MediCare Clinic: A Web-Based Role-Based Clinic Appointment Booking System Using Single-Page Application Architecture

[1]Bhaskar, [2]Ravi Kumar Chaudhary, [3]Rajendra Singh

[1] Department of Computer Science and Engineering, Raffles University, Neemrana, Rajasthan, India

[2] Assistant Professor, Department of Computer Science and Engineering, Raffles University, Neemrana, Rajasthan, India

[3] Dean, Department of Computer Science and Engineering, Raffles University, Neemrana, Rajasthan, India

***Abstract*—The digitization of administrative workflows in small healthcare facilities presents a distinct set of software engineering challenges that extend beyond database design and basic web development. This paper presents the design, implementation, and evaluation of MediCare Clinic, a complete web-based clinic appointment booking system for small independent clinics, with specific emphasis on role-based access control architecture, single-page application design, appointment conflict prevention, and responsive user interface engineering. The proposed system provides a professional browser-based interface through which three distinct user roles — Admin, Doctor, and Patient — each access a dedicated dashboard with role-appropriate features and permissions. Patients can self-register, browse available doctors, select time slots, and book appointments at any time without telephone contact. Doctors receive real-time visibility into their appointment schedule with status management controls. The Admin maintains complete oversight of all doctors, patients, and appointments through a unified dashboard with live statistics. The system is implemented as a self-contained single HTML file using HTML5, CSS3, and vanilla JavaScript, requiring no backend server, no database installation, and no software dependencies. A dynamic time slot blocking mechanism prevents double-booking by querying the in-memory data store before rendering available slots. The appointment lifecycle is managed through four status states — Pending, Confirmed, Completed, and Cancelled — with role-controlled state transitions enforced at the application logic layer. The system was evaluated through ten functional test cases covering all major user flows, achieving 100% pass rate. Performance measurements show role navigation switching under 50 milliseconds and appointment booking completion**

under 100 milliseconds, representing an improvement factor exceeding 3,000 times over the traditional manual phone-and-register booking process.

***Index Terms*—Clinic Management System, Web Application, Role-Based Access Control, Single-Page Application, Appointment Booking, Healthcare IT, HTML5, CSS3, JavaScript, Responsive Design**

I. INTRODUCTION

India's healthcare delivery infrastructure is characterised by a large number of small, independently operated clinics that collectively serve the majority of the urban and semi-urban population. The National Health Profile published by the Central Bureau of Health Intelligence documents over seven lakh registered clinical establishments in India, the vast majority of which are small single-doctor or multi-doctor clinics operating with limited administrative staff and no dedicated IT infrastructure. These clinics represent a segment that has been largely bypassed by the commercial clinic management platforms designed for larger hospital chains, creating a persistent gap between available technology and actual adoption at the grassroots level of healthcare delivery.

The dominant appointment management method in these facilities remains the paper register. A receptionist manually records appointment entries when patients call during business hours. Cancellations are crossed out. Doctors consult the physical register to review their schedule. This approach introduces scheduling conflicts from simultaneous bookings, prevents patients from booking outside business hours, offers no remote schedule visibility for doctors, and provides no aggregated statistical view for administrators. Paper records are fragile, unsearchable, and provide no automated reminders, contributing to high patient no-show rates.

The widespread adoption of mobile internet access and modern web browsers across India, including in smaller cities and towns, has created a practical opportunity for lightweight browser-based solutions that require no software installation and no dedicated server hardware. A single-page web application that runs entirely in the browser, can be hosted on any static file server, and is accessible from any smartphone represents an ideal fit for this deployment context.

This paper presents MediCare Clinic, a web-based clinic appointment booking system that digitizes the complete appointment lifecycle for small clinics. The system's primary contributions are distinct from prior work in the field, which has focused predominantly on enterprise-scale hospital information systems, telemedicine platforms, and AI-assisted diagnosis tools. The specific contribution of this work is a documented approach to building a production-quality, role-based, conflict-preventing appointment management application using only standard browser technologies without any framework or server dependency.

Specifically, this paper contributes: first, a documented three-role access control system implemented within a single HTML file using session-based JavaScript state management; second, a dynamic time slot rendering architecture that prevents double-booking through in-

memory query and slot disabling; third, an appointment lifecycle management system with four status states and role-controlled transition enforcement; fourth, a responsive CSS design system using custom properties and CSS Grid appropriate for healthcare application aesthetics; and fifth, functional evaluation through ten test cases and performance benchmarking against traditional manual appointment processes.

II. RELATED WORK

A. Traditional Clinic Management Systems

Clinic and hospital management systems have evolved significantly over the past three decades. Chaudhry et al. conducted a comprehensive review of health information technology covering over 257 studies, establishing that computerized appointment systems reduced scheduling errors, decreased patient waiting times, and improved staff productivity compared to paper-based alternatives [1]. Their findings remain the foundational evidence base for the value of digitizing healthcare administrative processes and directly motivate the development of accessible digital solutions for small clinic contexts.

In the Indian context, the adoption of digital clinic management has been slower than in Western countries, largely due to cost barriers and the predominance of small independent clinics. Studies from the Public Health Foundation of India have documented the use of basic digital tools, including spreadsheets and messaging platforms, as informal appointment solutions in urban clinics, demonstrating demand for more structured systems [2]. Commercial platforms such as Practo and Lybrate have demonstrated the market viability of digital booking in India, but are designed primarily for larger facilities and require ongoing subscriptions, leaving a significant gap for lightweight self-contained solutions.

B. Web-Based Healthcare Applications

Sittig and Singh identified security, usability, and interoperability as the three primary challenges in web-based health application development, and their framework has influenced subsequent design guidelines emphasizing role-based access control as the primary mechanism for ensuring that different user types access only information relevant to their role [3]. The development of responsive web design, formalized by Marcotte, transformed healthcare web application development by making it practical to build a single application that works correctly across screen sizes [4]. This is particularly significant in the Indian context where patients predominantly access the internet via smartphone.

Modern healthcare web applications commonly follow the Single-Page Application architecture, where the entire application is loaded once and subsequent interactions update page content dynamically. This approach, associated with frameworks including Angular, React, and Vue.js, provides a smoother user experience that reduces perceived latency and more closely resembles a native application.

C. Role-Based Access Control in Web Systems

Role-Based Access Control was formally defined by Ferraiolo and Kuhn and has since become the dominant access control model in enterprise software [5]. In clinic management systems, RBAC is essential for ensuring that patients access only their own records, doctors manage only their own appointments, and administrative functions are restricted to authorized users. Sandhu et al. extended the original model to define RBAC96, providing the theoretical foundation for modern web application authorization systems [6]. In practice, RBAC in web applications is implemented through session-based or token-based authentication where the user role is stored and checked before sensitive operations.

D. Single-Page Application Architecture

A Single-Page Application loads a single HTML page and dynamically updates content as the user interacts, rather than loading entirely new pages from the server. This architecture was popularised by Gmail and has become the standard for complex web applications. The key advantage for a clinic management system is the smooth, app-like experience where transitions between the dashboard, booking screen, and profile are instantaneous. Mikowski and Powell documented the technical foundations of SPA architecture using the Fetch API and DOM manipulation [7], providing the basis for the vanilla JavaScript implementation used in this work.

E. UI/UX in Healthcare Applications

Nielsen's ten usability heuristics remain the most widely cited framework for evaluating user interface quality [8]. Key heuristics for healthcare applications include visibility of system status, match between system and the real world, and error prevention. Caglar and Mealha found that green color schemes in healthcare interfaces were associated with positive perceptions of cleanliness, trust, and professionalism by both patients and medical staff [9], informing the deep forest green primary color selection in the MediCare Clinic design system.

III. SYSTEM DESIGN AND ARCHITECTURE

A. Overall Application Architecture

The MediCare Clinic system follows a three-tier presentation architecture implemented entirely on the client side. Unlike a traditional three-tier web application where the tiers correspond to browser, web server, and database server, in this single-page application all three tiers execute within the browser. The Presentation Layer consists of UI components including the login screen, navigation tabs, statistics cards, data tables, booking forms, and modal dialogs, all rendered dynamically by JavaScript functions that construct HTML strings and inject them into the DOM. The Application Logic Layer contains authentication, role determination, navigation routing, appointment booking validation, status management, and data transformation logic implemented as organized JavaScript functions. The Data Layer is a JavaScript object that holds all system data — users, appointments, and auto-increment counters — simulating a relational database for

demonstration purposes.

