1. Project Scope Description

Project Objectives

The objective of this project is to develop a robust **Hospital Management System (HMS)** that facilitates efficient digital management of core hospital operations. The system is designed to streamline processes such as patient registration, doctor and nurse assignments, appointment scheduling, test management, billing, and department oversight.

By following core relational database design principles, the HMS ensures data consistency, eliminates redundancy, and supports scalability and security across multiple user roles. The system also enforces integrity constraints using primary and foreign keys, and models complex relationships through normalized schema and ERD constructs.

Project Functionalities

The Hospital Management System includes the following key functionalities:

- **Patient Registration:** Allows new patients to register with personal information, contact details, and emergency contacts.
- **Doctor & Nurse Assignment:** Assigns doctors and nurses to patients based on specialization and availability.
- **Appointment Scheduling:** Enables patients to book appointments with available doctors, while avoiding time-slot conflicts.
- **Medical Records Management:** Stores and manages patient diagnoses, prescriptions, test assignments, and results.
- **Billing System:** Automatically generates bills based on the services rendered, including appointments, treatments, and tests.
- **Department Management:** Manages departments and links them to respective doctors and nurses.

Entity-Relationship Diagram (ERD) Concepts Implemented

Entities

- **Strong Entities:** *Patient, Doctor, Nurse, Appointment, Bill, Department, Room*
- **Weak Entities:** *TestResults* (dependent on *Test* and *Patient*)

Relationships

- **One-to-One:** A patient is linked to one unique medical history.
- **One-to-Many:** A doctor can handle multiple appointments.
- **Many-to-Many:** A patient can receive multiple treatments/tests, and each test/treatment may apply to multiple patients.

Generalization/Specialization

- The entity *Staff* is generalized and specialized into *Doctor* and *Nurse*, enabling reusability and logical inheritance.

Role

- The system is overseen by an *Admin*, who performs managerial operations like registering users, managing departments, and assigning roles.

Ternary Relationship

- Tests represent a ternary relationship between *Patient*, *Doctor*, and *Nurse*, where:
    - A **Doctor** prescribes a test
    - A **Nurse** manages or performs it
    - The **Patient** undergoes the test

Attributes

- **Composite Attribute:** *Address* is composed of *Street*, *City*, *State*, and *ZipCode*.
- **Multivalued Attribute:** A *Doctor* can possess multiple *Specializations*.
- **Derived Attribute:** *Total Bill Amount* is calculated dynamically from associated service records (appointments, tests, etc.).

