Hospital

Management System

Lecturer: Thear Sophal / Co-Lecturer: Ang Mengchhoung

Team members

Chhun Nika

Kon Sotheara

Ly Sreypich

Kim Solida

Content

1

Introduction

Problem

Manual and Disconnected Systems

Paper-based or separate systems lead to inefficiency and fragmented data

Data Loss and Errors

Manual handling increases the risk of mistakes and missing records

Slow and Disorganized Workflows

Tasks like registration and billing become delayed without integration

Poor Department Communication

Lack of centralized data causes coordination issues among staff

Solution

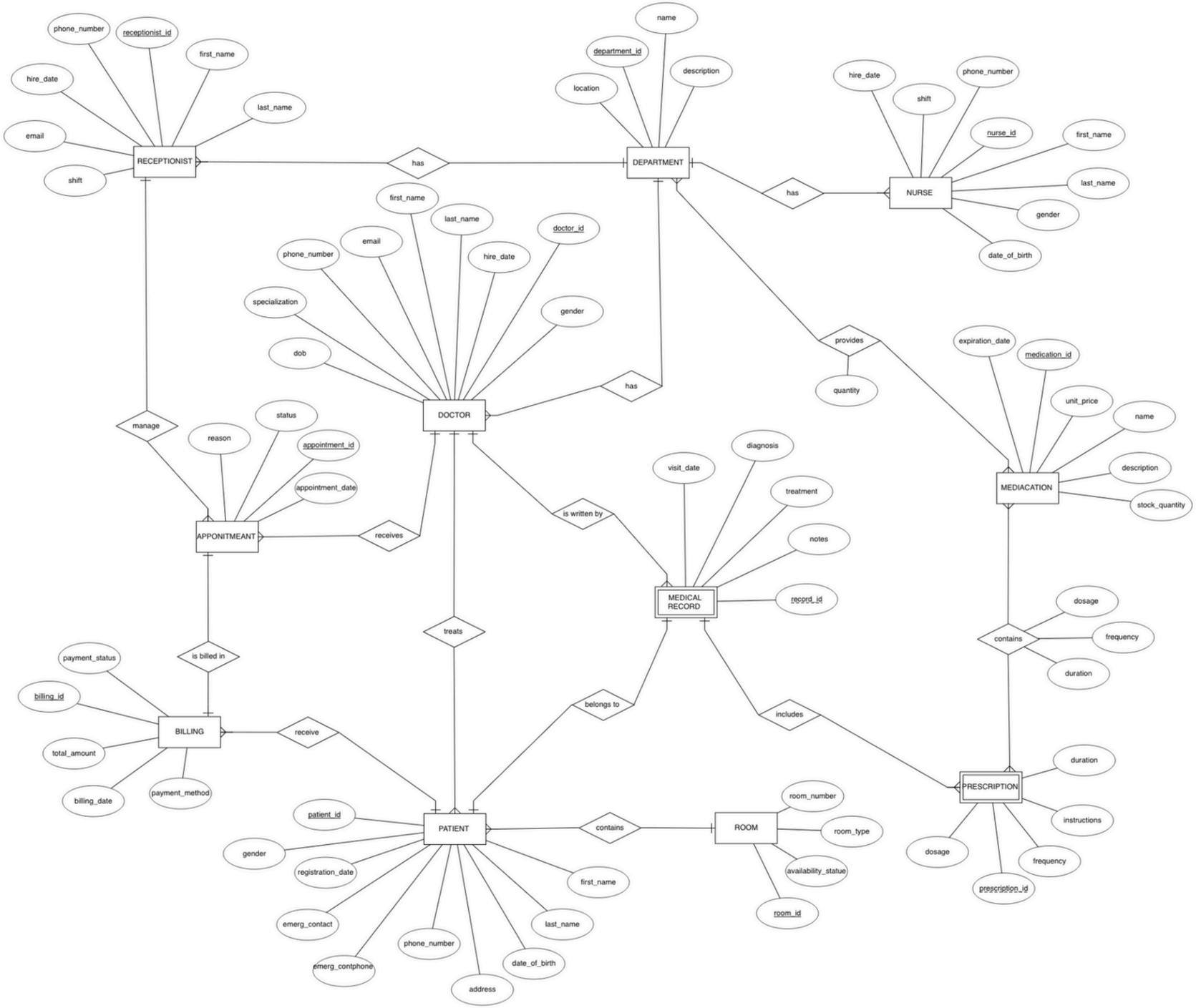
A Hospital Management System (HMS)

is software that helps hospitals manage their daily tasks in one place. It handles things like patient registration, appointments, medical records, billing, and staff management. HMS makes work faster, reduces errors, improves communication, and replaces manual paperwork with a digital system.

