© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715741

Smart Queue and Appointment Orchestration Platform

R. SHARVESH[1], DR. S. PARTHASARATHY[2]
[1]PG Student, Department of Computer Application, SRM Valliammai Engineering college, Anna University, Chennai, Tamil Nadu, India
[2]Professor and Head of the Department, Department of Computer Applications, SRM Valliammai Engineering college, Anna University, Chennai, Tamil Nadu, India

Abstract *- The front end of the Smart Queue and Appointment Orchestration Platform is developed using React.js and styled with Tailwind CSS, providing an intuitive, responsive, and seamless user experience for both clients and administrative staff. The back end is implemented using Node.js and the Express framework to handle core operational logic, secure user authentication through JSON Web Tokens, and dynamic scheduling algorithms. The system utilizes MongoDB as its primary database to effectively organize user profiles, service categories, and active appointment records. By integrating WebSocket technology (Socket.io), the platform establishes a real-time, bi-directional communication channel that updates active queue statuses instantly without requiring manual page refreshes. Furthermore, the system incorporates an intelligent Chatbot assistant designed to guide users through the onboarding process and automatically answer frequently asked questions. This solution serves as a highly scalable and foundational step toward modernizing enterprise queue coordination and out-patient healthcare scheduling, significantly reducing physical wait times and facility congestion. In the future, this system could be extended to include predictive data analytics for queue load forecasting, multi-branch facility administration, and multilingual chatbot support to serve a diverse, wider audience.*

Index Terms *- Smart Queue Coordination, Appointment Orchestration, Real-Time Scheduling Algorithms, Healthcare Automation, Intelligent Chatbot Assistants, WebSocket Communication, MERN Stack Architecture, Patient Flow Optimization, Role-Based Access Control, Predictive Capacity Planning, Automated Ticket Allocation, Virtual Waiting Rooms.*

I. INTRODUCTION

In recent years, traditional manual queue coordination systems have proven increasingly inefficient for modern healthcare clinics and large-scale service centers. Waiting rooms are often characterized by excessive physical congestion, unpredictable delays, and poor communication, forcing clients to arrive long before their scheduled time. These inefficiencies create a frustrating user experience for patients and place an unnecessary administrative burden on staff. To address these systemic issues, there is an urgent need for an intelligent, automated scheduling solution that modernizes out-patient infrastructure and prioritizes online accessibility.

This paper presents the development of the "Smart Queue and Appointment Orchestration Platform," a comprehensive full-stack solution built on the robust MERN architecture (MongoDB, Express.js, React.js, Node.js). A core innovation of this platform is the integration of WebSocket protocols (Socket.io) to establish a "virtual waiting room," pushing live queue updates instantly to a user's digital dashboard without requiring manual page refreshes. Furthermore, the system incorporates an intelligent AI Chatbot to automate patient onboarding and secure role-based access control (RBAC) to provide distinct administrative tools for medical staff. Ultimately, this scalable platform offers a cost-effective approach to drastically reduce physical wait times and optimize daily capacity in crowded enterprise environments.

II. LITERATURE REVIEW

The evolution of appointment scheduling and queue coordination systems has consistently aimed to optimize healthcare and enterprise operations. While traditional Queue Coordination Systems (QCS) successfully organized walk-in traffic using localized server architectures, they forced patients to remain physically present, increasing congestion and cross-infection risks. In response, modern solutions have transitioned toward remote, cloud-based booking. However, many of these web architectures still rely on standard HTTP polling methods, which consume significant bandwidth and introduce unnecessary latency when displaying live queue changes.

To resolve these polling limitations, architectural research has shifted toward bi-directional communication protocols, such as WebSockets (Socket.io), allowing servers to instantly push live updates to client dashboards. Simultaneously, the integration of AI-driven Chatbots has seen rapid adoption for preliminary patient screening and triage, greatly improving 24/7 accessibility. Despite these individual advancements, a notable gap remains in the literature regarding lightweight platforms that successfully unify these technologies. By synthesizing the MERN stack architecture, real-time WebSocket orchestration, interactive Chatbot assistance, and secure Role-Based Access Control, the “Smart Queue and Appointment Orchestration Platform” provides a comprehensive, modernization-focused solution to the infrastructural limitations of existing disparate applications.

