iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 14 **Issue:** II **Month of publication:** February 2026

**DOI:** https://doi.org/10.22214/ijraset.2026.77310

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue II Feb 2026- Available at www.ijraset.com*

# Design and Implementation of a Web-Based Hotel Booking System Using Java and Modern Web Technologies

Lalam NavaDeepthi[1], Chinthapalli Dinesh Pallam Naidu[2], Kotipalli Siva Naga Pavan[3], Adabala Sri Lalitha Sampath Sai Saranya[4], Kusuma Ritesh Paul[5], M S V Gopalarao[6]

[1, 2, 3, 4, 5]*Department of Computer Science and Engineering, Bonam Venkata Chalamayya Engineering College, Affiliated to JNTU Kakinada, Andhra Pradesh, India*

[6]*Project Guide, Department of Computer Science and Engineering*

***Abstract:*** ***This paper presents the design and implementation of a web-based Hotel Booking System. The system provides a comprehensive solution for both hotel administrators and cus-tomers, enabling efficient room management, booking operations, payment processing, and administrative oversight.***

***The system is developed using modern web technologies including React for the frontend, Spring Boot for the backend, and MySQL for data storage. The application implements a role-based access control mechanism, distinguishing between adminis-trative and customer functionalities. Administrators can manage room inventory, view booking statistics, process payments, and generate reports. Customers can search for available rooms, make reservations, process payments, and manage their bookings.***

***The system incorporates various modules including authenti-cation, room management, availability search, booking manage-ment, payment gateway simulation, invoice generation, refund processing, and coupon management. The implementation follows RESTful API architecture principles, ensuring scalability and maintainability.***

***The project demonstrates the application of software en-gineering principles in developing a real-world solution. The system has been tested for functionality, usability, and reliability, meeting the specified requirements. The implementation provides a foundation for future enhancements such as real payment gateway integration, advanced analytics, and mobile application support.***

***Index Terms: Hotel management, online booking, Java, Spring Boot, React, MySQL, REST API, role-based access control, payment workflow, invoicing.***

## I. INTRODUCTION

Digital transformation has become essential in the hospital-ity sector, where customers expect 24×7 access to room avail-ability, transparent pricing, and instant booking confirmations. Manual reservation handling can lead to delayed responses, inconsistent records, and overbooking. A hotel booking system enables online room search, reservation creation, and payment processing while providing administrators a centralized plat-form to manage room inventory and bookings.

For small-to-medium hotels, adopting a cost-effective, main-tainable, and secure solution remains a challenge. Enterprise hotel property management systems (PMS) may offer robust features, but they are often expensive and complex to configure for smaller properties. Therefore, an affordable and modular platform is needed to support real-time discovery, booking lifecycle management, and administrative reporting. This work introduces a web-based Hotel Booking System developed using React for the frontend, Spring Boot for backend services, and MySQL for storage. The system im-plements role-based access for customers and administrators, real-time availability search, booking creation and cancella-tion, simulated payment workflows, and invoice-ready booking records. The architecture follows a layered controller–service–repository model and is designed for extensibility.

## II. RELATED WORK AND MOTIVATION

Prior work on hotel booking management systems emphasizes web-based interfaces, role separation between customer and administrative users, and automated record keeping. The IJCRT-based system describes three modules—Admin, Re-ception, and User—with functions such as room category management, booking approval, and report generation, along with DFD and UML modeling for system understanding [**?**].

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue II Feb 2026- Available at www.ijraset.com*

The IJESAT paper focuses on a Java-backed booking work-flow with room availability updates, booking confirmation, and testing stages. It also highlights future scope such as AI automation and IoT smart hotel integration [**?**]. Such works motivate our approach to implement an extensible full-stack system that balances design clarity, modular implementation,and security.

Motivated by these systems, the proposed solution delivers a maintainable implementation using React + Spring Boot + MySQL with improved modularity, secure access control, and REST APIs. The system is optimized for smaller organizations that require fast deployment, operational accuracy, and affordable maintenance.

## III. PROBLEM STATEMENT

The traditional hotel booking ecosystem faces numerous challenges that impact both customers and hotel operators. These challenges stem from outdated processes, lack of in-tegration between systems, manual data management, and limited technological infrastructure. Understanding these prob-lems is essential for appreciating the value proposition of an automated booking system.

One of the primary problems is the reliance on manual booking processes. Customers must typically contact hotels through telephone calls or email correspondence to inquire about room availability. This process is time-consuming for both parties, requires staff availability during business hours, and often results in incomplete or inaccurate information being communicated. Customers may need to make multiple calls to different hotels, compare prices manually, and wait for email responses that may take hours or days.

Hotel staff face significant challenges in managing room inventory manually. They must maintain physical or basic digital records of room availability, update these records after each booking, and manually calculate pricing based on dates, room types, and seasonal rates. This manual process is prone to human error, leading to situations where rooms may be double-booked or incorrectly marked as available when they are actually occupied.

The lack of real-time availability information creates a critical problem. When customers call to check availability, the information provided may be outdated by the time they decide to make a booking. Other customers may have booked the same room in the interim, leading to overbooking situations that damage customer relationships and hotel reputation. Ho-tels may also miss revenue opportunities when rooms remain unbooked due to inefficient communication channels.

Payment processing presents another significant challenge in traditional systems. Hotels often require customers to provide credit card information over the phone or through email, which raises security concerns. Payment verification and processing may take additional time, delaying booking confirmations. Re-fund processing for cancellations is typically manual and time-consuming, requiring staff intervention and multiple steps.

The absence of integrated systems creates operational ineffi-ciencies. Hotels may use separate systems or manual processes for booking management, payment processing, invoice gener-ation, and reporting. This lack of integration means that data must be manually transferred between systems, increasing the likelihood of errors and requiring additional staff time for data entry and reconciliation.

Administrative tasks such as generating reports, analyz-ing booking trends, calculating revenue, and managing room inventory require significant manual effort. Hotel managers may need to compile data from multiple sources, perform calculations manually, and create reports using spreadsheet applications. This process is time-consuming and may not provide timely insights needed for decision-making.

Customer experience suffers in traditional booking systems. Customers cannot access booking information outside business hours, must wait for staff responses, and may experience delays in receiving confirmations. The booking process may require multiple interactions with hotel staff, creating friction and reducing customer satisfaction.

Cost considerations also present challenges. Enterprise hotel management systems are often expensive, requiring significant upfront investment and ongoing licensing fees. These systems may be over-engineered for smaller hotels, providing features that are not needed while lacking customization options. Third-party booking platforms charge commission fees that reduce hotel profit margins.

The problem statement can be summarized as follows: The hospitality industry requires a unified, automated, web-based system that integrates room management, booking operations, payment processing, and administrative functions into a sin-gle platform. This system must provide real-time availability information, enable 24/7 customer access, automate routine tasks, ensure data accuracy, and provide comprehensive administrative tools while remaining cost-effective and scalable.

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue II Feb 2026- Available at www.ijraset.com*

## IV. SYSTEM OBJECTIVES

The objectives of the proposed system are:

*1)* Provide a seamless customer-facing interface for searching and booking rooms.
*2)* Implement real-time availability updates to prevent over-booking.
*3)* Support administrative control over rooms, bookings, and users.
*4)* Offer a payment workflow (test-mode) and invoice generation.
*5)* Maintain data integrity via normalized relational schemas.
*6)* Ensure security through authentication and role-based access control.

## V. SYSTEM ARCHITECTURE

The system follows a three-tier architecture. The presentation layer is developed using React to provide responsive user interfaces. The application layer is implemented with Spring Boot, exposing REST endpoints for authentication, room search, booking operations, payment status updates, and reporting. The data layer uses MySQL for persistent storage of users, rooms, bookings, payments, and coupon/offer entities.

This separation improves maintainability and supports in-dependent scaling of the frontend and backend services. A layered controller–service–repository structure is used in the backend to isolate business logic from data access.

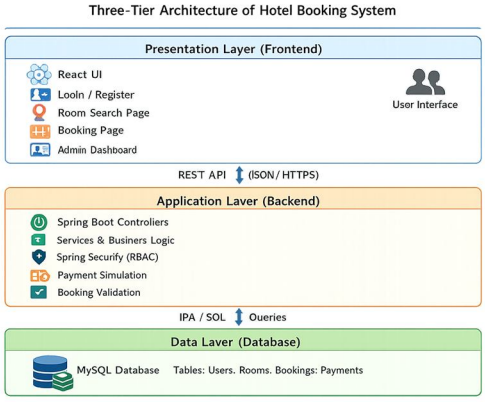



Fig. 1: Three-Tier Architecture of the Hotel Booking System

## VI. MODULE DESIGN

The solution is organized into modules aligned with core hotel workflows. This modular approach improves maintain-ability and supports incremental enhancement.

### *A. Authentication Module*

The authentication module supports registration and login. Passwords are stored using encrypted hashing (e.g., BCrypt) to protect user credentials. Role-based authorization restricts sensitive endpoints to administrators.

### *B. Room Management Module*

Administrators can create and update room records includ-ing room number, room type, price, and status. Rooms can be logically disabled to prevent booking while maintaining historical data.

### *C. Availability Search Module*

The availability module returns rooms that do not overlap with existing bookings for the requested date range. This prevents overbooking and ensures accurate availability display.

### *D. Booking Management Module*

The booking module creates booking records, computes stay duration and totals, and maintains booking states (pend-ing/confirmed/cancelled). Cancellation triggers status updates and restores availability.

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue II Feb 2026- Available at www.ijraset.com*

### *E. Payment Module (Simulation)*

The payment workflow is simulated in test mode. It captures the payment method, reference ID, amount, and status. Suc-cessful payment confirms the booking, while failed payment keeps it pending.

### *F. Invoice Module*

After confirmation, invoice-ready records are prepared. In-voices include booking code, user information, room details, stay duration, and payment status.

### *G. Admin Dashboard Module*

The admin dashboard shows occupancy and revenue statis-tics. It supports booking monitoring, room status view, and basic reporting features.

## VII. CORE DIAGRAMS

### *A. Data Flow Diagram (DFD)*

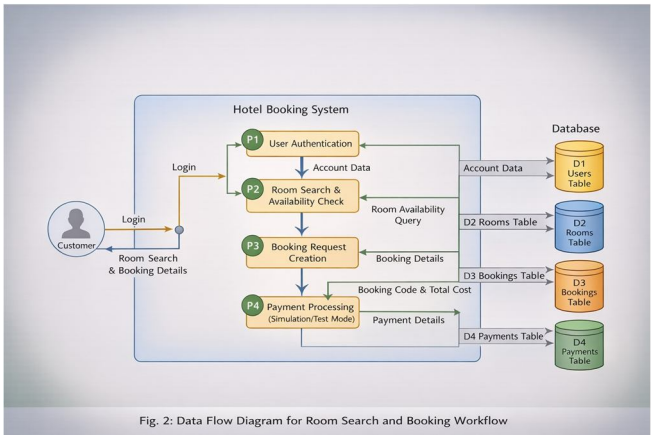



Fig. 2: Data Flow Diagram for Room Search and Booking Workflow

The DFD illustrates how customer actions (login, search, booking, cancellation) interact with backend services and database storage. Administrative actions include room updates and booking oversight.

### *B. Entity Relationship (ER) Diagram*

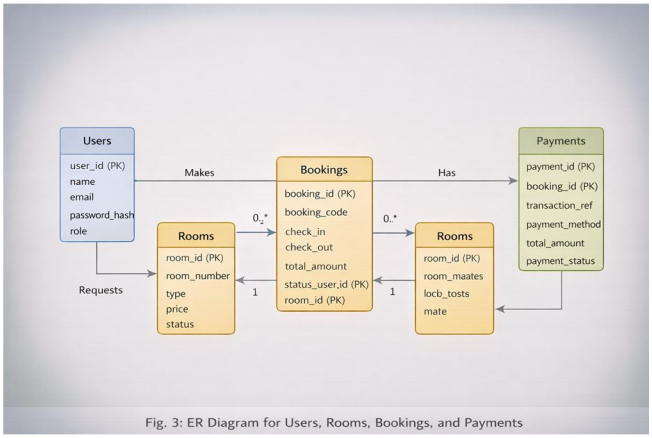
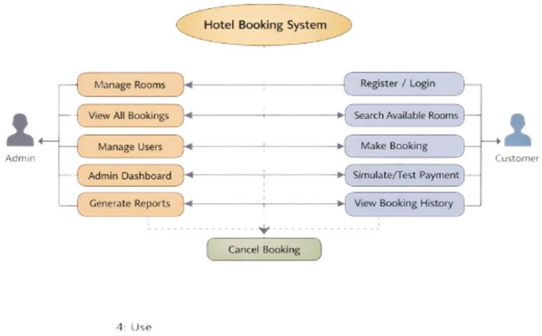



Fig. 3: ER Diagram for Users, Rooms, Bookings, and Pay-ments

### C. Use Case Diagram




4: Use

Fig. 4: Use Case Diagram for Admin and Customer Roles

## VIII. DATABASE DESIGN

A relational database schema is used to ensure consistency and prevent redundancy. Key tables include: Users, Rooms, Bookings, Payments, and optional Offers/Coupons for dis-counts.

