

RV COLLEGE OF ENGINEERING[®], BENGALURU-560059

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Hospital Management System

Mini - Project Report

Submitted by

L V S Aditya 1RV22CS095

Mehul Maheshwari 1RV22CS114

Mohammed Ilham 1RV22CS117

in partial fulfillment for the requirement of 5th Semester

DATABASE MANAGEMENT SYSTEMS (CD252IA)

Under the Guidance of

Dr. Suma B Rao

**Assist. Professor, Department of Computer Science and
Engineering, R.V. College of Engineering**

Academic Year 2024 - 2025

RV COLLEGE OF ENGINEERING[®], BENGALURU 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the project work titled '**Hospital Management System**' is carried out by **L V S Aditya (1RV22CS095), Mehul Maheshwari (1RV22CS114), Mohammed Ilham (1RV22CS117)**, who are bonafide students of R. V. College of Engineering, Bengaluru, in partial fulfillment of the curriculum requirement of 5th Semester **Database Management Systems(CD252IA)** Laboratory Mini Project during the academic year **2024-2025**. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report. The report has been approved as it satisfies the academic requirements in all respect laboratory mini-project work prescribed by the institution.

Dr Suma B Rao
Faculty Incharge
Department of CSE
R. V. C. E, Bengaluru

Dr. Shanta Rangaswamy
Head of the Department
Dept. of CSE
R. V. C. E, Bengaluru

External Examination

Name of Examiners

Signature with date

1

2

Acknowledgement

We would like to express our heartfelt gratitude to all those who contributed to the successful completion of our project titled **Hospital Management System**

First and foremost, we are extremely grateful to **Dr. Suma B**, our project guide, for their valuable guidance, support, and encouragement throughout the duration of this mini-project. Their expertise and insights have been instrumental in shaping this project.

We extend our sincere thanks to **Dr. Shantha Rangaswamy**, Head of the Department, Computer Science and Engineering, **R.V. College of Engineering**, Bengaluru, for providing us with the necessary resources and a conducive environment for our work.

We are also thankful to the **Database Management Systems Laboratory (CD252IA)** faculty and staff for their support and feedback, which helped us refine our work.

It has been a great learning experience, and we are grateful for the opportunity to work on this mini-project as part of the 5th Semester curriculum for the academic year 2024-2025.

Thank you all for your guidance and support.

L V S Aditya (1RV22CS095)

Mehul Maheshwari (1RV22CS114)

Mohammed Ilham (1RV22CS117)

5th Semester, Computer Science and Engineering,
R. V. College of Engineering, Bengaluru.

Abstract

The Hospital Management System (HMS) is an integrated solution that automates hospital processes, including patient registration, appointment scheduling, medical record management, and department coordination, providing real-time access to hospital information via role-based access for patients, doctors, and administrators. Powered by large language models (LLMs), the AI-based chatbot minimizes manual workload by enabling natural language database queries and streamlining task automation. Data is securely stored in a relational system with encryption and authentication mechanisms, designed for scalability and future integrations such as voice recognition and advanced analytics. The user-friendly Streamlit interface improves operational efficiency, reduces bottlenecks, and enhances patient-care outcomes. Built with Python and Streamlit, the system allows patients to easily manage appointments and access records, while doctors efficiently handle their schedules and clinical records. Using advanced database management techniques like entity-relationship modeling, normalization, and secure transaction handling, HMS ensures reliable data storage. With secure role-based access and an AI-powered chatbot automating database interactions, HMS demonstrates modern software engineering practices to deliver an impactful healthcare management solution. for 7 seconds

The Hospital Management System (HMS) is an integrated solution designed to streamline hospital processes by automating tasks such as patient registration, appointment scheduling, medical record management, and department coordination. It employs a role-based access model to ensure tailored functionality for patients, doctors, and administrators, and is powered by large language models (LLMs) and an AI-based chatbot that reduces manual workload through natural language database queries. All data is securely stored in a relational system with robust encryption and authentication mechanisms, ensuring the protection of sensitive medical information.

Built using Python and Streamlit, HMS offers a user-friendly interface that allows patients to register, book appointments, and access records easily while enabling doctors to manage schedules and update clinical histories efficiently. Leveraging advanced database management techniques—such as entity-relationship modeling, normalization, and secure transaction handling—the system is designed for scalability and future integrations like voice recognition and advanced analytics. Overall, HMS minimizes administrative bottlenecks, enhances data retrieval, and improves both operational efficiency and patient care outcomes.

Table of Contents

	Page No.
1. Introduction	6
1.1 Objective.....	6
1.2 Scope.....	7
1.3 Definitions, Acronyms, and Abbreviations.....	7
1.4 Overview.....	7
2. Software Requirement Specification	8
2.1 Software Requirements.....	8
2.2 Hardware Requirements.....	10
2.3 Functional Requirements.....	10
2.4 Non-Functional Requirements.....	11
3. Entity Relationship Diagram	14
4. Data Flow Diagram	18
4.1 DFD Level 0.....	18
4.2 DFD Level 1.....	19
5. Relational Schema and Normalization	22
6. Conclusion	25
7. References	26
8. Appendix	27

List of Figures

Figure No	Figure Name	Page No.
1.	Entity Relationship Diagram	14
2.	Data Flow Diagram Level-0	19
3.	Data Flow Diagram Level-1	21
4.	Relational Schema	22

Glossary

ER : Entity Relationship Diagram

DFD : Data Flow Diagram

SRS : Software Requirement Specification

1. Introduction

In today's rapidly evolving healthcare landscape, managing patient and doctor interactions efficiently is more critical than ever. Our project harnesses the power of advanced large language models (LLMs) to revolutionize hospital management systems by automating complex administrative tasks. This approach not only mitigates the need for manual intervention but also ensures that interactions are both intuitive and context-aware.

By seamlessly integrating with a robust SQL database, the chatbot is designed to handle a range of essential functions including patient registration, appointment scheduling, and medical record management. It further enhances operational efficiency by guiding users through various departmental services, ensuring that both patients and staff have immediate access to the information they need.

Leveraging the capabilities of LLMs, the system delivers precise and user-friendly responses that adapt to the dynamic nature of healthcare environments. This integration of cutting-edge language technology with proven database management practices results in a powerful tool that streamlines administrative processes, reduces operational bottlenecks, and ultimately contributes to a more responsive and effective healthcare delivery system.

1.1 Objectives

- **Automate Database Queries:** Streamline the execution of complex SQL queries through an intuitive chatbot interface.
- **Enhance Patient Registration:** Simplify and expedite the process of adding new patients to the hospital management system.
- **Streamline Appointment Booking:** Enable efficient scheduling that reduces wait times and enhances the patient experience.
- **Facilitate Medical Records Access:** Provide quick and secure retrieval of patient medical histories and relevant data.
- **Improve Department Navigation:** Offer immediate access to department-specific information to assist users effectively.
- **Leverage Advanced LLM Technology:** Utilize large language models to ensure accurate, context-aware interactions and minimize manual intervention.

1.2 Scope

This project aims to develop a comprehensive hospital management system that leverages a role-based access model and integrates with a relational MySQL database to streamline administrative operations. The system is built using Streamlit to deliver an intuitive, web-based interface that caters to different user roles—patients, doctors, and administrators. It automates key functions including patient registration, appointment scheduling, and medical record management, ensuring that data is securely stored and efficiently retrieved. Patients can easily view their personal details, appointment history, and medical records, while doctors are empowered to manage their schedules and add new medical records. Administrators have access to an advanced AI-powered chatbot, which utilizes a large language model to translate natural language queries into SQL commands, thereby simplifying database interactions and maintenance.

1.3 Definitions, Acronyms, and Abbreviations

- **LLM** – Large Language Model
- **DBMS** – Database Management System
- **API** – Application Programming Interface
- **SRS** – Software Requirements Specification

1.4 Overview

Hospital Management System – The goal is to develop a robust, scalable, and user-friendly solution that streamlines patient-doctor interactions, automates administrative tasks, and facilitates seamless communication among healthcare providers and patients.

The proposed system leverages relational Database Management Systems (DBMS) to securely store and manage critical patient records, appointment data, and medical histories. It supports multi-user access with role-based authentication for patients, doctors, and administrators, integrates with external healthcare databases, and provides real-time analytics to enhance operational efficiency and patient care.

By implementing modern database-driven technology along with advanced AI capabilities, such as a large language model-powered chatbot, this system aims to overcome the limitations of traditional hospital management methods. It reduces manual intervention, ensures timely medical service delivery, and contributes to a more efficient, responsive, and safer healthcare environment.

2. Software Requirements Specification

A software requirements specification (SRS) is a description of a software system to be developed. The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules

2.1 Software Requirements

2.1.1. Programming Languages and Technologies

Python is used for the backend, managing chatbot logic, database connections, and AI-based responses. Frameworks like Flask or FastAPI handle communication, while NLP libraries like NLTK and BERT enhance chatbot understanding. A database like SQLite or MySQL stores patient records, appointments, and doctor availability.

JavaScript powers the interactive chatbot interface, allowing real-time updates and smooth user interactions. HTML and CSS structure and style the chatbot, with frameworks like Bootstrap or Tailwind CSS ensuring responsiveness. The frontend includes a chatbox, message input, and interactive animations for an engaging user experience.

2.1.2. Database Management System (DBMS)

The system uses MySQL as the primary database management system, allowing efficient storage and retrieval of patient details, doctor information, departments, appointments, and medical records. The pymysql library facilitates the connection between the Python backend and the MySQL database, ensuring smooth data transactions.

2.1.3. LLM and AI-Based NLP Tools

For natural language processing and chatbot functionalities, the system integrates various AI tools. streamlit is used for the frontend interface, while Groq serves as the LLM model provider. The chatbot is powered by ChatGroq for natural language understanding and response generation. Additionally, LangChain and its components, such as RecursiveCharacterTextSplitter, FAISS for vector storage, and GoogleGenerativeAIEmbeddings, enhance document retrieval and query processing. TensorFlow and NumPy handle computational tasks, particularly when dealing with image processing using the PIL library.

2.1.4. Backend Frameworks & Libraries

The system utilizes Streamlit for the web-based UI and Flask/FastAPI if an additional backend API is needed. For database connectivity, PyMySQL is required. The chatbot leverages LangChain for AI-powered query generation and Groq API for AI-based SQL query optimization. Additionally, Google Generative AI API is used for embedding generation, while FAISS provides efficient vector search for document retrieval. Other essential libraries include NumPy for numerical operations, TensorFlow for machine learning integration, and PIL (Pillow) for image processing.

2.1.5. Libraries and Tools

The system uses several essential libraries for development and data handling. NumPy and Pandas are employed for numerical operations and data analysis, respectively. PyMySQL connects the Python backend to the MySQL database, ensuring smooth data management. Streamlit is the primary tool for creating the web-based user interface, while LangChain enhances NLP capabilities with tools like FAISS for efficient document retrieval and RecursiveCharacterTextSplitter.

For AI and machine learning tasks, the system utilizes TensorFlow and Pillow (PIL) for computational tasks and image processing. Groq supports natural language understanding, while GoogleGenerativeAIEmbeddings aids in query processing. JWT can be used for secure token-based authentication. Additionally, Flask or FastAPI may be employed for backend API development if needed.

2.2 Hardware Requirement Specification

- CPU, RAM (min 4GB), Processor: Intel i5 12th Gen (Quad Core or Higher)
- GPU: Integrated GPU (Intel Iris, Nvidia)

2.3 Functional Requirements

2.3.1. User Authentication & Role-Based Access

The system implements a secure authentication mechanism that allows users to log in using their credentials. Based on the role assigned (patient, doctor, or admin), users receive different access permissions. Patients can view personal details, appointments, and medical records. Doctors can manage patient data, view their appointments, and update medical records. Administrators have access to execute SQL queries and manage the entire hospital database. This ensures controlled and secure access to sensitive hospital data.

2.3.2. Patient Information Retrieval

The system enables users to retrieve patient details through natural language queries. When a user inputs a query like *"Show details of patient John"*, the chatbot processes the request and converts it into an SQL statement that fetches the relevant data from the database. The retrieved information, including First Name, Last Name, Age, Gender, Contact Number, and Address, is then displayed in a tabular format within the chatbot interface, ensuring quick and structured access to patient data.

2.3.3. Doctor Information Retrieval

Users can inquire about doctors based on their specialty or department. For example, if a user queries *"List all cardiologists"*, the NLP module processes the request and generates an appropriate SQL query. The system then fetches doctor details such as Name, Specialty, Contact Number, and Department, displaying them in a structured format. This function helps patients and administrators easily access doctor-related information within the hospital database.

2.3.4. Appointment Management & Scheduling

The system allows patients to view their appointments and book new ones through an interactive interface. When a patient enters a query such as *"Show my appointments on December 1, 2024"*, the

chatbot processes the request and retrieves all relevant appointment details, including Doctor Name, Date, Time, and Reason for the visit. Additionally, patients can book new appointments by selecting a doctor, date, and time, ensuring a seamless booking experience.

2.3.5. Department Information Retrieval

The chatbot assists users in retrieving department-related information. If a user asks *"Which department is on the 3rd floor?"*, the system converts this into an SQL query to fetch relevant data from the Departments table. The output includes the Department ID and Name, allowing users to quickly access hospital facility details.

2.3.6. Medical Records Retrieval & Management

The system allows patients and doctors to access and manage medical records efficiently. When a patient queries *"Show my medical records"*, the chatbot fetches records containing Diagnosis, Treatment, Date, and the consulting Doctor's details. Similarly, doctors can update patient medical history by entering new diagnoses and treatments, ensuring that medical data remains up-to-date and easily accessible.

2.3.7. AI-Powered Chatbot for Query Execution

An AI-driven chatbot integrated into the system processes user queries in natural language and converts them into structured SQL queries. This ensures non-technical users, such as patients or administrative staff, can efficiently retrieve data without requiring SQL knowledge. For administrative purposes, the chatbot also allows the execution of complex SQL queries, providing real-time database interaction and ensuring smooth hospital management operations.

2.4 Non-Functional Requirements

2.4.1. Performance

- **Response Time:**

The system should process user queries and display results within 2 seconds to ensure a seamless user experience.

- **Database Efficiency:**

SQL queries should be optimized to minimize execution time, especially for large datasets. Indexing and caching may be used to improve performance.

- **System Throughput:**

The system should handle multiple simultaneous user queries without noticeable slowdowns.

2.4.2. Scalability

- **Horizontal Scaling:**

Add more servers as user demand grows to distribute load and prevent bottlenecks.

- **Vertical Scaling:**

Upgrade hardware resources (e.g., CPU, RAM) on existing servers to boost performance.

- **Database Scalability:**

Use techniques such as partitioning or clustering to manage large datasets efficiently as the system grows.

2.4.3. Security

- **Data Encryption:**

All data transmitted between the client and server should be encrypted using HTTPS to protect against eavesdropping.

- **Input Validation and Sanitization:**

User inputs should be validated and sanitized to prevent security risks like SQL injection and cross-site scripting (XSS).

- **Authentication:**

If user login is implemented, passwords must be securely hashed using algorithms like bcrypt. Multi-factor authentication can enhance security.

- **Access Control:**

Limit access to sensitive database operations based on user roles and permissions.

2.4.4. Usability

- **User Interface Design:**

The chatbot interface should be simple and intuitive, ensuring users of all technical levels can easily interact with it.

- **Error Messaging:**

Provide clear error messages to guide users in correcting their queries (e.g., "Invalid patient ID.").

- **User Guidance:**

Include tooltips or instructions to help users frame their queries properly, enhancing the overall experience.

3. ENTITY RELATIONSHIP DIAGRAM

The Entity-Relationship (ER) Diagram provides a visual representation of the hospital management system's database structure. It illustrates the key entities—such as Patients, Doctors, Departments, Appointments, Medical Records, and Users—their attributes, and the relationships among them. This diagram is essential for designing a robust relational database that ensures data consistency, integrity, and accessibility. The relationships among these entities highlight how patient details, appointment scheduling, and medical record management are handled efficiently within the system, ultimately streamlining hospital operations.

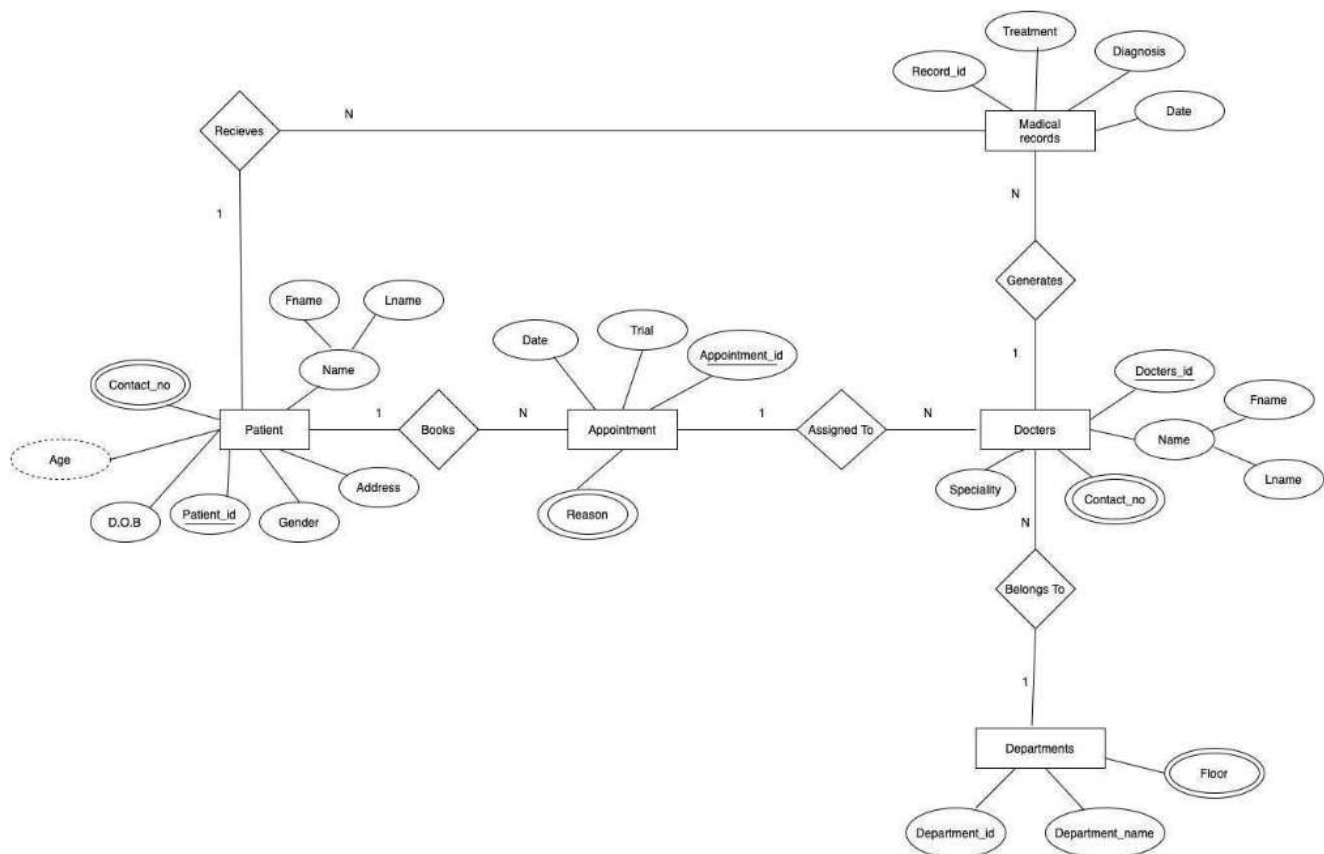


Figure 3.1 : ER Diagram of Vaccination Management System

Entities and Attributes

Patients

- Represents individuals receiving care in the hospital.
- Attributes include personal details such as first name, last name, age, gender, contact information, and address.

Doctors

- Captures details of the hospital's medical practitioners.
- Attributes include doctor ID, name, specialty, contact information, and the associated department.

Departments

- Stores information about the various departments within the hospital.
- Attributes include department ID, department name, and floor number.

Appointments

- Records the scheduled appointments between patients and doctors.
- Attributes include appointment ID, patient ID, doctor ID, date, time, and reason for the appointment.

Medical Record

- Maintains a history of patient diagnoses and treatments.
- Attributes include record ID, patient ID, doctor ID, diagnosis, treatment, and the date of the record.

Users

- Represents the system users, including patients, doctors, and admins.
- Attributes include user ID, username, password, role, and optional links to patient or doctor IDs.

Relationships

1. **Patient – Appointments:** A single patient can have multiple appointments, but each appointment is associated with exactly one patient.
2. **Doctor – Appointments:** A doctor can handle multiple appointments, but each appointment is tied to a single doctor.
3. **Doctor – Departments:** Each doctor belongs to one department, while a department can have multiple doctors.
4. **Medical Records – Patient & Doctor:** Each medical record links to exactly one patient and one doctor; a patient and doctor can be associated with multiple medical records over time.
5. **Users – Patients/Doctors:** A user can be a patient, doctor, or admin, with optional foreign keys referencing the Patients or Doctors tables.

Attributes

- **Patients:** PatientID, FirstName, LastName, Age, Gender, ContactNumber, Address
- **Doctors:** DoctorID, FirstName, LastName, Specialty, ContactNumber, DepartmentID
- **Departments:** DepartmentID, DepartmentName, Floor
- **Appointments:** AppointmentID, PatientID, DoctorID, Date, Time, Reason
- **MedicalRecords:** RecordID, PatientID, DoctorID, Diagnosis, Treatment, Date

Flow of Information

Patient Registration and Management

- Patients register on the platform by providing essential details such as PatientID, first name, last name, age, gender, contact information, and address.
- Administrators and healthcare staff manage patient records to ensure the data remains accurate, secure, and up-to-date.

Appointment Scheduling and Processing

- Patients book appointments by selecting a doctor, date, and time, with the system capturing details such as AppointmentID, PatientID, DoctorID, date, time, and the reason for the visit.
- Doctors access their schedules to view upcoming appointments, and administrators oversee appointment processing to maintain an efficient workflow.

Medical Record Management

- Doctors create and update medical records by recording diagnosis, treatment, and relevant dates, linking these records to the respective PatientID and DoctorID.

- Both patients and doctors can access these records, ensuring that critical medical history is readily available while administrators maintain data integrity.

Doctor and Department Management

- The system stores comprehensive doctor details, including DoctorID, first name, last name, specialty, contact information, and associated DepartmentID, alongside department data such as department name and floor.
- Administrators regularly update these records to ensure accurate scheduling, proper resource allocation, and seamless inter-departmental communication.

Admin Management

- Administrators oversee the overall operation of the system, managing patient records, appointments, medical records, and doctor/department data.
- They utilize an AI-powered chatbot interface to execute database queries, monitor system performance, and ensure data security, thereby facilitating smooth and efficient hospital operations.

4. DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a common visual tool used to show how information moves through a system, giving a clear overview of its requirements. It can include manual, automated, or a mix of processes, showing where data comes from, where it goes, how it's processed, and where it's stored. The main purpose of a DFD is to clearly define the boundaries and scope of a system. DFDs are structured in levels, with each level providing more detailed insights into the system and its data. These levels, typically labeled from 0 to 2, progressively show more complex details about the system's parts and processes.

4.1 DFD Level 0

A Data Flow Diagram (DFD) Level 0, also known as the context diagram, provides a high-level view of the Hospital Management System. It encapsulates all software requirements within a single bubble labeled "Hospital Management System," illustrating how information flows between external entities—namely the Patient, Doctors, and the Administrator—and the central system. Arrows depict inputs (such as personal details and appointment requests) and outputs (like appointment details and needed medical records), highlighting the primary data exchanges.

1. Administrator: Manages hospital departments and oversees system-wide functions, including departmental organization and high-level operational decisions.
2. Patient: Provides personal details, requests appointments, and receives appointment confirmations and other pertinent information from the system.
3. Doctors: Receive scheduled appointments and submit or access patient medical records through the system.

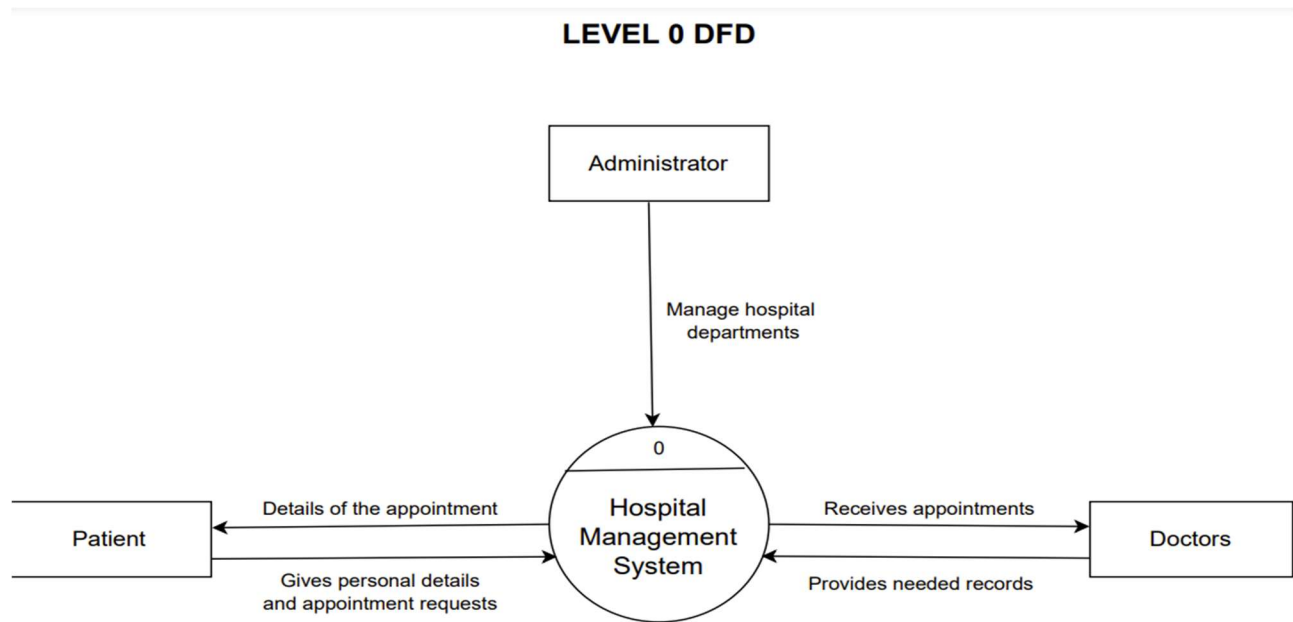


Figure 4.1 : DFD Level 0

4.2 DFD Level 1

Level 1 of the Data Flow Diagram (DFD) for the Hospital Management System provides a more detailed look at the internal processes that were introduced at Level 0. Each process or bubble in this diagram focuses on a specific aspect of the system, breaking down high-level functions into smaller, more manageable components. Below is an overview of the main processes and their corresponding data stores or tables:

1. Patient Management

- Manages the registration of new patients and updates existing patient information.
- Interacts with the **Patient Table** to store and retrieve details such as name, age, and contact information.

2. Patient Table

- Stores all patient records, serving as a central repository for patient-related data.
- Provides patient information to processes like **Schedule Appointment** and **Medical Record Maintain** as needed.

3. Schedule Appointment

- Oversees the scheduling process, matching patient requests with available doctors.
- Sends newly created appointment details to the **Appointment Table** for record-keeping.

4. Appointment Table

- Maintains a list of all appointments, including the patient ID, doctor ID, date, time, and reason for the visit.
- Data is fed into this table by **Schedule Appointment** and can be accessed by other processes like **Data Management**

5. Medical Record Maintain

- Handles the creation, update, and retrieval of patient medical records.
- Works closely with the **Medical Record** data store to store diagnoses, treatments, and other clinical details.

6. Doctors Table

- Contains comprehensive information about the hospital's doctors, including personal details, specialties, and contact information.
- Shares relevant data with **Data Management** and helps in scheduling via **Schedule Appointment**.

7. Medical Record

- Acts as the central repository for patient medical records.
- Updated and accessed by **Medical Record Maintain** to ensure accurate tracking of diagnoses, treatments, and follow-up data.

8. Data Management

- Coordinates the flow of information related to doctors and appointments.
- May retrieve or update data in **Doctors Table** and **Appointment Table** to keep schedules and records consistent.

9. Department Table

- Stores information about various hospital departments, including department name and floor.
- Interacts with **Department Management** to keep department data current.

10. Department Management

- Manages the creation, update, and organization of departments within the hospital.
- Updates the **Department Table** with new or modified department information.

11. Admin

- Represents the administrative user or entity responsible for high-level tasks.
- Oversees departmental organization, manages user roles, and ensures that the entire system operates smoothly.

Through these processes and data stores, the Level 1 DFD reveals how patient registration, appointment scheduling, medical record maintenance, and departmental management function in greater detail. It also shows how data flows among patients, doctors, and administrative staff, forming a clear blueprint for the system's operational structure.

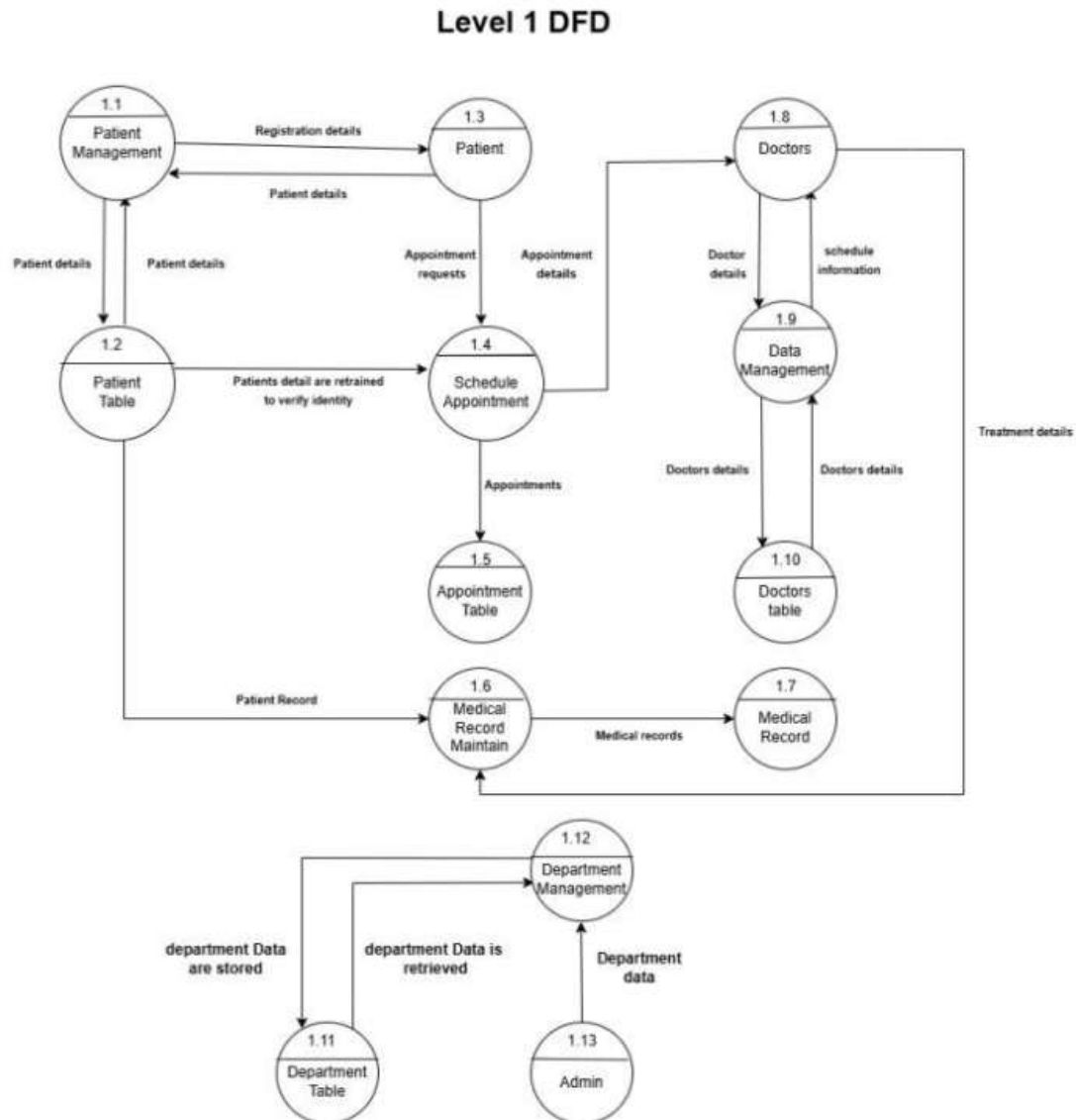


Figure 4.2 : DFD Level 1

5. Relational Schema

Relation schema defines the design and structure of the relation like it consists of the relation name, set of attributes/field names/column names. Every attribute would have an associated domain.

Relational Schema

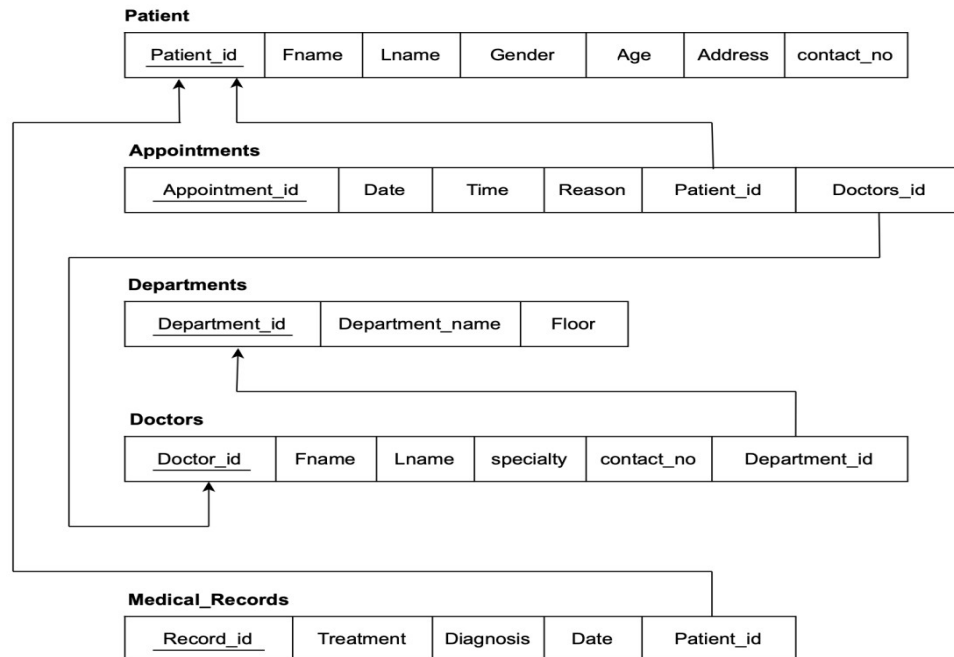


Figure 5.1 : Relational Model

Normalization and description

Normalization is the process of organizing data in a database. It includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Normalization works through a series of stages called normal forms.

1st Normal Form (1NF)

Normalization begins with ensuring that each table in the database contains only atomic (indivisible) values, with no repeating groups or arrays. In our Hospital Management System, each attribute is designed to store a single value. For example, in the Patients table, attributes such as FirstName, LastName, Age, Gender, ContactNumber, and Address each hold one specific piece of data. This approach prevents the storage of multiple values in a single column and maintains data