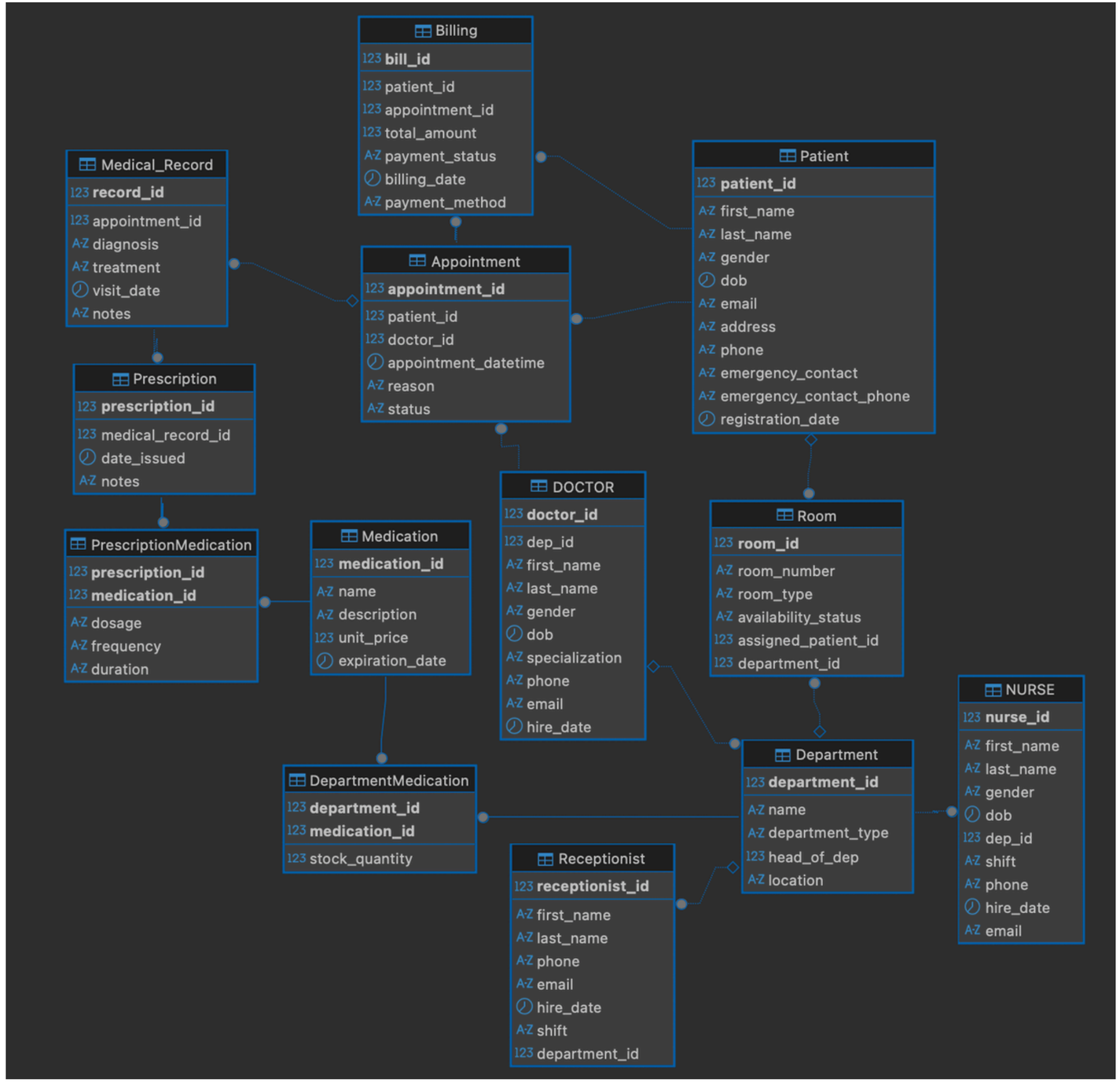
2

Database Design

ER Diagram


RECEPTIONIST
phone_number
receptionist_id
first_name
hire_date
last_name
email
shift
has
DEPARTMENT
name
department_id
description
location
has
NURSE
hire_date
phone_number
shift
nurse_id
first_name
last_name
gender
date_of_birth
DOCTOR
first_name
last_name
doctor_id
email
phone_number
hire_date
specialization
gender
dob
has
provides
quantity
manage
APPONITMEANT
status
reason
appointment_id
appointment_date
receives
is written by
MEDICAL RECORD
visit_date
diagnosis
treatment
notes
record_id
MEDIACATION
expiration_date
medication_id
unit_price
name
description
stock_quantity
treats
is billed in
BILLING
payment_status
billing_id
total_amount
billing_date
payment_method
receive
belongs to
includes
contains
dosage
frequency
duration
PATIENT
patient_id
gender
registration_date
emerg_contact
emerg_contphone
phone_number
address
date_of_birth
last_name
first_name
contains
ROOM
room_number
room_type
availability_statue
room_id
PRESCRIPTION
duration
instructions
dosage
frequency
prescription_id

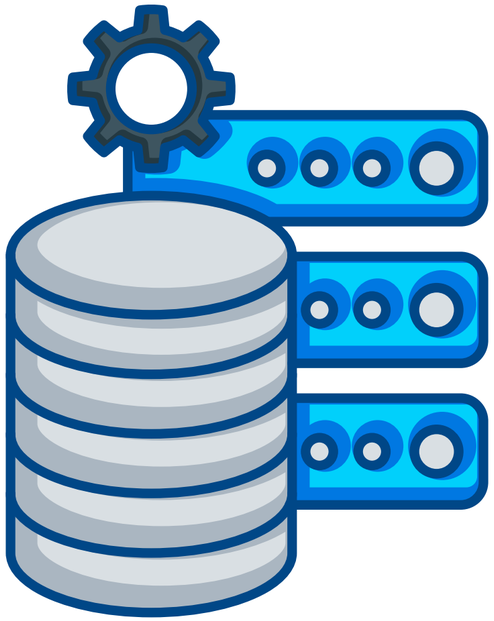
Relational Model



3

Database Implementation

Entities



Entities

we have 11 entities

Patient

The patient receives medical services and is the core of the system, linked to appointments, billing, and medical records.

Doctor

The doctor provides healthcare services and is linked to appointments and medical records.

Nurse

The nurse supports patient care and is assigned to specific departments and shifts.

Medication

The medication lists all drugs used in the hospital, including their quantity and price.

Room

The room tracks where patients stay, including type and availability.

Receptionist

The receptionist handles patient registration and appointment scheduling at the front desk.

Department

The department represents different medical units in the hospital, such as Pediatrics or Surgery.

Appointment

The Appointments table stores scheduled meetings with date/time, patient ID, and doctor ID.

Billing

Records financial transactions for patient treatments and visits.

Weak entities

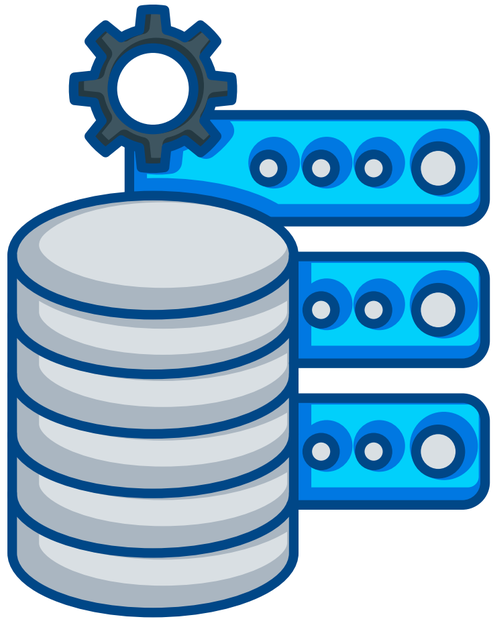
Medical Record

The medical record logs a patient’s visit, including diagnosis and treatment, and depends on the patient and doctor.

Prescription

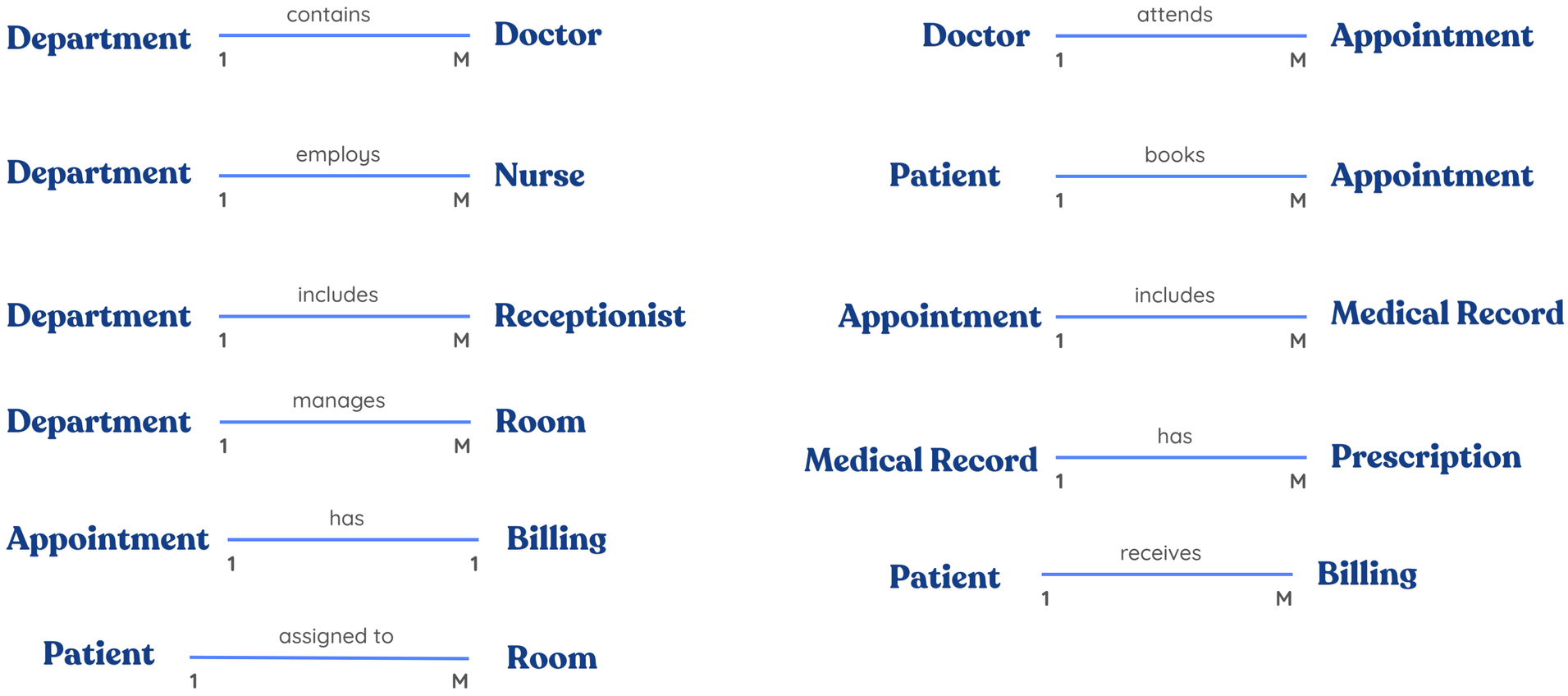
The prescription is tied to a medical record and contains medication details for the patient.

Relationships



Relationships

we have 14 relationships



Relationships

we have 14 relationships

Department — stocks (M — N) — **Medication**

Prescription — contains (M — N) — **Medication**

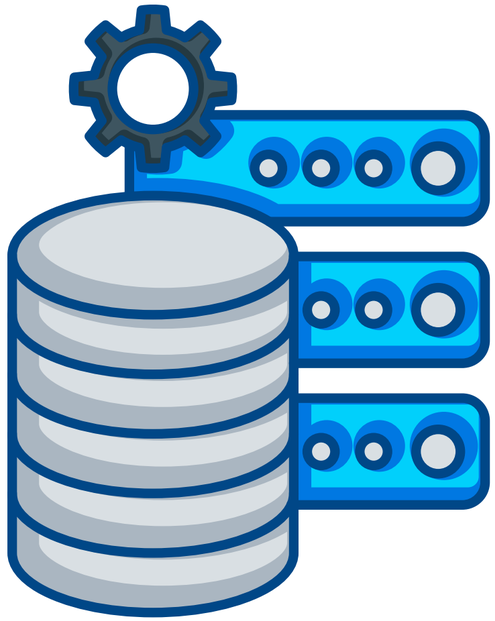
Department Medication

to track which medications are available in which departments along with stock quantity

Prescription Medication

to associate each prescription with multiple medications and include dosage, frequency, and duration

Queries



Join

Combining data from different tables

All Appointments Summary

Displays all appointments with patient and doctor full names, date/time, and current status.

Doctors by Department

Lists each doctor along with their associated department name.

rooms and the patient assigned

Shows room details including type, availability, and the name of the assigned patient.

Subqueries

Used within queries to perform multi-level filtering or logic.

Patients Who Visited Cardiologists

Lists patients who had appointments with doctors specializing in Cardiology.

Frequent Patients

Lists patients who had more than 3 appointments.

Departments with Multiple Doctors

Shows departments that have more than one doctor assigned.

Aggregations

Queries that use functions like SUM, COUNT, and GROUP BY to generate reports.

Total Billing Amount per Patient

Calculates the total billed amount for each patient.

Count of Medication Types per Department

Counts the distinct types of medications stored in each department.

Total Appointments per Doctor

Displays how many appointments each doctor has.

Function

Reusable SQL functions created to perform calculations and return specific values.

Calculate Age

Takes a patient’s date of birth and returns their current age in years.

Calculate Discounted Bill

Takes original billing amount and discount percentage, returns the final amount after discount.

Triggers

Event-based SQL logic to enforce business rules automatically when certain actions occur.

Prevent Negative Medication Stock

Prevents inserting records into DepartmentMedication with stock quantity less than 0.

Auto-Decrease Stock After Prescription

After inserting a new prescription, automatically decreases the corresponding medication’s stock in the assigned department.

Procedures

Procedures are created to input data for any tables inside database

Get Doctor's Appointments

Retrieves all appointments for a given doctor, ordered by date and time.

Add Medication and Stock

Inserts a new medication and initializes stock for a specific department.

4

Database Population

Tools

These are the tools that used to populate data

python™

The Faker Library

Programming language

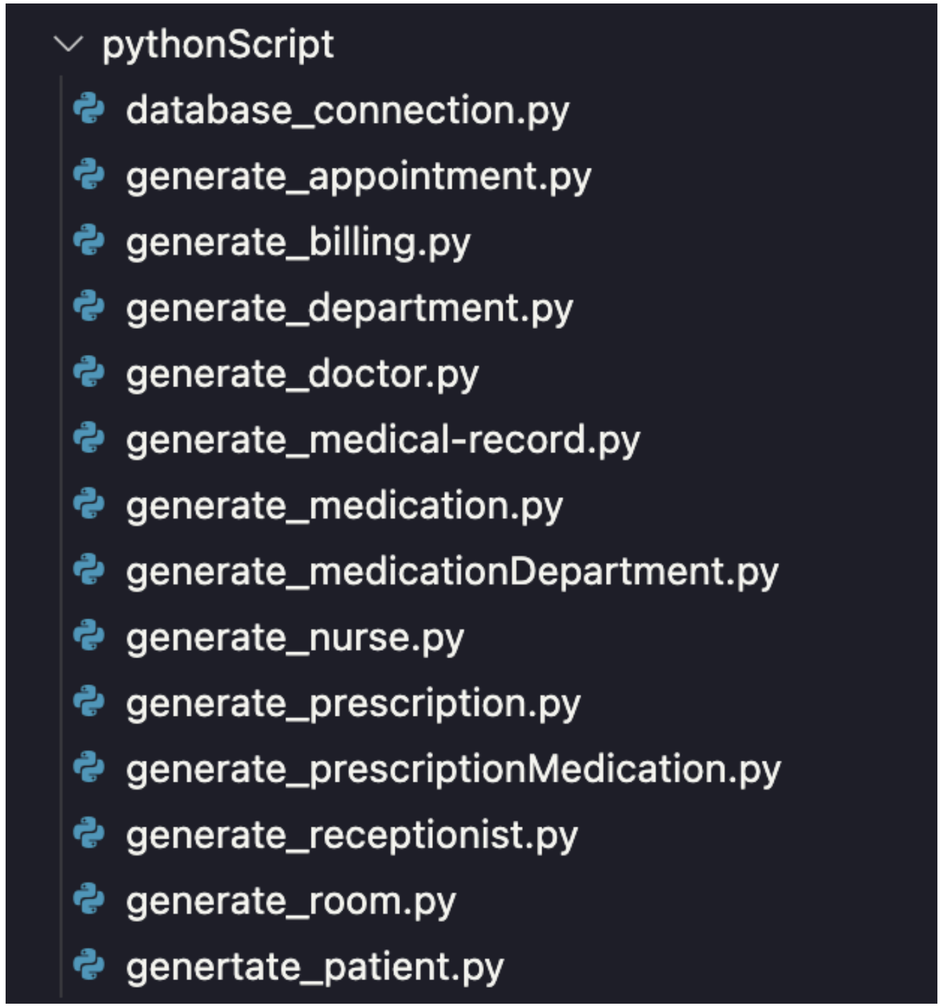
Used as the main scripting language to automate data insertion into the database and manage the data generation process.

