Hospital Management System Overview and Tutorial

Section 1: Application Background

Application Overview

This application is a database-driven web project built using Python and the Flask web framework. It is designed to provide a simple yet robust backend for managing and interacting with data stored in a relational database. The project structure is organized to separate concerns, making it easy to maintain, extend, and understand.

Project Structure

The core of the application resides in the app.py file, which serves as the entry point and orchestrates the setup of the Flask application, including routing and configuration. The models.py file defines the database schema using SQLAlchemy, an Object Relational Mapper (ORM) that allows for seamless interaction between Python objects and database tables. The seed_data.py script is used to populate the database with initial data, which is useful for development and testing. HTML templates for rendering web pages are stored in the templates/ directory, following Flask's conventions. The venv/ directory contains the project's virtual environment, isolating dependencies from the global Python installation.

Tools, Libraries, and Languages

The backend is written entirely in Python, using several libraries:

Flask: A lightweight web framework for building web applications and APIs.

Flask-SQLAlchemy: Integrates SQLAlchemy with Flask, providing ORM capabilities and database session management.

psycopg2: A PostgreSQL database adapter for Python, used for connecting to and interacting with a PostgreSQL database.

python-dotenv: Loads environment variables from a .env file, allowing for secure and flexible configuration management.

The application uses Jinja2 templating (bundled with Flask) for rendering dynamic HTML pages. All dependencies are managed via requirements.txt, and a virtual environment (venv/) is used to ensure consistent package versions.

Development Process

To set up the application, developers should first create and activate a Python virtual environment, then install the required dependencies using pip install -r requirements.txt. The database schema is defined in models.py, and can be initialized and seeded with data using the seed_data.py script. The application is started by running app.py, which launches the Flask development server. Routes and business logic are handled within app.py, while database models and relationships are encapsulated in models.py.

Technical Details

The backend leverages SQLAlchemy's ORM features to abstract away raw SQL queries, allowing developers to interact with the database using Python classes and objects. This not only improves code readability but also enhances security by mitigating SQL injection risks. Flask's modular design enables easy addition of new routes, blueprints, and extensions. Environment variables are used for sensitive configuration such as database connection strings, keeping credentials out of the source code. The application is structured to facilitate both development and production deployments, with clear separation between application logic, data models, and presentation templates.

Section 2: Home Page

When you first access the application, you are greeted by the Home Page. This page serves as the central hub and starting point for users, providing a clear overview of the application's purpose and navigation options. The design is intended to be intuitive, ensuring that both new and returning users can easily find their way around.

Test Database Button

One of the key features on the Home Page is the "Test Database" button. This button allows users to quickly verify the connection to the backend database and ensure that the application is functioning correctly. When clicked, the application performs a simple query against the database and displays a message indicating whether the connection was successful. This is particularly useful for troubleshooting and confirming that the backend services are operational before proceeding further.

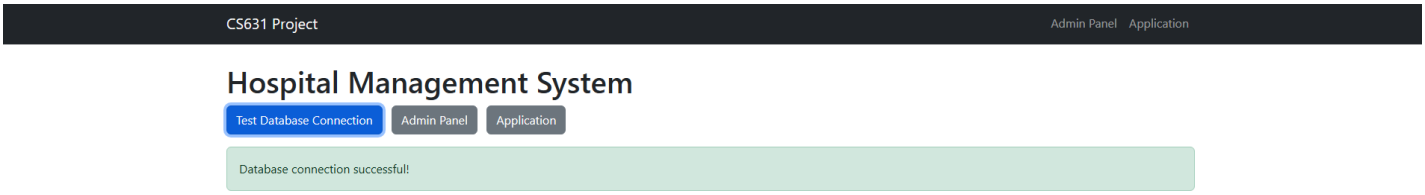
How it works:

When you click the "Test Database" button, the application sends a request to the backend, which attempts to connect to the database and execute a basic query (such as fetching a test record or checking the database status). The result is then displayed on the page, letting you know if the database is accessible.

Navigating to the Application Page

Below or alongside the test functionality, you will find a link or button to proceed to the main Application Page. This is where the core features of the application are accessible, such as viewing, adding, or managing records in the database. The Home Page thus acts as a gateway, ensuring that users can verify system health before engaging with the main application features.

In summary, the Home Page is designed to provide a welcoming introduction, a quick system check via the Test Database button, and a clear path forward to the main application functionality.



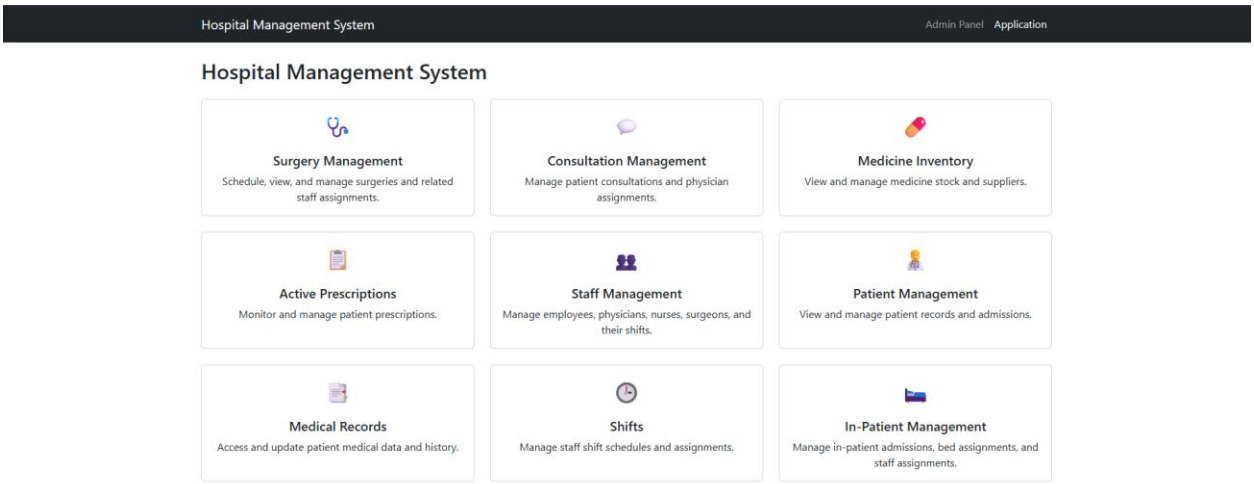
Section 3: Application Page

The application features several management pages, each dedicated to a specific aspect of hospital operations, such as patients, surgeons, surgeries, and more. While the content and user interface elements may differ to suit the needs of each domain, all management

pages share a consistent structure and workflow to ensure a smooth and intuitive user experience.

Each management page typically displays a table listing all current records for that category (e.g., all surgeries, all patients). Users can search, filter, and sort through these records using built-in controls. Common actions such as adding new entries, editing existing ones, or deleting records are available through clearly labeled buttons. When adding or editing a record, a modal form appears, allowing users to input or update the necessary information. The forms are designed to validate input and provide helpful feedback if any required fields are missing or incorrect.

This unified approach across management pages ensures that once users are familiar with one section, they can easily navigate and operate the others with minimal learning curve.



Surgery Management Page:

The Surgery Management page is dedicated to scheduling and managing surgical procedures within the hospital. Upon navigating to this page, users are presented with a comprehensive table listing all scheduled surgeries, including details such as patient name, surgeon, surgery type, date, theater, and shift. The interface is designed to make it easy to review, search, and manage surgeries at a glance.

How to Use the Surgery Management Page

Viewing Surgeries:

All scheduled surgeries are displayed in a table. You can use the search bar to filter surgeries by patient, surgeon, type, or date.

Scheduling a Surgery:

Click the "Schedule Surgery" button to open the scheduling form. Fill in the required details, such as patient, surgeon, surgery type, date, theater, and shift. The form will also prompt you to assign at least two eligible nurses, ensuring proper staffing for the procedure. Once all fields are completed, click "Save" to add the surgery to the schedule.

Editing a Surgery:

To modify an existing surgery, click the "Edit" button next to the relevant entry. The form will open pre-filled with the current details, allowing you to make changes as needed. Save your changes to update the record.

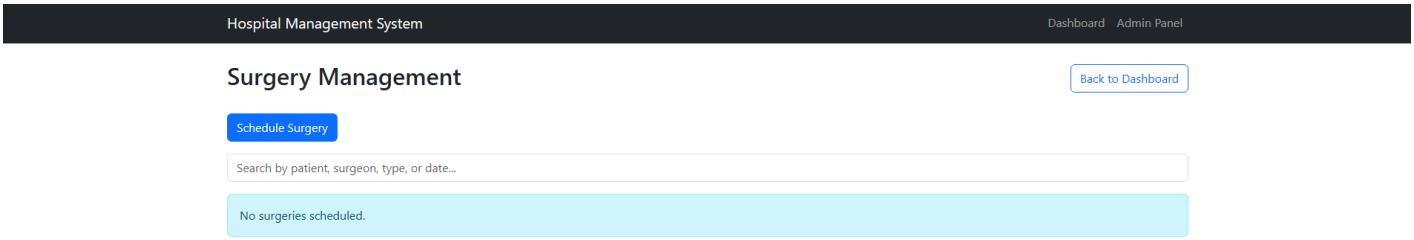
Deleting a Surgery:

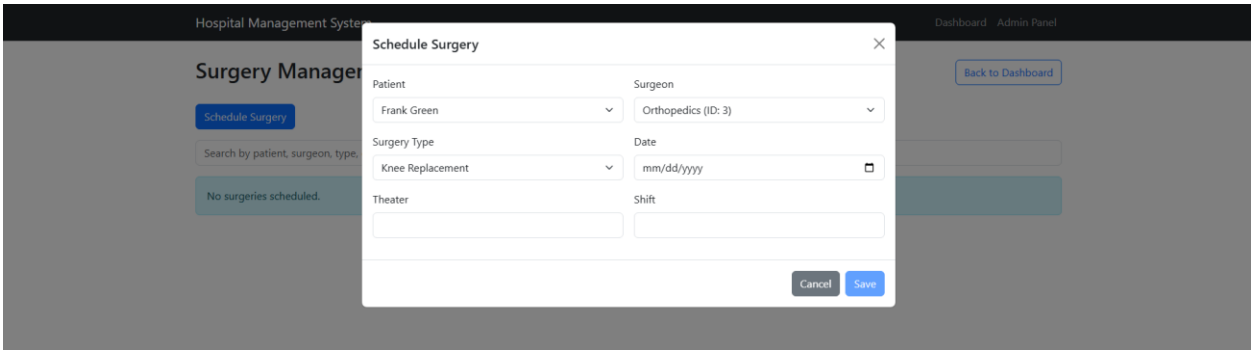
If a surgery needs to be removed, click the "Delete" button. You will be asked to confirm the deletion before the record is permanently removed.

Refreshing Data:

The page automatically updates after any changes, ensuring you always see the most current information.

The Surgery Management page streamlines the process of organizing surgical operations, making it easy to coordinate between patients, surgeons, and nursing staff, and ensuring that all necessary details are captured for each procedure.





Consultation Management

The Consultation Management page is designed to help users efficiently schedule and manage patient consultations within the hospital. This page provides a clear overview of all upcoming and past consultations, making it easy to keep track of appointments and ensure that patients receive timely care.

How to Use the Consultation Management Page

Viewing Consultations:

The main table displays all consultations, including details such as patient name, consulting doctor, date, time, and consultation room. You can use the search bar to quickly filter consultations by patient, doctor, or date.

Scheduling a Consultation:

To add a new consultation, click the "Schedule Consultation" button. This opens a form where you can select the patient, assign a doctor, choose a date and time, and specify the consultation room. Complete all required fields and click "Save" to schedule the consultation.

Editing a Consultation:

If you need to update the details of a consultation, click the "Edit" button next to the relevant entry. The form will appear with the current information pre-filled, allowing you to make any necessary changes. Save your updates to apply them.

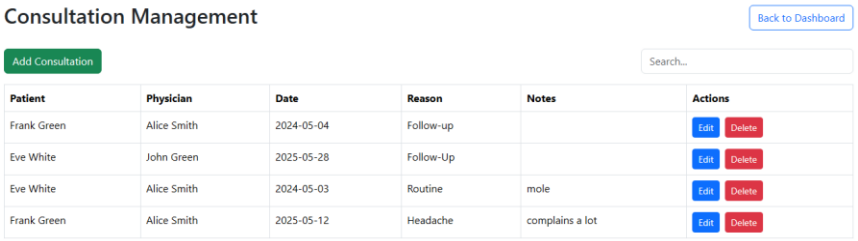
Deleting a Consultation:

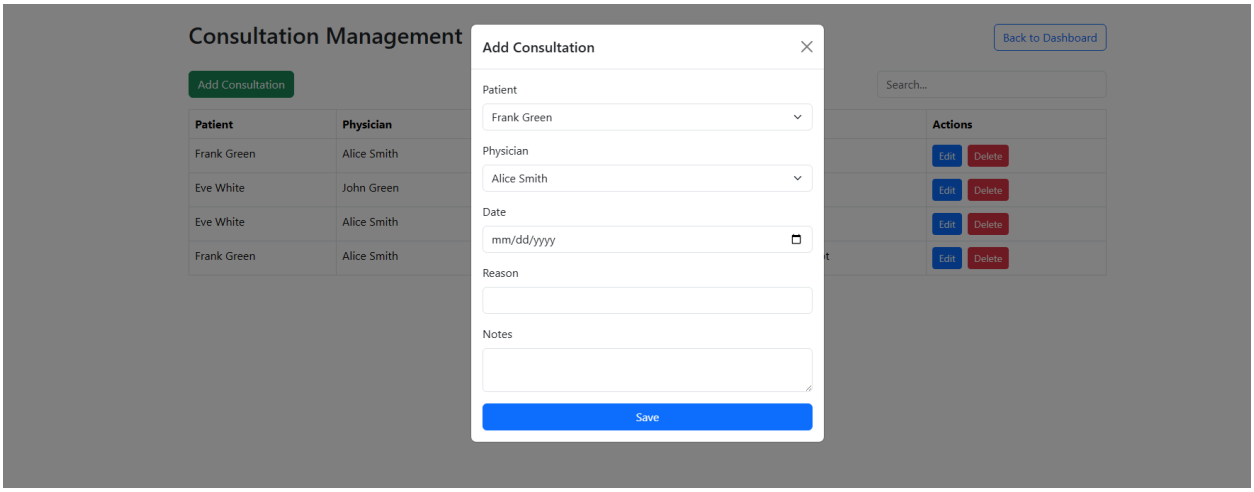
To remove a consultation from the schedule, click the "Delete" button. You will be prompted to confirm the deletion before the record is permanently removed.

