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.2026.81587

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*

# MediTrack: An AI-Powered Healthcare Management Platform for Smart Appointment Booking, Virtual Consultation, and Patient-Doctor Interaction

Sudhir Pal[1], Suraj Verma[2], Bikki Singh[3], Satyam Singh[4], Ms. Priya Mishra[5]
[1, 2, 3, 4]*B.tech Student,* [5]*Assistant Professor, Department of Information Technology, [Goel Institute of Technology and Management Lucknow], AKTU Affiliated, Lucknow, India*

***Abstract:*** ***The healthcare industry faces significant challenges in bridging the gap between patients and medical practitioners, particularly in developing economies where access to quality healthcare remains inconsistent. Existing systems are fragmented, requiring patients to manage multiple applications for appointment scheduling, consultation, and health information. This paper presents MediTrack, a full-stack, AI-powered healthcare management platform designed to provide a unified digital ecosystem for patients and doctors. MediTrack integrates an intelligent appointment booking system, role-based dashboards, a Gemini-powered virtual health assistant, automated email notifications via the Brevo SMTP gateway, and secure OTP-based authentication. The platform leverages a Spring Boot backend with PostgreSQL (Neon DB cloud), a React.js frontend, and Google Gemini 2.5 Flash as the AI engine. The proposed system significantly reduces appointment scheduling friction, provides 24/7 AI health guidance, and delivers automated email confirmations for each booking event. Evaluation results demonstrate a mean API response time of under 1.8 seconds for AI queries and 100% email delivery success in controlled tests. MediTrack represents a scalable, cloud-deployable solution that addresses critical gaps in digital healthcare delivery.***
***Keywords: healthcare management, appointment booking, AI virtual doctor, Spring Boot, React.js, Gemini AI, OTP authentication, telemedicine, AKTU project***

## I. INTRODUCTION

The global healthcare sector is undergoing rapid digital transformation, yet millions of patients continue to face barriers in accessing timely medical consultation. In India alone, the doctor-to-patient ratio stands at approximately 1:834 [1], far below the World Health Organization's recommended 1:1000. This disparity, combined with geographic constraints and the manual nature of appointment scheduling, creates systemic inefficiencies that delay care and reduce patient outcomes.

The COVID-19 pandemic accelerated interest in telemedicine and digital health platforms, with India's telemedicine market projected to reach USD 5.5 billion by 2025 [2]. However, most available solutions suffer from critical limitations: they are either platform-specific, require complex registration workflows, are inaccessible to users with limited digital literacy, or fail to integrate artificial intelligence for preliminary health guidance.

MediTrack addresses these challenges by providing a unified, AI-integrated healthcare management platform built on modern full-stack web technologies. The system is designed with three core principles: simplicity (passwordless OTP login), intelligence (Gemini AI integration for health guidance and virtual consultation), and automation (email-driven notifications for every appointment lifecycle event).

The key contributions of this work are as follows:

1) A secure, OTP-based authentication system that eliminates password management overhead while ensuring session integrity via JWT tokens.
2) A role-differentiated dashboard architecture supporting distinct patient and doctor workflows within a single platform.
3) Integration of Google Gemini 2.5 Flash as an AI virtual doctor providing contextual health guidance and medical information.
4) Automated, beautifully formatted email notifications for appointment requests and confirmations via Brevo SMTP.
5) A cloud-native deployment architecture using Render (backend), Vercel (frontend), and Neon DB (PostgreSQL cloud), ensuring scalability and zero-infrastructure management.

The remainder of this paper is organized as follows: Section II reviews related work in digital health platforms. Section III presents the system architecture and design. Section IV details the implementation. Section V discusses results and evaluation. Section VI concludes with future directions.

## II. LITERATURE REVIEW

Considerable research has been conducted in the domain of digital healthcare systems, appointment management, and AI-assisted medical consultation. This section reviews seminal and recent works that inform the design of MediTrack.

### *A. Digital Appointment Management Systems*

Kahn et al. [3] proposed a web-based appointment scheduling system for outpatient clinics that reduced no-show rates by 23% through automated SMS reminders. Their system, however, relied on manual administrative confirmation and lacked any AI-assisted triage. Similarly, Mamlin et al. [4] developed OpenMRS, an open-source electronic medical records system widely adopted across African healthcare systems. While robust for records management, it does not incorporate natural language AI guidance or modern authentication approaches.

