MULTI HOSPITAL COMMUNICATION SYSTEM

A PROJECT REPORT

Submitted by

ARWIN V G	**811722104017**
DHARANISH S	**811722104029**
GIRIPRAKASH K	**811722104044**

in partial fulfillment of the requirements for the award degree of

Bachelor in Engineering

20CS7503 DESIGN PROJECT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

NOVEMBER 2025

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM - 621112

BONAFIDE CERTIFICATE

The work embodied in the present project report entitled "**MULTI HOSPITAL COMMUNICATION SYSTEM"** has been carried out by the students ARWIN V G, DHARANISH S, GIRIPRAKASH K, the work reported herein is original and we declare that the project is their own work, except where specifically acknowledged, and has not been copied from other sources or been previously submitted for assessment.

Date of Viva Voce: ……

Mrs.R. SATHYA, M.E,(Ph.D).,
SUPERVISOR
Assistant professor
Department of CSE
K. Ramakrishnan College of Technology (Autonomous)
Samayapuram-621 112

Mr.R.RAJAVARMAN,M.E,(Ph.D).,
HEAD OF THE DEPARTMENT
Professor
Department of CSE
K. Ramakrishnan College of Technology (Autonomous)
Samayapuram-621 112

INTERNAL EXAMINER **EXTERNAL EXAMNIER**

ABSTRACT

The Multi-Hospital Communication System is designed to enhance collaboration among hospitals and healthcare facilities by enabling secure, efficient, and real-time communication. Instead of sharing patient-specific information, the system focuses on exchanging operational data such as bed and resource availability, equipment requests, ambulance coordination, blood bank status, staff communication, and emergency alerts. This ensures hospitals can coordinate effectively while fully maintaining patient privacy and adhering to healthcare data protection regulations. With encrypted communication, role-based access, and centralized monitoring, the system improves coordination during emergencies, optimizes resource allocation, and reduces delays caused by manual communication methods. Ultimately, the Multi-Hospital Communication System strengthens inter-hospital collaboration and improves overall healthcare workflow efficiency without compromising sensitive patient data.

Keywords: Multi-hospital communication, healthcare collaboration, resource coordination, real-time messaging, emergency alert system, secure healthcare communication, encrypted communication, role-based access control, hospital operations management, and operational data exchange.

ACKNOWLEDGEMENT

We thank our **Dr. N. Vasudevan**, Principal, for his valuable suggestions and support during the course of my research work.

We thank our **Mr.R. Rajavarman,** Head of the Department, **COMPUTER SCIENCE AND ENGINEERING**, for his valuable suggestions and support during the course of my research work.

We wish to record my deep sense of gratitude and profound thanks to my Guide **Mrs.R. Sathya,** Assistant Professor, **COMPUTER SCIENCE AND ENGINEERING** for his keen interest, inspiring guidance, constant encouragement with my work during all stages, to bring this thesis into fruition.

We are extremely indebted to our project coordinator **Mrs.R.Ramasaraswathi, COMPUTER SCIENCE AND ENGINEERING**, for her valuable suggestions and support during the course of my research work.

We also thank the faculty and non-teaching staff members of the Computer Science and Engineering, K. Ramakrishnan College of Technology, Samayapuram, for their valuable support throughout the course of my research work.

Finally, we thank our parents, friends and our well wishes for their kind support.

SIGNATURE

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	**iii**
	LIST OF FIGURES	**vii**
	LIST OF ABBREVIATIONS	**viii**
1	**INTRODUCTION**	**1**
	1.1 PROJECT OVERVIEW	1
	1.2 PROBLEM STATEMENT	2
	1.3 SCOPE OF THE PROJECT	3
	1.4 OBJECTIVES	4
2	**LITERATURE SURVEY**	**5**
3	**EXISTING SYSTEM**	**10**
4	**PROBLEM IDENTIFICATION**	**12**
5	**PROPOSED SYSTEM**	**14**
6	**SYSTEM REQUIREMENTS**	**16**
	6.1 HARDWARE REQUIREMENTS	16
	6.2 SOFTWARE REQUIREMENTS	17
	6.3 HARDWARE DESCRIPTION	18
	6.4 SOFTWARE DESCRIPTION	18
	6.4.1 React js	18
	6.4.2 Html and css	19
	6.4.3 MongoDB	20
	6.4.4 Visual Studio Code	20

7	**SYSTEM IMPLEMENTATION**	**21**
	7.1 LIST OF MODULES	21
	7.2 MODULES DESCRIPTION	21
	7.2.1 User Authentication Module Implementation	22
	7.2.2 Resources Status Management Module	23
	7.2.3 Request Handling Module Implementation	24
	7.2.4 Alert Broadcasting Module Implementation	24
	7.2.5 Admin Monitoring and Reporting Module	25
8	**SYSTEM TESTING**	**26**
	8.1 TESTING STEPS	26
	8.2 TYPES OF TESTING	27
	8.2.1 Unit Testing	28
	8.2.2 Integration Testing	28
	8.2.3 System Testing	29
	8.2.4 User Acceptance Testing	29
9	**RESULT AND DISCUSSION**	**30**
10	**CONCLUSION AND FUTURE SCOPE**	**31**
	APPENDIX A - SOURCE CODE	**33**
	APPENDIX B - SCREENSHOTS	**42**
	REFERENCE	**45**

LIST OF FIGURES

FIGURE	FIGURE NAME	PAGE No.
B.1	LOGIN PAGE	43
B.2	ADMIN DASHBOARD	43
B.3	RESOURCE FINDER	44
B.4	KNOWLEDGE BASE	44
B.5	RESOURCE REQUEST PAGE	45

LIST OF ABBREVIATIONS

MHCS	-	Multi Hospital Communication System
ID	-	Identification
UAT	-	User Acceptance Testing
HIPAA	-	Health Insurance Portability and Accountability Act
GDPR	-	General Data Protection Regulation
HIS	-	Hospital Information System
HMS	-	Hospital Management System
HER	-	Electronic Health Record
PHI	-	Protected Health Information
UI	-	User Interface
UX	-	User Experience
API	-	Application Programming Interface
RDBMS	-	Relational Database Management System
NoSQL	-	Not Only Structured Query Language
HTTP	-	Hyper Text Transfer Protocol
HTTPS	-	Hyper Text Transfer Protocol Secure
HTML	-	Hyper Text Markup Language
CSS	-	Cascading Style Sheets
JS	-	JavaScript
JSON	-	JavaScript Object Notation
RBAC	-	Role-Based Access Control

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The “Multi-Hospital Communication System” is intended to serve as a dedicated platform for operational coordination between multiple hospitals that are part of a wider healthcare network or operate within the same region. In most real-world settings, hospitals frequently need to communicate with one another to manage referrals, identify available capacity, request specialized equipment, arrange blood components, and coordinate responses during emergencies. At present, this coordination largely relies on informal mechanisms such as telephone calls, email exchanges and generic messaging applications, which are not designed for structured, auditable or privacy-aware communication. The proposed system aims to replace these fragmented approaches with a single, organized solution that supports real-time updates and controlled access. In this project, the Multi-Hospital Communication System is conceptualized as a web-based application accessed by authorized staff from each participating hospital. Every hospital is registered as an entity in the system, and designated users such as hospital coordinators, department representatives and administrators are given login credentials with appropriate roles. Once authenticated, these users can publish and view operational information including current resource availability, equipment shortages, blood bank status and general emergency alerts. All interactions with the system are recorded in a central database, allowing administrators to review historical data, understand how decisions were made, and identify areas where coordination can be improved.

1.2 PROBLEM STATEMENT

Hospitals today face growing pressure to deliver high-quality care using limited resources while also responding effectively to surges in demand caused by accidents, outbreaks or other emergencies. Although individual institutions may have reasonably well-developed internal information systems, the mechanisms used for inter-hospital coordination remain largely manual and ad-hoc. When one hospital needs to locate an available intensive care bed, specialized equipment or specific blood component, staff members must typically place a series of phone calls to personal contacts at other hospitals, send unstructured messages through social media applications or rely on incomplete email chains. This process is slow, difficult to track and highly dependent on the availability and personal networks of individual staff members.

The lack of a unified, institutional platform for inter-hospital operational communication leads to several concrete problems. First, valuable time is lost while information is gathered from multiple sources, which can be critical in emergencies where each minute matters. Second, there is no single, authoritative view of the overall capacity and resource status across the network, so decisions are often made using outdated or incomplete information. Third, since most communication occurs through informal channels, there is minimal logging or documentation, making it challenging to review how decisions were taken or to derive lessons from past incidents. Finally, the use of generic messaging tools increases the risk that staff will accidentally share patient-identifiable details across institutions in ways that are not compliant with privacy regulations or institutional policies.

1.3 SCOPE OF THE PROJECT

The scope of this project is carefully limited to the design and implementation of a prototype communication platform that facilitates operational coordination between hospitals. The system provides features for hospital registration, user authentication, role assignment, resource availability updates, equipment request handling, blood bank status sharing and emergency alert broadcasting. All these features are realized through a web-based user interface that can be accessed within a secure network environment using standard web browsers. The project focuses on ensuring that the core workflows related to posting updates, viewing shared information and responding to alerts are intuitive, reliable and adequately documented.

In defining the scope, it is equally important to identify what is intentionally excluded. The Multi-Hospital Communication System is not intended to function as a full Hospital Management System or Electronic Health Record platform. It does not manage patient registration, clinical documentation, billing, laboratory results or other internal departmental processes. Instead, it is conceived as a complementary layer that can coexist with existing hospital systems, providing an additional channel for operational information sharing at an inter-institutional level. Integration with existing hospital databases, large-scale performance optimization, and deployment in a geographically distributed production environment are considered beyond the immediate scope of the present work and are proposed as directions for future enhancement once the prototype has been validated.

1.4 OBJECTIVES

The broad vision of the Multi-Hospital Communication System is broken down into a set of specific, measurable objectives so that the success of the project can be evaluated systematically. The first objective is to conduct a detailed study of the current methods used for inter-hospital operational communication and to identify the limitations and risks associated with those methods. This involves understanding how different hospitals currently coordinate resource sharing, what information is typically exchanged, and which pain points are most frequently encountered by staff. The findings from this investigation form the foundation for defining the functional requirements of the new system.

The second objective is to design an architecture for a web-based platform that can support the identified requirements in a modular and scalable manner. This includes specifying the major components of the system, the relationships between them, and the manner in which data will flow from one component to another. The third objective is to implement the core modules of this architecture, such as hospital and user management, resource availability management, request and response handling, and emergency alert broadcasting, using appropriate development tools and technologies.

Finally, the project seeks to evaluate how well the proposed system supports the original goals of reducing communication delays, improving visibility of network-wide capacity, and preventing the exchange of patient-identifiable information between hospitals. By achieving these objectives, the project aims to demonstrate that a dedicated multi-hospital communication platform can make a meaningful contribution to the efficiency and safety of healthcare networks.

CHAPTER 2

LITERATURE SURVEY

2.1 DEVELOPMENT OF MULTI-HOSPITAL OPERATIONAL COMMUNICATION SYSTEM

The development of Multi-Hospital Operational Communication Systems represents a significant advancement in the realm of healthcare network coordination, particularly in the context of inter-hospital facilities and collaborative care environments. This comprehensive research addresses the various challenges encountered in managing multi-hospital networks efficiently by leveraging modern technology and software solutions. The multi-hospital communication platform offers a suite of features including real-time resource availability sharing, equipment request management, ambulance coordination tracking, blood bank inventory management, emergency alert broadcasting, and staff communication channels. Through the development of this integrated communication system, the project endeavours to address the fragmentation present in current inter-hospital coordination practices, enhance information accessibility, improve response times during emergencies, and optimize resource allocation across hospital networks. The insights and findings gained from this comprehensive study are invaluable not only for hospital administrators and network coordinators but also for stakeholders in the broader healthcare industry seeking innovative solutions to optimize their multi-hospital operations and network efficiency.

2.2 SECURE MESSAGING PLATFORMS AND HEALTHCARE OPERATIONAL COMMUNICATION

Garcia, Nelson, and Rodriguez conduct a comprehensive investigation into the use of secure messaging platforms to support day-to-day hospital operations and inter-departmental communication workflows. Their extensive study compares traditional communication channels such as telephone calls, fax transmissions, and email communication with modern encrypted messaging applications that offer advanced features including group conversations, broadcast distribution lists, message threading, attachment capabilities, and comprehensive message archiving and search functionality. The authors observe through their research that the implementation of secure messaging platforms significantly reduces response times for common coordination tasks such as arranging patient transfers between facilities, locating critical medical equipment, coordinating staff availability, and managing resource requests. However, their analysis also identifies critical concerns noting that when generic consumer messaging applications are adopted without clear institutional policies and governance frameworks, there is a pronounced tendency for clinical discussions to become intermingled with non-clinical operational communications. To mitigate these identified concerns and risks, the authors propose an integrated solution that combines dedicated institutional messaging modules with robust hospital authentication systems, implements strict separation protocols between official operational communication channels and informal personal communication areas, and ensures comprehensive logging and audit capabilities for all official communications.

2.3 REAL-TIME ALERT SYSTEMS FOR EMERGENCY HEALTHCARE NETWORKS AND DISASTER

Kim and his colleagues focus their research on real-time alerting mechanisms used in emergency healthcare networks covering multiple hospitals, pre-hospital emergency medical services, and regional disaster response coordination centres. Their research examines systems that enable authorized emergency coordinators and administrators to issue incident-related alerts, capacity warnings, patient diversion notices, and critical resource shortage notifications to a wide set of stakeholders and participating hospitals simultaneously. The comprehensive study demonstrates that alerts are most effective when categorized by priority levels and incident types, enabling recipients to properly prioritize their response. The authors document that recipients should be able to acknowledge or provide short status updates through the same alert interface, creating a feedback loop that improves situational awareness. The research indicates that standardized alert templates, predefined recipient group configurations, and clear escalation rules significantly help reduce confusion and response delays during large-scale emergency events such as mass casualty incidents, pandemics, or infrastructure failures. The authors further emphasize the critical importance of maintaining a concise, focused set of data elements in each alert message, enabling key information to be processed quickly and accurately even under extreme time pressure and high-stress emergency conditions. Their findings establish that when alerts are properly structured with clear prioritization, defined recipient routing, and confirmation mechanisms, they become dramatically more effective at coordinating network-wide emergency response compared to ad-hoc informal communication methods.

2.4 DATA PRIVACY CONSIDERATIONS IN INTER-HOSPITAL INFORMATION SHARING AND REGULATORY

Rahman and Evans undertake a detailed and thorough examination of the privacy and regulatory implications inherent in sharing health-related information between independent hospitals and healthcare institutions. Their comprehensive legal and technical analysis carefully reviews national and international guidance on data protection requirements including HIPAA regulations in the United States, GDPR requirements in the European Union, and similar frameworks established by healthcare authorities in other jurisdictions. The authors identify and document common risk scenarios and compliance pitfalls, including situations such as inadvertently transmitting detailed patient records across organizational boundaries without appropriate consent documentation, storing shared data in unsecured or inadequately protected repositories, implementing insufficient access controls, and failing to maintain adequate audit trails. The authors propose a comprehensive classification framework that distinguishes between patient-identifiable information, pseudonymized information where identities are masked but could potentially be re-linked, and fully anonymized information where any linkage to specific individuals is permanently eliminated. For each category, they recommend appropriate technical safeguards, organizational policies, access control mechanisms, and encryption approaches tailored to the sensitivity level of the information in that category.

2.5 COMPREHENSIVE SUMMARY OF LITERATURE ON MULTI-HOSPITAL COMMUNICATION SYSTEMS

The extensive literature review conducted on multi-hospital communication systems, healthcare network coordination, secure messaging platforms, and privacy-preserving information sharing collectively highlights three major, recurring themes that are directly relevant and applicable to the Multi-Hospital Communication System being proposed and developed in the current project. The first major theme, consistently emphasized across multiple research studies and practical implementations, is the recognition that effective communication is a critical and often rate-limiting factor in hospital and broader healthcare network-level performance. The second major theme identifies a significant gap in the current landscape of healthcare communication tools: existing tools and platforms either concentrate narrowly on single-hospital internal workflows and departmental communication, or alternatively address very specific, narrow emergency coordination scenarios such as disaster response management. This gap leaves an unmet need for a lightweight, flexible, general-purpose communication platform that can be adopted across diverse institutional contexts, multiple types of hospital networks, and varying organizational structures. The third major theme demonstrates that privacy and regulatory concerns can be substantially reduced and simplified when systems are deliberately designed around aggregated operational data instead of patient-identifiable information, thereby lowering barriers to adoption and enabling participation by institutions with heightened privacy concerns or resource constraints.

CHAPTER 3

EXISTING SYSTEM

The existing operational environment across most multi-hospital healthcare networks, inter-hospital coordination regarding resource availability, equipment needs, and emergency response is primarily achieved through entirely manual and semi-structured methods that have remained largely unchanged for several decades. Staff members tasked with inter-hospital coordination typically utilize a combination of telephone calls to known contacts at other institutions, email communication with distribution lists, informal social messaging group chats created spontaneously for network communication, paper-based forms and records, and disconnected spreadsheet documents attempting to track resource requests. When one hospital requires urgent information about bed capacity, ventilator availability, or specialized equipment at another hospital, the standard procedure requires staff members to systematically identify and contact multiple hospitals sequentially, often placing repeated phone calls when initial attempts fail to reach appropriate personnel, resulting in significant delays and consumption of valuable clinical staff time.

Information obtained through these ad-hoc coordination processes is often stored in personal devices, informal notes, or email inboxes rather than in any centralized, institutional repository accessible to other staff members. Furthermore, when generic social messaging applications such as WhatsApp, Telegram, or Facebook Messenger are used informally by hospital staff for network coordination, official institutional communication becomes mixed and mingled with purely personal and informal conversations, leading to significant challenges in ensuring message retention, regulatory compliance, and proper governance.

The addition to the general drawbacks already discussed, the present communication practices suffer from serious limitations in terms of scalability and standardization. As the number of hospitals in a network increases, the volume of phone calls, emails and informal messages grows rapidly. Each new institution must establish its own set of contacts and unofficial procedures, which leads to a patchwork of different methods in use at the same time. Without a central system, there is no guarantee that all hospitals are following the same conventions for reporting availability or raising requests. One hospital may describe its capacity in terms of number of “free beds”, while another reports percentages or broad qualitative terms such as “almost full”. This lack of standardization makes it difficult to compare information across institutions and to make reliable decisions during peak demand.

The absence of analytics and reporting capabilities in the existing setup is another important weakness. Since most inter-hospital communication takes place through transient channels such as phone conversations, administrators have little data to analyse after an incident has passed. Questions like “How long did it take to find a hospital with available capacity?”, “How many requests were denied due to shortage?” or “Which departments are making the most requests?” cannot be accurately answered. As a result, planning for future improvements relies on anecdotal evidence rather than on objective metrics. Without proper records, it is also challenging to demonstrate to regulatory bodies or funding agencies that hospitals used all available resources efficiently. Overall, the existing system does not provide the transparency, consistency or analytical insight required for modern, data-driven healthcare management.

CHAPTER 4

PROBLEM IDENTIFICATION

The rapid growth of hospital networks and referral relationships has created a strong need for effective operational communication between multiple hospitals. In practice, however, most coordination is still handled using traditional methods such as telephone calls, emails and informal messaging groups. When one hospital needs to know which nearby facility has available capacity, which institution can supply a particular piece of equipment, or where a specific blood component is currently in stock, staff members must manually contact several colleagues, wait for replies and consolidate information by hand. This manual process is slow and heavily dependent on the personal contacts and initiative of individual staff, which makes it unreliable in situations where decisions must be taken quickly. Because there is no unified digital platform for inter-hospital operational communication, information is fragmented across different channels and is rarely up to date. Resource availability may change within minutes, but spreadsheets or notes used to track phone calls are not automatically updated. As a result, hospitals may continue to call institutions that are already full, or may fail to use capacity that is actually available elsewhere. The lack of a single, authoritative view of network-wide resources leads to inefficiencies such as delayed transfers, duplicated efforts and under-utilization of critical equipment. In emergency situations with high patient inflow, these inefficiencies can directly impact the quality and timeliness of care.

Furthermore, the current use of generic messaging tools introduces potential privacy and security concerns. Although the goal of inter-hospital coordination is often purely operational, staff under pressure may inadvertently share patient names or other identifying details while discussing cases in informal groups. Such disclosures may conflict with data protection regulations and hospital policies, exposing institutions to legal and ethical risks. At the same time, hospitals are hesitant to adopt heavy, patient-centric information systems solely for operational communication because of cost, complexity and integration overhead. This creates a gap where no lightweight, privacy-aware solution is available to support everyday coordination activities.

In summary, the key problem identified in this project is the lack of a dedicated, structured and secure platform for multi-hospital operational communication that does not rely on patient-level data. Existing methods are fragmented, slow, difficult to audit and prone to privacy issues, while current enterprise-scale systems are often too complex or narrowly focused on single institutions. Addressing this problem requires a solution that can centralize operational information, support real-time updates, enforce role-based access and generate an auditable history of coordination activities, all while remaining simple enough to be adopted by hospitals with different levels of technical infrastructure.

CHAPTER 5

PROPOSED SYSTEM

The proposed Multi-Hospital Communication System introduces a fundamentally different approach to inter-hospital operational coordination by creating a centralized, institutionally controlled platform that is specifically and deliberately designed for operational communication between hospitals within a coordinated network. Rather than relying on informal phone calls, email chains, and uncontrolled messaging applications, the proposed system provides a structured, secure digital platform where authorized representatives from each participating hospital can publish real-time operational status information, submit and track requests for resources and equipment, monitor and respond to requests from other hospitals, and receive and acknowledge emergency alerts from network coordinators.

Each hospital participating in the network registers through an administratively controlled enrolment process and is assigned a unique institutional identifier. Authorized users from each hospital are created with specific roles such as hospital administrator, hospital network coordinator, departmental unit head, or departmental staff member. After completing secure authentication with unique credentials, these authorized users interact with carefully designed, intuitive user interface screens and forms to execute key operational tasks. For example, a hospital resource coordinator can navigate to the resource update module and enter current availability numbers for critical resources such as intensive care unit capacity, ventilators, cardiac monitoring equipment, and other specialized equipment. These updates are immediately and automatically saved to the centralized database and become visible in real-time to authorized users at other hospitals through the system's dashboard and resource board views.

The proposed Multi-Hospital Communication System is designed not only as a technical solution but also as a framework for standardizing coordination procedures across hospitals. By defining explicit data fields for each module-for example, specifying how resource types are classified, how urgency levels are coded and how alert priorities are expressed-the system encourages all participating institutions to adopt a common vocabulary. When every hospital reports its operational status using these shared definitions, comparisons become more meaningful and decisions can be taken on a consistent basis. The platform therefore acts as a reference point for network-wide policies, and its data structures implicitly capture the standard operating procedures agreed upon by the hospitals.

Another important feature of the proposed system is its extensibility. The architecture is intentionally modular so that additional functionality can be introduced without redesigning the core. For instance, new modules for ambulance tracking, bed-level occupancy statistics or integration with laboratory systems can be added later by exposing new APIs and creating corresponding user-interface components. Because the communication platform is built using web technologies and a service-oriented design, these extensions can be developed incrementally while keeping the existing features operational. This flexibility is crucial in a healthcare environment where requirements evolve over time, regulations change and new kinds of collaborations emerge between institutions. By providing a stable base with clear extension points, the proposed system can grow along with the needs of the hospital network rather than becoming obsolete after a few years of use.

CHAPTER 6

SYSTEM REQUIREMENTS

6.1 HARDWARE REQUIREMENTS

- Operating System : Windows 10 / 11
- Processor : Intel Core i5 or higher
- RAM : 8 GB (minimum), 16 GB (recommended)
- Hard Disk : 500 GB or higher
- Monitor : 15" or higher, Full HD
- Input Devices : Standard Keyboard and Optical Mouse

The above hardware configuration is sufficient to develop and run the Multi-Hospital Communication System using a modern JavaScript stack. A multi-core processor and adequate memory ensure that the Node.js development server, React build tools and MongoDB database can operate concurrently without performance degradation. A full HD monitor improves readability of the development environment, browser tools and database consoles, which is important when working with complex user interfaces and long log files.

6.2 SOFTWARE REQUIREMENTS

- Operating System : Windows 10 / 11 (64-bit)
- Frontend Technologies : React JS, HTML5, CSS3
- Backend Runtime : Node.js with npm
- Database : MongoDB Community Edition
- IDE / Code Editor : Visual Studio Code
- Browser : Google Chrome / Mozilla Firefox (latest)
- Version Control : Git

These software components together provide a complete environment for full-stack web application development. React JS, combined with HTML5 and CSS3, is used for building the dynamic and responsive user interface of the communication system. Node.js serves as the backend runtime for handling API requests, processing business logic and interacting with MongoDB, which stores operational data such as hospital details, resource status, requests and alerts. Visual Studio Code offers a rich ecosystem of extensions for JavaScript, React, and MongoDB, simplifying development, debugging and source-control operations.

6.3 HARDWARE DESCRIPTION

The system is developed and tested on a personal computer configured with an Intel Core i5 processor, 8 GB RAM and a 500 GB hard disk running a 64-bit Windows operating system. This machine hosts both the Node.js server and the MongoDB database instance during development. The processor provides sufficient computational power for compiling React components, executing JavaScript logic and managing HTTP requests in parallel. The memory capacity ensures that the browser, development tools, server processes and database engine can run simultaneously without frequent swapping or slowdowns.

Storage space is utilized for installing the operating system, development tools, Node.js packages and MongoDB data files. Although the prototype does not require very large data volumes, the reserved capacity allows for growth as more test records and log files are generated. The machine is connected to a local network, enabling multiple client browsers within the same lab or environment to access the application during demonstration and testing sessions.

6.4 SOFTWARE DESCRIPTION

6.4.1 React JS

React JS is used as the primary JavaScript library for building the user interface of the Multi-Hospital Communication System. React's component-based architecture allows the interface to be decomposed into reusable elements such as login forms, dashboards, resource cards, request tables and alert panels. Each component manages its own state and can be composed with others to form larger views, which simplifies maintenance and enhances consistency across pages. React's virtual DOM mechanism ensures

that only the parts of the page that actually change are re-rendered, providing a smooth and responsive user experience even when multiple updates occur in quick succession.

Using React also facilitates the creation of a single-page application where navigation between major sections-such as resources, requests and alerts-occurs without full page reloads. This is particularly useful in a communication system that is expected to be active for long periods on staff workstations. Additional React libraries, such as React Router for navigation and Axios or Fetch API for HTTP communication, integrate seamlessly and help structure client-side code in a clean and modular way.

6.4.2 Html and CSS

HTML5 is employed to define the structural layout of each page rendered by the React components. Semantic HTML tags are used to represent meaningful sections of the interface, such as headers, navigation bars, main content areas and tables. This improves accessibility and makes the markup easier to understand and maintain. All textual content, form fields, buttons and tables are defined using standard HTML elements, which are then enhanced by React for dynamic behaviour.

CSS3 is used to control the visual presentation of the application, including colors, fonts, spacing, alignment and responsive behavior. Styles are organized using either traditional stylesheets or modern approaches such as CSS modules or styled components, depending on project conventions. Consistent styling is applied across all components so that screens for hospital registration, resource dashboards and alert overviews share a common look and feel. Media queries and flexible layouts help ensure that the interface remains usable on different screen sizes, such as desktop monitors and laptops commonly used in hospitals.

6.4.3 MongoDB

MongoDB is chosen as the database for this project because of its document-oriented model and flexibility in handling evolving data structures. Collections are created for hospitals, user accounts, roles, resource status entries, requests, responses and alerts. Each document stores its attributes in JSON-like format, which aligns naturally with the JavaScript objects used by the Node.js backend and React frontend. This reduces the impedance mismatch between application code and database schema and speeds up development.

The non-relational nature of MongoDB allows the system to store complex, nested structures such as lists of status updates or alert recipients within a single document when appropriate. Indexes are configured on frequently queried fields, for example hospital identifiers and alert timestamps, to maintain acceptable query performance. For the purposes of this project, MongoDB runs as a standalone instance on the local machine, but the same configuration can later be extended to a replica set or cloud-hosted deployment for production usage.

6.4.4 Visual Studio Code

Visual Studio Code (VS Code) is used as the main integrated development environment for implementing the Multi-Hospital Communication System. It provides built-in support for JavaScript, TypeScript and JSX, which are essential for React development. Extensions for ESLint and Prettier are used to enforce consistent coding style and automatically format code, while the React and MongoDB extensions assist with component navigation and database exploration. The integrated terminal allows Node.js servers, React build scripts and Git commands to be executed without leaving the editor.

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 LIST OF MODULES

- User Authentication Module
- Resources Status Management Module
- Request Handling Module
- Alert Broadcasting Module
- Admin Monitoring and Reporting Module

7.2 MODULE DESCRIPTION

This module serves as an integral component of the system, responsible for managing all activities related to its designated functional area. It handles user interactions, processes incoming data, applies necessary validations, and executes the core operations required to fulfill its purpose. The module also ensures secure and efficient communication with other system modules to maintain a smooth and coordinated workflow. By continuously monitoring data consistency, system states, and operational rules, it supports accurate decision-making and timely updates for both users and administrators. Additionally, the module contributes to overall system performance by managing resource usage effectively, ensuring reliability during high-load situations, and maintaining the integrity of stored and processed information. Its comprehensive functionality helps the system operate cohesively, ensuring a robust, user-friendly, and scalable environment.

7.2.1 User Authentication Module

Step 1: A “users” collection is created in MongoDB with fields for username, password hash, role and associated hospital. Indexes are defined on the username field to ensure fast look-ups.

Step 2: On the backend, Express routes are implemented for user registration and login. Passwords are hashed using a secure hashing algorithm before storage, and the login route verifies the entered password against the stored hash. Upon successful login, a signed token or server-side session is generated to identify the user during subsequent requests.

Step 3: On the frontend, React components are developed for the login page and basic navigation. The login form captures credentials and sends them to the backend using Axios. Depending on the response, the application either displays an error message or redirects the user to the main dashboard while storing the authentication token in memory or secure storage.

Step 4: A higher-order component or route guard is implemented to protect private routes. This component checks that a valid token or session exists before allowing access to pages such as resource management or request handling. This ensures that only authenticated users can interact with sensitive operations in the system.

7.2.2 Resources Status Management Module

Step 1: A “resources” collection is created in MongoDB to store operational status per hospital and resource type. Each document includes the hospital identifier, resource category, available quantity and a timestamp for the last update.

Step 2: Backend API endpoints are developed to allow authorized users to create, update and retrieve resource status records. These endpoints validate that the user belongs to a registered hospital and that the data conforms to defined constraints, such as non-negative quantities.

Step 3: React components are implemented for the resource dashboard and update forms. Coordinators can view a table of current resources for their hospital, edit existing values and submit updates. A network-wide view is also provided, in which coordinators can see summarized availability across all hospitals. Axios is used to fetch data from the backend and to push changes back to the server.

Step 4: After implementation, unit tests and manual checks are performed to confirm that updates made by one hospital appear correctly on the dashboards of other hospitals, and that unauthorized users cannot modify resource data.

7.2.3 Request Handling Module

Step 1: The “requests” collection is designed to hold equipment and blood requests. Each record contains the requesting hospital, type of resource, required quantity, urgency level, current status and a list of responses from other hospitals.

Step 2: Backend routes are defined for creating new requests, listing open requests and posting responses. Business logic ensures that only the originating hospital can close or cancel a request, while other hospitals can only add responses indicating whether they can help.

Step 3: On the frontend, a React form is constructed for coordinators to raise new requests. Dropdown lists are used for selecting resource types and urgency levels to maintain consistency. A separate view lists all open requests, with filters for resource type and status.

Step 4: A response interface is implemented, allowing users from other hospitals to open a request, enter the quantity they can provide and submit their response. The UI visually distinguishes between fulfilled, partially fulfilled and unfulfilled requests. All interactions are synchronized with the backend through API calls and confirmed through success or error messages.

7.2.4 Alert Broadcasting Module

Step 1: An “alerts” collection is created in MongoDB to store emergency alerts with fields such as title, message, priority, issuing hospital, target audience and timestamps.

Step 2: Backend logic is added to accept new alerts from authorized users. The server validates the requester’s role (for example, coordinator or admin) and records the alert in the database. It then prepares a list of intended

recipients based on the chosen target scope (individual hospitals or all hospitals).

Step 3: On the frontend, a React component is built for creating alerts. Users specify the alert heading, description, priority and target hospitals via form elements. After submission, the page confirms that the alert has been recorded successfully.

Step 4: A notification panel is implemented on the dashboard to display active alerts to all logged-in users. New alerts are fetched via API calls and displayed in order of priority and time. Simple visual cues such as color coding are used to differentiate high-priority alerts from routine messages.

7.2.5 Admin Monitoring and Reporting Module

Step 1: Aggregation queries are implemented on the backend to compute statistics such as the number of requests raised by each hospital, the average response time for fulfilled requests and counts of alerts by priority.

Step 2: React components are created for administrative dashboards that present this information using tables and simple charts. These dashboards are accessible only to users with the administrator role, enforced by the route guard.

Step 3: Export functionality is added to allow selected summaries to be downloaded in CSV format for offline analysis or inclusion in periodic reports. This is implemented by generating structured data on the server and returning it as downloadable files through dedicated routes.

CHAPTER 8

SYSTEM TESTING

8.1 TESTING STEPS

System testing for the Multi-Hospital Communication System is carried out in a structured sequence of steps to ensure that each component behaves as intended and that the integrated system supports key workflows reliably. The first step is unit testing, where individual functions and methods within each module are tested in isolation using representative inputs and expected outputs. For example, the authentication routine is tested with valid and invalid credentials to check that it grants access correctly and rejects unauthorized attempts. Similarly, database operations for inserting and retrieving resource records are exercised with sample data to confirm that constraints and relationships are enforced correctly.

Once unit testing reaches a satisfactory state, the second step involves integration testing, during which modules that have already been individually validated are combined and tested as composite units. Common integration scenarios include logging in and immediately updating resource availability, creating a request and then viewing it from another hospital's account, and issuing an alert that should appear on multiple dashboards. Any discrepancies or unexpected behaviours observed during integration testing are analysed and addressed by refining module interfaces or adjusting database design. The final step is system testing, where the fully integrated system is exercised through realistic user scenarios that mirror actual usage in a hospital network. This includes test runs for daily status updates, batch requests during simulated high-load periods, and emergency alert drills. Results from these tests are documented, and issues are rectified before the system is considered ready for demonstration.

8.2 TYPES OF TESTING

Several types of testing are applied within the scope of this project to evaluate different aspects of the Multi-Hospital Communication System. Functional testing verifies that the implemented features match the specifications derived from the requirements analysis. Test cases are designed to cover all major functions, such as managing hospitals and users, viewing and editing resource status, creating and responding to requests, and sending alerts. Each test case specifies input conditions, expected results and actual outcomes, making it possible to systematically trace and correct deviations.

In addition to functional checks, usability testing is performed informally to assess how easily typical users can navigate the system and complete tasks. Feedback gathered during this process may lead to adjustments in the layout of forms, labelling of buttons or arrangement of dashboard elements. Basic performance testing is carried out by simulating multiple concurrent users, verifying that the system remains responsive under moderate load. While exhaustive security testing is beyond the current scope, simple checks are conducted to ensure that users cannot bypass authentication, access unauthorized screens or manipulate URLs to retrieve restricted information. Together, these testing activities provide reasonable confidence that the system is robust enough for demonstration purposes and that it can be further enhanced into a production-ready solution with additional effort. In the development of the Multi-Hospital Communication System, several types of testing were carried out to ensure that the application functions correctly and reliably under typical usage conditions. Each type of testing focuses on a particular aspect of software quality and together they provide reasonable confidence that the system can support day-to-day operational communication between hospitals.

