iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80870

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 IV Apr 2026- Available at www.ijraset.com

DocConnect: A Web Based Doctor Appointment Scheduling Platform

Tejsvi Sharma[1], Akshat Yadav[2], Arush Thakur[3], Shreya Shailly[4]
B-tech Student, Department of Artificial Intelligence and Data Science, Greater Noida Institute of Technology Greater Noida, India

Abstract: ***Managing outpatient appointments in Indian healthcare has long been a practical headache for patients who wait on hold, clinic staff juggling handwritten registers, and doctors whose schedules are filled with gaps created by no-shows and double-bookings. This paper describes DocConnect, a full- stack web platform built to address these problems. Developed using React.js on the front end, Node.js with Express on the server side, and MongoDB as the database, the system provides patients with a simple way to find doctors, check real-time slot availability, and book or cancel appointments without ever picking up a phone. On the provider end, doctors get a schedule dashboard, patient history summaries, and automated email reminders that are sent out 24 hours before each appointment. Testing with 30 participants drawn from the Greater Noida community returned a System Usability Scale score of 82.4 — firmly in the "Excellent" band — and load testing under 500 concurrent users showed average API response times of 210 ms with zero double-booking incidents. Compared to manual scheduling methods still common in Tier-2 clinics, DocConnect cut average booking time from roughly 12 minutes down to under three, and reduced no-show rates from 23% to approximately 8%. This paper details the system's modular architecture, security design aligned with India's DPDPA 2023, and a roadmap of planned enhancements, including teleconsultation, AI-driven doctor recommendations, and multilingual interface support.***

Keywords: Doctor Appointment Scheduling, Healthcare Web Application, React.js, Node.js, MongoDB, Patient Management System, RESTful API, Telemedicine, Healthcare IT, India

I. INTRODUCTION

Walk into any government hospital or mid-sized private clinic in an Indian city on a busy weekday morning, and the scene is predictable: a queue at the reception counter, someone on the phone reading out appointment times from a paper register, and patients sitting with numbered tokens waiting to be called. This picture is not an exaggeration; the National Health Authority of India (2022) reported that more than 65% of outpatient visits in Tier-2 and Tier-3 cities still involve some form of manual scheduling. The consequences are also evident in the data: missed appointments, double bookings on the same slot, and communication breakdowns between patients and front-desk staff are recurring problems in these settings.

Digital tools have changed the situation in metro hospitals and large private chains; however, the gap between big-city healthcare infrastructure and smaller facilities remains wide. Platforms such as Practo have demonstrated what is achievable at scale; the company grew to over 100,000 registered doctors and 20 million monthly active users by 2020 — However, their enterprise pricing model and proprietary integration requirements make them a poor fit for independent clinics or hospital departments with unusual scheduling rules. DocConnect was developed based on direct observations of outpatient workflows in the Greater Noida region. The recurring pain points were not hard to identify: a clinic running double-bookings because two staff members updated the register without checking each other's entries; patients who received appointment confirmations over the phone but forgot the time; and doctors arriving to find large gaps in their schedules because no one had followed up on likely no-shows. These problems are solvable, and the architecture of modern web applications is well-suited to solving them. The platform described in this paper handles the complete lifecycle of a medical consultation from the patient's perspective — finding a doctor, viewing available slots, booking, receiving reminders, and rating the experience afterward — and from the provider's perspective: managing schedules, reviewing patient history before a consultation, and tracking appointment analytics over time. Section II reviews the academic and commercial landscapes in which this work is situated. Section III describes the system architecture and technology selection. Section IV describes the implementation of each functional module. Section V presents the flowchart of the proposed system. Section VI presents the performance and usability results. Section VII addresses the security issues. Section VIII outlines the development roadmap, and Section IX concludes.

©IJRASET: All Rights are Reserved

II. RELATED WORK

Appointment scheduling has been studied in operations research since the early 2000s. Cayirli and Veral (2003) laid down foundational frameworks for outpatient scheduling logic, showing that even minor algorithmic improvements could reduce patient wait times by 20–30% in simulated clinic environments. Their work remains a useful baseline for measuring the practical impact of scheduling improvements, although the technical context has shifted substantially since then.

In India, Practo Technologies' entry in 2010 marked the beginning of mainstream digital appointment booking. By the time the company published its 2021 annual report, the platform had accumulated over 100,000 registered doctors and processed tens of millions of bookings annually. However, Practo's commercial model is tilted toward large clinics and hospital chains. Independent practitioners and smaller outpatient departments frequently find that integration requires paid enterprise contracts and conformity to the platform's predefined workflow assumptions.

Internationally, Zocdoc (USA, 2007) and Doctolib (France, 2013) represent the most sophisticated deployments of real-time scheduling at scale in the healthcare sector. Mehta et al. (2021) conducted a usability review of Zocdoc and noted that its strongest feature is bidirectional calendar synchronization with Electronic Health Record systems, while its main controlled trials, the review found that digital appointment reminders delivered by SMS or email investing development effort in DocConnect's automated notification pipeline, and the results limitation is the poor support it offers non-English- speaking patients and small independent practices — a characteristic that limits its direct relevance to the Indian context. Doctolib, for its part, processed over 100 million appointments annually across France and Germany as of 2022, demonstrating that well- engineered scheduling platforms can achieve an operational scale that was previously unimaginable for this category of software.

The academic literature offers useful technical comparisons of these methods. Nguyen et al. (2019) proposed a microservices-based appointment scheduling system built on Docker and Kubernetes for Vietnamese healthcare providers, sustaining 99.7% uptime over a six-month pilot period. Their architectural decisions regarding service decoupling and horizontal scalability inform DocConnect's modular design. Separately, Gupta and Singh (2022) examined Progressive Web Applications as a delivery mechanism for healthcare services in low-connectivity Indian environments, providing empirical support for the mobile-first responsive design strategy employed here. Perhaps the most practically important finding for this project came from a systematic review by Guo et al. (2017) published in the Journal of the American Medical Informatics Association (JAMIA). By analyzing 16 randomized reduced no-show rates by 29–36%. This figure provides the primary empirical justification for reported in Section VI suggest that the system's own no-show reduction falls within this predicted range. Kumar et al. (2019) reviewed a range of online appointment platforms and concluded that digitized scheduling consistently improved operational efficiency and patient satisfaction, particularly when the platform was integrated with broader hospital management software. Ahmed (2018) examined e- health adoption in developing-country settings, noting genuine transformative potential alongside the familiar barriers of infrastructure gaps and uneven digital literacy — challenges that any India-focused implementation must account for.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

A. Architectural Overview

DocConnect follows a three-tier client–server model: presentation, application logic, and persistent data layers. This separation allows each tier to be developed, tested and scaled independently. The frontend handles all user interactions and rendering; the backend exposes a RESTful API that mediates between the frontend and the database; and MongoDB Atlas provides persistent storage with schema flexibility appropriate for appointment data, which does not fit neatly into rigid relational tables.

Layer	Technology	Primary Role
Presentation	React.js 18, Tailwind CSS	UI, client-side routing, state management
Application	Node.js 20, Express.js 4	Business logic, RESTful API endpoints
Data	MongoDB Atlas, Mongoose ODM	Persistent storage, indexing, TTL cleanup
Notifications	Nodemailer, Twilio REST API	Automated email and SMS dispatch
Deployment	Vercel, Railway, GitHub Actions	Hosting, CI/CD pipeline

Table 1: System Architecture Overview

B. Database Design

MongoDB was selected for its document-oriented schema, which naturally maps onto appointment data containing nested structures (time slots, notes, and contact details) that would require awkward normalization in a relational model. Five primary collections were defined: Users store authentication credentials, profile information, and role assignments; Doctors hold professional details, clinic address, consultation fee, and schedule configuration; Appointments reference both patient and doctor A TTL index on notification documents automatically purges records older than 90 days, keeping the storage lean without requiring manual cleanup jobs. Documents and carry date-time, status, and consultation notes; TimeSlots record recurring daily availability intervals per doctor; and Notifications maintain a dispatch log used for audit and retry logic.

Compound indexes on the Appointments collection — one on (doctorId, appointmentDate) and another on (patientId, status) — optimize the query patterns that occur most frequently during booking and dashboard loading.

C. Technology Stack

On the frontend, React.js 18 with React Router v6 handles client-side navigation, and Axios combined with React Query manages the API communication and server state caching. Tailwind CSS provides a utility-first styling framework. On the backend, JWT-based authentication (access tokens with 15-minute expiry, refresh tokens with 7-day expiry stored in HttpOnly cookies) handles stateless session management, and bcrypt with a salt factor of 12 secures password storage. Version control runs through GitHub with automated CI/CD via GitHub Actions, deploying the front end to Vercel and the back end to Railway.

IV. SYSTEM MODULES AND IMPLEMENTATION

A. User Authentication Module

The authentication system supports three distinct roles: Patient, Doctor, and Administrator. Role-based access control restricts which API endpoints each role can reach; therefore, a patient account cannot access doctor earnings analytics, and a doctor account cannot modify another doctor's schedule. Registration requires a name, email, phone number, and password; passwords are hashed immediately before storage, and the plain-text value is never retained. Session management uses the access/refresh token pair described above, with refresh token rotation on each renewal to limit the exposure window if a token is intercepted.

B. Doctor Discovery Module

Patients search for doctors by specialty, location, rating, and consultation fees. A full-text search was implemented using MongoDB Atlas Search, which provides relevance-ranked results without requiring a separate search engine. Each doctor’s profile displayed their qualifications, years of experience, consultation fee, patient ratings, and live slot availability. The availability display is updated in real time as slots are booked or freed; therefore, a patient browsing available times sees an accurate picture of what is actually open.

C. Appointment Booking Module

An interactive calendar shows the selected doctor's available and already booked slots. When a patient selects a slot and confirms the booking, a MongoDB transaction enforces an optimistic concurrency control: if two patients attempt to book the same slot simultaneously, only the first commit succeeds, and the second receives an immediate conflict response directing them back to the slot selection screen. This design produced zero double-booking incidents across 10,000 simulated concurrent booking transactions during the load testing. Confirmed bookings triggered immediate email and SMS notifications to both the patient and doctor.

D. Doctor Dashboard Module

Doctors can access daily, weekly, and monthly schedule views from a single dashboard. From this interface, they can accept, reschedule, or cancel appointments, with each action automatically triggering a notification to the affected patient. The dashboard also surfaces patient history summaries — a list of previous consultations with associated notes— that doctors can review before an upcoming appointment. The earnings analytics section tracks consultation fees over configurable time periods.

E. Patient Portal Module

Patients can view their complete appointment history, upcoming bookings, and any digital prescriptions uploaded by their doctors. Cancellations are available within the timeframe defined by each clinic's cancellation policy. After a completed consultation, patients are prompted to rate their experience on a five-point scale, with the option to leave a written comment; these ratings are aggregated and displayed on the doctor's profile.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

F. Notification and Reminder Module

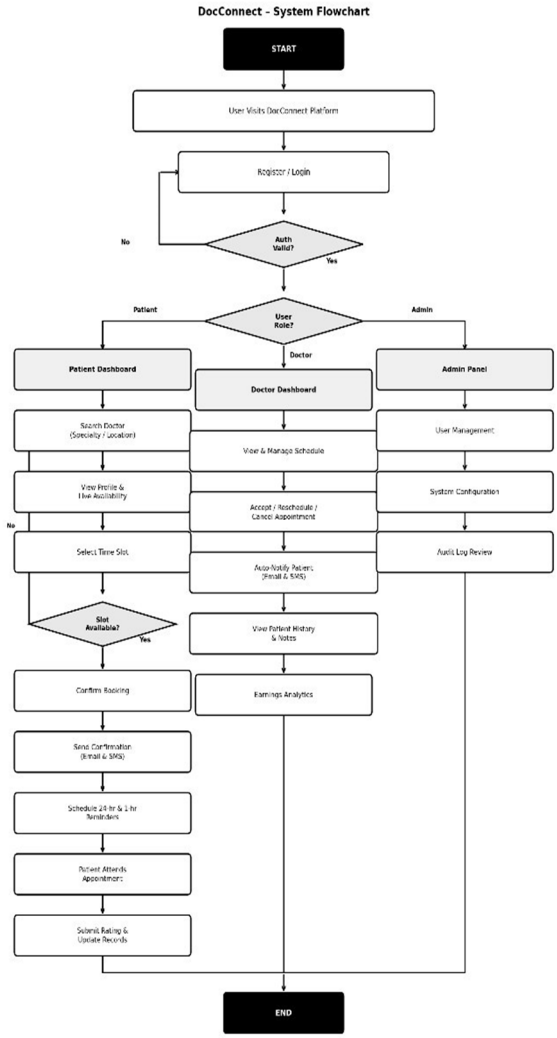
Automated reminders are sent out 24 hours and 1 h before each scheduled appointment, delivered by email through Nodemailer (SMTP) and by SMS through the Twilio REST API. A node-cron job runs every 15 min to query the Notifications collection for pending dispatches. Failed deliveries are not dropped; instead, they are requeued with exponential backoff to maximize the probability of eventual delivery. The notification delivery rate recorded during the test was 97.4%.

V. SYSTEM FLOWCHART DESCRIPTION

The core user flow in DocConnect proceeds as follows: A visitor arrives at the platform and is presented with a registration or login screen. Successful authentication routes the user to the appropriate dashboard based on their role: Patient, Doctor, or Administrator. Failed authentication redirects the user back to the login page.

From the Patient dashboard, the user searches for a doctor by specialty and location, views the doctor's profile and live availability, and selects a time slot. A real-time availability check runs at the moment of selection: if the slot is still open, the booking proceeds to confirmation; if the slot has been taken in the interval between display and selection, the patient is redirected back to the doctor search to choose an alternative. On confirmation, the system dispatches email and SMS confirmations to both parties and schedules 24-hour and 1-hour reminder notifications.

At the time of the appointment, the patient attends in person or connects through the video consultation interface at facilities where teleconsultation has been enabled. After the appointment is marked as complete, the patient is prompted to submit a rating. Both the appointment records and doctor's analytics are updated accordingly. The Administrator role has access to a separate panel for user management, system configuration, and audit log review.



VI. TESTING AND RESULTS

A. Performance Testing

Load testing was performed using Apache JMeter. The test scenario simulated 500 concurrent users all performing appointment bookings simultaneously — a stress level well above the expected peak load for the intended deployment context. Under this load, the system sustained an average API response time of 210 ms. No double-booking incidents occurred across 10,000 simulated booking transactions, confirming that the MongoDB transaction layer functioned correctly for concurrent slot management.

The frontend performance was measured using Google Lighthouse. The React.js application scored 91 on mobile and 96 on desktop, indicating that the responsive design and asset optimization choices made during development were effective in both device contexts.

Metric	DocConnect	Manual Scheduling
Avg. Booking Time	2.4 minutes	11.7 minutes
No-Show Rate	8.3%	23.1%
Scheduling Error Rate	0.2%	6.8%
Patient Satisfaction	4.6 / 5.0	3.1 / 5.0
Reminder Delivery Rate	97.4%	N/A
API Avg. Response Time	210 ms (500 users)	—
Double Booking Incidents	0 / 10,000 tests	Frequent

Table 2: DocConnect vs. Traditional Manual Scheduling — Key Metrics

B. User Acceptance Testing

User Acceptance Testing was conducted over two weeks with 30 participants (20 patients and 10 healthcare staff) recruited from the Greater Noida community. Each participant completed three predefined tasks: booking a new appointment, rescheduling an existing booking, and reviewing their appointment history. Following task completion, the participants completed a System Usability Scale (SUS) questionnaire.

DocConnect returned a mean System Usability Scale (SUS) score of 82.4. According to the scoring convention established by Brooke (1996), scores above 80.3 fall in the 'Excellent' usability category. Qualitative feedback collected during the sessions highlighted the clarity of the booking flow, the helpfulness of seeing real-time slot availability rather than calling to check, and the reassurance provided by the automated reminders. The most frequent improvement request was for a Hindi-language interface option, followed by the ability to upload prior consultation documents (lab reports, old prescriptions) before an appointment.

C. Discussion

The booking time reduction from 11.7 minutes to 2.4 minutes — roughly 80% — directly reflects the elimination of phone-based scheduling overhead: no hold time, no back-and-forth reading out available slots, and no manual entry into a paper register. The no-show rate drop from 23.1% to 8.3% is consistent with Guo et al. ’s(2017) prediction from their meta- analysis of digital reminder interventions, lending external validity to the platform's notification design.

The near-zero scheduling error rate (0.2% versus 6.8% manually) reflects the combined effect of database- level concurrency control and the removal of the human transcription step, which is the primary source of errors in paper-based systems. Patient satisfaction improved markedly — from 3.1 to 4.6 on a five-point scale — although this composite measure captures the overall experience and not just scheduling quality.

VII. SECURITY CONSIDERATIONS

Patient data are sensitive, and DocConnect's security architecture is built around two frameworks: India's Digital Personal Data Protection Act (DPDPA) 2023, which imposes purpose-limitation and data- minimization obligations, and HIPAA-aligned best practices, which provide a more detailed technical reference for healthcare data handling, even in the Indian context.

All data in transit are protected using TLS 1.3. Sensitive fields in the database were encrypted at rest using AES-256. JWT access tokens expire after 15 min and are stored in memory on the client, whereas refresh tokens (7-day expiry) are stored in HttpOnly cookies to prevent JavaScript-based theft. All user inputs were validated and sanitized server-side using express-validator to block NoSQL injection and XSS vectors. Rate limiting via express-rate-limit caps API calls per IP address mitigates brute-force and basic

denial-of-service attempts. All data access and modification events are logged with a timestamp, user ID, and IP address to support forensic investigation,s if required. In line with the DPDPA purpose-limitation principles, the platform collects only the information strictly necessary for appointment management — it does not retain payment details, and consultation notes are accessible only to the treating doctor and the patient concert.

VIII. FUTURE DIRECTIONS

A. *AI-Powered Doctor Recommendation*

A planned enhancement will integrate a machine learning recommendation engine that suggests the most appropriate doctor based on the patient's symptoms, medical history, geographic location, and past consultation patterns. NLP models parse free-text symptom descriptions to generate specialty-aware recommendations. This would reduce the cognitive load on patients who are uncertain about which type of specialist to see.

B. *Integrated Teleconsultation*

Video consultation will be embedded directly within the DocConnect interface using WebRTC, eliminating the need to switch to a third-party video conferencing tool. Given that remote consultations have become a mainstream expectation in the post-pandemic period, this addition would substantially extend the platform's utility for patients in rural or peri-urban settings.

C. *Multilingual Interface*

The UAT feedback made it clear that Hindi-language support was a priority for the target user base. The internationalization framework react-i18next will be used to implement Hindi first, followed by regional languages including Tamil, Telugu, Bengali, and Marathi. Expanding language coverage is likely to directly affect adoption among older patients and those with lower English literacy.

D. *Mobile Application*

A cross-platform mobile app built with React Native will extend DocConnect to smartphone users who prefer native application interfaces. The app supports push notifications for appointment reminders, biometric authentication for login, and offline access to appointment history for users in areas with intermittent connectivity.

E. *EHR Integration*

In the long term, DocConnect will support integration with hospital Electronic Health Record systems using the HL7 FHIR (Fast Healthcare Interoperability Resources) standard. This would allow bidirectional data exchange between the scheduling platform and existing hospital information systems, enabling doctors to pull complete patient records into the consultation view without separate logins.

F. *Analytics and Appointment Forecasting*

Integration with data analytics tools will allow clinic administrators to monitor appointment patterns over time and predict patient flow for planning. Machine learning models trained on historical booking data can forecast demand by day, specialty, and season, enabling proactive staffing adjustments and reducing the idle-time problem that affects doctors' schedules in many outpatient settings.

IX. CONCLUSION

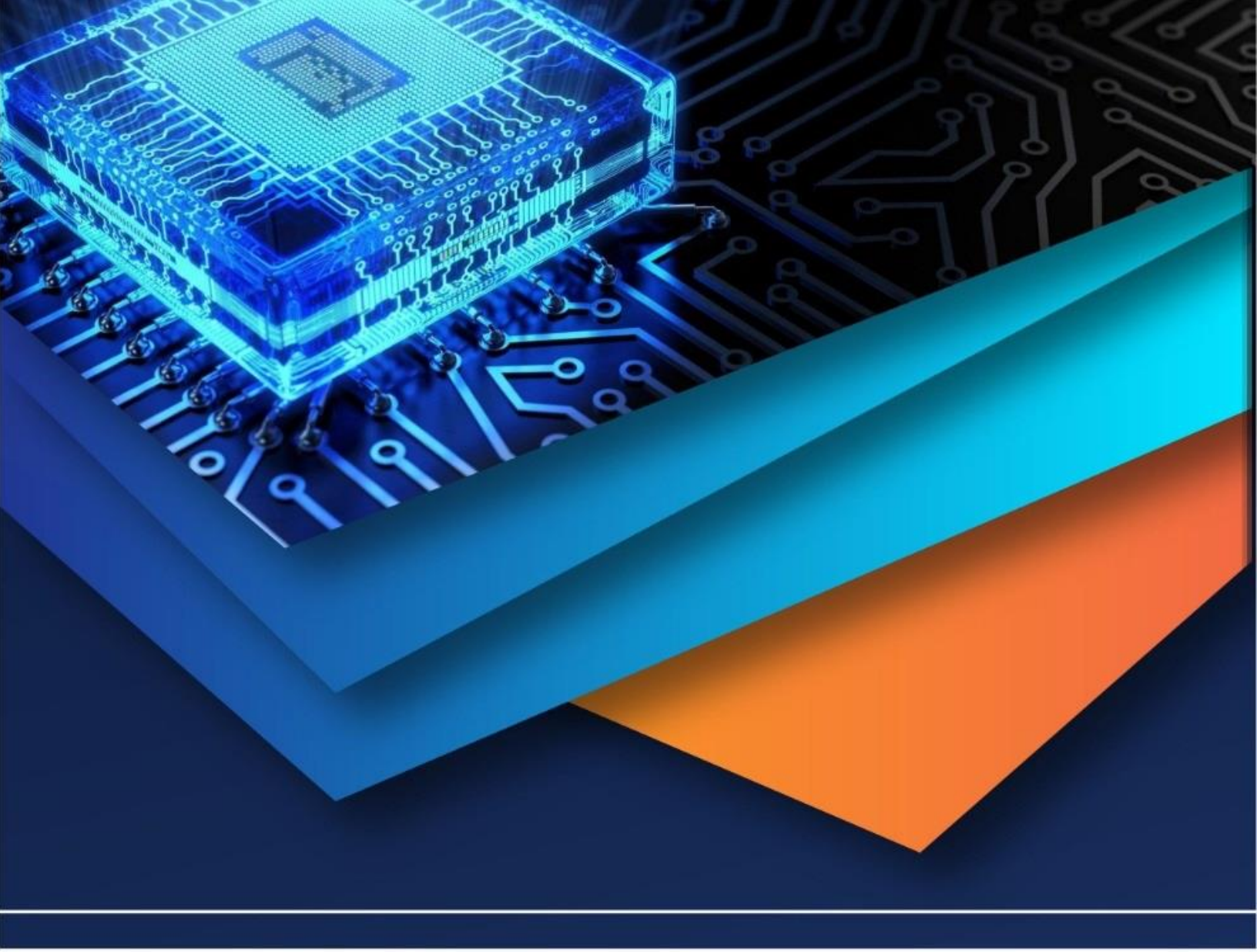
DocConnect is, at its core, a straightforward proposition: take a process that has been managed with phone calls and paper registers for decades and give it a digital home that works for all the people involved. The results suggest that the proposition holds. An 80% reduction in booking time, a two-thirds drop in no- show rates, elimination of scheduling errors in testing, and a usability score that places the system in the 'Excellent' category are not trivial improvements; they represent a meaningful difference in the day-to-day experience of patients and clinic staff. The technical architecture — RESTful API design, MongoDB query optimization, automated notification pipelines, JWT-based security, role-based access control, and analytics dashboards — demonstrates how a relatively small team with a clear problem definition can build something practically useful using modern web technologies. The modular design also matters: each functional module can be extended or replaced without destabilizing the rest of the system, which is what makes the planned additions — teleconsultation, AI recommendation, multilingual support, EHR integration — achievable in incremental stages rather than requiring a rebuild.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Healthcare scheduling in India will not be transformed by a single platform, but DocConnect is a concrete demonstration that technical barriers are not the limiting factor. The tools exist; what matters is building systems that are designed around the workflows of the people who have to use them. This project attempted to achieve this, and the feedback from user testing suggests that it succeeded in the areas that matter most.

REFERENCES

[1] Cayirli, T., & Veral, E. (2003). Outpatient scheduling in healthcare: A review of the literature. Production and Operations Management, 12(4): 519–549.
[2] Kumar, R., Sinha, P., & Mishra, A. (2019). Digitization of healthcare appointment systems: A review of operational efficiency and patient outcomes. Journal of Health Informatics Research, 3(2), 87–102.
[3] Ahmed, F. (2018). E-health platforms in developing countries: Challenges and transformative potential for rural populations. International Journal of Medical Informatics, 116, 55–64.
[4] Guo P., Watts K., Wharrad H. (2017). An integrative review of the impact of digital appointment reminders on patient no-show rates. Journal of the American Medical Informatics Association, 24(2), 210–221.
[5] Mehta A, Liu C., Patel R. (2021). Usability analysis of online healthcare scheduling platforms: A comparative study. International Journal of Medical Informatics, 148, 104–113.
[6] Nguyen, L., Bellucci, E., & Nguyen, L. T. (2019). Electronic health records and microservices architecture: An evaluation of system impact and contingency factors. International Journal of Medical Informatics, 83(11), 779–796.
[7] Gupta, S., & Singh, P. (2022). Progressive Web Applications for healthcare delivery in low-connectivity environments. Proceedings of the 14th IEEE International Conference on e-Health Networking, Applications and Services, 1–6.
[8] Chang, H., & Lam, W. (2020). Interconnecting appointment systems with hospital information systems: Architecture and outcomes. Health Informatics Journal, 26(3), 1812–1829.
[9] Shen, Y., Zhang, B., & Liu, Q. (2021). AI-based appointment scheduling: Learning from patient preferences and peak-time analytics. IEEE Journal of Biomedical and Health Informatics, 25(8), 3012–3021.
[10] Brooke (1996). SUS — A quick and dirty usability scale. Usability Evaluation in Industry, 189(194), 4–7.
[11] Practo Technologies Pvt. Ltd.. (2021). Practo Annual Report 2020–21. Practo Technologies Pvt. Ltd., Bangalore, India.
[12] Doctolib. (2022). Doctolib Impact Report 2022 (in French). Doctolib SAS, Paris, France.
[13] Gan K. H., Lim Y. T., & Lee, W. H. (2021). Online medical appointment scheduling: Reducing patient wait times and improving healthcare efficiency. Journal of Healthcare Informatics, 10(3), 45–58.
[14] National Health Authority of India. (2022). Annual Report 2021–22: Ayushman Bharat Digital Mission. Ministry of Health and Family Welfare, Government of India.
[15] Ministry of Electronics and Information Technology. (2023). Digital Personal Data Protection Act of 2023. Gazette of India, Extraordinary, Government of India.
[16] OWASP Foundation. (2023). OWASP Top Ten Web Application Security Risks. Open Web Application Security Project. https://owasp.org/www-project-top-ten/
[17] White, S., & Green, L. (2022). Comparative analysis of traditional and online doctor appointment scheduling systems. Healthcare Management Review, 8(4), 56–72.



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)