PROJECT REPORT

"Hospital Management System"

Course Name
Database Management System Lab

Course Code
CSC 434

Section
I

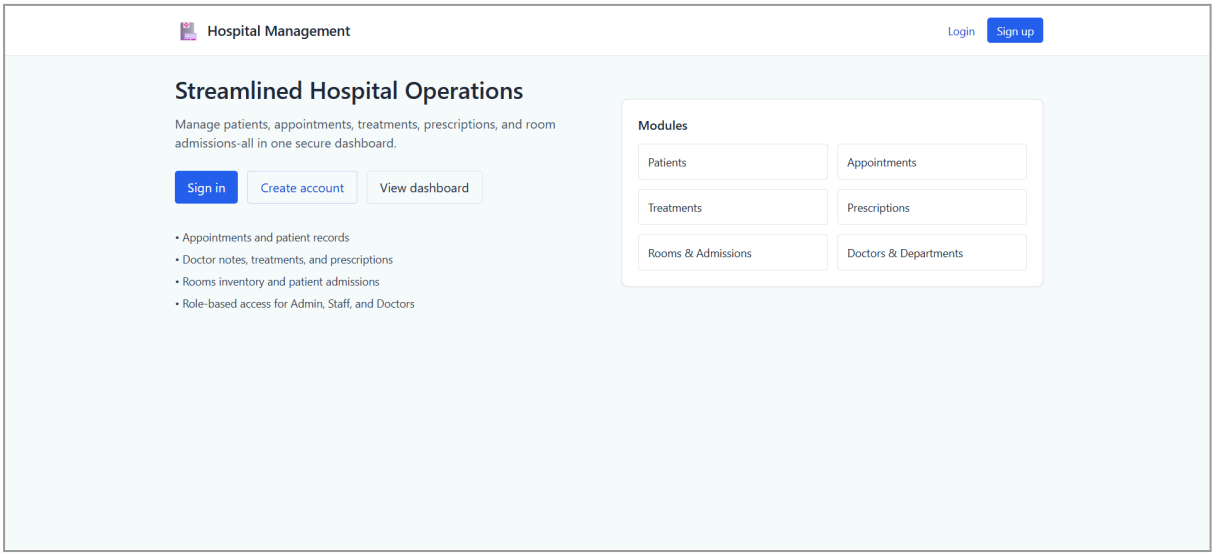
Semester
Summer 25

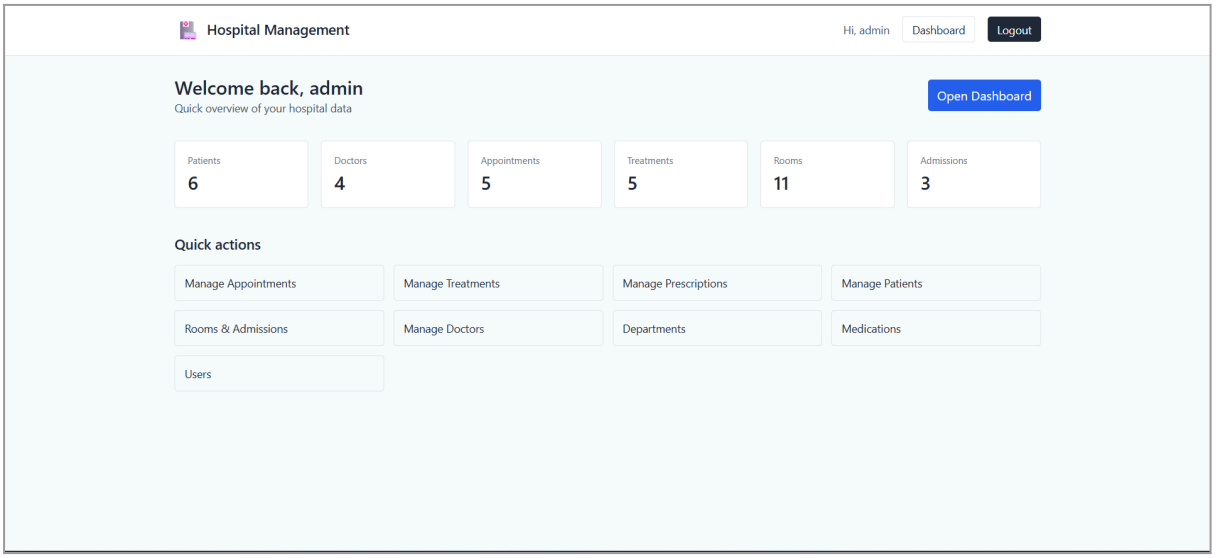
Submitted By
Md. Mahedi Zaman Zaber
ID: 23203134

Submitted To
Jubair Ahmed Nabin
Lecturer
Dept. of CSE, IUBAT

Date of Submission:
21-09-2025

HOSPITAL MANAGEMENT SYSTEM





1. Introduction

This report documents the design and implementation of a relational database and a role-aware web application for a **Hospital Management System (HMS)**. It provides a comprehensive view of the system's conceptual and technical design, covering the Entity–Relationship Diagram (ERD), schema definition, constraints, performance considerations, user interface, sample queries, limitations, and references.

Objective:
The primary objective is to build a normalized relational database and a minimal yet secure web front end capable of managing common hospital operations. These include patient and doctor records, appointment scheduling, treatments and prescriptions, room management and admissions, and role-based user access.

Scope:
The project adopts a database-first design methodology, guided by an ERD, and realized through a MySQL/InnoDB schema. A lightweight PHP backend and Tailwind CSS user interface validate the schema by enabling complete Create, Read, Update, Delete (CRUD) workflows across the major entities.

Importance:
A well-structured database is essential in healthcare contexts. Correct normalization reduces redundancy, while foreign keys and constraints preserve referential integrity. This foundation ensures accurate and reliable data for dashboards, reports, and daily hospital operations, supporting both efficiency and accountability.

2. Project Overview

The **Hospital Management System (HMS)** is designed to streamline key hospital operations through a unified, role-aware web platform. It supports multiple domains, including:

1. **Patient and Doctor Registries**: Maintain core records for patients, doctors, and departments.
2. **Appointment Scheduling**: Create, update, and track the status of appointments between patients and doctors.
3. **Clinical Management**: Record treatments and issue prescriptions tied to medications.
4. **Facility Management**: Manage room inventory and handle patient admissions and discharges.

5. **User Roles**: Admin, staff, and doctor accounts, with optional linkage between users and doctor profiles to enforce clinical ownership.

Technology Stack

- **Backend**: PHP with PDO (positional placeholders for prepared statements), MySQL/InnoDB running under XAMPP.
- **Frontend**: Tailwind CSS (via CDN) for styling and Alpine.js for lightweight interactivity, ensuring responsive navigation.
- **Security**: Session management, CSRF tokens on all POST forms and API endpoints, role checks, and doctor ownership validation.
- **Routing**: Relative paths are resolved using `__DIR__`; a redirect helper respects `BASE_URL` and nested folder structures to work seamlessly in XAMPP environments.

Authentication and User Experience

- A single login entry point serves all roles, with dashboards tailored to the specific role (admin, staff, or doctor).
- Doctors only see and manage data relevant to their patients and treatments.
- Destructive actions (e.g., deleting records) are restricted according to role-based privileges to ensure accountability and data integrity.

3. Requirements Analysis

Functional requirements:

1. Maintain master data for departments, doctors, patients, medications, and rooms.
2. Schedule appointments between patients and doctors, with status updates (`scheduled`, `completed`, `cancelled`).
3. Record treatments for patients, optionally linked to appointments.
4. Issue prescriptions for treatments, including dosage, frequency, and duration.
5. Admit patients into rooms and record discharges.

6. Provide a unified dashboard showing key counts, today's appointments, and - for staff/admin - widgets for rooms and admissions.
7. Enforce role-based access and ownership: doctors manage only their clinical records, while admins/staff manage broader hospital data.

Non-functional requirements:

1. Normalize data to at least **3NF** to minimize redundancy.
2. Enforce referential integrity with foreign keys and constraints.
3. Ensure acceptable performance with indexes and optimized queries.
4. Deliver a consistent, responsive UI using **Tailwind CSS**.

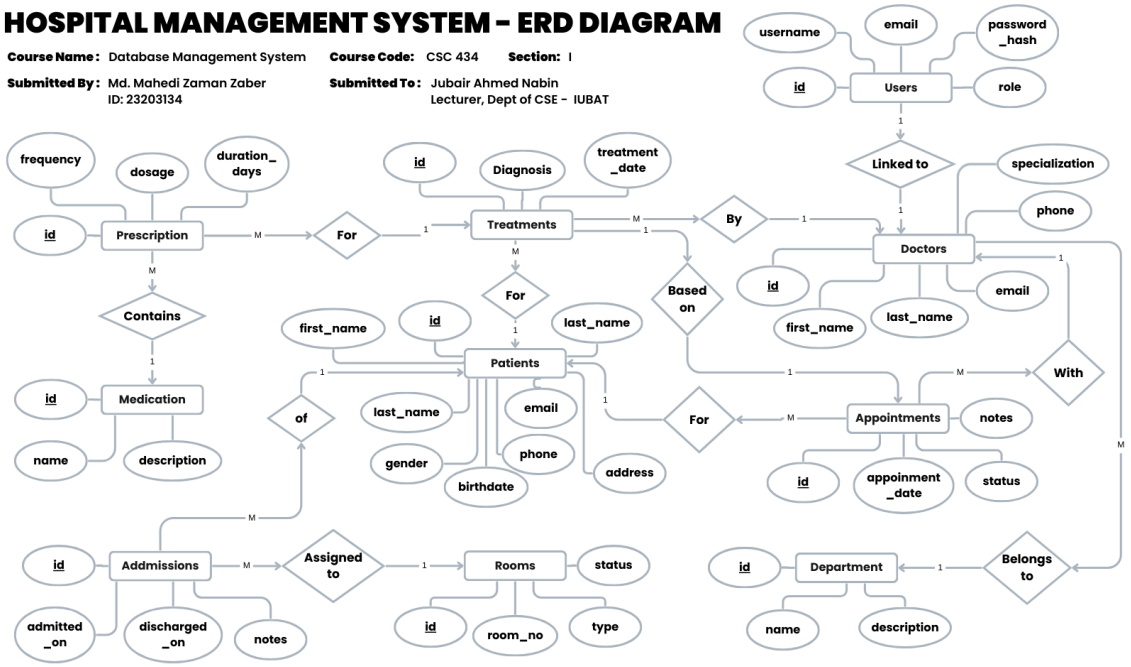
Key entities and relationships (high level):

1. **Department – Doctor**: One department can have many doctors.
2. **Patient – Appointment – Doctor**: A patient can have many appointments; each appointment is with one doctor.
3. **Patient – Treatment – Doctor/Appointment**: A patient can have many treatments; treatments may be linked to a doctor (nullable) and optionally tied to an appointment.
4. **Treatment – Prescription – Medication**: Each treatment can generate multiple prescriptions; prescriptions reference medications.
5. **Patient – Admission – Room**: A patient can be admitted to many rooms over time; each room can host many admissions.
6. **User – Doctor**: Each system user has a role (`admin`, `staff`, `doctor`), with doctors optionally linked to a user account through a unique foreign key.

4. Entity-Relationship Diagram (ERD)

The project's ERD is included in the repository root as *Hospital Management System ERD.png*. It illustrates the main entities, their attributes, and

cardinalities through foreign key relationships.



It models the entities listed above and their cardinalities via foreign keys.

Highlights:

1. **Appointments** connect patients and doctors, capturing date, time, and status.
2. **Treatments** may originate from an appointment or be recorded independently.
3. **Prescriptions** link treatments to medications, specifying dosage, frequency, and duration.
4. **Admissions** associate patients with rooms over defined time periods.
5. **Users** can be tied to doctors, supporting doctor-scoped permissions.

Normalization:

1. Attributes are factored into separate entities, for instance, medications are distinct from prescriptions.
2. One-to-many relationships are modeled with foreign keys, avoiding repeating groups.

3. Controlled fields (e.g., role, gender, room type/status, appointment status) use `ENUM`s to enforce consistency and reduce input errors.

5. Database Schema

The schema is defined in `schema.sql`. Major tables and constraints:

1. **departments**(id PK, name UNIQUE, description)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
name	varchar(150)	NO	UNI	*NULL*	
description	text	YES		*NULL*	

2. **patients**(id PK, first_name, last_name, gender ENUM('male','female','other') NULL, birthdate, phone, email, address)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
first_name	varchar(100)	NO		*NULL*	
last_name	varchar(100)	NO		*NULL*	
gender	enum('male','female','other')	YES		*NULL*	
birthdate	date	YES		*NULL*	
phone	varchar(30)	YES		*NULL*	
email	varchar(190)	YES		*NULL*	
address	varchar(255)	YES		*NULL*	

3. **doctors**(id PK, first_name, last_name, email UNIQUE, phone, specialization, department_id FK→departments.id ON DELETE SET

NULL)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
first_name	varchar(100)	NO		*NULL*	
last_name	varchar(100)	NO		*NULL*	
email	varchar(190)	NO	UNI	*NULL*	
phone	varchar(30)	YES		*NULL*	
specialization	varchar(150)	YES		*NULL*	
department_id	int(11)	YES	MUL	*NULL*	

4. **appointments**(id PK, patient_id FK→patients.id ON DELETE CASCADE, doctor_id FK→doctors.id ON DELETE CASCADE, appointment_date DATETIME, status ENUM('scheduled','completed','cancelled') DEFAULT 'scheduled', notes)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
patient_id	int(11)	NO	MUL	*NULL*	
doctor_id	int(11)	NO	MUL	*NULL*	
appointment_date	datetime	NO		*NULL*	
status	enum('scheduled','completed','cancelled')	NO		scheduled	
notes	text	YES		*NULL*	

5. **treatments**(id PK, patient_id FK→patients.id ON DELETE CASCADE, doctor_id FK→doctors.id ON DELETE SET NULL, appointment_id FK→appointments.id ON DELETE SET NULL, diagnosis, treatment_date DATE)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
patient_id	int(11)	NO	MUL	*NULL*	
doctor_id	int(11)	YES	MUL	*NULL*	
appointment_id	int(11)	YES	MUL	*NULL*	
diagnosis	text	YES		*NULL*	
treatment_date	date	NO		*NULL*	

6. **medications**(id PK, name UNIQUE, description)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
name	varchar(150)	NO	UNI	*NULL*	
description	text	YES		*NULL*	

7. **prescriptions**(id PK, treatment_id FK→treatments.id ON DELETE CASCADE, medication_id FK→medications.id ON DELETE RESTRICT, dosage, frequency, duration_days INT)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
treatment_id	int(11)	NO	MUL	*NULL*	
medication_id	int(11)	NO	MUL	*NULL*	
dosage	varchar(100)	NO		*NULL*	
frequency	varchar(100)	NO		*NULL*	
duration_days	int(11)	NO		*NULL*	

8. **rooms**(id PK, room_number UNIQUE, type ENUM('General','Private','ICU','Emergency','Maternity'), status ENUM('available','occupied','maintenance') DEFAULT 'available')

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
room_number	varchar(20)	NO	UNI	*NULL*	
type	enum('General','Private','ICU','Emergency','Maternity')	YES		*NULL*	
status	enum('available','occupied','maintenance')	NO		available	

9. **admissions**(id PK, patient_id FK→patients.id ON DELETE CASCADE, room_id FK→rooms.id ON DELETE RESTRICT, admitted_on DATETIME,

discharged_on DATETIME NULL, notes)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
patient_id	int(11)	NO	MUL	*NULL*	
room_id	int(11)	NO	MUL	*NULL*	
admitted_on	datetime	NO		*NULL*	
discharged_on	datetime	YES		*NULL*	
notes	text	YES		*NULL*	

10. **users**(id PK, username UNIQUE, email UNIQUE, password_hash, role ENUM('admin','staff','doctor') DEFAULT 'staff', linked_doctor_id UNIQUE NULL FK→doctors.id ON DELETE SET NULL)

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
username	varchar(120)	NO	UNI	*NULL*	
email	varchar(190)	NO	UNI	*NULL*	
password_hash	varchar(255)	NO		*NULL*	
role	enum('admin','staff','doctor')	NO		staff	
linked_doctor_id	int(11)	YES	UNI	*NULL*	

Design notes:

1. **Referential Integrity**: Foreign keys enforce consistent relationships between entities. `ON DELETE` rules mirror real-world expectations - for example, deleting a patient cascades to related appointments, treatments, and admissions, while deleting a doctor sets their clinical records to `NULL` rather than removing them.
2. **Controlled Vocabularies**: `ENUM` types are used to enforce consistency on key fields such as roles, appointment status, gender, and room type/status.
3. **Date/Time Semantics**: `DATETIME` is used for moment-specific events like appointments and admissions, while `DATE` is reserved for static values such as birth dates and treatment dates.

6. Performance Optimization

Schema-level:

1. Primary keys and `UNIQUE` constraints on natural keys (such as email, username, room number, and medication name) ensure fast lookups and prevent duplication.
2. Since InnoDB requires indexed foreign keys, MySQL automatically enforces indexes on FK columns, improving join performance.

Query-level:

1. Pagination with `LIMIT/OFFSET` is used for list pages to avoid scanning entire tables.
2. Targeted projections power dashboard elements like counts and "today's appointments."
3. Additional indexes are recommended for scalability:
    - `appointments(doctor_id, appointment_date)`
    - `treatments(doctor_id, treatment_date)`
    - `admissions(room_id, discharged_on)`
    - `prescriptions(treatment_id)`
    - `patients(last_name, first_name)`

Application-level:

1. Prepared statements (via PDO with positional placeholders) reduce parsing overhead and strengthen query safety.
2. Role-based and ownership filters restrict result sets to relevant doctors, ensuring both efficiency and security.

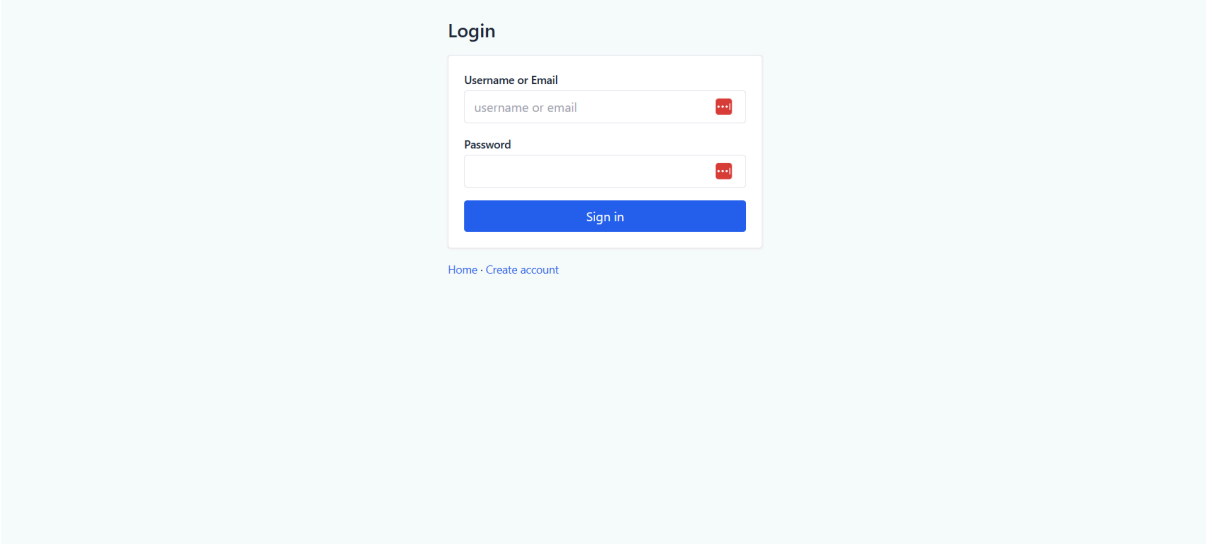
7. System Interface

Here are the interface preview of all pages (all Tailwind UI, single login, one dashboard curated by role) of the system:

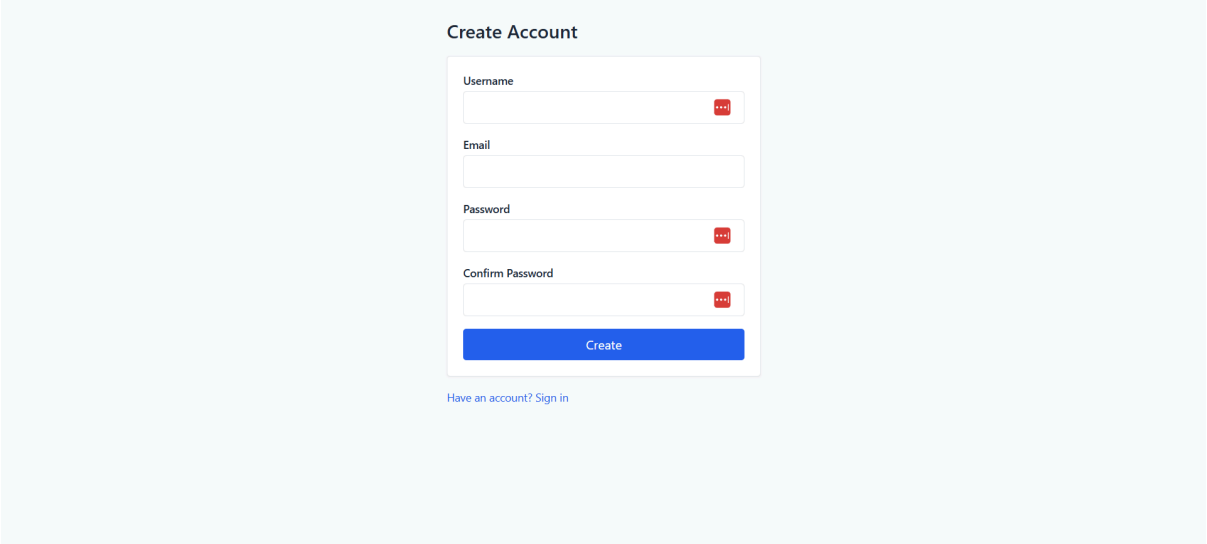
Authentication:

`auth/login.php`, `auth/signup.php`, `auth/logout.php`

Login Page:



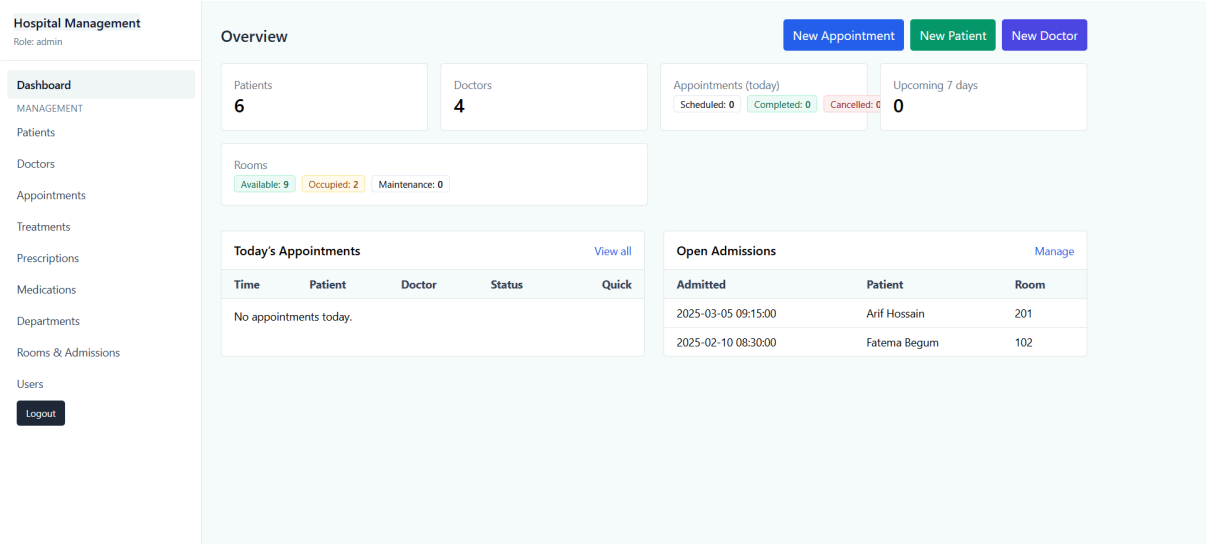
Sign Up Page:



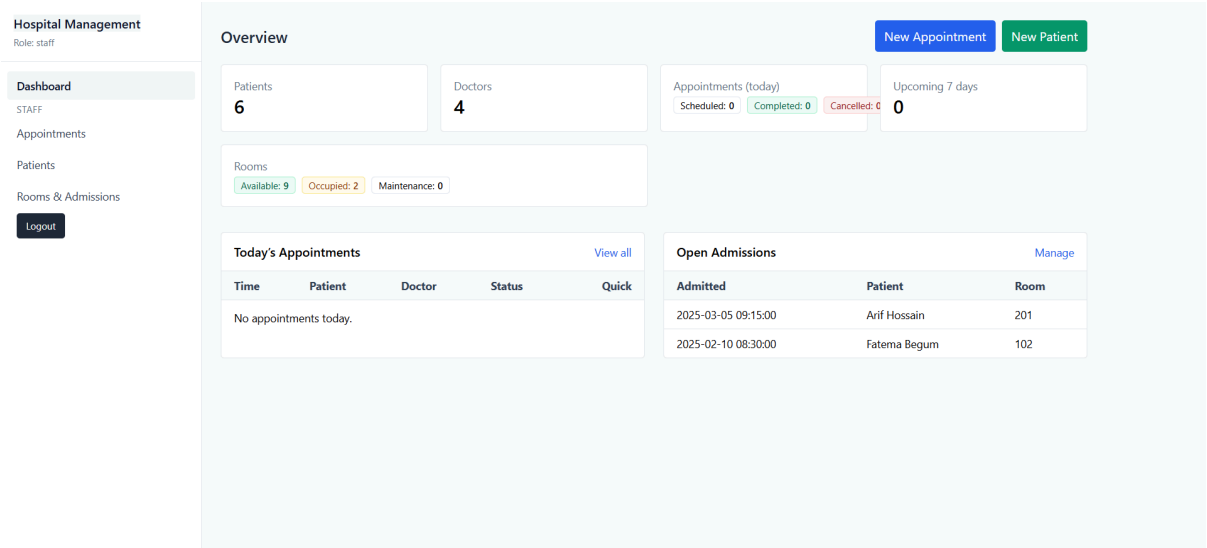
Dashboard:

`dashboard/index.php` with role-aware cards and "Today's Appointments" quick status actions:

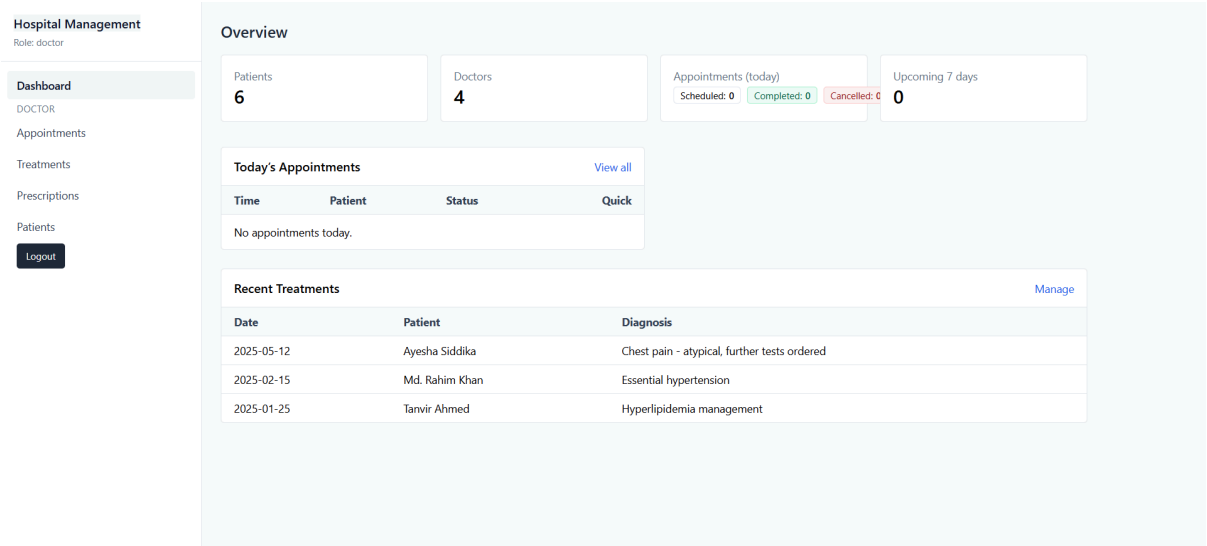
For **admins**:



For **Staffs**:



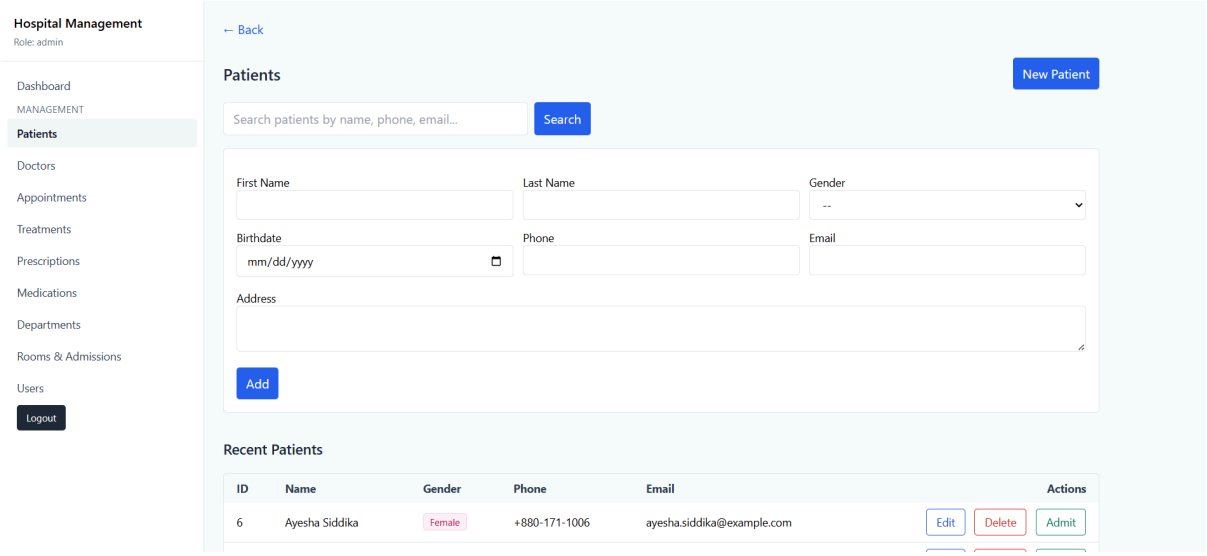
For **Doctors**:



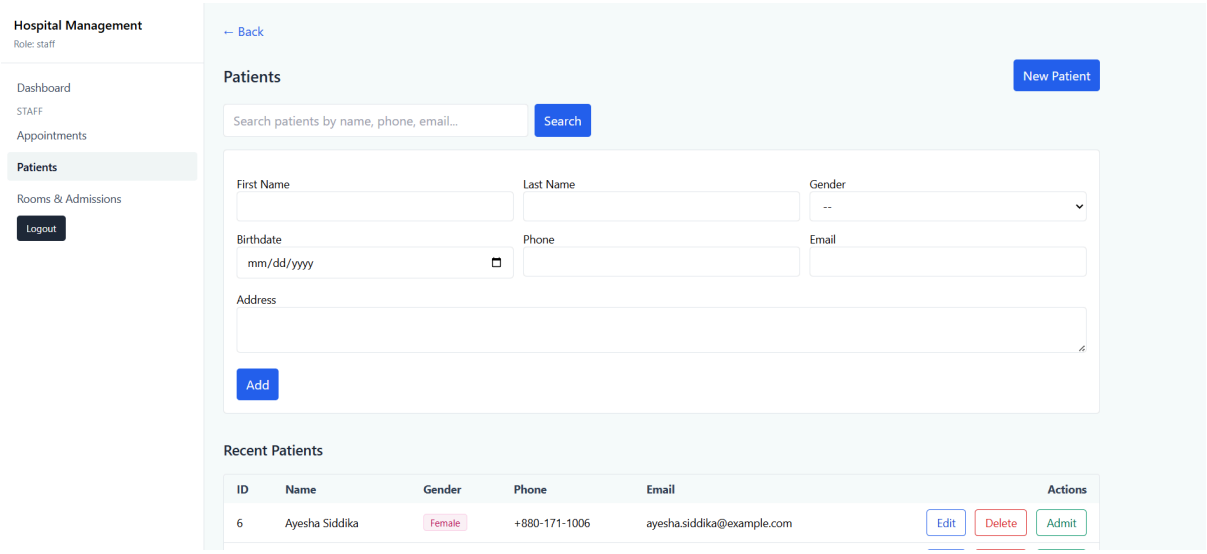
Patients Management:

`dashboard/manage-patients.php`

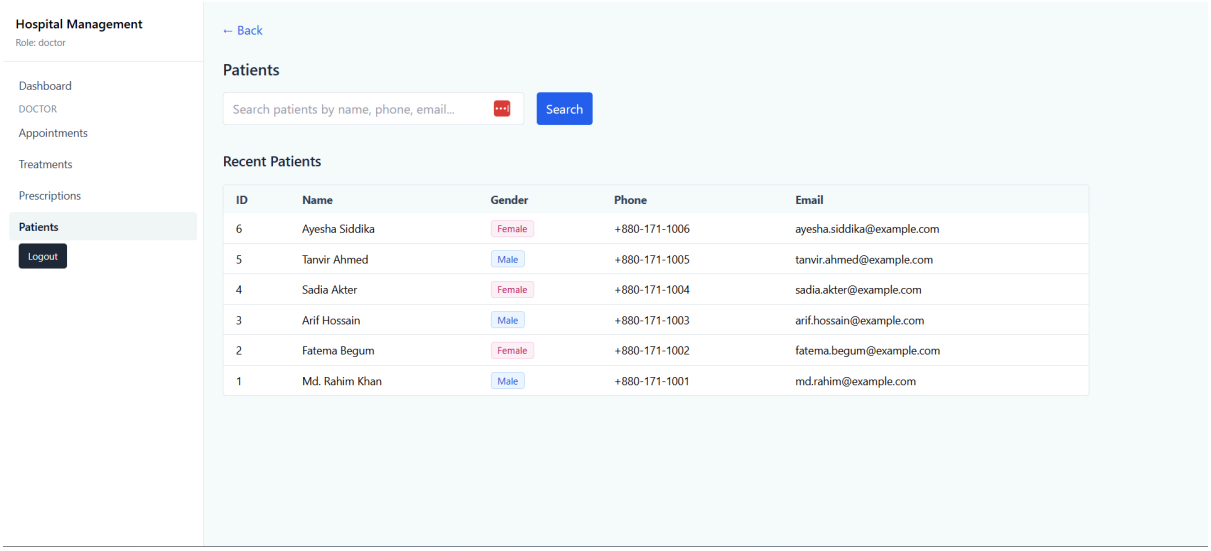
For **Admins**:



For **Staff**:

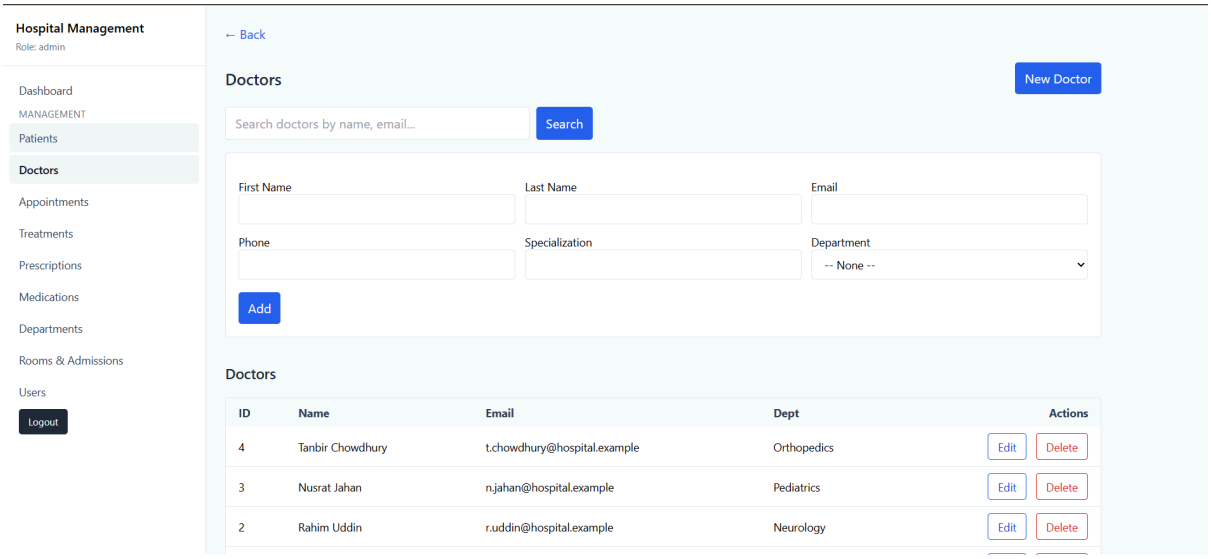


For **Doctor**:



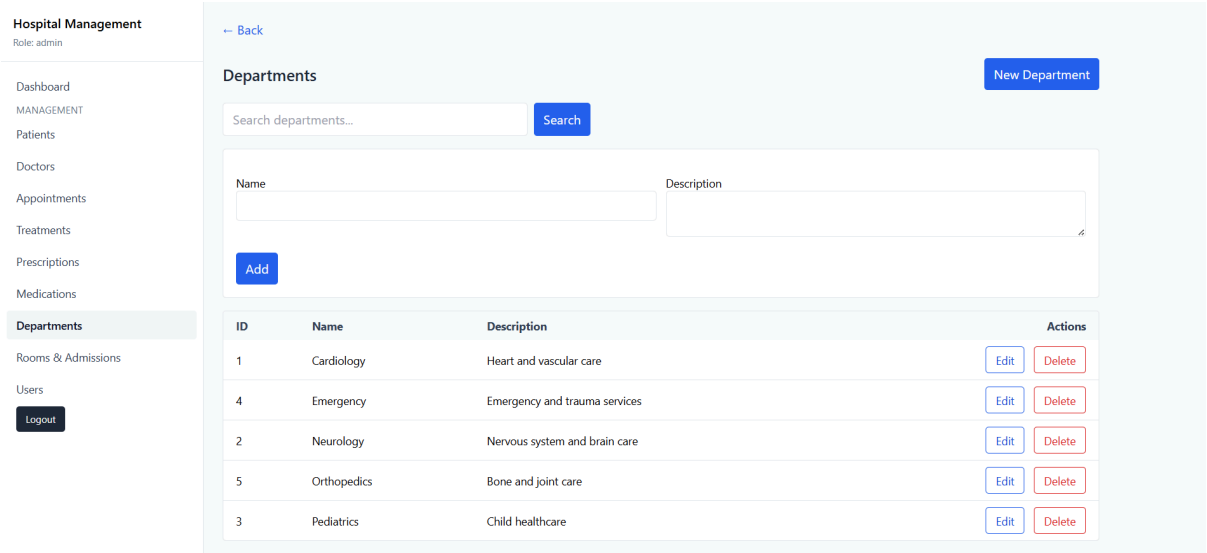
Doctors Management (admin only):

`dashboard/manage-doctors.php`



Departments Management (admin only):