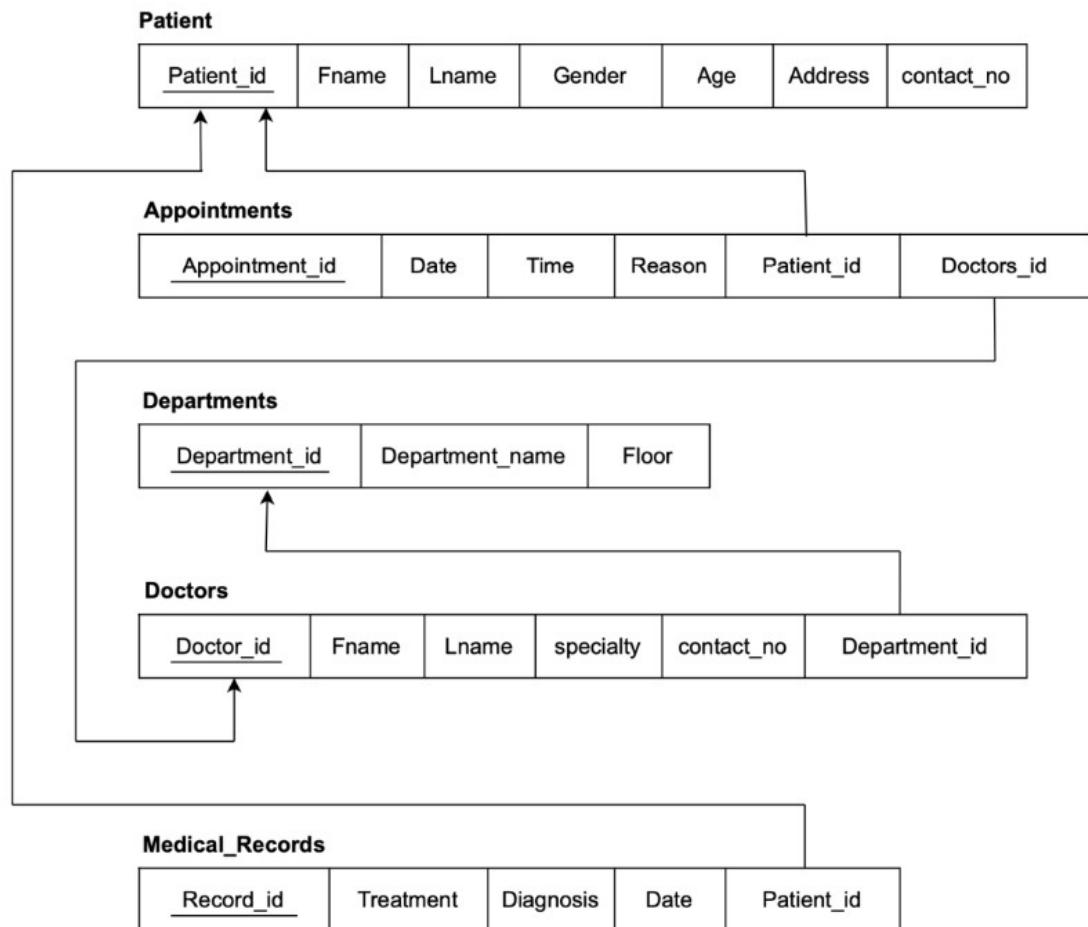
consistency. As every table in the system—whether it's Patients, Doctors, Departments, Appointments, or MedicalRecords—stores data in an atomic fashion and each record is unique, the schema naturally satisfies the requirements of 1NF.

2nd Normal Form (2NF)

Once 1NF is achieved, the next step is to ensure that the tables are in 2NF. A table meets 2NF if it is already in 1NF and all non-key attributes are fully functionally dependent on the entire primary key. In systems with composite primary keys, partial dependencies can occur when an attribute depends only on part of the key. However, in our Hospital Management System, every table uses a single-column primary key (for example, PatientID in Patients, DoctorID in Doctors, and AppointmentID in Appointments). Because these primary keys are singular and not composite, every non-key attribute inherently depends on the whole primary key. There is no possibility for a partial dependency since there is no subset of the primary key to depend upon. Thus, the schema satisfies 2NF without requiring any further decomposition or splitting of tables.

3rd Normal Form (3NF)

The final stage in our normalization process is achieving 3NF. A table is in 3NF if it is already in 2NF and every non-key attribute is directly dependent on the primary key, eliminating any transitive dependencies. Transitive dependencies occur when a non-key attribute depends on another non-key attribute rather than depending solely on the primary key. In our Hospital Management System, all non-key attributes in each table depend directly on their respective primary keys. For instance, in the Patients table, FirstName, LastName, and ContactNumber depend solely on PatientID and not on any other non-key attribute. This design holds true across all tables (Doctors, Departments, Appointments, MedicalRecords, etc.), ensuring that there are no indirect relationships that could lead to data redundancy or anomalies. Consequently, our schema meets the criteria for 3NF.



6. CONCLUSION

The Hospital Management System (HMS) is a comprehensive, integrated software solution designed to streamline the management of hospital processes. It automates key administrative tasks such as patient registration, appointment scheduling, medical record management, and department coordination, while also providing real-time access to critical hospital information. The system utilizes a role-based access model to ensure that patients, doctors, and administrators have access to relevant functionalities. Powered by large language models (LLMs), the system also features an AI-based chatbot, reducing manual workload by enabling natural language-based database queries and ensuring efficient task automation across hospital operations. Data is securely stored in a relational database system, with encryption and authentication mechanisms implemented to protect sensitive and confidential medical data. The system is designed with scalability and flexibility in mind, supporting future integrations including voice recognition systems and advanced data analytics for improved healthcare decision-making. Additionally, the frontend interface is designed using Streamlit, providing a user-friendly environment that ensures accessibility for all users, regardless of technical expertise. By reducing administrative bottlenecks and enhancing data retrieval.

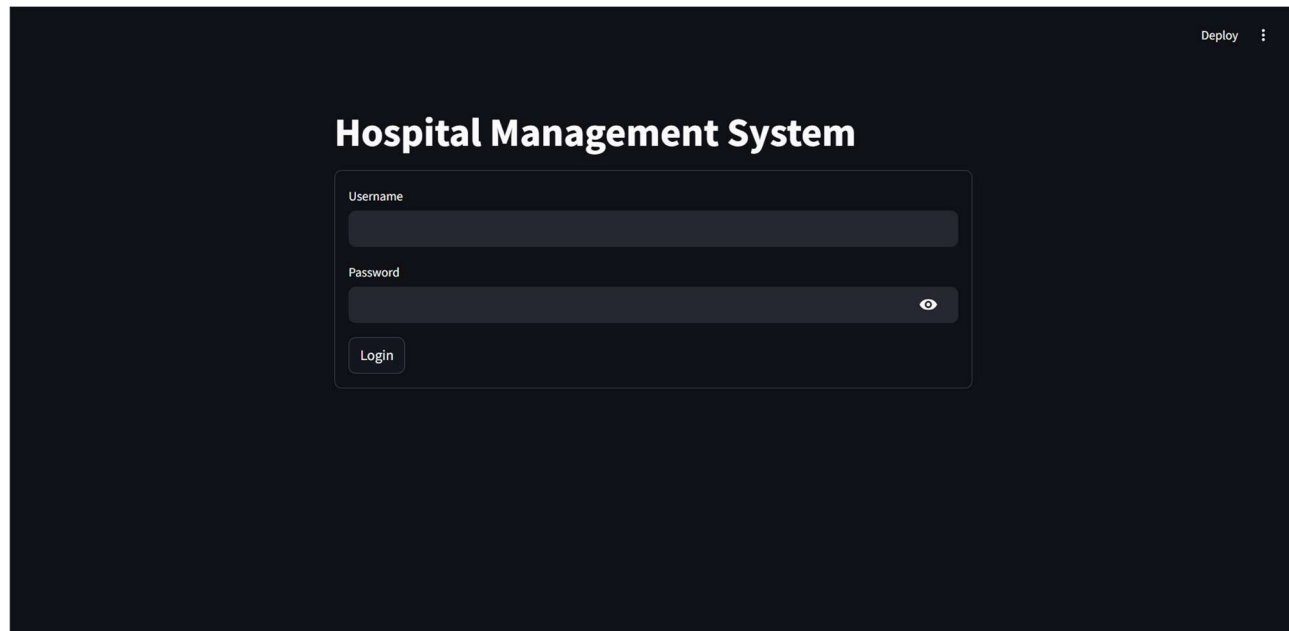
Hospital Management System is an integrated healthcare management solution designed to streamline patient-doctor interactions and optimize hospital operations. Built using Python and Streamlit, the system provides a user-friendly interface that allows patients to register, book appointments, and access medical records with ease. Simultaneously, doctors can manage their schedules, update patient histories, and maintain clinical records efficiently. The system leverages advanced database management concepts such as entity-relationship modeling, normalization, and secure transaction handling, ensuring reliable, consistent, and scalable data storage.

An AI-powered chatbot, utilizing large language model technology, facilitates natural language queries, automating routine database interactions and reducing manual workload. This intelligent interface enhances operational efficiency by providing real-time assistance and streamlining administrative tasks across the hospital. Role-based access control, coupled with robust encryption and authentication mechanisms, safeguards sensitive information and upholds data integrity. Overall, the Hospital Management System demonstrates a practical application of modern software engineering and database principles to deliver a secure, efficient, and impactful solution in the realm of healthcare management.

7. References

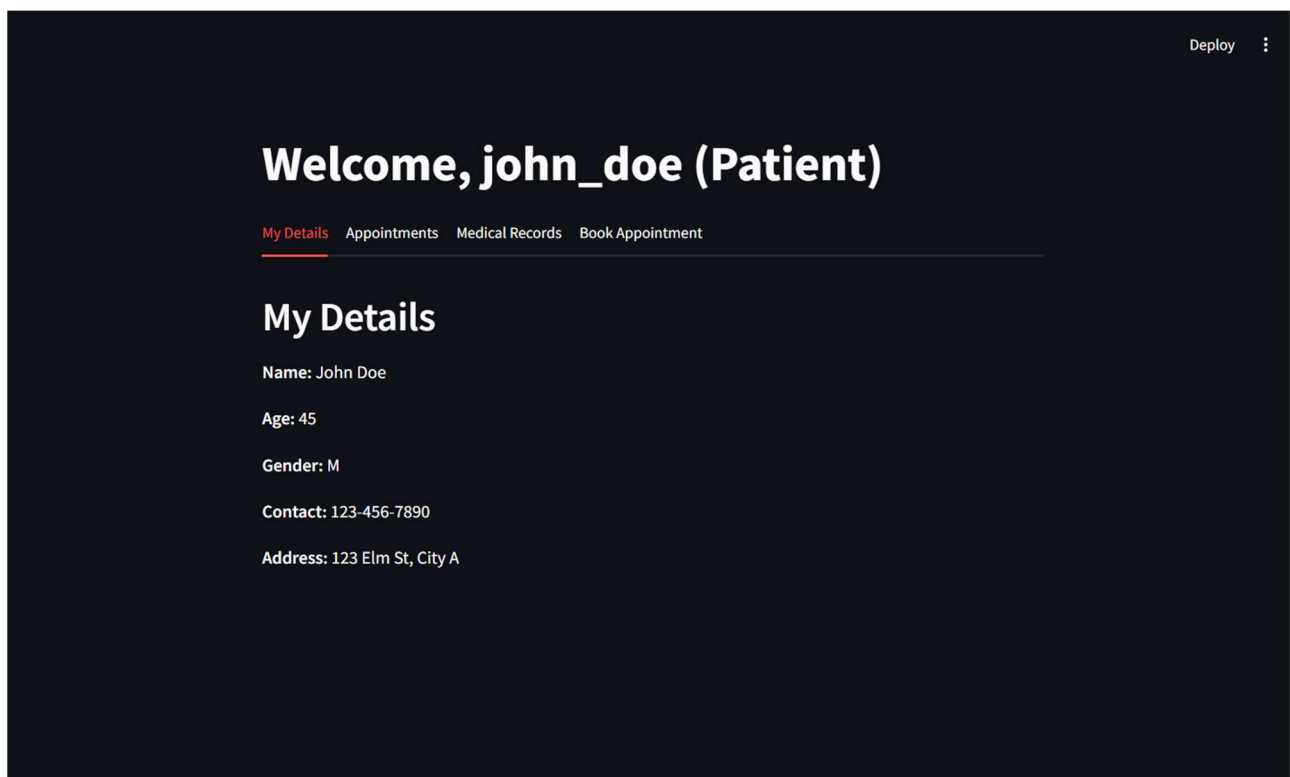
- [1] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 7th ed. Pearson, 2016.
- [2] J. Diederich, J. Milton, and P. B. Mitra, "Database principles and design for hospital information systems," *Int. J. Med. Inform.*, vol. 45, no. 1-2, pp. 43-60, 1997.
- [3] R. Agarwal and S. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, 1994, pp. 487-499.
- [4] MySQL Documentation, "MySQL 8.0 Reference Manual," [Online]. . [Accessed: Feb. 20, 2025].
- [5] FastAPI Documentation, "FastAPI," [Online]. Available: <https://fastapi.tiangolo.com/>. [Accessed: Feb. 20, 2025].
- [6] TensorFlow Documentation, "TensorFlow," [Online]. [Accessed: Feb. 20, 2025].
- [7] World Health Organization, "Digital Health Guidelines," 2021. [Online].. [Accessed: Feb. 20, 2025].
- [8] Healthcare Information and Management Systems Society (HIMSS), "Healthcare IT Trends," 2023. [Online]. Available: <https://www.himss.org/resources/>. [Accessed: Feb. 20, 2025].

8. Appendix



The screenshot shows a dark-themed login interface for the Hospital Management System. At the top right, there is a 'Deploy' button and a menu icon. The main heading 'Hospital Management System' is centered. Below it is a login form with two input fields: 'Username' and 'Password'. The 'Password' field has a toggle icon (an eye) to the right. A 'Login' button is positioned below the password field.

Fig 8.1 User Login



The screenshot displays the patient's home page for 'john_doe'. At the top right, there is a 'Deploy' button and a menu icon. The main heading 'Welcome, john_doe (Patient)' is centered. Below it is a navigation bar with four links: 'My Details' (highlighted with a red underline), 'Appointments', 'Medical Records', and 'Book Appointment'. The 'My Details' section is expanded, showing the following information: Name: John Doe, Age: 45, Gender: M, Contact: 123-456-7890, and Address: 123 Elm St, City A.

Fig 8.2 patients home page

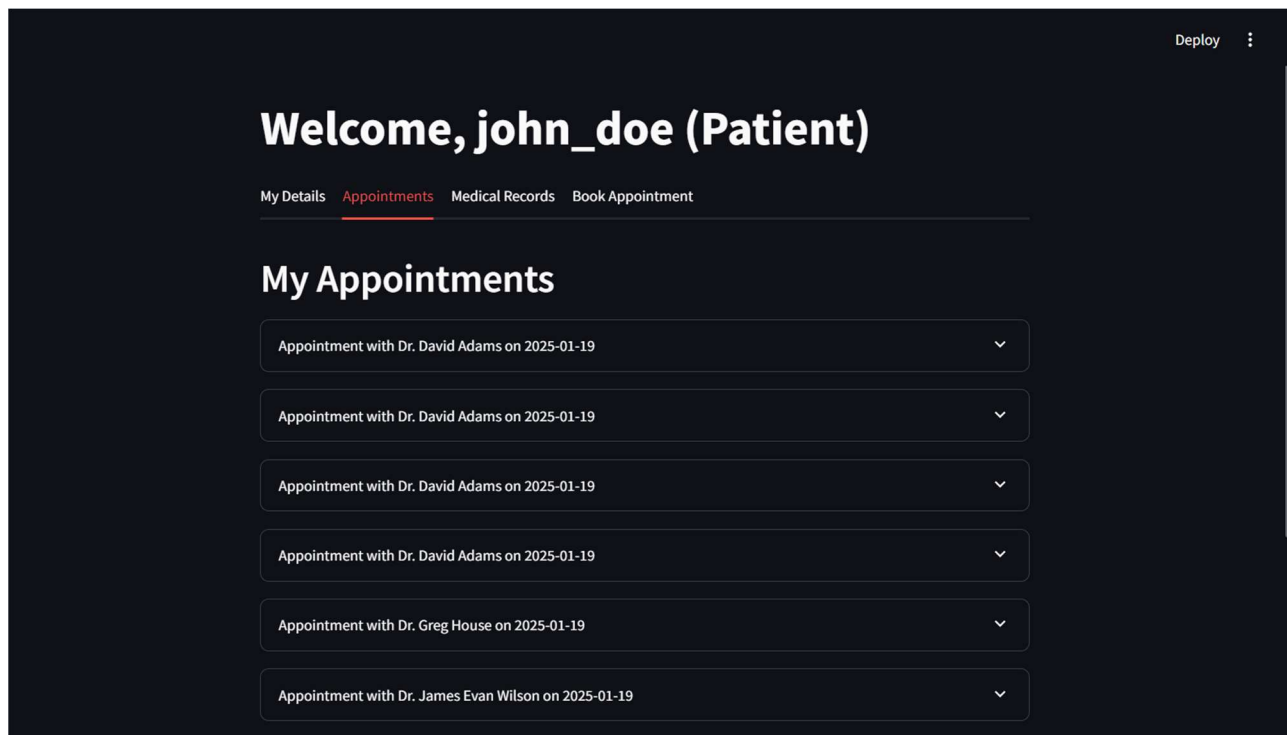


Fig 8.3 patients appointments page

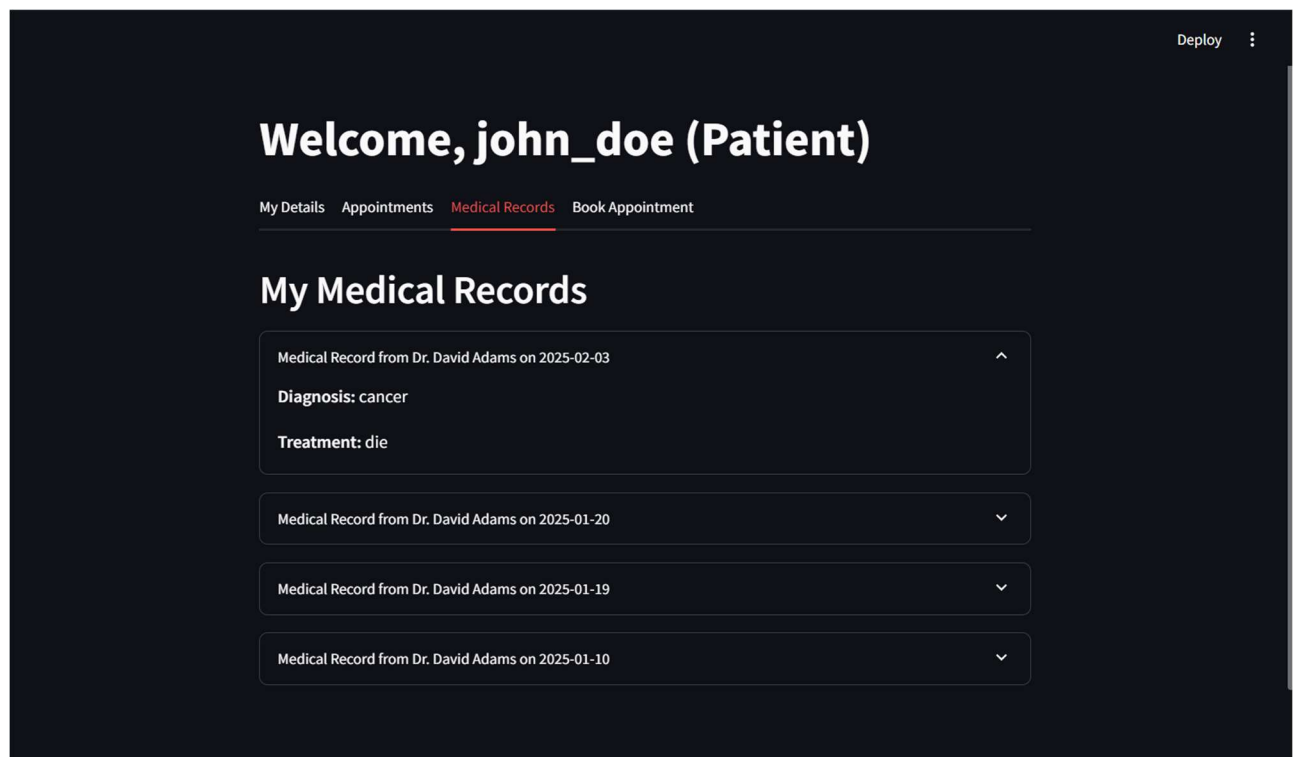


Fig 8.4 patient medical records

The screenshot shows a web interface for a patient named John Doe. The header includes a 'Deploy' button and a menu icon. The main heading is 'Welcome, john_doe (Patient)'. Below it is a navigation bar with links: 'My Details', 'Appointments', 'Medical Records', and 'Book Appointment' (which is highlighted in red). The main section is titled 'Book New Appointment'. It contains a form with the following fields: 'Select Doctor' (a dropdown menu showing 'Dr. David Adams (Cardiology)'), 'Date' (a text input showing '2025/02/20'), 'Time' (a dropdown menu showing '01:56'), and 'Reason for Appointment' (a large text area). At the bottom of the form is a 'Book Appointment' button.

Fig 8.5 patient Appointments booking

The screenshot shows a web interface for a doctor named David Adams. The header includes a 'Deploy' button and a menu icon. The main heading is 'Welcome, david_adams (Doctor)'. Below it is a navigation bar with links: 'My Details', 'Appointments' (which is highlighted in red), and 'Add Medical Record'. The main section is titled 'My Appointments'. It displays a list of appointments. Each appointment entry shows the date '2025-01-19', the time '0:41:00', and the reason 'dv'. The appointments are listed in a table-like structure with expandable/collapsible icons (upward and downward arrows) next to each entry.

Fig 8.6 Doctors Appointments

The screenshot shows a dark-themed web application interface. At the top right, there is a 'Deploy' button and a menu icon. The main heading is 'Welcome, david_adams (Doctor)'. Below this is a navigation bar with three links: 'My Details', 'Appointments', and 'Add Medical Record', which is currently selected and underlined. The main section is titled 'Add Medical Record'. It contains a form with three fields: 'Select Patient' with a dropdown menu showing 'John Doe', 'Diagnosis' with a large text area, and 'Treatment' with a large text area. Each text area has a small icon in the bottom right corner.

Deploy

Welcome, david_adams (Doctor)

My Details Appointments Add Medical Record

Add Medical Record

Select Patient

John Doe

Diagnosis

Treatment

Fig 8.7 Doctors medical records

Deploy ⋮

Welcome, admin (Admin)

Admin Chatbot

Enter your query:

list all the patients

Submit

Generated SQL Query:

```
SELECT * FROM Patients;
```

Query Results:

	0	1	2	3	4	5	6
0	1	John	Doe	45	M	123-456-7890	123 Elm St, City A
1	2	Jane	Smith	30	F	123-456-7891	456 Maple St, City B
2	3	Alice	Green	35	F	123-456-7892	789 Oak St, City C

Fig 8.8 Admin Page