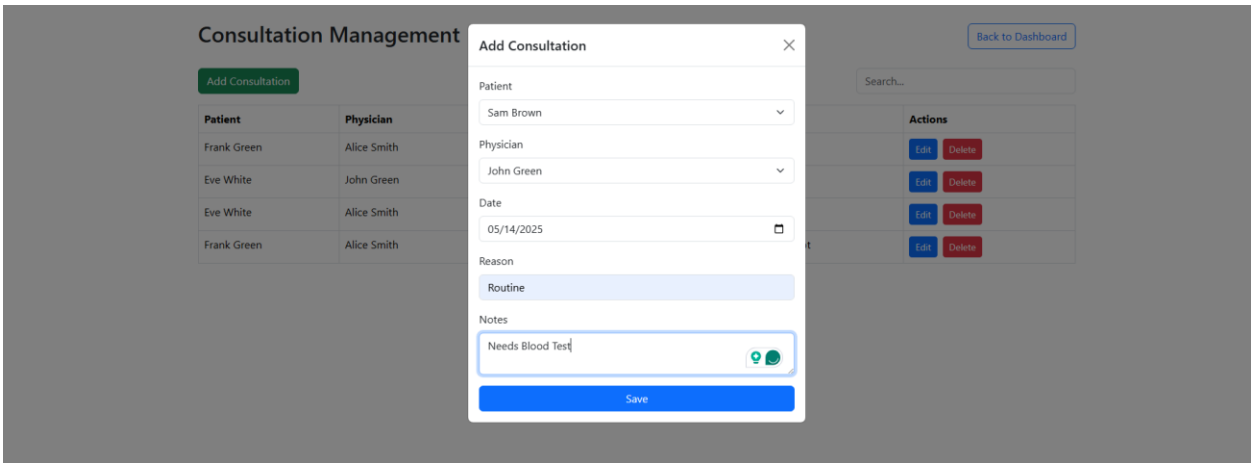
Refreshing Data:

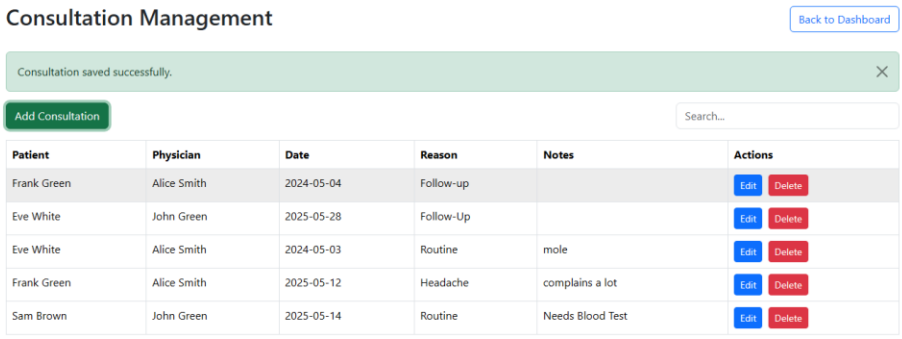
After any changes, the page will automatically refresh to display the most up-to-date list of consultations.

The Consultation Management page ensures that all appointments are organized and accessible, helping staff coordinate schedules and provide the best possible care to patients.









Medicine Inventory

The Medicine Inventory Management page provides a centralized interface for tracking and managing the hospital's medicine stock. This page is designed to help staff monitor inventory levels, update medicine details, and ensure that essential medications are always available when needed.

How to Use the Medicine Inventory Management Page

Viewing Medicines:

The main table lists all medicines currently in the inventory, displaying details such as code, name, manufacturer, usage, description, unit cost, quantity in stock, and quantity ordered. You can use the search bar to quickly filter medicines by any of these fields.

Adding a New Medicine:

To add a new medicine to the inventory, click the "Add Medicine" button. This opens a form where you can enter all relevant details, including the medicine code, name, manufacturer,

usage, description, unit cost, quantity, and quantity ordered. Complete the form and click "Save" to add the medicine to the inventory.

Editing Medicine Details:

To update the information for an existing medicine, click the "Edit" button next to the corresponding entry in the table. The form will appear pre-filled with the current details, allowing you to make any necessary changes. After editing, click "Save" to update the record.

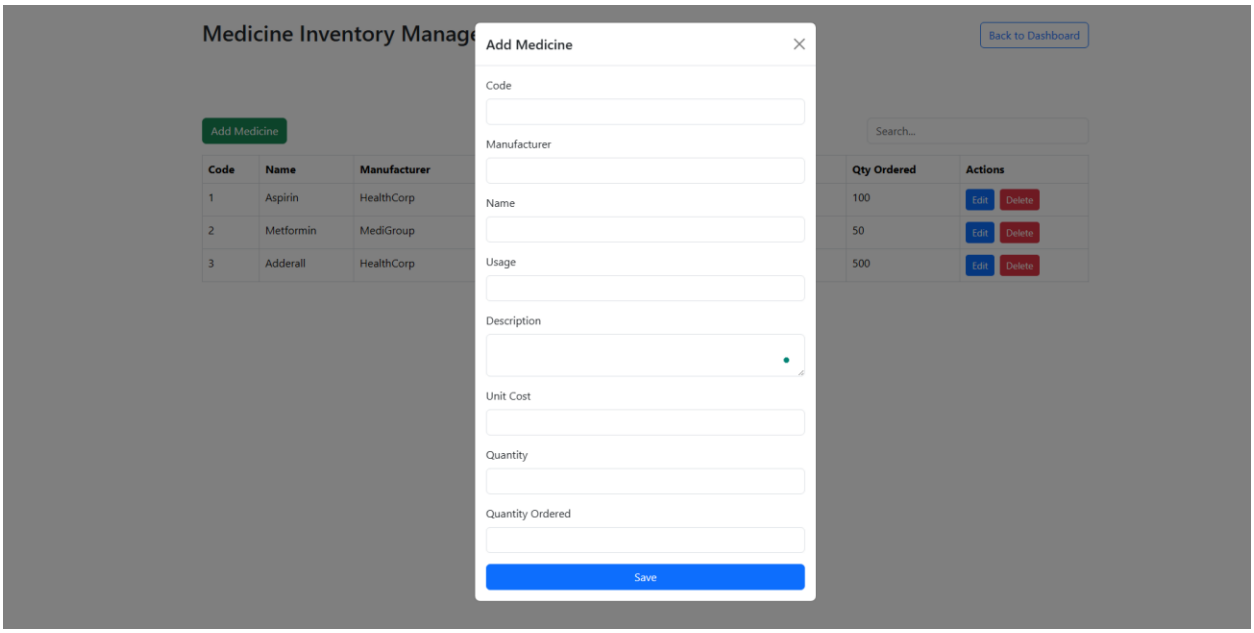
Deleting a Medicine:

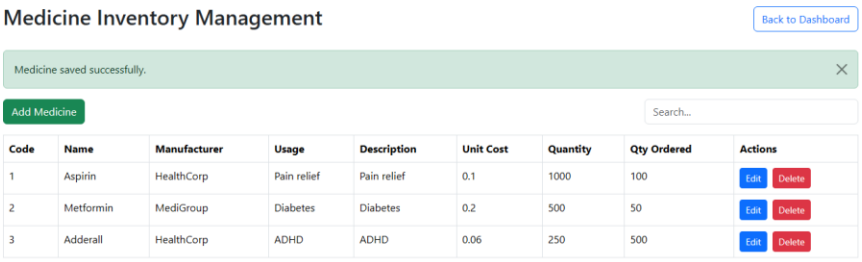
If a medicine is no longer needed or was entered by mistake, you can remove it from the inventory by clicking the "Delete" button. You will be prompted to confirm the deletion before the medicine is permanently removed from the system.

Refreshing Data:

The page automatically refreshes after any changes, ensuring that the displayed inventory is always up to date.

The Medicine Inventory Management page streamlines the process of maintaining accurate medicine records, making it easy to add, update, or remove medicines as needed. This helps ensure that the hospital's pharmacy is well-stocked and that staff can quickly locate and manage essential medications.





Active Prescriptions

The Active Prescriptions Management page provides a streamlined interface for tracking and managing all current prescriptions within the hospital. This page is essential for ensuring that patients receive the correct medications as prescribed by their healthcare providers, and for keeping an up-to-date record of ongoing treatments.

How to Use the Active Prescriptions Management Page

Viewing Active Prescriptions:

The main table displays all active prescriptions, including details such as patient name, medicine, prescriber, date prescribed, dosage, duration, frequency, and quantity. By default, only prescriptions that are currently active (i.e., not expired) are shown. You can use the search bar to quickly filter prescriptions by patient, medicine, prescriber, or any other field.

Adding a New Prescription:

To add a new prescription, click the "Add Prescription" button. This opens a form where you can select the patient, choose the medicine, assign a prescriber, and enter details such as start date, dosage, duration, frequency, and quantity. Complete all required fields and click "Save" to add the prescription to the system.

Editing a Prescription:

To update an existing prescription, click the "Edit" button next to the relevant entry in the table. The form will appear pre-filled with the current prescription details, allowing you to make any necessary changes. After editing, click "Save" to update the record.

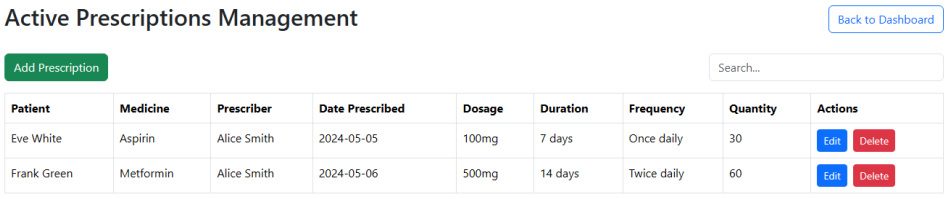
Deleting a Prescription:

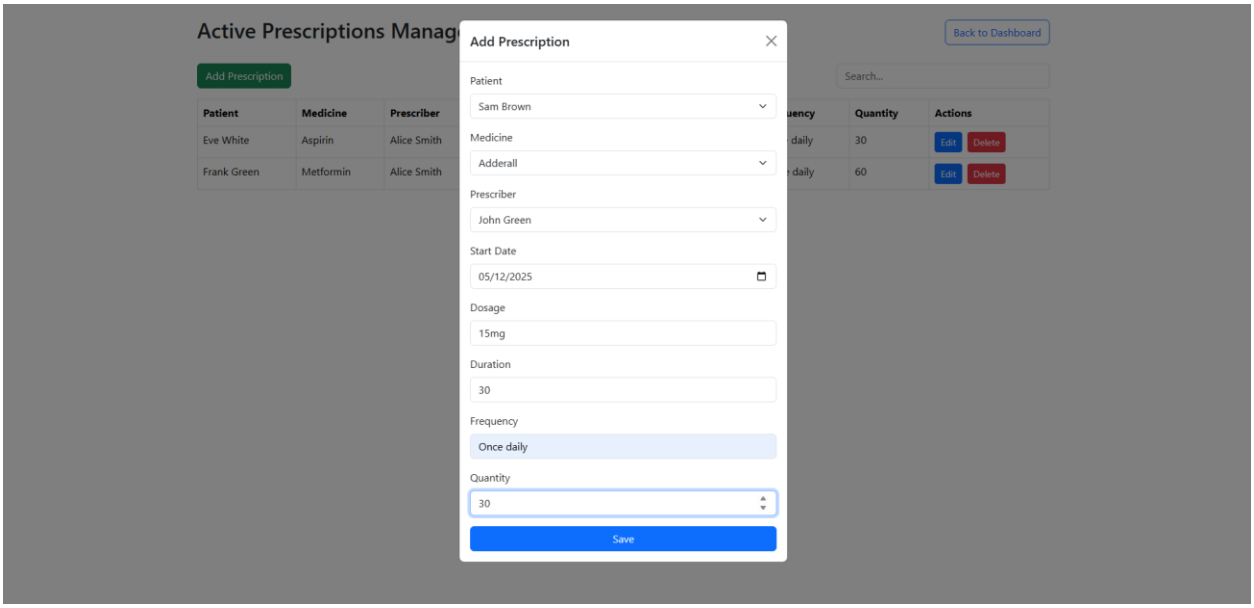
If a prescription needs to be removed (for example, if it was entered in error or is no longer needed), click the "Delete" button. You will be prompted to confirm the deletion before the prescription is permanently removed from the system.

