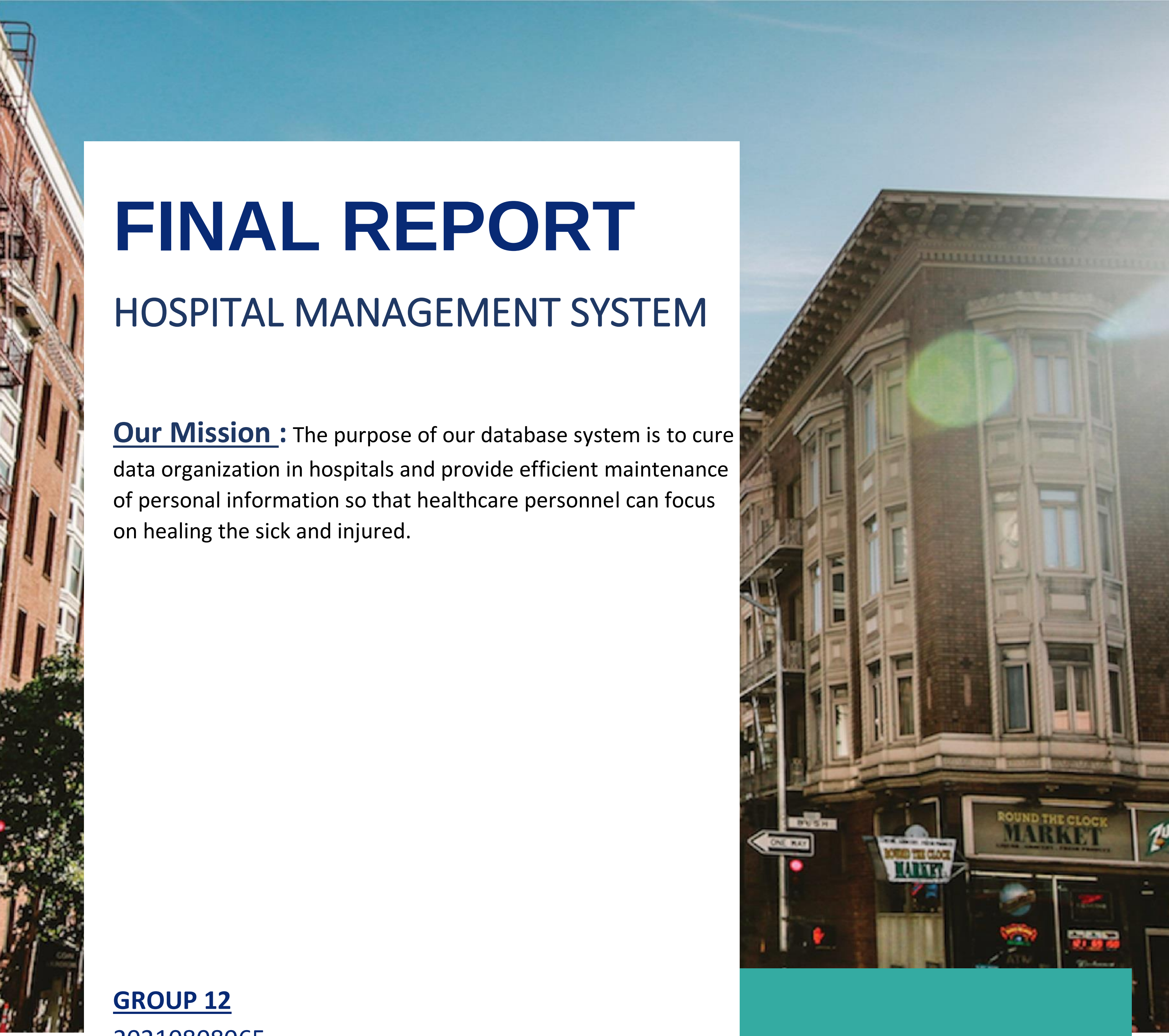


FINAL REPORT

HOSPITAL MANAGEMENT SYSTEM

Our Mission : The purpose of our database system is to cure data organization in hospitals and provide efficient maintenance of personal information so that healthcare personnel can focus on healing the sick and injured.

GROUP 12
20210808065
20200808035
202351056503
20220808611
20190808092
20220808070

1.KICKSTARTER

Hospitals are trusted to provide the best care and treatment for the sick and injured in our communities. Because they do some of the most important work in society, we want to aid them in providing a smooth-working and easy-to-understand management system for keeping track of all patients, staff, appointments, prescriptions, and treatments. Our database system will streamline the efforts and care that goes into the administrative processes of hospital work in order to free up all involved parties to focus on the important tasks of healing our sick and injured.

2.AUDIENCE

We are here, addressing you investors and managers, as well as the regular users of hospital data organization, seeking to help you in helping our society. Our management system is designed to be used by doctors, nurses, secretaries, and any other administrators at a hospital or medical center. This system is simple to learn, easy to implement, and all-encompassing for hospital files.
The simplicity to learn is important so that very little time and energy will need to be given to administrative tasks. It is easy to implement, saving time in accessing medical records. And it is all-encompassing, making this database system the one stop to access any and all important information for our medical professionals and their patients.

3.HOW WILL WE ACHIEVE OUR GOAL & FACT FINDING TECHNIQUES

In order to cure the necessary data organization, we have talked with medical professionals and administrators, learning their greatest needs and wants in a digital management system.

Through interviewing some doctors, we learned that they need a place to record information for each patient such as results from radiology and similar tests, prescribed medication, diagnoses, and medical history including past surgeries and operations, as well as personal information and blood type. Currently, many hospitals have similar systems in place, but they might need to go different systems or websites to get all the necessary information. From our research, the current systems provide doctors with certain permissions, such as adding tests and recording prescribed medications, but only the hospital has the right to schedule and organize appointments. If this is a hospital's goal, our system makes it possible to put limits on what each staff can access. On the other hand, if a hospital would like to provide more rights to the doctor for their own appointments and scheduling, our system makes that possible as well. Our system also incorporates a central location for patient information, according to what they are willing or required to provide to the doctors. Many hospitals currently rely on e-nabiz, which is a great application but greatly limits the information that the medical professionals can see, so with the patient's permission, our system can quickly and securely provide the medical history and personal information that is requested by the medical professionals.

We have sought to incorporate the greatest desires for this type of system, and we are flexible to make this system work for your hospital and your specific needs.

4.WHY CHOOSE US

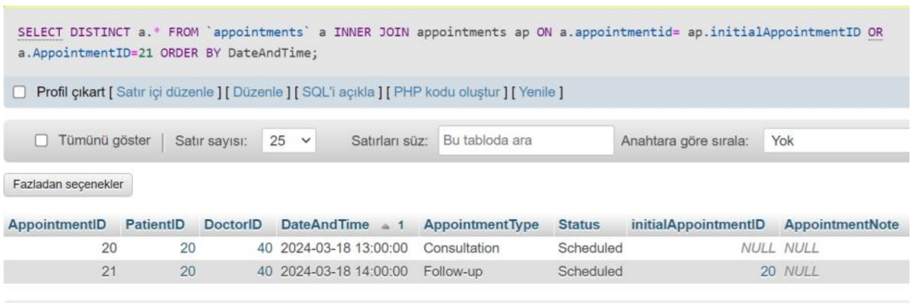
We have worked diligently and intentionally to create this easy-to-use and smooth-working system, so that your diligence and intentionality can be better spent with your patients and clientele. This system is organized and able to be viewed by possibly any organization system that you can imagine, for your needed information should be at your fingertips at all times. In this organization, all of your doctors and patients' information can be accessed by the administrators whom you have been granted permission, providing a layer of security for all involved parties.

