International Journal of Research in Engineering and Innovation (IJREI)
journal home page: http://www.ijrei.com
ISSN (Online): 2456-6934

**RESEARCH PAPER**

# Hotel booking and listing

**Tarun Verma, Virat, Saurabh Kumar, Tanishq Kumar, Amol Sharma**

*Department of Computer Science and Engineering, Meerut Institute of Technology, Meerut, India*

Article Information

Received: 11 May 2025
Revised: 18 May 2025
Accepted: 04 June 2025
Available online: 05 June 2025

Keywords:

Power BI
Sales Dashboard
Data Visualization
Business Intelligence
Sales Analytics

**Abstract**

**In this work**, we explore the design and deployment of a full-stack hotel booking and listing web application, reflecting the growing influence of digital technologies in the hospitality and tourism industries. As traditional booking methods—such as phone calls, walk-ins, and emails—decline, travelers increasingly prefer real-time, transparent, and user-friendly web platforms. To meet these demands, we developed a dynamic system using the MERN-style stack (Node.js, Express.js, MongoDB, and EJS), offering a seamless experience for users to search, filter, and book hotel rooms. The application includes core functionalities such as user registration, authentication, hotel listing management, real-time booking, and a review and rating system to support user decision-making. A standout feature is the integrated chatbot module, providing interactive assistance with typing animations, contextual feedback, and a conversational interface. The backend ensures secure data handling, structured MongoDB storage, and well-documented Express.js routing using the MVC architecture for maintainability and scalability. EJS templating enables real-time dynamic rendering without full-page reloads, enhancing responsiveness. This report documents the development process, tools, and challenges, and evaluates the application's usability and scalability. Future enhancements include payment integration, AI-based hotel recommendations, geolocation, and multilingual support, positioning the platform as a comprehensive digital solution for modern travelers. 

## 1. Introduction

The evolution of web-based hotel reservation portals has revolutionized the established industry benchmarks with their huge inventories of hotels and homestays, interactive search filters, dynamic pricing, user ratings, and secure payment gateway integration. These sites generally have mature backend systems and scalable databases, which are developed to run on microservices architectures, cloud hosting, and RESTful APIs to support millions of users and update data in real-time. Though such sites provide cutting-edge functionality, they are built with huge teams, huge cost, and often proprietary technology, making them inaccessible to learners and small-scale developers. Other features of hotel reservation systems have also been addressed in scholarly literature. A paper by Li and Law (2007) addressed the influence of customer comments and ratings on reservation decisions and suggested that social proof features be incorporated into hotel reservation systems. A paper by Ham, Kim, and Jeong (2005) addressed the role of usability and interface design in generating user confidence and satisfaction, and it challenged the developers to go beyond functionality to user-centered design principles. From the technology stack point of view, the majority of modern web applications make use of Node.js because of its event-driven, non-blocking nature that is well-designed for real-time data such as availability and booking status. Tilkov and Vinoski's (2010) work offers the advantages of the use of JavaScript across the stack—server-side and client-side—that enables development efficiency and maintainability. MongoDB, which is a NoSQL database, has also gained immense popularity because of the freedom it provides while handling

*Corresponding author: Virat*
*Email Address: virrattomar1@gmail.com*
https://doi.org/10.36037/IJREI.2025.9409

unstructured data and rapid development, especially where there is dynamic content and frequent updates. The MVC (Model-View-Controller) pattern, widely employed in web development, has been well supported in literature for its potential to decouple concerns, simplify prospects of chatbots in improving website user experience. Følstad and Brandtzæg (2017) found that conversational interfaces can make users more interactive and satisfied if they are human-like and context-aware. Creating a chatbot with typing animations and bill generation in this project follows these study findings and extends the scope of common hotel reservation systems by making them interactive and intuitive systems. In spite of the technological innovation in hotel booking systems, most of the existing implementations are not open-source, editable, or pedagogically transparent. This requirement highlights the necessity of creating full-stack web applications that are technologically appropriate and pedagogically useful, especially for students and junior programmers.

## 2. Literature Review

The evolution of web-based hotel reservation portals has revolutionized the hospitality industry significantly by increasing accessibility, efficiency, and customer satisfaction. Various architectures, facilities, and technologies have been examined and applied in various studies over the past few years to improve online booking. A critical review of existing research and web-based hotel reservation portals relevant to web-based hotel booking system design is presented in this section. A few major commercial websites, like Booking.com, Airbnb, and Agoda, have established industry benchmarks with their huge inventories of hotels and homestays, interactive search filters, dynamic pricing, user ratings, and secure payment gateway integration. These sites generally have mature backend systems and scalable databases, which are developed to run on microservices architectures, cloud hosting, and RESTful APIs to support millions of users and update data in real-time. Though such sites provide cutting-edge functionality, they are built with huge teams, high costs, and often proprietary technology, making them inaccessible to learners and small-scale developers. Other features of hotel reservation systems have also been addressed in scholarly literature. A paper by Li and Law (2007) addressed the influence of customer comments and ratings on reservation decisions and suggested that social proof features be incorporated into hotel reservation systems.

