Ghulam Ishaq Khan Institute of Engineering Sciences and Technology

Faculty of Computer Science and Engineering

Hospital Management System

A Database Management System for Efficient Medical Facilities and Resource Management

Submitted By:

Syed Murtaza Salman 2023694 BS Data Science

Muhammad Ashir Siddiqui 2023385 BS Data Science

Salman Shahzad Ali 2023504 BS Data Science

Under the guidance of:

Lecturer Abinta Mehmood Mir

Department of CS

Empowering knowledge management through technology

Table of Contents

1. Introduction
    1.1 Project Overview
    1.2 Objectives
2. Business Scenario Analysis
    2.1 Scenario Description
    2.2 Need for DBMS
3. Database Design
    3.1 Entity Relationship Diagram (ERD)
    3.2 Design Rationale
4. Relational Database Schema
5. Data Population
6. Querying and Reporting
    6.1 Query Examples
    6.2 Sample Outputs
7. User Interface
8. Implementation Details
9. Conclusion
10. References
11. Developers

1.1 Project Overview

This Hospital Management System is a full-stack web application built using Node.js, Express.js, PostgreSQL, and EJS templating. It is designed to efficiently manage hospital data, provide an intuitive interface for patients and staff, and streamline administrative tasks like appointment booking, medicine tracking, and doctor management.

1.2 Objective

The objective of this project is to develop a robust and user-friendly Hospital Management System that digitizes and automates key hospital operations such as user registration, appointment scheduling, doctor and medicine management, and administrative tasks. The system aims to provide a secure and efficient platform where patients can easily book appointments, view available doctors and medicines, and where admins can manage hospital data through a centralized interface. It leverages a PostgreSQL database for structured data storage, and utilizes Node.js with Express for backend logic, ensuring smooth communication between the server and frontend.

2.1 Scenario Description

In a hospital environment, multiple roles such as patients, doctors, and administrative staff interact with various systems for managing appointments, medicines, and personnel records. This project simulates such a real-world hospital scenario by providing an online platform where:

Patients can register themselves, log in securely, view a list of doctors, and book appointments based on availability.

Doctors' data is preloaded by the admin and includes essential information such as specialization, experience, contact details, and profile images.

Admins have dedicated features to manage the system, including adding or deleting doctors, managing medicine stock, and viewing or modifying appointment records.

Medicines are maintained with key details like stock quantity, expiry date, and price, helping simulate an inventory system within the hospital.

The system ensures secure authentication, proper data validation, and separation of concerns for different user roles, simulating a typical hospital's digital operations.

2.2 Need For Database

A DBMS is essential for securely storing and managing hospital data like patients, doctors, appointments, and medicines. It helps prevent data redundancy, ensures accuracy, and allows fast and reliable access to information. In this project, PostgreSQL is used to maintain data integrity, enforce constraints, and handle complex relationships between different entities efficiently.

3. Database Design

3.1 Entity Relationship Diagram (ERD)

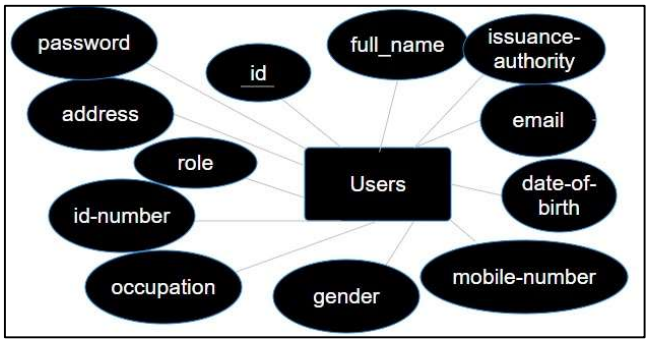
The Entity Relationship Diagram (ERD) for the Hospital Management System represents the core entities, their attributes, and relationships. The main entities are:

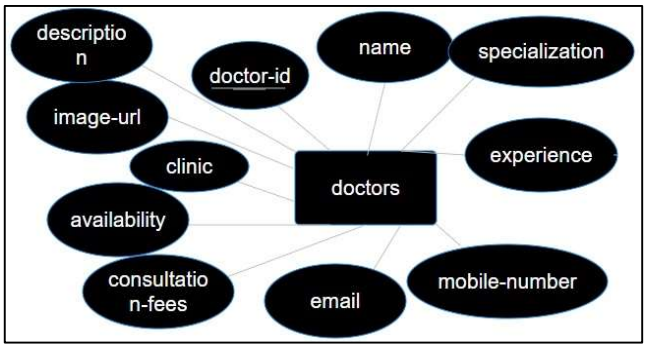
- Users: Stores user details (ID, Full Name, Email, DoB, Mobile Number, Gender, Occupation, ID Number, Issuance Authority, Role, Address, Password).
- Doctors: Stores doctor details (Doctor ID, Name, Specialization, Experience, Mobile Number, Email, Consultation Fees, Availability, Clinic, Image URL, Description).
- Medicines: Keeps record of medicines (Medicine ID, Image, Name, Price, Stock Quantity, Expiry Data).
- Appointment: Stores appointments (Appointment ID, Patient Name, Patient Email, Doctor Name, Doctor Email, Appointment Date, Created At).
- Admin: Stores admin details (ID, Full Name, Email, DoB, Mobile Number, Gender, Occupation, ID Number, Issuance Authority, Role, Address, Password).

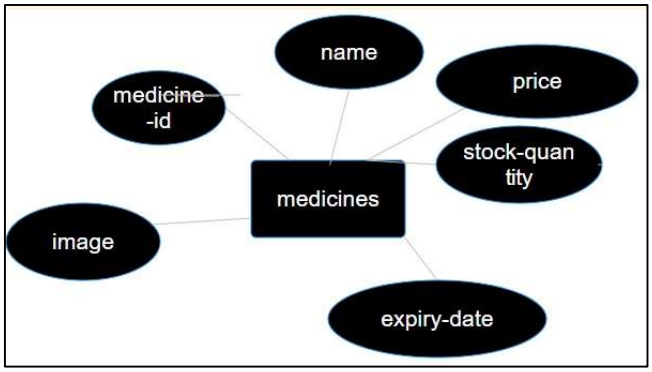
Relationships:

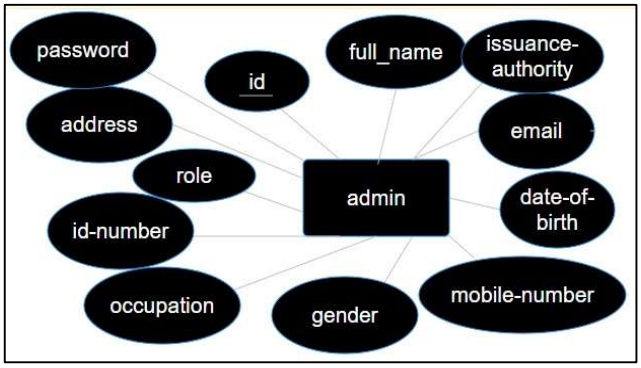
- A User can book multiple Appointments
- A Doctor can have multiple Appointments.
- Admins manage records of Doctors and Medicines.
- The system maintains a list of Medicines for hospital inventory purposes.

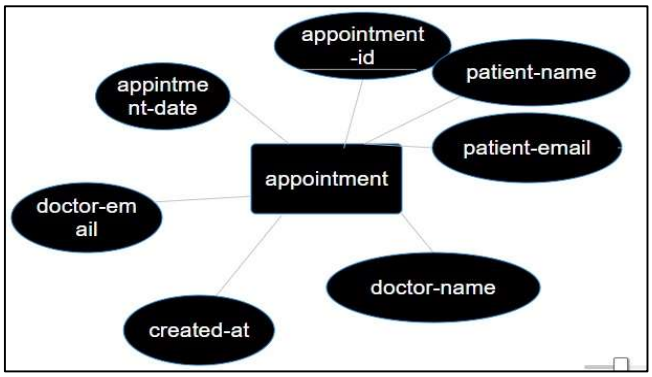
ERD Description











4. Relational Database Schema

Users	
Id	Integer
Full_name	Varchar(55)
Email	Varchar(55)
Date_of_birth	Date
Mobile_number	Varchar(11)
Gender	Varchar(10)
Occupation	Varchar(100)
Id_number	Varchar(13)
Issuance_authority	Varchar(255)
Role	Varchar(55)
Address	Text
Password	Varchar(255)

Doctors	
Doctor_id	Integer
Name	Varchar(55)
Specialization	Varchar(55)
Experience	Integer
Mobile_number	Varchar(15)
Email	Varchar(55)
Consulatation_fees	Numeric(10,2)
Availability	Text
Clinic	Varchar(55)
Image_url	Text
Description	Text

Medicines	
Medicine_id	Integer
Image	Varchar(55)
Name	Varchar(55)
Price	Integer
Stock_quantity	Integer
Expiry_date	date

Admin	
Id	Integer
Full_name	Varchar(55)
Email	Varchar(55)
Date_of_birth	Date
Mobile_number	Varchar(11)
Gender	Varchar(10)
Occupation	Varchar(100)
Id_number	Varchar(13)
Issuance_authority	Varchar(255)
Role	Varchar(55)
Address	Text
Password	Varchar(255)

Appointment	
Appointment_id	Integer
Patient_name	Varchar(55)
Patient_email	Varchar(55)
Doctor_name	Varchar(55)
Doctor_email	Varchar(55)
Appointment_date	Date
Created_at	date

5. Data Population

Overview:

The data population strategy in the healthcare application's backend initializes the PostgreSQL database with dummy data to support testing and development. Asynchronous functions populate the users, medicines, doctors, and appointment tables, ensuring no duplicates and maintaining data integrity.

Data Population Details:

Users Table: The insertDummyUsers function adds five users (e.g., John Doe, Patient; Dr. Michael Brown, Doctor; Admin User, Admin) with hashed passwords, unique emails, and valid roles, only if the table is empty.

Medicines Table: The insertdummymedicines function inserts 10 medications (e.g., Paracetamol, Amoxicillin) with unique names, positive prices, non-negative stock, and future expiry dates.

Doctors Table: The insertdummydoctors function adds 20 doctors (e.g., Dr. Alice Smith, Cardiologist; Dr. Emily Davis, Neurologist) with varied specializations, experience, and fees.

Appointment Table: The createtableappointment function creates the table without initial data, as appointments are added dynamically via user actions.

Implementation:

Execution: Functions run after a successful database connection, checking for existing records to avoid duplicates.

Security: Passwords are hashed using bcrypt.

Error Handling: Try-catch blocks log errors to prevent crashes.

Order: Tables are created and populated sequentially (users, medicines, doctors, appointment).

6. Query and Reporting

Overview:

The healthcare application's backend uses PostgreSQL and Express.js to execute SQL queries and render reports via EJS templates, supporting data retrieval and management for users and admins.

Key Queries:

Doctors: (/go-to-doctors, /go-to-adminview-doctors): SELECT * FROM doctors fetches all doctor records for user and admin views.

Medicines: (/go-to-medicines, /go-to-adminview-medicines): SELECT * FROM medicines retrieves medicine details for browsing and inventory management.

Appointments: (/go-to-adminview-appointments): SELECT * FROM appointment displays all appointments.

Users: (/update-users): SELECT * FROM users ORDER BY id ASC lists users for admin updates.

Delete Old Appointments: (/delete-old-appointments): DELETE FROM appointment WHERE appointment_date < '2025-05-02' removes outdated appointments.

Reporting Features:

Admin Views: Show doctors, medicines, and appointments for management.

User Views: Display doctors and medicines for browsing.

Update Interfaces: Allow admins to edit user, medicine, and doctor data.

Error Feedback: Renders error messages (e.g., "Invalid Doctor ID") for failed operations.

Security and Efficiency:

Parameterized Queries: Prevent SQL injection.

Session Checks: Restrict sensitive actions to authenticated users.

Optimized Queries: Use simple SELECT * with optional sorting for performance.

7. User Interface

Overview:

The GIKI Medical Centre's UI, built with EJS templates and styled with CSS, Bootstrap, and Font Awesome, offers a responsive frontend for hospital management. Integrated with an Express.js backend, it supports patients, admins, and receptionists through views for authentication, browsing, and record management.

Key UI Components:

Authentication:

Login (login.ejs): Bootstrap form for email/password with error feedback and links to register.

Registration (form.ejs): Multi-field form for user signup, validating inputs with error messages.

Dashboard:

Landing Page (landing_page.ejs): Bootstrap header with user name, image carousel, and icon cards linking to features (e.g., view doctors, admin tasks).

Browsing:

Doctors (doctors_ui.ejs): Card layout showing doctor details with a "Book Appointment" form.

Medicines (medicines_ui.ejs): Gallery of medicine cards with name, price, and "Buy Now" button.

Admin Views:

Doctors Admin (adminviewdoctors.ejs): Table of doctor details for management.

Medicines Admin (adminviewmedicines.ejs): Table for inventory tracking.

Add/Delete:

Add Medicine (addmedicine.ejs): Form for adding medicines.

Delete Doctor (deletedoctor.ejs): Form for removing doctors with error feedback.

8. Implementation Details

Frontend (HTML, CSS, JavaScript)

- HTML is used to define the structure of the web application, including pages for patient registration, appointment booking, and doctor listings.
- CSS provides styling for a clean and user-friendly interface. Layouts are designed to be responsive for use on both desktop and mobile devices.
- JavaScript handles client-side interactivity, such as:
    - Real-time form validation.

2. Backend (JavaScript with Node.js/Express)

- The backend is built using Node.js and the Express framework.
- It handles:
    - Routing for different HTTP requests like booking appointments, submitting patient forms, and retrieving doctor schedules.
    - Input validation and basic error handling.
    - Communication with the PostgreSQL database using SQL queries.

3. Database (PostgreSQL)

- A relational database structure is used to manage health center data.
- Queries are optimized for fetching available appointments, patient records, and doctor availability efficiently.

4. Authentication & Security (if applicable)

- The system may include login functionality for health staff using secure password hashing (e.g., bcrypt).

9. Conclusion

The GIKI Medical Centre application is a game-changer for hospital management, blending a solid PostgreSQL database with a slick Express.js backend and user-friendly EJS frontend. It makes life easier for patients, admins, and receptionists by handling everything from sign-ups and appointment bookings to managing doctors and medicines. The backend keeps data safe with encrypted passwords and secure queries, while the frontend offers clear, responsive pages that update on the fly. Preloaded dummy data gets the system up and running for testing, and smart reporting tools make tracking info a breeze. There's room to add cool upgrades like instant alerts or better search, but right now, this project nails it—delivering a practical, secure, and easy-to-use solution that brings tech and healthcare together for real people.

10. References

The online resources which helped us to accomplish this project are listed below:

Bootstrap. (2023). Bootstrap v5.3.3 documentation. (Free)

Code with Harry. (n.d.). Code with Harry YouTube channel. (Free)

EJS. (n.d.). Embedded JavaScript templates. (Free)

Express.js. (n.d.). Express.js documentation. (Free)

OpenAI. (n.d.). ChatGPT. Artificial Intelligence. (Free)

Perplexity AI. (n.d.). Perplexity AI documentation. Artificial Intelligence. (Free)

PostgreSQL. (n.d.). PostgreSQL documentation. Free database. (Free)

xAI. (n.d.). Grok. Artificial Intelligence. (Free)

Dr. Angela Yu, A. (2025). The complete 2023 web development bootcamp. (Paid)

Huge thanks to these resources, from killer tutorials to AI helpers, for guiding us through coding and designing our hospital system. They made the journey way smoother!

11. Developers

Developer 1: Syed Murtaza Salman, Student at GIKI,

GitHub: Murtaza436

Developer 2: Ashir Siddiqui, Student at GIKI,

GitHub: Ashirsiddiqui2004

Developer 3: Salman Shahzad Ali, Student at GIKI,

GitHub: SalmanShahzadAli

Link to GitHub repository:

https://github.com/SalmanShahzadAli/DBMS-Semester-Project.git