The design decision to implement the entire system in a single self-contained HTML file was motivated by deployment practicality. The resulting application requires no web server, no database server, no build tools, and no internet connection for operation. It can be opened directly in any modern browser by double-clicking the file, hosted on any static file service, or shared via USB drive, making it genuinely deployable in clinics with minimal technical infrastructure.

B.Role-Based Access Control Architecture

The system implements three distinct user roles — Admin, Doctor, and Patient — each with a separate set of navigation tabs, page modules, and permitted operations. Role determination occurs immediately after login by reading the role field from the authenticated user record. A central role router function reads this field and builds the appropriate navigation configuration and default landing page for that role.

The Admin role provides access to three modules: a Dashboard showing system-wide statistics and all appointments with confirmation and cancellation controls, a Doctors module for adding and removing doctor records, and a Patients module providing a read-only view of registered patients. The Doctor role provides access to two modules: an Appointments module showing the doctor's own schedule with status management controls, and a Profile module showing personal and professional information. The Patient role provides access to three modules: a My Appointments module showing the patient's own booking history with cancellation controls, a Book Appointment module showing the doctor grid for new booking, and a Profile module.

Table I presents the complete permissions matrix across all three roles.

TABLE I: USER ROLES AND PERMISSIONS MATRIX

Feature	Admin	Doctor	Patient
View all appointments	Yes	No	No
View own appointments	Yes	Yes	Yes
Confirm appointments	Yes	Yes	No
Cancel appointments	Yes	Yes	Own only
Mark as completed	No	Yes	No
Book new appointment	No	No	Yes
Add/remove doctors	Yes	No	No
View patient list	Yes	No	No
View statistics dashboard	Yes	Yes	Yes
Register new account	N/A	N/A	Yes

C. Appointment Booking Flow

The appointment booking process is the core workflow of the system and is designed to be intuitive for patients while preventing scheduling conflicts. The process follows six steps. In Step 1, the patient views a grid of doctor cards on the Book Appointment page, each displaying the doctor's name, speciality, years of experience, and consultation fee. In Step 2, clicking the Book Appointment button on a doctor card opens a modal dialog pre-populated with the selected doctor's name and speciality. In Step 3, when the patient selects a date, the time slot grid is dynamically refreshed by querying the data store for all non-cancelled appointments for that doctor on that date. In Step 4, already-booked slots are rendered with a disabled visual style and are unclickable, while available slots can be selected. In Step 5, the system performs three validations on submission: date selected, time slot selected, and reason for visit not empty. In Step 6, if validation passes, a new appointment record is written to the data store with status Pending and the patient is navigated to their appointments dashboard.

The system uses eight fixed daily time slots: 09:00 AM, 10:00 AM, 11:00 AM, 12:00 PM, 02:00 PM, 03:00 PM, 04:00 PM, and 05:00 PM, covering a standard clinic operating day with a midday break.

D. Appointment Lifecycle and Status Management

The appointment lifecycle is managed through four status states: Pending, Confirmed, Completed, and Cancelled. State transitions are enforced at the application logic layer with role-based controls. Pending appointments can transition to Confirmed by Admin or Doctor, or to Cancelled by Admin, Doctor, or Patient. Confirmed appointments can transition to Completed by Doctor only, or to Cancelled by Admin, Doctor, or Patient. Completed and Cancelled appointments have no further transitions and show no action buttons.

Table II presents the complete status transition matrix.

TABLE II: APPOINTMENT STATUS TRANSITIONS

From Status	To Status	Who Can Perform
Pending	Confirmed	Admin, Doctor
Pending	Cancelled	Admin, Doctor, Patient
Confirmed	Completed	Doctor
Confirmed	Cancelled	Admin, Doctor, Patient
Completed	—	No further transitions
Cancelled	—	No further transitions

E. Data Model Design

The data model is implemented as a JavaScript object with two main collections. The User Record stores id (integer, auto-increment), name (string), email (string, unique login key), password (string), role (admin/doctor/patient), phone (string), and role-specific fields: speciality, experience, and fee for doctors; age for patients. The Appointment Record stores id (integer, auto-increment), patientId (foreign key to user), doctorId (foreign key to user), date (YYYY-MM-DD), time (string), reason (string), status (pending/confirmed/completed/cancelled), and createdAt (YYYY-MM-DD).