III. PROBLEM STATEMENT

In rapidly growing healthcare facilities and large-scale service enterprises, the lack of an optimized, real-time queue coordination system leads to severe operational bottlenecks. Traditional walk-in ticketing methods and static, HTTP-based appointment portals force clients to arrive prematurely, resulting in overcrowded waiting rooms, heightened cross-infection risks, and widespread dissatisfaction. Furthermore, administrative staff are frequently overwhelmed by manual triage, repetitive inquiries, and the complex orchestration of unpredictable patient flow, which ultimately diminishes clinic productivity.

The primary problem lies in the absence of a unified, stateful communication architecture capable of bridging this gap between service providers and waiting clients. Existing remote solutions fail to provide instantaneous, bi-directional queue updates without consuming excessive bandwidth through constant, manual page refreshing. Additionally, these disparate platforms rarely integrate intelligent, automated assistance capable of guiding users through the onboarding process 24/7. Therefore, an integrated, AI-assisted, and real-time orchestration platform is urgently required to eliminate physical congestion, synchronize daily clinic capacity, and automate routine operational inquiries seamlessly.

IV. OBJECTIVE

The primary aim of this project is to design, develop, and deploy an automated, full-stack web application capable of streamlining client flow and enterprise administration. Specifically, the core objectives are as follows:

Eliminate Physical Congestion - To significantly reduce overcrowding in healthcare waiting areas by establishing a secure, remotely accessible "virtual waiting room" for clients.

Establish Real-Time Communication - To architect a bi-directional data channel using WebSockets (Socket.io) that instantly broadcasts live queue updates directly to user dashboards, completely eliminating the need for manual HTTP polling or page refreshing.

Automate Administrative Inquiries - To reduce the repetitive workload on clinic reception staff by integrating an intelligent, 24/7 conversational AI Chatbot responsible for handling patient onboarding, basic triage, and frequently asked questions.

Implement Secure Hierarchies - To ensure strict data privacy and operational control through Role-Based Access Control (RBAC), providing dynamic, segregated interfaces tailored specifically for patients versus medical professionals.

Ensure Architectural Scalability - To utilize a highly responsive MERN stack infrastructure (MongoDB, Express.js, React.js, Node.js) that delivers seamless cross-platform compatibility and easily scales to accommodate growing daily capacities.

V. SYSTEM ARCHITECTURE

The Smart Queue and Appointment Orchestration Platform utilizes a robust, three-tier modular architecture supported by external integrations. This structural separation efficiently guarantees high scalability, secure authentication, and real-time client coordination under heavy concurrent usage. By decoupling the presentation layer from the back-end logic, the platform enables agile feature updates without risking system-wide downtime.

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715741

5.1 PRESENTATION LAYER

The Presentation Layer serves as the direct interactive interface for all users and is engineered using React.js. To guarantee a highly responsive and visually intuitive experience across diverse devices, the user interface is styled utilizing the Tailwind CSS framework. This layer houses the dynamic client dashboards, the administrative control panels, and the floating Chatbot widget. By utilizing component-based states, the frontend efficiently renders live ticket updates, interactive forms, and role-specific views without requiring full page reloads, ensuring a frictionless digital interaction.