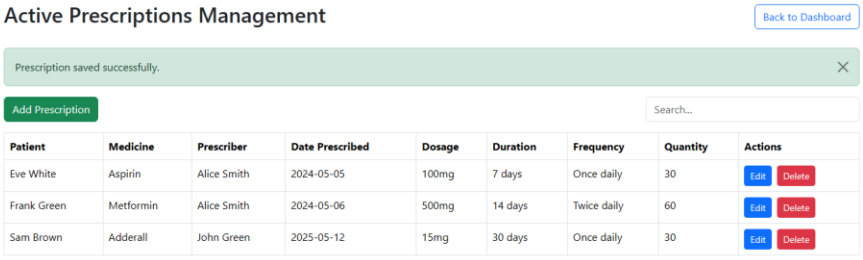
Refreshing Data:

The page automatically refreshes after any changes, ensuring that the list of active prescriptions is always current.

The Active Prescriptions Management page helps healthcare staff maintain accurate and up-to-date records of all ongoing prescriptions, making it easy to add, update, or remove prescriptions as needed. This ensures that patients receive the correct medications and that the hospital's records remain organized and reliable.







Staff Management

The Staff Management page provides a comprehensive interface for managing all hospital employees, including physicians, nurses, surgeons, and support staff. This page is designed to help administrators keep accurate records of all staff members, their roles, and their contact information, while also supporting the unique requirements of each staff type.

How to Use the Staff Management Page

Viewing Staff:

The page features a summary of staff counts by type (Physicians, Nurses, Surgeons, Support) at the top, followed by a tabbed interface. You can view all staff or filter by specific categories using the tabs. Each tab displays a searchable, sortable table of staff members, showing key details such as name, type, contact information, and actions.

Adding a New Staff Member:

To add a new staff member, click the "Add Staff" button (or the corresponding button in each tab, such as "Add Nurse" or "Add Physician"). This opens a form where you enter the staff member's personal, contact, and address information.

Dynamic Fields Based on Employee Type:

When you select the staff member's type (Physician, Nurse, Surgeon, or Support), the form dynamically displays additional fields specific to that role:

Nurse: Grade and Years of Experience

Physician: Specialty and Physician Type

Surgeon: Specialty, Contract Type, Contract Duration, and Contract Amount

Support: No extra fields beyond the standard information

This ensures that all relevant details are captured for each staff member, tailored to their responsibilities within the hospital.

Editing Staff Information:

To update a staff member's information, use the "Edit" button next to their entry. The form will appear pre-filled with their current details, including any type-specific fields. Make your changes and save to update the record.

Deleting Staff:

To remove a staff member, click the "Delete" button. You will be prompted to confirm before the staff member is permanently removed from the system.

Searching and Filtering:

Each tab includes a search bar, allowing you to quickly find staff by name, type, SSN, or other attributes.

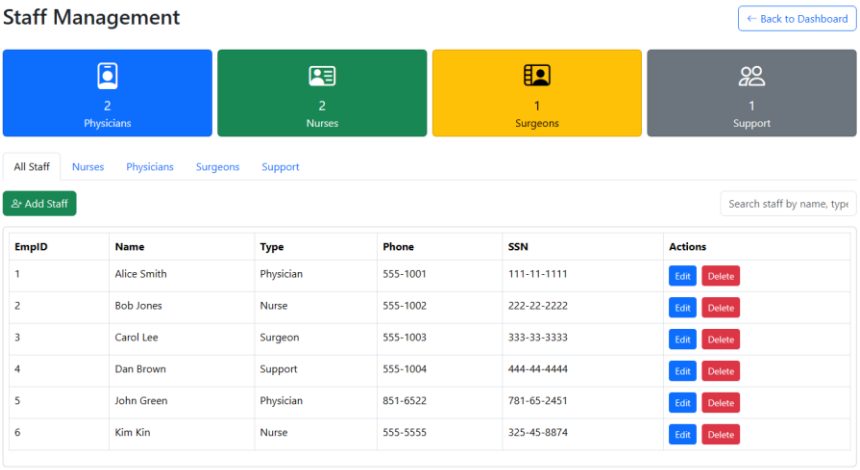
Inline Editing:

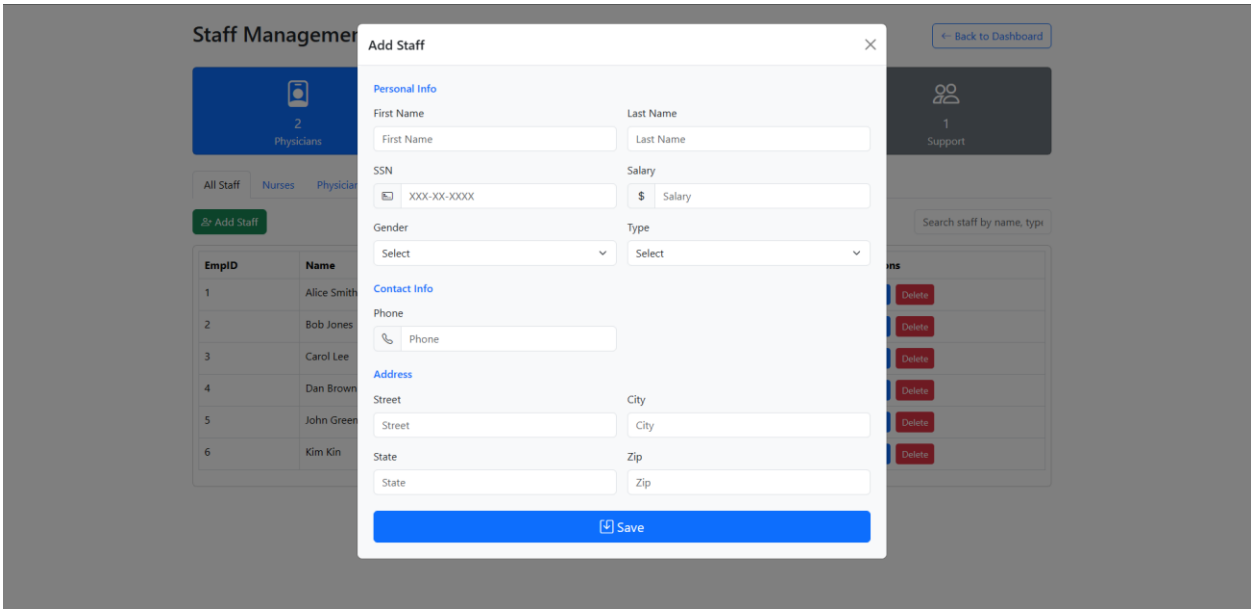
For convenience, some fields can be edited directly within the table using inline editing controls, streamlining quick updates for common fields.

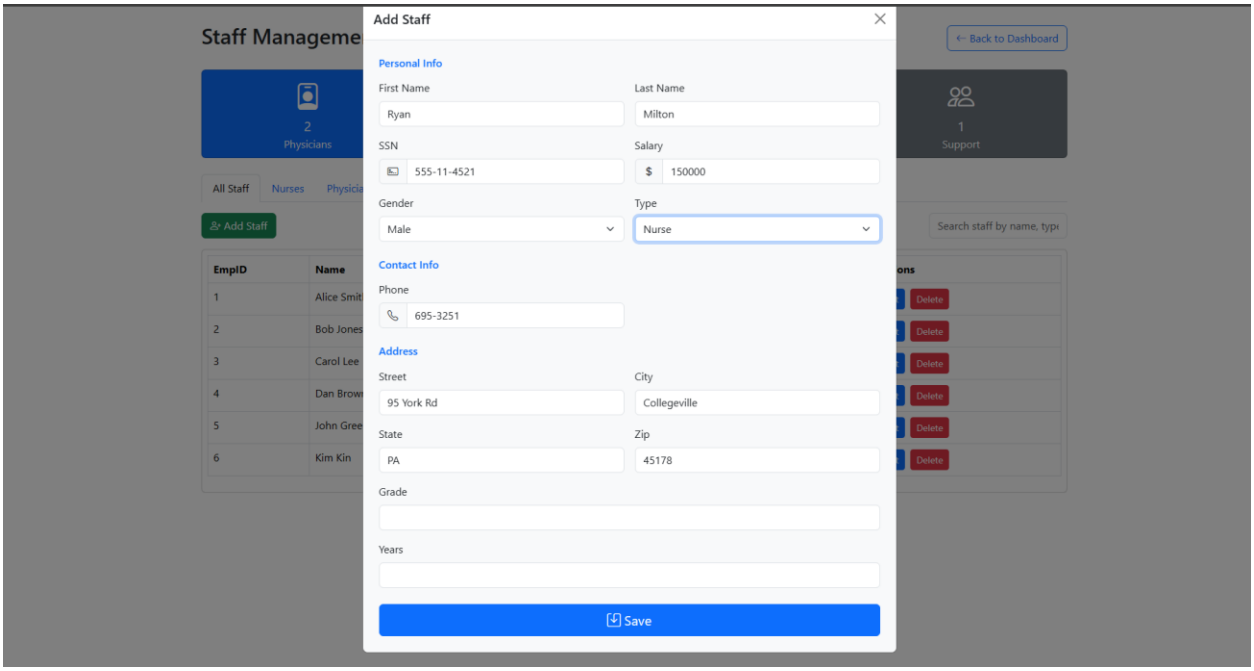
Refreshing Data:

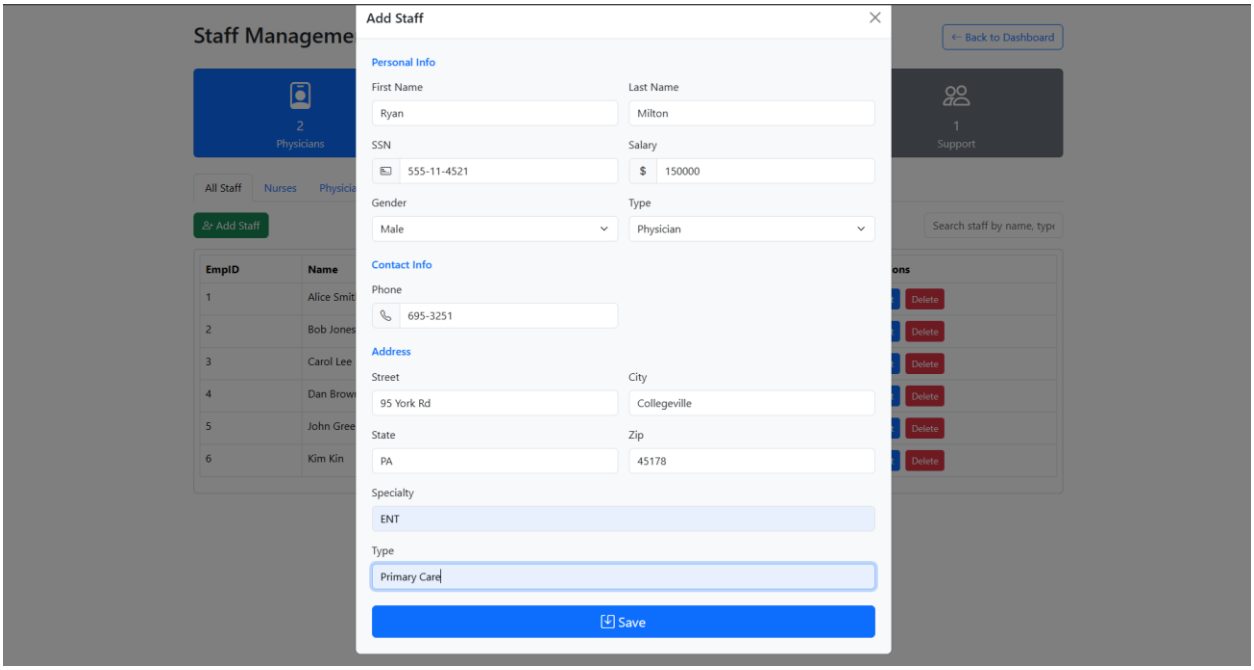
The page automatically refreshes after any changes, ensuring that the staff directory is always up to date.

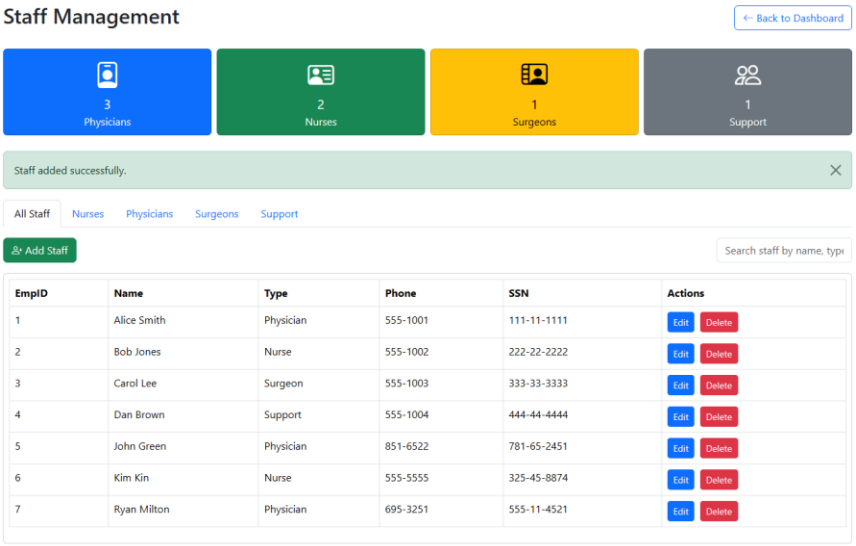
The Staff Management page is a powerful tool for maintaining a complete and accurate record of all hospital personnel. Its dynamic forms and intuitive interface make it easy to add, update, or remove staff, while ensuring that all necessary information—general and role-specific—is always collected and organized.






Staff Manageme
Back to Dashboard
2
Physicians
1
Support
All Staff
Nurses
Physicia
Add Staff
Search staff by name, typ
EmpID
Name
1
Alice Smit
2
Bob Jones
3
Carol Lee
4
Dan Brow
5
John Gree
6
Kim Kin
Delete
Add Staff
Personal Info
First Name
Ryan
Last Name
Milton
SSN
555-11-4521
Salary
$
150000
Gender
Male
Type
Nurse
Contact Info
Phone
695-3251
Address
Street
95 York Rd
City
Collegeville
State
PA
Zip
45178
Grade
Years
Save


Staff Manageme
Back to Dashboard
2
Physicians
1
Support
All Staff
Nurses
Physicia
Add Staff
Search staff by name, typ
EmpID
Name
1
Alice Smit
2
Bob Jones
3
Carol Lee
4
Dan Brow
5
John Gree
6
Kim Kin
Delete
Add Staff
Personal Info
First Name
Ryan
Last Name
Milton
SSN
555-11-4521
Salary
$
150000
Gender
Male
Type
Physician
Contact Info
Phone
695-3251
Address
Street
95 York Rd
City
Collegeville
State
PA
Zip
45178
Specialty
ENT
Type
Primary Care
Save



Patient Management

The Patient Management page is the central hub for handling all patient-related information within the hospital system. It allows staff to add new patients, search and view existing patients, update patient details, and manage appointments—all from a single, intuitive interface.

How to Use the Patient Management Page

Adding a New Patient:

At the top of the page, you'll find a form labeled "Insert a New Patient". Fill in the required fields, including SSN, first and last name, date of birth, gender, blood type, address, and primary physician. The primary physician dropdown is dynamically populated with available doctors. Once all fields are complete, click "Add Patient" to register the new patient in the system. A confirmation message will appear upon successful addition.

Searching and Viewing Patients:

Use the search bar to quickly find patients by name, SSN, or other details. The results are displayed as cards, each showing the patient's name, SSN, and location. Click "View Details" on any card to open a detailed modal with comprehensive patient information.

Viewing and Editing Patient Details:

In the patient details modal, you can see and (if needed) edit the patient's personal information, such as name, gender, blood type, address, and primary physician. Click "Edit" to enable editing, make your changes, and then click "Save" to update the record.

You can also view and manage the patient's allergies, diagnoses, illnesses, and medical data directly from this modal.

Managing Allergies, Diagnoses, and Medical Data:

The details modal provides sections for allergies, diagnoses, illnesses, and medical data (such as blood sugar, cholesterol, etc.). You can add or remove allergies, and view a history of diagnoses and illnesses. This ensures that all relevant medical information is easily accessible and up to date.

Scheduling Appointments:

The "Schedule an Appointment with a Doctor" section allows you to book new consultations for patients. Select the patient and physician, choose a date, and specify the type or reason for the appointment. Click "Schedule Appointment" to add it to the system.

Viewing Scheduled Appointments:

The "Scheduled Appointments" section displays all upcoming and past appointments, grouped by doctor and date. This makes it easy to see each doctor's schedule and the types of appointments booked.

Refreshing Data:

The page automatically updates after any changes, ensuring that all patient and appointment information is current.

The Patient Management page is designed to streamline the process of registering, updating, and tracking patients and their medical interactions. Its comprehensive features and user-friendly layout make it easy for hospital staff to maintain accurate and complete patient records.

Patient Management

← Back to Dashboard

Insert a New Patient

SSN | First Name | Last Name | Primary Physician

mm/dd/yyyy | Street | City | State (e.g. NY)

ZIP | Gender | Blood Type

Add Patient

Search patients... Search

Frank Green
SSN: 666-66-6666
City: Gotham, **State:** NJ
View Details

Sam Brown
SSN: 468-48-7821
City: New York, **State:** NY
View Details

Eve White
SSN: 555-55-5555
City: Metropolis, **State:** NY
View Details

Schedule an Appointment with a Doctor

Select Patient | Select Doctor | mm/dd/yyyy | Type/Reason

Schedule Appointment

Schedule an Appointment with a Doctor

Select Patient | Select Doctor | mm/dd/yyyy | Type/Reason

Schedule Appointment

Scheduled Appointments

Doctor: Alice Smith

Date: 2024-05-04

Patient	Type/Reason
Frank Green	Follow-up

Date: 2024-05-03

Patient	Type/Reason
Eve White	Routine

Date: 2025-05-12

Patient	Type/Reason
Frank Green	Headache

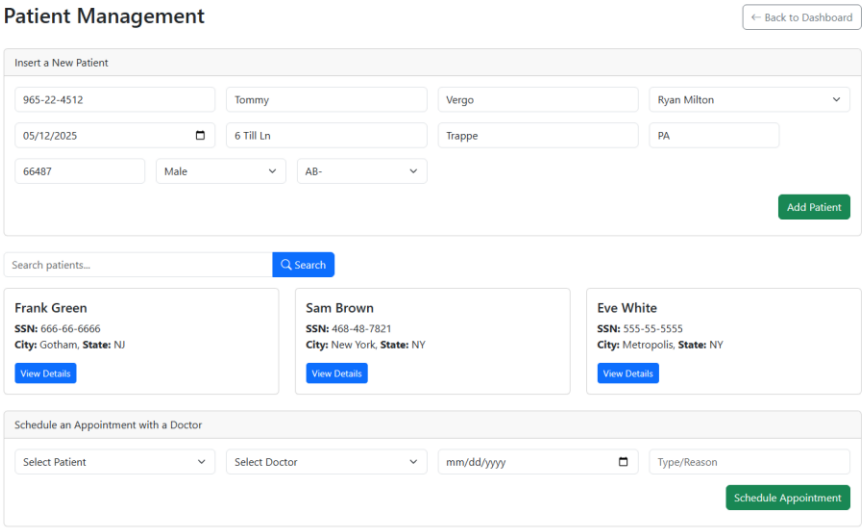
Doctor: John Green

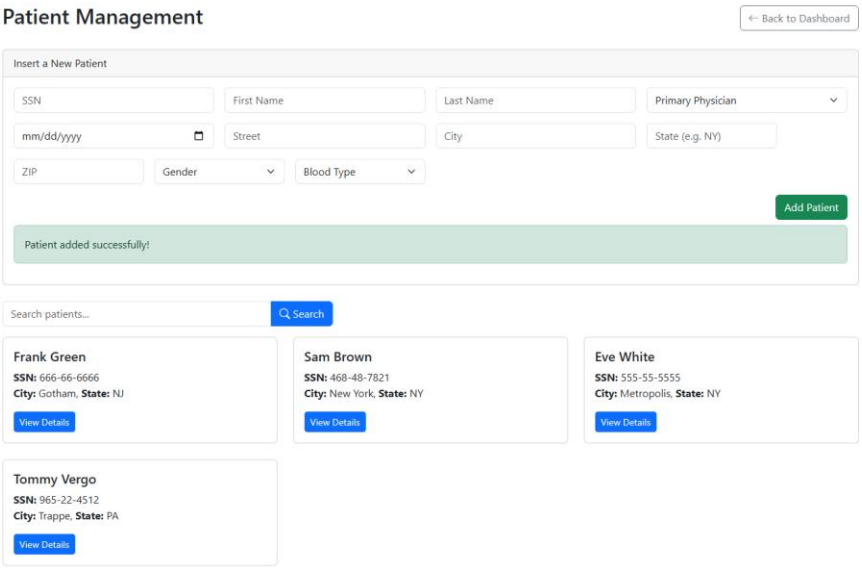
Date: 2025-05-28

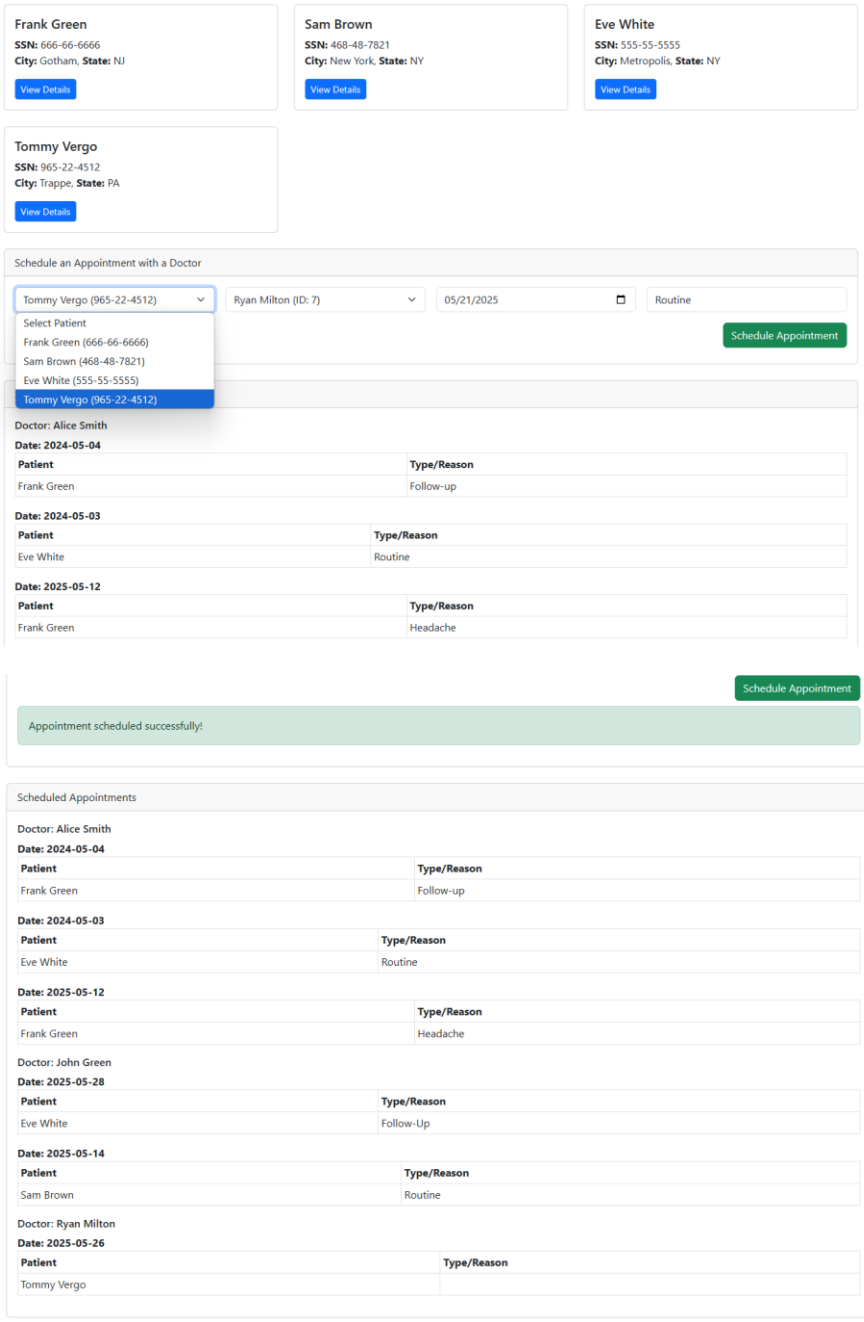
Patient	Type/Reason
Eve White	Follow-Up

Date: 2025-05-14

Patient	Type/Reason
Sam Brown	Routine


Patient Management
← Back to Dashboard
Insert a New Patient
965-22-4512
Tommy
Vergo
Ryan Milton
05/12/2025
6 Till Ln
Trappe
PA
66487
Male
AB-
Add Patient
Search patients...
Search
Frank Green
SSN: 666-66-6666
City: Gotham, State: NJ
View Details
Sam Brown
SSN: 468-48-7821
City: New York, State: NY
View Details
Eve White
SSN: 555-55-5555
City: Metropolis, State: NY
View Details
Schedule an Appointment with a Doctor
Select Patient
Select Doctor
mm/dd/yyyy
Type/Reason
Schedule Appointment


Patient Management
← Back to Dashboard
Insert a New Patient
SSN
First Name
Last Name
Primary Physician
mm/dd/yyyy
Street
City
State (e.g. NY)
ZIP
Gender
Blood Type
Add Patient
Patient added successfully!
Search patients...
Search
Frank Green
SSN: 666-66-6666
City: Gotham, State: NJ
View Details
Sam Brown
SSN: 468-48-7821
City: New York, State: NY
View Details
Eve White
SSN: 555-55-5555
City: Metropolis, State: NY
View Details
Tommy Vergo
SSN: 965-22-4512
City: Trappe, State: PA
View Details



Medical Records Management

The Medical Records Management page provides a centralized location for viewing and updating detailed medical information for each patient. This page is essential for maintaining a comprehensive and accurate history of a patient's health, diagnoses, treatments, and test results.

