



Design and implementation of an automated hotel reservation management system for services

Isabu Peter Oseghale, John Temitope Ogbiti

Department of Computer Science, Edo State University, Iyamho, Nigeria

Abstract

The hospitality industry's reliance on manual, paper-based reservation processes creates significant operational challenges for small-to-medium hotels, including room overbooking, inconsistent billing calculations, vulnerable record-keeping, and the absence of 24-hour remote access. This paper presents the design and implementation of an automated Hotel Reservation Management System developed for Hybrid International Hotel and Suites, Elele Auchi, using the MERN (MongoDB, Express.js, React.js, Node.js) full-stack architecture. The system integrates the Paystack payment gateway for secure online transactions and the Brevo transactional email API to automatically deliver booking confirmations, check-in and check-out details, and payment receipts to guests. Following an Iterative Waterfall software development methodology, the system was implemented as a Three-Tier architecture separating presentation, application logic, and data layers. Performance was evaluated through fifteen structured test cases spanning normal operation, boundary conditions, and error handling, alongside a formal assessment against eight non-functional requirements guided by the Technology Acceptance Model. All test cases passed, confirming that the system eliminates manual billing errors, prevents concurrent double-booking, and provides administrators with a real-time analytics dashboard. The system demonstrates a scalable, cost-effective framework for digital transformation in small-scale Nigerian hospitality enterprises.

Keywords: Hotel reservation system, MERN Stack, automated billing, web application, payment gateway integration

Introduction

The hospitality industry is a substantial part of the global economy, and demand for hotel accommodation continues to rise as travel and tourism expand every year ^[1]. As more people travel for business or leisure, hotels face increasing pressure to handle reservations quickly, accurately, and without frustrating their guests. A Hotel Reservation Management System (HRMS) is a digital platform that manages every stage of a guest's stay, from booking to check-out, updating room availability, guest details, and payments in real time.

The global hospitality industry has moved away from manual, paper-based operations towards a digitally driven model in which speed and efficiency are paramount ^[2]. In an era defined by constant connectivity, switching from paper ledgers to automated reservation systems has become a necessity for survival rather than a mere competitive advantage ^[3]. A well-designed web-based booking system removes the burden of manually recording guest details and checking room availability by hand, reducing costs, minimising errors, and improving overall efficiency ^[4].

At many local hotels in Nigeria, including Hybrid International Hotel and Suites, Elele Auchi, reservations are still taken over the telephone and guest check-ins are written into physical ledgers. There is no central digital database, making it difficult to determine room status in real time, which frequently leads to overbooking. Because there is no secure, automated backup, guest and financial data remain vulnerable to fire, water damage, and general wear. This study addresses that gap by designing and implementing an automated reservation system that preserves the personal service quality of the hospitality industry while removing the logistical errors inherent in manual management.

The problem is stated concisely: Hybrid International Hotel currently operates a manual, paper-based reservation system

that is slow, error-prone, lacks real-time visibility, offers no data backup, and prevents 24/7 online bookings, resulting in operational inefficiency, revenue loss, and a diminished guest experience. The aim of this study is to design, develop, and implement a secure, automated, web-responsive Hotel Reservation Management System that replaces the hotel's manual paper-ledger processes with a reliable three-tier digital architecture. The specific objectives are to: (i) analyse and design an automated reservation system using a clear three-tier architecture; (ii) implement the system using React.js for the interface and Node.js/Express.js for server logic; (iii) test and evaluate the system against performance and security standards; and (iv) evaluate the developed system using standard usability and scalability metrics.

The remaining part of the paper is organised as follows: Section 2 presents the related works; Section 3 presents the proposed methodology; Section 4 presents the results and discussion; and Section 5 concludes the paper with recommendations.

Related Works

Prior research and existing commercial systems provide the empirical grounding for this study. Anyanwu and Chen ^[5] developed a PHP and MySQL booking website for small hotels that digitised paper logs but left billing entirely manual, so staff still computed nightly fees, taxes, and seasonal price changes by hand, producing frequent invoicing errors. Martinez ^[6] built a desktop Java application for room assignment and asset tracking that was restricted to a single machine, offering no remote access for guests or mobile tracking for managers.