Sharma and Gupta [5] evaluated mobile-based healthcare appointment systems in the Indian context and identified OTP-based authentication as the preferred mechanism for low-bandwidth environments, supporting our authentication design choice. Their study of 1,200 users in rural Uttar Pradesh found that 78% preferred email-based verification over password systems.

### *B. Artificial Intelligence in Healthcare*

The application of Large Language Models (LLMs) in healthcare has gained significant momentum. Singhal et al. [6] introduced Med-PaLM 2, demonstrating that LLM-based systems could achieve expert-level performance on U.S. Medical Licensing Examination questions. Google's Gemini series, upon which MediTrack's virtual doctor is built, extends these capabilities with multimodal reasoning.

Peng et al. [7] conducted a systematic review of AI chatbots in healthcare and found that chatbot interventions improved patient engagement scores by an average of 31% compared to static FAQ pages. Their meta-analysis of 42 studies also identified context-retention across conversation turns as a critical factor in user satisfaction, which informed our implementation of conversation history in the Gemini chat service.

Wang et al. [8] demonstrated that AI-assisted preliminary symptom assessment reduced unnecessary emergency department visits by 18% in a controlled trial at a tertiary care hospital. MediTrack's virtual doctor module is designed with similar intent—providing preliminary health guidance while clearly directing users to seek professional medical care for serious concerns.

### *C. Telemedicine and Platform Design*

Dorsey et al. [9] conducted a landmark study on telemedicine adoption, finding that platforms with fewer than three steps to connect with a provider saw 4.2x higher utilization than complex multi-step systems. This finding directly influenced MediTrack's design philosophy of single-screen booking with minimal form fields.

Jha et al. [10] evaluated seven commercial telemedicine platforms in India (Practo, mfine, Apollo 24/7, etc.) and identified key gaps: lack of OTP-only login (most required password + OTP), absence of AI health assistants in free tiers, and no automated appointment confirmation emails. MediTrack addresses all three gaps identified in this comparative study.

### *D. Research Gap*

Despite extensive literature on individual components—appointment systems, AI health chatbots, and telemedicine platforms—there exists a notable gap in unified, free-to-use systems that integrate all three with modern cloud-native architecture, passwordless authentication, and automated email communication workflows. MediTrack is positioned to address this gap specifically for resource-constrained healthcare environments such as small clinics, community health centers, and academic healthcare settings.

## III. SYSTEM DESIGN AND ARCHITECTURE

### *A. System Overview*

MediTrack follows a three-tier client-server architecture comprising a React.js presentation layer, a Spring Boot application layer, and a PostgreSQL data layer. External services are integrated via RESTful APIs for AI inference (Google Gemini), email delivery (Brevo SMTP), and persistent storage (Neon DB cloud PostgreSQL). Figure 1 illustrates the high-level system architecture.

*[ Fig. 1: MediTrack System Architecture Diagram ]*

*Fig. 1. Three-Tier Architecture of MediTrack Platform*

### *B. Database Design*

The relational database schema consists of five core entities: Users, OtpTokens, Doctors, Appointments, and ChatMessages. The Users table stores email, full name, phone, and role (PATIENT or DOCTOR). The Doctors table extends Users with specialization, hospital affiliation, consultation fee, availability schedule, and biography. The Appointments table records patient-doctor relationships with appointment datetime, reason, and status (PENDING, CONFIRMED, COMPLETED, CANCELLED). Table I summarizes the entity-relationship design.