How to Use the Medical Records Management Page

Accessing Medical Records:

You can access a patient’s medical records directly from the Patient Management page by clicking the "View Medical Records" button in the patient details modal. This will take you to the Medical Records Management page for the selected patient.

Viewing Medical Data:

The page displays a summary of the patient’s key medical data, such as blood sugar, cholesterol, triglycerides, HDL, LDL, and other relevant health metrics. This information is typically presented in a clear, tabular or card-based format for easy review.

Reviewing Diagnoses and Illnesses:

The medical records page includes a chronological list of all diagnoses and illnesses associated with the patient. Each entry provides details such as the diagnosis description, date, notes, and any associated comments. This helps healthcare providers quickly understand the patient’s medical history and ongoing conditions.

Managing Allergies:

Allergies are also listed as part of the medical record. You can add new allergies or remove existing ones, ensuring that the patient’s allergy information is always up to date and visible to all relevant staff.

Editing Medical Records:

If you have the appropriate permissions, you can update medical data fields, add new diagnoses or illnesses, and manage allergies directly from this page. Simply click the "Edit" or "Add" buttons where available, make your changes, and save to update the patient’s record.

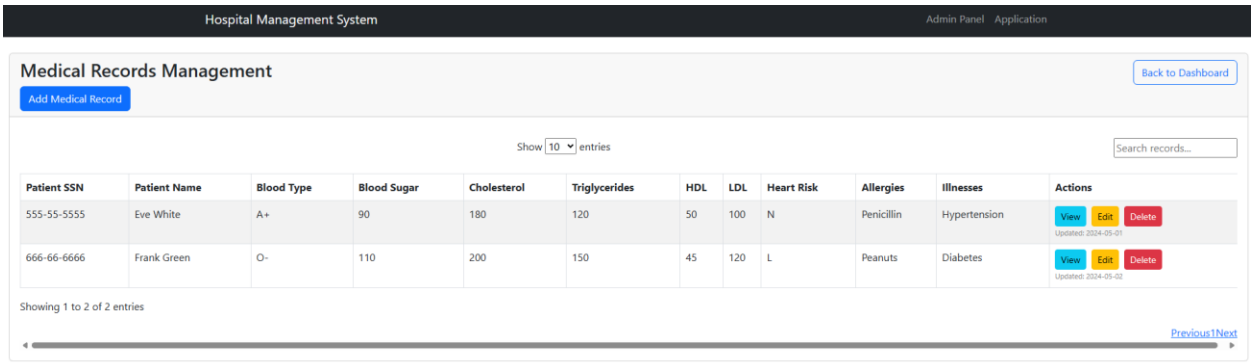
Navigation and Integration:

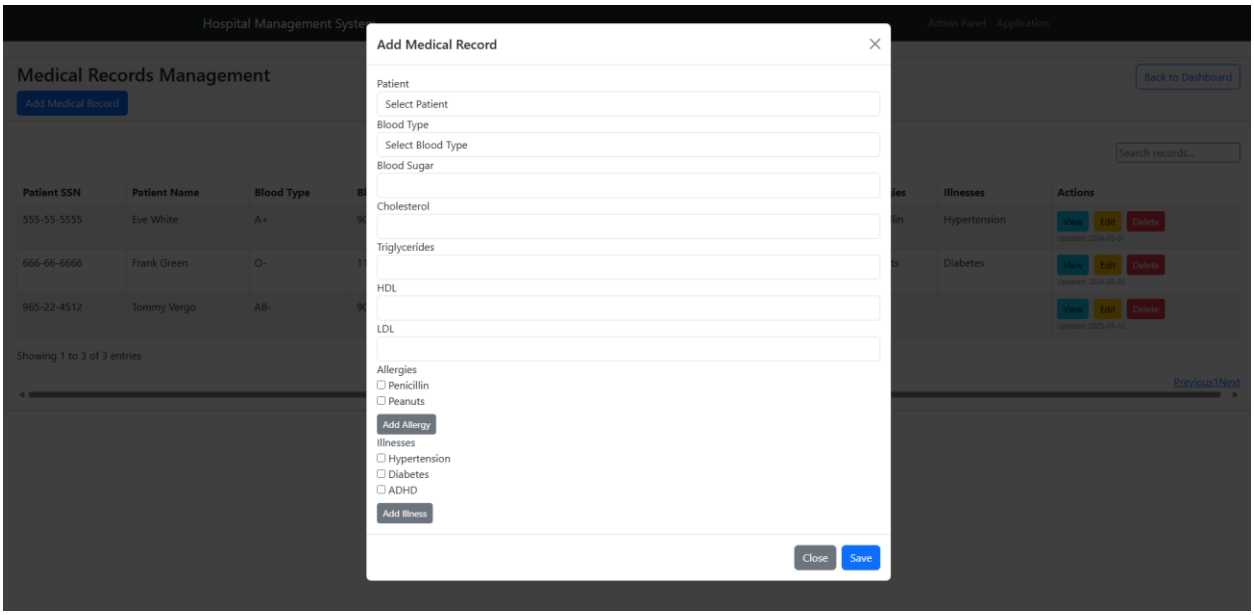
The Medical Records Management page is tightly integrated with other parts of the application. You can easily navigate back to the Patient Management page or other related sections using the provided navigation links or buttons.

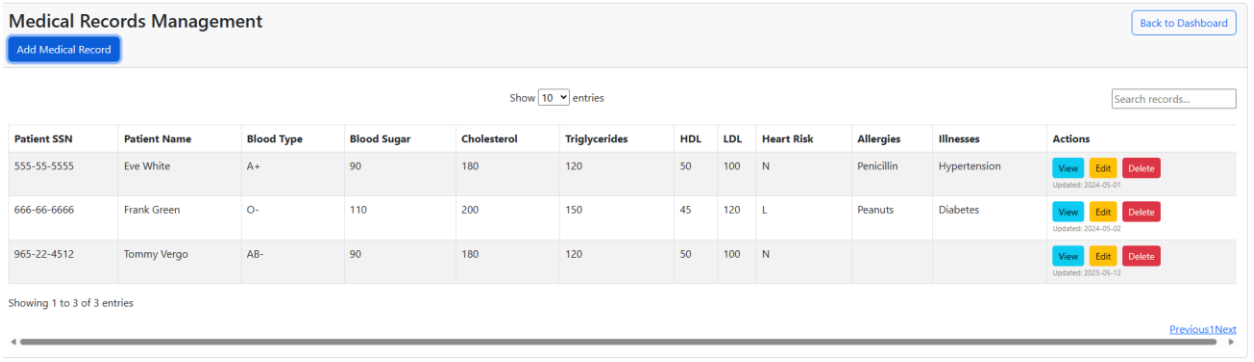
Refreshing Data:

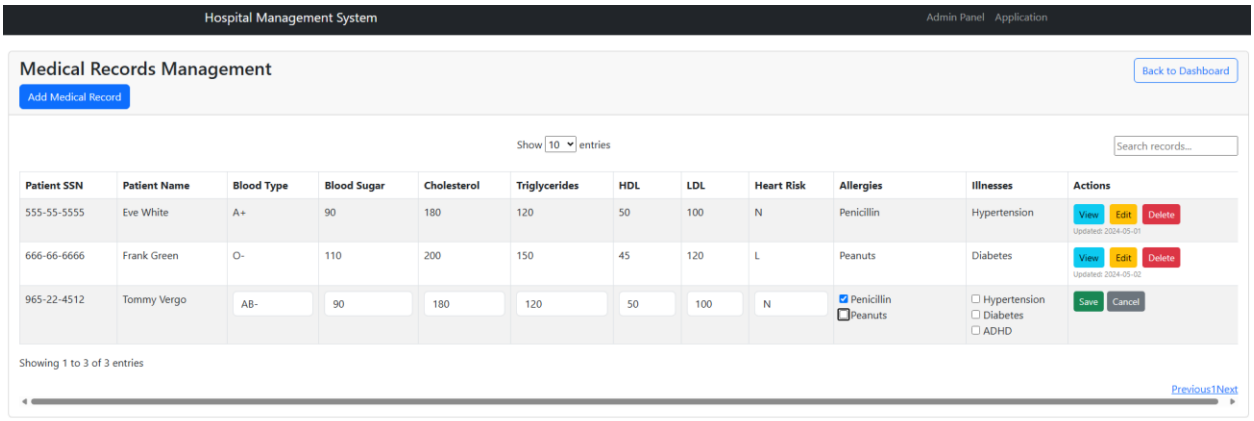
The page automatically refreshes after any changes, ensuring that all displayed information is current and accurate.

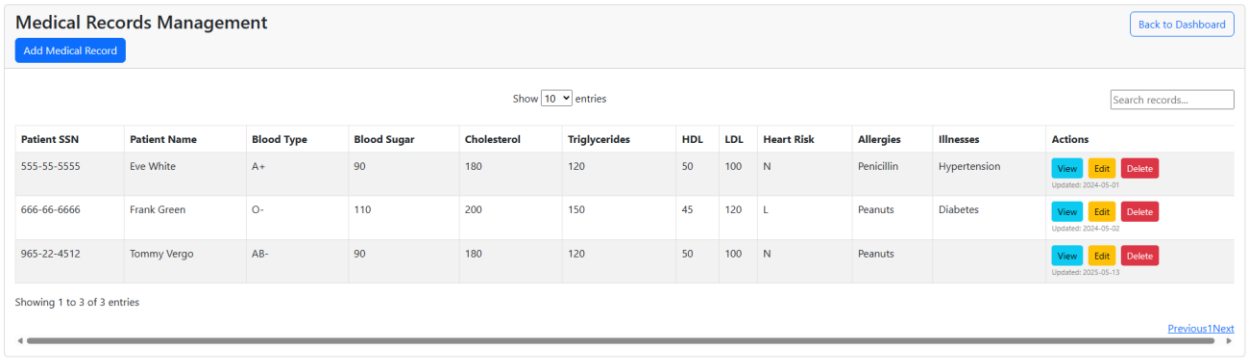
The Medical Records Management page is a vital tool for healthcare providers, enabling them to maintain a complete and up-to-date view of each patient’s health status. By centralizing all relevant medical information, the system supports better clinical decision-making and improves the quality of patient care.


Hospital Management System
Admin Panel
Application
Medical Records Management
Add Medical Record
Back to Dashboard
Show 10 entries
Search records...
Patient SSN
Patient Name
Blood Type
Blood Sugar
Cholesterol
Triglycerides
HDL
LDL
Heart Risk
Allergies
Illnesses
Actions
555-55-5555
Eve White
A+
90
180
120
50
100
N
Penicillin
Hypertension
View
Edit
Delete
666-66-6666
Frank Green
O-
110
200
150
45
120
L
Peanuts
Diabetes
Showing 1 to 2 of 2 entries
Previous1Next


Add Medical Record
Patient
Select Patient
Blood Type
Select Blood Type
Blood Sugar
Cholesterol
Triglycerides
HDL
LDL
Allergies
Penicillin
Peanuts
Add Allergy
Illnesses
Hypertension
Diabetes
ADHD
Add Illness
Close
Save


Medical Records Management
Add Medical Record
Back to Dashboard
Show 10 entries
Search records...
Patient SSN
Patient Name
Blood Type
Blood Sugar
Cholesterol
Triglycerides
HDL
LDL
Heart Risk
Allergies
Illnesses
Actions
555-55-5555
Eve White
A+
90
180
120
50
100
N
Penicillin
Hypertension
View
Edit
Delete
666-66-6666
Frank Green
O-
110
200
150
45
120
L
Peanuts
Diabetes
965-22-4512
Tommy Vergo
AB-
90
180
120
50
100
N
Showing 1 to 3 of 3 entries
Previous1Next





Shift Management

The Shifts Management page is designed to help hospital administrators and staff efficiently organize and oversee work schedules for all employees. This page provides a clear overview of current and upcoming shifts, making it easy to ensure that the hospital is always adequately staffed.

How to Use the Shifts Management Page

Viewing Shifts:

The main table on this page displays all scheduled shifts, including details such as the staff member's name, role (e.g., nurse, physician, surgeon, support), shift date, start and end times, and assigned department or location. You can use the search bar to quickly filter shifts by staff name, date, or department.

Adding a New Shift:

To schedule a new shift, click the "Add Shift" button. This opens a form where you can select the staff member, specify the date, start and end times, and assign the shift to a

particular department or area. Complete all required fields and click "Save" to add the shift to the schedule.

Editing a Shift:

To update an existing shift, click the "Edit" button next to the relevant entry. The form will appear pre-filled with the current shift details, allowing you to make any necessary changes. After editing, click "Save" to update the shift.

Deleting a Shift:

If a shift needs to be removed or was scheduled in error, click the "Delete" button. You will be prompted to confirm the deletion before the shift is permanently removed from the schedule.