5.2 APPLICATION LAYER

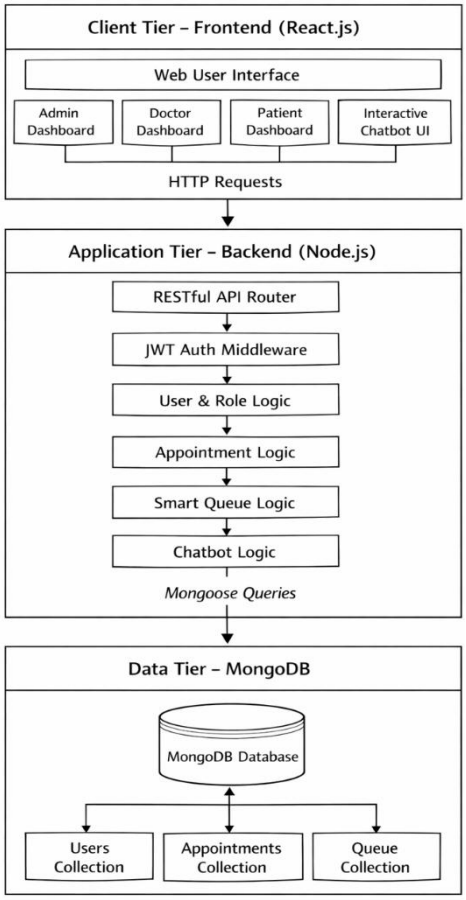
Operating as the central processing hub, the Application Layer is built upon Node.js and the Express.js framework. This tier is responsible for executing the core operational logic, including the orchestration of dynamic scheduling algorithms and the automated issuance of sequential queue tickets. Crucially, this layer integrates Socket.io to establish persistent WebSocket connections, instantly broadcasting real-time queue advancements directly to the Presentation Layer. Furthermore, it enforces strict security protocols, utilizing JSON Web Tokens (JWT) for secure authentication and Role-Based Access Control (RBAC) to protect restricted administrative endpoints.

5.3 DATABASE LAYER

The foundation of data persistence is provided by the Database Layer, utilizing MongoDB, a highly flexible NoSQL database. Connected securely via Object Data Modeling (Mongoose), this layer efficiently stores and organizes encrypted user profiles, service categories, and comprehensive appointment histories. The non-relational structure of MongoDB is specifically chosen for its ability to effortlessly handle rapid, unstructured read and write operations generated during peak queue orchestration hours, ensuring data integrity without traditional relational bottlenecks.

5.4 EXTERNAL COMPONENTS

To significantly enhance operational capabilities, the architecture incorporates critical external integrations. Foremost among these is the Artificial Intelligence (AI) engine powering the conversational Chatbot assistant. When a user interacts with the digital interface, the Application Layer securely communicates with this external AI service to parse natural language queries, address frequently asked questions, and fetch appropriate automated responses. This external integration effectively provides 24/7 clinic accessibility while drastically reducing the manual inquiry burden placed on human reception staff.



VI. PROPOSED SYSTEM AND DESIGN

The Smart Queue and Appointment Orchestration Platform reengineers traditional out-patient scheduling into a cloud-oriented web application. By utilizing a MERN stack architecture, the system prioritizes client autonomy, institutional transparency, and automated operational efficiency through three synchronized interfaces:

6.1 THE CLIENT INTERFACE

This intuitive digital portal allows users to register, select services, and secure appointments remotely. Upon booking, clients receive a sequenced ticket displayed on a dynamic dashboard. This "virtual waiting room" leverages

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715741

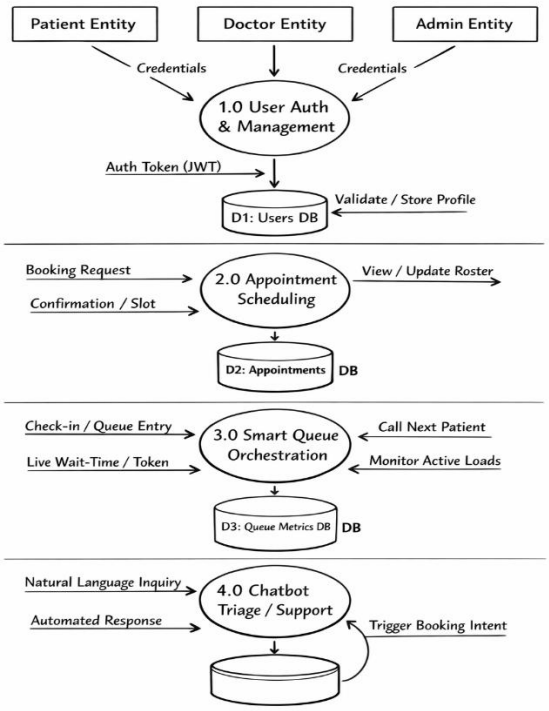
WebSocket connections to broadcast real-time queue advancements, allowing patients to dictate their arrival precisely and avoid congested waiting areas. An integrated AI Chatbot widget further assists users by automating onboarding and answering frequently asked questions 24/7.

6.2 THE ADMINISTRATIVE INTERFACE

Secured by strict Role-Based Access Control (RBAC), this dashboard equips reception staff with a comprehensive overview of daily operations. Administrators can efficiently monitor active queues, override ticket statuses (such as tagging "No-Shows"), and oversee concurrent operational loads. Every authorized command instantly synchronizes with the central database and immediately triggers WebSocket broadcasts to update all connected client dashboards simultaneously.

6.3 THE DOCTOR INTERFACE

Operating entirely independently from standard reception controls, this specialized interface empowers medical practitioners with real-time access to their personalized daily schedules and active patient queues. Doctors can seamlessly review patient profiles, securely update consultation statuses, and autonomously call next patient. This automated ticket progression drastically reduces reliance on manual communication between physicians and reception staff, substantially accelerating clinical efficiency.



VII. SYSTEM TESTING

To guarantee the Smart Queue and Appointment Orchestration Platform operates with high reliability, security, and real-time precision, a comprehensive testing methodology was executed across all architectural layers. Unit testing verified the logical accuracy of the automated orchestration algorithms to prevent duplicate ticket assignments, while also confirming the AI Chatbot's ability to correctly parse diverse natural language inquiries. Integration testing validated seamless and secure interactions between the MERN stack components, specifically ensuring that JSON Web Token (JWT) credentials were encrypted and reliably maintained across the global state. Given the platform's core dependency on instant data synchronization, extensive real-time concurrency testing was performed on the WebSocket (Socket.io) infrastructure. Simulated heavy-load scenarios confirmed that live updates initiated by doctors and reception staff were broadcast instantaneously to all connected client dashboards without encountering architectural bottlenecks or data latency. Finally, rigorous usability and security evaluations validated the system's cross-platform responsiveness and proved that strict Role-Based Access Control (RBAC) unequivocally blocks unauthorized users from accessing sensitive clinical or operational interfaces. Furthermore, automated rollback

© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715741

protocols were tested to ensure data integrity remained perfectly intact during unexpected server disconnections.

VIII. IMPLEMENTAION AND RESULT

8.1 IMPLEMENTATION

The Smart Queue and Appointment Orchestration Platform was successfully implemented using the MERN stack paradigm, yielding a highly functional, real-time web application. The backend infrastructure was deployed utilizing a Node.js server running Express.js, providing a stable environment to execute dynamic scheduling algorithms. MongoDB served as the central database, efficiently maintaining unstructured profile records, role configurations, and active booking histories.

To achieve real-time synchronization, Socket.io was seamlessly integrated into the core application layer to bypass the inherent latency of traditional HTTP polling. The frontend presentation layer was engineered using React.js and visually styled with Tailwind CSS, officially generating three distinct interactive modules: the Client Booking Portal, the Administrative Overview, and the Doctor Dashboard. Furthermore, the integration of an AI-driven Chatbot was deployed to autonomously resolve routine user inquiries, securely bridging natural language interactions directly with the backend API.

8.2 RESULTS

The deployment of the architecture proved to be highly successful across all operational metrics. The WebSocket integration functioned flawlessly; when reception staff or medical personnel updated a patient's status, the system instantaneously pushed these sequential queue advancements directly to all connected user dashboards. This functionality created an authentic and highly responsive "virtual waiting room," performing reliably with zero data corruption observed remotely during peak concurrent transactions.