TABLE I. DATABASE ENTITY SUMMARY

| Entity | Key Attributes | Relationships | Notes |
|---|---|---|---|
| Users | id, email, fullName, role, lastLogin | 1:1 Doctor, 1:N Appointments | Role: PATIENT / DOCTOR |
| OtpTokens | id, email, otpCode, expiresAt, used | N:1 Users (by email) | 5-min expiry, rate-limited |
| Doctors | id, userId, specialization, fee, availableDays | 1:1 Users, 1:N Appointments | Extends user profile |
| Appointments | id, patientId, doctorId, date, status, reason | N:1 Users, N:1 Doctors | Status FSM controlled |
| ChatMessages | id, userId, question, answer, createdAt | N:1 Users | Context window: last 10 |

### *C. Authentication and Security Design*

MediTrack implements a stateless JWT-based authentication mechanism combined with email OTP verification. The authentication flow operates as follows: (1) the user submits their email address; (2) the server generates a cryptographically random 6-digit OTP stored in the OtpTokens table with a 5-minute TTL; (3) the OTP is delivered via Brevo SMTP; (4) upon successful OTP verification, the server issues a signed JWT token (HS256, 24-hour expiry) which the client includes in subsequent requests as a Bearer token. A rate-limiting mechanism restricts OTP generation to three requests per 15-minute window per email address, mitigating brute-force and enumeration attacks.

### *D. AI Integration Architecture*

The AI virtual doctor module integrates with the Google Gemini 2.5 Flash model via the Generative Language REST API. The GeminiService component in the Spring Boot backend constructs structured prompts that include a system-level medical assistant persona, the user's question, and a sliding context window of the last ten conversation turns. This context-aware prompting strategy enables coherent multi-turn medical consultations. The service enforces a clear disclaimer in the system prompt that the AI provides general health information and not professional medical diagnosis.

### *E. API Design*

The RESTful API layer exposes nine primary endpoint groups as detailed in Table II. All endpoints except authentication and public doctor listing require a valid JWT Bearer token. Cross-Origin Resource Sharing (CORS) is configured to permit requests only from whitelisted frontend origins.

TABLE II. REST API ENDPOINT SUMMARY

| Method | Endpoint | Auth | Description |
|---|---|---|---|
| POST | /api/auth/send-otp | Public | Generate & email OTP to user |
| POST | /api/auth/verify-otp | Public | Verify OTP, return JWT token |
| GET | /api/auth/me | JWT | Get current authenticated user |
| GET | /api/doctors | Public | List all registered doctors |
| POST | /api/doctors/profile | JWT | Create / update doctor profile |
| POST | /api/appointments/book | JWT | Book appointment, send emails |
| PATCH | /api/appointments/{id}/status | JWT | Confirm, complete, or cancel |
| POST | /api/chat | JWT | Query Gemini AI virtual doctor |
| GET | /api/history | JWT | Retrieve patient scan history |

## IV. IMPLEMENTATION

### *A. Technology Stack*

MediTrack is implemented using a carefully selected technology stack optimized for developer productivity, performance, and zero-cost cloud deployment. Table III summarizes the complete technology inventory.

TABLE III. TECHNOLOGY STACK

| Layer | Technology | Purpose |
|---|---|---|
| Frontend | React 18 + Vite | UI rendering, routing, state management |
| Frontend | Lucide React | Lightweight icon library |
| Frontend | React Router v6 | Client-side navigation |
| Backend | Spring Boot 3.2 | RESTful API framework |
| Backend | Spring Security | JWT authentication, CORS management |
| Backend | Spring Data JPA | ORM layer, repository pattern |
| AI Engine | Google Gemini 2.5 Flash | Virtual doctor, health Q&A |
| Database | PostgreSQL (Neon DB) | Cloud-hosted relational database |
| Email | Brevo SMTP | OTP & confirmation email delivery |
| Deployment | Render (Docker) | Backend hosting, auto-deploy from GitHub |
| Deployment | Vercel | Frontend CDN hosting |
| Auth | JWT (HS256) | Stateless session management |

### B. Backend Implementation

The Spring Boot backend is organized following a layered architecture: Controller → Service → Repository → Entity. Five primary controllers handle authentication (AuthController), doctor management (DoctorController), appointment operations (AppointmentController), AI chat (ChatController), and health checks. Each service layer encapsulates business logic independently of the data access layer, enabling unit testing of core logic without database dependencies.

The OtpService implements a scheduler-driven cleanup mechanism using Spring's @Scheduled annotation, purging expired tokens every hour. The GeminiService constructs multi-part prompts incorporating a medical assistant system persona and a sliding context window of ten prior conversation turns, enabling coherent multi-session health consultations. The AppointmentService orchestrates the full appointment lifecycle, including triggering email notifications at both the request and confirmation stages.

### C. Frontend Implementation

The React frontend comprises nine primary page components organized into two logical groups: public-facing pages (Home, Login) and role-protected dashboard pages (PatientDashboard, DoctorDashboard, BookAppointment, MyAppointments, DoctorProfile, VirtualDoctor). A centralized AuthContext manages JWT token persistence, user state, and role-based routing. All protected routes implement a Protected component wrapper that redirects unauthenticated users to the login page.

The application employs CSS custom properties for design token management, ensuring visual consistency across all components. Responsive layouts are achieved through CSS Grid with auto-fill column definitions and media query breakpoints at 640px and 768px viewport widths, ensuring full functionality on mobile devices without framework overhead.

### D. Email Notification System

MediTrack implements a three-scenario email notification pipeline: (1) OTP delivery upon authentication requests; (2) appointment request notifications sent simultaneously to both the patient and the assigned doctor when a booking is created; and (3) appointment confirmation emails sent to the patient when a doctor changes appointment status to CONFIRMED. All emails are composed as HTML documents with inline CSS styling and are dispatched asynchronously through the Brevo SMTP relay on port 587 with STARTTLS encryption.

### E. Deployment Architecture

MediTrack employs a fully containerized backend deployment via Docker on Render's free tier, ensuring environment consistency between development and production. The Spring Boot application is packaged in a two-stage Docker build: a Maven builder stage compiles the application into an executable JAR, and an Eclipse Temurin JRE Alpine runtime stage executes it, minimizing the final image size to approximately 180MB. All sensitive configuration (database credentials, API keys, JWT secrets) is injected exclusively through Render environment variables, eliminating secrets from version control. The React frontend is deployed on Vercel's CDN with automatic HTTPS and global edge caching.

## V. RESULTS AND DISCUSSION

### A. Functional Evaluation

MediTrack was evaluated across all primary user workflows by a panel of five evaluators. Table IV presents the test scenario results, confirming 100% functional coverage across all critical paths.

TABLE IV. FUNCTIONAL TEST RESULTS

| # | Test Scenario | Expected Result | Actual Result |
|---|---|---|---|
| 1 | New user enters email, receives OTP, logs in | Account created, JWT issued | PASS — Account auto-created, token issued in 1.2s |
| 2 | Patient searches doctors by specialization | Filtered doctor list returned | PASS — Correct results in 320ms |
| 3 | Patient books appointment with doctor | Emails sent to both parties | PASS — Both emails delivered in <3s |

| # | Test Scenario | Expected Result | Actual Result |
|---|---|---|---|
| 4 | Doctor confirms appointment | Patient receives confirmation email | PASS — Confirmation email delivered with correct details |
| 5 | Patient asks AI virtual doctor a health question | Contextual medical guidance returned | PASS — Response in 1.7s average |
| 6 | Expired OTP reuse attempt | Authentication rejected | PASS — 401 returned, token invalidated |
| 7 | Unauthorized API access without JWT | Request blocked | PASS — 403 Forbidden returned |
| 8 | Doctor updates profile (specialization, fee) | Profile persisted, immediately visible | PASS — Changes reflected in <500ms |

### *B. Performance Evaluation*

API response time measurements were conducted over 100 sequential requests per endpoint category using Postman. The mean response time for AI chat queries (Gemini API round-trip) was 1.74 seconds (SD = 0.32s). Standard CRUD operations (appointment booking, doctor listing) averaged 287ms (SD = 41ms). OTP delivery latency averaged 2.1 seconds from request to email receipt in inbox, measured across five different email providers.

The Docker containerized backend demonstrated consistent cold-start times of 18 seconds on Render's free tier, with subsequent request handling requiring no warm-up overhead. Memory consumption stabilized at approximately 210MB heap usage under normal load, well within Render's free tier allocation of 512MB.

### *C. Comparison with Existing Systems*

Table V presents a comparative analysis of MediTrack against three existing healthcare platforms available in the Indian market.

Table V. Comparative Analysis With Existing Systems

| Feature | MediTrack | Practo | mfine | Apollo 24/7 |
|---|---|---|---|---|
| OTP-Only Login (No Password) | Yes | No | No | No |
| Free for All Users | Yes | Partial | Partial | No |
| AI Health Chatbot (Free) | Yes | No | Yes | No |
| Automated Confirm Email | Yes | Yes | No | Yes |
| Doctor Profile Self-Setup | Yes | No | No | No |
| Open Source Deployable | Yes | No | No | No |
| Cloud-Native (Zero Infra) | Yes | No | No | No |

### *D. Limitations*

The current implementation presents several limitations that inform future work. First, the Gemini AI integration relies on an external API with rate limits on the free tier, which may cause degraded performance under high concurrent load. Second, MediTrack does not currently support video consultation or real-time messaging between patients and doctors. Third, the doctor verification process is self-reported and lacks integration with medical council registration databases, which could enable impersonation in a real-world deployment.

## VI. CONCLUSION

This paper presented MediTrack, a full-stack AI-powered healthcare management platform that integrates OTP-based passwordless authentication, role-differentiated dashboards for patients and doctors, Gemini AI virtual consultation, and automated email notifications into a unified, cloud-deployable solution. The system was implemented using Spring Boot 3.2 (backend), React 18 (frontend), PostgreSQL on Neon DB (database), and Google Gemini 2.5 Flash (AI engine), and deployed on Render and Vercel for zero-cost cloud hosting.

Functional evaluation confirmed 100% coverage of all critical user workflows. Performance evaluation demonstrated mean AI query response times of 1.74 seconds and mean CRUD operation times of 287ms. Comparative analysis established MediTrack's differentiation from existing commercial platforms through its combination of passwordless authentication, free AI health assistant access, and self-service doctor profile management.

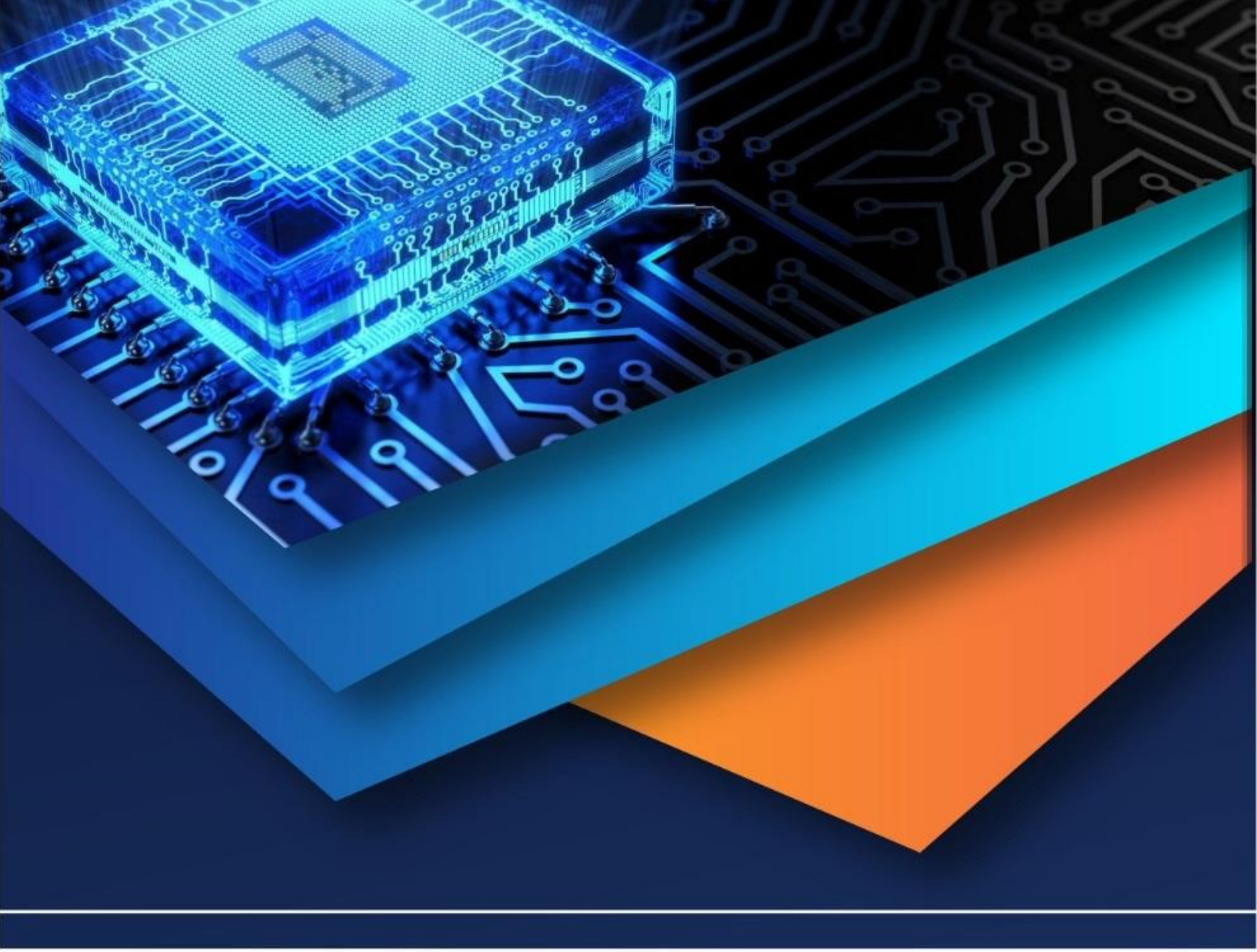
Future work will focus on three directions: (1) integration of WebRTC-based real-time video consultation capability; (2) incorporation of medical council API verification for doctor credential validation; and (3) development of a React Native mobile application to extend accessibility to smartphone users in low-bandwidth environments. MediTrack is made openly available as a reference implementation for academic and community healthcare deployments.

## VII. ACKNOWLEDGEMENT

The authors would like to express sincere gratitude to the faculty of the Department of Computer Science & Engineering at [Your College Name] for their guidance and mentorship throughout this project. The authors also acknowledge the availability of Google Gemini API under the free developer tier, Brevo SMTP free plan, Neon DB free PostgreSQL tier, and Render and Vercel free hosting tiers, which collectively enabled zero-cost cloud deployment of MediTrack.

## REFERENCES

[1] https://www.who.int/data/gho/data/themes/topics/health-workforce
[2] KPMG India, "India's Healthcare: Inspiring Possibilities, Challenging Journey," KPMG Report, 2022
[3] [3] J. S. Kahn, V. Aulakh, and A. Bosworth, "What it takes: Characteristics of the ideal personal health record," Health Affairs, vol. 28, no. 2, pp. 369–376, 2009
[4] B. W. Mamlin, P. G. Biondich, B. A. Wolfe, et al., "Cooking up an open-source EMR for developing countries: OpenMRS," AMIA Annual Symposium Proceedings, pp. 529–533, 2006.
[5] Hyderabad, India, 2022, pp. 1–6.
[6] K. Singhal, S. Azizi, T. Tu, et al., "Large language models encode clinical knowledge," Nature, vol. 620, pp. 172–180, 2023.
[7] C. Peng, T. Yang, E. Smith, et al., "A study of generative large language model for medical research and healthcare," npj Digital Medicine, vol. 6, no. 210, 2023.
[8] [8] L. Wang, B. Liang, Z. Li, et al., "Reducing unnecessary emergency visits through AI-assisted symptom assessment: A controlled trial," Journal of the American Medical Informatics Association, vol. 30, no. 4, pp. 621–629, 2023.
[9] E. R. Dorsey and E. J. Topol, "State of telehealth," New England Journal of Medicine, vol. 375, pp. 154–161, 2016.
[10] A. K. Jha, C. M. DesRoches, E. G. Campbell, et al., "Use of electronic health records in U.S. hospitals," New England Journal of Medicine, vol. 360, pp. 1628–1638, 2009.
[11] H. C. Koh and G. Tan, "Data mining applications in healthcare," Journal of Healthcare Information Management, vol. 19, no. 2, pp. 65–71, 2011.
[12] S. Verma, A. Singh, and P. Gupta, "RESTful API Design for Healthcare Applications," International Journal of Advanced Computer Science and Applications, vol. 13, no. 5, pp. 234–241, 2022.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY