**Hospital Management System**

*A Project Report Submitted*

*to*

**MANIPAL ACADEMY OF HIGHER EDUCATION**

*For Partial Fulfillment of the Requirement for the*

*Award of the Degree*

*Of*

**Bachelor of Technology**

*in*

**Information Technology**

*by*

**Manshita Agarwal, Riddhima Chauhan, Dakshayani Sharma**

**230911132, 230911072, 230911066**

*Under the guidance of*

Ms. Swathi B. P Mr. Ranjith K

Assistant Professor – Senior Scale Assistant Professor

Department of I&CT Department of I&CT

Manipal Institute of Technology Manipal Institute of Technology

****Manipal, Karnataka, India Manipal, Karnataka, India

**April 2025**

**Table of Contents**

1. **Abstract 3**
2. **List of Tables.......................................................................................................................4**
3. **List of Figures.....................................................................................................................5**
4. **Abbreviations.....................................................................................................................5**
5. **Chapter1.............................................................................................................................6**

* Introduction........................................................................................................................................

1.1 Background...........................................................................................................................

1.2 Motivation.............................................................................................................................

1.3 Overview of the System........................................................................................................

1.4 Scope of the project...............................................................................................................

**6. Chapter 2............................................................................................................................7**

* Literature Survey.....................................................................................................................

2.1 Existing Systems..........................................................................................................

2.2 Technological Foundation.....................................................................................................

2.3 Importance of Modular Architecture.....................................................................................

2.4 Relevance of Database Normalization...................................................................................

**7. Chapter 3............................................................................................................................8**

* Problem Statement..................................................................................................................

3.1 Problem Statement................................................................................................................

3.2 System Objectives.................................................................................................................

3.3 Key Features...............................................................................................................

**8. Chapter 4............................................................................................................................9**

* Data Design ...........................................................................................................................

4.1 ER Diagram..............................................................................................................9

4.2 Reduction.................................................................................................................9

4.3 Relational Schema............................................................................................................10

4.4 Functional Dependencies..................................................................................................10

4.5 Closure Property......................................................................................................12

4.6 Normalization..........................................................................................................13

**9. Chapter 5............................................................................................................................15**

* Methodology**...........................................................................................................................**

5.1 System Design Approach.................................................................................................15

5.2 Requirement Analysis.......................................................................................................15

5.3 System Architecture.........................................................................................................15

5.4 Database Design and Normalization................................................................................15

5.5 Development Methodology..............................................................................................16

5.6 Security and Data Validation............................................................................................16

5.7 Review and Feedback System..........................................................................................16

**10. Chapter 6...........................................................................................................................17**

* Results....................................................................................................................................

6.1 Functional Implementation..............................................................................................17

6.2 Database Accuracy and Normalization...........................................................................17

6.3 Technology Integration...................................................................................................17

6.4 Performance and Reliability............................................................................................17

6.5 User Feedback System.....................................................................................................18

**11. Chapter 7...........................................................................................................................19**

* Conclusion .............................................................................................................................

7.1 Conclusion..........................................................................................................19

7.2 Future Work........................................................................................................19

**12. Chapter 8..........................................................................................................................20**

* References............................................................................................................................

**ABSTRACT**

The Hospital Management System (HMS) is a comprehensive software solution developed to improve the efficiency and accuracy of hospital operations. It addresses common issues found in traditional healthcare management, such as delayed processes, manual errors, and difficulty in maintaining organized records. HMS integrates essential hospital functions including patient registration, appointment scheduling, doctor and treatment management, room allocation, billing, and discharge procedures.

Built using Visual C# for the frontend and SQL Server for backend data handling, the system offers a user-friendly interface and a secure, well-structured database. Its modular design allows for easy maintenance and future enhancements. By automating routine tasks and centralizing hospital data, HMS contributes to better decision-making, reduced administrative workload, and improved patient care.

**ACM Taxonomy**

* **[Information Systems]**: Database management system engines; Information systems applications; Decision support systems
* **[Software Engineering]**: Software system models
* **[Human-centred computing]**: Graphical user interfaces
* **Sustainable Development Context:**  
  Optimizing hospital operations contributes to sustainable urban healthcare services by reducing paper usage, minimizing resource waste, and promoting efficient and timely medical care delivery.

**List of Tables**

* Patient
* Doctor
* Appointment
* Room
* Room\_detail
* Bill
* Lab
* Test\_Price
* Inventory
* Patient\_Report
* Prescribed\_Med

**List of Figures**

Figure 4.1: Entity-Relationship (ER) Diagram of Hospital Management System

Figure 4.3: Relational Schema Diagram after Normalization

**Abbreviations**

| **Abbreviation** | | | **Full Form** |
| --- | --- | --- | --- |
| HMS | Hospital Management System | | |
| ER | Entity-Relationship | | |
| DBMS | Database Management System | | |
| SQL | Structured Query Language | | |
| UI | User Interface | | |
| SRS | Software Requirements Specification | | |
| PK | Primary Key | | |
| FK | Foreign Key | | |
| FD | Functional Dependency | | |
| 1NF | First Normal Form | | |
| 2NF | Second Normal Form | | |
| 3NF | Third Normal Form | | |
| API | Application Programming Interface | | |

**Chapter 1**

**INTRODUCTION**

* 1. **Background**

Hospitals handle vast amounts of data across patient care, appointments, billing, and diagnostics. Manual processes are time-consuming, error-prone, and inefficient. As healthcare demands grow, a centralized digital system is essential to streamline operations and improve service quality.

* 1. **Motivation**

Traditional hospital workflows often lead to delays, misplaced records, and billing errors. Patients expect timely care and transparency, while staff need efficient tools to reduce administrative load. A Hospital Management System (HMS) addresses these issues by automating core tasks and ensuring reliable data management.

* 1. **Overview of the System**

The HMS is a desktop application with modules for patient registration, appointments, billing, diagnostics, and inventory. Built with C# and SQL Server, it supports role-based access for admins, doctors, and receptionists, offering a secure and user-friendly interface.

**1.4 Scope of the Project**

The system includes:

* Patient data management and appointment scheduling
* Billing with multi-mode payment options
* Inventory and lab test tracking
* Report generation and user feedback
* Secure login and role-based access

**Chapter 2**

**LITERATURE SURVEY**

**2.1 Existing Systems**

Traditional hospital systems are fragmented, often handling patient records, billing, and inventory separately. This leads to data redundancy, inefficiencies, and delays in patient care.

**2.2 Technological Foundations**

* **Frontend**: C# with Visual Studio
* **Backend**: SQL Server
* **Architecture**: Multi-tier, ensuring separation of concerns and scalability

**2.3 Importance of Modular Architecture**

Breaking the system into modules—Patient, Appointment, Billing, Lab, Inventory—simplifies development, enhances maintainability, and allows easy future upgrades.

**2.4 Relevance of Database Normalization**

Normalization (up to 3NF) ensures data consistency, reduces redundancy, and improves query performance—vital for handling sensitive healthcare data.

**Chapter 3**

**PROBLEM STATEMENT**

**3.1 Problem Statement**

Manual hospital operations—such as patient registration, appointment scheduling, billing, and inventory tracking—are inefficient and error-prone. There is a lack of integration between these processes, leading to delays, mismanagement, and poor patient experience.

**3.2 System Objectives**

The HMS aims to:

* Centralize patient, doctor, and billing data
* Automate appointment scheduling and notifications
* Track medicine inventory and lab diagnostics
* Enable role-based access and data security
* Provide reports and analytics for hospital administration

**3.3 Key Features**

* All-in-one platform for hospital operations
* Real-time data updates across modules
* Automated billing and multi-mode payments
* Secure login for admins, doctors, and receptionists
* User-friendly interface with dashboard and reports

**Chapter 4**

**Data Design**

**4.1 ER Diagram**

A diagram of a company

Description automatically generated

Figure 4. 1

**4.2 Reduction**

Strong Entities:

* PATIENT (patient\_id, name, gender, dob, phone, email, address)
* DOCTOR (doctor\_id, name, specialist, phone)
* APPOINTMENT (appointment\_id, patient\_id, doctor\_id, appointment\_date, appointment\_time, appointment\_desc)
* ROOM (room\_no, room\_type, no\_of\_days)
* ROOM\_DETAIL (room\_no, room\_type, room\_price, no\_bed)
* BILL (bill\_no, patient\_id, doc\_charge, med\_charge, lab\_charge, room\_charge)
* LAB (lab\_no, patient\_id, test\_type)
* TEST\_PRICE (test\_code, test\_type, test\_price)
* INVENTORY (med\_id, med\_name, company, quantity, exp\_date, med\_price)
* PATIENT\_REPORT (report\_id, patient\_id, diagnose, doctor\_id)

Weak Entities:

* PRESCRIBED\_MED (report\_id, med\_id, dosage, duration)

Composite key: (report\_id, med\_id)

* HAS\_REPORT, ALLOCATED\_TO, BELONGS\_TO

Captured using foreign keys in associated tables (e.g., PATIENT\_REPORT, ROOM\_DETAIL, PRESCRIBED\_MED)

Relational Tables:

* Assigned\_To (assignment\_id, order\_id, staff\_id, assignment\_type, assignment\_time)

Used for tracking which staff member is assigned to which task (e.g., test, prescription, patient care).

Relationships and Their Reductions:

* A **patient** can have multiple **appointments**; each appointment is linked to one patient → *Many-to-One (PATIENT ← APPOINTMENT)*
* A **doctor** can attend multiple appointments → *Many-to-One (DOCTOR ← APPOINTMENT)*
* **ROOM\_DETAIL** links patients to rooms → *ALLOCATED\_TO* (ROOM ←→ PATIENT)
* **BILL** is linked to each **PATIENT** and summarizes charges → *One-to-One (PATIENT ← BILL)*
* **PATIENT\_REPORT** stores diagnosis data and is linked to **PRESCRIBED\_MED** and **DOCTOR**, forming the basis for prescriptions → *One-to-Many.*

**4.3 Relational Schema**

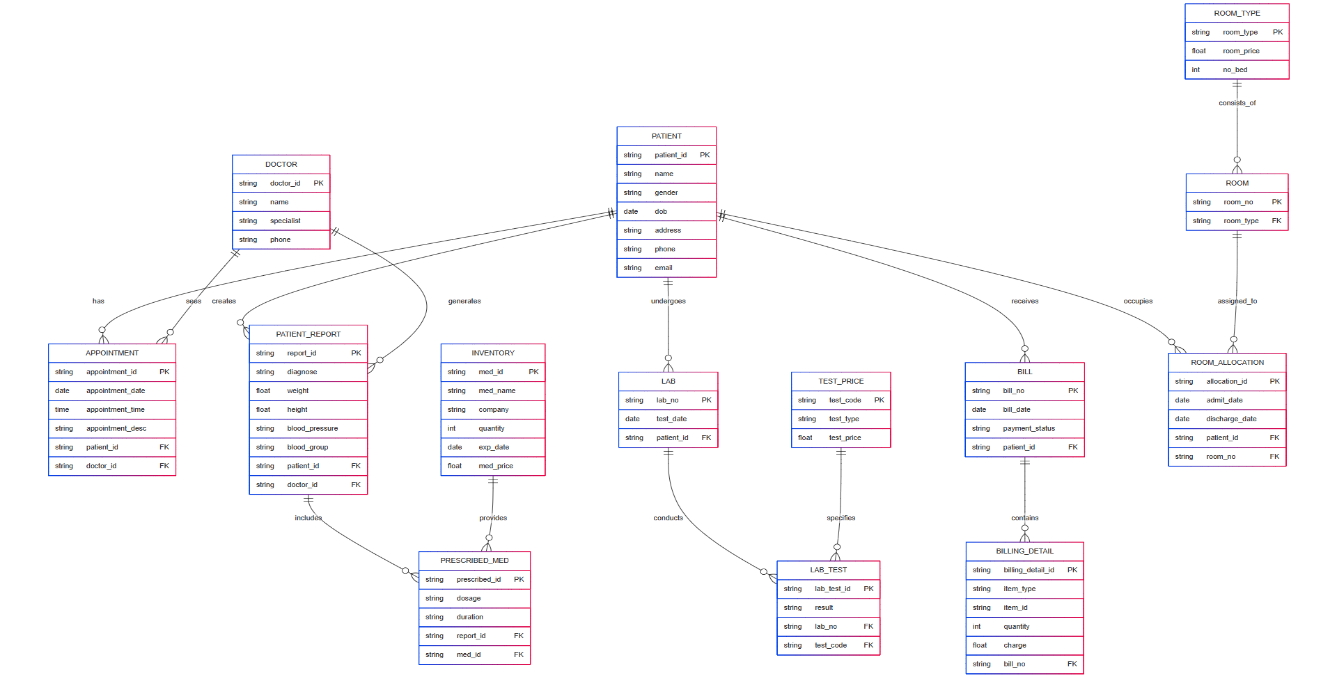


Figure 4.3

**4.4 Functional Dependencies:**

* **Patient**

patient\_id → name, gender, dob, phone, email, address

* **Doctor**

doctor\_id → name, specialist, phone

* **Appointment**

appointment\_id → patient\_id, doctor\_id, appointment\_date, appointment\_time, appointment\_desc

* **Room**

room\_no → room\_type, no\_bed

* **Room\_Detail**

room\_no, patient\_id → no\_of\_days

* **Inventory**

med\_id → med\_name, company, quantity, exp\_date, med\_price

* **Test\_Price**

test\_code → test\_type, test\_price

* **Bill**

bill\_no → patient\_id, doc\_charge, med\_charge, lab\_charge, room\_charge

* **Patient\_Report**

report\_id → patient\_id, diagnose, doctor\_id

* **Prescribed\_Med**

report\_id, med\_id → dosage, duration

* **Lab**

lab\_no → patient\_id, test\_type

* **Assigned\_To**

assignment\_id → staff\_id, task\_type, task\_id, assigned\_time

**4.5 Closure Property:**

* Patient

{ patient\_id }⁺ = { patient\_id, name, gender, dob, phone, email, address }

* Doctor

{ doctor\_id }⁺ = { doctor\_id, name, specialist, phone }

* Appointment

{ appointment\_id }⁺ = { appointment\_id, patient\_id, doctor\_id, appointment\_date, appointment\_time, appointment\_desc }

* Room

{ room\_no }⁺ = { room\_no, room\_type, no\_bed }

* Room\_Detail

{ room\_no, patient\_id }⁺ = { room\_no, patient\_id, no\_of\_days }

* Inventory

{ med\_id }⁺ = { med\_id, med\_name, company, quantity, exp\_date, med\_price }

* Test\_Price

{ test\_code }⁺ = { test\_code, test\_type, test\_price }

* Bill

{ bill\_no }⁺ = { bill\_no, patient\_id, doc\_charge, med\_charge, lab\_charge, room\_charge }

* Patient\_Report

{ report\_id }⁺ = { report\_id, patient\_id, diagnose, doctor\_id }

* Prescribed\_Med

{ report\_id, med\_id }⁺ = { report\_id, med\_id, dosage, duration }

* Lab

{ lab\_no }⁺ = { lab\_no, patient\_id, test\_type }

* Assigned\_To

{ assignment\_id }⁺ = { assignment\_id, staff\_id, task\_type, task\_id, assigned\_time }

**4.6 Normalization**

**First Normal Form (1NF):**

* All attributes contain atomic values.
* Each table has a primary key.
* Status: All entities satisfy 1NF.

**Second Normal Form (2NF):**

* No partial dependencies on part of any composite key.
* Status: All entities either have atomic keys or full dependency on composite keys—satisfy 2NF.

**Third Normal Form (3NF):**

* No transitive dependencies.
* Issues and fixes:
  + **ROOM**: Transitive dependency via room\_type -> room\_price  
      → Split into room and room\_type
  + **BILL**: total\_bill was derived from other charges  
      → Remove total\_bill and split into bill and billing\_detail.
  + **LAB**: Patient vitals depended on patient\_id, not lab\_no  
      → Moved vitals to patient\_report.
  + **INVENTORY**: Potential dependency med\_name -> med\_price  
      → Optional split into medicine and medicine\_price.

**Normalization Outcomes:**

* All entities are in 3NF.
* Derived and transitive attributes were removed.
* Supporting tables were introduced to preserve relational clarity and efficiency.

**Final Schema after Normalization:**

Patient: Patient (patient\_id, name, gender, dob, phone, email, address)

Doctor: Doctor (doctor\_id, name, specialist, phone)

Appointment: Appointment (appointment\_id, patient\_id, doctor\_id, appointment\_date, appointment\_time, appointment\_desc)

Room: Room (room\_no, room\_type)

Room\_Detail: Room\_Detail (room\_no, patient\_id, no\_of\_days)

Room\_Type: Room\_Type (room\_type, room\_price, no\_bed)

Bill: Bill (bill\_no, patient\_id, bill\_date)

Billing\_Detail: Billing\_Detail (billing\_detail\_id, bill\_no, item\_type, item\_id, charge)

Inventory: Inventory (med\_id, med\_name, company, quantity, exp\_date, med\_price)

Test\_Price: Test\_Price(test\_code, test\_type, test\_price)

Lab: Lab(lab\_no, patient\_id, test\_type)

Patient\_Report: Patient\_Report(report\_id, patient\_id, diagnose, doctor\_id)

Prescribed\_Med: Prescribed\_Med(report\_id, med\_id, dosage, duration)

Assigned\_To: Assigned\_To(assignment\_id, staff\_id, task\_type, task\_id, assigned\_time)

**Chapter 5**

**Methodology**

The Hospital Management System (HMS) was developed through a step-by-step implementation process, emphasizing modularity, relational database design, and secure user interaction. The methodology was focused on clarity, data integrity, and functional efficiency.

**5.1 System Design Approach**

The system was divided into independent functional modules to simplify development and allow for focused testing and maintenance:

* Patient and Appointment Management
* Doctor and Staff Information
* Billing and Payment Processing
* Laboratory and Diagnostic Module
* Inventory and Pharmacy Management
* Reports and Feedback Interface

This modular structure ensured that each component could be developed and tested independently.

**5.2 Requirement Analysis**

* **Functional Requirements**: Manage patients, schedule appointments, handle billing, assign rooms, track medications, generate lab reports.
* **Non-Functional Requirements**: System usability, data consistency, basic access control, and performance under typical load conditions.

**5.3 System Architecture**

The application was built using a layered architecture with clear separation of concerns:

* **User Interface**: C# desktop application using Visual Studio
* **Database Backend**: Microsoft SQL Server
* **Interaction Layer**: SQL queries for CRUD operations (Create, Read, Update, Delete)

**5.4 Database Design and Normalization**

The database was designed using an ER model and converted into normalized relational tables:

* Normalization was done up to **Third Normal Form (3NF)**
* Transitive dependencies and derived attributes were eliminated
* Foreign keys were used to maintain **referential integrity**

**5.5 Implementation Steps**

The system was developed through a series of well-defined phases:

1. **Interface Design** – Creating forms for patient entry, billing, etc.
2. **Database Integration** – Connecting the frontend to SQL Server
3. **Module Development** – Building each functional area (appointments, billing, etc.)
4. **Testing** – Manual testing for correctness and usability
5. **Final Assembly** – Integrating all modules into a single working system

**5.6 Security and Data Validation**

* Simple **role-based restrictions** for access control (Admin, Doctor, Receptionist)
* **Input validation** to avoid incorrect or malicious data entries
* **Secure password storage** using basic hashing functions
* Logical constraints on date fields, ID formats, and relationships

**5.7 Review and Feedback System**

A feedback interface allows patients to submit service ratings and comments. This helps in monitoring system performance and identifying areas for future improvement.

**Chapter 6**

**Results**

The successful implementation of the Hospital Management System (HMS) led to a fully functional, modular application capable of managing essential hospital operations efficiently and accurately.

**6.1 Functional Implementation**

All key modules were developed and tested successfully:

* **Patient Module**: Enables registration, patient record viewing, and update.
* **Appointment Module**: Allows scheduling, updating, and canceling appointments.
* **Billing Module**: Automatically calculates charges for doctor consultations, lab tests, medicines, and room stays.
* **Staff Management**: Admins can assign roles and responsibilities to doctors and staff.
* **Lab and Inventory Module**: Tracks tests, generates reports, and manages medicine stock and expiry.
* **Report Module**: Allows diagnosis entry, prescription linking, and retrieval of patient records.

**6.2 Database Accuracy and Normalization**

* All database entities were normalized up to **Third Normal Form (3NF)**.
* Functional dependencies and candidate keys were well-defined for each table.
* Referential integrity was maintained using appropriate foreign keys.
* Weak entities such as prescribed\_med and relationship tables like room\_detail were modeled correctly with composite keys where necessary.

**6.3 Technology Integration**

* **Frontend**: Implemented using **C# and Windows Forms** for an intuitive and responsive interface.
* **Backend**: Data is managed through **SQL Server**, enabling structured queries, indexing, and reliable relational data storage.

**6.4 Performance and Reliability**

* The system handled data input, retrieval, and updates efficiently during testing.
* Queries returned results quickly due to optimized schema and indexing.
* No data anomalies were observed in patient, appointment, billing, or inventory operations.
* Consistent and accurate report generation and prescription tracking were achieved.

**6.5 User Feedback System**

* A feedback mechanism allowed users (patients) to rate service quality and submit suggestions.
* This feedback:
  + Helped in assessing user satisfaction.
  + Offered insights for improving hospital workflows and user experience.

**Chapter 7**

**Conclusion And Future Work**

### **7.1 Conclusion**

The Hospital Management System (HMS) developed as part of this project provides a functional and organized solution for managing core hospital operations. Built using **C# and SQL Server**, the system successfully integrates key modules such as patient management, appointments, billing, diagnostics, inventory, and staff coordination into a single desktop-based application.

The database was carefully designed and normalized up to **Third Normal Form (3NF)** to ensure consistency, eliminate redundancy, and support efficient querying. Each feature was implemented with a focus on usability, data integrity, and modularity, making the system reliable and easy to maintain.

This project demonstrates how college-level skills and tools can be applied to solve real-world administrative challenges in healthcare. It serves as a solid foundation for further development and expansion.

### **7.2 Future Work**

While the system meets its intended goals, several enhancements could be pursued to increase its usability and functionality—many of which are practical and achievable with continued learning and the tools already used in this project:

* **Basic Mobile Integration**: A lightweight mobile app (using tools like Android Studio or .NET MAUI) for patients to check appointments or receive updates.
* **Automated Notifications**: Email or SMS alerts for appointment reminders and report availability using C# mail libraries or third-party APIs like Twilio.
* **Enhanced User Roles and Dashboards**: Custom dashboards for admins, doctors, and receptionists to improve navigation and task tracking.
* **Basic Analytics and Reports**: Visual summaries of appointments, billing, and inventory using charting libraries (e.g., LiveCharts in C#).
* **Data Export**: Ability to export bills and reports as PDF documents using libraries like iTextSharp.
* **Multi-language Interface**: Supporting multiple languages for better accessibility, using resource files in .NET.

**Chapter 8**

**References**

 "Database System Concepts" by Silberschatz, Korth, and Sudarshan (for ER diagrams, normalization, relational schema)

 [gfg\_er\_diagram\_reference](file:///C:\Users\Dakshayani%20Sharma\AppData\Local\Microsoft\Windows\INetCache\IE\P5E77XH1\gfg_er_diagram_reference)

* [er\_diagram\_reference](file:///C:\Users\Dakshayani%20Sharma\AppData\Local\Microsoft\Windows\INetCache\IE\P5E77XH1\er_diagram_reference)
* [Govt. of India SDGs](https://sdgs.un.org/goals)
* [IEEE Format Style Guide](https://ieeeauthorcenter.ieee.org)
* [gfg\_windowsc#forms\_tutorials](file:///C:\Users\Dakshayani%20Sharma\AppData\Local\Microsoft\Windows\INetCache\IE\P5E77XH1\gfg_windowsc#forms_tutorials)