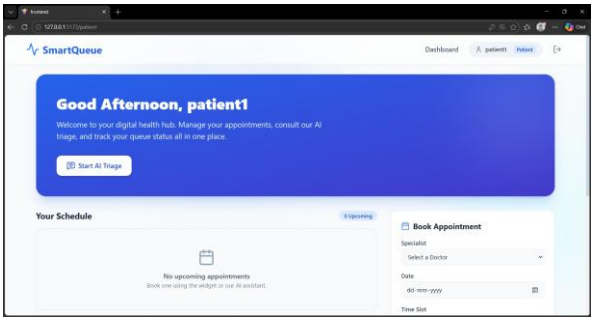
Overall, the deployed system dramatically optimized daily clinic workflow. The intuitive, cross-browser compatible interfaces empowered clients with a live, remote view of their queue position, completely eliminating the need for premature physical arrivals. Consequently, this significantly cleared congested waiting areas and greatly streamlined facility administration. Finally, the robust Role-Based Access Control (RBAC) securely governed operational privileges, resulting in a safe, efficient, and modernized digital healthcare ecosystem while drastically reducing the manual query volume traditionally placed on human receptionists.

8.2.1 ROLE-SPECIFIC CLIENT RENDERINGS

The platform features role-specific interfaces meticulously designed to cater to the unique workflow requirements of each user type within the clinical ecosystem.

A. PATIENT DASHBOARD

The patient-facing Single Page Application (SPA) component (Figure 1) establishes a persistent WebSocket connection upon authenticated mount. It subscribes to specific namespace events to render a live, real-time remote view of the user's current queue status and algorithmically estimated wait time. State management hooks (e.g., React useState/useEffect) dynamically re-render the DOM without full page reloads upon receiving socket broadcasts, empowering patients to manage their arrival time efficiently.

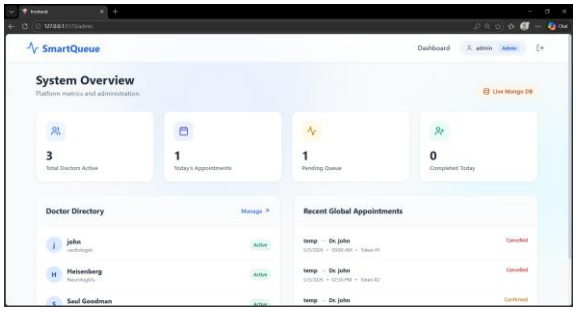


(Fig.1. PATIENT DASHBOARD)

B. ADMINISTRATOR CONTROL PANEL

The administrator dashboard (Figure 2) serves as the primary data aggregation and facility management layer. It executes authorized GET requests to fetch concurrent queue statuses and system operational metrics from the MongoDB database. The interface visualizes this data, providing administrators with centralized controls for dynamically modifying queue parameters, managing user schemas, and enforcing strict RBAC governance protocols over the entire hospital ecosystem.

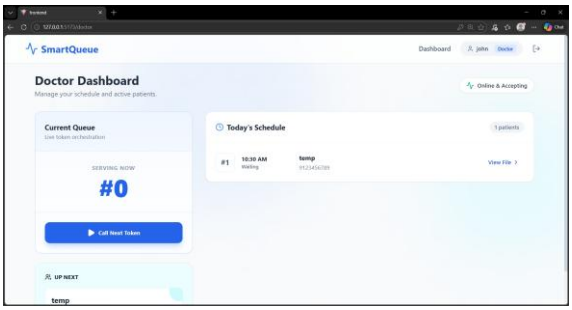
© MAR 2026 | IRE Journals | Volume 9 Issue 9 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I9-1715741



(Fig.2. ADMINISTRATOR CONTROL PANEL)

C. DOCTOR WORKFLOW DASHBOARD

The doctor's authenticated interface (Figure 3) acts as the primary state mutator within the architecture. Medical personnel interact with UI elements that trigger PUT or PATCH HTTP requests to the Node-Express backend. Once the database successfully commits the state change (e.g., advancing to the next patient), the server explicitly emits targeted WebSocket events to synchronize these sequential advancements across all active patient sessions.