F. User Interface Design System

The user interface follows a professional healthcare aesthetic implemented through a CSS custom properties design system. The primary color is deep forest green (#1a5c4a), chosen for its associations with health, trust, and professionalism as documented by Caglar and Mealha [9]. The accent color is warm coral-orange (#e8704a) used for call-to-action elements. The background is warm off-white (#f4f1ec) to provide an approachable rather than clinical feel, and body text is near-black (#1a1a2e) for maximum readability. Playfair Display serif is used for headings, providing distinctive professional character, while DM Sans is used for body text and UI controls. Status badges use universally understood color conventions: yellow for Pending, green for Confirmed, blue for Completed, and red for Cancelled.

IV. IMPLEMENTATION

A. Technology Stack

The technology stack was selected to maximize portability and ease of deployment. All selected technologies are built into modern web browsers and require no installation. Table III summarizes the complete technology stack.

TABLE III: APPLICATION TECHNOLOGY STACK

Component	Technology	Function
Frontend Structure	HTML5	Semantic page structure and forms
Styling	CSS3 with Custom Properties	Theming, responsive layout
Layout Engine	CSS Grid and Flexbox	Multi-column and card layouts
Application Logic	ES6+ JavaScript	Role routing, booking, state management
Data Storage	JavaScript Object (in-memory)	Users and appointments store
Typography	Google Fonts (Playfair Display, DM Sans)	Professional typeface pairing
Deployment	Single HTML file	Static hosting or local file open

B. Frontend Implementation

The implementation was completed as a single self-contained HTML file of approximately 1,200 lines including all CSS and JavaScript inline. This single-file approach was deliberate: the entire application can be opened in any modern browser without any server and run offline. The codebase is organized into a CSS styles section (approximately 350 lines), an HTML structure section (approximately 250 lines defining the static page skeleton), and a JavaScript section (approximately 600 lines) organized into logical subsections covering data store initialization, authentication, navigation, admin modules, doctor modules, patient modules, booking functions, status management, and utility functions.

The login screen is a centered card on a full-screen gradient background containing the clinic logo, a Login/Register tab bar, form fields, and a Quick Demo Login section with preset one-click buttons for each user role to facilitate demonstration. The top navigation bar is fixed to the viewport and displays the clinic branding on the left and a user pill showing the logged-in user's initials, name, role label, and logout button on the right.

C. Admin Module Implementation

The Admin Doctor Management module provides a modal form for adding new doctors collecting full name, speciality (dropdown with nine options), email, phone, password, years of experience, and consultation fee. On submission, required fields are validated, a duplicate email check is performed, and the new user record is added to the data store with the role set to doctor. Doctor removal deletes both the doctor user record and all associated appointments after browser confirmation to prevent accidental deletion. The Patients module shows a read-only table of all registered patients with their appointment counts. The Dashboard shows four real-time statistics cards: total doctors, total patients, today's appointments, and confirmed appointments.

D. Doctor Module Implementation

The Doctor Appointments module shows a table of all appointments where the doctorId matches the logged-in doctor's ID. Action buttons are dynamically rendered based on appointment status: Pending appointments show Confirm and Cancel; Confirmed appointments show Complete and Cancel; Completed and Cancelled appointments show no actions. Status changes call the updateStatus() function which updates the appointment record in the data store and re-renders the active page. Statistics cards show today's appointments, confirmed appointments, and total appointments for that doctor.

E. Patient Module Implementation

The Patient Book Appointment module renders a grid of doctor cards by filtering the users store for records with the doctor role. Each card shows the doctor's name, speciality, phone, experience, and consultation fee. Clicking Book Appointment on a card calls the openBooking() function with the doctor's ID, which stores the doctor ID, populates the modal, sets the date input minimum to today, and renders the time slot grid. The My Appointments module shows the

patient's own appointment history with cancellation options and statistics cards for total, confirmed, and pending counts.