2. Entity-Relationship Diagram (ERD)

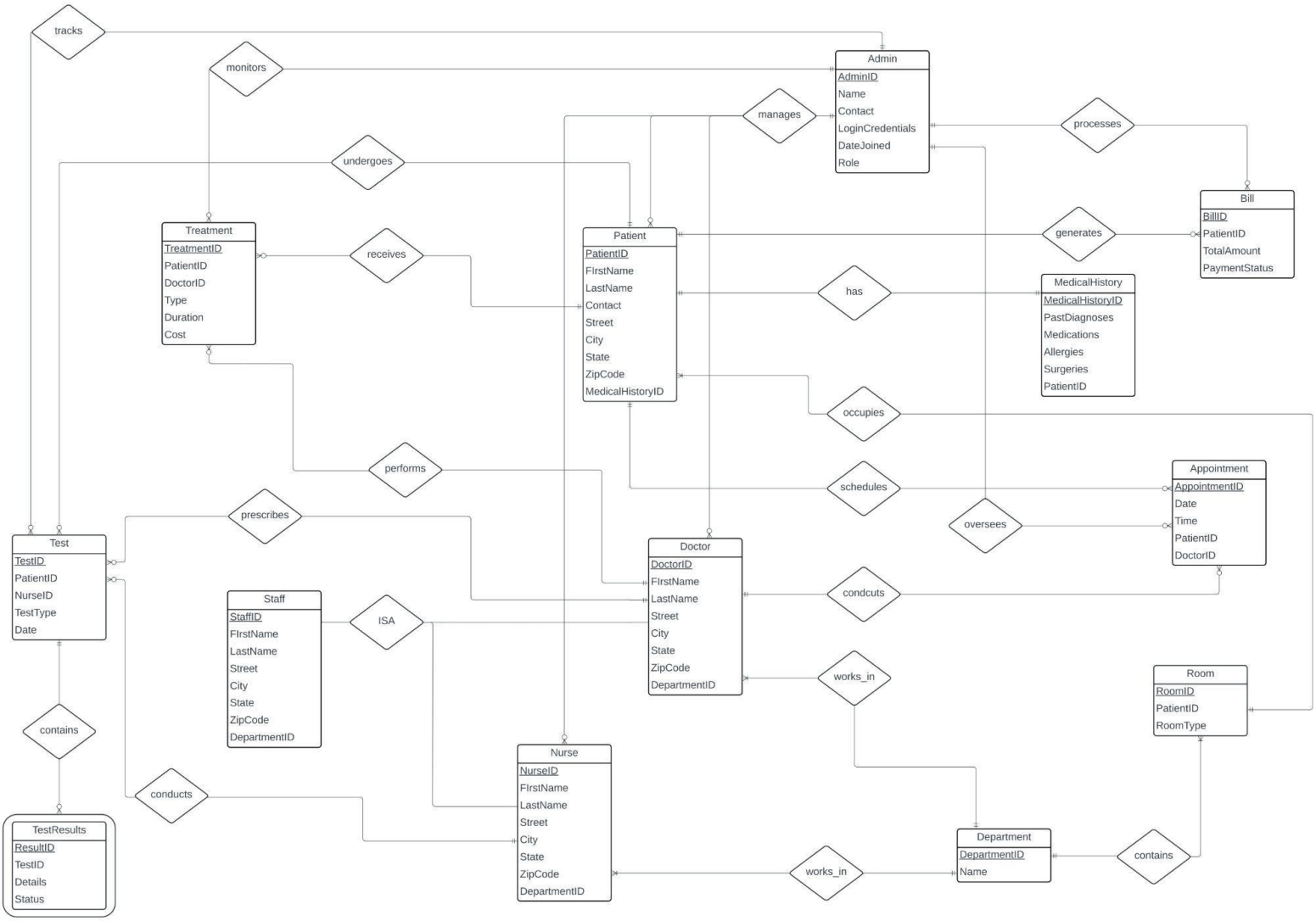


Figure 1: Entity-Relationship Diagram (ERD) of the Hospital Management System.

3. Relational Schema

Table Name	Attributes	Primary Key	Foreign Keys
Patient	PatientID, FirstName, LastName, Contact, Street, City, State, ZipCode, MedicalHistoryID	PatientID	—
Doctor	DoctorID, FirstName, LastName, Specialization, DepartmentID	DoctorID	DepartmentID
Nurse	NurseID, FirstName, LastName, DepartmentID	NurseID	DepartmentID
Appointment	AppointmentID, PatientID, DoctorID, Date, Time	AppointmentID	PatientID DoctorID
Bill	BillID, PatientID, TotalAmount, PaymentStatus	BillID	PatientID
Treatment	TreatmentID, PatientID, DoctorID, Type, Duration, Cost	TreatmentID	PatientID DoctorID
Test	TestID, PatientID, DoctorID, NurseID, TestType, Date	TestID	PatientID DoctorID NurseID
TestResults	ResultID, TestID, Status, Details	ResultID	TestID
Department	DepartmentID, Name	DepartmentID	—
Room	RoomID, PatientID, RoomType	RoomID	PatientID

Table 1: Relational Schema for the Hospital Management System.