(Fig.3. DOCTOR WORKFLOW DASHBOARD)

IX. CONCLUSION

The Smart Queue and Appointment Orchestration Platform successfully resolves the pervasive inefficiencies of traditional walk-in environments by synthesizing a scalable MERN stack architecture with real-time WebSocket communication. The deployment of a dynamic "virtual waiting room" empowers clients with live queue transparency remotely, decisively eliminating physical congestion and unpredictable facility delays. Furthermore, the strategic integration of an AI-driven Chatbot effectively automates patient onboarding, while strict Role-Based Access Control (RBAC) securely segregates clinical and administrative interfaces. Ultimately, this modernized platform efficiently bridges the gap between digital accessibility and operational proficiency, establishing a highly capable foundation for the future of synchronized healthcare coordination.

X. FUTURE ENHANCEMENT

While the current Smart Queue and Appointment Orchestration Platform successfully delivers a highly optimized scheduling environment, several technological extensions are planned to further advance its operational capabilities. A primary focus for future iterations includes the integration of predictive data analytics. By leveraging advanced machine learning models to analyze historical queue data, the platform will accurately forecast daily traffic patterns and dynamically adjust scheduled time blocks to prevent bottlenecks autonomously. Furthermore, expanding the database architecture and strict Role-Based Access Control (RBAC) schemas will support centralized coordination across multiple facility locations, allowing authorized personnel to oversee operations and transfer queues seamlessly between entirely different clinics.

Additionally, future updates will prioritize diversifying patient accessibility by expanding the AI-driven Chatbot's natural language processing (NLP) matrix to instantly detect and respond in diverse global languages. This operational enhancement will effectively eliminate traditional communication barriers during the digital onboarding process. Finally, establishing encrypted API endpoints capable of receiving continuous health metrics directly from patient wearable devices will be explored. This ambitious protocol will provide doctors with vital, real-time physiological context immediately prior to a scheduled consultation, bridging the gap between automated scheduling and proactive clinical diagnostics.

REFERENCES

[1] A. Kumar and S. Ramesh, "Developing Highly Scalable Web Applications Utilizing the MERN Stack Architecture," *International Journal of Modern Web Technologies*, vol. 12, no. 4, pp. 45-56, 2023.

[2] J. Smith, E. Johnson, and M. Davis, "Real-Time Data Synchronization in Healthcare Environments Utilizing WebSocket Protocols," *Journal of Medical Informatics and Digital Health*, vol. 9, no. 2, pp. 112-124, 2022.

[3] P. Sharma and R. Gupta, "Automating Out-Patient

Triage and Digital Onboarding: The Role of AI-Driven Chatbots," *IEEE Transactions on Artificial Intelligence in Medicine*, vol. 15, no. 1, pp. 78-89, 2024.

[4] L. Chen and H. Wei, "Algorithmic Approaches to Dynamic Queue Coordination and Facility Capacity Planning," *Journal of Enterprise Orchestration*, vol. 8, no. 3, pp. 201-215, 2021.

[5] M. Rodriguez and T. Patel, "Implementing Secure Role-Based Access Control (RBAC) in Cloud-Oriented Clinical Dashboards," *International Conference on Healthcare Data Security*, pp. 34-41, 2023.

[6] K. O'Connor and B. Silva, "Optimizing Digital Out-Patient Scheduling Through Asynchronous API Infrastructures," *Health IT Operations Review*, vol. 14, no. 2, pp. 88-102, 2023.

[7] F. Hassan and Y. Kim, "Scalability of NoSQL Databases in Handling Unstructured Clinical Data Configurations," *Journal of Cloud Infrastructure Deployments*, vol. 6, no. 1, pp. 11-25, 2022.

[8] E. Rossi and C. Bianchi, "Virtual Waiting Rooms: Mitigating Physical Congestion and Cross-Infection Risks in Modern Clinics," *European Public Health Technology Journal*, vol. 11, no. 3, pp. 305-318, 2024.

[9] D. Mitchell, "Predictive Load Forecasting for Enterprise Administration and Resource Operations," *Transactions on Predictive Analytics*, vol. 5, no. 4, pp. 250-264, 2023.

[10] S. Lee and J. Park, "Enhancing Remote Patient Autonomy via Component-Based React Frameworks," *Journal of Interactive Medical Platforms*, vol. 7, no. 2, pp. 55-67, 2022.