A paper by Ham, Kim, and Jeong addressed the role of usability and interface design in generating user confidence and satisfaction, and it challenged developers to go beyond functionality to user-centered design principles.

From the technology stack point of view, the majority of modern web applications make use of Node.js because of its event-driven, non-blocking nature that is well-suited for real-time data such as availability and booking status. Tilkov and Vinoski's (2010) work highlights the advantages of using JavaScript across the stack—server-side and client-side—which enables development efficiency and maintainability.

MongoDB, a NoSQL database, has also gained immense popularity because of the freedom it provides while handling unstructured data and enabling rapid development, especially where there is dynamic content and frequent updates.

The MVC (Model-View-Controller) pattern, widely employed in web development, is well supported in the literature for its potential to decouple concerns, simplify testing, and enable scalable code structures. Express.js, a well-known web framework for Node.js, encourages the MVC pattern and has a middleware architecture that facilitates effortless routing and server-side logic.

Recent research has also explored the prospects of chatbots in improving website user experience. Følstad and Brandtzæg (2017) found that conversational interfaces can make users more interactive and satisfied if they are human-like and context-aware. Creating a chatbot with typing animations and bill generation in this project follows these study findings and extends the scope of common hotel reservation systems by making them interactive and intuitive systems.

In spite of the technological innovation in hotel booking systems, most of the existing implementations are not open-source, editable, or pedagogically transparent. This requirement highlights the necessity of creating full-stack web applications that are technologically appropriate and pedagogically useful, especially for students and junior programmers.

## 3. Methodology

The hotel listing and web application booking development process was architecture, user-centric features, and fluid interaction. This section presents the technologies utilized, system design, development phases, and key implementation strategies.

### *3.1* System Architecture

The application employs the Model-View-Controller (MVC) paradigm to isolate business logic, user interface, and data operations separately:

- Model: It is the data structure and is tasked with communicating with the MongoDB database for handling hotels, users, bookings, and reviews.
- View: Created using EJS (Embedded JavaScript Templates) to render dynamic HTML pages by taking advantage of data sent by the controller.
- Controller: It is written in Express.js and is utilized for the application logic, the routing, and model and view interaction.

### *3.2 Technology Stack*

Table 1provides an overview of the technology stack used in the development of the hotel booking web application, categorized by different layers of the system architecture. The frontend is built using HTML5, CSS3, JavaScript, and

EJS, enabling the creation of a responsive and dynamic user interface. On the backend, Node.js and Express.js are employed to handle server-side operations, including routing, API creation, and middleware management. The database layer utilizes MongoDB, a NoSQL database known for its flexibility and scalability, allowing efficient storage and retrieval of user, hotel, and booking data. For dynamic web page rendering, EJS (Embedded JavaScript Templates) is used as the template engine, allowing real-time content updates and custom views without full page reloads. Finally, Git and GitHub are employed for version control, enabling collaborative development, efficient tracking of code changes, and secure source code management. Together, these technologies provide a robust, scalable, and maintainable framework for building a full-stack hotel booking application.

*Table 1: Technology Stack Used in the Hotel Booking Web Application*

| Layer | Technology Used |
|---|---|
| Frontend | HTML5, CSS3, JavaScript, EJS |
| Backend | Node.js, Express.js |
| Database | MongoDB (NoSQL) |
| Template Engine | EJS (Embedded JavaScript Templates) |
| Version Control | Git & GitHub |