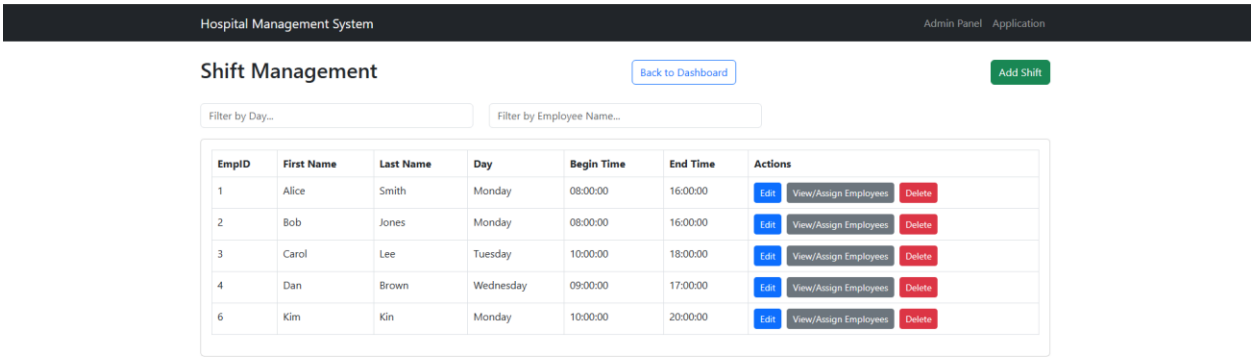
Refreshing Data:

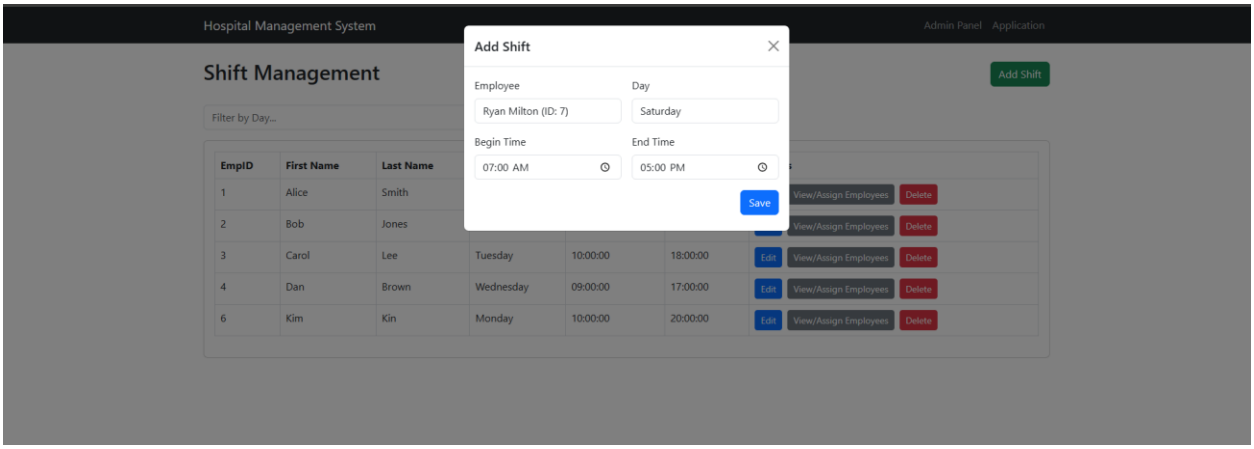
The page automatically refreshes after any changes, ensuring that the shift schedule is always up to date.

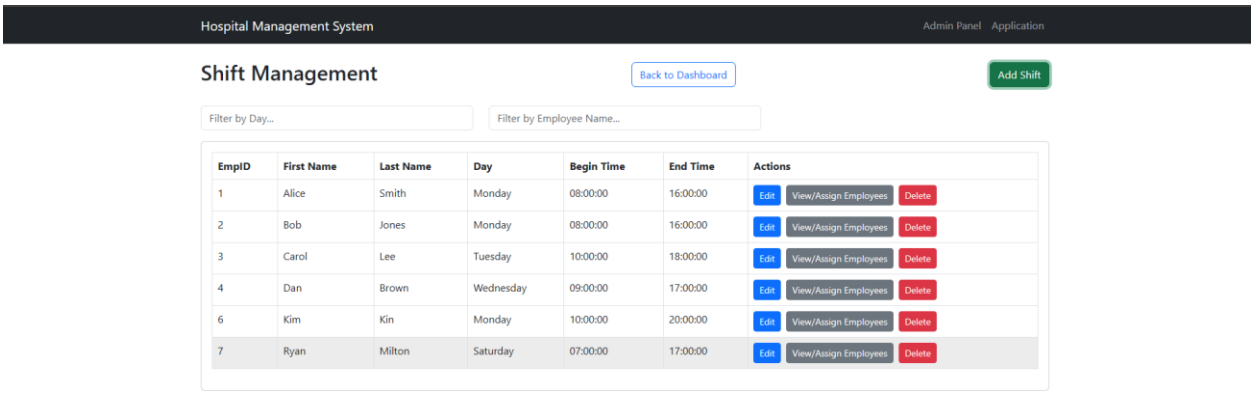
Ensuring Coverage:

The Shifts Management page helps administrators quickly identify gaps in coverage and make adjustments as needed, supporting smooth hospital operations and optimal patient care.

The Shifts Management page streamlines the process of scheduling, updating, and tracking staff shifts, making it easy to maintain a well-organized and responsive workforce.







In-Patient Management

The In-Patient Management page is designed to help hospital staff efficiently manage patients who are currently admitted to the hospital. This page provides a comprehensive overview of bed availability, current in-patients, and staff assignments, as well as tools for admitting patients and managing their care teams.

How to Use the In-Patient Management Page

Viewing Available Beds:

The left side of the page displays a table of all available beds, including details such as clinic, room, bed, unit, and wing. This helps staff quickly identify where new patients can be admitted.

Viewing Current In-Patients:

The main table lists all patients currently admitted to the hospital, showing their assigned bed, admission and discharge dates, and available actions (such as removing an admission).

Admitting a Patient:

Use the "Admit Patient to Bed" form to assign a patient to an available bed. Select the patient and bed from the dropdowns, specify the admission date (and optional discharge date), and click "Admit". The system will update the list of in-patients and available beds automatically.

Assigning Staff to Patients:

The "Assign Doctor/Nurse to Patient" form allows you to assign a primary doctor and one or more nurses to a patient. Select the patient, choose a doctor (optional), and select a nurse from the list. Click "Assign" to update the patient's care team. The system prevents assigning the same nurse to a patient more than once.

Viewing and Managing Assignments:

The "Current Assignments" section displays each patient's assigned doctor and nurses. You can remove a doctor or nurse from a patient using the provided buttons, ensuring that assignments are always up to date.

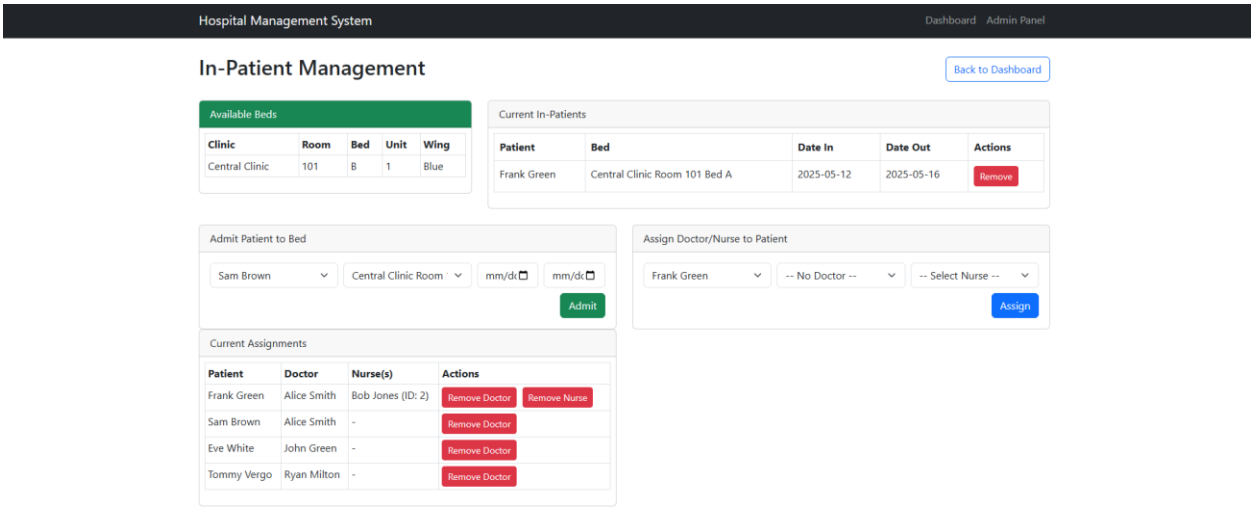
Removing Admissions:

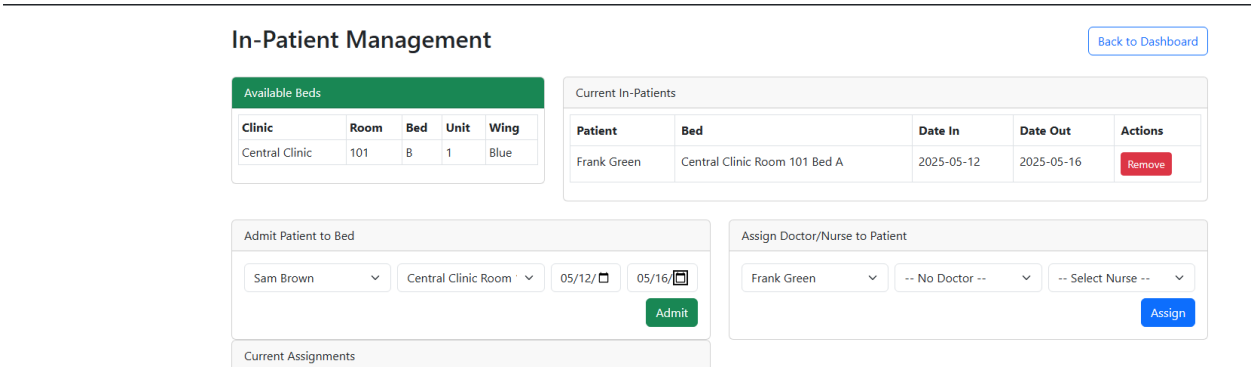
If a patient is discharged or needs to be removed from their bed assignment, use the "Remove" button in the in-patient table. This will free up the bed for new admissions and update the patient's status.

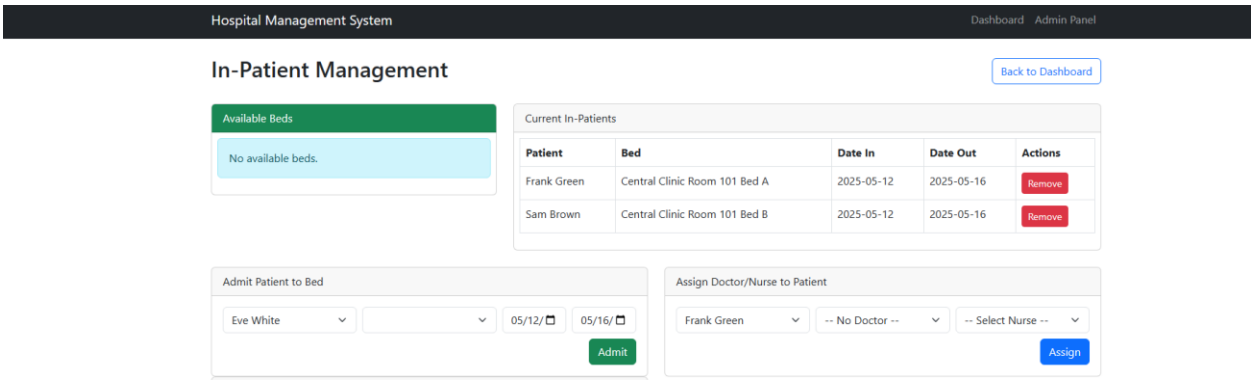
Refreshing Data:

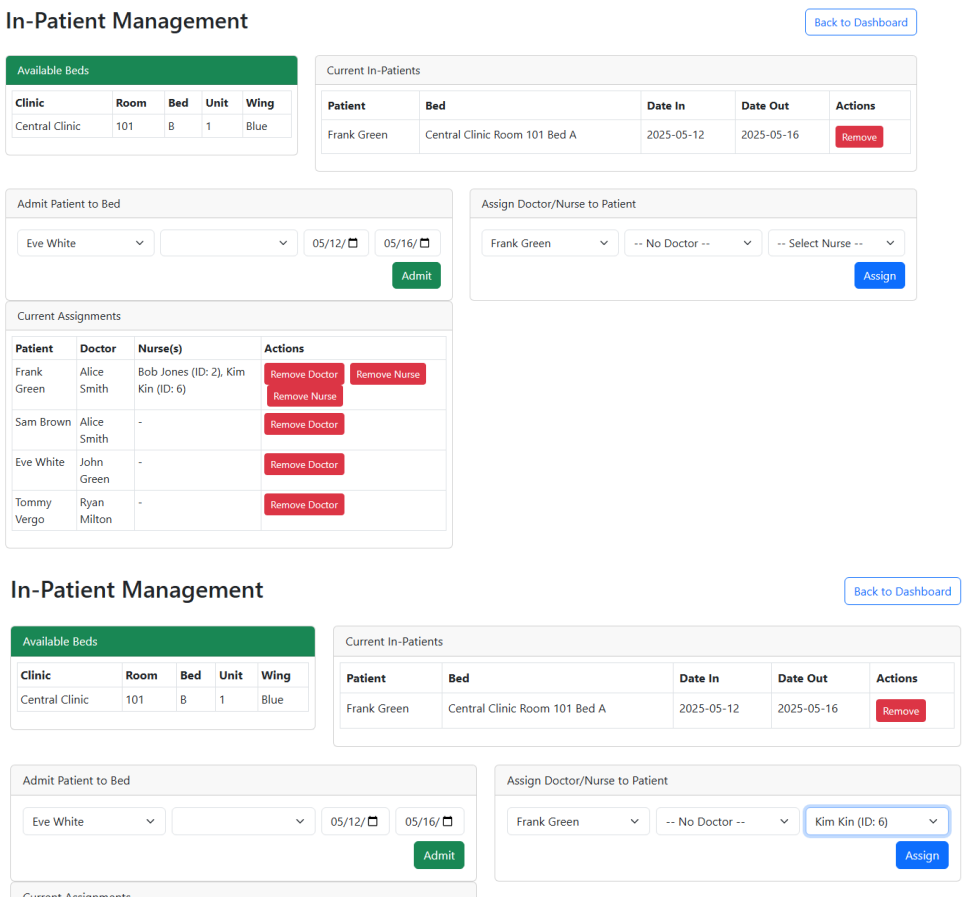
The page automatically refreshes after any changes, ensuring that all information about beds, patients, and assignments is current.

