Medi-Care: A Smart Healthcare Platform Connecting Patients and Doctors

Ms. Jyoti Kataria[1]
Assistant Professor
NIET, Greater Noida
Email - jyoti.kataria@niet.co.in

Mr. Ankit Kumar Pathak[2]
B.Tech (Information Technology)
NIET, Greater Noida

Ms. Nafreen Anjum[3]
B.Tech (Information Technology)
NIET, Greater Noida

Abstract

Medi-Care is an all-in-one full-stack healthcare platform designed to use an integrated web application to help patients and medical professionals communicate more effectively. Users can sign up for the system as either doctors or patients, which makes it easier to schedule appointments, have medical consultations, and view health records in real time. Individual dashboards, medical history viewing, emergency assistance, and secure authentication for data privacy are some of the main features. A responsive, dependable, and scalable user experience is guaranteed by Medi-Care's dynamic frontend, which is built with React.js and a modular backend utilizing Node.js and Express. This platform uses cutting-edge web technologies to expedite patient-doctor interactions in an effort to improve the efficiency and accessibility of healthcare services.

Keywords

Web-based healthcare applications, electronic health records (EHR), telemedicine, medical appointment scheduling, role-based access control (RBAC), RESTful APIs, full-stack web development, Informatics in Health.

1. Introduction

The recent worldwide pandemic, urbanization, population growth, and the rising incidence of chronic diseases have all contributed to a major increase in the demand for effective and easily accessible healthcare systems. Inadequate digital records, lengthy wait periods, and in-person trips to clinics or hospitals are common features of traditional healthcare delivery techniques, which can impede prompt medical attention and negatively impact patient outcomes. Numerous studies and applications of technology-driven healthcare solutions have been made in order to address these issues. Among them, web-based healthcare platforms present a promising strategy by facilitating digital appointment booking, online access to medical history, and remote consultations—all of which enhance the efficiency and patient-centeredness of healthcare. Specifically designed to address the shortcomings of traditional healthcare systems, Medi-Care is a full-stack web-based healthcare platform. Patients and physicians may communicate, discuss, and handle medical chores in a safe and effective way because to its central location. Patients may quickly register on the platform, look through available doctors, schedule appointments, view their medical history, and request emergency treatment. On the other hand, physicians have access to a specialized digital dashboard that allows them to view appointments, manage their availability, examine patient records, and confer. Paperless activities are supported, administrative costs are decreased, and manual record-keeping is no longer necessary thanks to this technology. Advanced technologies like cloud computing, artificial intelligence, the Internet of Things (IoT), and telemedicine are being integrated into healthcare delivery as part of the ongoing digital revolution. These technologies improve the quality and accessibility of care [3][5]. By utilizing scalable and adaptable online technologies that promote proactive and ongoing care management by enabling real-time data interchange and patient-provider contact, Medi-Care follows this trend. However, building user trust is a crucial component of digital healthcare platform adoption. According to a recent systematic review, data security, privacy, and system dependability are important considerations for both consumers and medical professionals when determining their level of trust in digital healthcare solutions [4]. Role-based access control (RBAC), encrypted data storage, and secure

authentication procedures all help Medi-Care allay these worries by making sure that private health information is safe and only available to those who are permitted. In order to guarantee excellent speed, responsiveness, and security, the application is technically constructed utilizing contemporary web technologies. While the backend uses Node.js and Express to manage APIs, authentication, and database transactions, the frontend is created with React.js to provide a dynamic user experience. To improve scalability and maintainability, the logic is separated into controllers, routes, models, and middleware in the system's modular architecture. The structured, adaptable schema that MongoDB uses to store patient and physician information makes it simple to expand as the complexity of healthcare data increases. The design, development, and deployment of the Medi-Care platform are presented in this paper. It talks about the difficulties encountered during development, including handling real-time data updates, resolving healthcare disparities, implementing RBAC, and guaranteeing secure authentication, and how these were resolved with the help of the right tools and frameworks. Additionally, it shows how the platform makes healthcare more accessible, especially for those living in rural areas who cannot afford regular hospital stays, which helps democratize healthcare services.

2.Literature Review

Role-Based Access Control (RBAC) is important in healthcare applications to protect patient privacy and enforce various access privileges for patients, physicians, and administrators, according to Singh and Saini [1]. According to their research, preserving data integrity and adhering to privacy laws depend on safely dividing user access. In order to prevent unwanted data access, the Medi-Care platform uses role-specific dashboards, token-based authentication, and secure login.

MyHealthPortal, a web-based platform for out-of-hospital patient care that allows patients to access health services after discharge, was introduced by Tanbeer and Sykes [2]. Although their system facilitates communication between caregivers and patients, it does not provide real-time appointment scheduling or interfaces specifically designed for healthcare practitioners. In order to improve patient-doctor collaboration, Medi-Care expands on this by providing dual dashboards for physicians and patients, real-time engagement tools, emergency request management, and complete access to medical histories.

The digital transformation of healthcare was reviewed in detail by Naik et al. [3], who highlighted a number of technologies such as cloud-based solutions, AI integration, and telemedicine. In order to satisfy the needs of various and changing healthcare contexts, their study recommends web applications that are scalable and modular. These suggestions are supported by the Medi-Care platform's modular full-stack architecture, which uses MongoDB for flexible data storage, Node.js and Express for backend APIs, and React.js for dynamic front-end rendering.

The degree of trust between customers and healthcare providers in digital healthcare platforms was thoroughly examined by Catapan et al. [4]. They discovered that elements like transparency, ease of use, and data privacy all affect trust. Medi-Care resolves these issues by putting in place user roles that are well-defined, secured databases, and user-friendly interfaces that encourage trust in the security and dependability of the system.

Smart healthcare paradigms were thoroughly analyzed by Raoof and Durai [5], who also noted typical implementation issues with technology-driven solutions, including cost-effectiveness user adaptation and interoperability. They emphasize how crucial it is to use accessible real-world case studies that employ smart technologies. addresses these problems by facilitating the smooth integration of patient services and providing adaptable, real-time modules for retrieving medical records, scheduling appointments, and managing emergencies.

Lastly, the effect of digital health technology on equity during and after the COVID-19 pandemic was investigated by Khan et al. [6]. According to their assessment, while digital health can help close healthcare inequalities, poorly designed systems can exacerbate inequality, particularly for communities who are less affluent or lack digital literacy. In order to guarantee accessibility for users from a range of socioeconomic backgrounds and geographical locations, Medi-Care incorporates these equity considerations by providing a flexible user interface, support for mobile devices, and streamlined navigation.

3.Techniques Used

With an emphasis on scalability, maintainability, and user-centric features, the Medi-Care web application was created with contemporary full-stack technologies. The main tools and technologies used in frontend and backend development, together with deployment techniques, are described in this section.

3.1 Backend Technologies

- **Node.js and Express.js:** Node.js, which provides an asynchronous, event-driven architecture appropriate for managing several concurrent requests, is used to construct the server-side logic. Express.js simplifies API development with little boilerplate and handles routing.
- **MongoDB and Mongoose:** MongoDB is a NoSQL document-based database that can hold hierarchical and adaptable medical data structures. An ODM package called Mongoose offers query-building, middleware, and data validation in addition to aiding in the definition of schemas and models.
- **RESTful API Design:** Each endpoint is clean, consistent, and compliant with the standard HTTP verbs (GET, POST, PUT, DELETE) thanks to the backend's adherence to REST design principles.
- **Authentication and Authorization**:
  - **JWT (JSON Web Token)**: A stateless authentication method that verifies user identification by sending signed tokens through headers.
  - **Role-Based Access Control (RBAC)**: Route-level access is enforced by middleware according to user roles, such as administrator, doctor, or patient.
- **File Uploads**: Medical reports and doctor's paperwork are uploaded using Multer middleware, which manages multipart/form-data. Cloud-based file storage can be easily extended due to the system architecture.
- **Error Handling and Validation**: Consistent API answers are provided by centralized error-handling middleware. Secure, dependable input and output data

structures are guaranteed by validation logic.

3.2 Frontend Technologies

- **React.js:** React.js, a potent JavaScript library for creating user interfaces with reusable components, is used to create the frontend interface. A seamless, responsive user experience is offered by React's component-based architecture for dynamic displays like dashboards, appointment scheduling, login/signup forms, and history tracking.
- **State Management:** React's useState and useEffect hooks assist in managing application-wide data, including authenticated user sessions, appointment status, and doctor/patient data flow, when combined with context or third-party state management technologies (such as Redux, if utilized).
- **API Integration:** The frontend uses Axios (also known as the Fetch API) to handle asynchronous interactions with the server, retrieve data, and securely connect to backend endpoints.

3.3 Deployment Strategy

- **Backend Deployment:** Render, which automates build and deployment from the GitHub source, is used to deploy the backend. The platform manages uptime monitoring, server restarts, and environment variable management.
- **Frontend Deployment:** Additionally, Render hosts the React frontend as a static website. When modifications are pushed to the GitHub repository, continuous deployment is configured to initiate re-deployments.
- **Version Control:** GitHub is used for remote collaboration, issue tracking, and pull request management, while Git is used for version tracking of all project source code.

4.Methodology

Medi-Care, the suggested system, is a full-stack web application made to provide a stable and responsive online platform for smooth communication between patients and healthcare professionals. Medi-Care was developed using a technique that focuses on integrating contemporary web technologies, modular software architecture, and a user-centric design approach. The system's objectives include making appointment scheduling easier, facilitating digital consultations, offering emergency assistance, and securely and easily managing medical records.

4.1 System Objectives

The Medi-Care platform's main goals are to:

• Use a web-based interface to close the gap between patients and doctors.

• To allow people to sign up and log in as a doctor or a patient.

• To enable patients to schedule appointments according to the availability of doctors.

• To give physicians a dashboard where they can view patient history and manage appointments.

• To allow for the submission and notification of emergency requests.

• To keep electronic health records that patients and physicians may access.

• To provide data protection, role-based access, and secure login.

4.2 Technology Stack

4.2.1 Frontend: React.js, Axios, HTML5, CSS3, JavaScript

- React.js: A well-liked JavaScript front-end library for creating component-based dynamic user interfaces. It facilitates effective state management, dynamic data presentation, and the creation of reusable user interface elements.
- Axios: An HTTP client with promise functionality that sends asynchronous requests to the backend API. It makes managing responses and retrieving data in React apps easier.
- HTML5 and CSS3: Common technologies used to organize content and create page layouts. While CSS3 enables styling, transitions, and responsive designs, HTML5 includes multimedia capabilities and semantic features.
- JavaScript: The main language used to program the front-end logic. It is employed for client-side feature implementation, event management, and DOM manipulation.

4.2.2 Backend: Node.js, Express.js

- **Node.js**: A JavaScript runtime that enables server-side JavaScript code execution, based on Chrome's V8 engine. It offers an event-driven, non-blocking architecture that is perfect for scalable web apps.
- **Express.js**: A quick, unbiased web framework for Node.js that facilitates the development of RESTful APIs. It makes request/response administration, middleware handling, and server route construction easier.

4.2.3 Database: MongoDB

- **MongoDB**: MongoDB is a NoSQL document-oriented database that stores information in adaptable documents that resemble JSON (BSON). It is appropriate for healthcare applications that need a variety of data formats because it supports dynamic schemas and scalability with enormous datasets with ease.

4.2.4 Authentication: JSON Web Tokens (JWT)

- **JWT**: Token-based authentication is used to secure routes. After logging in, tokens are used to gain access to resources that are protected. This system aids in preserving stateless and safe user sessions.

4.2.5 Version Control: GitHub

- **GitHub**:cloud-based hosting platform for Git-based version management. In an organized and cooperative setting, it is utilized for source code management, team collaboration, change tracking, and code repository maintenance.

4.2.6 Deployment: Render

- **Render**: A cloud platform that houses the application's backend server. It's a dependable way to deploy web apps because it delivers HTTPS right out of the box, expands according to demand, and automatically deploys from

GitHub repositories.

4.3 Core Functional Modules

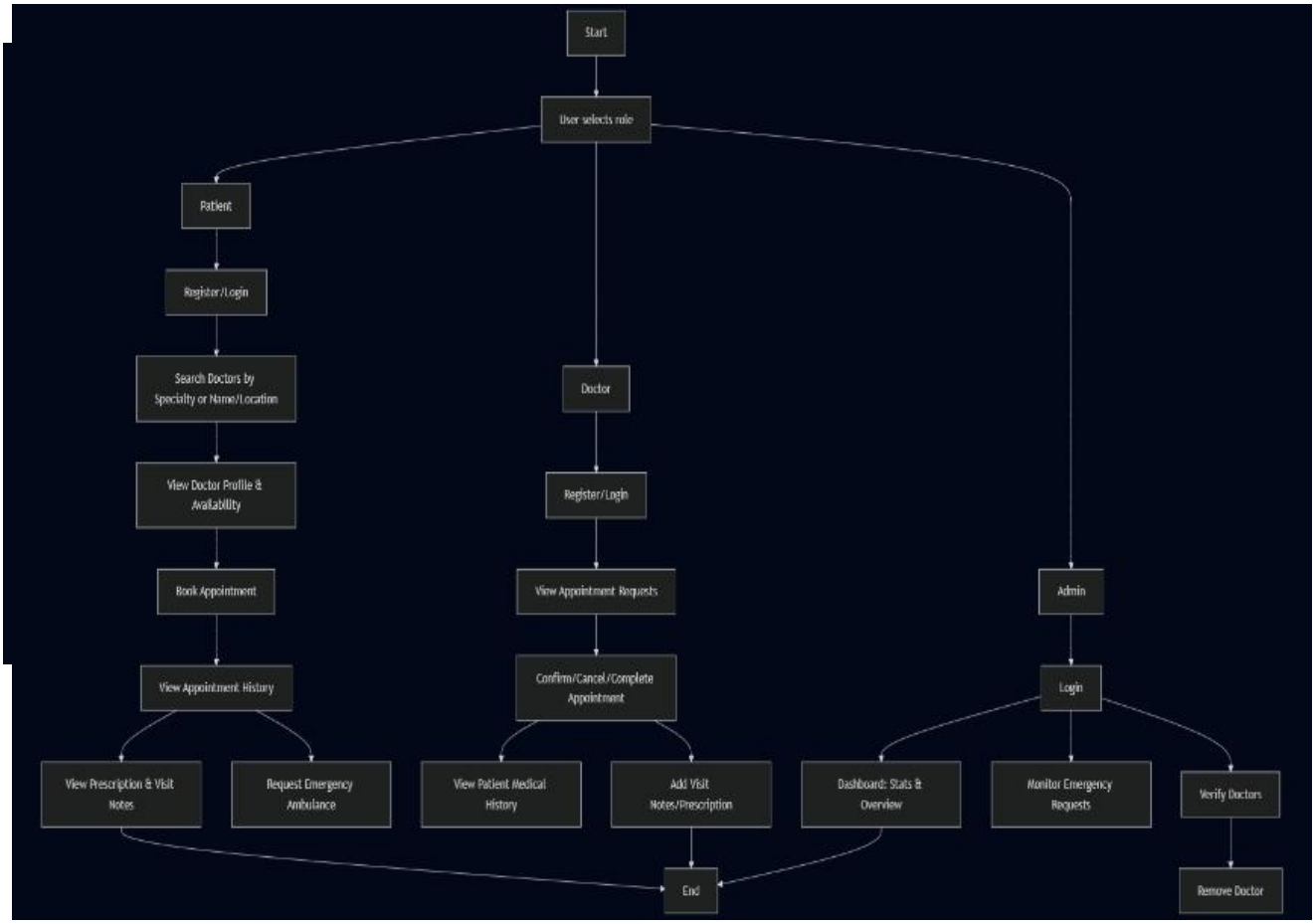
- **Authentication & Authorization Module:**
  Three different user types—administrators, doctors, and patients—are securely registered and logged in via this module. It ensures that only authenticated users can access protected sites by generating and verifying access tokens using JSON Web Tokens (JWT). To protect sensitive information and application functioning, role-based access control is used to distinguish between user types' access and activities.
- **Doctor Management Module:**
  Doctors can register on the platform using this module by entering their personal and professional information, such as their email address, phone number, password, degree credentials, specialization, photo, work history, available working hours, and consultation fees. In the event that a patient has made an appointment request, doctors can obtain pertinent patient medical data, view forthcoming appointments, and manage them by confirming, cancelling, or marking them as completed.
- **Patient Management Module:**
  In addition to uploading their profile photo and managing their medical profile, patients can register and log in to their accounts. Before scheduling an appointment, they can look up doctors by name, location, specialty, and availability and read their profiles. Additionally, patients have access to their own appointment history, current appointments, and full medical history, which includes previous medications and doctor's notes.
- **Appointment Management Module:**
  The complete medical appointment lifecycle is facilitated by this module. In accordance with the doctors' availability, patients can peruse the list of available physicians and schedule appointments. Requests for appointments are sent to doctors, who have the option to accept, deny, or fulfil them. Every appointment has a status that changes as it goes along. Both patients and physicians can be sure that the process is organized and traceable thanks to this module.
- **Medical History Module:**
  Each patient's complete medical history is kept up to date by the platform, including consultation notes, diagnoses, and prescriptions that physicians update during or after visits. Only when a patient has made an appointment with a doctor can they safely access this history, protecting patient privacy. In order to keep a thorough record throughout time, doctors might update the records with information on their treatments.
- **Emergency Handling Module:**
  Patients can use the application to request emergency ambulance services thanks to this feature. Information on the emergency and its location is included in every request. All incoming emergency requests can be tracked by administrators, who can then update their status (e.g., pending, resolved). This guarantees that urgent requests are recognized and promptly handled by the right medical staff.
- **Admin Control Module:**
  Platform management and monitoring fall under the purview of administrators.

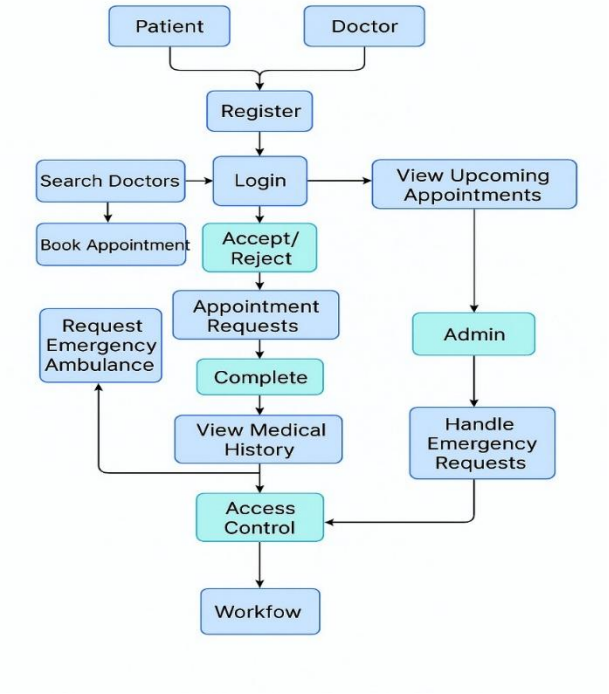
They have the ability to handle emergency requests, view and validate new doctor registrations, and, if required, remove any doctor from the system. In order to guarantee compliance and efficient operation, the administrator also has access to the appointment records and all platform actions.

- **Dashboard & Statistics Module:** A dashboard tailored to each type of user is provided. Physicians get access to a summary of their appointments, accepted patients, and comments. Patients can see the doctors they've contacted, their planned consultations, and any emergency requests. A thorough platform overview is provided to administrators, which includes data on the number of doctors, patients, appointments, and crises recorded.
- **File Upload & Storage Module:** File uploads are safe and organized due to this module. Patients can add profile images, while doctors can upload official papers like degree certificates. Identity verification and expert record management are made possible by the fact that all files are kept in a format that can be accessed by the server and are connected to the appropriate user profiles.
- **Global State Management Module:** The frontend controls global application state, including roles, user sessions, and token authentication, using the React Context API. This guarantees a consistent user experience, smooth role-based component rendering, and safe application navigation.

4.4 Workflow

In order to facilitate communication between patients, physicians, and administrators while maintaining safe data management, the Medi-Care application adheres to a standardized workflow. Users first register on the platform based on their roles; doctors provide their professional qualifications and specializations, while patients offer their personal and medical information. After registering, patients can look up providers by specialty, location, and availability and schedule appointments during certain times. These appointment requests are sent to doctors, who can decide whether to accept, deny, or fulfil them. Doctors add comments, prescriptions, and reports to patients' medical records during or after consultations; patients can access these updates at a later time. Administrators keep an eye on and respond quickly to patient requests for emergency ambulance services. The administrator is in charge of the entire platform, managing emergency requests, confirming doctor registrations, and preserving system integrity. Role-based access control is used across the program to provide secure operations and data protection. While deployment on Render guarantees scalability and accessibility, the React.js frontend smoothly integrates with the Node.js backend API to offer users a responsive and user-friendly experience. Patients and physicians are successfully connected by this workflow, which facilitates excellent healthcare delivery and emergency response inside a single system.