### *3.3 Development Phases*

The development of the hotel booking web application was carried out in multiple structured phases. Phase 1 focused on planning and requirement analysis, where the core functionalities were identified, including user registration, hotel listings, room booking, chatbot interaction, and a review system. Wireframes and flowcharts were created for key interfaces such as the hotel detail page, booking form, and chatbot UI to visualize the user journey. In Phase 2, backend development was initiated using Node.js and Express.js. A connection to MongoDB was established via Mongoose, with schemas defined for User, Hotel, Booking, and Review models. Secure user login and authorization were implemented using Passport.js and session-based authentication. Phase 3 involved frontend development, where responsive user interfaces were built using HTML, CSS, and EJS. Interactive views were created for hotel filtering, booking forms with real-time validation, and review submission pages. The chatbot was also integrated with typing effects and live responses for a more dynamic experience. In Phase 4, a custom JavaScript-based chatbot was developed to guide users through hotel reservations with features such as typing animations, a realistic billing UI, and prompt-based interactions. It was placed contextually within the hotel detail pages. Phase 5 introduced the rating and review system. Users could leave reviews only after successful bookings, and the backend ensured that only logged-in users could submit them. Real-time average rating calculations were displayed per hotel. Phase 6 focused on testing and error handling. Server-side validation was handled using Joi to sanitize inputs, and functional testing ensured smooth routing, form submissions, and overall booking logic. Middleware was added to handle errors and display user-friendly messages. Finally, in Phase 7, version control was managed through Git, with regular commits documenting each module. The project was tested locally and prepared for deployment on platforms such as Render, Railway, or Heroku.

### *3.4 Security Measures*

To ensure the security of the application, several protective mechanisms were implemented. Passwords are securely hashed using encryption methods to prevent unauthorized access. Input validation is enforced throughout the system to safeguard against injection attacks and other malicious inputs. Additionally, access to key functionalities—such as booking a hotel or posting a review—is restricted to registered and authenticated users only, maintaining the integrity of user-generated content.

### *3.5 Tools Used*

A variety of tools supported the development and management of the application. Visual Studio Code served as the primary development environment, offering a flexible and efficient coding experience. Postman was utilized for API testing and validation of server responses during backend development. MongoDB Compass provided an intuitive interface for managing the MongoDB database, enabling easier inspection and manipulation of stored data. Finally, Git and GitHub were used for version control and team collaboration, ensuring proper tracking of changes and seamless integration of development modules.

## 4. Results

The hotel listing and booking web application was successfully implemented and created as a full-stack web solution that adheres to functional and non-functional requirements taken into account at the planning phase. The completed application can carry out a set of core operations with ease and provide users with an interactive and reliable hotel booking experience. The outcomes are grouped below to illustrate functionality, performance, user experience, and additional features.

### *4.1 Functional Achievements*

The application successfully achieved all its intended functional goals, delivering a complete and interactive hotel booking experience. User authentication and authorization were effectively implemented, allowing users to securely register, log in, and log out. Access control ensured that only authenticated users could make bookings or post reviews, maintaining system integrity. The hotel listing and

administration module featured a dynamic interface that displayed available hotels, each with a dedicated detail page showing descriptions, prices, amenities, reviews, and a booking form. Hotel data was fetched in real-time from MongoDB and rendered using EJS templates for seamless user interaction. The booking system allowed users to select check-in/check-out dates, provide guest information, and confirm bookings, which were then stored in the database and reflected on the user's profile. A booking confirmation mechanism was also integrated. The review and rating feature was designed so only users who had completed a stay could leave a review, ensuring authenticity. Reviews appeared with timestamps and usernames, while each hotel displayed an average rating to aid user decisions.

Finally, chatbot integration enhanced user support by providing hotel details, answering booking-related queries, and simulating a real-time billing interface based on user preferences, thereby enriching the overall booking experience.

*4.2 User Experience*

The application ensures intuitive navigation. Pages are responsive and adapt on devices and screen sizes. The chatbot and booking features are blended seamlessly without compromising on usability. Use of EJS allows for live updates without reloads of the whole page, which improves responsiveness.

*4.3 Performance and Reliability*

The app functions correctly in local test environments. MongoDB can handle low-latency data reads and writes. Routes, middleware, and forms were extensively tested via Postman as well as testing manually. All the main flows (register → login → view hotels → book → review → chatbot) worked flawlessly without crashes or significant bugs.

*4.4 Error Handling and Security*

Input validation was performed on client as well as server sides. It dealt with errors like invalid booking dates, repeated reviews, or unauthorized activity in a graceful manner. Security features like hashed passwords (bcrypt) and session-based authentication ensured the protection against unauthorized access and data breaches.

*4.5 Screenshots*

Fig. 1 illustrates the homepage interface of the hotel booking application, showcasing dynamic hotel listings fetched from the database. Each listing includes an image, name, and price per guest, providing users with a visually organized and accessible overview. The navigation bar includes essential features such as search functionality, user login, signup options, and the ability to add new listings, thereby enhancing user interaction and system usability.

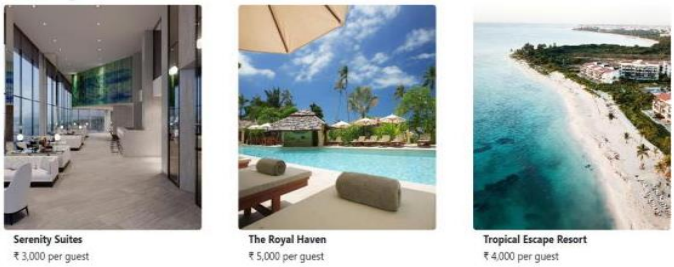


*Figure 1: Homepage with Hotel listings.*

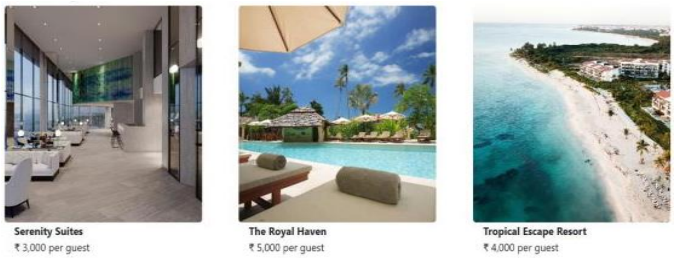


*Figure 2: Hotel detail view with chatbot and booking form*

Fig. 2 presents the detailed hotel view interface, where users can access specific information about each property, such as pricing, amenities, and room availability. The page integrates a booking form and an interactive chatbot designed to assist users in real time with queries regarding check-in dates, guest count, and bill details. This layout enhances the user experience by combining informative content with functional booking support in a seamless manner.

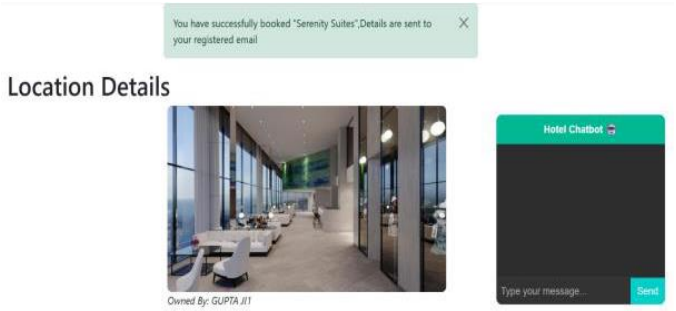


*Figure 3: Booking confirmation Page*

Fig. 3 illustrates the booking confirmation interface, which provides real-time feedback upon successful hotel reservation. A success message is prominently displayed to inform users that booking details have been sent to their registered email. The page also includes hotel location information and an integrated chatbot for continued assistance, ensuring a streamlined and reassuring user experience post-booking.

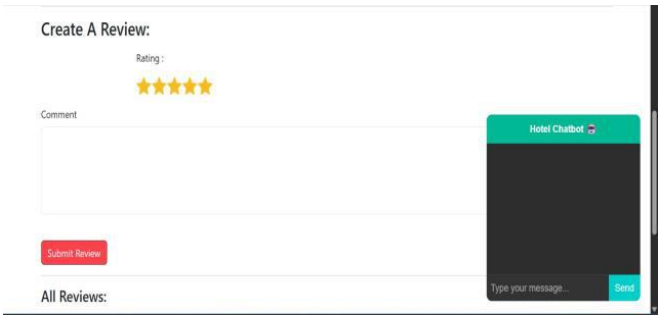


*Figure 4: Review section for hotels.*

Fig. 4 displays the review interface where authenticated users can submit feedback based on their stay. The section includes a star-based rating system, a text box for comments, and a visible submission button. Integrated alongside the chatbot, this feature promotes user engagement and transparency by showcasing all reviews under the form, contributing to credibility and informed booking decisions.

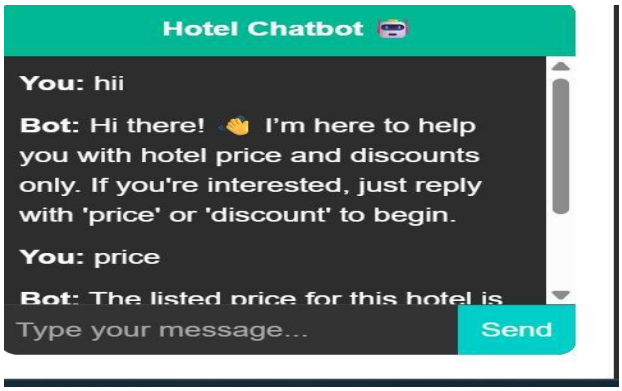


*Figure 5: Chatbot Interface.*

Figure 5 highlights the chatbot feature designed to assist users with hotel pricing and discounts. The bot initiates conversation in a friendly tone and guides users based on predefined prompts like 'price' or 'discount'. This interactive component enhances user support, delivering instant responses and improving the overall experience during the booking process.