V. EXPERIMENTAL RESULTS

A. Functional Evaluation

The system was evaluated through ten test cases covering all major functional paths. All ten test cases produced the expected results. Table IV presents the complete functional evaluation results.

TABLE IV: FUNCTIONAL EVALUATION RESULTS

ID	Action	Input	Expected Result	Actual Result	Status
TC-01	Admin login	admin@clinic.com / admin123	Admin dashboard displayed	Admin dashboard displayed	PASS
TC-02	Doctor login	dr.sharma@clinic.com	Doctor dashboard displayed	Doctor dashboard displayed	PASS
TC-03	Patient login	rahul@gmail.com / pat123	Patient dashboard displayed	Patient dashboard displayed	PASS
TC-04	Invalid login	wrong@email.com / wrongpass	Error toast shown	Invalid email or password toast	PASS
TC-05	Patient registration	New email, valid fields	Account created, auto-login	Registered and logged in	PASS
TC-06	Patient books appointment	Doctor, date, slot, reason	Appointment created, Pending status	Appointment visible in dashboard	PASS
TC-07	Double booking prevention	Same slot already taken	Slot shown as disabled	Slot displayed as booked, unclickable	PASS
TC-08	Doctor confirms appointment	Click Confirm on Pending	Status changes to Confirmed	Badge changed, Confirm button removed	PASS
TC-09	Admin removes doctor	Click Remove on doctor row	Doctor and appointments deleted	Doctor and records removed	PASS
TC-10	Patient cancels appointment	Click Cancel on own appointment	Status changes to Cancelled	Badge changed, no further actions shown	PASS

All ten test cases passed. Role-based routing correctly directed each user type to their appropriate dashboard. The double-booking prevention mechanism correctly disabled already-booked slots. The appointment lifecycle transitions enforced role-based permissions at each state change.

B. Performance Analysis

Table V presents response time measurements for all major system operations, benchmarked

against traditional manual appointment management.

TABLE V: PERFORMANCE METRICS

Metric	Value	Notes
Initial page load time	< 1 second	Single HTML file, no external requests
Role navigation switch time	< 50 ms	Pure DOM manipulation, no server round-trip
Appointment booking time (UI)	< 100 ms	In-memory data operation
Time slot rendering	< 20 ms	Small dataset, simple loop
Doctor card grid rendering	< 30 ms	Typically 3–10 doctors
Login processing time	< 10 ms	Array search, no network call
Traditional manual booking	5–10 minutes	Phone call, register entry
Improvement factor	3,000x+	Digital vs. manual comparison

The improvement factor exceeding 3,000 times over traditional manual booking reflects the elimination of the phone call, receptionist availability dependency, and manual register entry steps. Beyond raw speed, the system also enables booking outside business hours, which the manual process cannot accommodate at all.

C. Manual vs. Digital Feature Comparison

Table VI presents a structured comparison of feature capabilities between the traditional manual approach and the MediCare Clinic system.

TABLE VI: COMPARISON OF MANUAL AND DIGITAL APPOINTMENT MANAGEMENT

Feature	Manual (Paper Register)	MediCare Clinic
Booking method	Phone call during clinic hours	Self-service, any time
Double booking risk	High	None (slot blocking)
Schedule visibility for doctor	Only at clinic	Accessible anywhere
Record durability	Paper can be lost or damaged	Digital, persistent in session
Patient history	Manual search through pages	Instant filter by patient
Admin oversight	Count entries manually	Real-time statistics dashboard
Cancellation	Phone call required	One-click self-service
Operating hours for booking	Business hours only	24/7

VI. CONCLUSION AND FUTURE WORK

This paper presented MediCare Clinic, a complete web-based clinic appointment booking system designed for small independent healthcare facilities. The system addresses the six fundamental problems of traditional manual appointment management — scheduling conflicts, lack of patient self-service, poor real-time visibility for doctors, fragile paper records, absence of status tracking, and poor administrative oversight — through a clean role-based digital solution accessible from any web browser on any device.