4. SQL DDL for the Relation Schema

Created the HMS database

```
CREATE DATABASE IF NOT EXISTS hospital_db;
USE hospital_db;
```

Created the Patient Table

```
CREATE TABLE Patient (
   PatientID INT AUTO_INCREMENT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Contact VARCHAR(20),
   Street VARCHAR(100),
   City VARCHAR(50),
   State VARCHAR(50),
   ZipCode VARCHAR(10),
   MedicalHistoryID INT);
```

Created the MedicalHistory Table

```
CREATE TABLE MedicalHistory (
   MedicalHistoryID INT AUTO_INCREMENT PRIMARY KEY,
   PastDiagnoses TEXT,
   Medications TEXT,
   Allergies TEXT,
   Surgeries TEXT,
   PatientID INT,
   FOREIGN KEY (PatientID) REFERENCES Patient(PatientID));
```

Created the Doctor Table

```
CREATE TABLE Doctor (
   DoctorID INT AUTO_INCREMENT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Street VARCHAR(100),
   City VARCHAR(50),
   State VARCHAR(50),
   ZipCode VARCHAR(10),
   DepartmentID INT,
   FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID));
```

Created the Nurse Table

```
CREATE TABLE Nurse (
   NurseID INT AUTO_INCREMENT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Street VARCHAR(100),
   City VARCHAR(50),
   State VARCHAR(50),
   ZipCode VARCHAR(10),
   DepartmentID INT,
   FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID));
```

Created the Staff Table

```
CREATE TABLE Staff (
   StaffID INT AUTO_INCREMENT PRIMARY KEY,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Street VARCHAR(100),
   City VARCHAR(50),
   State VARCHAR(50),
   ZipCode VARCHAR(10),
   DepartmentID INT,
   FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID));
```

Created the Department Table

```
CREATE TABLE Department (
   DepartmentID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(100));
```

Created the Appointment Table

```
CREATE TABLE Appointment (
   AppointmentID INT AUTO_INCREMENT PRIMARY KEY,
   Date DATE,
   Time TIME,
   PatientID INT,
   DoctorID INT,
   FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
   FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID));
```

Created the Treatment Table

```
CREATE TABLE Treatment (
   TreatmentID INT AUTO_INCREMENT PRIMARY KEY,
   PatientID INT,
   DoctorID INT,
   Type VARCHAR(100),
   Duration VARCHAR(50),
   Cost DECIMAL(10,2),
   FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
   FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID));
```

Created the Test Table

```
CREATE TABLE Test (
   TestID INT AUTO_INCREMENT PRIMARY KEY,
   PatientID INT,
   NurseID INT,
   TestType VARCHAR(100),
   Date DATE,
   FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
   FOREIGN KEY (NurseID) REFERENCES Nurse(NurseID));
```

Created the TestResults Table

```
CREATE TABLE TestResults (
   ResultID INT AUTO_INCREMENT PRIMARY KEY,
   TestID INT,
   Details TEXT,
   Status VARCHAR(50),
   FOREIGN KEY (TestID) REFERENCES Test(TestID));
```

Created the Bill Table

```
CREATE TABLE Bill (
   BillID INT AUTO_INCREMENT PRIMARY KEY,
   PatientID INT,
   TotalAmount DECIMAL(10,2),
   PaymentStatus VARCHAR(50),
   FOREIGN KEY (PatientID) REFERENCES Patient(PatientID));
```

Inserted Default Department Records

```
INSERT INTO Department (Name)
VALUES ('Cardiology'), ('Neurology'), ('Pediatrics'), ('Orthopedics');
```

Created the Room Table

```
CREATE TABLE Room (
   RoomID INT AUTO_INCREMENT PRIMARY KEY,
   PatientID INT,
   RoomType VARCHAR(50),
   FOREIGN KEY (PatientID) REFERENCES Patient(PatientID));
```

Created the Admin Table

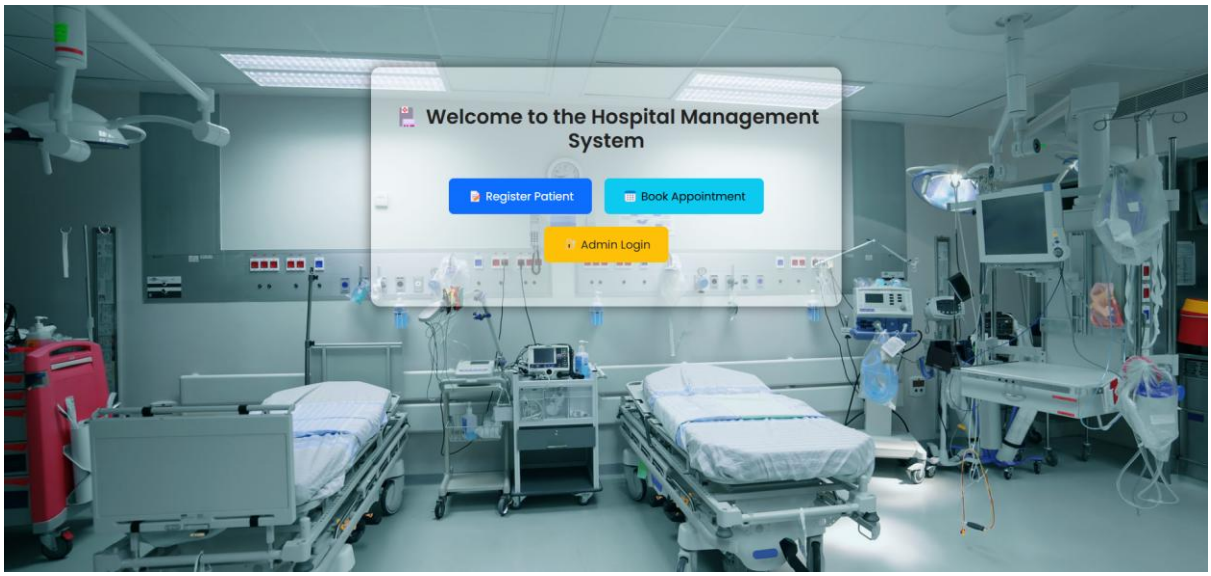
```
CREATE TABLE Admin (
   AdminID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(100),
   Contact VARCHAR(20),
   LoginCredentials VARCHAR(100),
   DateJoined DATE,
   Role VARCHAR(50));
```

Inserted Default Admin Record

```
INSERT INTO Admin (Name, Contact, LoginCredentials, DateJoined, Role)
VALUES ('admin', '0123456789', '1234', CURDATE(), 'SuperAdmin');
```

5. User Interface (UI) Screenshots

Homepage Interface

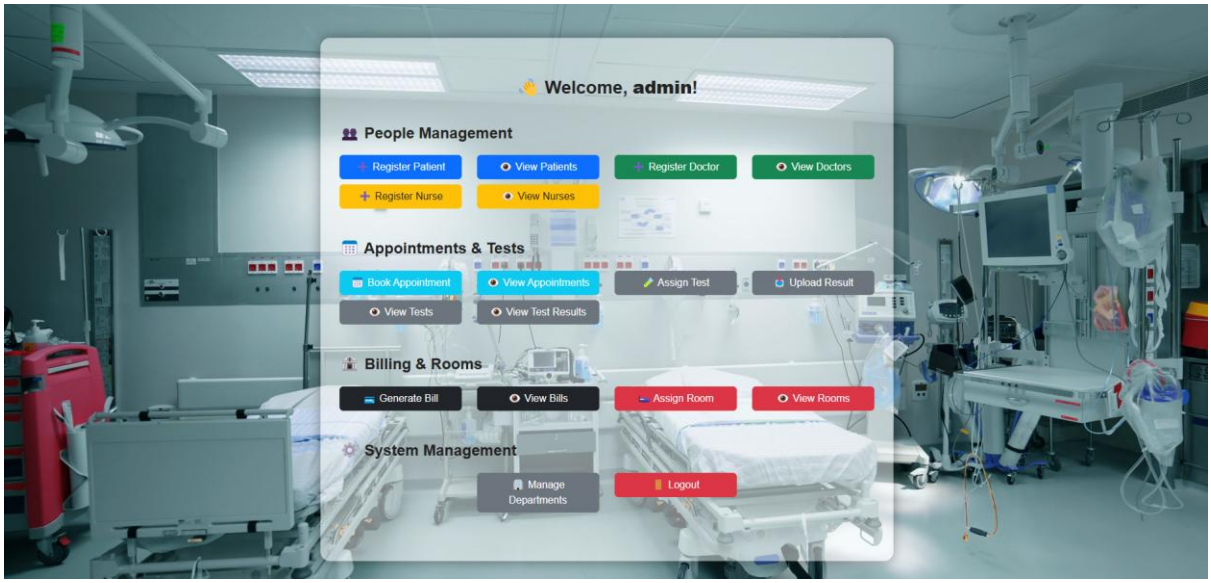


Simple landing page with access to patient registration and admin login.

