



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85067>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

ApexMed: Doctor Appointment and Management System

Muzammil Ahmad², Aman Hidami³, Sanket Weladi⁴, Adil jatu⁵, Vansh Leley⁷, Mohit Sahu⁶, Mr. Amol Dhankar¹

Department of Computer Science and Engineering G H Raisoni University, Amravati, Maharashtra, India

Abstract: Healthcare appointment management can be improved through web-based systems that centralize doctor discovery, availability, appointment scheduling, patient records, and administrative operations. This paper presents ApexMed, a full-stack doctor appointment and management system developed using React.js, Vite, Tailwind CSS, Node.js, Express.js, MongoDB, and Mongoose. The system provides patient, doctor, and administrator roles with separate authentication and access privileges. Major functions include patient registration and login, doctor search, specialization filtering, doctor availability, appointment booking, cancellation, appointment history, profile management, dashboards, administrator-controlled doctor management, and a simulated payment workflow. JWT is used for authentication and role-based authorization, while Bcrypt protects passwords. Multer and Cloudinary are used for profile-image handling. The literature review indicates that recent healthcare appointment systems increasingly use MERN/full-stack architectures and extend appointment management with AI, OCR, payment, and notification features. ApexMed adopts a focused architecture for core appointment operations while remaining extensible for future healthcare capabilities.

Index Terms: Doctor appointment, MERN stack, React.js, Node.js, Express.js, MongoDB, JWT, healthcare management, role-based authentication, simulated payment.

I. INTRODUCTION

Digital healthcare platforms are increasingly being used to simplify interactions between patients, doctors, and healthcare administrators. Appointment scheduling is one of the important areas where a web application can reduce dependence on manual booking and telephone communication. A centralized system can allow patients to discover doctors, identify specializations, check availability, select an appointment slot, and track appointment information.

Recent research demonstrates the suitability of MERN and other full-stack architectures for healthcare appointment applications. Existing systems have explored online doctor search, availability, appointment booking, cancellation, dashboards, payment integration, AI consultation, and prescription recognition. The reviewed literature shows that healthcare appointment systems can be implemented as modular web applications while supporting multiple user roles.

ApexMed was developed as a focused doctor appointment and management platform. The system concentrates on the core workflow of patient appointment management while also providing separate doctor and administrator functions. The project uses React.js for the user interface, Node.js and Express.js for server-side operations, and MongoDB for persistent data storage.

II. LITERATURE REVIEW

- 1) Smart Doctor Appointment Booking and Patient Management System: Das, Sahoo and Sarangi presented a MERN-based system covering appointment scheduling, patient management, doctor availability and healthcare communication. The work demonstrates the suitability of a three-tier MERN architecture for digital appointment management.
- 2) AI-Powered Online Doctor Consultation and Booking System: Singh, Pandey and Deepak proposed a MERN-based system that combines appointment booking and role-based dashboards with AI-assisted consultation functionality. The study illustrates how AI can extend a conventional appointment platform, although such functionality is outside the current ApexMed scope.
- 3) Smart Healthcare Appointment System: Suresh and Yuvaraj described a MERN-based healthcare appointment system supporting booking, rescheduling, cancellation, doctor availability, dashboards, payment integration, notifications and related healthcare functions.
- 4) Full Stack Doctor Appointment Booking System: Patil et al. developed a full-stack platform connecting patients, doctors and administrators through doctor search, appointment booking, history and management functions. This work is closely related to the core purpose of ApexMed.

- 5) Doctor Appointment Booking and Handwriting Recognition System: Nandwalkar et al. combined appointment booking with handwritten prescription recognition using a CNN/OCR approach. The additional intelligent document-processing component distinguishes the work from the focused appointment scope of ApexMed.
- 6) A Full-Stack MERN Hospital Management System: Praharaj, Parida and Mahanta presented a broader MERN hospital-management platform integrating appointment operations with payment, communication, cloud storage and AI-assisted documentation.

III. RESEARCH GAP

The reviewed studies cover a wide range of healthcare functions, from basic appointment scheduling to AI consultation, OCR, payment integration and broader hospital management. A practical gap remains for a focused academic platform that clearly separates patient, doctor and administrator workflows while implementing the essential appointment lifecycle without depending on advanced AI or production payment services. ApexMed addresses this requirement through three role-specific interfaces, JWT-based authentication, specialization-based doctor search, availability management, appointment booking and cancellation, appointment history, profile management, administrator-controlled doctor creation, cloud image storage, and a simulated payment workflow. This focused architecture also leaves a clear path for future extensions.

IV. PROPOSED SYSTEM

The proposed ApexMed system consists of three main user roles: Patient, Doctor and Administrator. Patients can register, authenticate, search doctors, check specialization and availability, book appointments, use the simulated payment flow, view appointment history, cancel appointments, and manage their profile. Doctors authenticate through a dedicated portal, manage their profile and availability, and access their appointment dashboard. Administrators manage doctor accounts and related system information.

Patient / Doctor / Admin
React.js + Vite + Tailwind CSS
REST API / Node.js + Express.js
JWT + Role-Based Authorization
MongoDB + Mongoose
Multer + Cloudinary (profile images)
Mock / Simulated Payment

Figure 1: Proposed ApexMed system architecture

V. METHODOLOGY

The system was developed in successive stages. Requirement analysis identified the required patient, doctor and administrator workflows. System design separated the presentation, application, database, authentication, cloud-storage and appointment components. React.js and Vite were used for the frontend, while Node.js and Express.js were used for REST API development. MongoDB with Mongoose was selected for persistent storage. Authentication and authorization were then integrated using JWT and role-specific middleware. Multer and Cloudinary were integrated for image uploads. Finally, the major appointment and profile workflows were tested through the application interfaces.

VI. SYSTEM MODULES

- 1) Patient Registration and Login: Patients create accounts and authenticate through the user login interface. Passwords are hashed using Bcrypt before storage. Successful authentication results in a JWT used for protected requests.
- 2) Doctor Login and Management: Doctor accounts are created and managed through the administrator interface. Doctors use a dedicated login and can manage profile information, availability and appointment-related dashboard functions.
- 3) Administrator Management: Administrators authenticate through a separate portal and manage doctor accounts, including doctor creation and doctor-list operations.
- 4) Doctor Search and Specialization: Patients can browse doctors and filter them by specialization. Doctor profile information and availability are displayed before booking.

- 5) Availability Management: Doctor availability is maintained through the doctor profile/dashboard workflow so that patients can view the current availability state before requesting an appointment.
- 6) Appointment Booking: Patients select a doctor, date and time and submit the booking request through a protected REST API. The appointment is stored in MongoDB and associated with the patient and doctor.
- 7) Appointment Cancellation and History: Patients can cancel appointments and retrieve their appointment records through the My Appointments interface. Appointment state is maintained in the database rather than requiring manual deletion.
- 8) Profile Management: Separate patient and doctor profile interfaces allow authenticated users to retrieve and update their information.

VII. IMPLEMENTATION

- 1) Frontend: React.js provides component-based development, Vite provides the application build environment, Tailwind CSS is used for responsive styling, React Router manages navigation, and Axios communicates with backend APIs.
- 2) Backend: Node.js and Express.js implement REST APIs, controllers, authentication middleware, appointment operations and profile operations. The backend separates user, doctor and administrator responsibilities.
- 3) Database: MongoDB stores User, Doctor and Appointment documents. Mongoose provides schema-based access and simplifies communication between the application and MongoDB.

VIII. JWT AUTHENTICATION AND SECURITY

Security is implemented using JSON Web Tokens. The application separates authentication middleware into user, doctor and administrator authorization functions. After successful login, a token is generated and used for protected requests. The server verifies the token before executing role-specific operations.

Passwords are protected using Bcrypt hashing. Role-based authorization reduces unauthorized access by preventing a patient from directly using doctor or administrator functions.

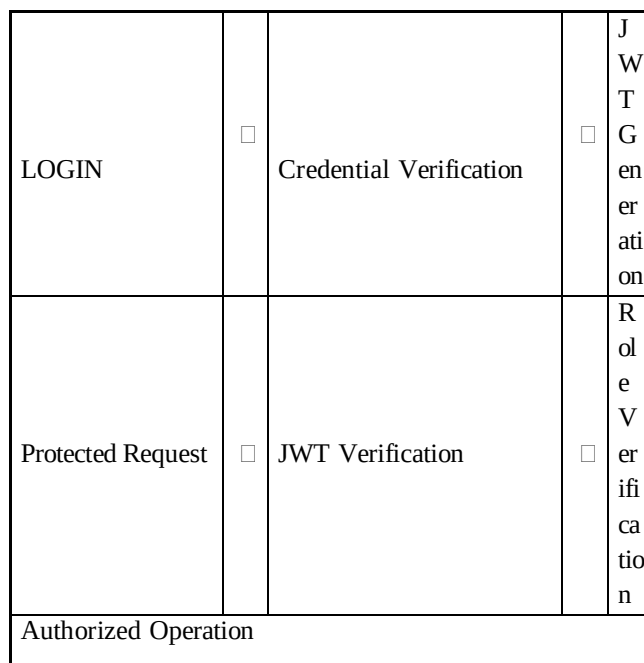


Figure 2: JWT authentication and authorization flow

The architecture provides an application-level security boundary; production deployment would additionally require HTTPS, secure token storage, rate limiting, stronger input validation, audit logging and healthcare-specific privacy controls.

IX. DATABASE DESIGN

The database contains three major entities. The User entity stores patient account and profile information. The Doctor entity stores doctor identity, contact information, specialization, experience, password, image information and availability. The Appointment entity connects a patient and doctor and stores date, time, status, cancellation, completion and payment state.

This model supports the appointment lifecycle without requiring separate records for every user action. Appointment state fields allow the system to distinguish active, cancelled, completed and payment-related states.

X. APPOINTMENT WORKFLOW

1. Login / Authentication
↓
2. Search Doctor
↓
3. Check Specialization & Availability
↓
4. Select Date / Time
↓
5. Book Appointment
↓
6. Mock Payment
↓
7. Appointment History / Cancellation

Figure 3: Patient appointment workflow

XI. SIMULATED PAYMENT MODULE

The current ApexMed implementation contains a simulated payment method rather than a live Razorpay or Stripe integration. The purpose is to demonstrate the payment state transition within the academic project. After an appointment is created, the user can initiate the mock payment operation, after which the appointment payment state is updated. This implementation should be described as simulated/mock payment in the research paper and project demonstration. A future production version can replace the mock endpoint with a verified payment gateway.

XII. IMAGE UPLOAD AND CLOUD STORAGE

Multer is used to process uploaded profile images and Cloudinary provides cloud-based image storage. The resulting cloud image reference is associated with the corresponding user or doctor record. This separates image storage from the application server and supports profile-image retrieval through the web interface.

XIII. RESULTS AND DISCUSSION

The implemented ApexMed application demonstrates the major functional workflow required for an online doctor appointment platform. Patients can register and authenticate, browse doctors, filter by specialization, view availability, create appointments, access appointment history, cancel appointments, and manage their profile. Doctors can authenticate, access their dashboard, update their profile and change availability. Administrators can authenticate and manage doctor accounts.

The application also demonstrates JWT-based role separation, Bcrypt password protection, MongoDB persistence, cloud image storage, and the simulated payment state. These components form an integrated full-stack workflow rather than isolated demonstration pages. No numerical response-time or scalability claims are made here because verified benchmark measurements for ApexMed were not supplied. Such measurements should be added only after controlled testing using defined datasets and workloads.

XIV. COMPARISON WITH EXISTING SYSTEMS

Feature	ApexMed	Literature Trend
Doctor search	Yes	Common
Specialization filter	Yes	Common
Availability	Yes	Common
Booking/cancellation	Yes	Common
Role-based login	Yes	Common
Cloud image storage	Yes	Used in some systems
Live payment gateway	No; simulated	Used in some systems
AI/OCR	No	Present in some studies

Table 1: Functional comparison of ApexMed with major literature trends

XV. ADVANTAGES

The system provides centralized appointment management, reduced dependence on manual scheduling, role-based access, doctor specialization search, availability information, appointment history, profile management, cloud image storage and a modular architecture. Because frontend, backend, database, authentication and storage responsibilities are separated, the system can be extended without redesigning the complete application.

XVI. LIMITATIONS

The present implementation uses simulated payment rather than a production gateway. It does not currently implement AI diagnosis, AI consultation, prescription OCR or a complete hospital-management suite. Large-scale performance benchmarking has not been established. Production healthcare deployment would also require stronger privacy, security, audit and compliance mechanisms.

XVII. FUTURE WORK

Future development may include live payment integration, email/SMS appointment notifications, AI-based appointment assistance, telemedicine/video consultation, prescription management, OCR-based prescription recognition, location-based doctor search, doctor ratings, analytics, and advanced security controls. These extensions can be added while preserving the existing role-based architecture.

XVIII. CONCLUSION

ApexMed presents a focused full-stack solution for doctor appointment and management operations. The system combines React.js, Node.js, Express.js, MongoDB, JWT authentication, Bcrypt, Multer and Cloudinary to provide patient, doctor and administrator workflows. The implementation supports doctor discovery, specialization filtering, availability, appointment booking, cancellation, appointment history, profile management, doctor administration and simulated payment.

The literature review shows that modern healthcare appointment systems are moving toward integrated full-stack platforms with additional capabilities such as AI, OCR, payment and communication services. ApexMed provides a practical foundation for these future enhancements while maintaining a clear and manageable academic scope.

REFERENCES

- [1] A. Das, S. Sahoo and S. S. Sarangi, "Smart Doctor Appointment Booking and Patient Management System Using MERN Stack," American Journal of AI Cyber Computing Management, vol. 6, no. 2(2), pp. 112–119, 2026.
- [2] S. Singh, P. Pandey and Deepak, "Design and Development of an AI-Powered Online Doctor Consultation and Booking System," International Journal of Scientific Research in Science, Engineering and Technology, vol. 13, no. 1, pp. 262–279, 2026.
- [3] M. R. Sukesh and V. Yuvaraj, "Smart Healthcare Appointment System," International Journal of Research

- [4] Publication and Reviews, vol. 6, no. 3, pp. 7618–7623, 2025.
- [5] V. Patil, A. Sinha, S. Anand, S. Priya and A. A. Sayyad, “Full Stack Doctor Appointment Booking System,” Open Access International Journal of Science and Engineering, vol. 9, no. 1s, pp. 161–174, 2026.
- [6] J. R. Nandwalkar, A. K. Bokde, P. A. Adke and R. S. Jethé, “Doctor Appointment Booking and Handwriting Recognition System,” International Journal of Innovative Science and Research Technology, vol. 10, no. 4, pp. 1955–1962, 2025.
- [7] S. Praharaj, A. Parida and P. K. Mahanta, “A Full-Stack MERN Hospital Management System,” International Journal of AI eBiomedicine Innovations, vol. 2, no. 2, pp. 177–182, 2026.

APPENDIX: APEXMED IMPLEMENTATION SUMMARY

The following implementation summary records the project-specific technologies and modules used in the current ApexMed version.

Frontend: React.js, Vite, Tailwind CSS, React Router, Axios.

Backend: Node.js, Express.js, REST APIs.

Database: MongoDB with Mongoose. **Authentication:** JWT with separate user, doctor and administrator authorization middleware.

Password security: Bcrypt hashing.

File handling: Multer for uploads and Cloudinary for image storage.

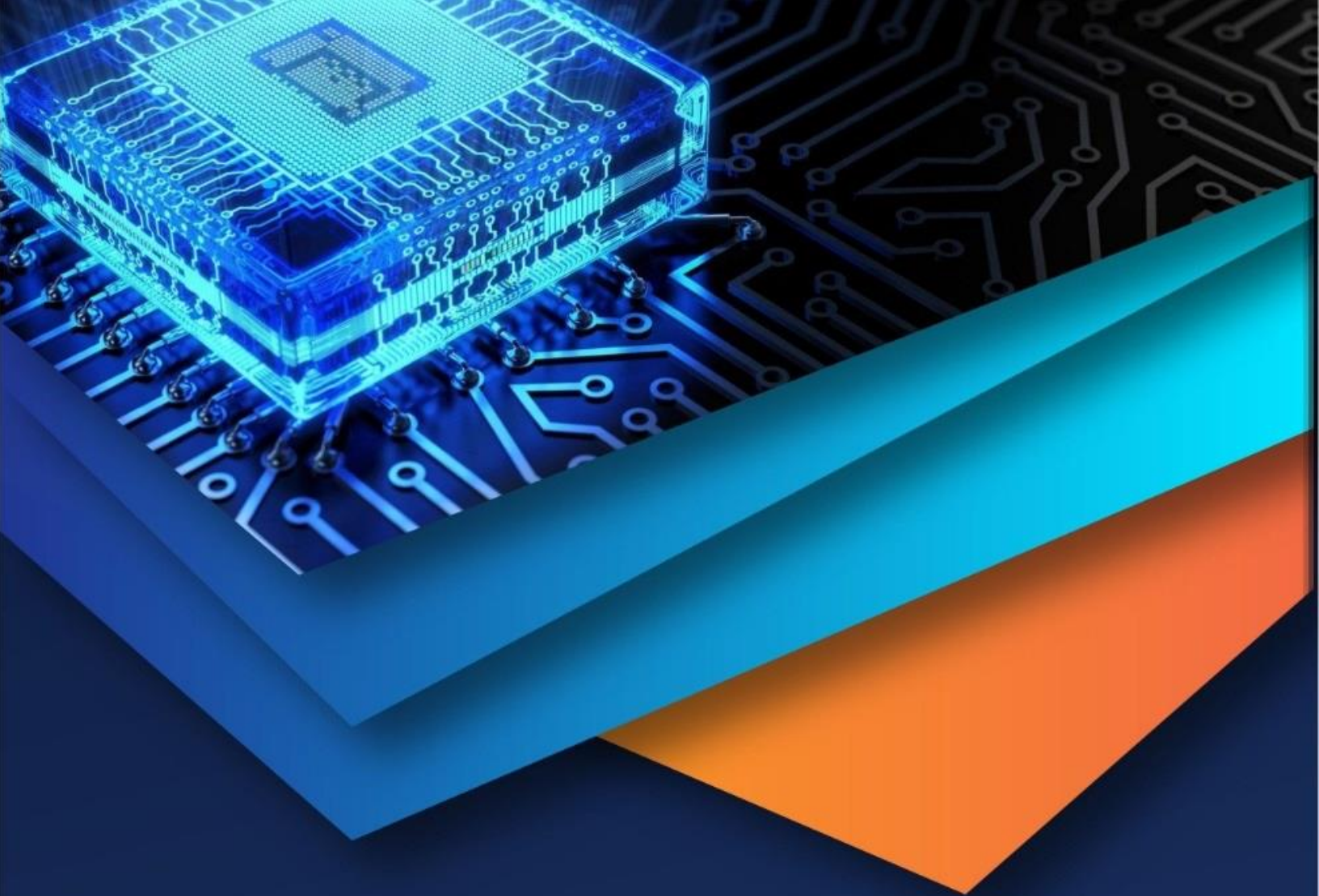
Payment: Simulated/mock payment workflow; no live Razorpay/Stripe transaction is claimed.

Primary entities: User, Doctor and Appointment.

Primary roles: Patient, Doctor and Administrator.

Key API operations: Separate REST operations are used for user registration and login, doctor login, administrator login, doctor listing, appointment booking, appointment cancellation, appointment history, profile retrieval and update, doctor profile management, availability change, administrator doctor creation and doctor listing, and mock payment state update.

Project workflow: Patient ☐ Login ☐ Search Doctor ☐ Specialization ☐ Availability ☐ Appointment ☐ Mock Payment ☐ Appointment History. Doctor ☐ Login ☐ Dashboard ☐ Profile ☐ Availability ☐ Appointment Management. Admin ☐ Login ☐ Dashboard ☐ Add Doctor ☐ Doctors List ☐ Doctor Management.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)