



# **CS353 Project**

## **Hospital Appointment Management System - MediSync**

### **Design Report**

**Group 3: Veribaz**

Asya Ünal - 22204017

Barış Yayıcı - 22202981

Aybars Buğra Aksoy - 22202680

Eren Berk Eraslan - 22201772

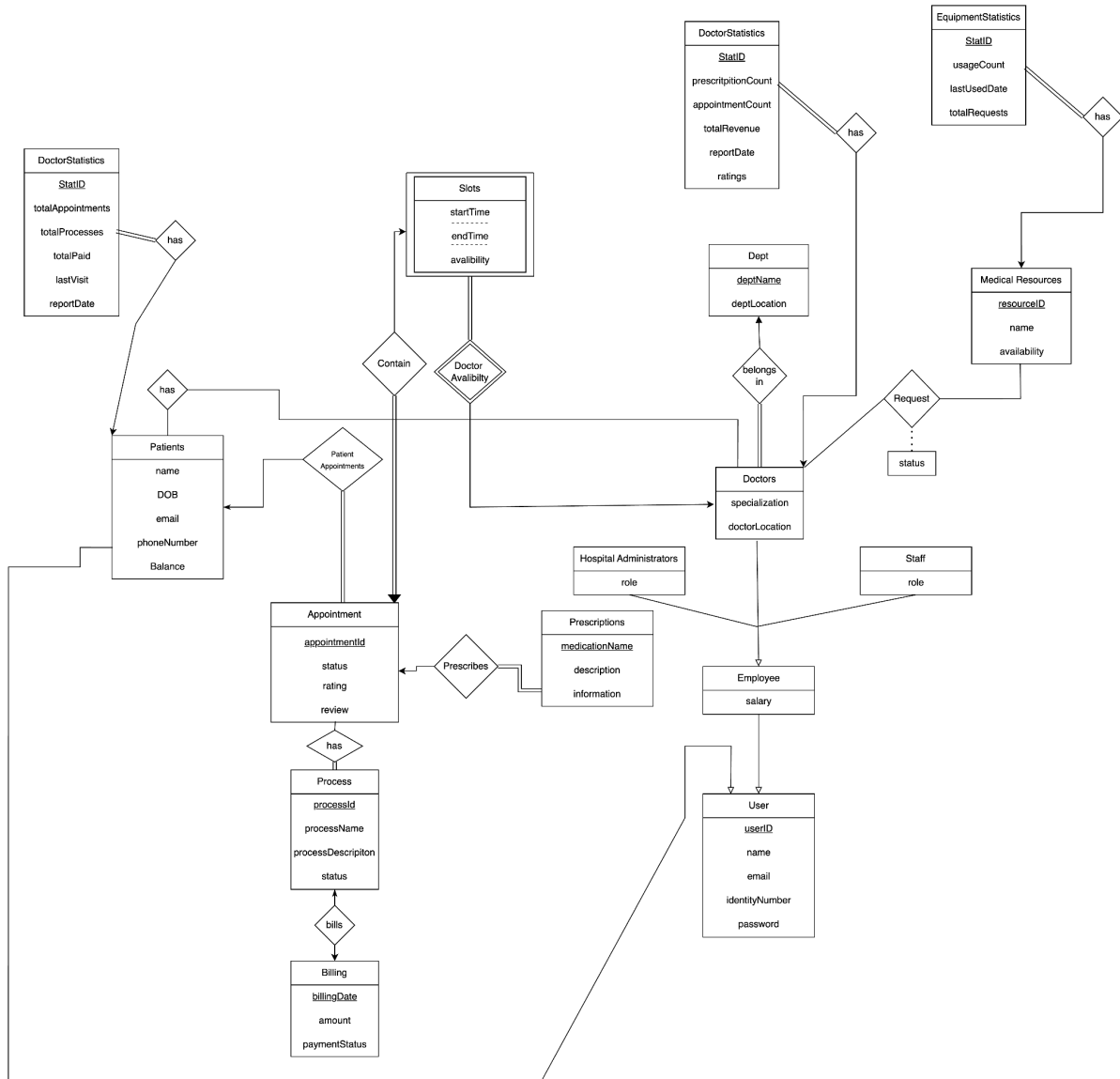
Bora Yetkin - 22203661

**Instructor:** Özgür Ulusoy

**TA:** Hasan Alp Caferoğlu

# 1. Design of the Database

## 1.1 Revised E/R Diagram



## 1.2 Relations

**DoctorStatistics**(**StatID**, prescriptionCount, appointmentCount, totalRevenue, reportDate, ratings, doctorID)

**Domains:**

- StatID: Integer (unique identifier)
- prescriptionCount: Integer (non-negative count)
- appointmentCount: Integer (non-negative count)
- totalRevenue: Decimal/Money (non-negative amount)
- reportDate: Date (valid calendar date)
- rating: Decimal/Float (typically 0-5 scale)
- doctorID: Integer (matches existing doctor ID)

**Candidate Keys:**

- StatID
- (doctorID, reportDate) - assuming one statistics record per doctor per date

**Primary Key:** **StatID** (as indicated in bold in the schema)

**Foreign Keys:** doctorID references Doctors(employeeID)

**EquipmentStatistics**(**StatID**, usageCount, lastUsedDate, totalRequests, equipmentID)

**Domains:**

- StatID: Integer (unique identifier)
- usageCount: Integer (non-negative count)
- lastUsedDate: Date (valid calendar date)
- totalRequests: Integer (non-negative count)
- equipmentID: Integer (matches existing equipment ID)

**Candidate Keys:**

- StatID
- (equipmentID, lastUsedDate) - assuming one statistics record per equipment per date

**Primary Key:** **StatID** (as indicated in bold in the schema)

**Foreign Keys:** equipmentID references MedicalResources(resourceID)

**MedicalResources(resourceID, name, availability)**

**Domains:**

- resourceID: Integer (unique identifier)
- name: String (equipment name)
- availability: Boolean/String (e.g., "Available", "In Use", "Maintenance")

**Candidate Keys:**

- resourceID
- name - assuming equipment names are unique in the system

**Primary Key:** resourceID (as indicated in bold in the schema)

**Foreign Keys:** None

**Dept(deptName, deptLocation)**

**Domains:**

- deptName: String (department name)
- deptLocation: String (physical location)

**Candidate Keys:**

- deptName
- deptLocation - assuming each location houses only one department

**Primary Key:** deptName (as indicated in bold in the schema)

**Foreign Keys:** None

**Doctors(employeeID, specialization, doctorLocation, deptName)**

**Domains:**

- employeeID: Integer (unique identifier)
- specialization: String (medical specialty)
- doctorLocation: String (office/clinic location)
- deptName: String (department name)

**Candidate Keys:**

- employeeID

**Primary Key: employeeID** (as indicated in bold in the schema)

**Foreign Keys:**

- deptName references Dept(deptName)
- employeeID references Employee(employeeID)

**Staff(employeeID, role)**

**Domains:**

- employeeID: Integer (unique identifier)
- role: String (job title/role)

**Candidate Keys:**

- employeeID

**Primary Key: employeeID** (as indicated in bold in the schema)

**Foreign Keys:** employeeID references Employee(employeeID)

**DoctorPatient(doctorID,patientID)**

**Domains:**

- doctorID: Integer (doctor identifier)
- patientID: Integer (patient identifier)

**Candidate Keys:**

- (doctorID, patientID)

**Primary Key: (doctorID, patientID)** (as indicated in bold in the schema)

**Foreign Keys:**

- doctorID references Doctors(employeeID)
- patientID references Patient(patientID)

### **HospitalAdministrators(employeeID, role)**

#### **Domains:**

- employeeID: Integer (unique identifier)
- role: String (administrative role)

#### **Candidate Keys:**

- employeeID

**Primary Key:** **employeeID** (as indicated in bold in the schema)

**Foreign Keys:** employeeID refer

ences Employee(employeeID)

### **Patients(patientID, name, DOB, email, phoneNumber, Balance)**

#### **Domains:**

- patientID: Integer (unique identifier)
- name: String (patient name)
- DOB: Date (date of birth)
- email: String (valid email format)
- phoneNumber: String (valid phone number format)
- Balance: Decimal/Money (account balance)

#### **Candidate Keys:**

- patientID
- email - assuming email addresses are unique
- phoneNumber - assuming phone numbers are unique
- (name, DOB) - assuming name and birth date combinations are unique

**Primary Key:** **patientID** (as indicated in bold in the schema)

**Foreign Keys:** patientID references User(userID)

### **PatientStatistics(StatID, patientID, totalAppointments, totalProcesses, totalPaid, lastVisit, reportDate)**

#### **Domains:**

- StatID: Integer (unique identifier)
- patientID: Integer (patient identifier)
- totalAppointments: Integer (non-negative count)