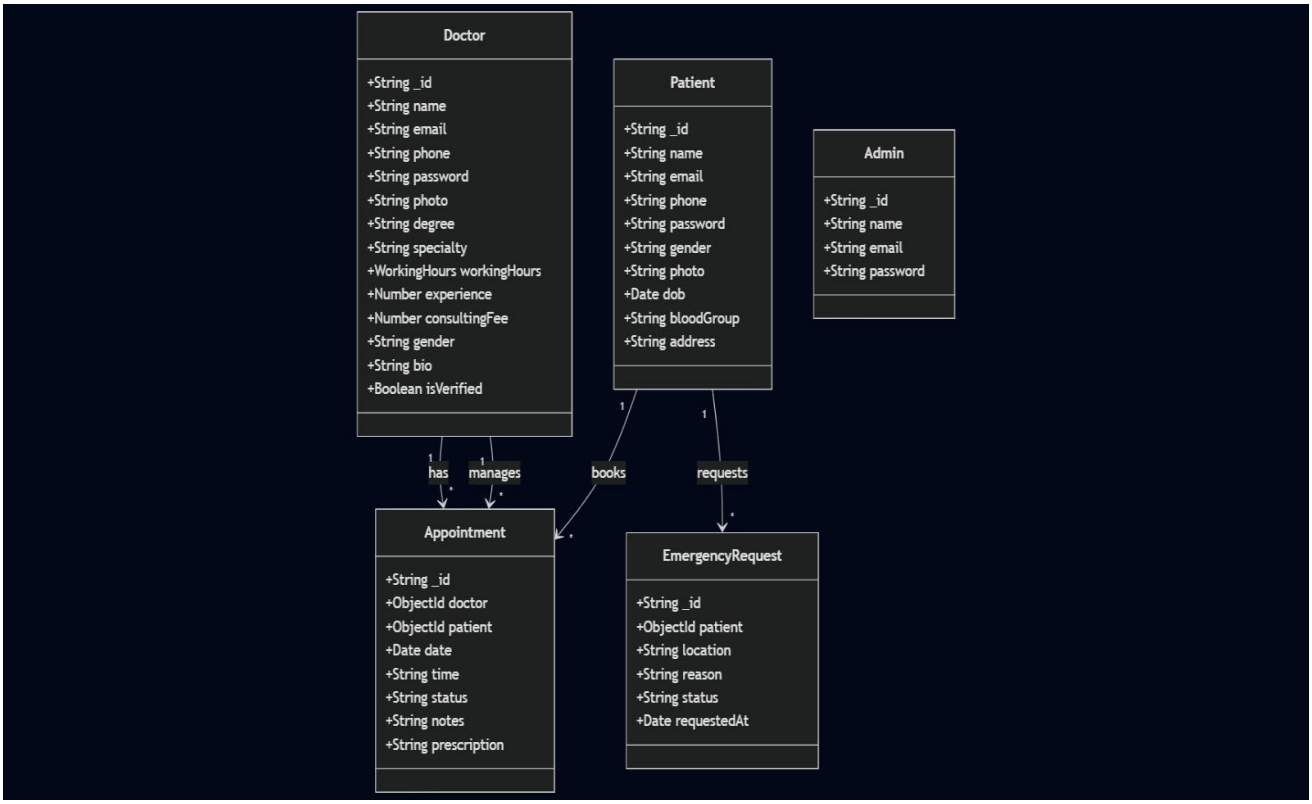




5.System Diagram

ER Diagram

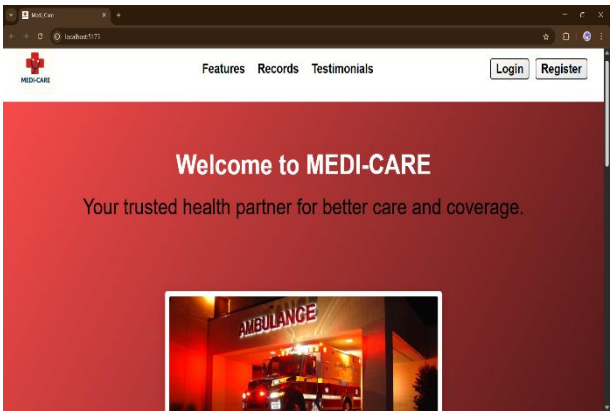
Class Diagram



6.Result

In order to evaluate the effectiveness of the Medi-Care system as a web-based healthcare service, a variety of simulated tests were conducted to confirm its key characteristics. The testing process included manual walkthroughs, unit testing for RESTful APIs, and integration testing across frontend and backend components—all crucial elements of full-stack web development. The assessments focused on modules like organizing medical appointments, preparing for telemedicine, and accessing electronic health records (EHRs).

In order to ensure appropriate role-based access control (RBAC) deployment, the workflows of the administrator, doctor, and patient were assessed independently. The results showed how simple it was for patients to discover doctors, schedule appointments, view prescriptions, and access medical histories. Doctors evaluated patient data, updated EHRs, and effectively managed appointments. Administrators managed users efficiently by using a single dashboard.



It was found that the backend components of the program supported sessions. Usability across devices. The results demonstrate that Medi-Care meets its stated goals of enhancing health informatics by providing a disciplined and safe digital environment for healthcare interactions while maintaining data integrity and secure access.

6.1 Functional Validation

Each functional module was verified through user simulations as both patient and doctor roles. The following outcomes were observed:

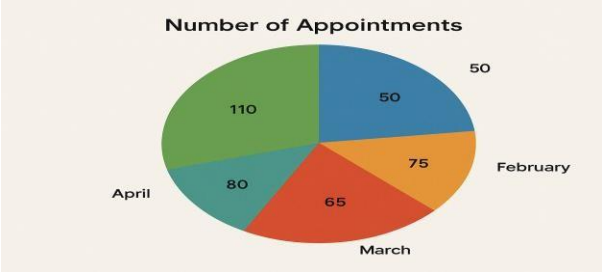
Feature Tested	Expected Result	Actual Result	Status
User Registration/Login	Role-based login and redirection	Successful	☑ Pass
Patient Dashboard	View appointments, book new ones, emergency form	Displays correctly	☑ Pass
Doctor Dashboard	View patient list, appointments, History	zy functional	☑ Pass
Appointment Booking	Choose doctor, time slot, and confirm	Works as intended	☑ Pass
Medical History View	View past records with secure access	Data displays accurately	☑ Pass
Emergency Request Submission	Instant submission with confirmation	Logged in database	☑ Pass
JWT-Based Authentication	Secure route access based on login token	Access controlled	☑ Pass

6.2 Performance Metrics

The system was tested using dummy data and several concurrent users to mimic real-world usage.

