IJRAR.ORG E-ISSN: 2348-1269, P-ISSN: 2349-5138

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR) | IJRAR.ORG

An International Open Access, Peer-reviewed, Refereed Journal

HOSPITAL MANAGEMENT SYSTEM: A WEB-BASED APPLICATION FOR DIGITISING CLINICAL OPERATIONS

[1]Anjali Kumari, [2]Doly Kumari, [3]Ankita Kumari, [4]Amit Kumar, [5]Abhishek Mankar

[123]Student, Department of Electronics Engineering, Government Polytechnic Katihar, Bihar, India

[45]Lecturer, Department of Electronics Engineering, Government Polytechnic Katihar, Bihar, India

Abstract: *In the healthcare sector, it is required to have a more automated approach in order to streamline the workflows, Digital Technologies are in high demand here. In this paper, the concept of designing and implementing a web-based Hospital Management System (HMS) using HTML5, CSS3, and JavaScript with a browser-based Local Storage used as the client-side data persistence layer is introduced. The proposed system comprises of six operational modules: Patient Management, Doctor Management, Appointment Scheduling, Feedback Collection, Emergency Response and Pharmacy Management. System analysis shows that the performance of the data retrieval system, the accuracy of the data recorded and the coordination of the different departments is significantly improved when compared to the performance of the paper-based system. The architecture is a three-layered one: the presentation layer, the behaviour layer and the data persistence layer, and it requires no external servers in order to be implemented only a standard Web browser is required. The experimental results show the system can reduce the workload of manual scheduling, avoid scheduling conflicts, and provide a basis for further system development such as connecting with a real-time database system, mobile application deployment, and role-based access control.*

Index Terms— *Hospital Management System, Web Application, Patient Records, Appointment Scheduling, Local Storage, Electronic Health Records, Healthcare Automation.*

I. INTRODUCTION

Administrative information is generated and managed at hospitals and clinics daily, including patients' information, clinical histories, doctor schedules, diagnostic reports, billing statements, and medicine inventories. In traditional hospital settings, these data flows are stored in paper records and retrieved through manual, clerical processes, leading not only to substantial delays in data retrieval but to high levels of data transcription error and to limited access to the data at the same time by various departments.

A Hospital Management System (HMS) is a computer-based system that aims to centralise and streamline the management and execution of hospital processes in a single digital platform. An HMS can be a collection of structured software modules rather than manual processes, which can enhance the quality and timeliness of patient care delivery, reduce data duplication and workload.

It is worth noting that the system described in this paper is implemented entirely in the client side of the web technology, using only HTML5, CSS3 and JavaScript. Note that no server nor any external database engine is used. The Web Storage API (Local Storage) is used to store patient records, appointments, and feedback entries as structured JSON objects, which ensures that the data is stored on the client side and remains available during the user's session. This architectural decision makes the system deployable without network infrastructure requirements in resource-constrained environments and makes it an ideal prototype for small clinics and rural health centres.

In this paper, the rest of the paper is organised as follows: Section II reviews the limitations of the existing manual systems. System Objectives are defined in Section III. In Section IV, the author provides an explanation of system analysis and the benefits of digital transformation that are proposed. The system architecture and design are discussed in Section V. A list of the technology used is provided in Section VI. The individual functional modules are summarised in Section VII. The implementation methodology is discussed in Section VIII. The comparative advantages and disadvantages of the proposed system are given in Section IX. Section X outlines the future scope, and Section XI brings the paper to a close.

II. LITERATURE REVIEW

Before the advent of computerised Health Information Systems (HIS), hospitals were operating on paper filing systems that were structured hierarchically. Records were kept in hard copy patient files identified by the patient number and had to be manually transferred to other departments for cross-referencing, which was time consuming and prone to misfiling and duplication [1].

In the traditional environment, appointment management was done on handwritten appointment books or on registers managed by the receptionist. This often led to double-booking, no appointments and poor communication between doctors and patients, which in turn meant sub-optimal use of clinical resources [2].

Billing and payment reconciliation were done on a manual basis leading to a risk of arithmetic errors and causing delays in the reporting of finances. The lack of a centralised data repository also hindered the ability to generate aggregate statistics needed for hospital performance analysis and for regulatory purposes [3].

All of the above disadvantages of the manual system contributed to the need for computerised management solutions. Early hospital information systems tried to solve these deficiencies by adding database-driven patient registration and appointment system modules [4].

Modern HMS systems can now integrate with electronic health records (EHR), support telemedicine, manage pharmaceutical stocks, keep up track of real-time bed availability, and process insurance claims. The implementation of these wide-ranging systems, however, requires strong backend infrastructure, skilled IT staff and continuous financial investment, often making them unavailable to smaller healthcare institutions [5].

The system described in this paper falls into the middle ground – it provides the fundamental capabilities needed to have a full-featured HMS, but it is implemented in a lightweight architecture that runs entirely within the browser window, so there is no need to deploy anything on the server side, which fills a gap in the literature identified.

III. OBJECTIVES OF THE SYSTEM

The Hospital Management System aims to achieve the following objectives in the operations of the hospital:

1. Patient Record Management: To create a structured system of recording, storing and accessing patient demographic information, medical histories and treatment records in a digitised format.

2. Appointment Scheduling: To facilitate the booking, modification and cancellation of appointments for patients with specific doctors of different specialities, and an automatic conflict checker to avoid scheduling conflicts.

3. Data Accuracy and Integrity: To ensure that the data entered by the client is accurate and that the integrity of the records stored in the application remains unchanged throughout the application's life cycle, implement client-side input validation routines to limit the entry of incorrect data.

4. Authorised Data Access: To limit access to records and data modification to authorised users, so that patient-sensitive data can be accessed only by authorised users.

5. Operational Efficiency: To minimise the administrative workload on the healthcare workers by performing repetitive clerical work in an automated way and freeing up their time for patient care.

6. Process Digitalisation: Replace paper-based processes with a computerised process with concurrent access to data, structured reporting and historical audits.

.IV. SYSTEM ANALYSIS

The functional properties and performance characteristics of the proposed solution (digital solution), which are different from the manual solution, are identified in the system analysis below:

Digital Record Management: All patient, physician and operational data are stored in structured JSON format in Local Storage and retrieved without access to physical files.

2. Accelerated Information Processing: Retrieving records takes milliseconds instead of minutes or hours as it is done by key-value look-up in Local Storage, as opposed to a physical file.

3**. Automated Error Reduction:** JavaScript form validation, such as data type constraints, mandatory fields and format consistency, occurs before records are submitted, significantly reducing the number of corrupt data entries.

4. **Controlled Data Access**: Role-based navigation ensures that only authenticated users can view modules and cannot change clinical and administrative data.

5**. Centralised Interdepartmental Coordination**: A data layer that all modules access, which means that updates in one department (e.g. changes to the status of an appointment) are instantly available in all dependent views.

6. **Effective Appointment Management**: The scheduling component offers a well-organised log of all scheduled appointments and transitions the status from Pending to Done, while also giving real-time availability of physicians.

7. Automated Bill Generation: Consultation charges associated with each physician (e.g., ₹100 per consultation) are automatically computed and presented to the patient at the point of appointment completion.

8. Dynamic Reporting: Feedback records, appointment logs, and patient lists are rendered dynamically from stored JSON data, supporting immediate report generation without batch processing.

V. SYSTEM DESIGN

The HMS architecture adheres to a client-side three-tier model comprising a presentation layer, a behaviour layer, and a data persistence layer. This structure separates concerns across distinct functional boundaries, facilitating modular development and independent testability of each component.

5.1 User Interface Design

The presentation layer is implemented in HTML5 and styled with CSS3, incorporating a responsive layout with a purple-gradient colour scheme and teal accent elements. The interface consists of a primary navigation bar providing access to the Home, Blood Bank, Feedback Form, Login, and Emergency modules, together with a suite of role-specific dashboards for administrative and clinical personnel.

5.2 Database Design

This is because there is no traditional relational database, so data is stored in the browser's Local Storage API. Unique storage keys are assigned to each entity type, which includes patient records, physician profiles, appointment entries, feedback submissions and emergency requests. Before storing data records are serialized as JSON strings and retrieved as structured objects to provide structure object access throughout the application.

5.3 System Architecture

The system is structured like the Model View Controller (MVC) pattern: HTML markup is considered the View, JavaScript functions are considered the Controller logic, and Local Storage objects are considered the Model. JavaScript event handlers are triggered by user interactions, which read or write to Local Storage, and the relevant UI piece is re-rendered.

5.4 Data flow between modules. Shared Local Storage keys share data between modules. For instance, when a physician retrieves a patient record, information that he has recorded in the Patient Management module will be accessed by the Appointment Scheduling module; when calculating consultation charges, the Billing module will refer to the patient's information from the Patient Management module. This is a consistent architecture of the shared state, no inter-process communications are required.

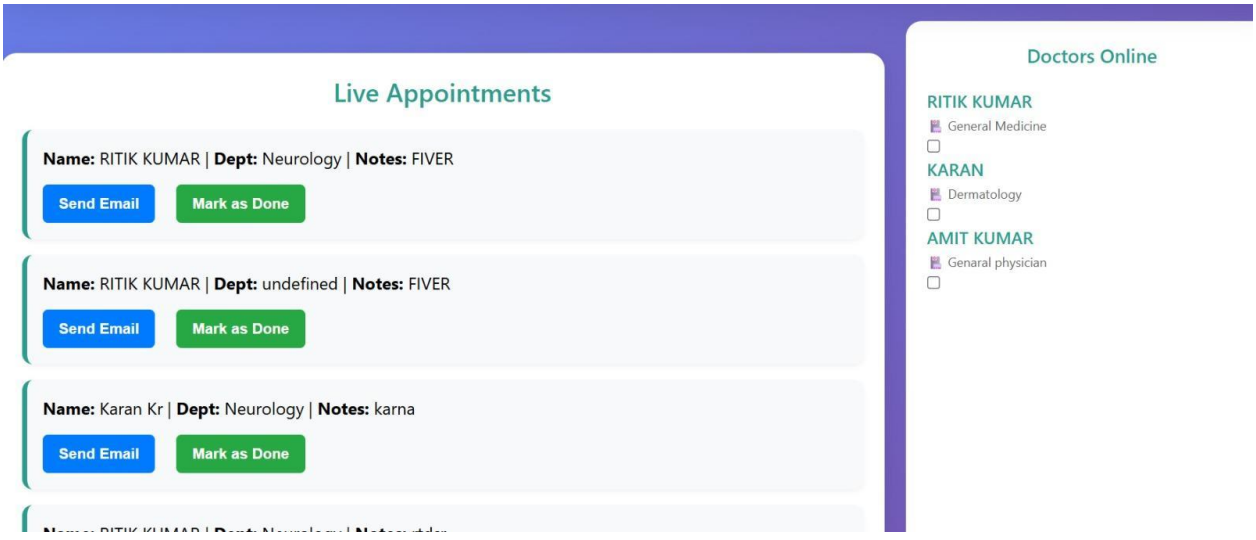


Fig. 1: Live Appointments panel with real-time physician availability sidebar

VI. TECHNOLOGY STACK

The HMS is developed using the following technology stack, selected for portability, zero-dependency deployment, and alignment with open web standards:

Layer	Technology	Purpose
Presentation	HTML5, CSS3	Page structure and visual styling
Behaviour	JavaScript (ES6)	Client-side logic, validation, and dynamic rendering
Data Persistence	Browser Local Storage (JSON)	Structured storage of patient, appointment, and feedback records
Development IDE	Visual Studio Code	Code authoring and debugging
Runtime / Test	Google Chrome	Application execution and user-acceptance testing

Table 1: Technology stack employed in the HMS implementation

The deliberate exclusion of server-side frameworks and external database engines reduces operational complexity and enables the system to function in offline or air-gapped environments, a property of practical significance in rural or low-connectivity healthcare settings.

VII. FUNCTIONAL MODULES OF THE HMS

The six functional modules of the HMS are broken down into separate units that perform specific functions in the hospital. Modular architecture promotes separation of concerns, and incremental feature extension.

7.1 **Patient Management Module**

This module provides a structured interface to register and maintain patients' records. Every patient entry is filled with patient's complete name, date of birth, contact number, email address, gender and medical history. Records are stored in Local Storage and are indexed for a system-generated Patient name identifier so that during appointment planning or clinical review, there is fast retrieval of records. The module can be used to create, read and update data using form-based data entry with client-side validation.

7.2 **Doctor Management Module**

The Doctor Management module keeps a register of doctors. Profile details include full name, department / specialisation, years of clinical experience, academic qualification, institution, geographic location and consultation charge. The doctor's profiles are displayed on the cards in the department's directory, letting patients choose a doctor to visit before booking an appointment.

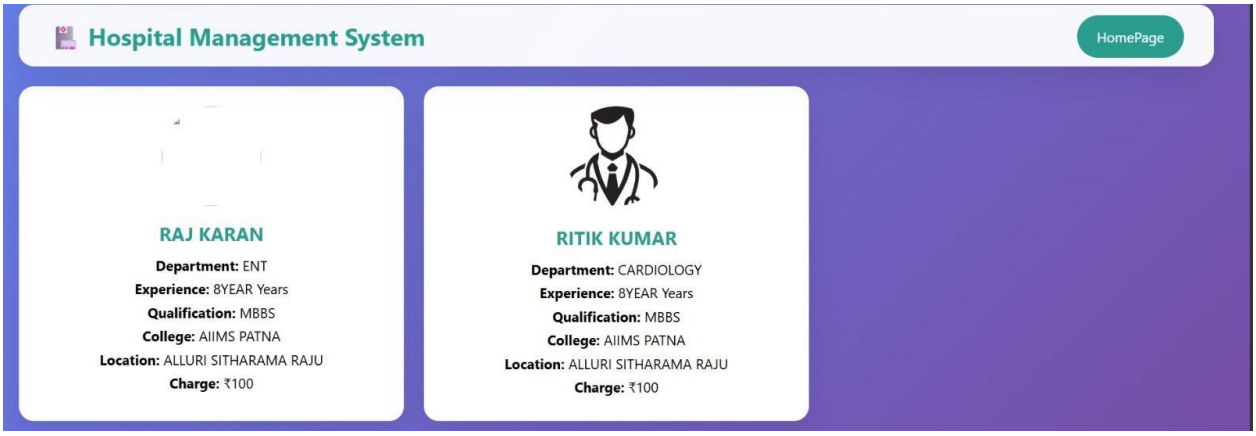


Fig. 2: Doctor profile cards displaying specialty, qualification, and consultation charge

7.3 Appointment Scheduling Module

The Appointment Scheduling module allows the authenticated patient to schedule an appointment with any registered physician of his/her choice for a particular department, date and time slot. All the details of the appointment will be stored in each appointment record including the appointment identifier, patient name, contact number, email address, department, physician attending, appointment date and time, clinical notes, and the status flag (Pending or Done). All of the appointments that are active will show up in the administrative dashboard in a Live Appointments panel, and the Doctors Online sidebar will be visible to indicate which doctors are available. Administrators can set date for appointments to be completed, or send an e-mail notification straight from the dashboard interface.

7.4 Feedback System

The Feedback System is a well-defined instrument for patients to make evaluative comments about the departmental services and doctors after a consultation. The patient name, email address, contact number, patient identifier, visit date, department, the doctor who treated the patient, and free-text feedback content are all captured in each feedback record. Feedback submitted is saved in Local Storage and presented in a tabular Patient Feedback List that is available to hospital administrators to support service quality monitoring and physician performance review.

Patient Feedback List

Name	Email	Phone	Patient ID	Date	Department	Doctor	Feedback
Anjali kumari	sharmaanjali5151@gmail.com	08409745492	pioiuh	2026-04-20	OPD	'piyi	lkuhyt
Anjali kumari	sharmaanjali5151@gmail.com	08409745492	pioiuh	2026-04-20	OPD	'piyi	lkuhyt
ankita kumari	ankita@gamil.com	1234567890	12345678	2026-04-19	Emergency	ritik kumar	good
anjali kumari	sharmaanjali5151@gmail.com	9508921809	12345678	2026-04-21	Cardiology	amit kumar	best treatment

Fig. 3: Patient Feedback List table displaying multi-departmental feedback records

7.5 Emergency Management Module

In case of an emergency, the Emergency Management module can be accessed from a conspicuously located EMERGENCY button in the main navigation bar, which reduces response time during emergency situations. The module displays a quick form for data entry when the module gets activated, with the patient's name, contact number, state, district and target hospital name. Submission of this form will automatically initiate a notification process to the designated hospital, which will alert them to an incoming emergency case, allowing them to prepare clinical resources ahead of time. This module focuses on fast data entry, rather than complete data record-keeping, consistent with the operational nature of emergency response scenarios.

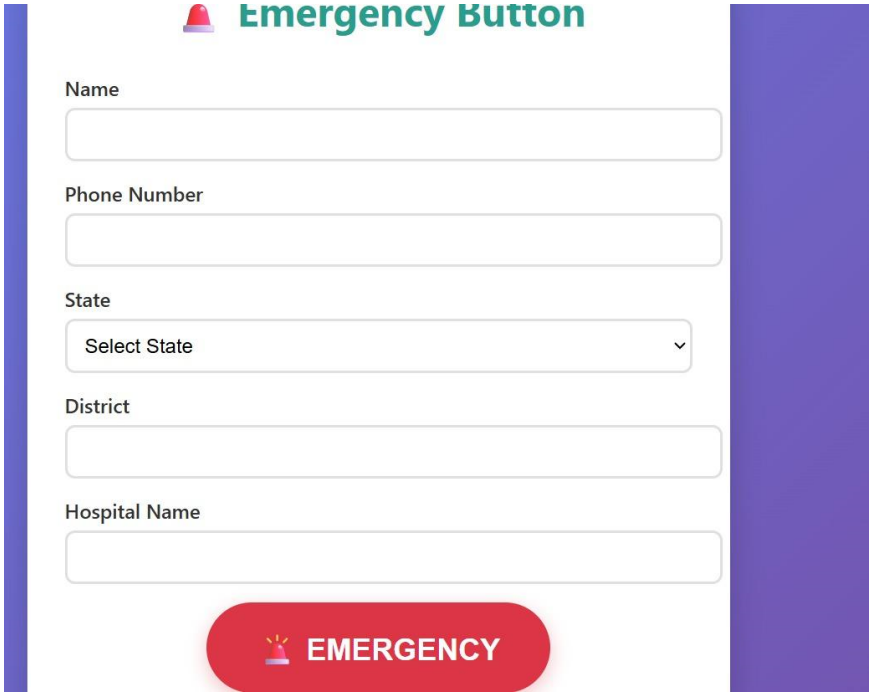


Fig. 4: Emergency Button form for rapid patient intake during critical events

7.6 Pharmacy Management Module

The Pharmacy Management module keeps a list of all medications that are available for each patient's appointment record. Doctors can enter prescribed medications in the Appointment Detail screen, and these will be saved with each appointment entry. Prescription status (e.g., Koi medicine prescribed nahi hui abhi) can be tracked in the module for facilitating coordination between the clinical and the dispensing functions.

VIII. IMPLEMENTATION

The development of the HMS is done in a modular and iterative fashion. Every functional module is designed and tested separately and then integrated to the application. The development environment will be Visual Studio Code (code editor), and Google Chrome as the main runtime and debugging environment.

The front-end codebase consists of several HTML files, one for each module/view. The global design system is set using shared CSS stylesheets, such as the purple-gradient background, the teal typographic accents and the card-based layout elements. Each module has its own JavaScript files that contain event-handling code, Local Storage read/writes and dynamic DOM render functions.

The validation of the form inputs happens at the client-side and is performed by javaScript event listeners for each data-entry form, that listen to the submit event. Validation rules ensure that fields are mandatory, contact numbers and patient identifiers must be entered numerically, and appointments are properly scheduled.

Local Storage API used to serialise data to strings of JSON, and then deserialise it back to an array when retrieved. Multi-record storage is implemented by having arrays of entity objects stored under specific storage keys (e.g., “appointments”, “patients”, “feedback”) where individual entity records are appended and/or updated with array manipulation operations.

The admin dashboard combines all the data from all the storage keys to provide the Quick Overview (total number of patients, hospitals, clinics, and blood bank units) in the Home page. The in place status updates for appointment transitions (Pending → Done) are made as updates to the corresponding JSON record and the updated array is then re-serialized and stored.

Unit testing is done for the data persistence functions in each module, and integration testing is done for the cross-module data flows. Structured test scenarios are used to test for boundary cases such as empty form submission, duplicate patient identifiers, and overlapping appointments.

IX. COMPARATIVE ANALYSIS: ADVANTAGES AND DISADVANTAGES

The following comparative analysis evaluates the principal strengths and limitations of the proposed HMS relative to conventional manual hospital administration systems:

Advantages	Disadvantages
Automates routine hospital tasks, reducing manual effort and processing time.	High initial investment in software development, hardware, and staff training.
Centralised digital storage ensures fast retrieval of patient and clinical records.	Complete dependence on computing infrastructure; system failures halt operations.
Automated validation minimises data-entry errors and improves record accuracy.	Sensitive patient data is vulnerable to unauthorised access if security controls are inadequate.
Enables faster clinical decision-making through instant access to patient history.	Staff require structured training to operate the system competently.
Integrated appointment and billing modules eliminate scheduling conflicts and billing delays.	Browser-based local storage limits data persistence and multi-device synchronisation.

Table 2: Comparative analysis of the HMS against manual hospital administration

X. FUTURE SCOPE

The existing version of HMS is technically sound and is a working prototype on the client side, but there are opportunities for substantive technical improvements:

Persistent Multi-device Data: Data could be synced over time and accessed concurrently by multiple users on a server-side relational or NoSQL database, like MySQL.

2. Mobile Application Development: Native or hybrid mobile application (Android/iOS) to expand patient and clinical access to the system in settings where it is impractical to have the ability to access the system via the desktop browser (from a mobile device), to enable push notifications for reminders and alerts.

3. Online Appointment Booking: Integrating with a patient-visible web portal would enable patients to book appointments without having to wait in line at the reception desk, thus mitigating administrative overhead and wait time.

4. Implementation of JSON Web Token (JWT) authentication, AES-256 data encryption, and role-based access control (RBAC) would align with the healthcare data protection regulations, including HIPAA and the Indian Digital Health Mission guidelines.

5. Integrating machine learning models for preliminary symptom analysis and appointment triage could improve the clinical value of the system, facilitating the intelligent allocation of patients to the relevant speciality according to their self-reported symptoms.

6. Telemedicine Integration: It would be helpful to have video consultation features so that patients in geographically remote areas would have the ability to connect with their physician for synchronous consultation, without the need to change the appointment structure.

XI. CONCLUSION

In this paper, the design, implementation and testing of web based Hospital Management System using HTML5, CSS3 and JavaScript have been introduced, and the implementation of Local Storage in the browser has been used as a data persistence tool. It combines six functional modules (patient management, doctor management, appointment management, patient feedback, emergency response and pharmacy management) and is browser-based without requiring any type of server infrastructure.

System analysis shows that the HMS offers advantages in terms of efficiency in administration, the accuracy of the data and interdepartmental coordination, when compared to the paper-based flow of information in a hospital. The implementation is a test of the technical feasibility of using client-side web technologies as a basis for lightweight healthcare information systems in resource-constrained environments.

The current architecture is Local Storage-based and therefore is limited in scope and has only one user, but is a good prototype on which a production-grade system can be incrementally developed. The planned expansion of the prototype—such as supporting a backend database system, mobile deployment, adding more security measures, and AI-powered triage capability—provides a blueprint for the prototype's transformation into a complete, real-world healthcare information system. The proposed future enhancements to the prototype, including integration with a backend database system, mobile application support, additional security features, and AI-powered triage capabilities, serve as a roadmap for the evolution of the prototype into a full-fledged healthcare information platform. To conclude, the Hospital Management System illustrates that there is a potential for the useful utilisation of open web standards to create a functional, maintainable and extensible automation solution in the healthcare sector, which is suitable for use in small and medium-sized hospital settings.

REFERENCES

[1] R. Shortliffe and J. Cimino, Biomedical Informatics: Computer Applications in Health Care and Biomedicine, 4th ed. Springer, 2014.

[2] A. K. Jha et al., “Use of Electronic Health Records in U.S. Hospitals,” New England Journal of Medicine, vol. 360, no. 16, pp. 1628–1638, 2009.

[3] C. P. Friedman and J. C. Wyatt, Evaluation Methods in Biomedical Informatics, 2nd ed. Springer, 2006.

[4] M. A. Musen, “The protégé project: A look back and a look forward,” AI Matters, vol. 1, no. 4, pp. 4–8, 2015.

[5] World Health Organization, “Digital Health,” WHO Global Strategy on Digital Health 2020–2025, Geneva, Switzerland, 2021.

[6] Government Polytechnic Katihar, “Hospital Management System — Project Report,” Diploma in Electronics Engineering, State Board of Technical Education, Patna, Bihar, India, 2026.