- totalProcesses: Integer (non-negative count)
- totalPaid: Decimal/Money (non-negative amount)
- lastVisit: Date (valid calendar date)
- reportDate: Date (valid calendar date)

**Candidate Keys:**

- StatID
- (patientID, reportDate) - assuming one statistics record per patient per date

**Primary Key:** StatID (as indicated in bold in the schema)

**Foreign Keys:** patientID references Patients(patientID)

**Slots(doctorID, slotID, startTime, endTime, availability)**

**Domains:**

- doctorID: Integer (doctor identifier)
- slotID: Integer (slot identifier)
- startTime: DateTime (appointment start time)
- endTime: DateTime (appointment end time)
- availability: Boolean/String (e.g., "Available", "Booked")

**Candidate Keys:**

- (doctorID, slotID)
- (doctorID, startTime, endTime) - assuming no overlapping slots for same doctor

**Primary Key:** (doctorID, slotID, startTime, endTime) (as indicated in bold in the schema)

**Foreign Keys:** doctorID references Doctors(employeeID)

**Contain(appointmentID, doctorID, slotID, startTime, endTime)**

**Domains:**

- appointmentID: Integer (appointment identifier)
- doctorID: Integer (doctor identifier)
- slotID: Integer (slot identifier)
- startTime: DateTime (start time)
- endTime: DateTime (end time)

**Candidate Keys:**

- (appointmentID, doctorID, slotID, startTime, endTime)
- appointmentID - assuming each appointment is assigned to exactly one slot

**Primary Key:** Composite key of (**appointmentID, doctorID, slotID, startTime, endTime**)

**Foreign Keys:**

- appointmentID references Appointment(appointmentID)
- (doctorID, slotID, startTime, endTime) references Slots(doctorID, slotID, startTime, endTime)

**Appointment**(**appointmentID**, status, rating, review, *patientID*, *doctorID*)

**Domains:**

- appointmentID: Integer (unique identifier)
- status: String (e.g., "Scheduled", "Completed", "Cancelled")
- rating: Integer/Float (typically 1-5 scale)
- review: Text (patient feedback)
- patientID: Integer (patient identifier)
- doctorID: Integer (doctor identifier)

**Candidate Keys:**

- appointmentID
- (patientID, doctorID, startTime, endTime) - assuming a patient can't have multiple appointments with the same doctor at the same time

**Primary Key:** **appointmentID** (as indicated in bold in the schema)

**Foreign Keys:**

- patientID references Patients(patientID)
- doctorID references Doctors(employeeID)

**Process**(**processID**, processName, processDescription, status, *appointmentID*)

**Domains:**

- processID: Integer (unique identifier)
- processName: String (name of medical procedure/service)



- processDescription: Text (detailed description)
- status: String (e.g., "Scheduled", "In Progress", "Completed")
- appointmentID: Integer (appointment identifier)

**Candidate Keys:**

- processID
- (appointmentID, processName) - assuming each procedure type is performed only once per appointment

**Primary Key:** **processID** (as indicated in bold in the schema)

**Foreign Keys:** appointmentID references Appointment(appointmentID)

**Billing(billingID, billingDate, amount, paymentStatus, *processID*)**

**Domains:**

- billingID: Integer (unique identifier)
- billingDate: Date (invoice date)
- amount: Decimal/Money (charged amount)
- paymentStatus: String (e.g., "Paid", "Pending", "Overdue")
- processID: Integer (process identifier)

**Candidate Keys:**

- billingID
- (processID, billingDate) - assuming one billing record per process per date

**Primary Key:** **billingID** (as indicated in bold in the schema)

**Foreign Keys:** processID references Process(processID)

**Prescriptions(prescriptionID, medicationName, description, information, *doctorID*, *appointmentID*)**

**Domains:**

- prescriptionID: Integer (unique identifier)
- medicationName: String (name of medication)
- description: Text (usage instructions)
- information: Text (additional information)
- doctorID: Integer (doctor identifier)
- appointmentID: Integer (appointment identifier)

**Candidate Keys:**

- prescriptionID
- (doctorID, appointmentID, medicationName) - assuming each medication is prescribed only once per doctor-appointment pair

**Primary Key:** **prescriptionID** (as indicated in bold in the schema)

**Foreign Keys:**

- doctorID references Doctors(employeeID)
- appointmentID references Appointment(appointmentID)

**Employee**(**employeeID**, salary)

**Domains:**

- employeeID: Integer (unique identifier)
- salary: Decimal/Money (employee salary)

**Candidate Keys:**

- employeeID

**Primary Key:** **employeeID** (as indicated in bold in the schema)

**Foreign Keys:** employeeID references User(userID)

**User**(**userID**, name, email, identityNumber, password)

**Domains:**

- userID: Integer (unique identifier)
- name: String (user name)
- email: String (valid email format)
- identityNumber: String (government-issued ID)
- password: String (hashed password)

**Candidate Keys:**

- userID
- email - assuming email addresses are unique
- identityNumber - assuming government IDs are unique

**Primary Key:** **userID** (as indicated in bold in the schema)

**Foreign Keys:** None

**Request**(*doctorID*, *resourceID*, status)

**Domains:**

- doctorID: Integer (doctor identifier)
- resourceID: Integer (resource identifier)
- status: String (e.g., "Pending", "Approved", "Denied")

**Candidate Keys:**

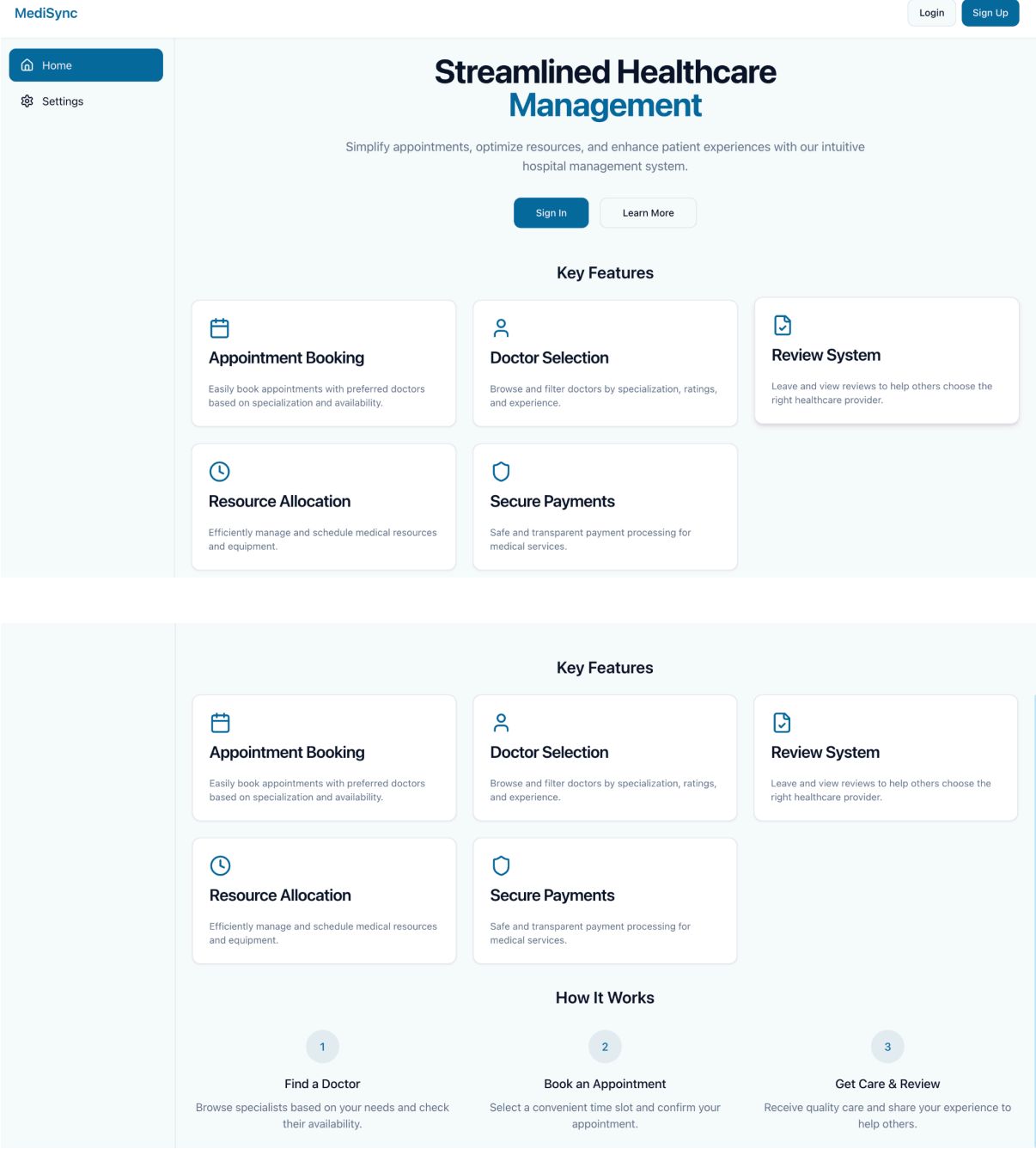
- (doctorID, resourceID)

**Primary Key:** (**doctorID**, **resourceID**) (composite key as indicated)

**Foreign Keys:**

- doctorID references Doctors(employeeID)
- resourceID references MedicalResources(resourceID)

# 2. User Interface Design & Corresponding SQL Statements



The information here is static.

# Sign In

**MediSync**

LoginSign Up

Home

Settings

Sign In

Enter your credentials to access your account

I am a:

☒ Patient☐ Doctor☐ Hospital Staff☐ Admin

Email

Enter your email

Password

Enter your password

Forgot password?

Demo credentials are pre-filled. Use:

- patient@example.com (Patient)
- doctor@example.com (Doctor)
- staff@example.com (Staff)
- admin@example.com (Admin)

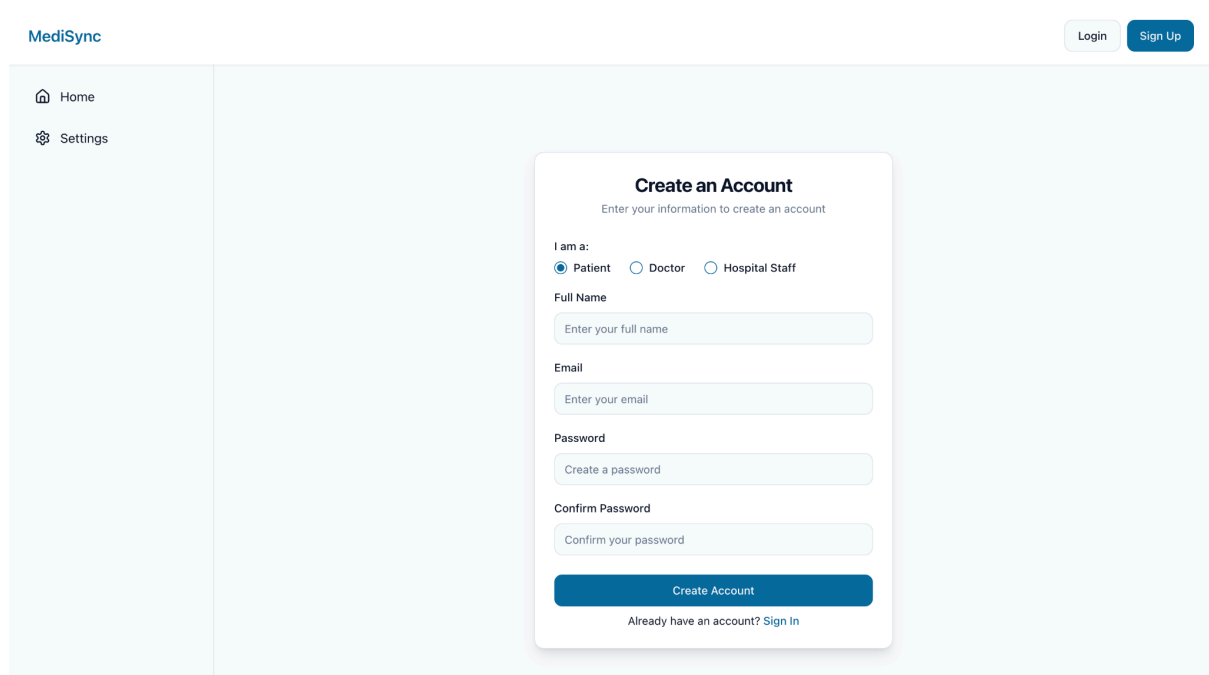
All use "password" as the password.

Sign In

Don't have an account? Sign Up

```
SELECT u.userID, u.name, u.email,
CASE
  WHEN p.patientID IS NOT NULL THEN 'Patient'
  WHEN d.employeeID IS NOT NULL THEN 'Doctor'
  WHEN s.employeeID IS NOT NULL THEN 'Staff'
  WHEN ha.employeeID IS NOT NULL THEN 'Admin'
END AS role
FROM User u
LEFT JOIN Patients p ON u.userID = p.patientID
LEFT JOIN Doctors d ON u.userID = d.employeeID
LEFT JOIN Staff s ON u.userID = s.employeeID
LEFT JOIN HospitalAdministrators ha ON u.userID = ha.employeeID
WHERE u.email = ? AND u.password = ?;
```

# Sign Up



The image shows a web application interface for 'MediSync'. In the top right corner, there are 'Login' and 'Sign Up' buttons. On the left side, there is a sidebar with 'Home' and 'Settings' links. The main content area features a 'Create an Account' form. The form has a title 'Create an Account' and a subtitle 'Enter your information to create an account'. It includes three radio buttons for 'I am a:' with 'Patient' selected. Below this are input fields for 'Full Name', 'Email', 'Password', and 'Confirm Password'. A 'Create Account' button is at the bottom of the form, and a link 'Already have an account? Sign In' is below it.

MediSync

Login Sign Up

Home Settings

### Create an Account

Enter your information to create an account

I am a:

☒ Patient ☐ Doctor ☐ Hospital Staff

Full Name

Enter your full name

Email

Enter your email

Password

Create a password

Confirm Password

Confirm your password

Create Account

Already have an account? Sign In

## Query to create a new patient user:

-- First insert into User table

```
INSERT INTO User (name, email, identityNumber, password)
VALUES (?, ?, ?, ?);
```

-- Then insert into Patients table with the generated userID

```
INSERT INTO Patients (patientID, name, DOB, email, phoneNumber, Balance)
VALUES (LAST_INSERT_ID(), ?, NULL, ?, ?, 0);
```

## Query to create a new doctor user:

-- First insert into User table

```
INSERT INTO User (name, email, identityNumber, password)
VALUES (?, ?, ?, ?);
```

-- Then insert into Employee table

```
INSERT INTO Employee (employeeID, salary)
VALUES (LAST_INSERT_ID(), NULL);
```

-- Finally insert into Doctors table

```
INSERT INTO Doctors (employeeID, specialization, doctorLocation, deptName)
VALUES (LAST_INSERT_ID(), NULL, NULL, NULL);
```

**Query to create a new staff user:**

-- First insert into User table

```
INSERT INTO User (name, email, identityNumber, password)
VALUES (?, ?, ?, ?);
```

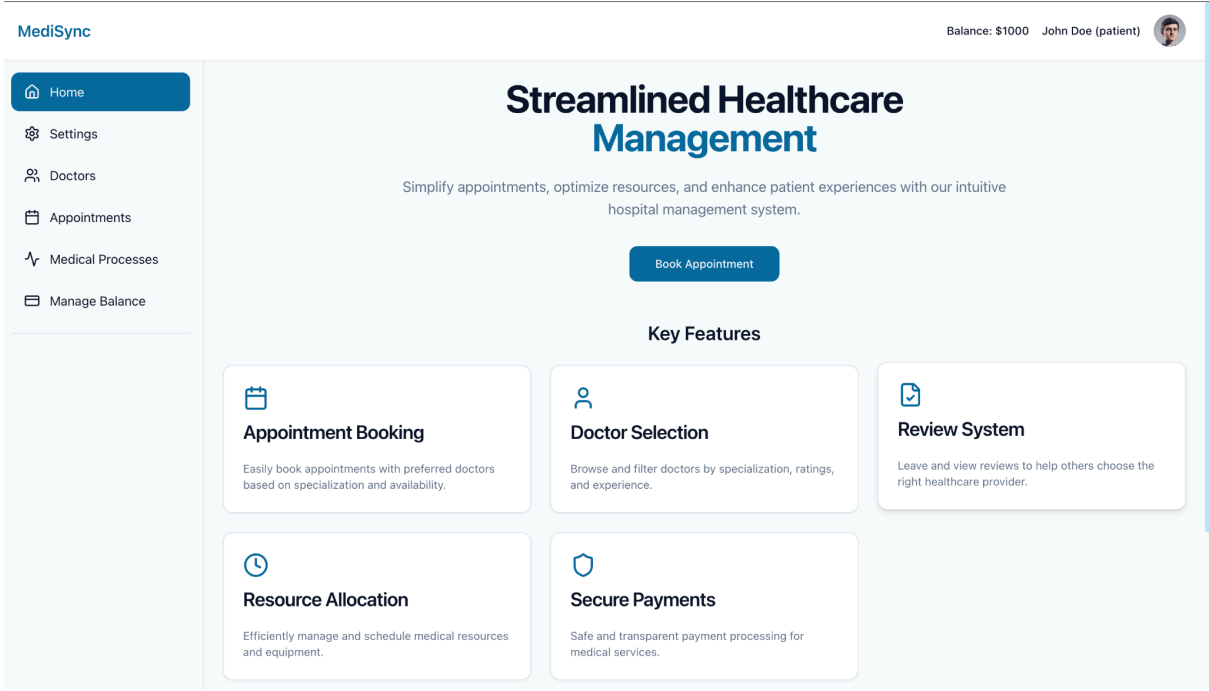
-- Then insert into Employee table

```
INSERT INTO Employee (employeeID, salary)
VALUES (LAST_INSERT_ID(), NULL);
```

-- Finally insert into Staff table

```
INSERT INTO Staff (employeeID, role)
VALUES (LAST_INSERT_ID(), ?);
```

# Patient Home Page




```
SELECT p.name, p.Balance
FROM Patients p
WHERE p.patientID = ?;
```



## Settings - Account Settings

**MediSync**

Balance: \$1000   John Doe (patient) 

Home

Settings

Doctors


Appointments

Medical Processes

Manage Balance

Account Settings

Profile   Security   Balance

 **Profile Information**

Update your personal information

Full Name

John Doe

Email

patient@example.com

Phone Number

123-456-7890

Save Changes

Logout

### Query to get patient profile information:


```
SELECT p.name, p.email, p.phoneNumber, p.DOB
FROM Patients p
WHERE p.patientID = ?;
```

### Query to update patient profile information:

```
-- Update in User table
UPDATE User
SET name = ?
WHERE userID = ?;
```

```
-- Update in Patients table
UPDATE Patients
SET name = ?, email = ?, phoneNumber = ?
WHERE patientID = ?;
```


## Settings - Security Settings

MediSync Balance: \$1000 John Doe (patient) 

[Home](#)  
[Settings](#)  
[Doctors](#)  
[Appointments](#)  
[Medical Processes](#)  
[Manage Balance](#)  
  
[Logout](#)

### Account Settings

Profile **Security** Balance

 **Security**

Update your password

Current Password

Enter your current password

New Password

Enter new password

Confirm New Password

Confirm new password

Change Password

-- First verify the current password

SELECT userID

FROM User

WHERE userID = ? AND password = ?;

-- If the verification succeeds, update the password


UPDATE User

SET password = ?

WHERE userID = ?;

## Settings - Balance Settings

MediSync

Balance: \$1000   John Doe (patient) 

Home

Settings

Doctors

Appointments

Medical Processes

Manage Balance

Account Settings

Profile   Security   **Balance**

\$ **Current Balance**

Your current account balance

**\$1000**

This balance can be used for:

- Paying for appointments
- Covering medical procedure costs
- Prescription medications

Add Balance

Add funds to your account

Amount

\$   Enter amount

Demo Mode

In a real app, this would connect to a payment processor. For this demo, balance is added instantly.

Add Funds

-- Query to retrieve the current balance for a patient

SELECT Balance

FROM Patients

WHERE patientID = ?;

-- Query to add funds to a patient's account


UPDATE Patients

SET Balance = Balance + ?

WHERE patientID = ?;

## Patient - Doctors Page

**MediSync**

Balance: \$1000   John Doe (patient) 

Home

Settings


**Doctors**


Appointments


Medical Processes


Manage Balance


### Our Doctors


Search doctors... 


**Dr. Emma Smith**  
Cardiology  
★ 4.8 Rating  

Book Appointment 


**Dr. Michael Brown**  
Neurology  
★ 4.5 Rating  


Book Appointment 


**Dr. Sarah Lee**  
Pediatrics  
★ 4.9 Rating  


Book Appointment 


### Favorite Doctors


**Dr. Emma Smith**  
Cardiology  
★ 4.8 Rating  

Book Appointment 

**Dr. Michael Brown**  
Neurology  
★ 4.5 Rating  

Book Appointment 

**Dr. Sarah Lee**  
Pediatrics  
★ 4.9 Rating  


Book Appointment 

Logout

```
SELECT D.employeeID,D.specialization, DS.ratings
FROM Doctors D
JOIN DoctorStatistics DS ON D.employeeID = DS.doctorID
WHERE D.specialization = 'Cardiology' AND DS.rating >= 4.5;
You can change or specify the specialization and rating here
```

## Patient - Book Appointment Page

MediSync

Balance: \$1000   John Doe (patient) 

Home

Settings


Doctors

Appointments

Medical Processes

Manage Balance

Book an Appointment



Dr. Emma Smith

Cardiology ★ 4.8

Specialized in cardiovascular health with 8+ years of experience.

\$ \$150 per visit

Available on: Monday Tuesday Wednesday Friday

1. Select a Date

Choose an available date for your appointment

< March 2025 >

Su	Mo	Tu	We	Th	Fr	Sa
23	24	25	26	27	28	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22

2. Choose a Time Slot

Available time slots for Friday, March 28, 2025

09:00 - 09:30

09:30 - 10:00

10:00 - 10:30

10:30 - 11:00

11:00 - 11:30

11:30 - 12:00

14:00 - 14:30

14:30 - 15:00

15:00 - 15:30

15:30 - 16:00

3. Confirm and Pay

Appointment Details

14:00 - 14:30

Doctor

Dr. Emma Smith

Specialization

Cardiology

Date

Friday, March 28, 2025

Fee

\$150

Total\$150

Payment will be processed from your account balance.

Confirm & Pay \$150

Logout

### 1. Get Doctor Information and Rating

-- Get doctor information including average rating

```
SELECT d.employeeID, u.name, d.specialization,
       (SELECT AVG(a.rating) FROM Appointment a WHERE a.doctorID = d.employeeID) AS
avg_rating
FROM Doctors d
JOIN User u ON d.employeeID = u.userID
WHERE d.employeeID = ?;
```

## 2. Get Doctor's Available Days

-- Get days of the week when the doctor has available slots

```
SELECT DISTINCT
  CASE DAYOFWEEK(s.startTime)
    WHEN 1 THEN 'Sunday'
    WHEN 2 THEN 'Monday'
    WHEN 3 THEN 'Tuesday'
    WHEN 4 THEN 'Wednesday'
    WHEN 5 THEN 'Thursday'
    WHEN 6 THEN 'Friday'
    WHEN 7 THEN 'Saturday'
  END AS available_day
FROM Slots s
WHERE s.doctorID = ?
AND s.availability = 'Available'
AND s.startTime > NOW()
ORDER BY DAYOFWEEK(s.startTime);
```

## 3. Get Available Dates for a Specific Month

-- Get available dates for a specific month

```
SELECT DISTINCT DATE(s.startTime) AS available_date
FROM Slots s
WHERE s.doctorID = ?
AND s.availability = 'Available'
AND YEAR(s.startTime) = ?
AND MONTH(s.startTime) = ?
ORDER BY available_date;
```

## 4. Get Available Time Slots for a Specific Date

-- Get available time slots for a specific date

```
SELECT s.slotID, s.startTime, s.endTime
FROM Slots s
WHERE s.doctorID = ?
AND s.availability = 'Available'
AND DATE(s.startTime) = ?
ORDER BY s.startTime;
```

## 5. Check Patient Balance

-- Check if patient has enough balance for the appointment

```
SELECT Balance
FROM Patients
WHERE patientID = ?
AND Balance >= ?;
```

## 6. Book the Appointment

-- 1. Create new appointment

```
INSERT INTO Appointment (status, rating, review, patientID, doctorID)
VALUES ('Scheduled', NULL, NULL, ?, ?);
```

-- 2. Get the newly created appointment ID

```
SET @appointment_id = LAST_INSERT_ID();
```

-- 3. Update slot to be booked

```
UPDATE Slots
SET availability = 'Booked'
WHERE doctorID = ? AND slotID = ?;
```

-- 4. Create the appointment-slot relationship


```
INSERT INTO Contain (appointmentID, doctorID, slotID, startTime, endTime)
VALUES (@appointment_id, ?, ?, ?, ?);
```


-- 5. Deduct appointment fee from patient balance


```
UPDATE Patients
SET Balance = Balance - ?
WHERE patientID = ?;
```


## Patients - Appointments Page


MediSync


Balance: \$1000   John Doe (patient) 


 Home

 Settings

 Doctors

 Appointments


 Medical Processes


 Manage Balance

Logout

### My Appointments

Upcoming (2)   Past (0)

 **Dr. Emma Smith**  
Cardiology  
Monday, July 15, 2024   10:00 - 10:30  
Amount: \$150   Unpaid

 **Dr. Sarah Lee**  
Pediatrics  
Wednesday, July 17, 2024   09:00 - 09:30  
Amount: \$120   Unpaid

Book New Appointment

```
SELECT
    U2.name AS doctor_name,
    D.specialization,
    A.status,
    C.startTime,
    C.endTime
FROM User U
JOIN Patients P ON U.userID = P.patientID
JOIN Appointment A ON P.patientID = A.patientID
JOIN Doctors D ON A.doctorID = D.employeeID
JOIN User U2 ON D.employeeID = U2.userID
JOIN Contain C ON A.appointmentID = C.appointmentID
WHERE U.name = 'John Doe'
ORDER BY C.startTime;
```



# Patients - Medical Processes Page

MediSync

Balance: \$1000   John Doe (patient)

Home

Settings

Doctors

Appointments

Medical Processes

Manage Balance

Logout

My Medical Processes

Doctors

Healthcare providers you've visited

Dr. Emma Smith

Cardiology

Rating: 4.8

Dr. Sarah Lee

Pediatrics

Rating: 4.9

Upcoming (1)

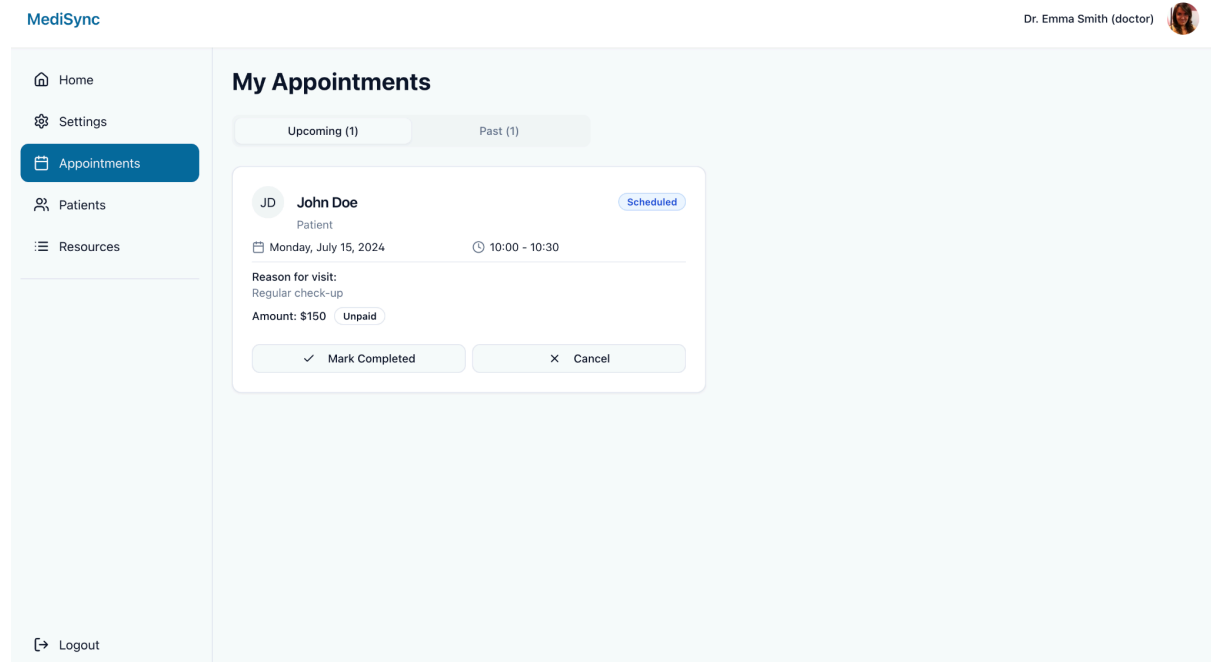
Completed (0)

Cancelled (0)

Process	Doctor	Date	Status	Billing	Action
X-Ray Scan Chest X-Ray for diagnosis	Dr. Emma Smith	Jul 15, 2024	Scheduled	\$ \$100 Pending	

```
SELECT
  P.processName,
  P.processDescription,
  U2.name AS doctor_name,
  DATE(C.startTime) AS process_date,
  P.status,
  B.amount,
  B.paymentStatus
FROM User U
JOIN Patients Pa ON U.userID = Pa.patientID
JOIN Appointment A ON Pa.patientID = A.patientID
JOIN Doctors D ON A.doctorID = D.employeeID
JOIN User U2 ON D.employeeID = U2.userID
JOIN Process P ON A.appointmentID = P.appointmentID
JOIN Billing B ON P.processID = B.processID
JOIN Contain C ON A.appointmentID = C.appointmentID
WHERE U.name = 'John Doe'
ORDER BY C.startTime;
```

## Doctor - Appointments Page



-- Get upcoming appointments for a doctor

```
SELECT
  a.appointmentID,
  p.patientID,
  u.name AS patient_name,
  c.startTime,
  c.endTime,
  a.status,
  'Regular check-up' AS visit_reason,
  '$150' AS amount,
  CASE WHEN b.paymentStatus = 'Paid' THEN 'Paid' ELSE 'Unpaid' END AS
payment_status
FROM Appointment a
JOIN Patients p ON a.patientID = p.patientID
JOIN User u ON p.patientID = u.userID
JOIN Contain c ON a.appointmentID = c.appointmentID
LEFT JOIN Process pr ON a.appointmentID = pr.appointmentID
LEFT JOIN Billing b ON pr.processID = b.processID
WHERE a.doctorID = ?
AND c.startTime > NOW()
AND a.status = 'Scheduled'
ORDER BY c.startTime;
```

-- Get past appointments for a doctor

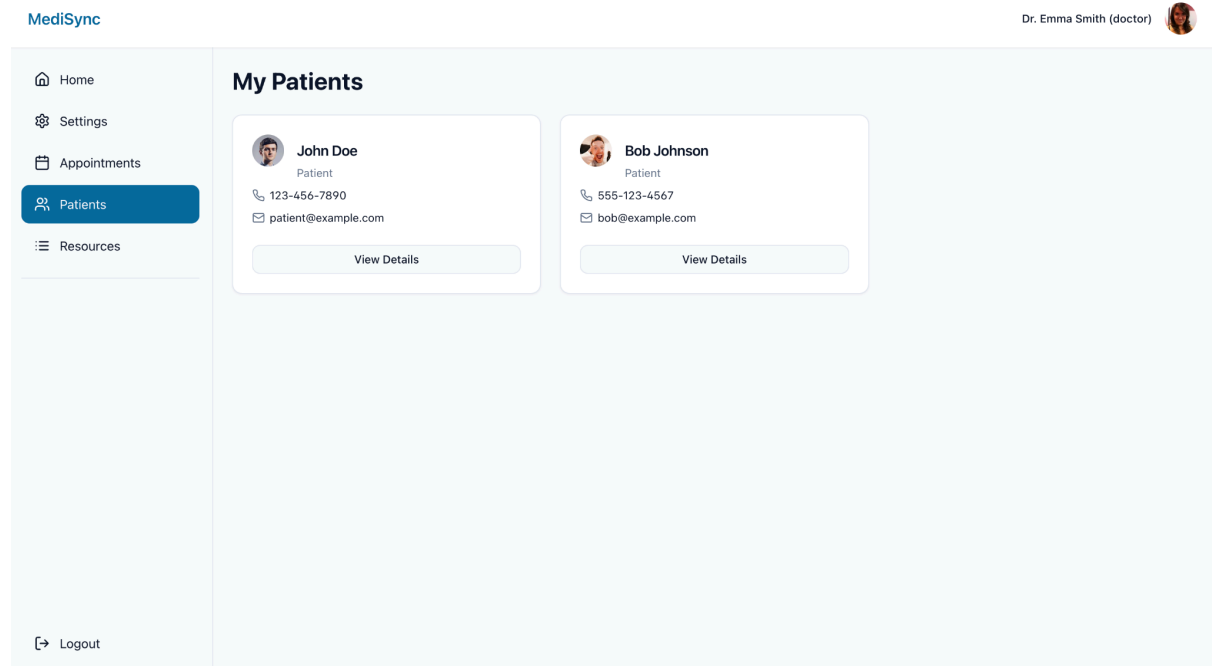
```
SELECT
  a.appointmentID,
  p.patientID,
```

```
u.name AS patient_name,  
c.startTime,  
c.endTime,  
a.status,  
'Regular check-up' AS visit_reason,  
'$150' AS amount,  
CASE WHEN b.paymentStatus = 'Paid' THEN 'Paid' ELSE 'Unpaid' END AS  
payment_status  
FROM Appointment a  
JOIN Patients p ON a.patientID = p.patientID  
JOIN User u ON p.patientID = u.userID  
JOIN Contain c ON a.appointmentID = c.appointmentID  
LEFT JOIN Process pr ON a.appointmentID = pr.appointmentID  
LEFT JOIN Billing b ON pr.processID = b.processID  
WHERE a.doctorID = ?  
AND c.startTime <= NOW()  
ORDER BY c.startTime DESC;
```

```
-- Mark appointment as completed  
UPDATE Appointment  
SET status = 'Completed'  
WHERE appointmentID = ?;
```

```
-- Cancel appointment  
UPDATE Appointment  
SET status = 'Cancelled'  
WHERE appointmentID = ?;
```

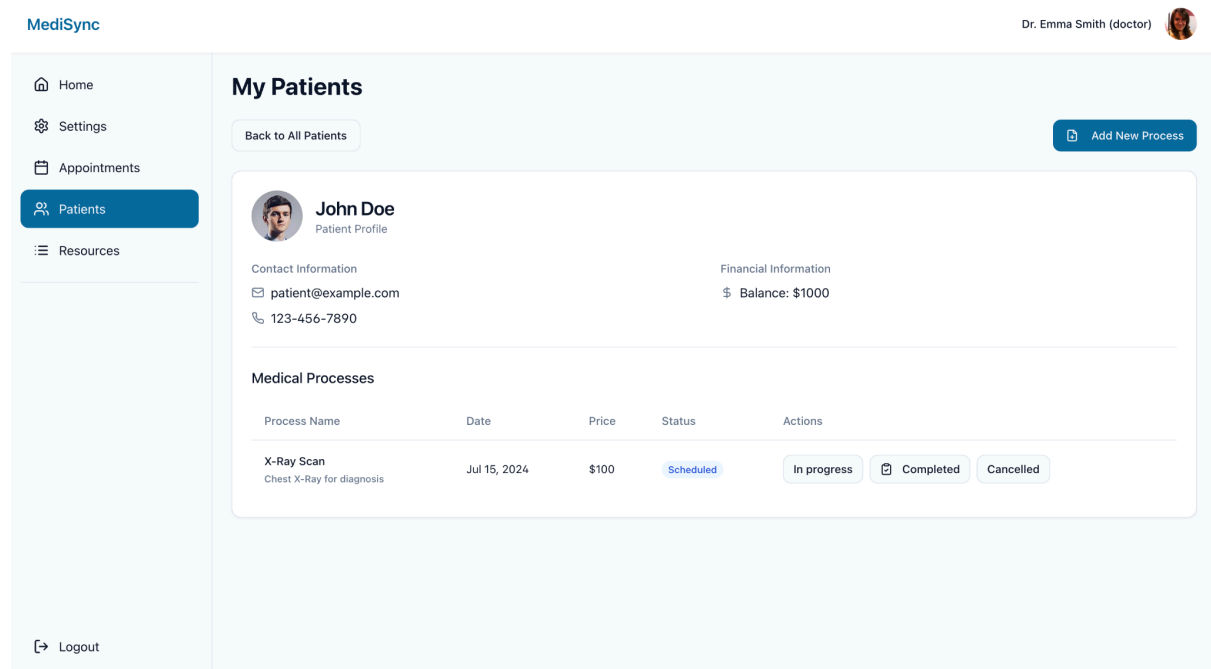
## Doctor - Patients Page



-- Get list of patients for a doctor

```
SELECT
  p.patientID,
  u.name,
  p.email,
  p.phoneNumber
FROM Patients p
JOIN User u ON p.patientID = u.userID
JOIN DoctorPatient dp ON p.patientID = dp.patientID
WHERE dp.doctorID = ?
ORDER BY u.name;
```

## Doctor - Patient Details Page



-- Get patient details

SELECT

p.patientID, u.name, p.email, p.phoneNumber, p.Balance

FROM Patients p

JOIN User u ON p.patientID = u.userID

WHERE p.patientID = ?;

-- Get medical processes for a specific patient

SELECT

pr.processID,

pr.processName,

pr.processDescription,

pr.status,

c.startTime AS date,

b.amount AS price

FROM Process pr

JOIN Appointment a ON pr.appointmentID = a.appointmentID

JOIN Contain c ON a.appointmentID = c.appointmentID

LEFT JOIN Billing b ON pr.processID = b.processID

WHERE a.patientID = ?

AND a.doctorID = ?

ORDER BY c.startTime DESC;

-- Update process status

UPDATE Process

SET status = ? -- 'Scheduled', 'In Progress', 'Completed', 'Cancelled'

WHERE processID = ?;

## Doctor - Process Adding Page

The screenshot shows the MediSync web application interface. On the left is a sidebar with navigation links: Home, Settings, Appointments, Patients (selected), and Resources. The main header shows 'MediSync' and 'Dr. Emma Smith (doctor)'. The main content area is titled 'My Patients' and shows a patient profile for 'John Doe'. A modal window titled 'Add New Medical Process' is open, prompting the user to 'Create a new medical process for this patient.' The modal contains the following fields: 'Select Appointment' (a dropdown menu showing '2024-07-15 (10:00)'), 'Process Name' (a text input with placeholder 'e.g., Blood Test, X-Ray, Physical Examination'), 'Description' (a text area with placeholder 'Describe the medical process...'), and 'Price (\$)' (a text input showing '0.00'). At the bottom of the modal are 'Cancel' and 'Add Process' buttons. In the background, a table of medical processes is partially visible, showing columns for 'Process Name', 'Status', and 'Completed', with one row for 'X-Ray Scan'.

```
-- Get appointments for a patient with a specific doctor
SELECT
    a.appointmentID, c.startTime
FROM Appointment a
JOIN Contain c ON a.appointmentID = c.appointmentID
WHERE a.patientID = ?
AND a.doctorID = ?
ORDER BY c.startTime DESC;
```


```
-- Add new medical process
INSERT INTO Process (
    processName, processDescription, status, appointmentID
)
VALUES (?, ?, 'Scheduled', ?);
```

```
-- Create billing record for the new process
INSERT INTO Billing (
    billingDate, amount, paymentStatus, processID
)
VALUES (NOW(), ?, 'Pending', LAST_INSERT_ID());
```

```
-- Update patient statistics
UPDATE PatientStatistics
SET totalProcesses = totalProcesses + 1
WHERE patientID = ?
```

AND reportDate = CURRENT\_DATE();

## Doctor - Medical Resources Page

MediSync Dr. Emma Smith (doctor) 

[Home](#)  
[Settings](#)  
[Appointments](#)  
[Patients](#)  
**[Resources](#)**  
  
[Logout](#)

### Medical Resources

**Filter Resources**

Resource Type  
All

Department  
All

☐ Show available resources only

**MRI Machine**

Imaging

Department: Radiology

Quantity: 1

Available

Reserve Resource

**Ventilator**

Life Support

Department: ICU

Quantity: 5

Available

Reserve Resource

**Surgical Kit**

Equipment

Department: Surgery

Quantity: 10

Unavailable

Reserve Resource

### 1. List All Medical Resources

SELECT

MR.resourceID,  
MR.name,  
MR.availability

FROM MedicalResources MR;

### 2. Filter by Resource Type (Name)

SELECT

MR.resourceID,  
MR.name,  
MR.availability

FROM MedicalResources MR

WHERE MR.name ILIKE '%Ventilator%'; -- or '%MRI%' etc.

### 3. Filter by Department

SELECT

MR.resourceID,  
MR.name,  
MR.availability,  
D.deptName

FROM MedicalResources MR

JOIN Request R ON MR.resourceID = R.resourceID

JOIN Doctors Doc ON R.doctorID = Doc.employeeID

JOIN Dept D ON Doc.deptName = D.deptName

WHERE D.deptName = 'Radiology';

### 4. Show Only Available Resources

```
SELECT
    MR.resourceID,
    MR.name,
    MR.availability
FROM MedicalResources MR
WHERE MR.availability = 'Available';
```

5. Reserve Resource

```
INSERT INTO Request (doctorID, resourceID, status)
VALUES (2, 3, 'Pending');
```

6. Update Resource Availability After Approval

```
UPDATE MedicalResources
SET availability = 'In Use'
WHERE resourceID = ?;
```



## Doctor - Resource Reservation Page

MediSync

Dr. Emma Smith (doctor)

Home Settings Appointments Patients Resources

Medical Resource

Filter Resources

Resource Type

All

MRI Machine

Imaging

Department: Radiology

Surgical Kit

Equipment

Department: Surgery

Logout

**Reserve Ventilator**

Complete the form below to reserve this resource

Date

Select date

Start Time

09:00

End Time

10:00

Quantity

1

Maximum: 5

**Reservation Summary**

Resource:	Ventilator
Department:	ICU
Date:	Not selected
Time:	09:00 - 10:00
Quantity:	1

Cancel Confirm Reservation

### Pre-fill Resource Info

SELECT

resourceID,

name,

availability

FROM MedicalResources

WHERE name = 'Ventilator';

Check if Doctor Already Requested This Resource

SELECT \*

FROM Request

WHERE doctorID = ? AND resourceID = ?;

Submit a Reservation Request

INSERT INTO Request (doctorID, resourceID, status)

VALUES (3, 2, 'Pending');

## Staff - Medical Resources Page

MediSync

Sarah Johnson (staff)

Home

Settings

Resources

Medical Resources

Filter Resources

Resource Type

Department

All

All

Show available resources only

MRI Machine

Imaging

Available

Department: Radiology

Quantity: 1

Reserve Resource

Ventilator

Life Support

Available

Department: ICU

Quantity: 5

Reserve Resource

Surgical Kit

Equipment

Unavailable

Department: Surgery

Quantity: 10

Reserve Resource

Logout

List All Medical Resources

```
SELECT
    resourceID,
    name,
    availability,
    quantity
FROM MedicalResources;
Filter by Resource Type (Name)
SELECT
    resourceID,
    name,
    availability,
    quantity
FROM MedicalResources
WHERE name LIKE '%Ventilator%';
Show Only Available Resources
SELECT
    resourceID,
    name,
    availability,
    quantity
FROM MedicalResources
```

```
WHERE availability = 'Available';
```

Filter by Department

```
SELECT
    resourceID,
    name,
    availability,
    quantity,
    deptName
FROM MedicalResources
WHERE deptName = 'Radiology';
```

4. Show Only Available Resources

```
SELECT
    MR.resourceID,
    MR.name,
    MR.availability
FROM MedicalResources MR
WHERE MR.availability = 'Available';
```

## Admin - Reports and Analytics Page



### Total Patients Count

```
SELECT COUNT(*) as totalPatients,
    (COUNT(*) -
    (SELECT COUNT(*) FROM Patients
    WHERE patientID IN
    (SELECT patientID FROM PatientStatistics
    WHERE reportDate BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL 2 MONTH)
    AND DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)))) /
    (SELECT COUNT(*) FROM Patients
    WHERE patientID IN
    (SELECT patientID FROM PatientStatistics
    WHERE reportDate BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL 2 MONTH)
    AND DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH))) * 100 as percentChange
FROM Patients
WHERE patientID IN (SELECT patientID FROM PatientStatistics
    WHERE reportDate > DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH));
```

### Total Appointments

```
SELECT COUNT(*) as totalAppointments,
    (COUNT(*) -
    (SELECT COUNT(*) FROM Appointment
    WHERE appointmentID IN
    (SELECT appointmentID FROM Contain
    WHERE startTime BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL 2 MONTH)
    AND DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)))) /
```

```

(SELECT COUNT(*) FROM Appointment
WHERE appointmentID IN
(SELECT appointmentID FROM Contain
WHERE startTime BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL 2 MONTH)
AND DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH))) * 100 as percentChange
FROM Appointment
WHERE appointmentID IN (SELECT appointmentID FROM Contain
WHERE startTime > DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH));

```

### **Total Revenue**

```

SELECT SUM(b.amount) as totalRevenue,
(SUM(b.amount) -
(SELECT SUM(b.amount) FROM Billing b
WHERE b.billingDate BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL 2
MONTH)
AND DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH))) /
(SELECT SUM(b.amount) FROM Billing b
WHERE b.billingDate BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL 2
MONTH)
AND DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)) * 100 as percentChange
FROM Billing b
WHERE b.billingDate > DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)
AND b.paymentStatus = 'Paid';

```

### **Medical Processes**

```

SELECT COUNT(*) as totalProcesses,
(COUNT(*) -
(SELECT COUNT(*) FROM Process
WHERE processID IN
(SELECT processID FROM Billing
WHERE billingDate BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL 2 MONTH)
AND DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)))) /
(SELECT COUNT(*) FROM Process
WHERE processID IN
(SELECT processID FROM Billing
WHERE billingDate BETWEEN DATE_SUB(CURRENT_DATE, INTERVAL 2 MONTH)
AND DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH))) * 100 as percentChange
FROM Process
WHERE processID IN (SELECT processID FROM Billing
WHERE billingDate > DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH));

```

### **Appointment Trends (Chart Data)**

```

SELECT
DATE_FORMAT(s.startTime, '%Y-%m-%d') as appointment_date,
COUNT(*) as appointment_count

```

```

FROM Appointment a
JOIN Contain c ON a.appointmentID = c.appointmentID
JOIN Slots s ON c.doctorID = s.doctorID
    AND c.slotID = s.slotID
    AND c.startTime = s.startTime
    AND c.endTime = s.endTime
WHERE s.startTime > DATE_SUB(CURRENT_DATE, INTERVAL 12 MONTH)
GROUP BY DATE_FORMAT(s.startTime, '%Y-%m-%d')
ORDER BY appointment_date;

```

### Doctor Performance Data

```

SELECT
    d.employeeID,
    u.name as doctorName,
    d.specialization,
    COUNT(a.appointmentID) as appointmentCount,
    AVG(a.rating) as averageRating,
    SUM(b.amount) as totalRevenue,
    (SELECT COUNT(*)
     FROM Prescriptions p
     WHERE p.doctorID = d.employeeID
     AND p.appointmentID IN (
         SELECT appointmentID
         FROM Contain
         WHERE startTime > DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)
     )) as prescriptionCount
FROM Doctors d
JOIN Employee e ON d.employeeID = e.employeeID
JOIN User u ON e.employeeID = u.userID
LEFT JOIN Appointment a ON d.employeeID = a.doctorID
LEFT JOIN Process pr ON a.appointmentID = pr.appointmentID
LEFT JOIN Billing b ON pr.processID = b.processID
WHERE b.billingDate > DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)
    OR b.billingDate IS NULL
GROUP BY d.employeeID, u.name, d.specialization
ORDER BY totalRevenue DESC;

```

### Revenue Breakdown by Department

```

SELECT
    d.deptName,
    SUM(b.amount) as totalRevenue,
    COUNT(DISTINCT a.appointmentID) as appointmentCount,
    COUNT(DISTINCT a.patientID) as patientCount,
    SUM(b.amount) / COUNT(DISTINCT a.appointmentID) as
averageRevenuePerAppointment
FROM Doctors doc

```

```
JOIN Dept d ON doc.deptName = d.deptName
JOIN Appointment a ON doc.employeeID = a.doctorID
JOIN Process pr ON a.appointmentID = pr.appointmentID
JOIN Billing b ON pr.processID = b.processID
WHERE b.billingDate > DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)
GROUP BY d.deptName
ORDER BY totalRevenue DESC;
```

### **Filter by Time Period (Monthly dropdown)**

-- This would modify the WHERE clauses in the above queries

-- For example, to filter for the last week:

```
WHERE b.billingDate > DATE_SUB(CURRENT_DATE, INTERVAL 1 WEEK)
```

-- For the last month:

```
WHERE b.billingDate > DATE_SUB(CURRENT_DATE, INTERVAL 1 MONTH)
```

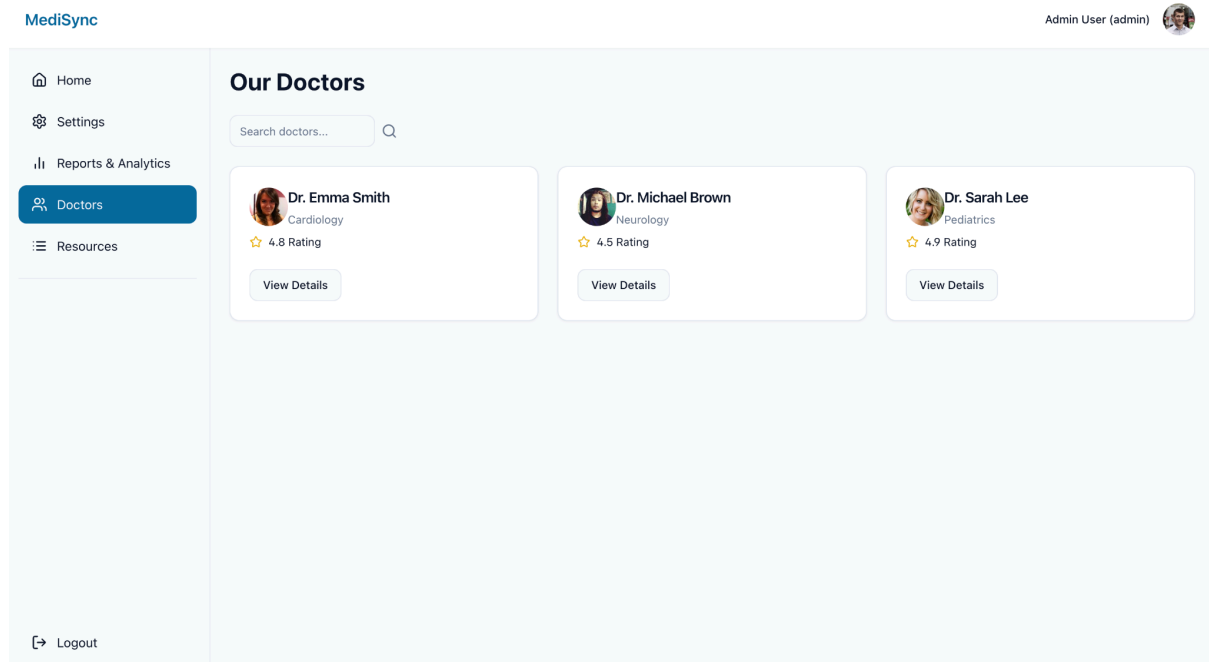
-- For the last quarter:

```
WHERE b.billingDate > DATE_SUB(CURRENT_DATE, INTERVAL 3 MONTH)
```

-- For the last year:

```
WHERE b.billingDate > DATE_SUB(CURRENT_DATE, INTERVAL 1 YEAR)
```

## Admin - Doctor Listing Page



### Fetching All Doctors with Basic Information

```
SELECT d.employeeID, u.name, d.specialization, AVG(a.rating) as avgRating
FROM Doctors d
JOIN Employee e ON d.employeeID = e.employeeID
JOIN User u ON e.employeeID = u.userID
LEFT JOIN Appointment a ON d.employeeID = a.doctorID
GROUP BY d.employeeID, u.name, d.specialization
ORDER BY u.name;
```

### Search Functionality for Doctors

```
SELECT d.employeeID, u.name, d.specialization, AVG(a.rating) as avgRating
FROM Doctors d
JOIN Employee e ON d.employeeID = e.employeeID
JOIN User u ON e.employeeID = u.userID
LEFT JOIN Appointment a ON d.employeeID = a.doctorID
WHERE u.name LIKE ? OR d.specialization LIKE ?
GROUP BY d.employeeID, u.name, d.specialization
ORDER BY u.name;
```

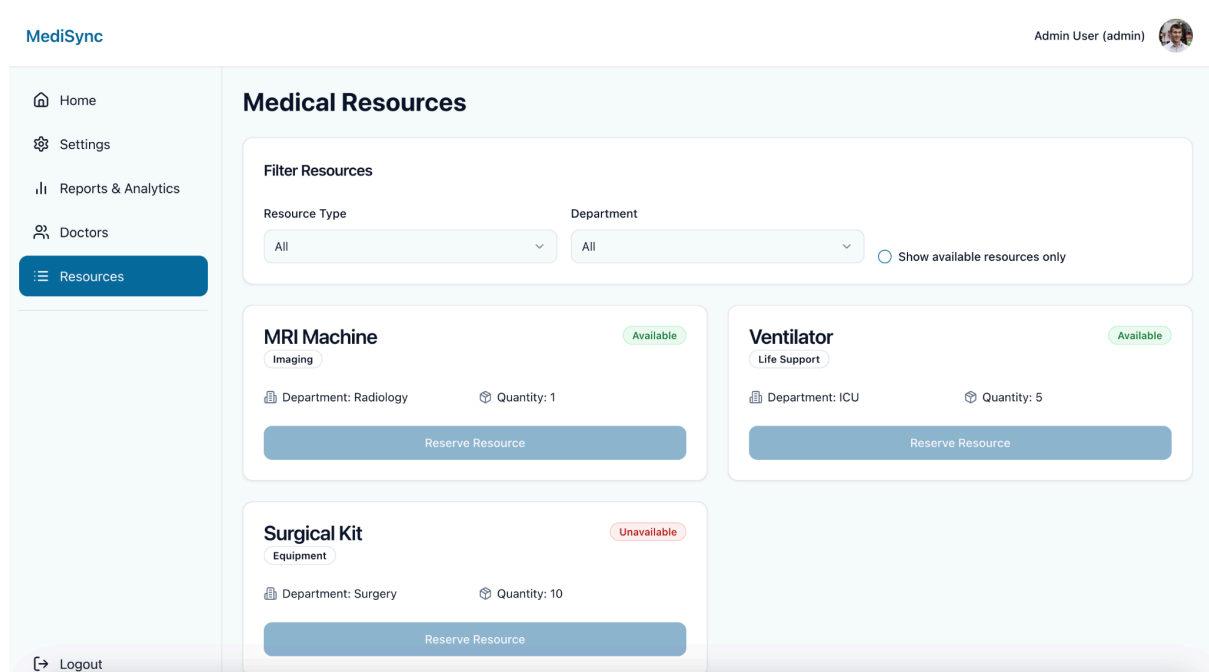
### Getting Doctor Details (for "View Details" button)

```
SELECT d.employeeID, u.name, d.specialization, d.doctorLocation, d.deptName,
       AVG(a.rating) as avgRating, COUNT(a.appointmentID) as totalAppointments,
       ds.prescriptionCount, ds.totalRevenue
FROM Doctors d
JOIN Employee e ON d.employeeID = e.employeeID
```



```
JOIN User u ON e.employeeID = u.userID
LEFT JOIN Appointment a ON d.employeeID = a.doctorID
LEFT JOIN DoctorStatistics ds ON d.employeeID = ds.doctorID
WHERE d.employeeID = ? -- Selected doctor's ID
GROUP BY d.employeeID, u.name, d.specialization, d.doctorLocation, d.deptName,
         ds.prescriptionCount, ds.totalRevenue;
```

## Admin - Medical Resources Page



### Fetching All Medical Resources

```
SELECT mr.resourceID, mr.name, mr.availability,  
       eq.usageCount, eq.lastUsedDate, eq.totalRequests  
FROM MedicalResources mr  
LEFT JOIN EquipmentStatistics eq ON mr.resourceID = eq.equipmentID;
```

### Filtering Resources by Type

```
SELECT mr.resourceID, mr.name, mr.availability  
FROM MedicalResources mr  
WHERE mr.name LIKE '%Machine%' -- For filtering by type (e.g., MRI Machine)  
ORDER BY mr.name;
```

### Filtering Resources by Department

```
SELECT mr.resourceID, mr.name, mr.availability  
FROM MedicalResources mr  
JOIN Request req ON mr.resourceID = req.resourceID  
JOIN Doctors d ON req.doctorID = d.employeeID  
WHERE d.deptName = 'Radiology'; -- Replace with the selected department
```

### Showing Only Available Resources

```
SELECT resourceID, name, availability  
FROM MedicalResources  
WHERE availability = 'Available';
```

### **Getting Resource Quantity Information**

```
SELECT mr.name, COUNT(*) as quantity
FROM MedicalResources mr
GROUP BY mr.name;
```

### **Resource Reservation (When "Reserve Resource" button is clicked)**

```
-- First check availability
SELECT availability
FROM MedicalResources
WHERE resourceID = ?;

-- Then update status if available
UPDATE MedicalResources
SET availability = 'In Use'
WHERE resourceID = ? AND availability = 'Available';

-- Then insert a new request entry
INSERT INTO Request (doctorID, resourceID, status)
VALUES (?, ?, 'Approved');

-- Update equipment statistics
UPDATE EquipmentStatistics
SET usageCount = usageCount + 1,
    lastUsedDate = CURRENT_DATE,
    totalRequests = totalRequests + 1
WHERE equipmentID = ?;
```

### **Adding a New Medical Resource (Admin functionality)**

```
INSERT INTO MedicalResources (name, availability)
VALUES (?, 'Available');

-- Initialize equipment statistics
INSERT INTO EquipmentStatistics (usageCount, lastUsedDate, totalRequests,
equipmentID)
VALUES (0, NULL, 0, LAST_INSERT_ID());
```

### **Getting Resource Details with Department Information**

```
SELECT mr.resourceID, mr.name, mr.availability, d.deptName
FROM MedicalResources mr
JOIN Request req ON mr.resourceID = req.resourceID
JOIN Doctors doc ON req.doctorID = doc.employeeID
JOIN Dept d ON doc.deptName = d.deptName
WHERE mr.resourceID = ?;
```

### 3.Implementation Plan

The implementation of the Hospital Appointment Management System will be carried out using a modern web-based architecture. The backend of the system will be developed using FastAPI, a high-performance Python framework that allows for the efficient creation of RESTful APIs. The frontend will be built with React.js, providing a responsive and dynamic user interface for different user roles including patients, doctors, staff, and administrators. The system will use PostgreSQL as the relational database management system due to its stability, support for advanced SQL features such as views, triggers, and constraints, and compatibility with raw SQL queries, which is a requirement for this project.

For the development environment, standard personal computing hardware will be used, including machines with at least 8 GB of RAM, Intel or Apple Silicon processors, and modern operating systems such as Windows 10/11, macOS, or Linux distributions. All backend operations interacting with the database will be executed using manually written raw SQL queries, without relying on any ORM features. The SQL statements will be embedded within the FastAPI routes and executed using PostgreSQL's native query interface via Python.

Development tasks are divided into three core components: database setup, backend API development, and frontend user interface design. The database schema will be implemented directly in PostgreSQL using SQL scripts. Backend tasks include writing routes for user authentication, appointment handling, resource management, and administrative reports, all using raw SQL. The frontend tasks involve designing and implementing interactive pages such as login/sign-up, appointment booking, dashboards, and analytics views using React. The system will be tested locally using tools like Postman for backend endpoints and browser-based testing for the frontend. Version control will be managed using GitHub.