Generate fake data

A Python library used to generate realistic fake data such as names, emails, phone numbers, and appointment dates for testing purposes

Files

We populate data by table



Number of data

Datas are generated based on real world scenario

Table name	Number of data
Appointment	6, 000, 000 records
Billing	6, 000, 000 records
Department	15 records
Doctor	3, 000 records
Medical_record	5, 000, 000 records
Medication	1, 000 records
DepartmentMedication	2, 043 records

Number of data

Datas are generated based on real world scenario

Table name	Number of data
Nurse	6, 000 records
Prescription	4, 000, 000 records
Prescription Medication	8, 002, 130 records
Receptionist	30 records
Room	200 records
Patient	3, 000, 000 records

Total: 32, 014, 418 records

5

User Management and Access Control

ROLE 01

Admin

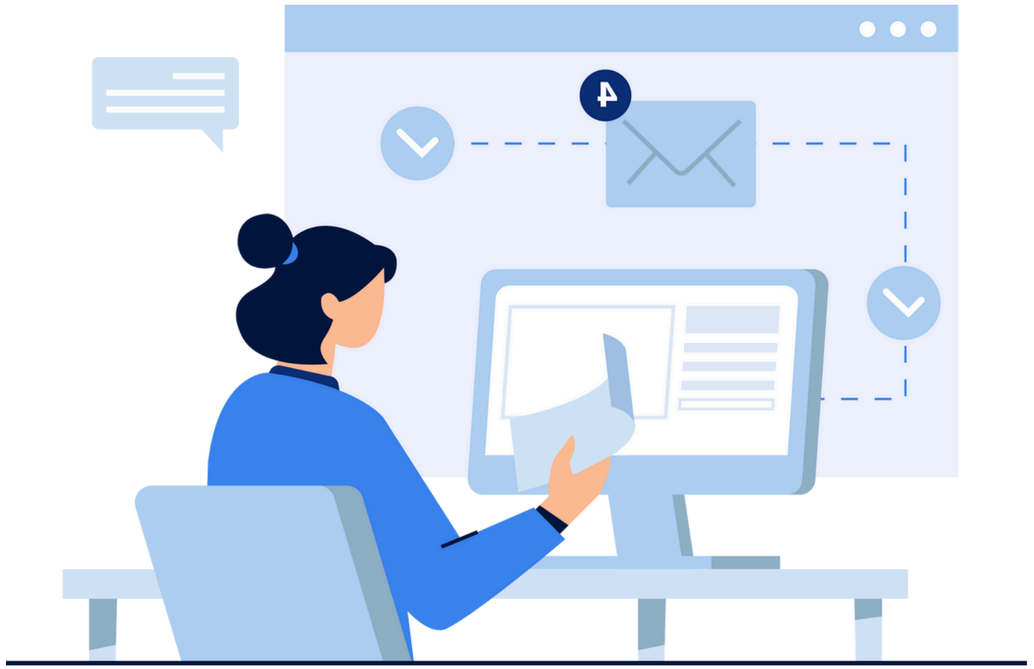
A general system administrator for the database application.

Privileges

ALL PRIVILEGES

Note

They have full access over database



ROLE 02

Data Analyst

A person who analyzes data to create reports, charts, or insights.

Privileges

SELECT

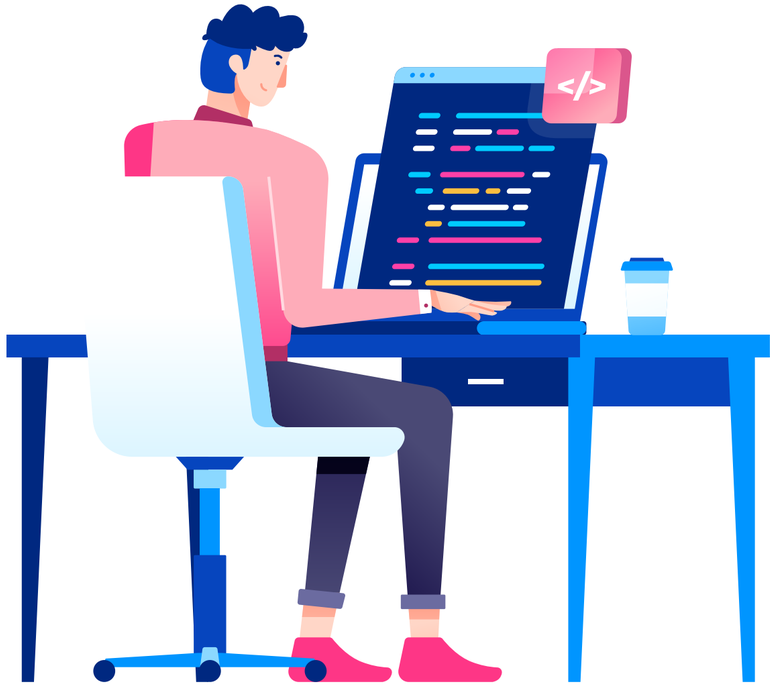
Note

They can only view and read the data

ROLE 03

Developer

Someone who writes code, builds features, and works on the application that uses the database.



Privileges

CREATE, READ, UPDATE, DELETE

Note

They can interact with database to build and test features during development phase

ROLE 04

DBA

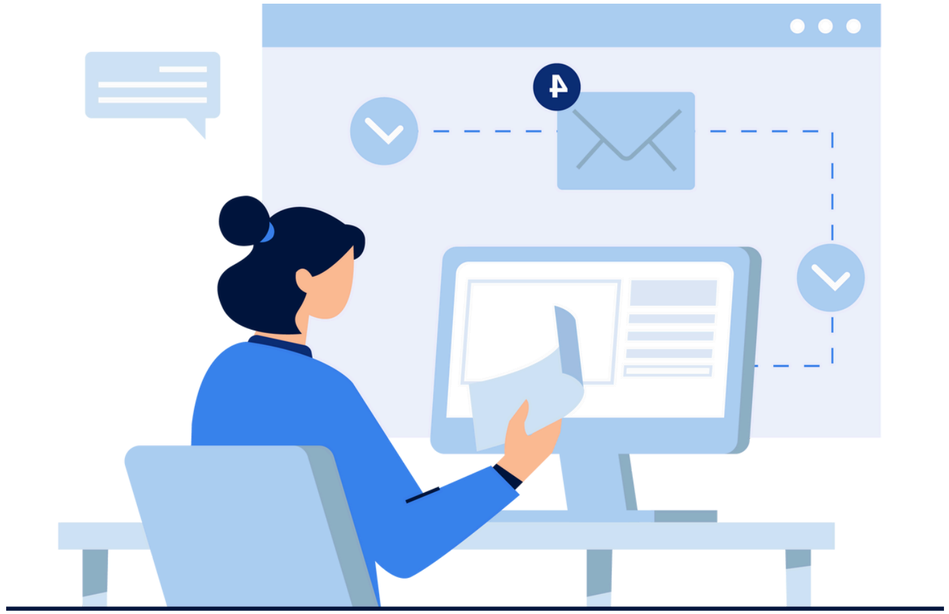
The highest-level authority over the database system. They manage security, performance, backups, users, etc.

Privileges

ALL PRIVELEGES

Note

Full access control (superuser)



ROLE 05

Auditor

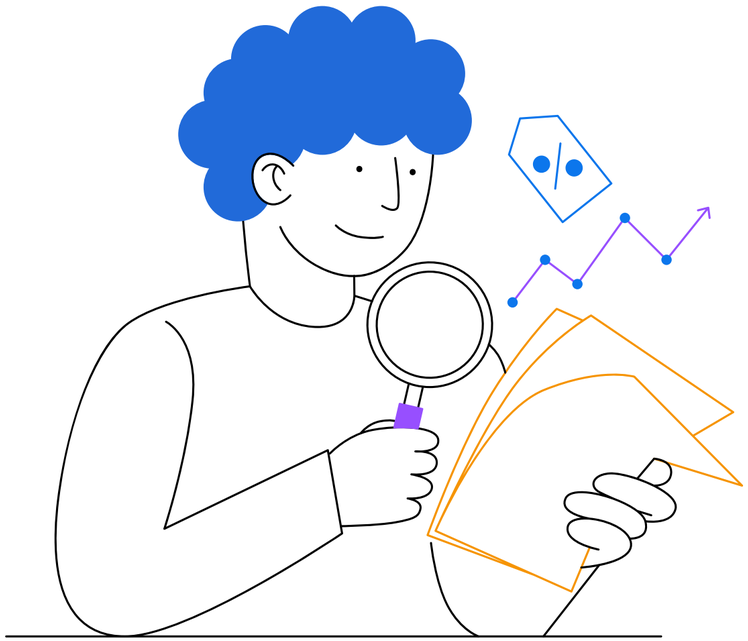
A person (internal or external) who reviews the system for compliance, security, or performance purposes.

Privileges

SELECT

Note

They can only view no modification access



ROLE 06

QA Support

Quality Assurance testers and customer support teams.

Privileges

SELECT, INSERT

Note

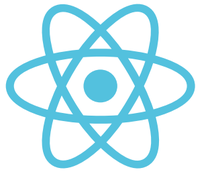
For testing and support purposes

RBAC Web-based

This web-based RBAC (Role-Based Access Control) system allows only administrators to log in and manage user access through role and privilege assignments. It ensures both application-level control and real MySQL user management.

Admin Capabilities	**View** dashboard summary of users, roles, and privileges
	View, edit, and **delete** user record
	Click **summary cards** for detailed role/privilege inf
	Add users and **assign** roles
	Automatically create **MySQL** users with appropriate privileges
	Perform database **backup** and **recovery**

Tools



React js

Frontend JavaScript library used to build dynamic user interfaces and manage component-based rendering.

Tailwind CSS

Utility-first CSS framework for fast and responsive UI styling.

Node.js & Express.js

Backend runtime and framework used to handle API routes, server logic, and database communication.