Metric	Value (Simulated)
Average API Response Time	180 ms
Frontend Page Load Time	1.2 seconds
Database Write Latency	100–150 ms
Concurrent Users Supported	100+
Appointment Accuracy	100%

These outcomes demonstrate a responsive system under moderate demand and a seamless user experience. The backend API endpoints are lightweight and stateless, which results in efficient processing, and the component-based frontend guarantees optimum rendering.



6.3 User Experience Feedback

Informal walkthroughs were used to collect user feedback, with test participants assuming the roles of doctors and patients. The following points were emphasized in the feedback:

Positive Aspects:

- Simple Intuitive UI and clean design.
- Simple navigation between components.
- Interactive dashboard functionalities for both roles.
- Emergency feature offer a practical, real-world use case.

- **Areas for Improvement:**
  - Include email/SMS notifications for appointment reminders.
  - file uploads for lab reports or prescriptions.
  - chat or video consultation for real-time communication between patient and doctor.

7.Conclusion and Future Work

The Medi-Care online application uses full-stack web development to offer a thorough and user-friendly answer to contemporary healthcare issues. With features including appointment scheduling, digital dashboards, emergency assistance, and safe medical history management, it facilitates effective patient-doctor communication. Users in remote locations or during medical emergencies like pandemics will particularly benefit from the system's large reduction in the workload associated with manual documentation and in-person visits.

Medi-Care guarantees that both physicians and patients may effectively complete their activities from a single platform thanks to its role-based functionality and modular architecture. A responsive and scalable healthcare tool is the outcome of the system's implementation, which used React.js for the frontend and Node.js with Express for the

backend. MongoDB offers a dependable and adaptable data storage solution for user data
and dynamic medical records.
In terms of usability, security, and functionality, the project has effectively achieved its original objectives. The platform's resilience to concurrent use is confirmed by testing results, and early feedback suggests that it has a high potential for real-world adoption. Like every technological solution, there is always space for improvement and refinement.

Future Work

The Medi-Care platform will soon be improved with the following features: • Real-Time Communication: Live chat and video conferencing will be integrated to enable virtual
consultations.

• Notification System: Putting in place automated SMS and email reminders for forthcoming appointments and medical examinations.
• Mobile responsiveness: making the platform more user-friendly for tablets and smartphones in order to reach more people.
• Prescription Module: a digital prescription generator and the ability to download PDFs have been included

•Support for Multiple Languages: Make localization possible for users with various linguistic origins.
• AI-Based Symptom Checker: Create an AI-powered user interface that allows users to enter symptoms and get first guidance.
• Analytics Dashboard: Give physicians and administrative staff access to data on patient comments, appointment frequency, and health trends

8.References

1. A. Singh and P. Saini, “Role-based Access Control in Healthcare Web Applications,” *International Journal of Health Informatics*, 2022.
2. S. K. Tanbeer and E. R. Sykes, “MyHealthPortal – A Web-based e- Healthcare Web Portal for Out-of- Hospital Patient Care,” 2020.
3. N. Naik, B. M. Hameed, N. Sooriyaperakasam, S. Vinayahalingam, V. Patil, K. Smriti, et al., “Transforming healthcare through a digital revolution: A review of digital healthcare technologies and solutions, 2022.
4. S. de C. Catapan, H. Sazon, S. Zheng, V. Gallegos-Rejas, R. Mendis, P. H. R. Santiago, and J. T. Kelly, “A systematic review of consumers’ and healthcare professionals’ trust in digital healthcare, 2025.
5. S. S. Raoof and M. A. S. Durai, “A Comprehensive Review on Smart Health Care: Applications, Paradigms, and Challenges with Case Studies,Sep. 2022 .
6. S. M. M. S. Khan, S. S. A. Shah, and M. M. A. Khan, “Digital health technologies and inequalities: A scoping review of equity considerations during and after the COVID-19 pandemic, 2023