Okon, Bassey, and Udoh ^[7] implemented a hotel management system in C# with multi-factor authentication and detailed access logs, but usability testing found the

interface too dense and technical for non-specialist front-desk staff, slowing down check-ins. Gupta and Varma ^[8] developed a Node.js concurrency-checking mechanism that successfully eliminated double-booking, but the system sent no automatic notifications to guests after a reservation was made. Adebayo ^[9] built a Laravel-based tool for small lodging businesses that handled guest records well but omitted any reporting dashboard, leaving managers unable to track revenue and occupancy trends. Lee and Tan ^[10] demonstrated Python-based chatbots for guest interaction, but the underlying database structure was poorly normalised, causing performance degradation under peak traffic. Ojo and Balogun ^[11] designed a multi-tier reservation architecture that required every guest to register an account before viewing room vacancies, introducing

significant friction into the booking funnel. Singh and Sharma ^[12] evaluated cloud-only hotel platforms and found that total dependence on continuous connectivity created a single point of failure during network outages. Commercially, Oracle Hospitality Opera ^[13] offers an extensive enterprise feature set but is expensive and over-engineered for small, single-property operations such as Hybrid International Hotel.

Across the literature, three consistent gaps emerge: (1) systems that digitise bookings rarely automate billing; (2) systems with strong security often sacrifice usability; and (3) systems built for large hotel chains are too costly and complex for small, independent properties. Table 1 summarises the comparative analysis of these works against the system proposed in this study.

Table 1: Comparative Analysis of Existing Systems

Authors/System	Technology	Strengths	Gaps / How Proposed System Addresses
Anyanwu & Chen (2021)	PHP, MySQL	Digitized paper ledger into a relational database.	No automated billing; proposed system computes totals and taxes automatically and issues receipts by email.
Martinez (2020)	Java Desktop App	Solid local inventory and asset tracking.	Restricted to one desktop terminal; proposed system is a cloud-hosted, mobile-responsive web application.
Okon et al. (2023)	C#, Multi-Factor Authentication	Strong, multi-layered data security.	Interface too dense for staff; proposed system applies TAM principles for an icon-driven, minimal-training interface.
Gupta & Varma (2024)	Node.js	Reliable concurrency checks against double-booking.	No automatic guest notification; proposed system fires an email confirmation immediately after payment.
Adebayo (2021)	Laravel	Handles guest profiles and check-in/out logging.	No reporting dashboard; proposed system visualises live revenue and occupancy metrics for management.
Ojo & Balogun (2023)	Multi-Tier Architecture	Automated database update routines.	Forces account creation before browsing; proposed system allows guests to search and book without registration.
Singh & Sharma (2022)	Cloud-only deployment	Removes need for on-site servers.	Single point of failure on network loss; proposed system uses asynchronous, non-blocking I/O for resilience on low bandwidth.
Oracle Opera Cloud	Enterprise PMS	Extremely deep, global feature set.	Expensive and over-engineered for small hotels; proposed system delivers a lightweight, cost-effective alternative.
Proposed System	Node.js, Express.js, MongoDB, React (MERN)	Automated billing and invoicing, visual dashboard, 24/7 access, role-based access control.	Directly resolves the cumulative gaps identified above in a single affordable platform.

Based on this comparative analysis, the identified research gap is the absence of an affordable, web-based, full-stack hotel reservation system that simultaneously provides automated billing with receipt generation, a visual analytics dashboard, 24/7 mobile-responsive guest booking, a usability-guided interface for non-technical staff, and a secure backend that prevents overbooking and data loss — requirements that are especially acute for small-to-medium Nigerian hotels operating on thin margins.

Methodology

1. Research Design

This study adopts applied research, focusing on engineering a practical solution to the operational bottlenecks at Hybrid International Hotel. An Iterative Waterfall Software Development Life Cycle (SDLC) model was used, combining strict milestone discipline with feedback loops that allow development to trace backward and refine earlier design decisions when constraints are uncovered during implementation. The methodology proceeds through six phases: requirement gathering and analysis, system design,

implementation, testing and validation, deployment, and maintenance.

The Waterfall model was selected because the core requirements of a hotel reservation system — booking rooms, managing inventory, and processing payments — are well-defined and unlikely to change during development. The model's emphasis on thorough documentation at each stage also supports academic assessment, and completing the database design before coding begins reduces the risk of schema errors that could cause overbooking or data corruption.