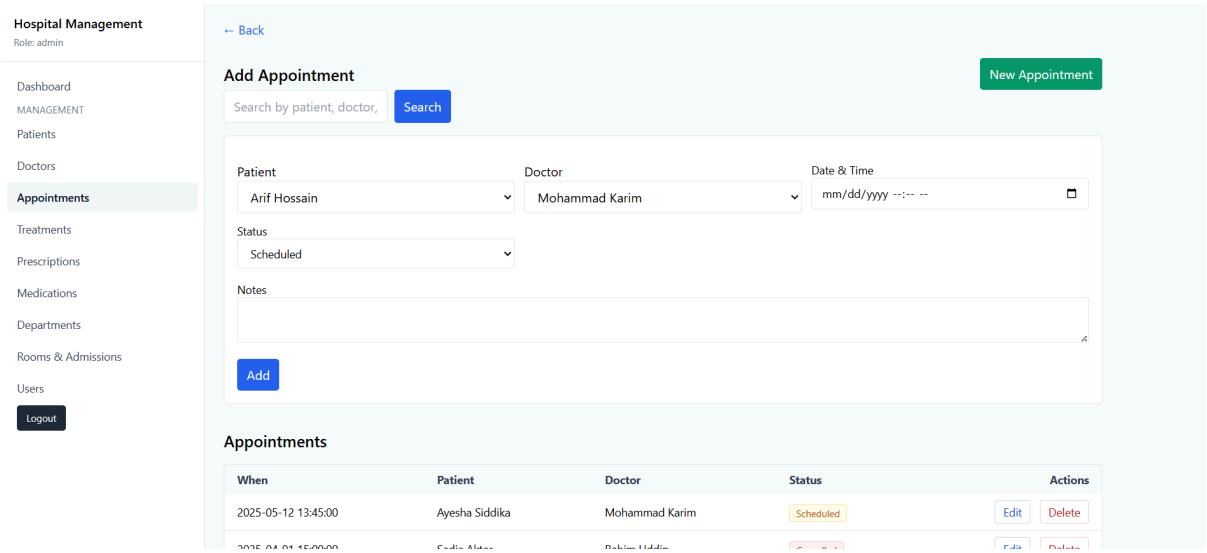
`dashboard/manage-departments.php`



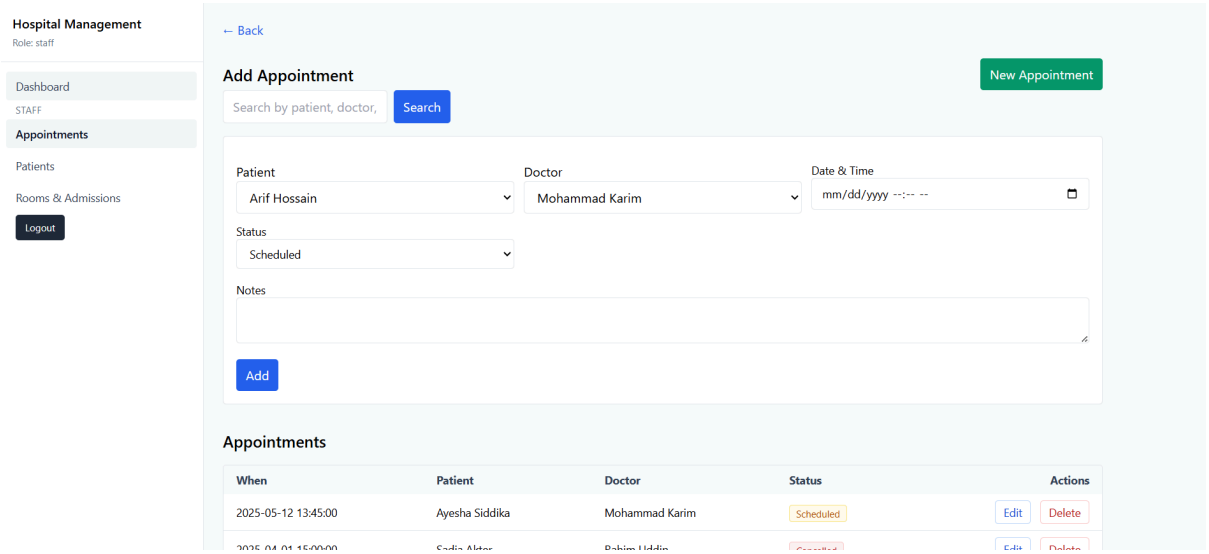
Appointments Management:

`dashboard/manage-appointments.php` (doctors edit own status/notes only)

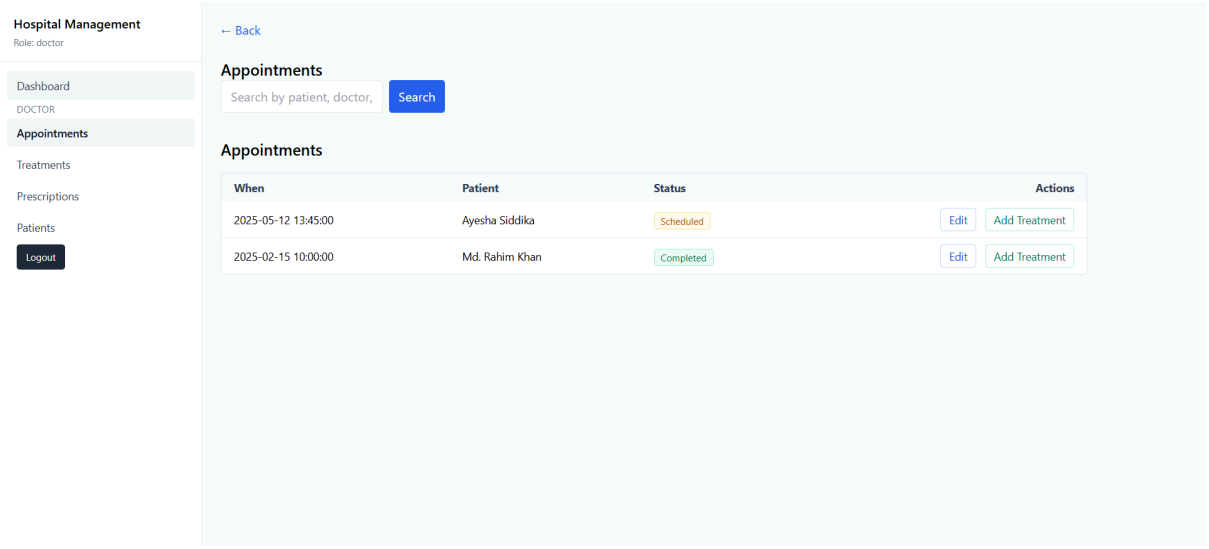
For **Admins**:



For **Staffs**:



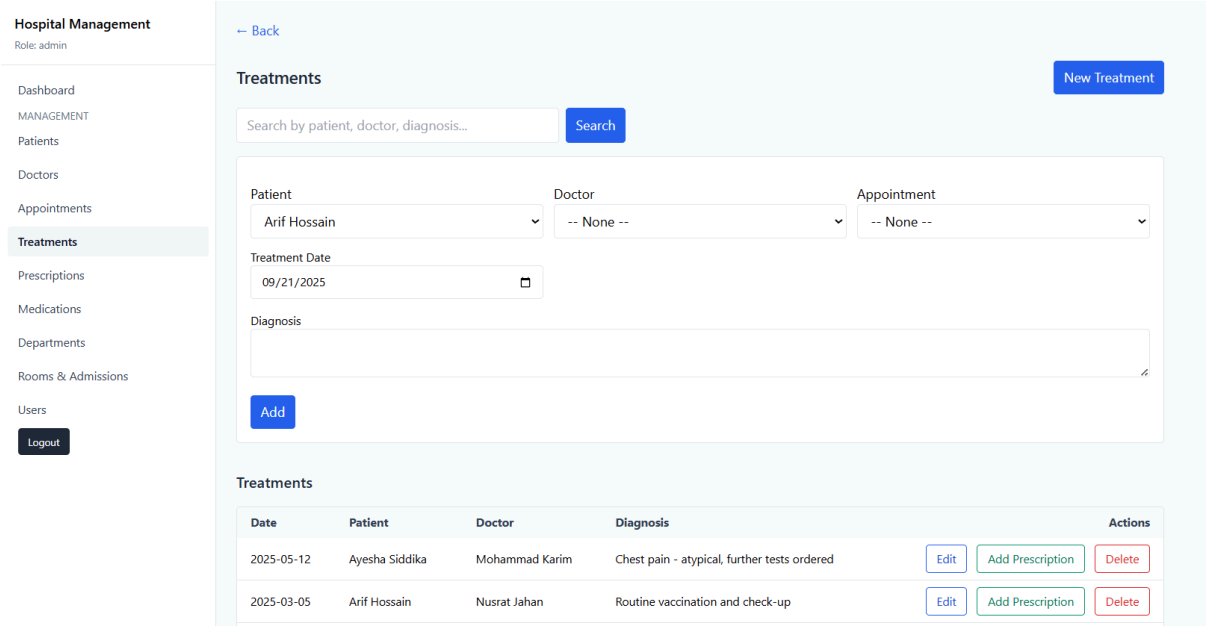
For **Doctors**:



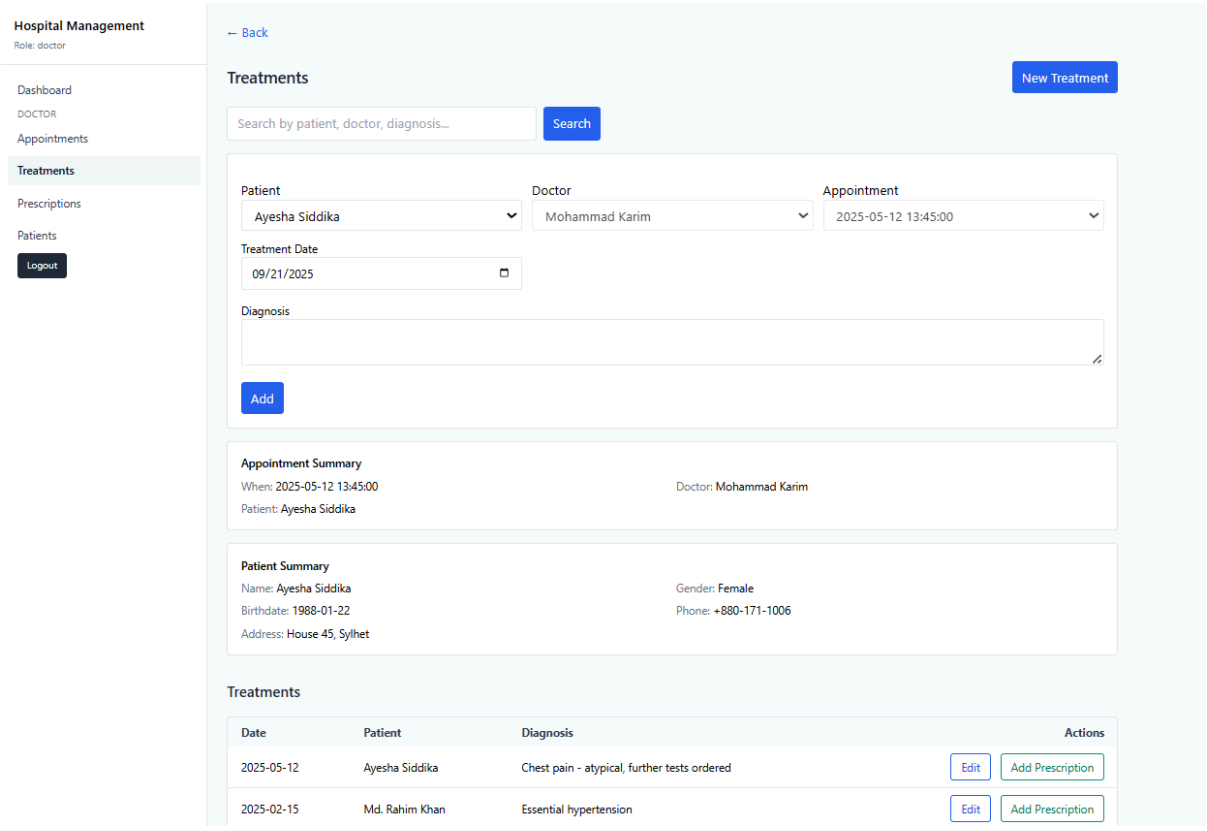
Treatments Management:

`dashboard/manage-treatments.php` (doctor ownership enforced; deep-link from appointments)

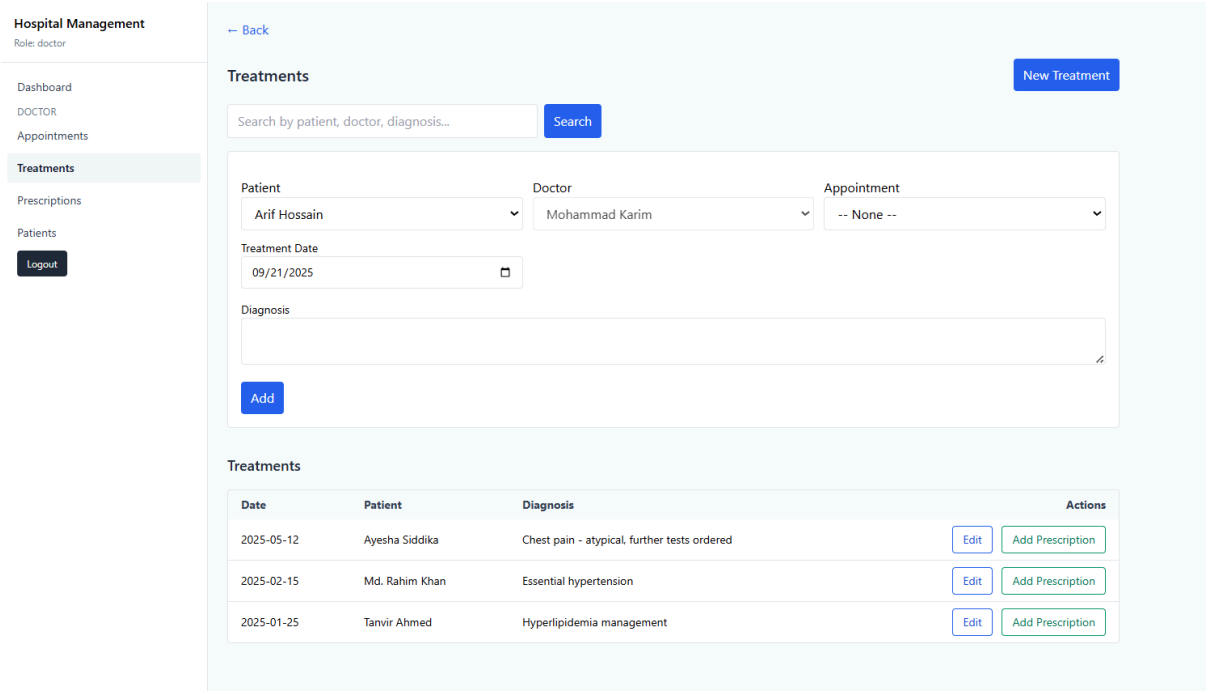
For **Admins**:



For **Doctors** (When redirected from appointment):



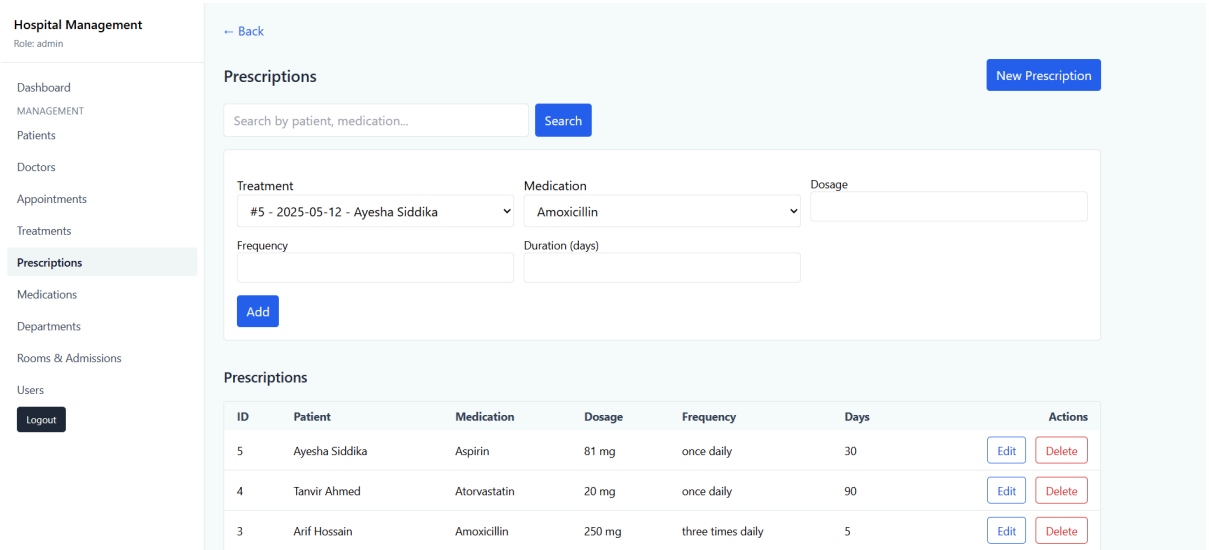
For **Doctors** (When directly accessed):



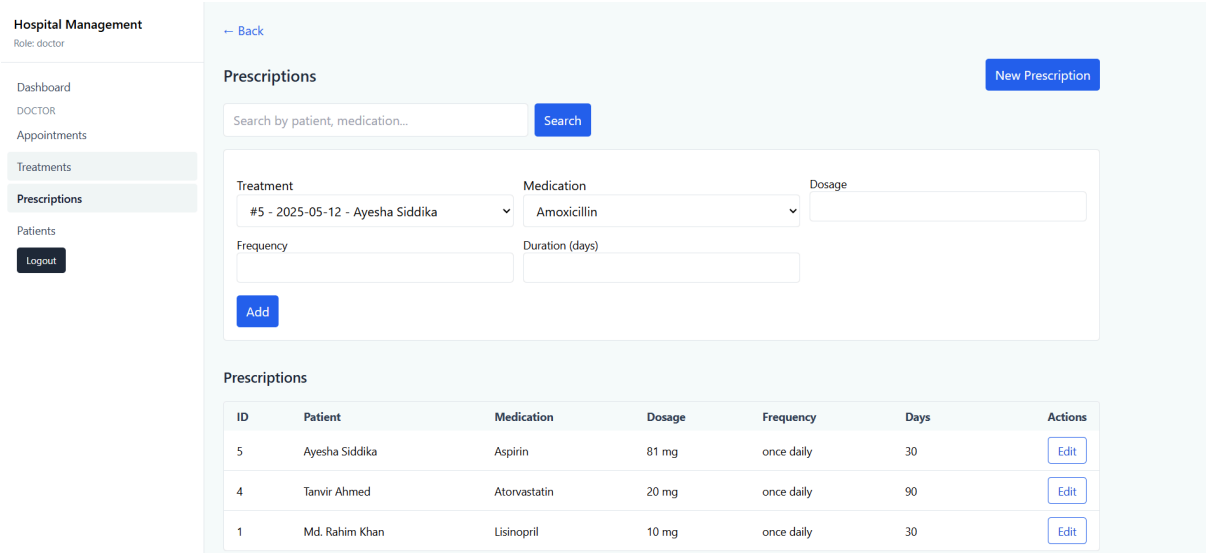
Prescriptions Management:

`dashboard/manage-prescriptions.php` (linked to treatments)

For **Admins**:

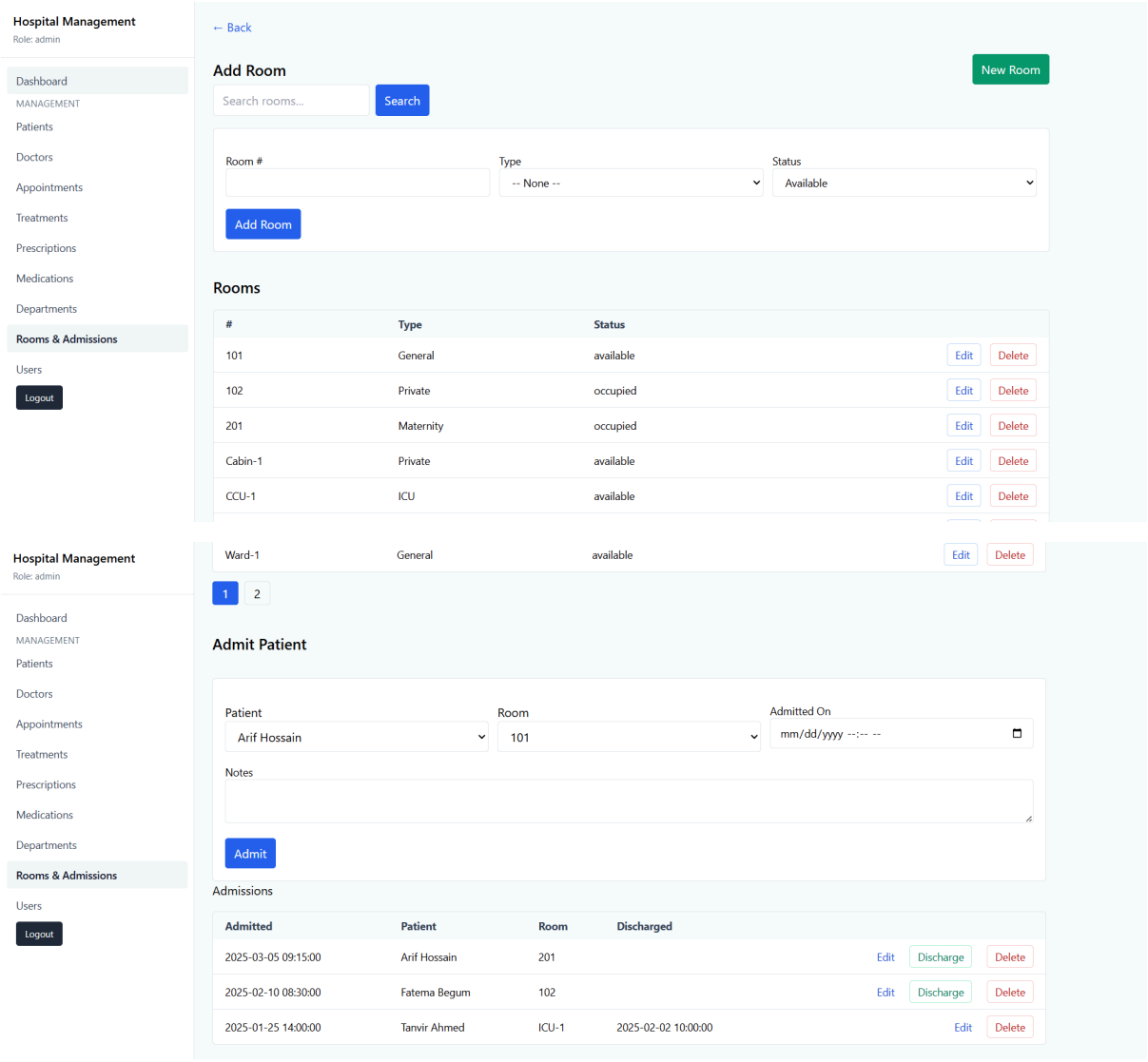


For **Doctors**:



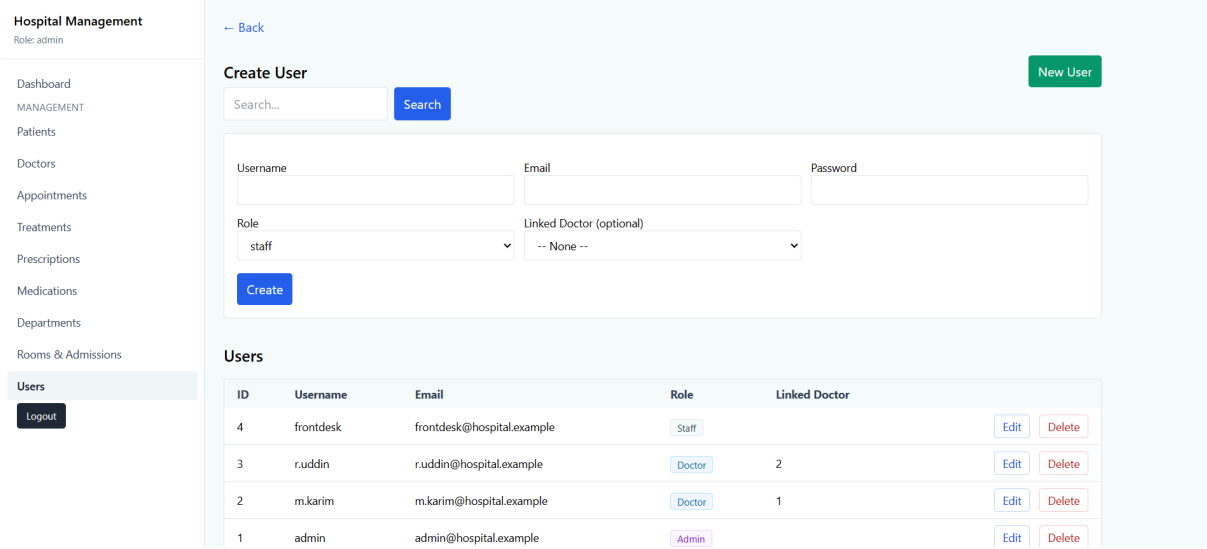
Rooms & Admissions Management (admin/staff):

dashboard/manage-rooms.php



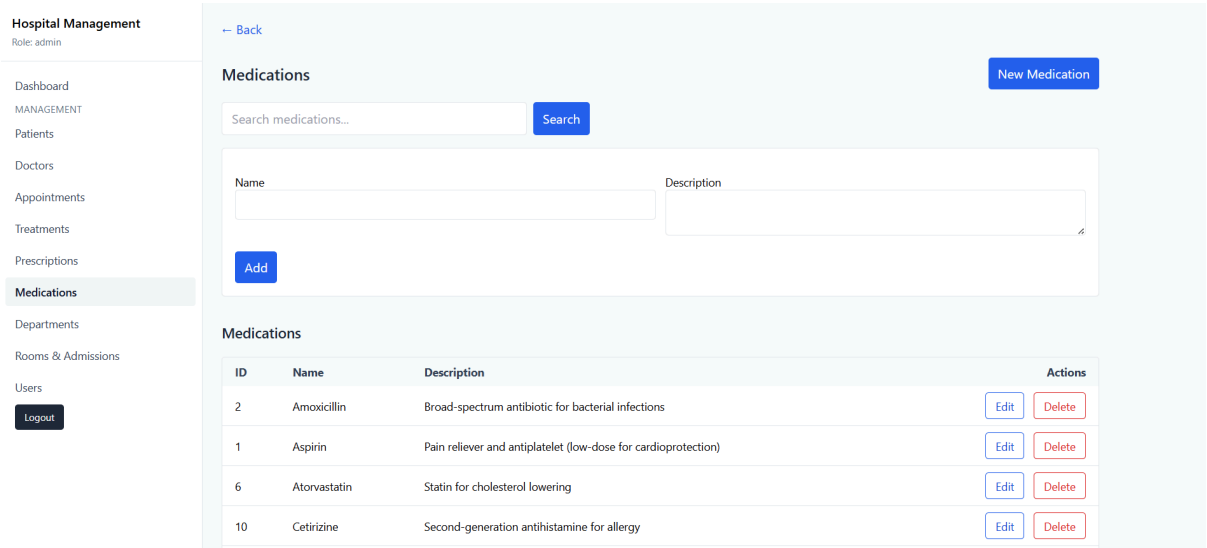
Users Management (admin only):

`dashboard/manage-users.php`



Medications Management (admin only):

`dashboard/manage-medications.php`



8. Sample Queries

Database and its table creation:

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

```
CREATE TABLE IF NOT EXISTS departments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(150) NOT NULL UNIQUE,
  description TEXT NULL
);

CREATE TABLE IF NOT EXISTS patients (
  id INT AUTO_INCREMENT PRIMARY KEY,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  gender ENUM('male','female','other') NULL,
  birthdate DATE NULL,
  phone VARCHAR(30) NULL,
  email VARCHAR(190) NULL,
  address VARCHAR(255) NULL
);

CREATE TABLE IF NOT EXISTS doctors (
  id INT AUTO_INCREMENT PRIMARY KEY,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  email VARCHAR(190) NOT NULL UNIQUE,
  phone VARCHAR(30) NULL,
  specialization VARCHAR(150) NULL,
  department_id INT NULL,
  FOREIGN KEY (department_id) REFERENCES departments(id) ON DELETE
SET NULL
);

CREATE TABLE IF NOT EXISTS appointments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  patient_id INT NOT NULL,
  doctor_id INT NOT NULL,
  appointment_date DATETIME NOT NULL,
  status ENUM('scheduled','completed','cancelled') NOT NULL DEFAULT
'scheduled',
  notes TEXT NULL,
  FOREIGN KEY (patient_id) REFERENCES patients(id) ON DELETE CASCADE,
```

```
  FOREIGN KEY (doctor_id) REFERENCES doctors(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS treatments (
  id INT AUTO_INCREMENT PRIMARY KEY,
  patient_id INT NOT NULL,
  doctor_id INT NULL,
  appointment_id INT NULL,
  diagnosis TEXT NULL,
  treatment_date DATE NOT NULL,
  FOREIGN KEY (patient_id) REFERENCES patients(id) ON DELETE CASCADE,
  FOREIGN KEY (doctor_id) REFERENCES doctors(id) ON DELETE SET NULL,
  FOREIGN KEY (appointment_id) REFERENCES appointments(id) ON DELETE
SET NULL
);

CREATE TABLE IF NOT EXISTS medications (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(150) NOT NULL UNIQUE,
  description TEXT NULL
);

CREATE TABLE IF NOT EXISTS prescriptions (
  id INT AUTO_INCREMENT PRIMARY KEY,
  treatment_id INT NOT NULL,
  medication_id INT NOT NULL,
  dosage VARCHAR(100) NOT NULL,
  frequency VARCHAR(100) NOT NULL,
  duration_days INT NOT NULL,
  FOREIGN KEY (treatment_id) REFERENCES treatments(id) ON DELETE
CASCADE,
  FOREIGN KEY (medication_id) REFERENCES medications(id) ON DELETE
RESTRICT
);

CREATE TABLE IF NOT EXISTS rooms (
  id INT AUTO_INCREMENT PRIMARY KEY,
  room_number VARCHAR(20) NOT NULL UNIQUE,
```

```
  type ENUM('General','Private','ICU','Emergency','Maternity') NULL,
  status ENUM('available','occupied','maintenance') NOT NULL DEFAULT
'available'
);

CREATE TABLE IF NOT EXISTS admissions (
  id INT AUTO_INCREMENT PRIMARY KEY,
  patient_id INT NOT NULL,
  room_id INT NOT NULL,
  admitted_on DATETIME NOT NULL,
  discharged_on DATETIME NULL,
  notes TEXT NULL,
  FOREIGN KEY (patient_id) REFERENCES patients(id) ON DELETE CASCADE,
  FOREIGN KEY (room_id) REFERENCES rooms(id) ON DELETE RESTRICT
);

CREATE TABLE IF NOT EXISTS users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(120) NOT NULL UNIQUE,
  email VARCHAR(190) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  role ENUM('admin','staff','doctor') NOT NULL DEFAULT 'staff',
  linked_doctor_id INT NULL UNIQUE,
  FOREIGN KEY (linked_doctor_id) REFERENCES doctors(id) ON DELETE SET
NULL
);
```

Sample Data Insertion:

```
-- Demo departments
INSERT IGNORE INTO departments (id,name,description) VALUES
(1,'Cardiology','Heart and vascular care'),
(2,'Neurology','Nervous system and brain care'),
(3,'Pediatrics','Child healthcare'),
(4,'Emergency','Emergency and trauma services'),
(5,'Orthopedics','Bone and joint care');
```

```
-- Demo patients (Bangladeshi names)
INSERT IGNORE INTO patients
(id,first_name,last_name,gender,birthdate,phone,email,address) VALUES
(1,'Md.
Rahim','Khan','male','1980-05-14','+880-171-1001','md.rahim@example.com','
House 12, Dhanmondi, Dhaka'),
(2,'Fatema','Begum','female','1992-11-02','+880-171-1002','fatema.begum@e
xample.com','Road 7, Uttara, Dhaka'),
(3,'Arif','Hossain','male','2015-07-20','+880-171-1003','arif.hossain@example.c
om','Sector 5, Mirpur, Dhaka'),
(4,'Sadia','Akter','female','1975-03-30','+880-171-1004','sadia.akter@example
.com','House 8, Gulshan, Dhaka'),
(5,'Tanvir','Ahmed','male','1968-09-12','+880-171-1005','tanvir.ahmed@exam
ple.com','Road 3, Chittagong'),
(6,'Ayesha','Siddika','female','1988-01-22','+880-171-1006','ayesha.siddika@e
xample.com','House 45, Sylhet');

-- Demo doctors (Bangladeshi names)
INSERT IGNORE INTO doctors
(id,first_name,last_name,email,phone,specialization,department_id)
VALUES
(1,'Mohammad','Karim','m.karim@hospital.example','+880-171-2001','Interve
ntional Cardiology',1),
(2,'Rahim','Uddin','r.uddin@hospital.example','+880-171-2002','Stroke
Specialist',2),
(3,'Nusrat','Jahan','n.jahan@hospital.example','+880-171-2003','Pediatrician'
,3),
(4,'Tanbir','Chowdhury','t.chowdhury@hospital.example','+880-171-2004','Or
thopedic Surgeon',5);

-- Demo medications
INSERT IGNORE INTO medications (id,name,description) VALUES
(1,'Aspirin','Pain reliever and antiplatelet (low-dose for cardioprotection)'),
(2,'Amoxicillin','Broad-spectrum antibiotic for bacterial infections'),
(3,'Lisinopril','ACE inhibitor for hypertension'),
(4,'Ibuprofen','NSAID for pain and inflammation'),
(5,'Paracetamol','Analgesic and antipyretic (acetaminophen)'),
(6,'Atorvastatin','Statin for cholesterol lowering'),
```

```
(7,'Metformin','First-line oral medication for type 2 diabetes'),
(8,'Omeprazole','Proton-pump inhibitor for acid-related disorders'),
(9,'Salbutamol','Short-acting bronchodilator for asthma/COPD'),
(10,'Cetirizine','Second-generation antihistamine for allergy');

-- Demo rooms
INSERT IGNORE INTO rooms (id,room_number,type,status) VALUES
(1,'101','General','available'),
(2,'102','Private','occupied'),
(3,'ICU-1','ICU','available'),
(4,'ER-1','Emergency','available'),
(5,'201','Maternity','occupied'),
(6,'Ward-1','General','available'),
(7,'Cabin-1','Private','available'),
(8,'VIP-1','Private','available'),
(9,'CCU-1','ICU','available'),
(10,'Labour-1','Maternity','available'),
(11,'Ward-2','General','available');

-- Demo admissions
INSERT IGNORE INTO admissions
(id,patient_id,room_id,admitted_on,discharged_on,notes) VALUES
(1,2,2,'2025-02-10 08:30:00',NULL,'Observation after minor surgery'),
(2,5,3,'2025-01-25 14:00:00','2025-02-02 10:00:00','Post-op recovery
complete'),
(3,3,5,'2025-03-05 09:15:00',NULL,'Routine newborn observation');

-- Demo appointments
INSERT IGNORE INTO appointments
(id,patient_id,doctor_id,appointment_date,status,notes) VALUES
(1,1,1,'2025-02-15 10:00:00','completed','Follow-up for hypertension'),
(2,2,4,'2025-02-18 11:30:00','scheduled','Knee pain evaluation'),
(3,3,3,'2025-03-05 09:00:00','completed','Well child visit'),
(4,4,2,'2025-04-01 15:00:00','cancelled','Patient requested reschedule'),
(5,6,1,'2025-05-12 13:45:00','scheduled','Chest pain assessment');

-- Demo treatments
INSERT IGNORE INTO treatments
```

```
(id,patient_id,doctor_id,appointment_id,diagnosis,treatment_date)
VALUES
(1,1,1,1,'Essential hypertension','2025-02-15'),
(2,2,4,2,'Meniscal tear, right knee','2025-02-18'),
(3,3,3,3,'Routine vaccination and check-up','2025-03-05'),
(4,5,1,NULL,'Hyperlipidemia management','2025-01-25'),
(5,6,1,5,'Chest pain - atypical, further tests ordered','2025-05-12');

-- Demo prescriptions
INSERT IGNORE INTO prescriptions
(id,treatment_id,medication_id,dosage,frequency,duration_days)
VALUES
(1,1,3,'10 mg','once daily',30),
(2,2,4,'400 mg','every 8 hours',7),
(3,3,2,'250 mg','three times daily',5),
(4,4,6,'20 mg','once daily',90),
(5,5,1,'81 mg','once daily',30);

-- Demo staff / users (doctor-linked users and a staff account)
INSERT IGNORE INTO users
(id,username,email,password_hash,role,linked_doctor_id) VALUES
(1,'admin','admin@hospital.example','$2y$10$6g1jCGd6xxfkxl2bmBNRM.XruYyo7wOoMkQQprfzVNo6oRqplY7gK','admin',NULL),
(2,'m.karim','m.karim@hospital.example','$2y$10$6g1jCGd6xxfkxl2bmBNRM.XruYyo7wOoMkQQprfzVNo6oRqplY7gK','doctor',1),
(3,'r.uddin','r.uddin@hospital.example','$2y$10$6g1jCGd6xxfkxl2bmBNRM.XruYyo7wOoMkQQprfzVNo6oRqplY7gK','doctor',2),
(4,'frontdesk','frontdesk@hospital.example','$2y$10$6g1jCGd6xxfkxl2bmBNRM.XruYyo7wOoMkQQprfzVNo6oRqplY7gK','staff',NULL);
```

Today’s appointments by doctor (time-ordered):

```
SELECT a.id, a.appointment_date, a.status,

p.first_name AS patient_first, p.last_name AS patient_last
```

```
FROM appointments a

JOIN patients p ON p.id = a.patient_id

WHERE DATE(a.appointment_date) = CURDATE() AND a.doctor_id = ?

ORDER BY a.appointment_date ASC;
```

Upcoming 7 days count (optionally per doctor):

```
SELECT COUNT(*)

FROM appointments a

WHERE a.appointment_date > NOW() AND a.appointment_date <=
DATE_ADD(CURDATE(), INTERVAL 7 DAY)

AND (? IS NULL OR a.doctor_id = ?);
```

Open admissions with patient and room:

```
SELECT a.id, a.admitted_on, p.first_name, p.last_name, r.room_number

FROM admissions a

JOIN patients p ON p.id = a.patient_id

JOIN rooms r ON r.id = a.room_id

WHERE a.discharged_on IS NULL

ORDER BY a.admitted_on DESC;
```

Prescriptions for a given patient (via treatments):

```
SELECT pr.id, m.name AS medication, pr.dosage, pr.frequency,
pr.duration_days

FROM prescriptions pr

JOIN treatments t ON t.id = pr.treatment_id

JOIN medications m ON m.id = pr.medication_id

WHERE t.patient_id = ?

ORDER BY pr.id DESC;
```

Available rooms by type:

```
SELECT id, room_number, type

FROM rooms

WHERE status = 'available'

ORDER BY type, room_number;
```

9. Limitations and Future Works

1. **Scheduling:** The current system lacks strict conflict detection for overlapping appointments. Implementing unique constraints or checks per doctor and time slot can prevent double-booking.
2. **Auditing:** No audit trail or versioning is in place. Adding history tables or triggers would ensure clinical and legal traceability.
3. **Soft deletes:** At present, deletes are permanent. Introducing a `deleted_at` column and filtering queries can allow recoverable "soft deletes."

4. **Validation:** Server-side validation is minimal. Extending it to include phone/email format checks and domain-specific rules will improve data integrity.
5. **Authorization granularity:** Access control is limited to roles and doctor ownership. Expanding this to a more detailed permissions matrix will support future growth.
6. **Internationalization/time zones:** Dates are stored as server times only. Incorporating time zones and localization will make the system more globally adaptable.
7. **Reporting:** Currently, there are no advanced reports. Building aggregate reports (e.g., by department, diagnoses, or bed occupancy trends) will enhance decision-making.

10. Conclusion

The **Hospital Management System (HMS)** schema closely follows the ERD, providing normalized tables, strong foreign-key relationships, and well-defined enums to support core hospital workflows. The accompanying PHP/Tailwind application demonstrates this design through complete CRUD operations, role-based views, CSRF protection, and doctor-specific ownership enforcement. With the addition of targeted indexes, conflict-detection mechanisms, and auditing features, the system is well-positioned to scale and adapt to more demanding operational environments.

11. References

1. MySQL 8.0 Reference Manual (InnoDB, FOREIGN KEY constraints)
2. PHP Manual (PDO, prepared statements)
3. Tailwind CSS Documentation
4. OWASP Cheat Sheets (Authentication, CSRF, Input Validation)