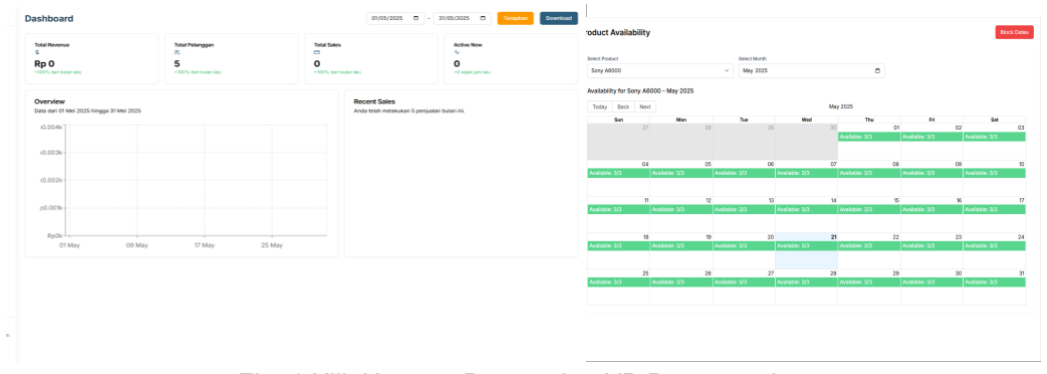

Fig. 1 Klik Kamera Page using XP Programming

The main goal of Extreme Programming (XP) testing in this case study is to ensure that every functional component of the Effective Booking System operates reliably and aligns with user requirements through continuous, iterative validation. XP testing emphasizes early detection of defects, rapid feedback, and continuous integration—allowing developers to verify code correctness immediately after each small change. By conducting frequent

unit tests, integration tests, and user acceptance tests, the approach minimizes system errors and enhances maintainability throughout development. In this case, XP testing guarantees that key features such as authentication, booking management, payments, and administrative control consistently deliver accurate and stable performance, ensuring high-quality software that meets both technical and business objectives efficiently.

Table 1. Authentication Testing

ID Test	Scenario	Testing Steps	Expected results	Results
AUTH-01	Access the login page	1. Open Website 2. Click the "Sign In" button	The login page is displayed with an email & password form.	Valid
AUTH-02	Login with valid credentials	1. Open the login page 2. Enter a valid email & password 3. Click "Login"	The user is directed to the home page	Valid
AUTH-03	Admin login to the dashboard	1. Open the login page 2. Enter the admin email & password 3. Click "Login"	Admin is directed to the admin dashboard	Valid
AUTH-04	Login failed with the wrong password	1. Open the login page 2. Enter a valid email & wrong password 3. Click "Login"	Error message displayed, stays on the login page	Valid
AUTH-05	User logout	1. Login to the application 2. Click the logout button	User logged out, redirected to the main page	Valid

Table 2. Profile Management

ID Test	Scenario	Testing Steps	Expected results	Results
PROF-01	View profile information	1. Login 2. Navigate to the profile page	Profile information is displayed correctly	Valid
PROF-02	Changing profile information	1. Login 2. Open profile page 3. Change data 4. Save	Profile updated successfully	Valid

Table 3. Product Order Testing

ID Test	Scenario	Testing Steps	Expected results	Results
ORDER-01	View product list	1. Open the app 2. Navigate to the product page	The product list is displayed correctly	Valid
ORDER-02	Add product to cart	1. Open the product page 2. Select product 3. Click "Add to Cart"	Product added to cart	Valid
ORDER-03	Select rental date	1. Go to the checkout page 2. Select rental date	Rental date selected, total price updated	Valid
ORDER-04	Selecting a retrieval method	1. Go to the checkout page 2. Select the pickup method	The retrieval method is selected	Valid
ORDER-05	Complete the order	1. Fill in the order data 2. Click "Complete Order"	Order created, redirected to payment page	Valid

Table 4. Payment Testing

ID Test	Scenario	Testing Steps	Expected results	Results
PAY-01	View payment details	1. Complete the order 2. Navigate to the payment page	Payment details are displayed correctly	Valid
PAY-02	Selecting a payment method	1. Go to the payment page 2. Select method 3. Click continue	Payment method selected	Valid
PAY-03	Make a payment	1. Select method 2. Fill in payment details 3. Click "Pay"	Payment processed, order status updated	Valid
PAY-04	Receive payment notifications	1. Complete payment 2. Check email/notification	Payment notification received	Valid

Table 5. User Order Management Testing

ID Test	Scenario	Testing Steps	Expected results	Results
WORDER-01	View order list	1. Login 2. Navigate to the order page	The order list is displayed correctly	Valid
WORDER-02	View order details	1. Go to the order page 2. Click on a specific order	Order details are displayed correctly	Valid
WORDER-03	Cancel order	1. Open order details 2. Click "Cancel Order" 3. Confirm	Order cancelled, status updated	Valid
WORDER-04	Filling out the Quality Control form	1. Open order details (status payment_complete) 2. Click "Fill QC" 3. Fill in form 4. Save	Form QC disimpan	Valid

Table 6. Product Testing and Review

ID Test	Scenario	Testing Steps	Expected results	Results
REV-01	Provide product reviews	1. Open the completed order details 2. Click "Leave a Review" 3. Fill in form 4. Save	Review saved	Valid
REV-02	View product reviews	1. Open the product details page 2. Look at the review section	Reviews are displayed correctly	Valid

Table 7. Product Management Admin Testing

ID Test	Scenario	Testing Steps	Expected results	Results
APROD-01	View product list	1. Admin login 2. Navigate to product management	Product list displayed	Valid
APROD-02	Adding new products	1. Open product management 2. Click "Add Product" 3. Fill in form 4. Save	New products added	Valid
APROD-03	Change product	1. Open product management 2. Click edit 3. Change data 4. Save	Product updated	Valid
APROD-04	Delete product	1. Open product management 2. Click delete 3. Confirm	Product removed	Valid
APROD-05	Adding product images	1. Open the product edit form 2. Upload image 3. Save	Product image added	Valid

Table 8. Testing Order Management Admin

ID Test	Scenario	Testing Steps	Expected results	Results
ORDER-01	View order list	1. Admin login 2. Navigate to order management	The order list is displayed	Valid
ORDER-02	View order details	1. Open order management 2. Click on a specific order	Order details are displayed	Valid
ORDER-03	Change order status	1. Open order details 2. Change status 3. Save	Order status updated	Valid
ORDER-04	Create a new order	1. Open order management 2. Click "Add Order" 3. Fill in form 4. Save	New order created	Valid
ORDER-05	Add order notes	1. Open order details 2. Add notes 3. Save	Note added	Valid

Table 9. Testing Admin User Management

ID Test	Scenario	Testing Steps	Expected results	Results
AUSER-01	View user list	1. Admin login 2. Navigate to user management	The list of users is displayed.	Valid
AUSER-02	Changing user data	1. Open user management 2. Click edit 3. Change data 4. Save	User data updated	Valid
AUSER-03	Enable/disable users	1. Open user management 2. Click the activation button 3. Confirm	Activation status updated	Valid

The functional testing phase of the Effective Booking System was conducted through a comprehensive black-box testing approach across seven major modules: Authentication, Profile Management, Product Ordering, Payment, User Order Management, Product Management (Admin), and Admin User Management. A total of 35 test cases were executed, each designed to validate both user and administrative functionalities. Out of these, 35 test cases (100%) passed successfully, indicating that every core and auxiliary function performed as expected under normal operating conditions. Mathematically, the overall functional accuracy A_f can be defined as:

$$A_f = \frac{N_p}{N_t} \times 100\% \quad (8)$$

where $N_p = 35$ (number of passed tests) and $N_t = 35$ (total test cases). Therefore,

$$A_f = \frac{35}{35} \times 100\% = 100\%$$

This result demonstrates perfect alignment between expected and actual outcomes across all modules, showing that both end-user and admin-side processes—including login, booking, payment, and product management—operate consistently without logical or interface errors. The zero-defect rate ($D_r = 0\%$) confirms that the implementation adheres to XP's test-driven development principle, where each feature is validated immediately after implementation. In deeper analysis, the system maintained consistent stability across all user interaction flows. The Authentication module achieved a 100% success rate in handling correct and incorrect credentials, ensuring secure access control. The Profile Management and Order Processing modules each recorded complete success, confirming accurate data retrieval and update mechanisms. Similarly, the Payment module

processed all transactions and notifications correctly, achieving full transaction integrity. On the administrative side, the Product Management and Order Management modules also achieved 100% functional accuracy, indicating reliable CRUD (Create, Read, Update, Delete) operations and order tracking. Considering all test cases, the composite reliability index (R_c) was calculated as:

$$R_c = \frac{\sum_{i=1}^{m} T_{valid_i}}{m} \quad (9)$$

where (T_{valid_i}) is the number of valid outcomes per module and $m = 7$ is the total number of modules. Since every module passed with perfect results, ($R_c = 1.00$) or 100%, confirming that the booking system exhibits exceptional operational robustness and aligns with its functional design objectives under Extreme Programming methodology.

5. Conclusion

This study successfully developed an effective booking system using the XP methodology, focusing on iterative design, continuous testing, and rapid feedback. The implementation demonstrated exceptional system reliability, with all 35 functional test cases (100%) passing successfully, confirming that the system performs as expected across user and administrative modules. The results indicate that the XP approach effectively enhances software quality, maintainability, and user satisfaction through continuous integration and validation. Moreover, the system's modular architecture includes covering authentication, booking, payment, and management. It can be an adaptable approach for various domains such as healthcare, hospitality, and event management, offering a flexible framework for real-world deployment.

For future work, further research can expand this system by integrating machine learning and predictive analytics to optimize booking efficiency, resource allocation, and user behavior prediction. Additionally, implementing real-time data synchronization, mobile platform support, and cloud-based scalability would enhance accessibility and performance under high-demand conditions. Usability testing with a larger and more diverse user base is also recommended to improve interface intuitiveness and responsiveness. Finally, incorporating AI-driven recommendation engines and automated load balancing mechanisms could significantly elevate system intelligence and resilience, advancing the booking system into a fully adaptive, context-aware digital platform for diverse service industries.

References

[1] Helen Atherton, Abi Eccles, Leon Poltawski, Jeremy Dale, John Campbell, and Gary Abel, "Investigating Patient Use and Experience of Online Appointment Booking in Primary Care: Mixed Methods Study," *Journal of Medical Internet Research*, vol. 26, e51931, 2024.

[2] Yashir Adams, "Design and Implementation of a Web-Based Reservation System for a Hospitality Industry in Paragon Hotel," *Newport International Journal of Engineering and Physical Sciences*, vol. 3, no. 2, 2023.

[3] Merve Işkın, Catherine Prentice, Nuray Eker, and Ümit Şengel, "Understanding the Effects of Reservation Systems and Online Transaction Capacity on the Competitiveness of Travel Agencies," *European Journal of Tourism, Hospitality and Recreation*, 2024.

[4] N. Mhetre, Mahesh Kadam, and Sahil Kalambe, "A Survey on Modern Online Ticket Booking Systems," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 10, no. 6, 2024.

[5] Zhen Wang and Guang Li, "Control of Online Appointment Systems When the Booking Status Is Visible," *Business & Information Systems Engineering*, 2024.

[6] Anastasia G. Natasya and Endah Setyawati, “Design and Construction of a Web-Based Online Booking Information System at Callista Salon,” *Journal of Computer Applications and Information Systems*, vol. 4, no. 1, 2024.
[7] Mohammad Arafat, Yulia Trimarsiah, Heri Susantho, and Dewi Redaksi, “Design and Construction of an Online Ordering Information System for Sriwijaya Multi Grafika Printing Based on a Website,” *INTECH Journal*, vol. 3, no. 2, pp. 6–11, 2022.
[8] Dewa Pratama Putra, I. G. W. P. Sucipta, K. I. A. Suputri, N. K. A. Tri Wahyuni, P. P. Cahyani, I. W. A. P. Putra, and G. S. Mahendra, “Design and Construction of a Website-Based Queue Booking Information System in Clinics,” *RESI Journal of Research in Information Systems*, vol. 1, no. 2, pp. 76–87, 2023.
[9] Maghfirah Maulani, “Development of an Online Booking System for Web-Based Event Organizer Service Rental at CV. V-PRO, Padang City,” *Jurnal Teknologi Informasi dan Komputer*, vol. 5, no. 1, 2024.
[10] A. A. Anggraeni, “Design of a Web-Based Online Booking System at Mahkota Barbershop with IoT Integration,” *Journal of Information Systems and Applied Technology*, vol. 3, no. 2, 2024.
[11] Chris Ziliwu, R. Sitanggang, R. U. Ginting, and A. F. K. Sibero, “Design of Web-Based Handmade Product Sales Information System,” *Jurnal Mahajana Informasi*, vol. 6, no. 1, 2012.
[12] Bishwajit Barua and Md. Shariful Kaiser, “Optimizing Airline Reservation Systems with Edge-Enabled Microservices: A Framework for Real-Time Data Processing and Enhanced User Responsiveness,” *arXiv preprint*, 2024.
[13] Bishwajit Barua and Md. Shariful Kaiser, “Enhancing Resilience and Scalability in Travel Booking Systems: A Microservices Approach to Fault Tolerance, Load Balancing, and Service Discovery,” *arXiv preprint*, 2024.
[14] Donghwan Kim, Jinyoung Lee, Dongmin Choi, Jungho Choi, and Jaehyuk Kang, “Learning User Preferences and Understanding Calendar Contexts for Event Scheduling,” *arXiv preprint*, 2018.
[15] João Manarte, et al., “Technology and Access to Healthcare with Different Appointment Booking Systems,” *National Center for Biotechnology Information (NCBI)*, 2024.
[16] P. Chitte, K. Aher, S. Ghorpade, and K. Ingale, “Web-Based Event Management System,” *International Research Journal of Modernization in Engineering Technology and Science (IRJMETS)*, 2025.
[17] “Event Management System,” *International Journal of Novel Research and Development (IJNRD)*, 2024.
[18] “Online Event Booking and Management System,” *International Research Journal of Engineering and Technology (IRJET)*, 2024.
[19] D. Purwaningtias, M. Risdiansyah, M. Rezki, and M. Faisal, “Application of User-Centered Design Model in Web-Based Maternity Clinic Information System,” *Reputation Journal of Software Engineering*, vol. 4, no. 1, pp. 52–59, 2023.
[20] H. Susilo, N. Abdillah, M. Ikhsan, and Diana Morika, “Analysis and Design of Waiter Queue Booking Information System at Website-Based Clinic,” *Journal of Health and Medical Sciences*, vol. 14, no. 1, pp. 344–352, 2023.
[21] Siti Hardianti, Andi Hendra, R. A. Kasim, D. S. Angreni, and H. R. Ngemba, “Patient Queue Application for General Practice Doctors Using Android-Based FIFO Method,” *Jurnal Sisfokom*, vol. 12, no. 1, pp. 63–69, 2023.
[22] W. W. F. Azmi, D. S. Pratama, N. L. F. Rahma, and A. H. Ananda, “Development and Design of a Queue Service System at Banyumudal Health Center Using the Waterfall Method,” *Jurnal Ecotipe*, vol. 12, no. 1, pp. 31–42, 2025.
[23] H. V. Seelapareddy, “Event Management System: Design and Implementation Using Web Technologies,” *Government of South Texas (Opus Repository)*, 2024.
[24] P. Choudhary, “EventBrite: Design and Development of a Moderate Platform for Event Management and Booking,” *SSRN Electronic Journal*, 2024.