Admin Login Panel

Secure login form for admin access.

Admin Dashboard



Centralized admin interface with grouped navigation buttons.

Register Patient Form

Input form to capture new patient details.

View Patients Table

List of all registered patients with delete functionality.

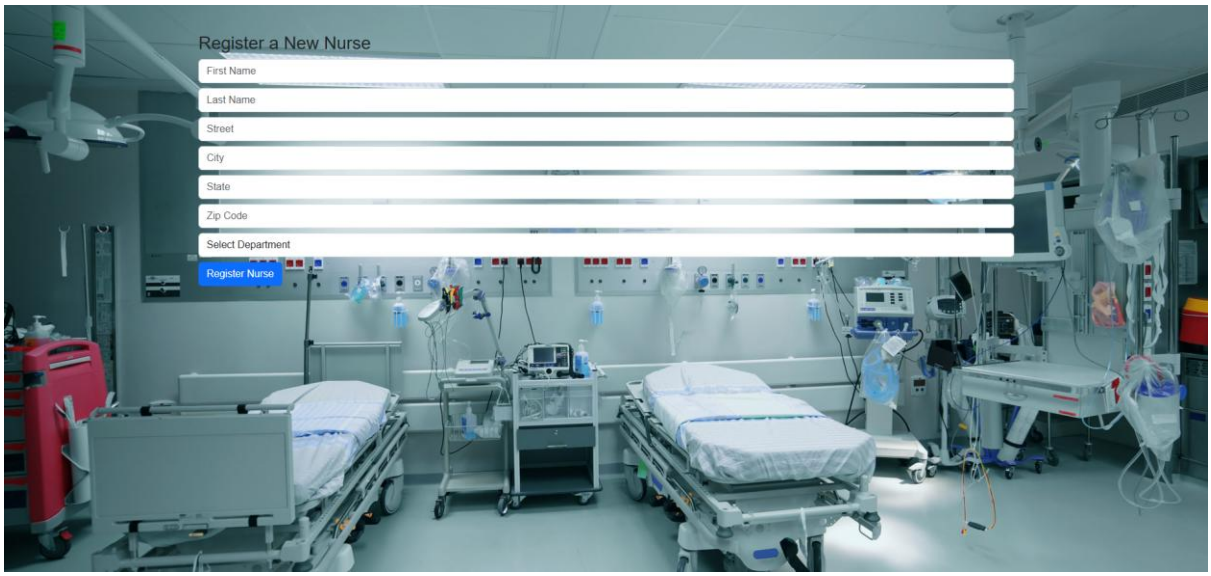
Register Doctor Form

Form to add a new doctor to the system.

View Doctors Table

Display of all doctors and their assigned departments.