Tools

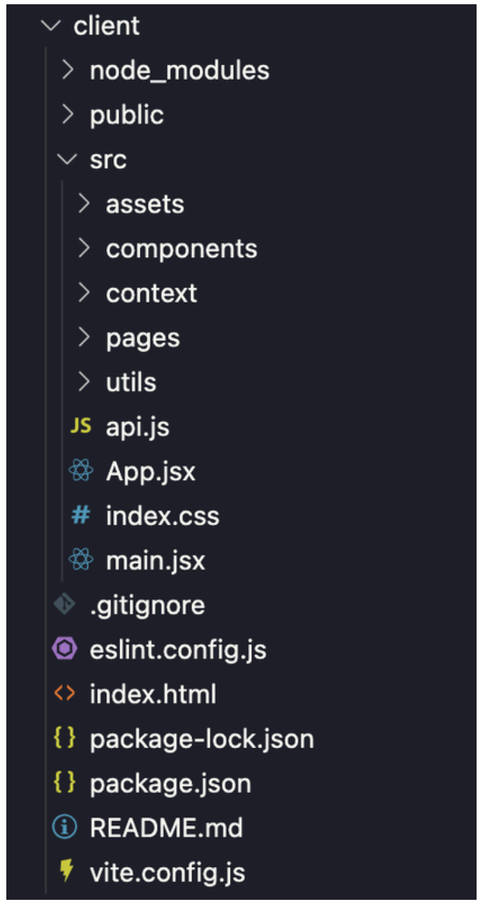
RESTful APIs

Interface that enables smooth communication between frontend and backend

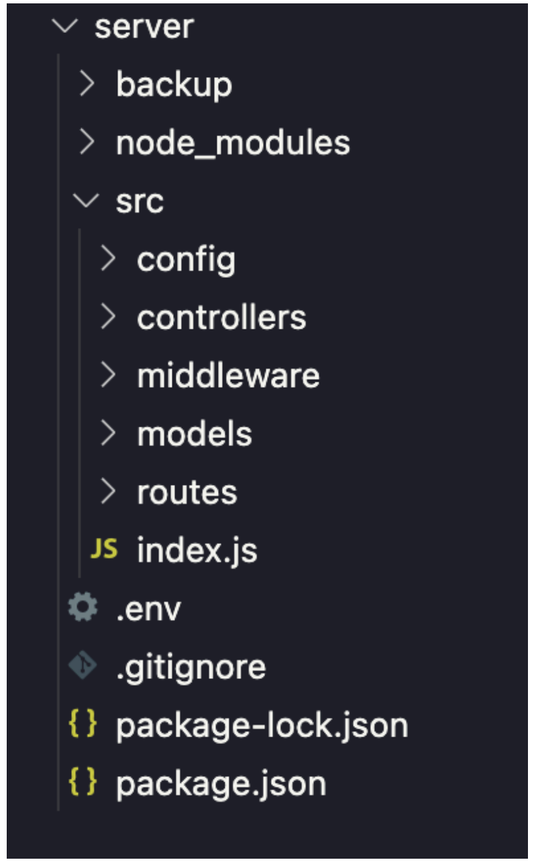
MySQL

Relational database system for storing users, roles, privileges, and access control data.

Architecture



Front end



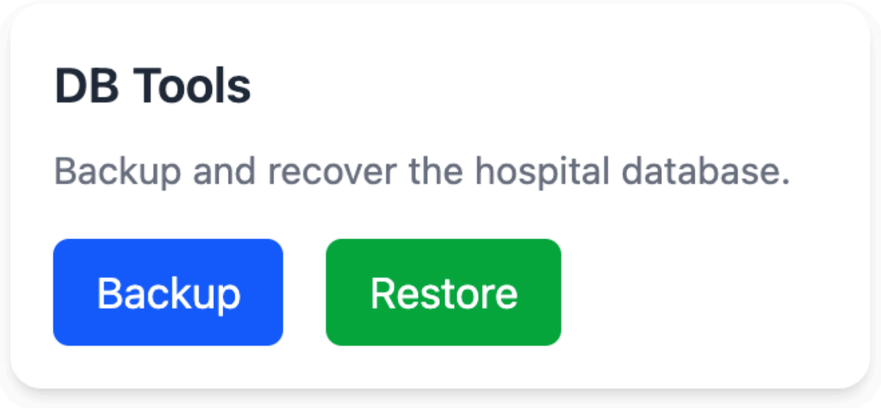
Back end

6

Backup and Recovery

RBAC Web-based

The back up and recovery process is integrated into the web-based system, enabling admins to manage backups and restores directly from the browser without using the command line.



Process

Backup Process

- Admin clicks Backup → backend runs mysqldump to export database.
- Backup saved as .sql file in server's backup folder.
- File named as databaseName_backup.sql.

Recovery Process

- Admin creates a new database manually in **MySQL** Workbench.
- Clicking Restore triggers backend to run mysql command and import the backup file.

Automation

- All commands executed via **JavaScript's exec** method.
- No manual command-line needed.
- Offers a user-friendly and secure way to manage backups through the web interface.