8.2.1 Unit Testing

Unit testing concentrates on verifying the correctness of individual program units such as functions, classes and small React components. For the frontend, units include form components for login, resource update and request creation, while on the backend they include API handlers and utility functions used for validation and database access. During unit testing, each unit is supplied with controlled input data and the actual output is compared with the expected result. Any discrepancies are investigated and corrected before integration with other modules. This approach helps detect logical errors at an early stage, when they are easier and less expensive to fix.

8.2.2 Integration Testing

Integration testing is performed after the individual units have been tested and considered stable. The objective is to ensure that different modules of the system work correctly when combined. In this project, examples include testing the interaction between the React frontend and the Node.js backend APIs, as well as verifying that the backend correctly reads from and writes to the MongoDB database. Typical integration scenarios cover logging in and then immediately updating resource availability, creating a request and confirming that it appears in the dashboards of other hospitals, and sending an alert and checking that it is stored and retrieved properly. Integration testing helps reveal interface mismatches, data format inconsistencies and other issues that only become visible when modules interact.

8.2.3 System Testing

System testing evaluates the complete, integrated Multi-Hospital Communication System as a single entity. The aim is to verify that all functional and non-functional requirements identified in the analysis phase are satisfied. Testers execute end-to-end scenarios that closely resemble real-world usage, such as coordinating resource sharing between multiple hospitals during a simulated high-demand situation. System testing checks not only functional correctness but also aspects like navigation flow, error handling, response times for typical operations and overall user experience. Any defects discovered at this stage are logged, prioritized and resolved to ensure that the deployed system behaves as expected from the user's perspective.

8.2.4 User Acceptance Testing

User Acceptance Testing is carried out with the involvement of representative end-users, such as faculty members or students acting as hospital coordinators and department staff. They are given access to the system with predefined scenarios and are asked to perform routine tasks including updating resources, raising requests and acknowledging alerts. Feedback is collected regarding ease of use, clarity of screens, suitability of workflows and any difficulties encountered. The purpose of UAT is to confirm that the system not only meets technical specifications but is also acceptable and practical for the people who will operate it. Adjustments recommended during this phase, such as renaming labels or simplifying forms, are incorporated to improve usability before final submission.

CHAPTER 9

RESULT AND DISCUSSION

The result and discussion of this project focus on evaluating how effectively the Multi-Hospital Communication System supports real-time operational communication between hospitals while excluding patient-identifiable data. After implementing the core modules using React JS, Node.js and MongoDB, the system was subjected to functional tests, scenario-based trials and informal user evaluations. In all test runs, authorized users were able to log in, update resource status, create and respond to equipment or blood requests and issue emergency alerts to other hospitals through a unified web interface. The information entered through the interface was correctly stored in the database and retrieved on demand without critical errors, which shows that the architecture and module interactions are stable for the intended prototype scale. Response times for these operations were consistently acceptable on the development server, and users could observe updated values and new alerts on their dashboards shortly after they were submitted by other participants, indicating that the system supports timely decision making in typical coordination scenarios.

The observed behaviour also demonstrates that the design successfully enforces the project's privacy constraint. All communication handled by the system is limited to aggregated operational data such as counts of resources, descriptions of equipment, blood group quantities and brief alert messages. No names, identifiers or clinical details of individual patients are stored or transmitted. Test users were still able to coordinate effectively using this aggregated information, confirming that inter-hospital collaboration can be improved without relying on sensitive patient data. At the same time, the evaluation revealed some limitations that must be considered.

CHAPTER 10

CONCLUSION AND FUTURE SCOPE

The Multi-Hospital Communication System developed in this project addresses a clear and significant gap in the current landscape of healthcare information systems. While many existing platforms focus primarily on internal hospital workflows and patient record management, the need for a dedicated, privacy-conscious solution for inter-hospital operational coordination has often been overlooked. By designing and implementing a web-based platform that concentrates exclusively on aggregated operational data such as resource availability, equipment needs and emergency alerts, the project demonstrates that it is possible to improve network-level coordination without exposing patient-identifiable information. Through the analysis, design, implementation and testing activities documented in earlier chapters, the system has shown that it can support key tasks such as updating status boards, raising structured requests and disseminating alerts in a more organized and auditable manner than the informal methods currently in use.

At the same time, the project acknowledges certain limitations inherent in the prototype stage of development. The system has been tested with a limited dataset and a modest number of simultaneous users; comprehensive evaluation under real-world load conditions and across diverse hospital environments remains to be undertaken. The current implementation also does not include advanced features such as automated integration with existing Hospital Management Systems, sophisticated performance optimization, large-scale disaster recovery planning or comprehensive security hardening. These aspects are essential for production deployment and would require further effort, including involvement from hospital IT departments and external security experts.

Looking ahead, there are several promising directions for future work. One important enhancement would be to integrate the Multi-Hospital Communication System with existing hospital databases and bed-management systems so that certain operational indicators can be updated automatically rather than manually. Another extension would be the development of mobile applications or progressive web interfaces, enabling coordinators and clinicians to access the system seamlessly from smartphones and tablets while on the move. Advanced analytics modules could be added to analyse historical data on requests, responses and alerts to identify trends, predict resource shortages and support strategic planning at the network level. With such enhancements, the Multi-Hospital Communication System could evolve from a prototype into a robust, production-ready platform that plays a central role in improving the resilience, efficiency and responsiveness of healthcare networks.

APPENDIX A

SOURCE CODE

Index.html

```
<!doctype html>
<html lang="en">
 <!-- client/index.html -->
<head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>MHCS</title>

    <!-- Add Google Fonts -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
                                                                    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;
600;700&display=swap" rel="stylesheet">
</head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
```

```
  </body>
</html>
```

Eslint.js

```
import js from '@eslint/js'
import globals from 'globals'
import reactHooks from 'eslint-plugin-react-hooks'
import reactRefresh from 'eslint-plugin-react-refresh'
import { defineConfig, globalIgnores } from 'eslint/config'

export default defineConfig([
  globalIgnores(['dist']),
  {
    files: ['**/*.{js,jsx}'],
    extends: [
      js.configs.recommended,
      reactHooks.configs['recommended-latest'],
      reactRefresh.configs.vite,
    ],
    languageOptions: {
      ecmaVersion: 2020,
      globals: globals.browser,
      parserOptions: {
```

```
      ecmaVersion: 'latest',
      ecmaFeatures: { jsx: true },
      sourceType: 'module',
    },
  },
  rules: {
    'no-unused-vars': ['error', { varsIgnorePattern: '^[A-Z_]' }],
  },
},
])
```

Server.js

```
// server/server.js (Final, Consolidated, and Correct Version)

// 1. IMPORTS
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const bcrypt = require('bcryptjs');
const session = require('express-session');
const MongoStore = require('connect-mongo');
```

```
// 2. INITIALIZATION
const app = express();
const PORT = process.env.PORT || 5000;

// 3. MIDDLEWARE
app.use(cors({
    origin: 'http://localhost:5173',
    credentials: true
}));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// 4. DATABASE CONNECTION
mongoose.connect(process.env.MONGO_URI)
    .then(() => console.log('MongoDB connected successfully.'))
    .catch(err => console.error('MongoDB connection error:', err));

// 5. SESSION MANAGEMENT
app.use(session({
    secret: process.env.SESSION_SECRET,
    resave: false,
```

```
    saveUninitialized: false,
    store: MongoStore.create({
        mongoUrl: process.env.MONGO_URI,
        collectionName: 'sessions'
    }),
    cookie: {
        maxAge: 1000 * 60 * 60 * 24 // 1 day
    }
}));

// 6. MONGOOSE SCHEMAS & MODELS
const AdminSchema = new mongoose.Schema({
    username: { type: String, required: true, unique: true, default: 'admin' },
    password: { type: String, required: true }
});
const Admin = mongoose.model('Admin', AdminSchema, 'admins');

const HospitalSchema = new mongoose.Schema({
    hospitalName: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    password: { type: String, required: true },
    address: { type: String },
```

```
  phone: { type: String },
  details: {
    oxygenCylinders: { total: { type: Number, default: 0 } },
    bloodAvailability: {
      'A+': { type: Number, default: 0 }, 'A-': { type: Number, default: 0 },
      'B+': { type: Number, default: 0 }, 'B-': { type: Number, default: 0 },
      'AB+': { type: Number, default: 0 }, 'AB-': { type: Number, default: 0 },
      'O+': { type: Number, default: 0 }, 'O-': { type: Number, default: 0 },
    },
    organAvailability: [{
        organName: { type: String, required: true, enum: ['Kidney', 'Liver', 'Heart', 'Lung', 'Pancreas', 'Cornea'] },
      bloodGroup: { type: String, required: true, enum: ['A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'] },
      age: { type: Number, required: true },
      notes: { type: String }
    }]
  }
}, { timestamps: true });
const Hospital = mongoose.model('Hospital', HospitalSchema, 'hospitals');

const RequestSchema = new mongoose.Schema({
```

```
    requestingHospital: { type: mongoose.Schema.Types.ObjectId, ref: 'Hospital', required: true },

  providingHospital: { type: mongoose.Schema.Types.ObjectId, ref: 'Hospital' },

  requestType: { type: String, enum: ['Oxygen', 'Blood', 'Organ'], required: true },

  status: { type: String, enum: ['Open', 'Accepted', 'Closed', 'Rejected'], default: 'Open' },

  details: {

    bloodGroup: { type: String, enum: ['A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'] },

    quantity: { type: Number },

      organName: { type: String, enum: ['Kidney', 'Liver', 'Heart', 'Lung', 'Pancreas', 'Cornea'] },

  },

  description: { type: String, required: true },

}, { timestamps: true });

const Request = mongoose.model('Request', RequestSchema, 'requests');



const KnowledgeArticleSchema = new mongoose.Schema({

  postingHospital: { type: mongoose.Schema.Types.ObjectId, ref: 'Hospital', required: true },

  title: { type: String, required: true },

  category: { type: String, required: true, enum: ['Clinical', 'Administrative', 'Operational', 'Other'] },

  content: { type: String, required: true },
```

```
}, { timestamps: true });

const KnowledgeArticle = mongoose.model('KnowledgeArticle', KnowledgeArticleSchema, 'knowledge_articles');


// 7. API ROUTES

const isAuthenticated = (req, res, next) => {

   if (req.session.userId) return next();

   res.status(401).json({ message: 'Unauthorized' });

};


// --- AUTHENTICATION ROUTES ---

app.post('/api/hospital/register', async (req, res) => {

   try {

      const { hospitalName, email, password } = req.body;

      const existingHospital = await Hospital.findOne({ email });

      if (existingHospital) {

         return res.status(400).json({ message: 'Hospital with this email already exists.' });

      }

      const hashedPassword = await bcrypt.hash(password, 12);

        const newHospital = new Hospital({ hospitalName, email, password: hashedPassword });

      await newHospital.save();
```

```
    res.status(201).json({ message: 'Hospital registered successfully.' });
} catch (error) {
    console.error('Registration Error:', error);
    res.status(500).json({ message: 'Server error', error: error.message });
}
```

APPENDIX-B

SCREENSHOTS

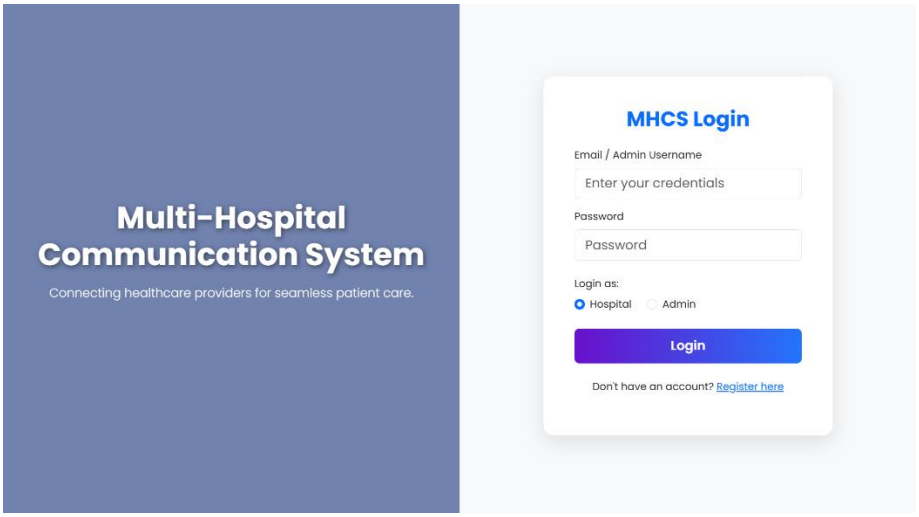

Fig B.1: Login Page

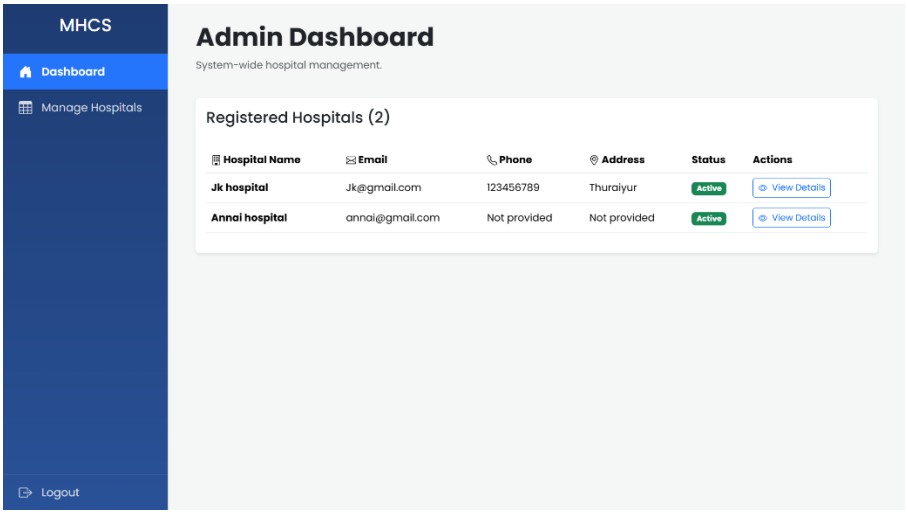

Fig B.2: Admin Dashboard

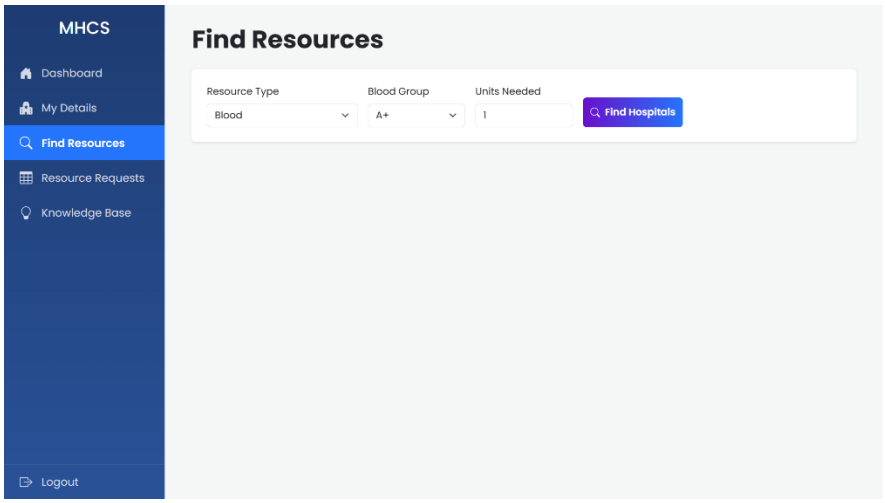

Fig B.3: Resource Finder

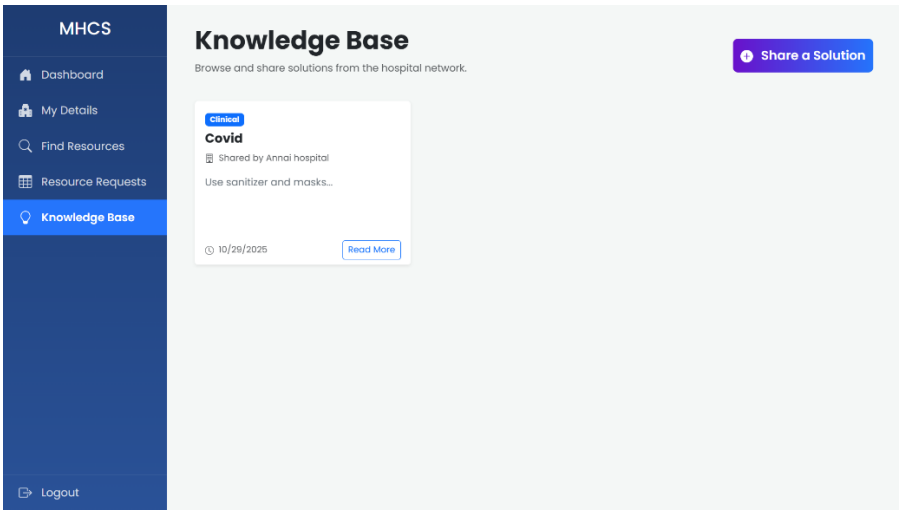

Fig B.4: Knowledge Base

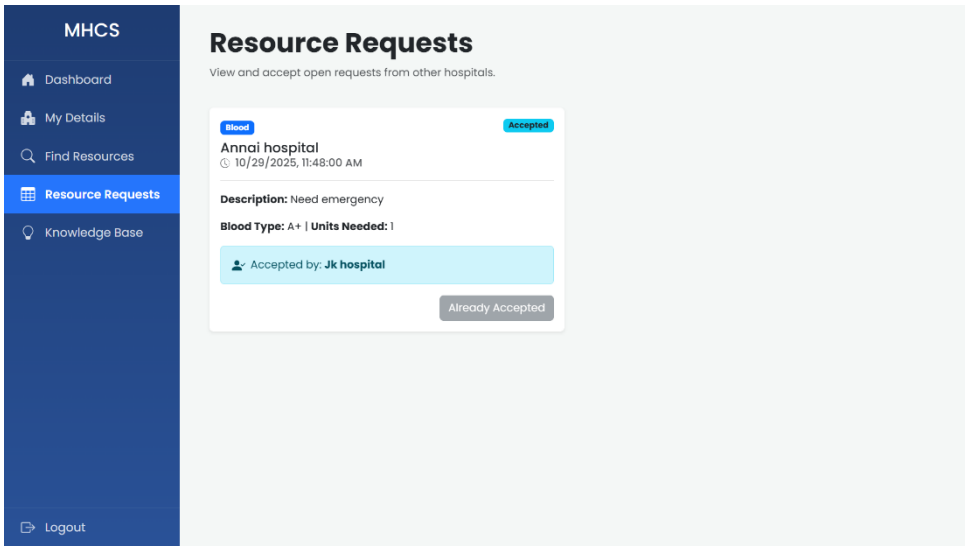

Fig B.5: Resource Request Page

REFERENCES

1. Cam, H.; Wenzlaff, B.; Gillespie, U. et al. “The complexities of communication at hospital discharge of older patients: a qualitative study of healthcare professionals’ views.” *BMC Health Services Research* 23:1211 (2023).

2. Huang, C-j.; Wang, L.; Han, X. “Vertical Federated Knowledge Transfer via Representation Distillation for Healthcare Collaboration Networks.” (2023).

3. Kester, Q.-A. “Using SOA with Web Services for effective Integration of Hospital Information Systems via an Enterprise Service Bus.” (2013).

4. Moreira, A.; Veiga, A. M. “An Overview of Omnichannel Interaction in Health Care Services.” *Health Policy and Technology* 12(2):100806 (2023).

5. Popovici, I.; Charles, C.; Dainty, K.; Sibbald, S. et al. “Technological Aspects of Hospital Communication Challenges: An Observational Study.” *International Journal for Quality in Health Care* 27(3):183–194 (2015).

6. Scioli, G.; Schäfer, W. L. A.; Boerma, W. G. W.; Spree Wenberg, P. M. M. et al. “Communication between general practitioners and medical specialists in the referral process: a cross-sectional survey in 34 countries.” *BMC Family Practice* 21:54 (2020).

7. Sheehan, J.; Laver, K.; Bhoti, A.; Rahja, M.; Usherwood, T.; Clemson, L.; Lannin, N. A. “Methods and Effectiveness of Communication Between Hospital Allied Health and Primary Care Practitioners: A Systematic Narrative Review.” *Journal of Multidisciplinary Healthcare* 14:493-511 (2021).

8. Study: “Technological aspects of hospital communication challenges: an observational study.” *International Journal for Quality in Health Care* 27(3):183-194 (2015).

9. Tushe, S.; Vanberkel, P.; Armony, M. “Impact of Asynchronous Telemedicine Adoption on Patient Flow in a Multichannel Healthcare System.” *Manufacturing & Service Operations Management* (2025).

10. Vermeir, P.; Vandijck, D.; Degroote, S.; Peleman, R. et al. “Communication in Healthcare: A Narrative Review of the Literature and Practical Recommendations.” *International Journal of Clinical Practice* 69(11):1257–1267 (2015).