TABLE I: Core Database Entities

| Entity | Attributes (Example) |
|---|---|
| Users | id, name, email, password_hash, role |
| Rooms | id, room_number, type, price, status |
| Bookings | id, booking_code, user_id, room_id, check_in, check_out, total_amount, status |
| Payments | id, booking_id, method, reference, amount, status |

Foreign-key constraints enforce integrity between booking and user/room entities. Indexes are created for frequent queries on room availability and booking lookup by booking code.

Database optimization ensures efficient data storage and retrieval, maintaining system performance even as data volume grows. Several optimization techniques are implemented to achieve this goal.

### A. Indexing Strategy

Indexes are created on frequently queried columns to im-prove query performance. Primary key indexes are automati-cally created on all tables. Foreign key columns are indexed to improve join performance. Email column in users table is indexed for fast login lookups. Room number column in rooms table is indexed for quick room searches. Booking code col-umn in bookings table is indexed for efficient booking lookups. Payment reference column in payments table is indexed for payment tracking. These indexes significantly improve query performance while adding minimal storage overhead.

### B. Query Optimization

Database queries are optimized to minimize execution time and resource usage. SELECT statements use specific column names instead of SELECT * to reduce data transfer. WHERE clauses are structured to utilize indexes effectively. JOIN operations are optimized to minimize the number of rows processed. Subqueries are used judiciously, with preference for JOINs when appropriate. Query execution plans are analyzed to identify optimization opportunities.

### C. Connection Pooling

Connection pooling manages database connections effi-ciently, reducing connection overhead and improving perfor-mance. A pool of database connections is maintained, allowing connections to be reused rather than created and destroyed for each request. This reduces connection establishment time and database server load. Connection pool size is configured based on expected concurrent users and system resources.

### D. Transaction Management

Transaction management ensures data consistency and opti-mal performance. Transactions are used for operations that modify multiple related records, ensuring atomicity. Trans-action boundaries are carefully defined to minimize lock duration. Read operations use appropriate isolation levels to balance consistency and performance. Deadlock prevention strategies are implemented to avoid transaction conflicts.

## IX. IMPLEMENTATION DETAILS

### A. Backend Implementation (Spring Boot)

The backend of the Hotel Booking System is developed using Java Spring Boot and follows a layered architecture con-sisting of Controller, Service, and Repository layers. RESTful APIs are created to handle major operations such as user authentication, room management, booking processing, and payment simulation.

The Controller layer receives requests from the frontend and forwards them to the service layer. The Service layer contains the core business logic such as validating user inputs, checking room availability, calculating booking costs, generat-ing booking codes, and updating booking/payment status. The Repository layer uses Spring Data JPA to perform database operations with MySQL.

Security is enforced using authentication and role-based ac-cess control, ensuring that only administrators can access priv-ileged operations like adding rooms, updating room details, and viewing reports. The backend also includes validation rules to prevent invalid booking dates, duplicate bookings, and incorrect payment updates.

### B. Frontend Implementation (React)

The frontend of the system is implemented using React to provide a responsive and user-friendly interface for both cus-tomers and administrators. React components are used to build different pages such as Login, Registration, Room Search, Booking Page, User Dashboard, and Admin Dashboard. Navi-gation between pages is handled using React Router, providing a smooth single-page application experience.

The frontend communicates with the backend through REST APIs using tools like Axios/Fetch, enabling real-time display of room availability, booking confirmations, and status updates. Form validations are performed on the client side to ensure correct input formats such as valid dates, email structure, and required fields before sending requests to the backend.

Overall, the React interface improves customer experience by enabling easy booking and tracking, while the admin panel supports effective management of hotel operations such as monitoring bookings and maintaining room inventory.

## X. SECURITY AND ACCESS CONTROL

Security is enforced through authentication and role-based access control. Passwords are stored using strong hashing to prevent credential leakage. Authorization rules restrict room creation and report access to administrators only.

### A. *Authentication* Mechanism

The system implements secure user authentication to verify the identity of both customers and administrators. During registration, user credentials are collected and stored securely in the database. Passwords are never stored in plain text; instead, they are protected using strong hashing algorithms such as BCrypt/SHA-based hashing, which prevents credential leakage even if the database is compromised. During login, the entered password is compared with the stored hashed value to validate user access.

### B. Role-Based Access Control (RBAC)

After successful authentication, users are assigned roles such as Admin or Customer, and access privileges are enforced accordingly. RBAC ensures that each user can only perform actions based on their role: • Admin users can manage rooms, update pricing, view all bookings, manage users, and gener-ate reports. • Customers can search rooms, make bookings, simulate payments, cancel bookings, and view their personal booking history.

Authorization rules restrict sensitive features like room creation, room deletion, and report generation strictly to administrators, preventing unauthorized modifications. This separation ensures both operational safety and data integrity.

### C. Input *Validation* and Data Sanitization

To prevent invalid data entry and security vulnerabilities, in-put validation is implemented at both the client side (frontend) and server side (backend). Examples include: • Validating check-in and check-out dates to prevent inconsistent booking requests. • Restricting negative values or unrealistic room pricing entries. • Enforcing required fields such as email and password format rules. • Sanitizing inputs to reduce the risk of injection-based attacks.
Server-side validation is considered the primary defense layer since client-side validation alone can be bypassed.

### D. *Secure* API Access and Session Control

The backend is exposed through RESTful APIs, and each API endpoint is protected based on authorization level. Ses-sions (or authentication tokens) ensure that a user remains logged in securely and cannot access restricted routes with-out proper verification. APIs return only the necessary data required for a request, reducing unnecessary exposure of sensitive information.

### E. *Data* Protection in Transit and Deployment Security

In production deployment, secure communication between the client and server is essential. The system recommends the use of HTTPS (SSL/TLS encryption) to protect sensitive information such as login credentials and booking confir-mations from eavesdropping or man-in-the-middle attacks. Additionally, database access should be restricted using secure credentials and environment-based configuration rather than hardcoded values.

### F. *Security* Benefits

By combining authentication, RBAC, validation, and secure communication practices, the proposed system provides: • Reduced risk of unauthorized access • Improved protection of user credentials • Safer admin operations and controlled privileges • Reliable booking workflow without manipulation • Stronger protection against malformed or malicious inputs

## XI. ROOM AVAILABILITY COMPUTATION

Availability is determined by checking overlap between requested booking dates and existing confirmed bookings for each room.

**Algorithm 1** Availability Check for a Requested Date Range

```
Input: checkIn, checkOut, roomType
candidateRooms ← Fetch rooms where type=roomType and status=AVAILABLE
for each room in candidateRooms do
    bookings ← Fetch confirmed bookings for room
    for each booking in bookings do
        overlap ← (checkIn ¡ booking.checkOut) AND (checkOut ¿ booking.checkIn)
        if overlap == TRUE then
            Mark room as unavailable
        end if
    end for
end for
Return rooms marked available
```

## XII. TESTING METHODOLOGY

Testing follows a staged approach: unit testing validates in-dividual services (availability computation, booking creation), integration testing validates interactions between modules (booking → payment → invoice), and system testing validates end-to-end user journeys.
Functional tests verify: search accuracy, correct total calcu-lation, payment status updates, and cancellation flows.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue II Feb 2026- Available at www.ijraset.com

TABLE II: Sample Functional Test Cases

| Test Case | Expected Result |
|---|---|
| Valid login | User redirected to correct dashboard based on role |
| Room search dates valid | Only rooms without overlap are returned |
| Booking confirmation | Booking status changes to CONFIRMED after payment success |
| Cancellation | Booking becomes CANCELLED and room becomes AVAILABLE |

Testing was conducted to ensure that the proposed Hotel Booking System meets functional requirements and performs reliably across all modules. Since the system integrates authen-tication, room availability checking, booking operations, and payment workflow, multiple testing stages were followed to validate both individual components and complete workflows. Unit Testing was performed to verify core functions such as user login/registration, room management, availability calcu-lation, booking creation, and cancellation logic. Each service was tested with valid and invalid inputs to confirm correct outputs and error handling.

Integration Testing ensured smooth communication between the frontend and backend through REST APIs. This included testing the end-to-end flow of room search, booking confir-mation, payment status update, and database consistency after cancellation.

Functional and System Testing validated complete user workflows, including customer booking, admin room manage-ment, and booking history tracking. Boundary testing was also applied to validate date inputs, prevent overlapping bookings, and ensure accurate total amount calculation.

Finally, Security Testing confirmed that role-based access control restricts admin-only operations, passwords are securely stored using hashing, and unauthorized users cannot access protected resources. Overall, the testing results confirmed that the system performs accurately, prevents double-booking, and maintains consistent data updates.

## XIII. RESULTS AND DISCUSSION

The system enables customers to discover available rooms quickly, reserve rooms online, and manage their bookings via a dashboard. Administrators can manage inventory and monitor booking statistics.

Compared to manual processes, the automated system re-duces time spent on availability checking and minimizes overbooking errors through synchronized room status updates. The modular design improves maintainability and supports incremental enhancement.

## XIV. LIMITATIONS

The current implementation includes payment processing in test/simulated mode and does not integrate with external payment gateways. Notification via email/SMS is optional and may be limited to development configurations. The scope is currently for single-property hotel management.

## XV. FUTURE ENHANCEMENTS

The future scope includes the integration of real payment gateways, automated notifications, and advanced analytics. AI-powered chatbots and dynamic pricing modules can improve customer engagement and revenue optimization. Integration with IoT devices can enable smart check-in/out and personal-ized room control.

1) Integration with real payment gateways for actual transactions
2) Email and SMS notification system for booking confirma-tions
3) Customer review and rating system for rooms and services
4) Mobile application development for iOS and Android plat-forms
5) Multi-property support for hotel chains
6) Advanced analytics with predictive insights
7) Loyalty program with points and rewards
8) Integration with external booking platforms

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue II Feb 2026- Available at www.ijraset.com

## XVI. LESSONS LEARNED

The development of the Hotel Booking System provided valuable learning experiences that contribute to professional growth and future project success. These lessons cover tech-nical, process, and project management aspects.

### A. Technical Lessons

Technical lessons include the importance of proper database design for system performance and maintainability. Security considerations must be integrated from the beginning rather than added later. API design should follow RESTful principles for consistency and predictability. Code organization and doc-umentation facilitate maintenance and future enhancements. Testing at multiple levels ensures system reliability and re-duces production issues.

### B. Process Lessons

Process lessons emphasize the value of iterative develop-ment for continuous improvement. Early requirement clarifica-tion prevents rework and delays. Regular communication and collaboration improve project outcomes. Version control and code reviews maintain code quality. Documentation through-out development supports future maintenance and knowledge transfer.

### C. Project Management Lessons

Project management lessons highlight the importance of realistic timeline estimation. Prioritization ensures critical fea-tures are completed first. Risk management identifies and addresses potential issues early. Stakeholder communication keeps all parties informed and aligned. Flexibility allows adaptation to changing requirements and constraints.

## XVII. PROJECT IMPACT AND BENEFITS

The Hotel Booking System provides significant benefits to various stakeholders including hotel operators, customers, and the broader hospitality industry. Understanding these benefits demonstrates the value and impact of the project.

### A. Benefits to Hotel Operators

Hotel operators benefit from reduced operational costs through automation of manual processes. Staff productivity increases as routine tasks are automated, allowing focus on customer service. Revenue opportunities expand through 24/7 booking availability. Data insights enable data-driven decision making about pricing and promotions. Customer relationships improve through direct booking channels without commission fees. System scalability supports business growth without proportional cost increases.

### B. Benefits to Customers

Customers benefit from convenient 24/7 booking access without requiring staff availability. Booking process is faster and more efficient compared to traditional methods. Real-time availability information prevents disappointment from unavailable rooms. Instant confirmations provide immediate booking assurance. Online payment processing offers secure and convenient transaction handling. Invoice access enables easy record keeping and expense management.

### C. Benefits to the Industry

The hospitality industry benefits from demonstration of affordable technology solutions for small and medium hotels. Open-source approach makes advanced functionality accessi-ble to establishments with limited budgets. Modular architec-ture provides foundation for future innovations. Best practices demonstrated can guide other development efforts. Educational value helps train future hospitality technology professionals.

## XVIII. CONCLUSION

This paper presented a practical and scalable Hotel Booking System built using Java and modern web technologies. The proposed architecture and modular design support real-time availability, secure authentication, booking lifecycle manage-ment, and administrative insights. The system reduces oper-ational effort and improves customer experience, making it suitable for small-to-medium hotels as an affordable alterna-tive to enterprise PMS platforms.

## REFERENCES

[1] Project Report, “Hotel Booking System: A Web-Based Application for Hotel Room Reservation Management,” BONAM VENKATA CHALA-MAYYA ENGINEERING COLLEGE, Academic Year 2024–2026.

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue II Feb 2026- Available at www.ijraset.com*

[2] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsman-ship. Upper Saddle River, NJ, USA: Prentice Hall, 2008.
[3] E. Freeman and E. Robson, Head First Design Patterns. Sebastopol, CA, USA: O'Reilly Media, 2020.
[4] A. Banks and E. Porcello, Learning React: Modern Patterns for Devel-oping React Apps. Sebastopol, CA, USA: O'Reilly Media, 2020.
[5] C. Walls, Spring in Action. Shelter Island, NY, USA: Manning Publica-tions, 2021.
[6] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts. New York, NY, USA: McGraw-Hill Education, 2019.
[7] D. Flanagan, JavaScript: The Definitive Guide. Sebastopol, CA, USA: O'Reilly Media, 2020.
[8] C. S. Horstmann, Core Java Volume I: Fundamentals. Upper Saddle River, NJ, USA: Prentice Hall, 2019.
[9] M. Fowler, Patterns *of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley, 2002.

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue II Feb 2026- Available at www.ijraset.com*

SYSTEM OUTPUT SCREENS (LAST PAGE)

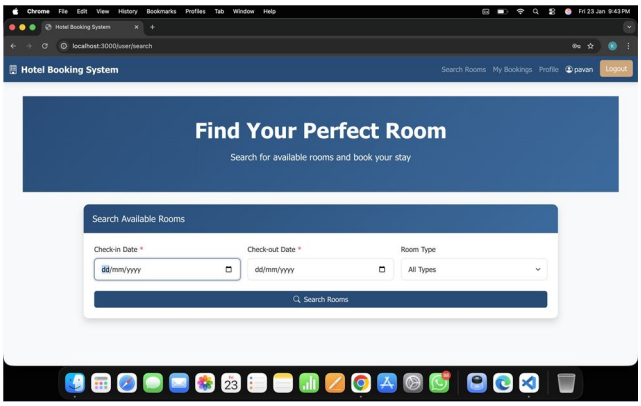


Fig. 5: Output Screen 1: Room Availability Search

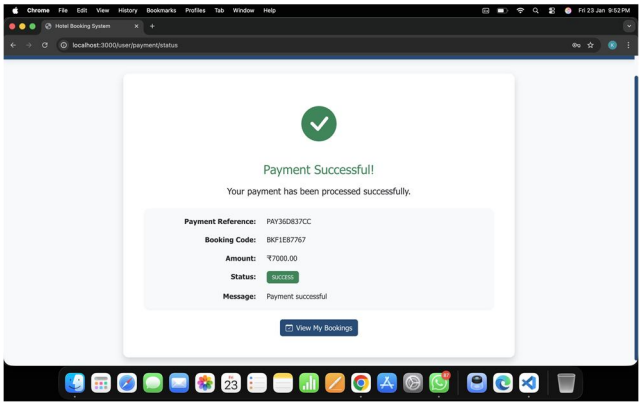


Fig. 6: Output Screen 2: Booking Confirmation

## BIOGRAPHIES OF AUTHORS

**Lalam NavaDeepthi** is currently pur-suing a Bachelor of Technology de-gree in Computer Science and En-gineering at Bonam Venkata Cha-lamayya Engineering College, affil-iated to JNTU Kakinada. Her in-terests include full-stack web devel-opment, RESTful application design, and database management.
Phone no: 7993799757
Email: 22221a0562@gmail.com

**Kotipalli Siva Naga Pavan** is pursu-ing a Bachelor of Technology degree in Computer Science and Engineering at Bonam Venkata Chalamayya Engi-neering College. His interests include system architecture and scalable web applications.
Phone no: 9110388011
Email: 22221a0556@gmail.com

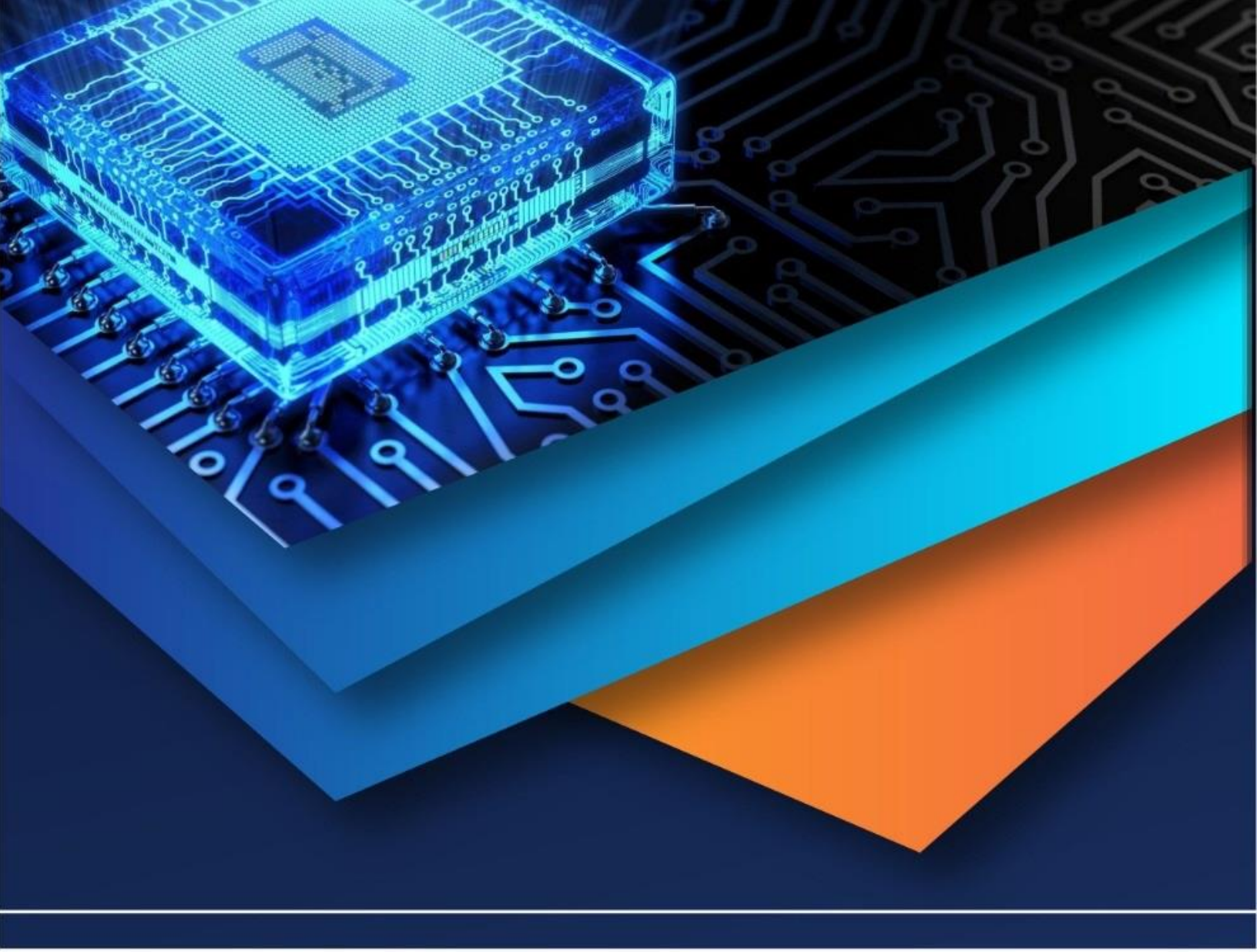
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue II Feb 2026- Available at www.ijraset.com*

**Kusuma Ritesh Paul** is a Bache-lor of Technology student in Com-puter Science and Engineering at Bonam Venkata Chalamayya Engi-neering College. His interests include software engineering and database systems.
Phone no: 701305956
Email: 22221a0561@gmail.com

**Chinthapalli Dinesh Pallam Naidu** is a Bachelor of Technology student in Computer Science and Engineering at Bonam Venkata Chalamayya Engi-neering College. His interests include Java backend development and API design.
Phone no: 8790495578
Email: 22221a0521@gmail.com

**Adabala Sri Lalitha Sampath Sai Saranya** is a Bachelor of Technology student in Computer Science and En-gineering at Bonam Venkata Chala-mayya Engineering College. Her in-terests include frontend development and UI/UX design.
Phone no: 9063723222
Email: 22221a0504@gmail.com

**M.S.V.Gopalarao** is an Associate Professor in the Department of Com-puter Science and Engineering at Bonam Venkata Chalamayya Engi-neering College, affiliated to JNTU Kakinada. His expertise includes soft-ware engineering and web technolo-gies.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY