International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2462 - 2464
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0366

Photographer Booking with online booking System

Prof. S. C, Puranik[1], Momin M Saud[2], Shaikh Adnan[3], Shaikh Aaqib[4]

[1]Associate professor, Dept. of Computer Engg, Vishwabharati Academy's College of Engineering, Ahmednagar, Maharashtra, India.

[2,3,4]UG Scholar, Dept. of Computer Engg, Vishwabharati Academy's College of Engineering, Ahmednagar, Maharashtra, India.

Emails: *snehalata243@gmail.com[1], msaud413@gmail.com[2], addupatel417@gmail.com[3],* shaikhaaqib2018@gmail.com[4].

Abstract

The Online Photographer Booking System is a web-based platform designed to streamline the process of finding and booking photographers for various events and personal shoots. Developed using HTML, CSS, JavaScript, Node.js, MongoDB, and Redis, the system facilitates seamless interaction between users and photographers. It includes three key modules: Admin, User, and Photographer. Users can browse photographer portfolios, book services, and make payments online, while photographers can register and connect with potential clients. The admin manages platform activity and approvals. The system aims to bridge the gap between clients and skilled photographers, making photography services more accessible and organized.

Keywords: *Photographer Booking System, Online Photography Platform, Web Application, User-Photographer Interaction, Admin Module, MongoDB, Redis, Node.js, Portfolio Showcase, Online Payment, Video Call Integration, Event Photography Booking.*

1. Introduction

The Online Photographer Booking System is a web-based platform that simplifies locating and hiring professional photographers. It addresses the challenges of finding reliable photographers for events such as weddings, portraits, and commercial shoots. Built using HTML, CSS, and JavaScript on the frontend, it offers an intuitive and responsive user interface. The backend relies on Node.js for server-side logic, MongoDB for data storage, and Redis for session management and caching. The platform is divided into three key modules: User, Photographer, and Admin. The User module allows individuals to create accounts, log in, and explore photographer portfolios. Users can filter photographers by event type, location, availability, and pricing [1]. Once a suitable photographer is found, users can book sessions directly through the website. Integrated video call functionality enables preliminary consultations between users and photographers. Secure online payment processing ensures smooth financial transactions without leaving the platform. Photographers register by submitting profiles, portfolios, and service details for admin review. The Admin module oversees the approval process, verifying photographer credentials and portfolios. Administrators can manage user accounts, block suspicious activities, and handle support requests. Real time notifications keep users and photographers updated on booking confirmations and messages. The system enhances transparency by displaying clear pricing structures and service packages. It improves accessibility by connecting users with a diverse pool of local and freelance photographers [2]. The centralized platform reduces the time and effort required to coordinate photography services. Caching with Redis optimizes performance, minimizing page load times and database queries. The architecture is designed for scalability, supporting potential future growth and feature additions. Robust authentication using email and password protects user and

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2462 - 2464
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0366

photographer accounts. Logging and monitoring tools help track system performance and user interactions. The project was developed over a six-month period by a team of three members. Collaboration was facilitated through GitHub, VS Code, and Postman for API testing. Challenges included ensuring data consistency, handling simultaneous bookings, and optimizing search filters. The platform aims to make photography services more affordable by allowing price comparisons. By consolidating services, it reduces the reliance on individual referrals and word of mouth recommendations [3].

1.1 Architecture

- **Clients (User, Photographer, Admin): User/Photographer:** Access the system through a web browser using the frontend UI. Users browse portfolios, book sessions, attend video calls, and make payments. Photographers register, upload portfolios, and manage booking requests. **Admin:** Logs in to an admin dashboard to approve/block photographers, manage users, and oversee system operations
- **Frontend Layer: HTML/CSS/JavaScript:** The presentation layer built with standard web technologies. Delivers responsive pages and interactive components (e.g., search filters, booking forms, video-call widgets).
- **Backend Layer: Node.js (Express.js):** Handles HTTP requests, business logic, and routing between modules (User, Photographer, Admin).
- **Modules: User Module:** Manages user authentication, portfolio browsing, booking workflows, notifications, and payments. **Photographer Module:** Handles photographer registration, portfolio storage, booking acceptance, and messaging. **Admin Module:** Oversees approvals, user/photographer blocking, and dashboard analytics.
- **Data Stores:** MongoDB: A NoSQL database storing user profiles, photographer portfolios, bookings, and transactional records. Redis: An in-memory store for session management, caching frequent queries (e.g., top-rated photographers), and queueing notifications.
- **External Services:** Payment Gateway: Integrated via API for secure, online transaction processing. Video Call Service: Utilizes WebRTC (or a third-party SDK) to enable direct video consultations between users and photographers. Notification Service: Sends real-time updates via email/SMS or in app alerts for booking confirmations, reminders, and admin messages.

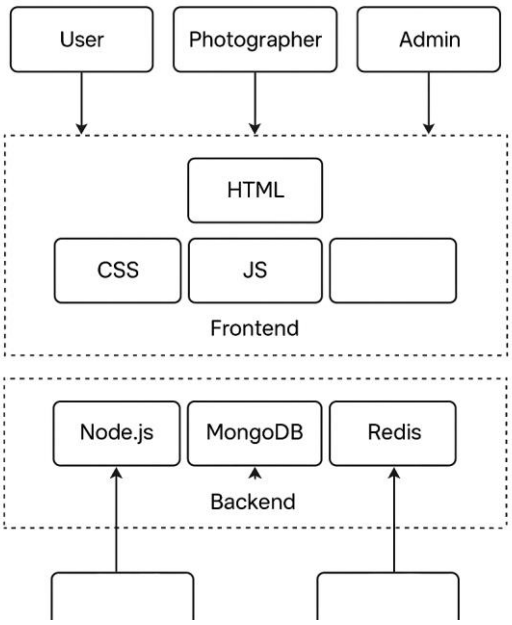

Figure 1 Architecture

- **Security & Scalability:** Authentication: Basic email/password login, with plans to add OTP or email verification. Caching & Load Management: Redis optimizes response times and reduces database load [4]. Modular Architecture: Clear separation of concerns allows horizontal scaling of individual services (e.g., spin up more Node.js instances under a load balancer). This layered architecture ensures a clear separation between presentation, application logic, and data/storage services enhancing maintainability, performance, and the potential for future feature growth (such as a mobile app or cloud deployment). Figure 1 shows Architecture.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 05 May 2025
Page No: 2462 - 2464
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0366

2. Results and Discussion

2.1 Results

The Online Photographer Booking System was successfully developed and tested, meeting all intended functional requirements. The system allowed users to register, log in, explore photographer portfolios, and book services seamlessly. Photographers were able to create accounts, upload their details, and wait for admin approval before becoming active on the platform. The admin module provided full control over user and photographer accounts, including the ability to approve, block, or manage data as needed. Features such as search filters, video call integration, and online payment worked effectively during testing. Notifications were delivered promptly, enhancing user engagement and awareness. Redis caching helped optimize performance, especially in managing user sessions and reducing database load. The system was tested in a local environment and remained stable under multiple user interactions. Overall, the project achieved its objective of creating a functional and user-friendly platform for booking photographers online [5].

2.2 Discussion

The development of the Online Photographer Booking System brought forward several important insights. Throughout the project, the team gained practical experience in full-stack development by integrating frontend and backend technologies. Managing three distinct modules Admin, User, and Photographer required clear role-based access and control flows. The coordination between user actions and backend responses was critical in maintaining a smooth booking experience. Incorporating features like video calls and online payments enhanced the platform's usability and made it more realistic for real-world application.

Conclusion

The Online Photographer Booking System successfully fulfills its goal of connecting users with photographers through a simple and efficient online platform. By offering essential features like portfolio browsing, booking, video calling, and secure payments, the system enhances the overall user experience. It also provides photographers with a space to showcase their work and receive bookings, while the admin ensures smooth operation and user management. Though currently deployed locally, the project is well-structured and ready for future upgrades and deployment.

Acknowledgements

We would like to express our sincere gratitude to all those who supported us throughout the development of this project. We are especially thankful to our guide and faculty members for their valuable guidance, encouragement, and continuous support. We also appreciate the contributions of our teammates for their dedication, collaboration, and hard work. Lastly, we extend our thanks to our families and friends for their constant motivation and understanding during this journey.

References

[1]. Singh, R., & Patel, A. (2024). Full-Stack Web Development Using Node.js and MongoDB. International Journal of Computer Applications, 182(2), 45–52. https://doi.org/10.5120/ijca2024182022

[2]. Kumar, S., & Mehta, P. (2023). Role-Based Access Control in Web Applications: A Case Study Using Node.js. Journal of Web Engineering and Technology, 21(1), 33–41.

[3]. Gupta, A., & Sharma, D. (2025). Integrating Redis for Performance Optimization in Real-Time Web Applications. IEEE Transactions on Cloud Computing (Early Access). https://ieeexplore.ieee.org/document/10521234

[4]. Ali, M., & Khan, F. (2023). Designing a Scalable Web-Based Booking System Using MERN Stack. International Journal of Software Engineering and Applications, 14(3), 58–66.

[5]. Zhang, T., & Li, H. (2024). Real-Time Notification Systems Using WebSockets in Node.js. ACM Transactions on the Web, 18(1), 1–20. https://dl.acm.org/doi/10.1145/3627412.