5.REQUESTED EXPANSION OF SYSTEM FEATURES BASED ON CUSTOMER REVIEW

- Customer would like the system to keep track of appointments that are considered as follow-ups to previous appointments. These modifications enable the system to effectively manage and distinguish follow-up appointments from initial appointments, providing enhanced functionality and tracking capabilities for appointment scheduling and better management of patient follow-ups.
- To accommodate the new functionality, the appointments table has been updated with additional columns and a foreign key constraint. The following changes were made:
    1. InitialAppointmentID column has been added which stores the ID of the initial appointment to which a follow-up appointment is linked. AppointmentNote column has been added which allows for any additional notes regarding the appointment. A foreign key constraint (fk_initial_appointment) has been added to the initialAppointmentID column, linking it to the AppointmentID column within the same table. 'ON DELETE CASCADE' and 'ON UPDATE CASCADE' options ensures **referential integrity** by enforcing that every initialAppointmentID (except for the very first appointment) must correspond to an existing appointment.
    2. To demonstrate the new functionality, consider the following example where a follow-up appointment is inserted into the system. This example shows an appointment with ID 21 that is a follow-up to an initial appointment with ID 20.
    3. To retrieve a list of appointments that includes both follow-up and their corresponding initial appointments, the following SQL query can be used. This query ensures that appointments are ordered by their scheduled date and time.

APPOINTMENTS TABLE:

```sql
1. ALTER TABLE appointments ADD COLUMN initialAppointmentID INT, ADD COLUMN
   AppointmentNote VARCHAR(255), ADD CONSTRAINT fk_initial_appointment FOREIGN KEY
   (initialAppointmentID)  REFERENCES appointments(AppointmentID) ON DELETE CASCADE ON
   UPDATE CASCADE;
2. INSERT INTO Appointments (AppointmentID, PatientID, DoctorID, DateAndTime,
   AppointmentType, Status, initialAppointmentID, AppointmentNote) VALUES (21, 20,
   40, '2024-03-18 14:00:00', 'Follow-up', 'Scheduled', 20, '' );
3. SELECT DISTINCT a.* FROM `appointments` a INNER JOIN appointments ap ON
   a.appointmentid= ap.initialAppointmentID OR a.AppointmentID=[AppointmentID] ORDER
   BY DateAndTime;
```



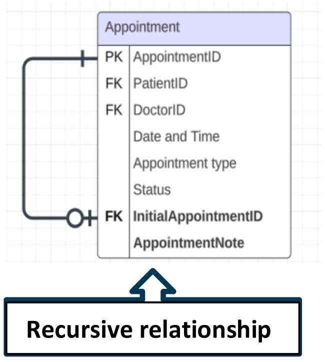
```
SELECT DISTINCT a.* FROM `appointments` a INNER JOIN appointments ap ON a.appointmentid= ap.initialAppointmentID OR
a.AppointmentID=21 ORDER BY DateAndTime;
```

Profil çıkart [Satır içi düzenle] [Düzenle] [SQL'i açıkla] [PHP kodu oluştur] [Yenile]

Tümünü göster | Satır sayısı: 25 Satırları süz: Bu tabloda ara Anahtara göre sırala: Yok

Fazladan seçenekler

AppointmentID	PatientID	DoctorID	DateAndTime ▲ 1	AppointmentType	Status	initialAppointmentID	AppointmentNote
20	20	40	2024-03-18 13:00:00	Consultation	Scheduled	NULL	NULL
21	20	40	2024-03-18 14:00:00	Follow-up	Scheduled	20	NULL



1. With the changes made, the hospital database system now automatically adds patients to the database at the time of their registration, while also recording the registration date and time. To achieve this, a new column named "registration_date" was added to the "Patients" table to store the date and time of registration. Additionally, "UserID" column (each patient is first added to the user table as a user of the system before their registration as a patient) was set to auto-increment using the AUTO_INCREMENT attribute. These modifications enhance the functionality of the database system, enabling more accurate and efficient tracking of patient registration processes.
2. Instead of using the names of the patient's relatives in the emergency contact column, we utilized their contact information.
3. In order to collect the blood type information of patients, we created a blood type table and added a column named 'bloodTypeID' to the patient table.

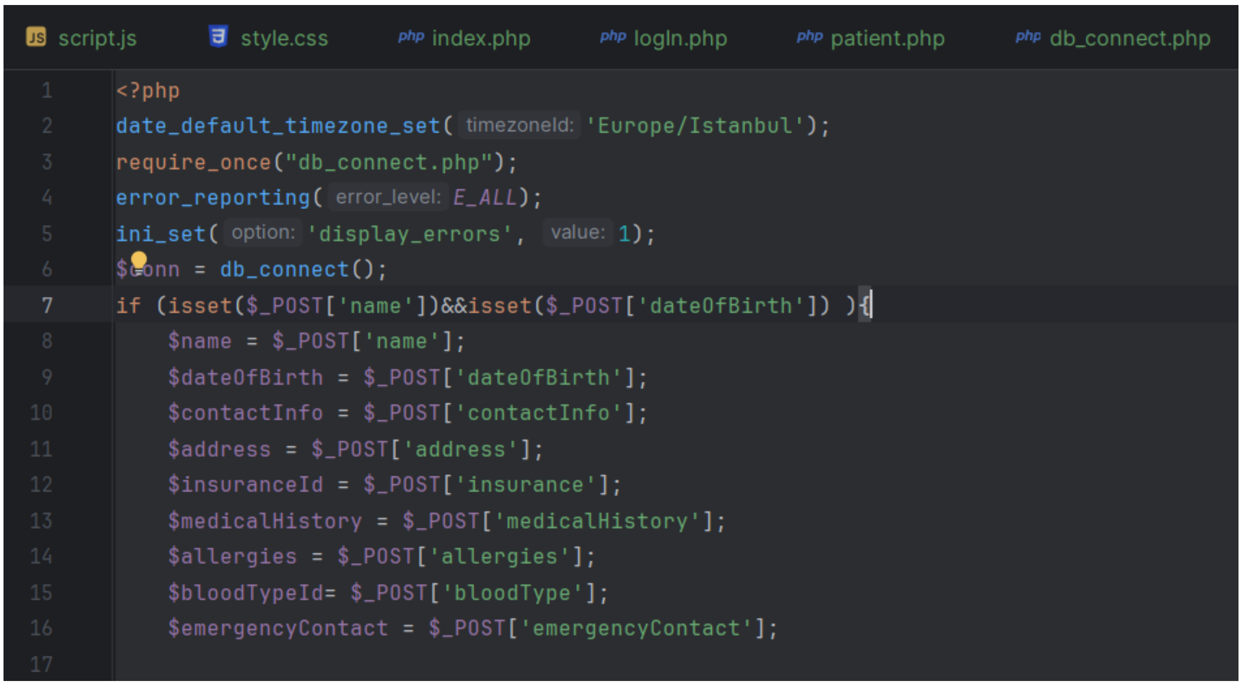
PATIENTS/ USERS TABLE:

1.
```
ALTER TABLE Patients ADD COLUMN registration_date DATETIME;

ALTER TABLE Patients MODIFY COLUMN PatientID INT;

SET @counter = 20;
UPDATE Patients SET PatientID = (@counter := @counter + 1);
ALTER TABLE Users MODIFY COLUMN UserID INT AUTO_INCREMENT;
```

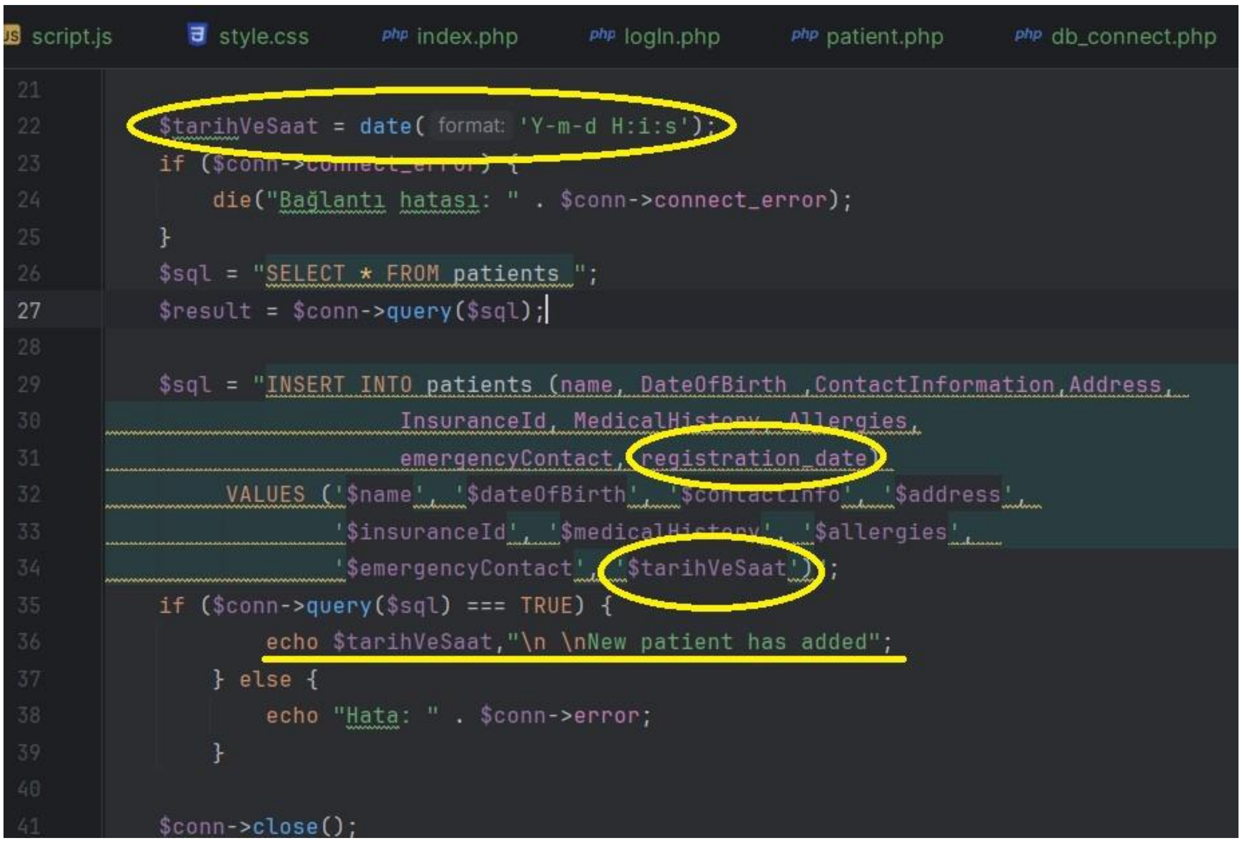
2.
```
START TRANSACTION;
UPDATE patients SET EmergencyContact = CASE WHEN patientID = 1 THEN
'5551234567' WHEN patientID = 2 THEN '5559876543' WHEN patientID = 3 THEN
'5551112233' WHEN patientID = 4 THEN '5554445566' WHEN patientID = 5 THEN
'5557778899' WHEN patientID = 6 THEN '5556667788' WHEN patientID = 7 THEN
'5553332211' WHEN patientID = 8 THEN '5559998877' WHEN patientID = 9 THEN
'5552223344' WHEN patientID = 10 THEN '5558887766' WHEN patientID = 11 THEN
'5556677889' WHEN patientID = 12 THEN '5555544332' WHEN patientID = 13 THEN
'5559988776' WHEN patientID = 14 THEN '5551122334' WHEN patientID = 15 THEN
'5558877665' WHEN patientID = 16 THEN '5552233441' WHEN patientID = 17 THEN
'5556677889' WHEN patientID = 18 THEN '5551122334' WHEN patientID = 19 THEN
'5558899776' WHEN patientID = 20 THEN '5554455667' END;
COMMIT;
```

3. CREATE TABLE bloodtype (bloodtypeID INT PRIMARY KEY, bloodtype VARCHAR(10);
   INSERT INTO bloodtype (bloodtypeID, bloodtype) VALUES
   (1, 'A+'), (2, 'A-'), (3, 'B+'), (4, 'B-'), (5, 'AB+'), (6, 'AB-'), (7, 'O+'), (8, 'O-');
   ALTER TABLE patients ADD COLUMN bloodtypeID INT;
```
ALTER TABLE patients ADD CONSTRAINT fk_patient_bloodtype FOREIGN KEY (bloodtypeID)
REFERENCES bloodtype(bloodtypeID);
```

- PHP Code for Automatic Registration and Date Tracking:



```php
<?php
date_default_timezone_set( timezoneId: 'Europe/Istanbul');
require_once("db_connect.php");
error_reporting( error_level: E_ALL);
ini_set( option: 'display_errors', value: 1);
$conn = db_connect();
if (isset($_POST['name'])&&isset($_POST['dateOfBirth']) ){
    $name = $_POST['name'];
    $dateOfBirth = $_POST['dateOfBirth'];
    $contactInfo = $_POST['contactInfo'];
    $address = $_POST['address'];
    $insuranceId = $_POST['insurance'];
    $medicalHistory = $_POST['medicalHistory'];
    $allergies = $_POST['allergies'];
    $bloodTypeId= $_POST['bloodType'];
    $emergencyContact = $_POST['emergencyContact'];

```



```php

    $tarihVeSaat = date( format: 'Y-m-d H:i:s');
    if ($conn->connect_error) {
        die("Bağlantı hatası: " . $conn->connect_error);
    }
    $sql = "SELECT * FROM patients ";
    $result = $conn->query($sql);

    $sql = "INSERT INTO patients (name, DateOfBirth ,ContactInformation,Address,
                        InsuranceId, MedicalHistory, Allergies,
                        emergencyContact, registration_date)
            VALUES ('$name', '$dateOfBirth', '$contactInfo', '$address',
                    '$insuranceId', '$medicalHistory', '$allergies',
                    '$emergencyContact', '$tarihVeSaat')";
    if ($conn->query($sql) === TRUE) {
            echo $tarihVeSaat,"\n \nNew patient has added";
        } else {
            echo "Hata: " . $conn->error;
        }

    $conn->close();
```

6.CLIENT-REQUESTED ADJUSTMENTS TO ORIGINAL ER DIAGRAM

- Customer would like to make sure that in the final system, we can have other types of people working at the hospital (nurses, nurse practitioners, etc.) who can see and treat patients. Additionally, the system should handle administrative roles and support staff within the hospital infrastructure.

```
--Managers
INSERT INTO Staff (StaffID, Name, PositionRole, DepartmentID, ShiftSchedule,
EmploymentStatus, SalaryInformation, TrainingHistory, LeaveRecords, PerformanceReviews,
ContactInformation) VALUES (59, 'David Johnson', 'Manager', NULL, NULL, 'Full-time',
100000.00, 'Completed management training', NULL, 'Excellent performance', '999-999-9999');
INSERT INTO Staff (StaffID, Name, PositionRole, DepartmentID, ShiftSchedule,
EmploymentStatus, SalaryInformation, TrainingHistory, LeaveRecords, PerformanceReviews,
ContactInformation) VALUES (60, 'Rachel Smith', 'Manager', NULL, NULL, 'Full-time',
95000.00, 'Completed leadership seminar', NULL, 'Above expectations', '888-888-8888');
```

```
-- Patient Registrar
INSERT INTO Staff (StaffID, Name, PositionRole, DepartmentID, ShiftSchedule,
EmploymentStatus, SalaryInformation, TrainingHistory, LeaveRecords, PerformanceReviews,
ContactInformation) VALUES (61, 'Emily Wilson', 'Patient Registrar', NULL, NULL, 'Full-
time', 60000.00, 'Completed patient management training', NULL, 'Satisfactory performance',
'777-777-7777');
```

```
-- Accountant
INSERT INTO Staff (StaffID, Name, PositionRole, DepartmentID, ShiftSchedule,
EmploymentStatus, SalaryInformation, TrainingHistory, LeaveRecords, PerformanceReviews,
ContactInformation) VALUES (62, 'Michael Thompson', 'Accountant', NULL, NULL, 'Full-
time', 80000.00, 'Certified in accounting', NULL, 'Outstanding performance', '666-666-
6666');
```

```
-- IT Support Specialist
INSERT INTO Staff (StaffID, Name, PositionRole, DepartmentID, ShiftSchedule,
EmploymentStatus, SalaryInformation, TrainingHistory, LeaveRecords, PerformanceReviews,
ContactInformation) VALUES (63, 'Jessica Miller', 'IT Support Specialist', NULL, NULL,
'Full-time', 70000.00, 'Completed IT troubleshooting course', NULL, 'Satisfactory
performance', '555-555-5555');
```

Implementation Details: To meet the requirements, we structured the database to include a comprehensive Staff table to capture common attributes for all staff members. We then created specific tables for different roles, such as Nurse and LabTechnician, linking them back to the Staff table to ensure data integrity and referential relationships. **These roles are now integrated into the system with the capability to see and treat patients.**

```
CREATE TABLE Nurse ( NurseID INT PRIMARY KEY, NurseSpecialization VARCHAR(255),
NurseCertifications VARCHAR(255), FOREIGN KEY (NurseID) REFERENCES Staff(StaffID) ON DELETE
CASCADE);
CREATE TABLE LabTechnician ( LabTechnicianID INT PRIMARY KEY,
LabTechnicianSpecialization VARCHAR(255), LabTechnicianCertifications VARCHAR(255), FOREIGN
KEY (LabTechnicianID) REFERENCES Staff(StaffID) ON DELETE CASCADE);
```

```
INSERT INTO Nurse (NurseID, NurseSpecialization, NurseCertifications) SELECT StaffID,
'', '' FROM Staff WHERE PositionRole = 'Nurse';
INSERT INTO LabTechnician (LabTechnicianID, LabTechnicianSpecialization,
LabTechnicianCertifications) SELECT StaffID, '', '' FROM Staff WHERE PositionRole = 'Lab
Technician';
```

```
ALTER TABLE wards ADD CONSTRAINT fk_nurses_wards FOREIGN KEY (nurseIncharge) REFERENCES
Nurse(NurseID);
ALTER TABLE labtests ADD CONSTRAINT fk_labtechnician_tests FOREIGN KEY (LabTechnicianID)
REFERENCES LabTechnician(LabTechnicianID);
```

- The ER diagram has been updated to include the addition of the users and userType tables. These tables are crucial for identifying the individuals who will interact with the database. The userType table serves an essential role in determining the types of interfaces that will be developed for users. When patients are registered at the hospital and become users of the system, their userID is linked directly to their patientID, creating a hierarchical relationship between these two tables. However, not all staff members listed in the staff table will necessarily be users of the system. For example, roles like cleaning staff may not require access to the database. To accommodate this variation, a new userID column has been introduced in the staff table. This column serves as a reference to the user table and establishes a one-to-one relationship. Consequently, the userID in the staff table may remain null for staff members who are not users of the system.

```
CREATE TABLE usertype ( userTypeID INT PRIMARY KEY, role VARCHAR(50) NOT NULL );
CREATE TABLE users ( userID INT PRIMARY KEY, userTypeID INT,
username VARCHAR(50) NOT NULL UNIQUE, password VARCHAR(255) NOT NULL,
FOREIGN KEY (userTypeID) REFERENCES usertype(userTypeID) );
```

```
INSERT INTO usertype (userTypeID, role) VALUES (1, 'Nurse'), (2, 'Physician'), (3, 'Lab
Technician'), (4, 'Doctor'), (5, 'Manager'), (6, 'Patient Registrar'), (7, 'Accountant'),
(8, 'IT Support Specialist'), (9, 'Patient');
```

```
ALTER TABLE staff ADD COLUMN userID INT, ADD CONSTRAINT fk_user FOREIGN KEY (userID)
REFERENCES users(userID);
ALTER TABLE patients ADD CONSTRAINT fk_user_patientt FOREIGN KEY (patientID) REFERENCES
users(userID);
```

- With the changes shown below, the unnecessary relationship between patientcare and patient has been eliminated. Since each admission in the admission table already represents one and only one patient, the admissionID was used as the primary key of the patientcare table as each admission is associated with only one patientcare, and the patientID column was removed.

```
ALTER TABLE patientcare DROP PRIMARY KEY;
```

```
-- Checking that each entry in the 'admission' table represents only one patient:
SELECT admissionID, COUNT(patientID) AS patient_count FROM admissions
GROUP BY admissionID HAVING patient_count > 1;
-- Checking that each entry in the 'admission' table is associated with only one 'patientcare'
record:
SELECT admissionID FROM admissions WHERE admissionID NOT IN
(SELECT DISTINCT admissionID FROM patientcare);
```

```
ALTER TABLE patientcare ADD PRIMARY KEY (admissionID);
ALTER TABLE patientcare DROP COLUMN patientID;
```

- The modifications shown below significantly enhance the management of nurse and patient care. Firstly, normalizing the CarePlan data in the PatientCare table and moving it to a separate CareType table improves data integrity and eliminates redundancy, allowing for more effective management and analysis of care plans. Additionally, adding CareTypeID and CareProviderID columns to the PatientCare table links each care plan to a specific type and the nurse responsible for it.
- Using **CASCADE** options for ‘admissionId’ will help **maintaining referential integrity** by automatically updating or deleting related records when the referenced key (admissionID) is updated or deleted.

```
ALTER TABLE PatientCare DROP PRIMARY KEY;
INSERT INTO PatientCare (AdmissionID, CarePlan, NursingNotes, VitalSigns,
MedicationAdministrationRecords, ProgressReports) SELECT AdmissionID,
TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(CarePlan, ',', n.digit+1), ',', -1)) AS CarePlan,
NursingNotes, VitalSigns, MedicationAdministrationRecords, ProgressReports FROM PatientCare
JOIN (SELECT 0 AS digit UNION ALL SELECT 1 UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL
SELECT 4) AS n ON LENGTH(CarePlan) - LENGTH(REPLACE(CarePlan, ',', '')) >= n.digit;
DELETE FROM PatientCare WHERE CarePlan LIKE '%,%';
```

```
CREATE TABLE temp_table00 AS SELECT DISTINCT AdmissionID, CarePlan, NursingNotes,
VitalSigns, MedicationAdministrationRecords, ProgressReports FROM PatientCare;
DELETE FROM PatientCare;
INSERT INTO PatientCare (AdmissionID, CarePlan, NursingNotes, VitalSigns,
MedicationAdministrationRecords, ProgressReports) SELECT * FROM temp_table00;
```

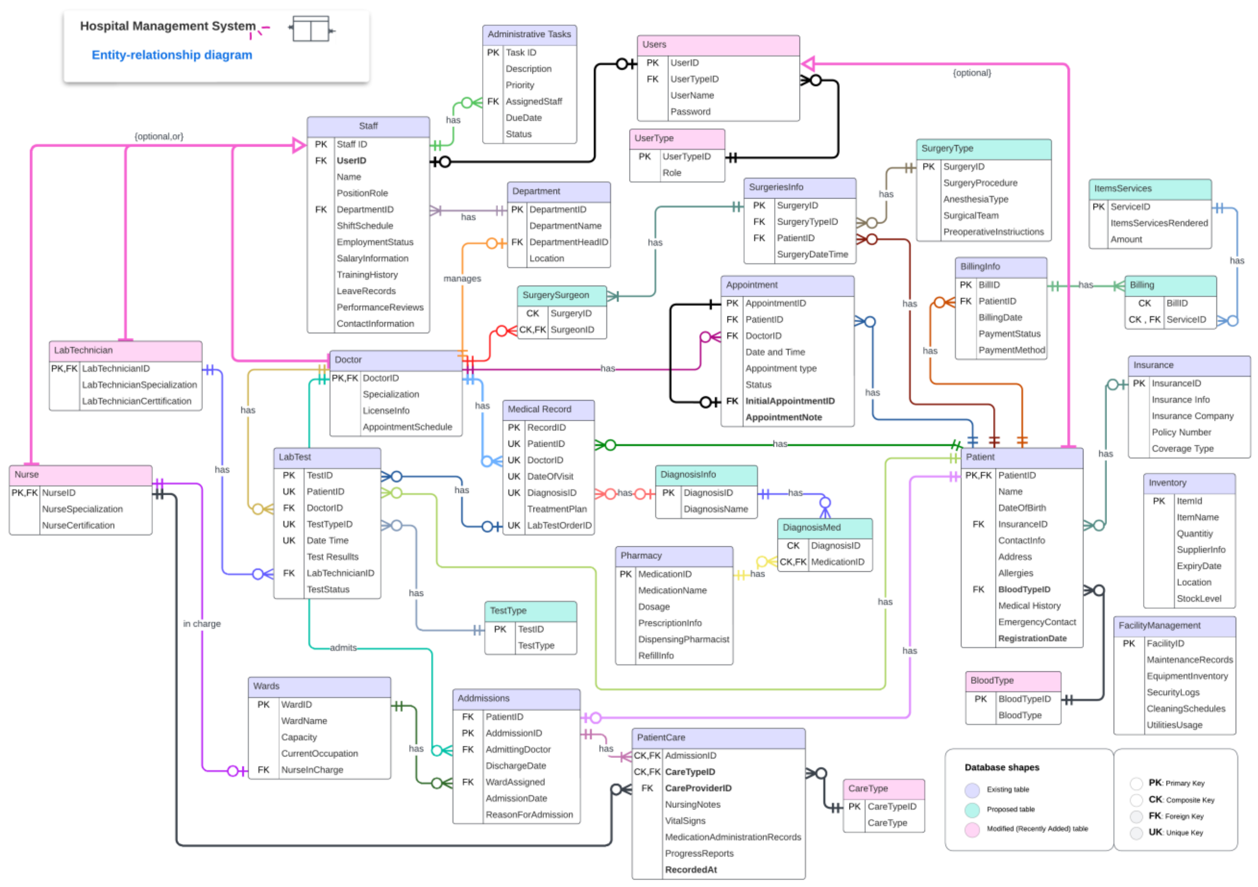
```
CREATE TABLE CareType ( CareTypeID INT AUTO_INCREMENT PRIMARY KEY, CarePlan
VARCHAR(255) UNIQUE );
INSERT INTO CareType (CarePlan) SELECT DISTINCT CarePlan FROM PatientCare;
```

```
ALTER TABLE PatientCare ADD COLUMN CareTypeID INT;
UPDATE PatientCare pc SET pc.CareTypeID = ( SELECT ct.CareTypeID FROM CareType ct WHERE
pc.CarePlan = ct.CarePlan );
```

```
ALTER TABLE PatientCare DROP COLUMN CarePlan;
ALTER TABLE PatientCare ADD CONSTRAINT fk_caretype FOREIGN KEY (CareTypeID) REFERENCES
CareType(CareTypeID);
ALTER TABLE patientcare ADD CONSTRAINT fk_admission2 FOREIGN KEY (admissionID)
REFERENCES admissions(AdmissionID) ON DELETE CASCADE ON UPDATE CASCADE;
ALTER TABLE PatientCare ADD PRIMARY KEY (admissionID, CareTypeID);
ALTER TABLE PatientCare ADD COLUMN CareProviderID INT, ADD CONSTRAINT fk_nurse FOREIGN
KEY (CareProviderID) REFERENCES Nurse(NurseID);
ALTER TABLE patientcare ADD COLUMN RecordedAt DATETIME;
```

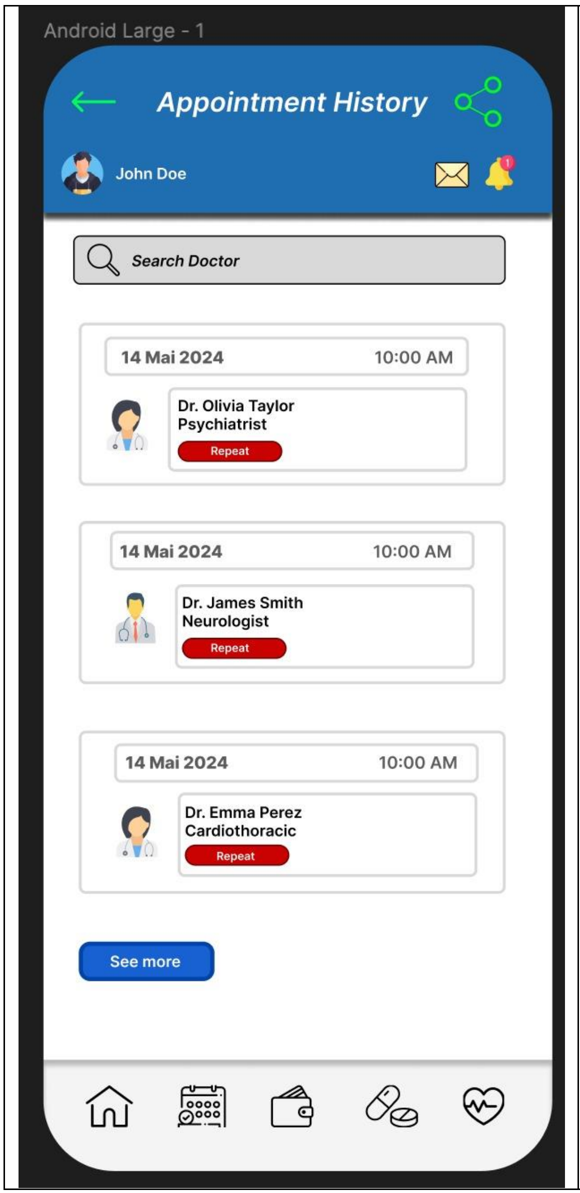
NOTE: The customer also wants to ensure that it's clear which patients are attended to by nurses, staff conducting tests, or doctors with scheduled surgeries. Regarding patient care, each nurse is responsible for the care provided to patients, establishing a direct relationship within the PatientCare table. Each entry in the PatientCare table includes a CareProviderID, which links to the nurse responsible for the patient's care. Consequently, it's easy to determine which nurse is attending to which patient. Similarly, through the foreign keys for lab technician IDs, doctor IDs, and patient IDs in the LabTests table, we can access information on which doctor ordered a test for a patient and which lab technician is responsible for conducting it. Likewise, in the SurgeriesInfo, Appointments, and MedicalRecords tables, the foreign keys for doctor IDs and patient IDs allow us to identify which doctor is attending to which patient.

Final E-R Diagram


Hospital Management System
Entity-relationship diagram
Administrative Tasks: PK Task ID, Description, Priority, FK AssignedStaff, DueDate, Status
Users: PK UserID, FK UserTypeID, UserName, Password
UserType: PK UserTypeID, Role
Staff: PK Staff ID, FK UserID, Name, PositionRole, FK DepartmentID, ShiftSchedule, EmploymentStatus, SalaryInformation, TrainingHistory, LeaveRecords, PerformanceReviews, ContactInformation
Department: PK DepartmentID, DepartmentName, FK DepartmentHeadID, Location
SurgeryType: PK SurgeryID, SurgeryProcedure, AnesthesiaType, SurgicalTeam, PreoperativeInstriuctions
SurgeriesInfo: PK SurgeryID, FK SurgeryTypeID, FK PatientID, SurgeryDateTime
ItemsServices: PK ServiceID, ItemsServicesRendered, Amount
SurgerySurgeon: CK SurgeryID, CK,FK SurgeonID
Appointment: PK AppointmentID, FK PatientID, FK DoctorID, Date and Time, Appointment type, Status, FK InitialAppointmentID, AppointmentNote
BillingInfo: PK BillID, FK PatientID, BillingDate, PaymentStatus, PaymentMethod
Billing: CK BillID, CK , FK ServiceID
LabTechnician: PK,FK LabTechnicianID, LabTechnicianSpecialization, LabTechnicianCerttification
Doctor: PK,FK DoctorID, Specialization, LicenseInfo, AppointmentSchedule
Insurance: PK InsuranceID, Insurance Info, Insurance Company, Policy Number, Coverage Type
Medical Record: PK RecordID, UK PatientID, UK DoctorID, UK DateOfVisit, UK DiagnosisID, TreatmentPlan, UK LabTestOrderID
Nurse: PK,FK NurseID, NurseSpecialization, NurseCertification
LabTest: PK TestID, UK PatientID, FK DoctorID, UK TestTypeID, UK Date Time, Test Resullts, FK LabTechnicianID, TestStatus
DiagnosisInfo: PK DiagnosisID, DiagnosisName
DiagnosisMed: CK DiagnosisID, CK,FK MedicationID
Patient: PK,FK PatientID, Name, DateOfBirth, FK InsuranceID, ContactInfo, Address, Allergies, FK BloodTypeID, Medical History, EmergencyContact, RegistrationDate
Inventory: PK ItemId, ItemName, Quantitiy, SupplierInfo, ExpiryDate, Location, StockLevel
Pharmacy: PK MedicationID, MedicationName, Dosage, PrescriptionInfo, DispensingPharmacist, RefillInfo
TestType: PK TestID, TestType
FacilityManagement: PK FacilityID, MaintenanceRecords, EquipmentInventory, SecurityLogs, CleaningSchedules, UtilitiesUsage
Wards: PK WardID, WardName, Capacity, CurrentOccupation, FK NurseInCharge
Addmissions: FK PatientID, PK AddmissionID, FK AdmittingDoctor, DischargeDate, FK WardAssigned, AdmissionDate, ReasonForAdmission
BloodType: PK BloodTypeID, BloodType
PatientCare: CK,FK AdmissionID, CK,FK CareTypeID, FK CareProviderID, NursingNotes, VitalSigns, MedicationAdministrationRecords, ProgressReports, RecordedAt
CareType: PK CareTypeID, CareType
{optional,or}
{optional}
has
manages
in charge
admits
Database shapes
Existing table
Proposed table
Modified (Recently Added) table
PK: Primary Key
CK: Composite Key
FK: Foreign Key
UK: Unique Key

7.USER INTERFACE FUNCTIONALITY REQUESTS: SQL Query Integration

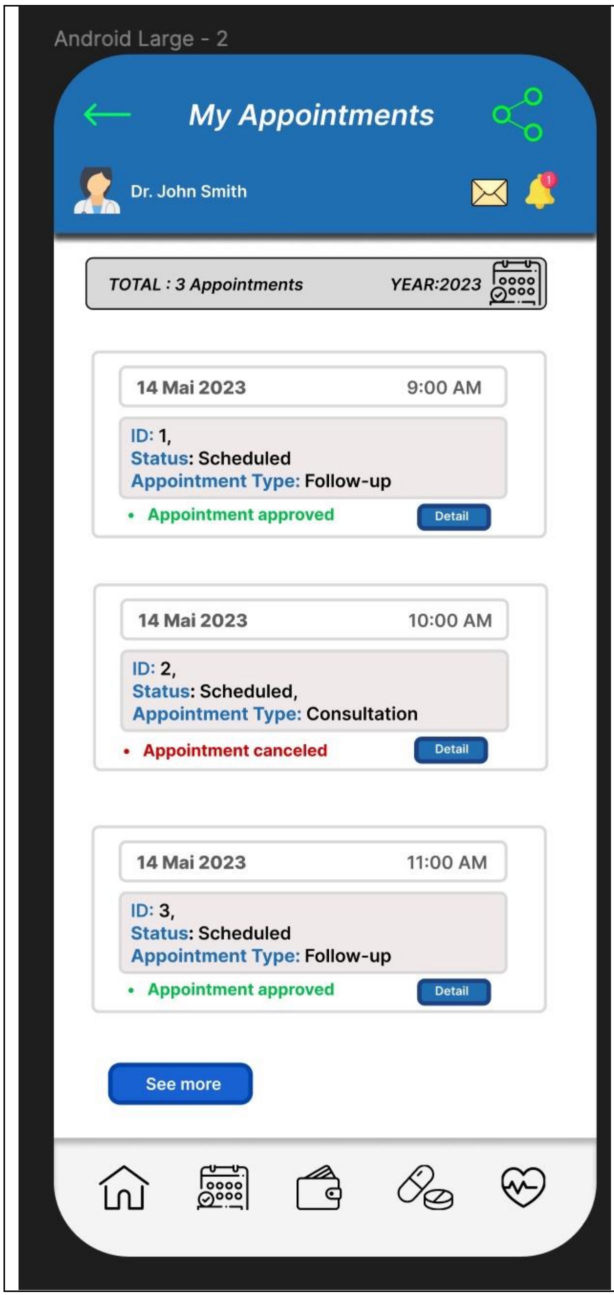
1. A list of all doctors that have seen a certain patient



```
SELECT DISTINCT Doctors.* FROM Doctors
INNER JOIN MedicalRecords ON
Doctors.DoctorID = MedicalRecords.DoctorID
WHERE MedicalRecords.PatientID
=[patientID];
```

- This query generates a list of all distinct doctors who have treated or seen a particular patient, identified by [patientID]. The INNER JOIN ensures that only doctors who have corresponding medical records for the specified patient are included in the result.
- **Patients:** Patients may have access to see which doctors have treated them to maintain transparency and for personal record-keeping.
- **Doctors:** Doctors who are treating the patient might need to know which other doctors have seen the patient for better coordination of care.
- **Authorized Medical Staff:** Nurses or other authorized staff who assist in the management of patient care.

2. A list of all patients and the number of appointments they have had in the past year

	See query below

Android Large - 2

My Appointments

Dr. John Smith

TOTAL : 3 Appointments YEAR:2023

14 Mai 2023 9:00 AM
ID: 1,
Status: Scheduled
Appointment Type: Follow-up
• Appointment approved

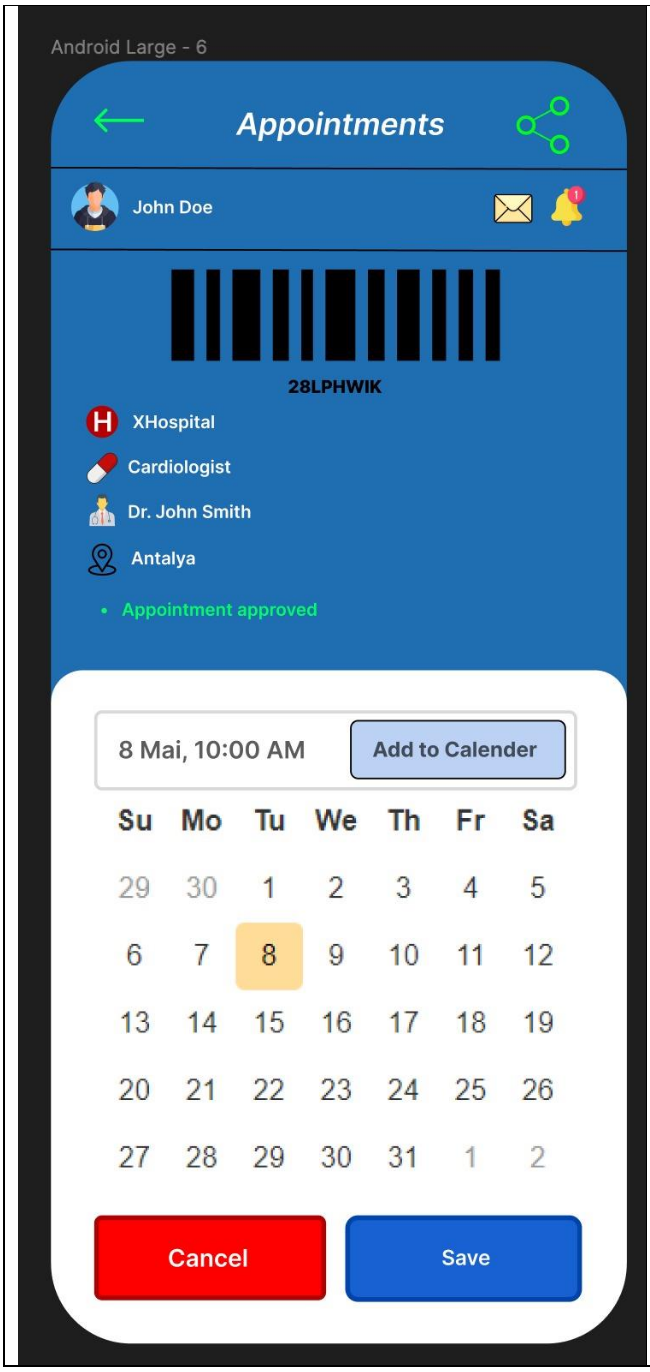
14 Mai 2023 10:00 AM
ID: 2,
Status: Scheduled,
Appointment Type: Consultation
• Appointment canceled

14 Mai 2023 11:00 AM
ID: 3,
Status: Scheduled
Appointment Type: Follow-up
• Appointment approved

```sql
SELECT Patients.PatientID,
Patients.Name,
COUNT(Appointments.AppointmentID)
AS CountAppointments FROM Patients
LEFT JOIN Appointments
ON Patients.PatientID =
Appointments.PatientID
WHERE Appointments.DateAndTime >=
DATE_SUB(CURRENT_DATE(),
INTERVAL 1 YEAR)
GROUP BY Patients.PatientID,
Patients.Name;
```

- This query generates a list of all patients along with the number of appointments they have had in the past year. The LEFT JOIN ensures that even patients without any appointments in the past year are included in the result with a count of zero.
- **Patient Registrators:** They handle scheduling and may need to provide information about appointment frequency.
- **Doctors:** Doctors might need to review the appointment history to understand patient adherence and follow-up needs.
- **Patients:** Patients may have access to their own appointment history to stay informed about their healthcare activities and follow-up schedules.

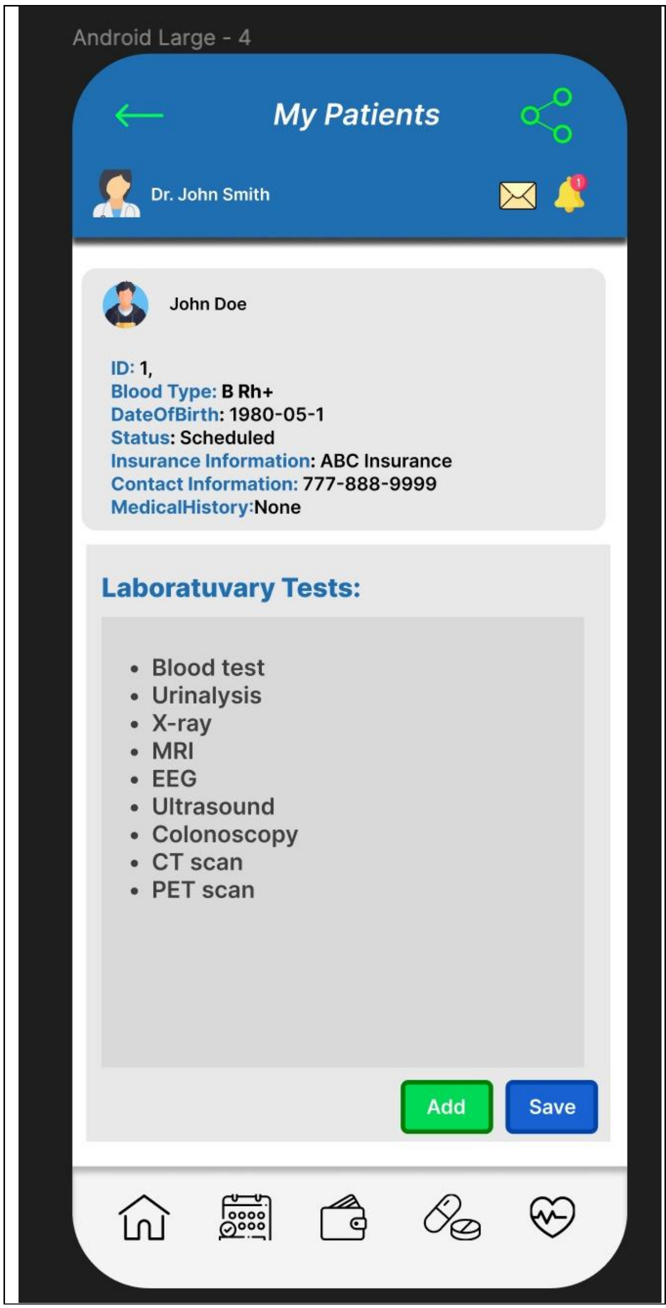
3. A way to change the date of an appointment



```
UPDATE Appointments
SET DateAndTime = '[DateAndTime]'
WHERE AppointmentID = [appointmentID];
```

- This method allows users to easily update their existing appointments. The calendar interface enables users to select a suitable date and time, and then save the changes. Subsequently, the SQL query updates the appointment date in the database.
- **Patients:** Patients may be allowed to reschedule their own appointments to ensure flexibility and convenience in managing their healthcare.
- **Patient Registrators:** They handle appointment scheduling and rescheduling.
- **Doctors:** In our system, doctors might have the authority to change appointments for their patients. This decision was made based on discussions and identified needs for doctors to have this capability.

4. A way to add a test for a patient


Android Large - 4

My Patients

Dr. John Smith

John Doe

ID: 1,
Blood Type: B Rh+
DateOfBirth: 1980-05-1
Status: Scheduled
Insurance Information: ABC Insurance
Contact Information: 777-888-9999
MedicalHistory:None

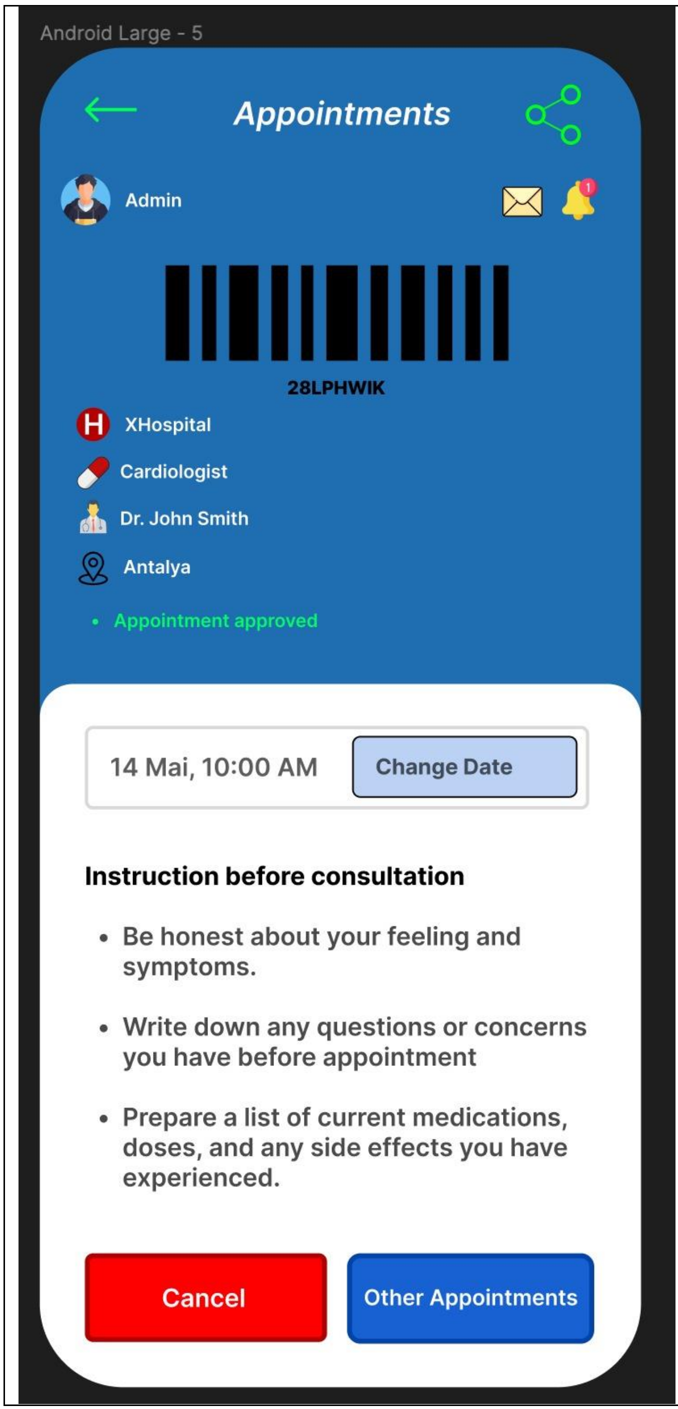
Laboratuvary Tests:

- Blood test
- Urinalysis
- X-ray
- MRI
- EEG
- Ultrasound
- Colonoscopy
- CT scan
- PET scan

```
INSERT INTO LabTests (PatientID,
DoctorID,  DateTime, TestResults,
LabTechnician, TestStatus,
TestTypeID)
VALUES ('[PatientID]', '[DoctorID]',
'[DateTime]', '[TestResults]',
'[LabTechnician]', '[TestStatus]',
'[TestTypeID]');
```

- This query allows the insertion of a new lab test record into the LabTests table for a specified patient. By providing the relevant details such as patient ID, doctor ID, date and time, test results, lab technician, test status, and test type ID, a complete record of the lab test is added to the database.
- **Doctors:** They request lab tests for their patients.
- **Medical Administrators:** They may need to enter test requests into the system.
- **Lab Technicians:** They might enter details of tests conducted or manage the lab test records.

5. A way to remove an appointment from the system



XHospital
Cardiologist
Dr. John Smith
Antalya
• Appointment approved

14 Mai, 10:00 AM

Instruction before consultation

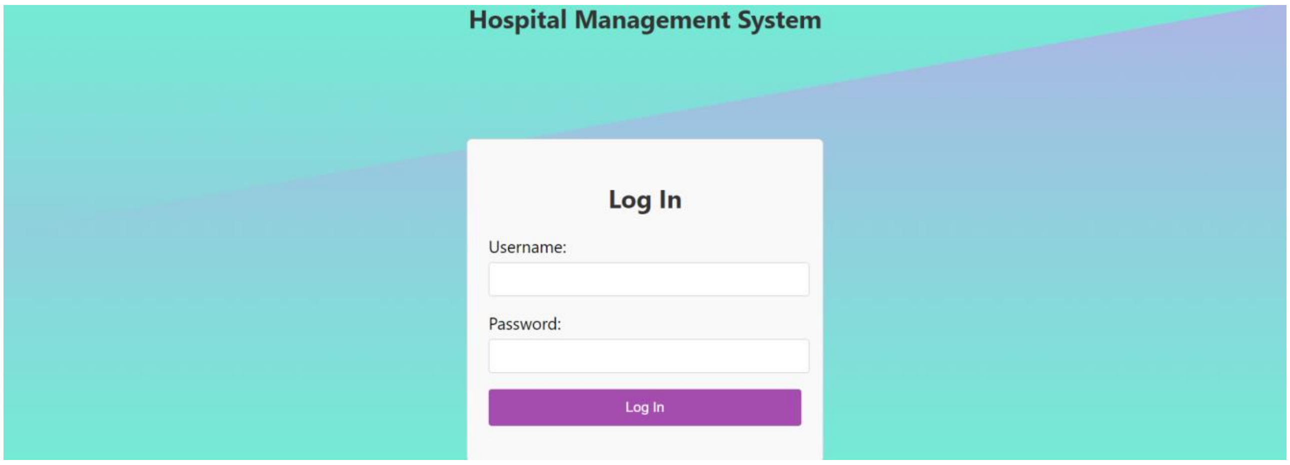
- Be honest about your feeling and symptoms.
- Write down any questions or concerns you have before appointment
- Prepare a list of current medications, doses, and any side effects you have experienced.

```
DELETE FROM Appointments
WHERE AppointmentID =
[appointmentID];
```

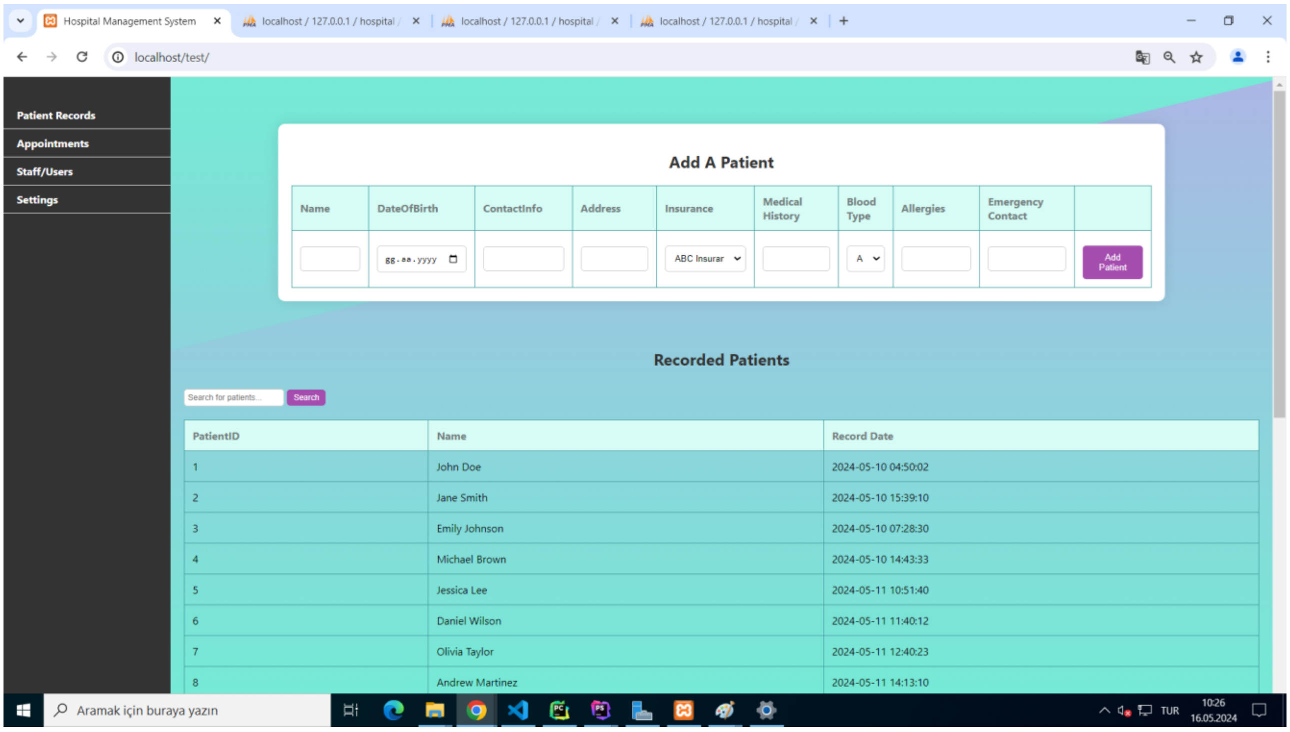
- This query removes an appointment from the Appointments table by identifying it through its unique AppointmentID. By specifying the AppointmentID, the query ensures that only the intended appointment is deleted from the system.
- **Patient Registrators:** They handle patient scheduling and cancellations.
- **Doctors:** They may need to cancel appointments if they are no longer necessary or need to reschedule.
- **Patients:** Patients should have the ability to cancel their own appointments for flexibility and convenience.

Working Prototype

LOGIN PAGE:



ADMINISTRATION PANEL:



Add A Patient

Name	DateOfBirth	ContactInfo	Address	Insurance	Medical History	Blood Type	Allergies	Emergency Contact	
	gg.aa.yyyy			ABC Insurar		A			Add Patient

Recorded Patients

PatientID	Name	Record Date
1	John Doe	2024-05-10 04:50:02
2	Jane Smith	2024-05-10 15:39:10
3	Emily Johnson	2024-05-10 07:28:30
4	Michael Brown	2024-05-10 14:43:33
5	Jessica Lee	2024-05-11 10:51:40
6	Daniel Wilson	2024-05-11 11:40:12
7	Olivia Taylor	2024-05-11 12:40:23
8	Andrew Martinez	2024-05-11 14:13:10

PRESENTATION LINK:

https://www.canva.com/design/DAGFCiKFXYM/cJeP7XJy64ZoqUneX4K3Ow/view?utm_content=DAGFCiKFXYM&utm_campaign=designshare&utm_medium=link&utm_source=editor