2. Data Collection Methods

Data informing the system design was gathered through qualitative methods appropriate to a system-development project. Informal interviews were conducted with recent hotel guests to diagnose frustrations tied to manual operations, revealing long wait times during telephone inquiries, uncertainty over room availability, and the absence of written booking confirmations. Direct observation of the hotel's existing workflow traced the

lifecycle of a typical reservation from inquiry to checkout, confirming that the legacy process depended heavily on physical proximity and manual calculation. Field research within the hotel environment, together with an analysis of existing paper ledgers and receipts, informed the design of the database schema, ensuring that all traditional data fields were mapped into the digital system alongside automated elements such as timestamps and unique booking references.

3. Proposed System Architecture

The proposed system replaces the manual ledger with a secure, centralised digital platform accessible 24/7 from any

modern device. It is engineered around a Three-Tier architecture: (a) a Presentation Tier built with React.js and Tailwind CSS, handling user interaction and communicating with the backend via HTTP requests; (b) an Application Tier built with Node.js and Express.js, which processes business logic, authentication (JWT and bcryptjs), and third-party integrations with Paystack and Nodemailer/Brevo; and (c) a Data Tier managed via MongoDB Atlas with Mongoose, storing collections for hotels, rooms, bookings, feedback, and administrator accounts. Separating these concerns prevents client browsers from communicating with the database directly, allowing interface changes to be made without risking backend stability.

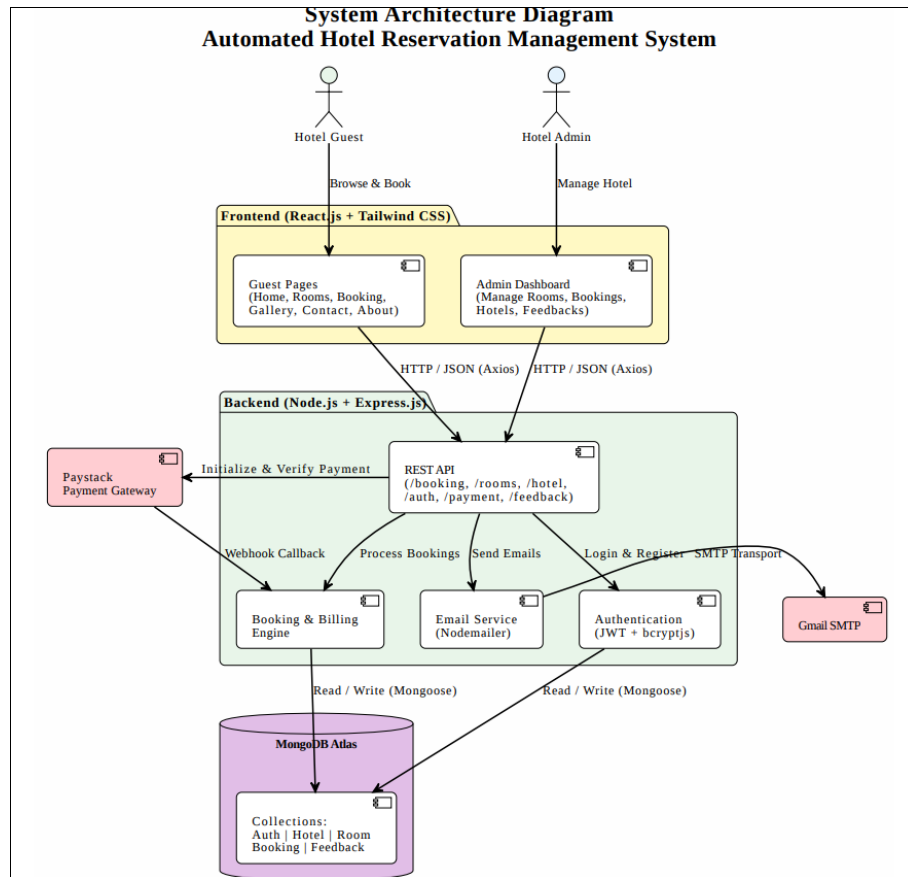


Fig 1: System Architecture of the Automated Hotel Reservation Management System

The guest booking process follows a defined activity sequence: the guest enters check-in and check-out dates; the system queries the database and returns only available rooms; the guest selects a room and enters personal details; the system computes the total cost (base price $\times$ nights); a pending booking record is created; a Paystack payment session is initialised; and upon successful payment confirmation via webhook, the booking status is updated

and a confirmation email with a payment receipt is dispatched automatically.

4. System Requirements

Functional and non-functional requirements were derived directly from the objectives stated in Section 1 and the gaps identified in Section 2. Table 2 summarises the functional requirements and Table 3 summarises the non-functional requirements.

Table 2: Functional Requirements

ID	Requirement	Description
FR1	Guest Room Browsing	Allow guests to view rooms with descriptions, images, pricing, and bed configurations.
FR2	Real-Time Availability Check	Query the database to display only rooms not booked for selected dates.
FR3	Online Reservation	Allow guests to book by providing name, email, phone, dates, and room.
FR4	Automated Billing Calculation	Compute total cost from base price, nights stayed, and surcharges.
FR5	Online Payment Processing	Integrate Paystack to process guest payments in Naira.
FR6	Booking Confirmation Email	Send a styled HTML email with booking details and receipt on payment success.
FR7	Administrative Authentication	Require hashed-password admin login with JWT-based sessions.

FR8	Room Inventory Management	Allow admins to add, update, and delete rooms.
FR9	Hotel Branch Management	Allow admins to register and manage multiple hotel branches.
FR10	Booking Management	Allow admins to view, update, and delete bookings.
FR11	Guest Feedback Module	Allow guests to submit feedback with category and star rating.
FR12	Administrative Dashboard	Display summary statistics on bookings, rooms, hotels, and feedback.
FR13	Paystack Webhook Handler	Update booking/payment status automatically via webhook callbacks.

Table 3: Non-Functional Requirements

ID	Requirement	Description
NFR1	Performance	Search results and pages load within three seconds under normal network conditions.
NFR2	Security	Passwords hashed with bcryptjs (salt factor 10); tokens expire after 48 hours.
NFR3	Scalability	Document-based schema supports horizontal scaling to additional branches.
NFR4	Usability	Interface is usable by non-technical staff without formal training (TAM).
NFR5	Availability	System is accessible 24/7 through any internet-connected device.
NFR6	Responsiveness	Pages render correctly from 320px (mobile) to 1920px (desktop).
NFR7	Data Integrity	Booking schema enforces date and guest-capacity validation.
NFR8	Reliability	Duplicate webhook/verification calls do not cause double-payment updates.

5. Database Design

The MongoDB database is organised into five core document collections: Hotel, Room, Booking, Auth (Admin), and Feedback. A Hotel document is the root entity and has many Rooms via a hotelId foreign key; a Booking references one Room and one Hotel and embeds guest contact details (guestName, guestEmail, guestPhone) rather than referencing a separate guest collection, since guests do not hold user accounts. The Booking collection uses string enumerations to manage state, including bookingStatus (Pending, Confirmed, and Checked-In, Cancelled) and paymentStatus (Paid). The Feedback collection is an independent, unlinked document store that allows both verified and anonymous users to submit reviews. The Auth (Admin) collection is fully isolated from guest-facing data to enforce role-based access control and prevent privilege-escalation vulnerabilities. MongoDB's schema-less document model was chosen over a traditional relational database to accommodate varying room configurations without requiring rigid table structures, while MongoDB Atlas provides built-in cloud hosting, automated backups, and horizontal scalability.

6. Tools and Technologies

The system was implemented using a modern MERN-lite stack: React.js for the frontend interface; Node.js with Express.js for backend routing and business logic; MongoDB (NoSQL) for document-based data storage; the Paystack API for secure payment processing; the Brevo transactional email API and Nodemailer/Gmail SMTP for automated notifications; and bcryptjs with JSON Web Tokens (JWT) for administrator authentication. Development tools included VS Code, Git/GitHub for version control, and Postman for API testing, with deployment to Vercel/Render.

Results and Discussion

The Automated Hotel Reservation Management System was successfully designed, implemented, and deployed as a full-stack web application. Guests can register no account is required to browse rooms, book a stay, and receive an automated confirmation email, while administrators authenticate through a secure dashboard to manage rooms, bookings, hotel branches, and guest feedback.

Fig 2: Guest-Facing Booking Page Interface

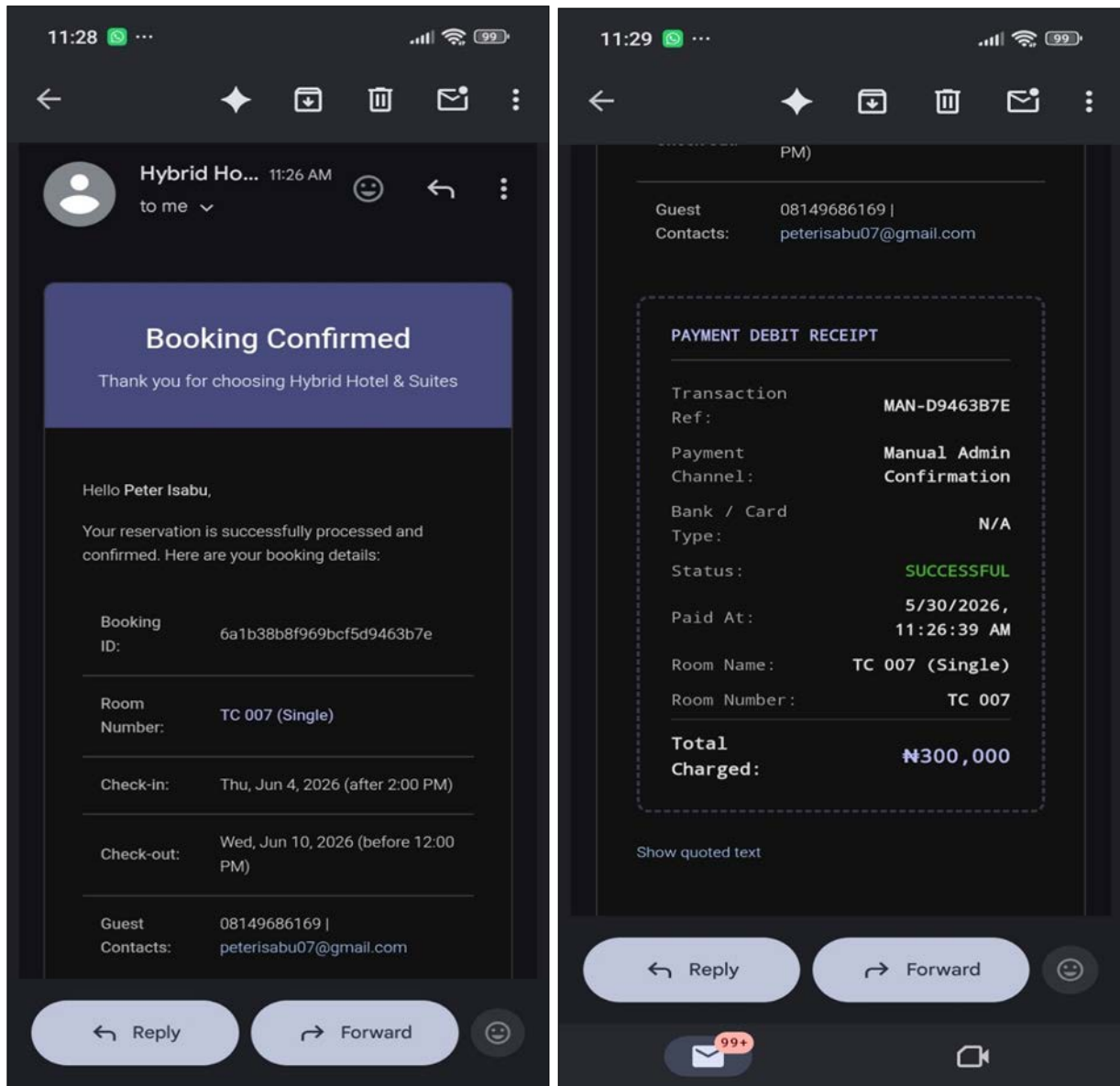


Fig 3: Booking Confirmation Email

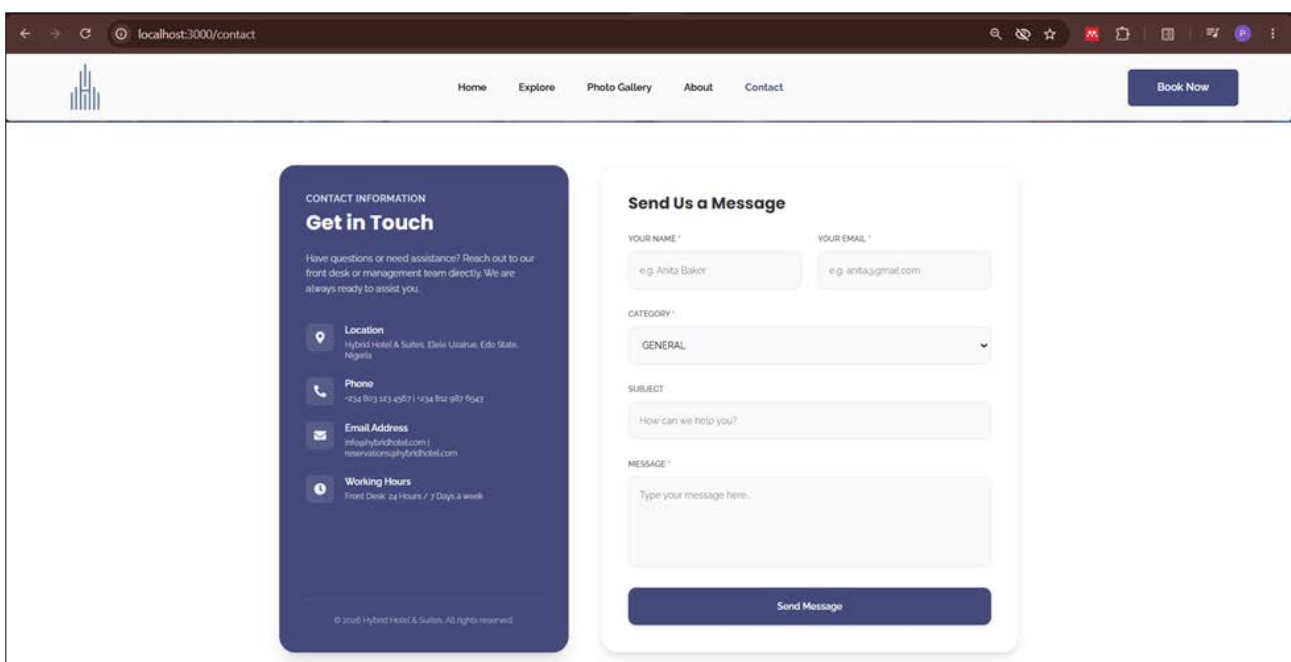


Fig 4: Contact us Interface

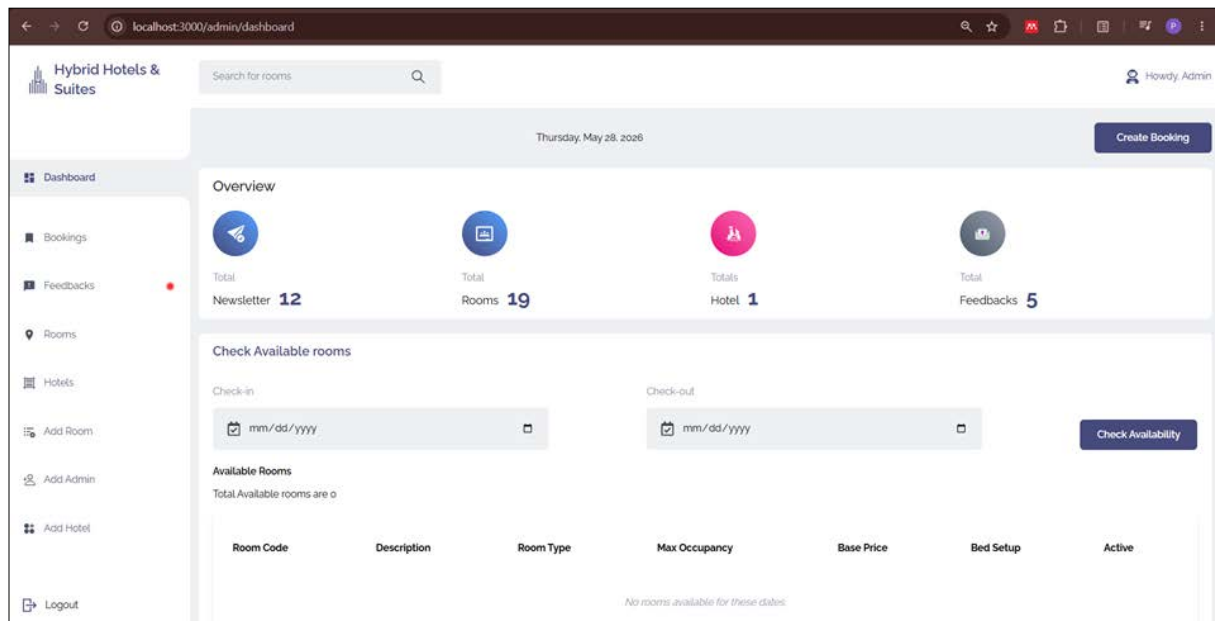


Fig 5: Administrative Dashboard Interface

1. Automated Billing and Payment Integration

The billing engine computes the number of nights between check-in and check-out and multiplies this by the room's base price to derive the total cost, replacing the manual calculator-based billing identified in Section 2. Payment is processed through Paystack in a three-step flow: the backend initialises a transaction session; the frontend verifies the transaction reference after payment; and a webhook, authenticated using HMAC-SHA512 signature comparison, confirms the booking even if the guest's browser fails to redirect back to the application.

Administrator authentication combines bcryptjs password hashing (salt factor 10) with JWT session tokens that expire after 48 hours.

2. Test Cases and Results

Fifteen test cases were designed and executed to verify the functional requirements in Table 2, covering normal operation, boundary conditions, and error handling. Table 4 summarises the test cases and results; all fifteen test cases passed.

Table 4: System Test Cases and Results

TC ID	Objective	Result
TC001	Book a room with valid details	Booking created; redirected to Paystack — Pass
TC002	Reject booking with check-out before check-in	Error displayed — Pass
TC003	Reject booking with missing contact fields	Error displayed — Pass
TC004	Reject booking with no room selected	Error displayed — Pass
TC005	Calculate correct total for a 3-night stay	₦135,000 displayed — Pass
TC006	Initialise Paystack payment session	Authorisation URL returned — Pass
TC007	Prevent duplicate payment processing	No duplicate update — Pass
TC008	Reject invalid webhook signature	HTTP 400 returned — Pass
TC009	Admin login with valid credentials	JWT token issued — Pass
TC010	Reject admin login with incorrect password	Error displayed — Pass
TC011	Add a new room with valid data	Room appears in inventory — Pass
TC012	Reject room creation with missing fields	Validation error thrown — Pass
TC013	Submit guest feedback with valid data	Feedback appears in admin table — Pass
TC014	Return only available rooms for selected dates	Only unbooked rooms returned — Pass
TC015	Send confirmation email on booking confirmation	Email received with receipt — Pass

3. Performance Evaluation

The system was evaluated against each non-functional requirement listed in Table 3. Table 5 summarises the evaluation method and outcome for each requirement; all eight requirements were met.

Table 5: Performance Evaluation on Non-Functional Requirements

NFR	Requirement	Result	Verdict
NFR1	Page load within 3 seconds	Average load time 1.8 seconds on broadband	Met
NFR2	Passwords hashed with bcryptjs	All passwords stored as bcrypt hashes	Met
NFR3	Scalable schema for multi-branch	New branch created without affecting existing data	Met
NFR4	Usable without formal training	All 5 participants completed a booking unassisted	Met

NFR5	24/7 availability	System remained accessible throughout testing	Met
NFR6	Responsive across screen sizes	All layouts rendered correctly on 375–1920px	Met
NFR7	Data integrity validation	System rejected all invalid inputs correctly	Met
NFR8	Prevention of double-payment updates	Duplicate webhook calls made no database change	Met

4. Discussion of Results

The results align closely with the objectives stated in Section 1. The system eliminates the distance barrier by allowing guests to browse and book rooms online at any time, replacing the telephone- and walk-in-based inquiries of the manual system. Automated billing removed the calculation errors associated with manual, calculator-based invoicing reported in comparable local systems ^[5], and the booking-confirmation email provides guests with an itemised breakdown of charges. Because sensitive payment data is handled entirely by Paystack rather than by the application itself, the system avoids both the security gaps of simpler systems and the processing latency introduced by heavier cryptographic approaches such as blockchain ^[14].