The In-Patient Management page streamlines the process of admitting patients, assigning staff, and tracking bed usage, helping hospital staff maintain an organized and efficient in-patient care environment.


Hospital Management System
Dashboard
Admin Panel
In-Patient Management
Back to Dashboard
Available Beds
Clinic
Room
Bed
Unit
Wing
Central Clinic
101
B
1
Blue
Current In-Patients
Patient
Bed
Date In
Date Out
Actions
Frank Green
Central Clinic Room 101 Bed A
2025-05-12
2025-05-16
Remove
Admit Patient to Bed
Sam Brown
Central Clinic Room
mm/dd
Admit
Assign Doctor/Nurse to Patient
Frank Green
-- No Doctor --
-- Select Nurse --
Assign
Current Assignments
Patient
Doctor
Nurse(s)
Actions
Frank Green
Alice Smith
Bob Jones (ID: 2)
Remove Doctor
Remove Nurse
Sam Brown
Alice Smith
-
Remove Doctor
Eve White
John Green
-
Remove Doctor
Tommy Vergo
Ryan Milton
-
Remove Doctor


In-Patient Management
Back to Dashboard
Available Beds
Clinic
Room
Bed
Unit
Wing
Central Clinic
101
B
1
Blue
Current In-Patients
Patient
Bed
Date In
Date Out
Actions
Frank Green
Central Clinic Room 101 Bed A
2025-05-12
2025-05-16
Remove
Admit Patient to Bed
Sam Brown
Central Clinic Room
05/12/
05/16/
Admit
Assign Doctor/Nurse to Patient
Frank Green
-- No Doctor --
-- Select Nurse --
Assign
Current Assignments


Hospital Management System
Dashboard
Admin Panel
In-Patient Management
Back to Dashboard
Available Beds
No available beds.
Current In-Patients
Patient
Bed
Date In
Date Out
Actions
Frank Green
Central Clinic Room 101 Bed A
2025-05-12
2025-05-16
Remove
Sam Brown
Central Clinic Room 101 Bed B
2025-05-12
2025-05-16
Remove
Admit Patient to Bed
Eve White
05/12/
05/16/
Admit
Assign Doctor/Nurse to Patient
Frank Green
-- No Doctor --
-- Select Nurse --
Assign



Section 4: Admin Panel

The Admin Panel is a powerful interface designed to directly manage all underlying data tables in the hospital management system. It provides comprehensive access to every major entity in the database, allowing for full CRUD (Create, Read, Update, Delete) operations on everything from staff and patients to clinics, beds, allergies, prescriptions, and more.

How to Use the Admin Panel

Selecting a Table:

At the top of the Admin Panel, use the "Select Table" dropdown to choose which database table you want to view or manage. Options include core entities like Employees, Patients, Physicians, Nurses, Surgeries, Shifts, Medical Data, Allergies, Clinic Beds, and many more.

Viewing and Editing Records:

Once a table is selected, its records are displayed in a detailed table format. Each row represents a record, and columns correspond to the fields in that table. For each record, you can:

Edit: Click the "Edit" button to open a modal form pre-filled with the record's current data. Make your changes and save to update the record.

Delete: Click the "Delete" button to remove the record from the database. You'll be prompted to confirm before deletion.

Adding New Records:

Below each table, you'll find a form for adding new records. Fill in the required fields and click "Add" to insert a new entry into the database. The form dynamically adjusts to the selected table, ensuring you only see relevant fields.

Type-Specific and Related Data:

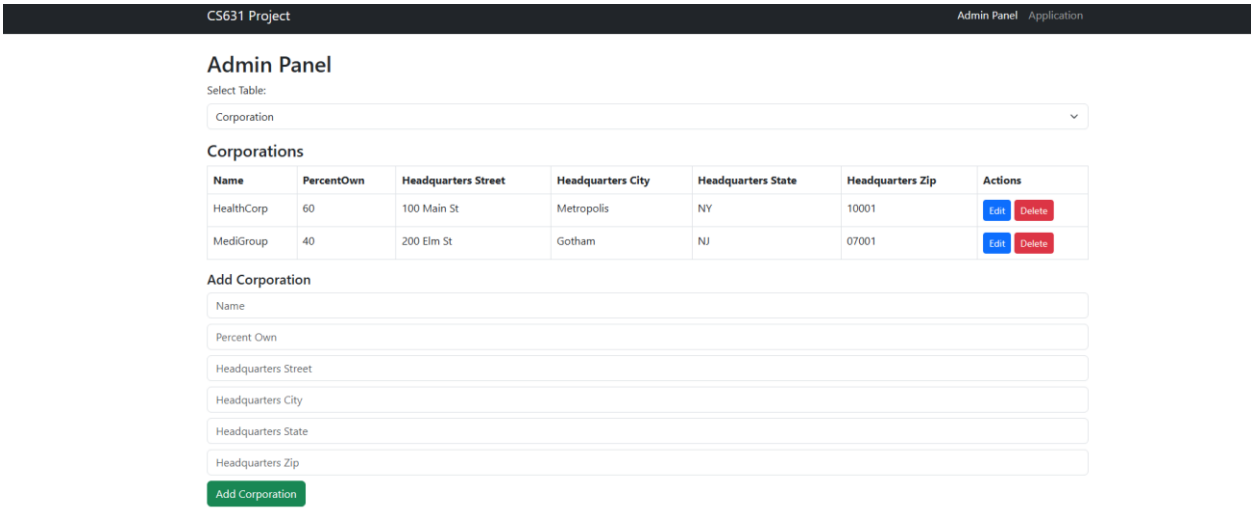
For tables with type-specific fields (such as Nurses, Physicians, or Surgeons), the forms will display additional fields as needed (e.g., grade and years for nurses, specialty and contract details for surgeons). The Admin Panel also allows you to manage many-to-many relationships and join tables, such as nurse skills, surgery types, and patient allergies.

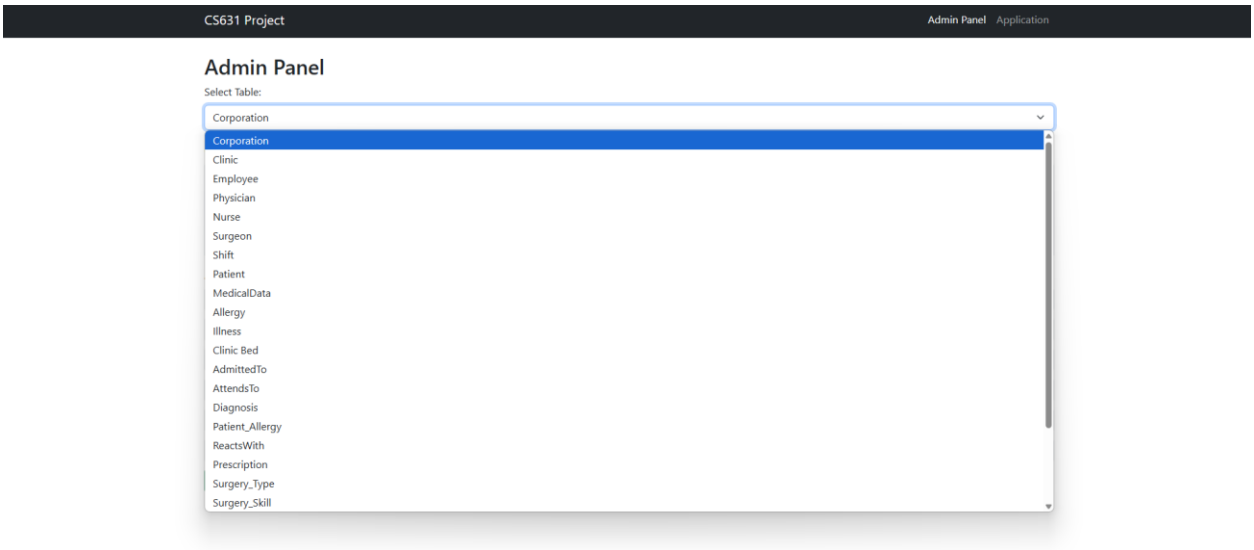
Bulk and Relational Management:

The Admin Panel is especially useful for managing related data and bulk operations. For example, you can quickly assign skills to nurses, link allergies to patients, or manage which nurses are eligible for specific surgery types.

Refreshing and Navigation:

After any operation, the table view refreshes automatically to reflect the latest data. You can switch between tables at any time using the dropdown.