The key technical contributions of this work are: a complete three-role access control system implemented within a single self-contained HTML file using standard browser technologies without any framework or backend dependency; a dynamic time slot rendering system that automatically disables booked slots based on in-memory appointment data, effectively eliminating double-booking; a four-state appointment lifecycle with role-controlled transitions enforced at the application logic layer; a mobile-responsive design system using CSS custom properties and CSS Grid appropriate for professional healthcare application aesthetics; and zero-infrastructure deployment enabling the entire application to be run from a single file with no server, no database, and no internet connection required.

Functional evaluation across ten test cases demonstrates 100% correctness across all major user flows. Performance benchmarking shows that the digital system completes operations in under 100 milliseconds compared to 5–10 minutes for the traditional manual process.

Future work will pursue several directions. First, integrating a backend server using Node.js with Express and a MySQL or MongoDB database would make data persistent across browser sessions and support simultaneous multi-device access. Second, adding email and SMS notifications via services such as SendGrid and Twilio would deliver automatic appointment reminders,

reducing patient no-show rates. Third, integrating a payment gateway such as Razorpay would allow patients to pay consultation fees at the time of booking. Fourth, developing a native Android application using Flutter would provide a more convenient interface for patients who prefer a dedicated mobile app. Fifth, adding an electronic prescription and medical records module would extend the system from a scheduling tool to a comprehensive clinic management solution. Sixth, implementing a multi-language interface supporting Hindi and regional Indian languages would improve accessibility for patients and staff who are more comfortable in their native language than in English.

ACKNOWLEDGMENT

I would like to sincerely thank Ravi Kumar Chaudhary, Assistant Professor, Department of Computer Science and Engineering, Raffles University, for his valuable guidance, continuous support, and helpful suggestions throughout this project.

I am also grateful to Rajendra Singh, Dean, Department of Computer Science and Engineering,

Raffles University, for his encouragement, academic support, and motivation during this research work.

REFERENCES

[1] B. Chaudhry, J. Wang, S. Wu, M. Maglione, W. Mojica, E. Roth, S. C. Morton, and P. G. Shekelle, "Systematic review: Impact of health information technology on quality, efficiency, and costs of medical care," Annals of Internal Medicine, vol. 144, no. 10, pp. 742–752, 2006.

[2] Public Health Foundation of India, "Digital Health in India: Current Landscape and Future Directions," PHFI Publications, 2021.

[3] D. F. Sittig and H. Singh, "A new sociotechnical model for studying health information technology in complex adaptive healthcare systems," Quality and Safety in Health Care, vol. 19, suppl. 3, pp. i68–i74, 2010.

[4] E. Marcotte, "Responsive web design," A List Apart, vol. 306, 2010. [Online]. Available: https://alistapart.com/article/responsive-web-design/

[5] D. Ferraiolo and R. Kuhn, "Role-based access control," in Proc. 15th National Computer Security Conference, pp. 554–563, 1992.

[6] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-based access control models," IEEE Computer, vol. 29, no. 2, pp. 38–47, 1996.

[7] M. S. Mikowski and J. C. Powell, Single Page Web Applications: JavaScript End-to-End. Manning Publications, 2013.

[8] J. Nielsen, "Heuristic evaluation," in Usability Inspection Methods, J. Nielsen and R. L. Mack, Eds. John Wiley and Sons, 1994.

[9] S. Caglar and O. Mealha, "Color associations in healthcare user interface design: A systematic literature review," Journal of Medical Systems, vol. 43, no. 7, pp. 1–12, 2019.

[10] Central Bureau of Health Intelligence, National Health Profile. Ministry of Health and Family Welfare, Government of India, 2022. [Online]. Available: https://cbhi.nic.in

[11] World Wide Web Consortium, "HTML Living Standard," 2023. [Online]. Available: https://html.spec.whatwg.org/

[12] Mozilla Developer Network, "CSS Custom Properties for Cascading Variables," 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties

[13] Mozilla Developer Network, "JavaScript — MDN Web Docs," 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[14] Google Fonts, "Playfair Display and DM Sans Font Families," 2024. [Online]. Available: https://fonts.google.com