## 5. Future Scope

While the current implementation of the hotel booking and listing web application fulfills essential functionalities and offers a seamless user experience, there is significant potential for future expansion and enhancement. With the growing demand for digital booking solutions in the hospitality sector, the following future advancements can be considered to improve scalability, usability, intelligence, and real-world applicability. The future enhancement of the hotel booking application involves integrating several advanced features to improve user experience, automation, and scalability. One key improvement is the integration of an online payment gateway. Currently, the platform lacks real-time payment functionality. Incorporating secure gateways such as Razorpay, Stripe, or PayPal would enable seamless in-app transactions. This enhancement would also support automated invoice generation, email-based payment confirmations, and efficient handling of refunds and cancellations. Another valuable addition is real-time availability and dynamic pricing. Implementing automated room availability checks and price adjustments based on seasonal trends, holidays, or occupancy rates would provide users with more accurate information. These updates could be managed through scheduled tasks (cron jobs) or external APIs. The application could also benefit from location-based services integration. Using tools like the Google Maps API, users would be able to search for nearby hotels, get directions, and filter listings based on proximity to landmarks or event venues, enhancing navigation and decision-making.

To improve user interaction, the chatbot can be upgraded with AI and machine learning capabilities. Enhancements like natural language understanding, personalized hotel recommendations, and multilingual support could be achieved using platforms such as Dialogflow or Rasa.

Lastly, a notification system featuring SMS and email alerts would streamline communication. It could confirm bookings, send reminders about check-in/check-out dates, and deliver personalized promotions, ensuring a more connected and engaging user experience.

## 6. Advantages

The proposed hotel booking and listing web application offers numerous advantages from both technical and user-centric perspectives. The application is designed with several key features that contribute to its functionality, security, and user engagement. The user-friendly interface, developed using HTML, CSS, JavaScript, and EJS, offers a clean, responsive, and intuitive design that ensures seamless navigation across devices, including desktops, tablets, and mobile phones. This enhances the overall user experience by making the platform accessible and easy to use. Dynamic data handling is achieved through the integration of MongoDB as the database and Express.js for backend routing. This allows real-time data fetching and display, enabling users to view updated hotel listings, reviews, and availability without the need to refresh pages, thereby improving responsiveness and efficiency.

Security is addressed through a secure authentication system. Passwords are hashed using bcrypt, and session-based login mechanisms are implemented. Route protection and middleware are used to restrict unauthorized access, ensuring both user privacy and system integrity.

The application also includes an integrated review and rating system, allowing only users who have completed bookings to post reviews. This verification builds trust and helps new users make informed decisions based on authentic experiences. Additionally, an interactive chatbot assists users by providing hotel information and answering booking-related queries, adding a personalized and engaging support element to the platform.

## 7. Conclusion

The development and implementation of the hotel booking and listing web application successfully demonstrate the practical integration of modern full-stack web technologies to solve real-world problems in the hospitality domain. The system was designed with a strong focus on modularity, responsiveness, user interactivity and functional completeness-offering features such as user registration, secure authentication, hotel listing and booking a dynamic review system and an intelligent chatbot assistant. Through the use of technologies like Node.js, Express.js, MongoDB and EJS, the project effectively applies the MVC architecture to build a scalable, maintainable and efficient web application.

The interactive chatbot feature and auto-generated bill interface. Moreover, this project highlights the importance of user-centered design, real-time dynamic data handling and secure backend operations in creating robust applications. It also serves as a strong learning foundation for students and developers aiming to explore the full-stack development process in-depth.

This project fills a practical gap between theoretical web development knowledge and real-world application by providing a working model that can be expanded with features like payment integration, location-based services, and advanced AI-driven recommendations.

## References

[1] M. Fowler, Patterns of Enterprise Application Architecture. Boston, MA: Addison-Wesley, 2003.

[2] E. Freeman and E. Robson, Head First Design Patterns, 1st ed. Sebastopol, CA: O'Reilly Media, 2004.

[3] I. Sommerville, Software Engineering, 10th ed. Boston, MA: Pearson, 2015.

[4] N. Tilkov and S. Vinoski, "Node.js: JavaScript to Build High-Performance Network Programs," IEEE Internet Computing, vol. 14, no. 6, pp. 80-83, Nov.-Dec. 2010, doi: 10.1109/MIC.2010.145.

***Cite this article as***: Tarun Verma, Virat, Saurabh Kumar, Tanishq Kumar, Amol Sharma, Hotel booking and listing, International Journal of Research in Engineering and Innovation Vol-9, Issue-4 (2025), 205-210.
*https://doi.org/10.36037/IJREI.2025.9409*