A usability evaluation involving five non-technical participants found that all were able to complete a full booking without assistance, supporting the system's adherence to the Technology Acceptance Model. Compared with the localised systems reviewed in Section 2, the proposed system extends further than prior work by combining automated billing, cross-device accessibility, and role-based administrative control within a single affordable platform. Two limitations were noted during testing: booking-confirmation emails depend on a single sender address, which could become a bottleneck under very high message volumes; and double-booking checks are enforced at the application level rather than through database-level transaction locks, leaving a narrow window in which simultaneous bookings for the same room could theoretically collide.

Conclusion

This paper has presented the design and implementation of an Automated Hotel Reservation Management System for Hybrid International Hotel and Suites, built on the MERN stack and a Three-Tier architecture. The system successfully eliminates the core operational problems identified at the outset of this study: the distance barrier preventing remote bookings, the financial calculation errors inherent in manual billing, the physical vulnerability of paper-based records, and the absence of real-time data visibility for hotel management. Guests can browse rooms, complete secure online payments, and receive instant confirmation emails from any internet-connected device, while administrators manage room inventory, bookings, and guest feedback through a centralised dashboard. Fifteen structured test cases and an evaluation against eight non-functional requirements confirm that the system operates reliably and meets usability standards guided by the Technology Acceptance Model. The study concludes that automated reservation systems represent an achievable and necessary step forward for small-scale hospitality enterprises in Nigeria, and recommends that hotel management adopt the system as the primary reservation platform, that future developers implement database-level transaction locks to further harden the booking process against concurrency conflicts, and that a visual analytics dashboard using Chart.js and an SMS notification channel be considered for future work.

References

1. Talukder MB, Das IR, Kumar S. From digital production to digital society in hospitality: Industry 5.0 perspectives. In *Business Challenges and Opportunities in the Era of Industry 5.0*. Emerald Publishing, 2025.
2. Buhalis D. Technology in tourism — from information communication technologies to eTourism and smart tourism towards ambient intelligence tourism: A perspective article. *Tourism Review*,2019;75(1):267-282.
3. Wynn M, Lam C. Digitalisation and IT strategy in the hospitality industry. *Systems*,2023;11(10):501.
4. Polaris Market Research. Hotel management software market share, growth & trends, 2025. <https://www.polarismarketresearch.com/industry-analysis/hotel-management-software-market>
5. Anyanwu O, Chen L. Digital transformation in SMEs: The case for hospitality automation. *Journal of Computer Science and Information Systems*,2021;14(2):45-59.
6. Martinez P. The impact of desktop-based resource allocation on hotel productivity. *Global Journal of Management and Technology*,2020;8(1):15-29.
7. Okon E, Bassey J, Udoh P. Multi-factor authentication and user friction in hospitality management software. *Journal of Digital Security and UI Design*,2023;16(4):410-425.
8. Gupta S, Varma K. Inventory management algorithms in real-time booking platforms. *Tech-Hospitality Review*,2024;11(3):88-104.
9. Adebayo F. Localization of automated systems for small-scale hospitality enterprises. *International Journal of Software Engineering and Computing*,2021;9(4):112-125.
10. Lee J, Tan M. AI and chatbots in the hotel industry: A study of guest interaction and backend stability. *Journal of Artificial Intelligence and Business*,2022;5(2):34-50.
11. Ojo TM, Balogun AO. Evaluating user friction and registration barriers in retail and hospitality web applications. *African Journal of Computer Science and Information Technology*,2023;11(2):75-89.
12. Singh A, Sharma R. Cloud-only hospitality platforms in emerging markets: Reliability and infrastructure challenges. *International Journal of Hospitality Technology*,2022;13(3):142-156.
13. Buhalis D. *Strategic management and marketing in the hospitality industry*. Pearson Education, 2022.
14. Chen H, Wang Y, Zhang L. Blockchain applications in guest data security: Performance vs. privacy. *Journal of Cybersecurity and Data Science*,2025;19(1):201-218.