7

Query Development and Optimization

Indexing

Query: total billing amount per patient

```sql
SELECT
    b.patient_id,
    CONCAT(p.first_name, ' ', p.last_name) AS patient_name,
    p.email,
    p.phone,
    SUM(b.total_amount) AS total_billed
FROM Billing b
JOIN Patient p ON b.patient_id = p.patient_id
GROUP BY b.patient_id, p.first_name, p.last_name, p.email, p.phone
```

Create indexing on Billling table

```sql
CREATE INDEX idx_billing_patient_total
ON Billing(patient_id, total_amount);
```

Result:

Without indexing

3m 6.2s

With indexing

2m 33.1s

Indexing

Query: List all appointments with patient and doctor full names and their status

```sql
SELECT
    a.appointment_id,
    CONCAT(p.first_name, ' ', p.last_name) AS patient_name,
    CONCAT(d.first_name, ' ', d.last_name) AS doctor_name,
    a.appointment_datetime,
    a.status
FROM Appointment a
JOIN Patient p ON a.patient_id = p.patient_id
JOIN Doctor d ON a.doctor_id = d.doctor_id;
```

Create indexing on Appointment table

```sql
CREATE INDEX idx_appointment_patient_doctor
ON Appointment(patient_id, doctor_id);
```

Result:

Without indexing

18m 9.6s

With indexing

8m 55.2s

Indexing

Query: total billing discount for patient

```
SELECT
    bill_id,
    total_amount,
    calculate_discounted_bill(total_amount, 10) AS discounted_total
FROM Billing;
```

Create indexing on Billling table

```
CREATE INDEX idx_billing_total_amount
ON Billing(total_amount);
```

Result:

Without indexing

4m 42.7s

With indexing

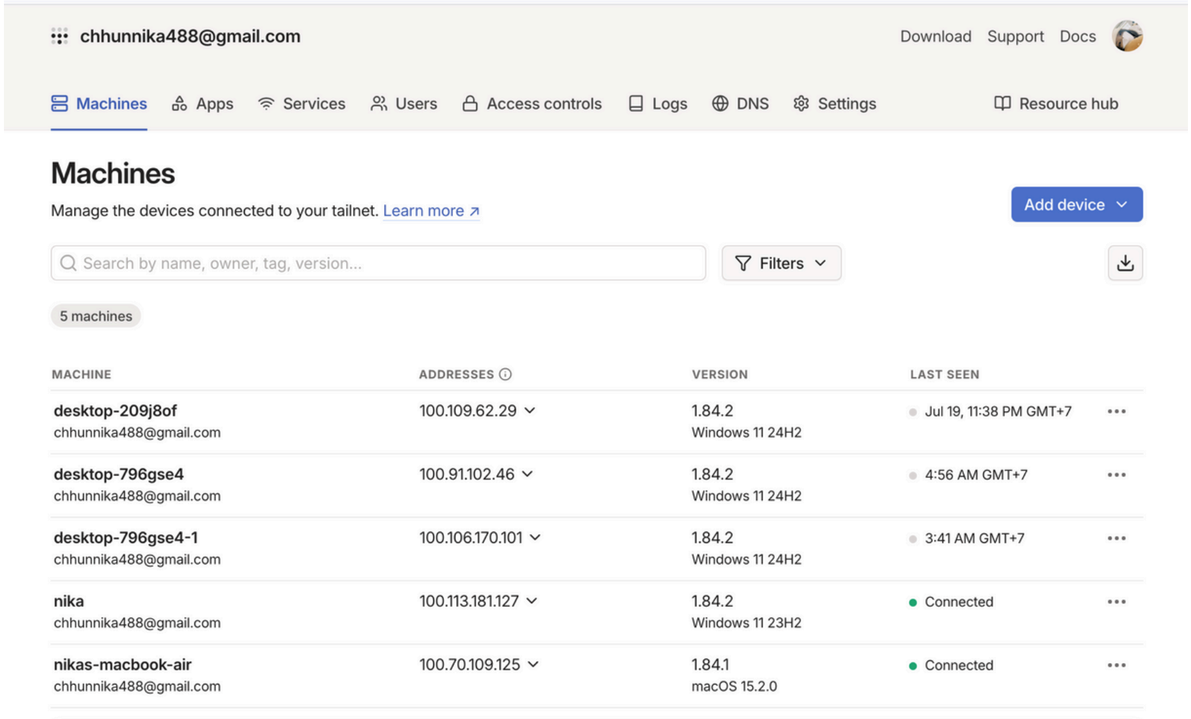
3m 37.5s

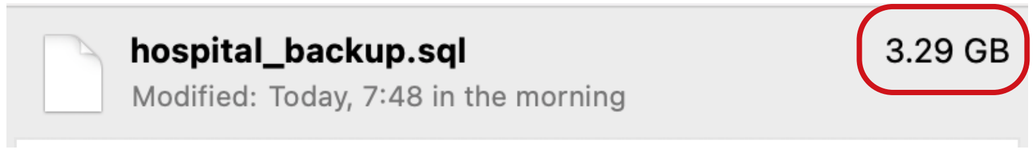
8

Database Server

tailscale

Tailscale was used to securely expose the MySQL database over a private network, enabling remote access during development and testing.





Due to size of the database we cannot host it that's why we expose it using tailscale

9

Conclusion

Challenges

Data generation

Populating millions of realistic records using Faker while maintaining data integrity using python

Performance optimization

Ensuring fast query performance with indexing and query tuning.

Access Control

Implementing a secure RBAC system to manage user permissions

Backup & Recovery integration

mbedding user-friendly backup and restore features into the web interface via JavaScript

Conclusion

Building the **Hospital Management System** was a valuable learning experience. We applied full-stack development practices, tackled real-world challenges, and created a secure, scalable, and user-friendly system for hospital data management.

Thank You