Section 5: SQL Code

-- ==

-- EMPLOYEE-RELATED TABLES

-- ==

CREATE TABLE Employee (

EmpID INT PRIMARY KEY,

First_Name VARCHAR(50) NOT NULL,

```
    Last_Name VARCHAR(50) NOT NULL,
    SSN CHAR(11) UNIQUE NOT NULL,
    Salary DECIMAL(10,2) CHECK (Salary BETWEEN 25000 AND 300000 OR Salary IS NULL),
    Gender CHAR(1) CHECK (Gender IN ('M', 'F')),
    EmployeeType VARCHAR(20) NOT NULL CHECK (EmployeeType IN ('Physician', 'Nurse',
'Surgeon', 'Support')),
    Phone VARCHAR(15) NOT NULL,
    Street VARCHAR(100) NOT NULL,
    City VARCHAR(50) NOT NULL,
    State CHAR(2) NOT NULL,
    Zip VARCHAR(10) NOT NULL
);

CREATE TABLE Physician (
    PhysicianID INT PRIMARY KEY,
    Specialty VARCHAR(50) NOT NULL,
    Type VARCHAR(50) NOT NULL,
    FOREIGN KEY (PhysicianID) REFERENCES Employee(EmpID) ON DELETE CASCADE
);

CREATE TABLE Nurse (
    NurseID INT PRIMARY KEY,
    Grade VARCHAR(20) NOT NULL,
    Years INT NOT NULL,
    FOREIGN KEY (NurseID) REFERENCES Employee(EmpID) ON DELETE CASCADE
);
```

```
CREATE TABLE Surgeon (
   SurgeonID INT PRIMARY KEY,
   Specialty VARCHAR(50) NOT NULL,
   ContractType VARCHAR(50) NOT NULL,
   ContractDuration INT NOT NULL,
   ContractAmount DECIMAL(10,2),
   FOREIGN KEY (SurgeonID) REFERENCES Employee(EmpID) ON DELETE CASCADE
);

-- ============================================
-- CLINIC-RELATED TABLES
-- ============================================

CREATE TABLE Clinic (
   Name VARCHAR(100) PRIMARY KEY,
   Street VARCHAR(100) NOT NULL,
   City VARCHAR(50) NOT NULL,
   State CHAR(2) NOT NULL,
   Zip VARCHAR(10) NOT NULL,
   OwnerID INT,
   FOREIGN KEY (OwnerID) REFERENCES Physician(PhysicianID) ON DELETE SET NULL
);

CREATE TABLE Corporation (
   Name VARCHAR(100) PRIMARY KEY,
```

```
    PercentOwn DECIMAL(5,2) NOT NULL CHECK (PercentOwn BETWEEN 0 AND 100),
    HeadquartersStreet VARCHAR(100) NOT NULL,
    HeadquartersCity VARCHAR(50) NOT NULL,
    HeadquartersState CHAR(2) NOT NULL,
    HeadquartersZip VARCHAR(10) NOT NULL
);

CREATE TABLE Clinic_Bed (
    BedID INT PRIMARY KEY,
    Clinic VARCHAR(100) NOT NULL,
    RoomNum INT NOT NULL,
    BedLetter CHAR(1) NOT NULL CHECK (BedLetter IN ('A', 'B')),
    Unit INT NOT NULL CHECK (Unit BETWEEN 1 AND 7),
    Wing VARCHAR(5) NOT NULL CHECK (Wing IN ('Blue', 'Green')),
    FOREIGN KEY (Clinic) REFERENCES Clinic(Name) ON DELETE CASCADE,
    CONSTRAINT UQ_Bed UNIQUE (Clinic, RoomNum, BedLetter)
);

-- ============================================
-- PATIENT-RELATED TABLES
-- ============================================

CREATE TABLE Patient (
    PID INT PRIMARY KEY,
    PrimaryPhysician INT NOT NULL,
    D_O_B DATE NOT NULL,
```

```
  SSN CHAR(11) UNIQUE NOT NULL,
  PFirstName VARCHAR(50) NOT NULL,
  PLastName VARCHAR(50) NOT NULL,
  Gender CHAR(1) CHECK (Gender IN ('M', 'F')),
  BloodType VARCHAR(3) NOT NULL CHECK (BloodType IN ('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-')),
  PStreet VARCHAR(100) NOT NULL,
  PCity VARCHAR(50) NOT NULL,
  PState CHAR(2) NOT NULL,
  PZip VARCHAR(10) NOT NULL,
  FOREIGN KEY (PrimaryPhysician) REFERENCES Physician(PhysicianID) ON DELETE SET NULL
);

CREATE TABLE Medical_Data (
  PID INT PRIMARY KEY,
  BloodSugar DECIMAL(5,1),
  Cholesterol DECIMAL(5,1),
  Triglycerides DECIMAL(5,1),
  HDL DECIMAL(5,1),
  LDL DECIMAL(5,1),
  LastUpdate DATE NOT NULL,
  Heart_Risk CHAR(1) CHECK (Heart_Risk IN ('N', 'L', 'M', 'H')),
  FOREIGN KEY (PID) REFERENCES Patient(PID) ON DELETE CASCADE
);

CREATE TABLE AdmittedTo (
```

```
    PID INT,
    BedID INT,
    Date_IN DATE NOT NULL,
    Date_OUT DATE,
    PRIMARY KEY (PID, BedID),
    FOREIGN KEY (PID) REFERENCES Patient(PID) ON DELETE CASCADE,
    FOREIGN KEY (BedID) REFERENCES Clinic_Bed(BedID) ON DELETE CASCADE,
    CHECK (Date_OUT IS NULL OR Date_OUT >= Date_IN)
);

CREATE TABLE AttendsTo (
    NurseID INT,
    PID INT,
    StartDate DATE NOT NULL,
    EndDate DATE,
    PRIMARY KEY (NurseID, PID),
    FOREIGN KEY (NurseID) REFERENCES Nurse(NurseID) ON DELETE CASCADE,
    FOREIGN KEY (PID) REFERENCES Patient(PID) ON DELETE CASCADE,
    CHECK (EndDate IS NULL OR EndDate >= StartDate)
);

-- ============================================
-- ILLNESS AND ALLERGY TABLES
-- ============================================

CREATE TABLE Illness (
```

```
    IllnessCode INT PRIMARY KEY,
    Description VARCHAR(255) NOT NULL
);

CREATE TABLE Allergy (
    AllergyCode INT PRIMARY KEY,
    Description VARCHAR(255) NOT NULL
);

CREATE TABLE Diagnosis (
    PID INT,
    PhysicianID INT,
    IllnessCode INT,
    Date DATE NOT NULL,
    Notes TEXT,
    Comment TEXT,
    PRIMARY KEY (PID, PhysicianID, IllnessCode),
    FOREIGN KEY (PID) REFERENCES Patient(PID) ON DELETE CASCADE,
    FOREIGN KEY (PhysicianID) REFERENCES Physician(PhysicianID) ON DELETE CASCADE,
    FOREIGN KEY (IllnessCode) REFERENCES Illness(IllnessCode) ON DELETE CASCADE
);

CREATE TABLE Patient_Allergy (
    PID INT,
    AllergyCode INT,
    PRIMARY KEY (PID, AllergyCode),
```

```
  FOREIGN KEY (PID) REFERENCES Patient(PID) ON DELETE CASCADE,
  FOREIGN KEY (AllergyCode) REFERENCES Allergy(AllergyCode) ON DELETE CASCADE
);

-- ============================================
-- CONSULTATION AND PRESCRIPTION TABLES
-- ============================================

CREATE TABLE Consultation (
  PID INT,
  PhysicianID INT,
  Date DATE,
  Type VARCHAR(50) NOT NULL,
        Notes VARCHAR(250),
  PRIMARY KEY (PID, PhysicianID, Date),
  FOREIGN KEY (PID) REFERENCES Patient(PID) ON DELETE CASCADE,
  FOREIGN KEY (PhysicianID) REFERENCES Physician(PhysicianID) ON DELETE CASCADE
);

CREATE TABLE Medicine (
  Code INT PRIMARY KEY,
  MadeBy VARCHAR(100),
  Name VARCHAR(100) NOT NULL,
  Usage VARCHAR(255),
  UnitCost DECIMAL(10,2) NOT NULL,
  Quantity INT NOT NULL,
```

```
    QtyOrdered INT NOT NULL,
    FOREIGN KEY (MadeBy) REFERENCES Corporation(Name) ON DELETE SET NULL
);

CREATE TABLE ReactsWith (
    Medicine1 INT,
    Medicine2 INT,
    Severity CHAR(1) NOT NULL CHECK (Severity IN ('S', 'M', 'L', 'N')),
    PRIMARY KEY (Medicine1, Medicine2),
    FOREIGN KEY (Medicine1) REFERENCES Medicine(Code) ON DELETE CASCADE,
    FOREIGN KEY (Medicine2) REFERENCES Medicine(Code) ON DELETE CASCADE,
    CHECK (Medicine1 < Medicine2) -- Prevent duplicate entries
);

CREATE TABLE Prescription (
    PrescriptionID INT PRIMARY KEY,
    PID INT NOT NULL,
    MedicineCode INT NOT NULL,
    Prescriber INT NOT NULL,
    DatePrescrbed DATE NOT NULL,
    DateFilled DATE,
    Dosage VARCHAR(50) NOT NULL,
    Duration VARCHAR(50) NOT NULL,
    Frequency VARCHAR(50) NOT NULL,
        Quantity INT NOT NULL DEFAULT 0,
    FOREIGN KEY (PID) REFERENCES Patient(PID) ON DELETE CASCADE,
```

```
  FOREIGN KEY (MedicineCode) REFERENCES Medicine(Code) ON DELETE CASCADE,
  FOREIGN KEY (Prescriber) REFERENCES Physician(PhysicianID) ON DELETE CASCADE,
  UNIQUE (PID, MedicineCode), -- No two physicians can prescribe same medicine to same patient
        CONSTRAINT check_quantity_positive CHECK (Quantity >= 0)
);

-- ============================================
-- SURGERY-RELATED TABLES
-- ============================================

CREATE TABLE Surgery_Type (
  SurgeryCode INT PRIMARY KEY,
  Name VARCHAR(100) NOT NULL,
  Type CHAR(1) NOT NULL CHECK (Type IN ('H', 'O')), -- H for hospitalization, O for outpatient
  AnatomicLocation VARCHAR(100) NOT NULL,
  SpecialNeeds VARCHAR(255)
);

CREATE TABLE Surgery_Skill (
  SkillCode INT PRIMARY KEY,
  Description VARCHAR(255) NOT NULL
);

CREATE TABLE Surgery_Type_Skill (
  SurgeryCode INT,
```

```
  SkillCode INT,
  PRIMARY KEY (SurgeryCode, SkillCode),
  FOREIGN KEY (SurgeryCode) REFERENCES Surgery_Type(SurgeryCode) ON DELETE CASCADE,
  FOREIGN KEY (SkillCode) REFERENCES Surgery_Skill(SkillCode) ON DELETE CASCADE
);

CREATE TABLE Nurse_Skill (
  NurseID INT,
  SkillCode INT,
  PRIMARY KEY (NurseID, SkillCode),
  FOREIGN KEY (NurseID) REFERENCES Nurse(NurseID) ON DELETE CASCADE,
  FOREIGN KEY (SkillCode) REFERENCES Surgery_Skill(SkillCode) ON DELETE CASCADE
);

CREATE TABLE Nurse_Surgery_Type (
  NurseID INT,
  SurgeryCode INT,
  PRIMARY KEY (NurseID, SurgeryCode),
  FOREIGN KEY (NurseID) REFERENCES Nurse(NurseID) ON DELETE CASCADE,
  FOREIGN KEY (SurgeryCode) REFERENCES Surgery_Type(SurgeryCode) ON DELETE CASCADE
);

CREATE TABLE Surgery_Schedule (
  NurseID INT,
  SurgeryShift VARCHAR(20),
```

```
    Date DATE,
    PRIMARY KEY (NurseID, SurgeryShift, Date),
    FOREIGN KEY (NurseID) REFERENCES Nurse(NurseID) ON DELETE CASCADE
);

CREATE TABLE Surgery (
    PID INT,
    SurgeonID INT,
    SurgeryCode INT,
    Date DATE,
    Theater VARCHAR(50) NOT NULL,
    SurgeryShift VARCHAR(20),
    PRIMARY KEY (PID, SurgeonID, SurgeryCode, Date),
    FOREIGN KEY (PID) REFERENCES Patient(PID) ON DELETE CASCADE,
    FOREIGN KEY (SurgeonID) REFERENCES Surgeon(SurgeonID) ON DELETE RESTRICT, --
Cannot delete surgeon with surgery records
    FOREIGN KEY (SurgeryCode) REFERENCES Surgery_Type(SurgeryCode) ON DELETE
CASCADE
);

-- ============================================
-- SHIFT MANAGEMENT
-- ============================================

CREATE TABLE Shift (
    EmpID INT,
    Day VARCHAR(10),
```

```
    BeginTime TIME NOT NULL,
    EndTime TIME NOT NULL,
    PRIMARY KEY (EmpID, Day),
    FOREIGN KEY (EmpID) REFERENCES Employee(EmpID) ON DELETE CASCADE,
    CHECK (EndTime > BeginTime)
);

-- ============================================
-- TRIGGERS
-- ============================================

-- Trigger to ensure a nurse is not assigned to more than one surgery type
CREATE OR REPLACE FUNCTION check_nurse_surgery_type_limit()
RETURNS TRIGGER AS $$
BEGIN
    IF (SELECT COUNT(*) FROM Nurse_Surgery_Type WHERE NurseID = NEW.NurseID) > 0
THEN
        RAISE EXCEPTION 'A nurse cannot be assigned to more than one surgery type';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_nurse_surgery_type_limit_trigger
BEFORE INSERT ON Nurse_Surgery_Type
FOR EACH ROW
```

```
EXECUTE FUNCTION check_nurse_surgery_type_limit();


-- Trigger to calculate heart risk category when medical data is updated
CREATE OR REPLACE FUNCTION calculate_heart_risk()
RETURNS TRIGGER AS $$
DECLARE
  total_cholesterol DECIMAL(5,1);
  ratio DECIMAL(5,2);
  risk_category CHAR(1);
BEGIN
  -- Calculate total cholesterol
  total_cholesterol := NEW.HDL + NEW.LDL + (NEW.Triglycerides / 5);


  -- Calculate ratio
  IF NEW.HDL > 0 THEN
    ratio := total_cholesterol / NEW.HDL;
  ELSE
    ratio := 0;
  END IF;


  -- Determine risk category
  IF ratio < 4 THEN
    risk_category := 'N'; -- None
  ELSIF ratio >= 4 AND ratio < 5 THEN
    risk_category := 'L'; -- Low
  ELSIF ratio >= 5 THEN
```

```
    risk_category := 'M'; -- Moderate
  ELSE
    risk_category := 'N'; -- Default
  END IF;

  -- Update the risk category
  NEW.Heart_Risk := risk_category;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER calculate_heart_risk_trigger
BEFORE INSERT OR UPDATE ON Medical_Data
FOR EACH ROW
EXECUTE FUNCTION calculate_heart_risk();

-- Trigger to handle physician deletion
CREATE OR REPLACE FUNCTION handle_physician_deletion()
RETURNS TRIGGER AS $$
DECLARE
  chief_of_staff INT;
BEGIN
  -- Find the chief of staff
  SELECT PhysicianID INTO chief_of_staff
  FROM Physician
```

```
    WHERE Type = 'Chief of Staff'
    LIMIT 1;

    -- Reassign patients to chief of staff
    UPDATE Patient
    SET PrimaryPhysician = chief_of_staff
    WHERE PrimaryPhysician = OLD.PhysicianID;

    -- Remove prescriptions by this physician
    DELETE FROM Prescription
    WHERE Prescriber = OLD.PhysicianID;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER handle_physician_deletion_trigger
BEFORE DELETE ON Physician
FOR EACH ROW
EXECUTE FUNCTION handle_physician_deletion();

-- Trigger to handle nurse deletion
CREATE OR REPLACE FUNCTION handle_nurse_deletion()
RETURNS TRIGGER AS $$
BEGIN
    -- Temporarily remove associations with in-patients
```

```
    UPDATE AttendsTo
    SET End_Date = CURRENT_DATE
    WHERE NurseID = OLD.NurseID AND End_Date IS NULL;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER handle_nurse_deletion_trigger
BEFORE DELETE ON Nurse
FOR EACH ROW
EXECUTE FUNCTION handle_nurse_deletion();
```