Register Nurse Form

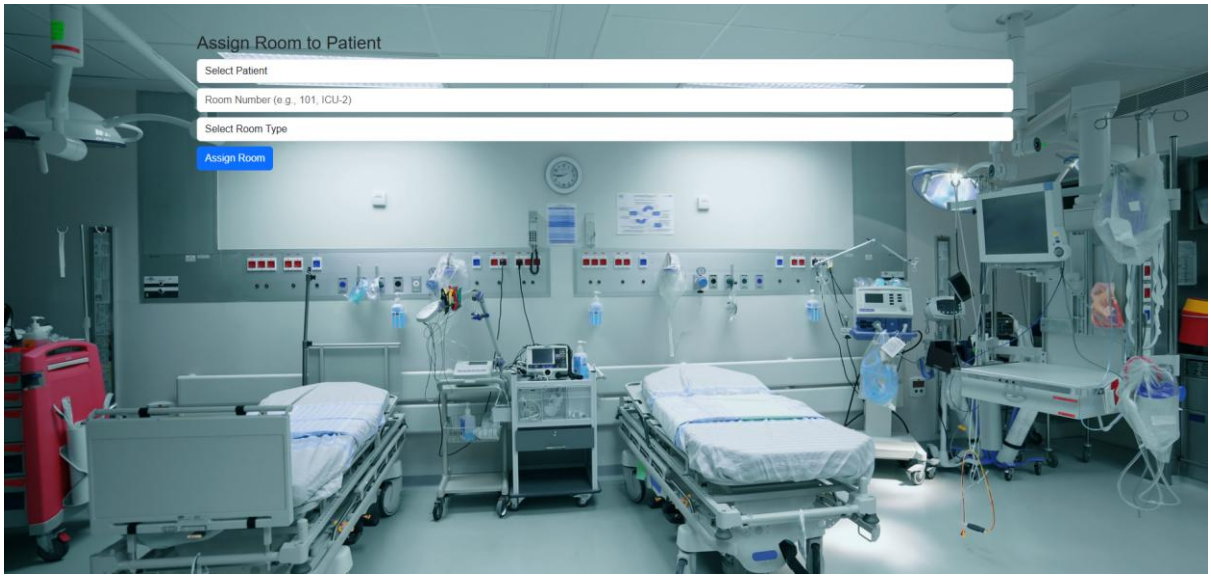


Form to register nurses under departments.

View Nurse's Table

Display of all nurses and their assigned departments.

Assign Room to Patient



Interface for assigning room type and room number.

View Assigned Rooms

Displays all patient room assignments with room type and delete option.

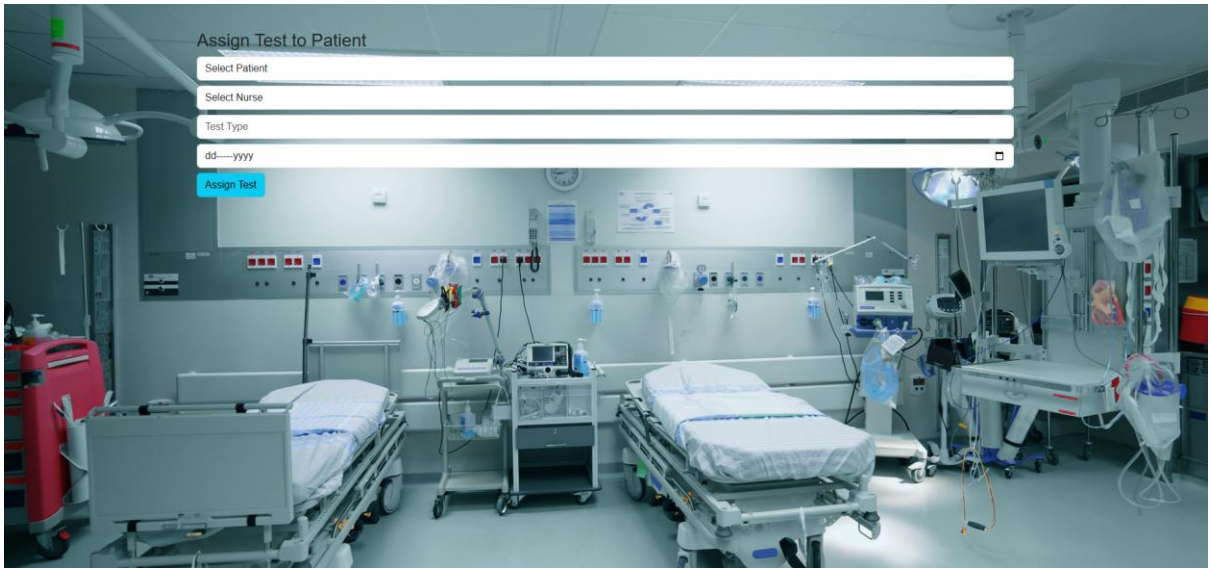
Book Appointment Page

Schedule patient appointments with doctors.

View Appointments Table

Overview of scheduled appointments.

Assign Test to Patient



Form to assign a diagnostic test and responsible nurse.

View Assigned Tests

Shows all tests assigned to patients along with test type, date, and nurse ID.

Upload Test Result

Interface to update test outcomes.

View Test Results Table

Table showing uploaded test results and status.

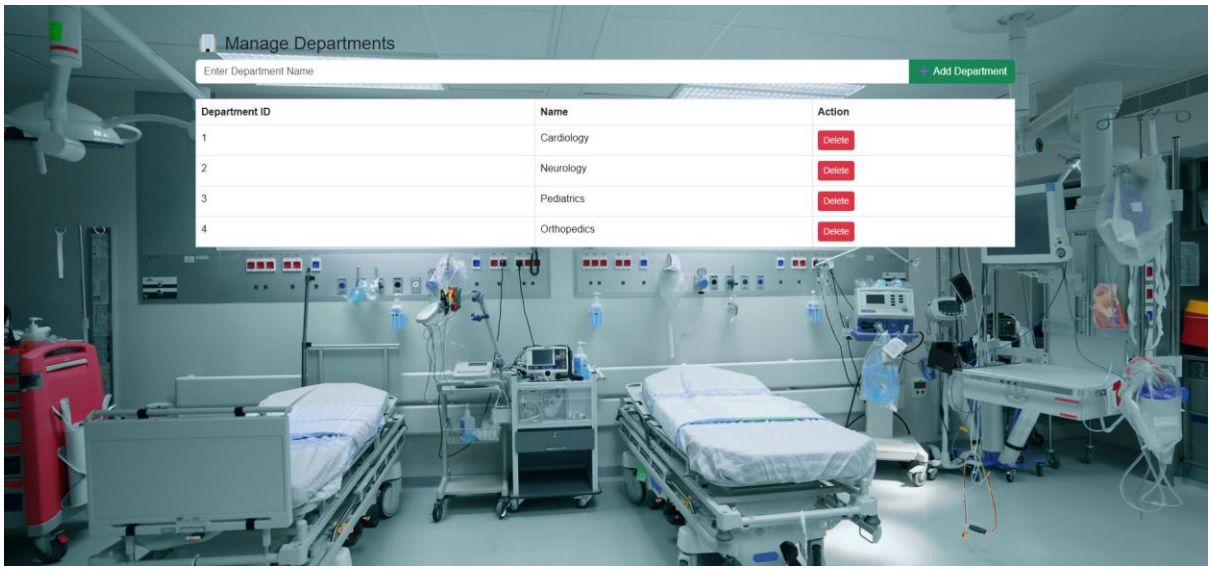
Generate Bill Page

Interface for creating patient billing records.

View Bills Table

Overview of all generated bills and payment statuses.

Manage Departments



Interface to add or delete medical departments.

6. Conclusion

The Hospital Management System (HMS) project successfully demonstrates the practical application of database design principles in developing a comprehensive and functional web-based solution for managing hospital operations. Through this project, we were able to integrate core database concepts such as entity relationships, normalization, foreign key constraints, and relational schema translation into a working system.

Key Findings

- The system efficiently supports core functionalities such as patient registration, staff management, appointment scheduling, test handling, billing, and room assignment.
- Implementation of foreign key relationships and referential integrity ensures consistent and reliable data flow across all modules.
- The ERD, relational schema, and SQL DDL were carefully aligned to reflect real-world healthcare relationships and operations.
- Modular and reusable components such as *header.php, db.php*, and *footer.php* improved maintainability and consistency throughout the application.

Challenges Faced

- Maintaining referential integrity during deletion operations required cascading deletions or manual handling of dependent records (e.g., deleting test results before a test).
- Designing the system to prevent appointment and room conflicts involved additional backend logic for validation.
- Ensuring consistency between ERD concepts and implementation required careful planning and normalization of data structures.
- Coordinating frontend/backend development across group members while maintaining consistency in UI and functionality.

Alignment with Database Design Principles

- The system strictly follows 1NF, 2NF, and 3NF to eliminate redundancy and maintain logical integrity.
- All ERD concepts such as strong and weak entities, generalization/specialization, composite and derived attributes, and ternary relationships were implemented or represented conceptually.
- Proper use of primary and foreign keys, along with indexing via *AUTO_INCREMENT*, ensures unique identification and scalability.
- Real-time form validation and foreign key enforcement ensure data accuracy and consistency during user operations.