Hospital Management System

SJSU
Girma Jembere, Arlan Prado, Jade Webb
Team 5
CS157A Section 80

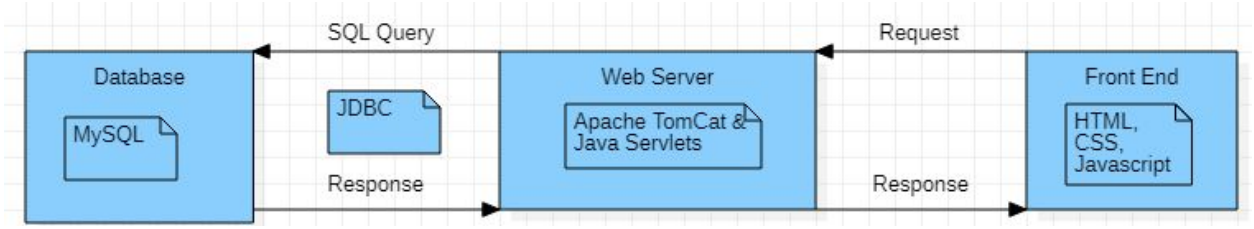
Project Overview

Project Description

- This project will be a database application that is based in the hospital domain. The goal of this hospital database application is to provide a convenient way to record, access, and manipulate data in the hospital system. Hospital stakeholders will benefit from this application, as they will be able to easily organize their employee and patient data to increase efficiency in management and overall success. This application will also be designed so that only those who are authorized to access certain information (e.g. the type of drug that is given to patients) are able to do so. Therefore, there will be personalized views of the database for individuals of different clearance levels. The motivation behind this application is to provide a user-friendly interface that is comprehensive, clear, and intuitive in order to support top-notch employee performance and facilitate patient satisfaction and ultimate recovery. In a time of crisis due to the COVID-19 pandemic, it is vital that quality hospital services remain available for the severely ill and the general public. This database application will enable hospital services to operate smoothly while accommodating for COVID-19 patient care and sanitization measures. Remote access will be enabled, since paperwork will be minimized through the use of this electronic database. All information will be stored within the database, which will minimize the contact between employees and patients. Paperwork such as doctors' notes, laboratory results, and medical data are all digitized, allowing them to be accessed anywhere by users for convenience and safety.

System Environment

Structure of the System



Hardware and Software used

- Apache Tomcat

RDBMS

- MySQL Community Server 8.0.20

Application Languages

- Java 8, Java EE 7, HTML, CSS, Javascript, XML, SQL

Revised Functional Requirements:

System

The system provides functionality for multiple types and subtypes of users, including employees (e.g. doctors, nurses, specialists, surgeons, and supervisors) as well as patients. Users must have a login to access the dashboards, to have a login they must sign up. Signing up defaults them to a patient view until they are hired as an employee by another employee. Patients have a dashboard that will differ from the employee dashboard in terms of functionality, but will be structured similarly. The structure of the dashboard is general and immediate information needed on the page. Inserting information can be done through tab buttons leading to a different page or dashboard buttons to a pop up window. Patients will also have less control on the dashboard than employees. Patients are only allowed to view information that pertain to them and modify information that is easily susceptible to change. Employees are able to have the most control on their dashboard, allowing them to change most information.

Functions and Features

Patient Functions

The two different users have different functionality allowed to both of them. On the patient dashboard they can view their medical history, their diagnosis, the next immediate appointment, the doctors notes made on them, and the healthcare development team in charge of them. Their tab buttons include viewing their prescriptions, requesting appointments, and viewing or editing their profile info.

Functional Requirements

- View Medical History
    - The patient's personal medical history is viewed on the dashboard. Fields of the included medical history: medication taken before, allergies, diseases, symptoms, and family history
- View Current Diagnosis
    - Patients will see their diagnosis, current condition, and their admission date
- View Next Appointment

    - Patients have appointments with certain employees. Their nearest appointment that is scheduled the current day or in the future will be shown on the patient dashboard
    - Fields shown will be the date, the time, and the employee conducting the appointment
- View Doctors Note
    - Through a collapsable message, the details of the note and the date are visible to the patient
- View Healthcare Development Team
    - These are the employees assigned to assist in the patient's full recovery
    - A list of employees are shown including the ID, position, and name
- View All Prescriptions
    - Patients are able to view a table of their prescriptions that includes the name of the medication, total amount given, the dosage, and the frequency of the dosage
- Request Appointment
    - Patients can request appointments with a specific employee at a specified time and date. This will be logged in the request appointment table.
- View/Edit Profile
    - All basic information is viewable to the patient: Name, ID, Birth Date, Gender, Phone Number, Address, and Email
    - Patients can edit their Phone Number, Address, Email, and Password

Employee Functions

On the employee dashboard basic info is shown and their next upcoming appointment (that they conduct). Through a pop up window they could look at their patient list, write a doctor's note, and hire an existing user as an employee. Through the tabs they can view all their upcoming appointments, order prescriptions for patients, create appointments, search for other patients, and view or edit their profile.

Functional Requirements

- View Next Upcoming Appointments
    - Employees can view the next appointment they need to conduct
    - This shows the patient name, the room number, the start time, end time and the date of the appointment
    - This is only shows appointments that are in the near future, no past appointments

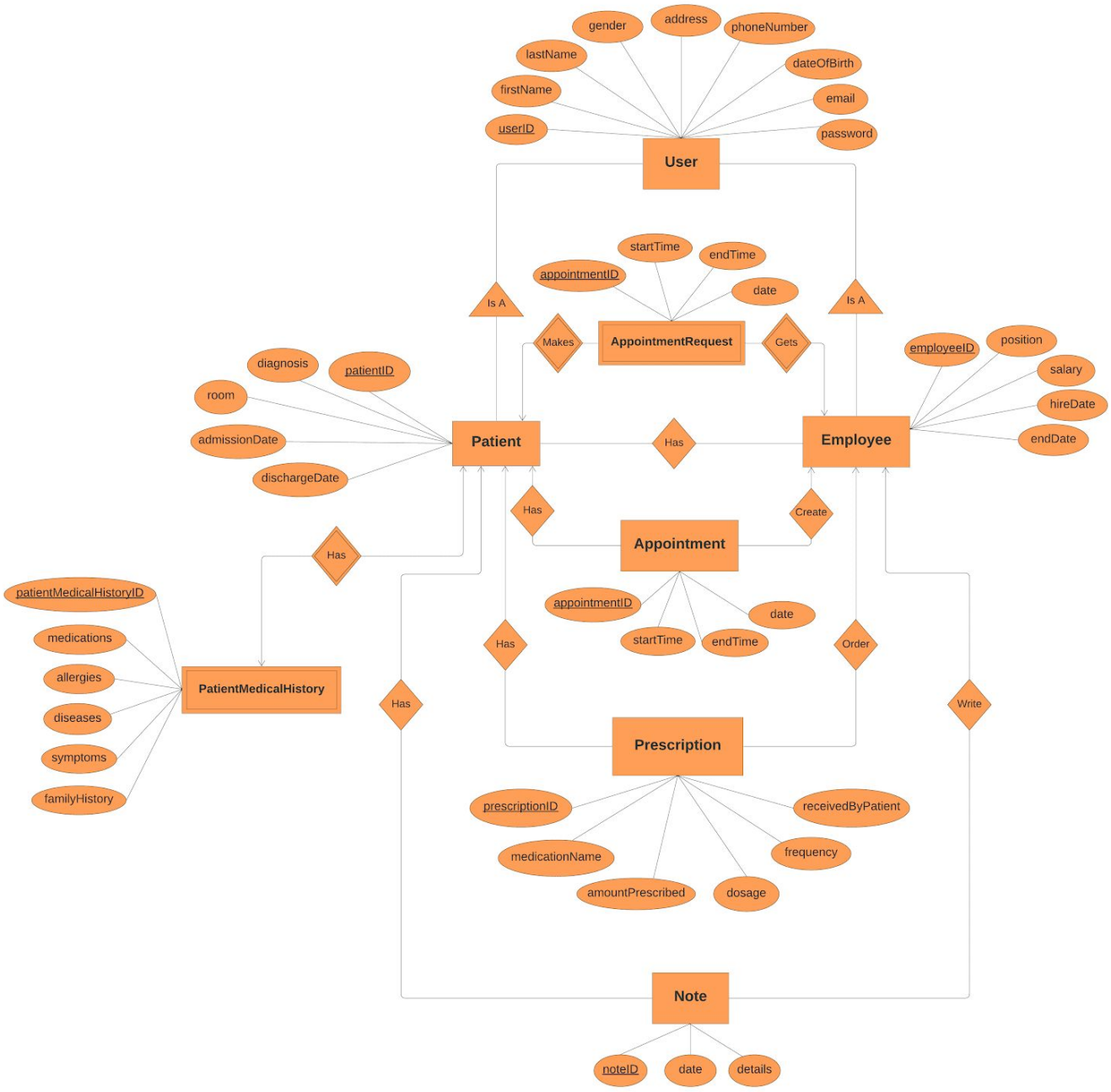
- Write Note

    - Submits a doctor's note about a particular patient identified by ID, along with details of the note, and date written
- View list of Current Patients
    - This displays the list of patients that are currently under the employee's care
    - The patients are identified with ID and name
- Hire Employee
    - Registers an already created user account as an employee
    - To hire an employee, it requires knowledge of their User ID, salary, position, and the hire date
- View All Upcoming Appointments
    - Employees can view all appointments scheduled in the future instead of the next one like on the dashboard
    - In this separate page lists the appointment ID, room number, start time, end time, date, patient ID, and patient name
- Order Prescriptions
    - Employees can fill out forms to order a prescription for a patient
    - This requires the patient ID, patient Name, the medication, total amount, dosage, and frequency
    - Confirmation is also required, which includes entering their own ID and name
- Create Appointment
    - Employees can set appointments with patients, requiring the patient ID, start time, end time, and date
    - They can see any requested appointments
        - To confirm the appointment, they will create the appointment using the same form as before. If the fields of appointment form correlate to the requested appointment, it will be removed and re-entered in the appointment table

- Patient Search
    - Patients can be searched for using their ID, first name, or their last name
        - Each search will return the full name and ID of the potential patient
    - Viewing patient info is also another function included here, but can only be done if the patient is under the care of the employee doing the search
- View/Edit Profile
    - Employee info displayed: ID, Name, Birth Date, Gender, Phone Number, Address, Email, Position, Salary, and Hire Date
    - Information available for edit are the Address, Phone Number and Email

Project Design:

E/R Diagram:



Descriptions:

User

- User represents a user of the system. Each user has a userID, firstName, lastName, address, phoneNumber, password, email, and dateOfBirth. The userID is used to identify each user as the primary key.
    - The ID of the user will be automatically generated.

Employee

- Employee is a subclass of the user entity. It represents a hospital employee with a userID referred to as employeeID, a salary, a position, a hireDate, and an endDate. The employeeID is used to identify each employee as the primary key.
    - The ID of the employee will be the automatically generated user ID.

Patient

- Patient is a subclass of the user entity. It represents a hospital patient with a userID referred to as patientID, a condition, a diagnosis, a room, an admissionDate, and a dischargeDate. The patientID is used to identify each patient as the primary key.
    - The ID of the patient will be the automatically generated user ID.

EmployeeHasPatients

- Each employee has a number of patients, and each patient has a number of employees that tend to them. Many employees can attend one patient, and many patients can be attended by many employees (nurses, doctors, specialists, etc.).
    - This table will keep track of which employee tends to which patient.

AppointmentRequest

- This tracks all the appointment requests made by patients to employees. It has an ID, a start time, end time, and a date. When the appointment is confirmed by the employee, the ID does not carry over into the appointment table. A new appointment is created with the same start time, end time, and date. The tuple will then be deleted once the appointment has been made.

Appointment

- Appointment is an entity that represents an appointment between a patient and an employee. It has an appointmentID, the start time, the approximate end time of the appointment, and the date of the appointment. The appointmentID is used to identify each appointment as the primary key.
    - The relationship with patients and appointments, and employees and appointments will be in their own separate relation tables
    - The ID of the appointment will be automatically generated.

PatientHasAppointment

- This correlates the patient and appointment tables to keep track of which patient is attending each appointment. Each appointment has only one patient.
    - Patient ID and Appointment ID will be tracked in this table alone.

EmployeeCreatesAppointment

- Employees can create appointments and this table tracks which employee has created and will conduct the appointment. Each appointment is created by only one employee.
    - Employee ID and Appointment ID will be tracked in this table alone.

PatientMedicalHistory

- PatientMedicalHistory is an entity that represents the records of a patient's medical history. This is entered when a patient is first admitted to the hospital. It has a patientMedicalHistoryID, the medications previously or currently taken, any allergies that the patient has, any diseases that they have had, the patient's symptoms, and any family history of illnesses or conditions. The patientMedicalHistoryID is used to identify each medical history document as the primary key.
    - The ID of the patient medical history will be automatically generated.

Prescription

- Prescription is an entity that represents a prescription that has been ordered by an employee for a certain patient. It has a prescriptionID, the medicationName, the total amount prescribed, the dosage, the frequency of which the medicine should be taken, and a boolean indicating validation of whether it has been received by the patient. The prescriptionID is used to identify each prescription as the primary key.
    - Employees order prescriptions for the patient, while the patient is able to view their prescription
    - The ID of the prescription will be automatically generated.

PatientHasPrescription

- Patient and prescription ID will be associated within this table. Each patient can have many prescriptions and a prescription belongs to only one patient.
    - Patient ID and Prescription ID will be tracked in this table alone.

EmployeeOrdersPrescription

- Employee and prescription ID will be associated within this table. Each employee can order many prescriptions and a prescription is ordered by only one employee.
    - Employee ID and Prescription ID will be tracked in this table alone.

Note

- Note is an entity set that represents a medical employee's note for a particular patient. They have an automatically generated noteID, the date it was written, and the details of the notes. The primary key is the noteID.
    - The ID of the note will be automatically generated.

PatientHasNote

- The set represents the relationship between the patient the note was for and the note. One patient can have many notes, but only one note can be associated with each patient.
    - Patient ID and Note ID will be tracked in this table alone.

EmployeeWritesNote

- This set represents the relationship between the employee who has written the note and the note itself. One employee can make many notes but one note can only be associated with one employee.
    - Employee ID and Note ID will be tracked in this table alone.

Schemas:

Database Name: “Hospital”

Database: Entity Sets

1. User (userID, firstName, lastName, dateofBirth, gender, phoneNumber, address, email, password, CREATED_DATE)
2. Employee (employeeID, salary, position, hireDate, endDate, CREATED_DATE)
3. Patient (patientID, diagnosis, room, admissionDate, dischargeDate, CREATED_DATE)
4. Appointment (appointmentID, start_time, end_time, date, CREATED_DATE)
5. PatientMedicalHistory (patientMedicalHistoryID, patientID, medications, allergies, diseases, symptoms, familyHistory, CREATED_DATE)
6. Prescription (prescriptionID, medicationName, totalAmount, receivedByPatient, prescribedAmount, frequency, CREATED_DATE)
7. Note (noteID, detail, date, CREATED_DATE)
8. AppointmentRequest(appointmentID, start_time, end_time, date, CREATED_DATE)

Database: Relationships

1. EmployeeHasPatients (patientID, employeeID, CREATED_DATE)
2. EmployeeCreateAppointment (appointmentID, employeeID, CREATED_DATE)
3. PatientHasAppointment (appointmentID, patientID, CREATED_DATE)
4. EmployeeOrdersPrescription (prescriptionID, employeeID, CREATED_DATE)
5. PatientHasPrescription (prescriptionID, patientID, CREATED_DATE)
6. EmployeeWritesNote (noteID, employeeID, CREATED_DATE)
7. PatientHasNote (noteID, patientID, CREATED_DATE)

Screenshots of Tables:

User Table

```
1 •  DESCRIBE hospital.user;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
userID	int unsigned	NO	PRI	NULL	auto_increment
firstName	varchar(45)	NO		NULL	
lastName	varchar(45)	NO		NULL	
dateOfBirth	varchar(45)	NO		NULL	
gender	varchar(45)	NO		NULL	
phoneNumber	varchar(45)	YES		NULL	
address	varchar(45)	YES		NULL	
email	varchar(45)	NO	UNI	NULL	
password	varchar(255)	NO		NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 •  SELECT * FROM hospital.user;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

userID	firstName	lastName	dateOfBirth	gender	phoneNumber	address	email	password	CREATED_DATE
16	Roman	Ballard	1970-09-12	m	133434830	801 Pretlow St, Franklin, VA, 23851	airship@grufefse.cf	291029	2020-08-04 00:00:00
17	Marisela	Gadison	1966-05-13	f	143434830	63 Twin Peaks Ln, Sequim, WA, 98382	crimsane@grufefse.cf	234819	2020-08-04 00:00:00
18	Cliff	Woullard	1989-06-20	m	153434830	6908 Longview Beach Rd, Jeffersonville	chrisk@grufefse.cf	112271	2020-08-04 00:00:00
19	Domingo	Zobel	1978-11-02	m	163434830	8863 Black Point Rd, Syracuse, IN, 46567	aribaud@grufefse.cf	189961	2020-08-04 00:00:00
20	Andria	Straley	1978-04-19	f	153174830	10318 Kk Rd, Rapid River, MI, 49878	iamcal@grufefse.cf	290731	2020-08-04 00:00:00
21	Santina	Linker	1994-03-17	f	163184830	87 Gore Rd, Lancaster, NH, 03584	msloan@grufefse.cf	456878	2020-08-04 00:00:00
22	Lucretia	Grissom	1965-07-06	f	173194830	150 Barren, Conshohocken, PA, 19428	kodeman@grufefse.cf	255553	2020-08-04 00:00:00
23	Tianna	Nielson	1955-08-23	f	183420830	30 Northington Dr, Avon, CT, 06001	delpino@grufefse.cf	987234	2020-08-04 00:00:00
24	Angelia	Sisk	1998-03-16	f	193214830	472 870th Hwy, Winnsboro, LA, 71295	arnold@grufefse.cf	87431	2020-08-04 00:00:00
25	Sheilah	Rueb	1979-09-26	f	203224830	625 E Plum St, Vineland, NJ, 08360	portele@grufefse.cf	918394	2020-08-04 00:00:00
26	Mafalda	Call	1984-01-02	f	213234830	234 E Black Point Rd, Syracuse, IN, 46567	presoff@grufefse.cf	987888	2020-08-04 00:00:00
27	Lilia	Thorsen	1995-08-12	f	222434830	23 Gore st, Lancaster, NH, 03584	vsprintf@grufefse.cf	456845	2020-08-04 00:00:00
28	Mafalda	Thorsen	1998-05-07	f	92434830	77 Gore st, hamilton, NH, 03584	Mafalda@grufefse.cf	684322	2020-08-04 00:00:00
29	Lilia	Sheilah	1973-04-11	f	222434830	567 Gore st, Lancaster, NH, 03584	Lilia45@grufefse.cf	888444	2020-08-04 00:00:00
30	Angelia	Thorsen	1975-09-30	f	792434830	568 Gore st, Lancaster, NH, 03584	Angelia56@grufefse.cf	832563	2020-08-04 00:00:00
31	Bailey	Rome	1970-09-18	m	133434830	801 Pretlow St, Franklin, VA, 23851	rrirship@grufefse.cf	928838	2020-08-04 00:00:00
32	Gary	Marisela	1990-06-02	m	143434830	63 Twin Peaks Ln, Sequim, WA, 98382	hgimsane@grufefse.cf	262262	2020-08-04 00:00:00
33	William	Cliff	1969-03-15	m	153434830	6908 Longview Beach Rd, Jeffersonville	jfrisk@grufefse.cf	99763	2020-08-04 00:00:00
34	Zobel	Domingo	1986-01-12	m	163434830	8863 Black Point Rd, Syracuse, IN, 46567	mjibaud@grufefse.cf	853124	2020-08-04 00:00:00
35	Stan	Andria	1987-10-19	m	153174830	10318 Kk Rd, Rapid River, MI, 49878	rsmcal@grufefse.cf	953245	2020-08-04 00:00:00
36	Lincoln	Santina	1976-08-02	m	163184830	87 Gore Rd, Lancaster, NH, 03584	wfloan@grufefse.cf	221346	2020-08-04 00:00:00
37	Gus	Lucret	1974-05-22	m	173194830	150 Barren, Conshohocken, PA, 19428	kudeman@grufefse.cf	920787	2020-08-04 00:00:00
38	Niel	Tian	1979-05-27	m	183420830	30 Northington Dr, Avon, CT, 06001	whlpino@grufefse.cf	909753	2020-08-04 00:00:00
39	Sal	Angel	1970-11-12	m	193214830	472 870th Hwy, Winnsboro, LA, 71295	ulnold@grufefse.cf	567789	2020-08-04 00:00:00
40	Robert	Sheilah	1979-05-13	m	203224830	625 E Plum St, Vineland, NJ, 08360	qdrtele@grufefse.cf	927678	2020-08-04 00:00:00
41	Calum	Mafald	1987-04-07	m	213234830	234 E Black Point Rd, Syracuse, IN, 46567	whgsoff@grufefse.cf	436301	2020-08-04 00:00:00
42	Thora	Lillard	1989-07-09	f	222434830	23 Gore st, Lancaster, NH, 03584	tvprintf@grufefse.cf	918648	2020-08-04 00:00:00

Employee Table

```
1 •  DESCRIBE hospital.employee;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
employeeID	int unsigned	NO	MUL	NULL	auto_increment
salary	int	YES		NULL	
position	varchar(45)	YES		NULL	
hireDate	date	NO		NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
endDate	date	YES		NULL	

```
1 •  SELECT * FROM hospital.employee;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employeeID	salary	position	hireDate	endDate	CREATED_DATE
31	120000	Doctor	2019-09-12	NULL	2020-08-04 00:00:00
32	100000	Nurse	2018-09-12	NULL	2020-08-04 00:00:00
33	150000	Surgeon	2018-09-12	NULL	2020-08-04 00:00:00
34	150000	Specialist	2017-09-12	NULL	2020-08-04 00:00:00
35	100000	Nurse	2010-09-12	NULL	2020-08-04 00:00:00
36	160000	Surgeon	2019-09-12	NULL	2020-08-04 00:00:00
37	140000	Doctor	2018-09-01	NULL	2020-08-04 00:00:00
38	150000	Pharmacist	2017-09-12	NULL	2020-08-04 00:00:00
39	130000	Doctor	2019-09-12	NULL	2020-08-04 00:00:00
40	120000	Surgeon	2018-09-12	NULL	2020-08-04 00:00:00
41	135000	Pharmacist	2019-09-12	NULL	2020-08-04 00:00:00
42	160000	Doctor	2016-09-12	NULL	2020-08-04 00:00:00
43	136000	Nurse	2018-09-12	NULL	2020-08-04 00:00:00
44	110000	Surgeon	2016-09-12	NULL	2020-08-04 00:00:00
45	140000	Specialist	2015-09-12	NULL	2020-08-04 00:00:00

Patient Table

```
1 •  DESCRIBE hospital.patient;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
patientID	int unsigned	NO	MUL	NULL	auto_increment
diagnosis	varchar(45)	YES		NULL	
room	int unsigned	NO		NULL	
admissionDate	date	NO		NULL	
dischargeDate	date	YES		NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 •  SELECT * FROM hospital.patient;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patientID	diagnosis	room	patient_condition	admissionDate	dischargeDate	CREATED_DATE
16	Latex Allergy	101	undetermined	2020-06-12	2020-06-18	2020-08-04 00:00:00
17	Cold & Flu	104	undetermined	2020-03-03	2020-03-14	2020-08-04 00:00:00
18	High Cholesterol	105	undetermined	2020-07-13	2020-07-16	2020-08-04 00:00:00
19	HIV/AIDS	106	undetermined	2020-01-15	2020-01-16	2020-08-04 00:00:00
20	COVID-19	107	undetermined	2020-01-11	2020-04-11	2020-08-04 00:00:00
21	Lung Conditions	108	undetermined	2020-05-10	2020-05-23	2020-08-04 00:00:00
22	Cancer	109	undetermined	2020-02-01	2020-02-14	2020-08-04 00:00:00
23	COVID-19	110	undetermined	2020-03-01	2020-03-17	2020-08-04 00:00:00
24	Measles	111	undetermined	2020-05-01	2020-05-19	2020-08-04 00:00:00
25	Lung cancer	112	undetermined	2020-06-01	2020-06-20	2020-08-04 00:00:00
26	COVID-19	113	undetermined	2020-04-01	2020-04-24	2020-08-04 00:00:00
27	Depression	114	undetermined	2019-12-01	2019-12-26	2020-08-04 00:00:00
28	Bipolar disorder	115	undetermined	2020-07-01	2020-07-18	2020-08-04 00:00:00
29	Glaucoma	117	undetermined	2020-01-02	2020-01-20	2020-08-04 00:00:00
30	Epilepsy	119	undetermined	2020-02-13	2020-02-23	2020-08-04 00:00:00

Appointment Table

```
1 • DESCRIBE hospital.appointment;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
appointmentID	int unsigned	NO	PRI	NULL	auto_increment
start_time	time	NO		NULL	
end_time	time	NO		NULL	
date	date	NO		NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 • SELECT * FROM hospital.appointment;
```

Result Grid | Filter Rows: | Edit: | Export

appointmentID	start_time	end_time	date	CREATED_DATE
16	2:00	2:30	2020-08-15	2020-08-04 00:00:00
17	3:00	3:30	2020-08-21	2020-08-04 00:00:00
18	4:00	4:30	2020-08-14	2020-08-04 00:00:00
19	4:30	5:00	2020-09-18	2020-08-04 00:00:00
20	1:00	1:30	2020-09-21	2020-08-04 00:00:00
21	1:15	1:45	2020-09-05	2020-08-04 00:00:00
22	1:40	2:00	2020-10-07	2020-08-04 00:00:00
23	10:00	10:30	2020-10-08	2020-08-04 00:00:00
24	8:45	9:20	2020-10-17	2020-08-04 00:00:00
25	11:00	11:45	2020-11-16	2020-08-04 00:00:00
26	10:00	10:30	2020-11-12	2020-08-04 00:00:00
27	11:00	11:30	2020-11-12	2020-08-04 00:00:00
28	11:30	12:00	2020-11-12	2020-08-04 00:00:00
29	12:00	12:30	2020-11-12	2020-08-04 00:00:00
30	12:30	1:00	2020-11-12	2020-08-04 00:00:00
NULL	NULL	NULL	NULL	NULL

PatientMedicalHistory

```
1 •  DESCRIBE hospital.patientmedicalhistory;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
patientMedicalHistoryID	int unsigned	NO	PRI	NULL	auto_increment
patientID	int unsigned	NO	MUL	NULL	
medications	varchar(100)	YES		NULL	
allergies	varchar(100)	YES		NULL	
diseases	varchar(100)	YES		NULL	
symptoms	varchar(100)	YES		NULL	
familyHistory	varchar(100)	YES		NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 •  SELECT * FROM hospital.patientmedicalhistory;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

patientMedicalHistoryID	patientID	medications	allergies	diseases	symptoms	familyHistory	CREATED_DATE
16	16	Clindamycin	Latex Allergy	Asthma	breathlessness, chest tightness	none	2020-07-20 00:00:00
17	17	Ibuprofen	Food Allergy	Asthma	breathlessness, chest tightness	none	2020-07-20 00:00:00
18	18	Adderall	Mold Allergy	Asthma	breathlessness, chest tightness	none	2020-07-20 00:00:00
19	19	Amlodipine	Pet Allergy	none	breathlessness, Muscle aches	none	2020-07-20 00:00:00
20	20	Omeprazole	Pollen Allergy	Asthma	Fever	none	2020-07-20 00:00:00
21	21	Amoxicillin	Food Allergy	none	breathlessness, chest tightness	none	2020-07-20 00:00:00
22	22	Loratadine	Pollen Allergy	hemochromatosis	breathlessness, chest tightness	none	2020-07-20 00:00:00
23	23	Trazodone	Food Allergy	Asthma	Coughing, chest tightness	none	2020-07-20 00:00:00
24	24	Ozempic	Pet Allergy	Asthma	breathlessness, chest tightness	none	2020-07-20 00:00:00
25	25	Benzonatate	Drug Allergy	none	breathlessness, Muscle aches	none	2020-07-20 00:00:00
26	26	Kevzara	Insect Allergy	Asthma	breathlessness, chest tightness	none	2020-07-20 00:00:00
27	27	Lisinopril	Drug Allergy	none	Coughing, chest tightness	none	2020-07-20 00:00:00
28	28	Benzonatate	Drug Allergy	none	chest tightness	none	2020-07-20 00:00:00
29	29	Ozempic	none	none	Coughing	none	2020-07-20 00:00:00
30	30	Trazodone	Insect Allergy	none	Coughing, chest tightness	none	2020-07-20 00:00:00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Prescription

```
1 • DESCRIBE hospital.prescription;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
prescriptionID	int unsigned	NO	PRI	NULL	auto_increment
medicationName	varchar(45)	NO		NULL	
totalAmount	varchar(45)	NO		NULL	
receivedByPatient	tinyint(1)	YES		0	
prescribedAmount	varchar(45)	NO		NULL	
frequency	varchar(45)	NO		NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 • SELECT * FROM hospital.prescription;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

prescriptionID	medicationName	totalAmount	receivedByPatient	prescribedAmount	frequency	CREATED_DATE
16	hydrocodone	1000 mg	1	500 mg	2 times	2020-07-20 00:00:00
17	levothyroxine	800 mg	0	800 mg	no	2020-07-20 00:00:00
18	prednisone	1000 mg	1	1000 mg	no	2020-07-20 00:00:00
19	amoxicillin	800 mg	0	400 mg	1 times	2020-07-20 00:00:00
20	gabapentin	200 mg	1	200 mg	no	2020-07-20 00:00:00
21	lisinopril	500 mg	1	500 mg	no	2020-07-20 00:00:00
22	atorvastatin	800 mg	1	800 mg	no	2020-07-20 00:00:00
23	metformin	1000 mg	0	800 mg	1 times	2020-07-20 00:00:00
24	ondansetron	1000 mg	1	800 mg	1 times	2020-07-20 00:00:00
25	ibuprofen	800 mg	0	800 mg	no	2020-07-20 00:00:00
26	ibuprofen	700 mg	1	700 mg	no	2020-07-20 00:00:00
27	naproxen	1000 mg	0	800 mg	1 times	2020-07-20 00:00:00
28	acetaminophen	2000 mg	1	800 mg	2 times	2020-07-20 00:00:00
29	ibuprofen	500 mg	0	500 mg	no	2020-07-20 00:00:00
30	naproxen	1000 mg	1	800 mg	1 times	2020-07-20 00:00:00
31	randoscilin	700 mg	0	69 mg	1 a day	2020-07-23 23:12:41

Note

```
1 •  DESCRIBE hospital.note;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

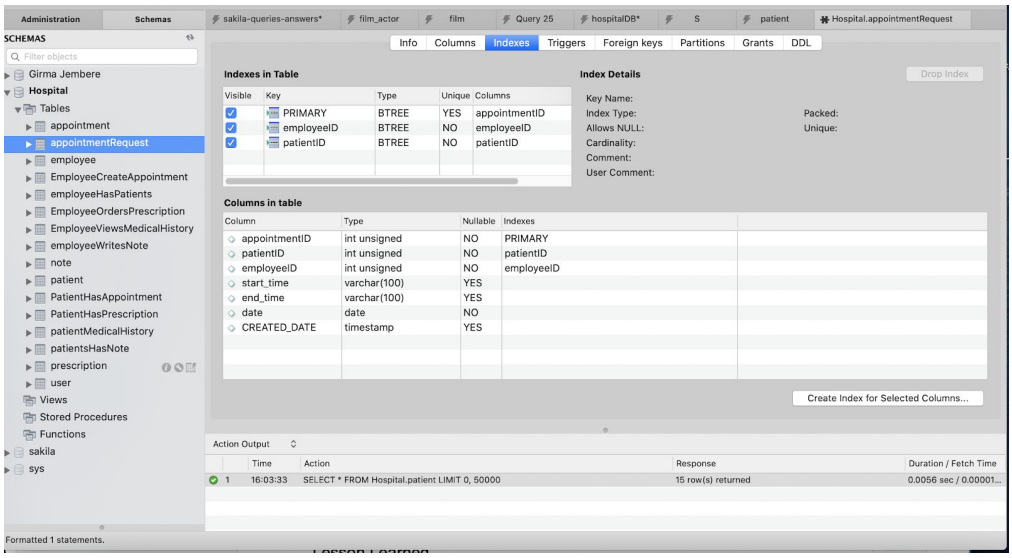
Field	Type	Null	Key	Default	Extra
noteID	int unsigned	NO	PRI	NULL	auto_increment
detail	varchar(21844)	NO		NULL	
date	date	NO		NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

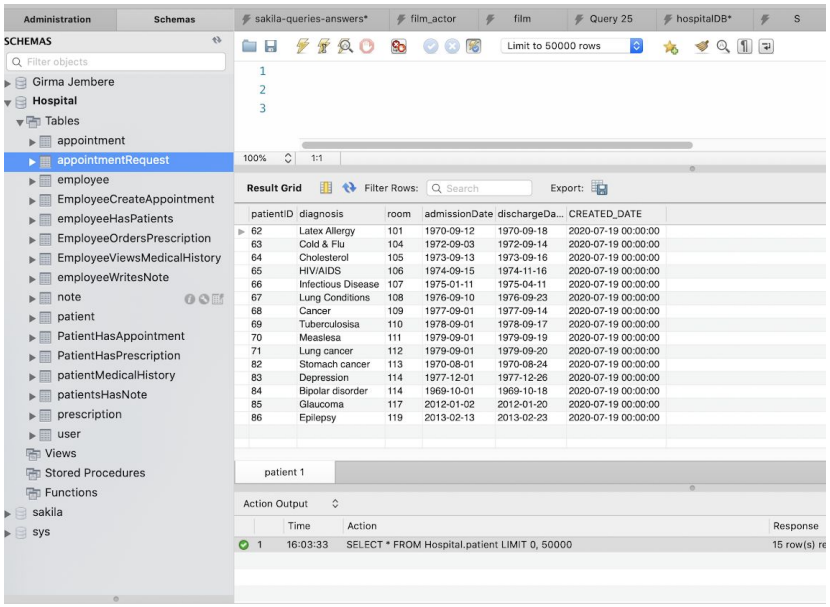
```
1 •  SELECT * FROM hospital.note;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

noteID	detail	date	CREATED_DATE
16	Blood pressure is lower. Feet are inspected and...	2019-07-18	2020-08-04 00:00:00
17	Oral cavity and pharynx normal. No inflammatio...	2019-04-12	2020-08-04 00:00:00
18	External auditory canals and tympanic membran...	2019-09-02	2020-08-04 00:00:00
19	Neck supple, non-tender without lymphadenopa...	2019-10-17	2020-08-04 00:00:00
20	No inflammation, swelling, exudate, or lesions. ...	2019-02-12	2020-08-04 00:00:00
21	No significant deformity or joint abnormality. No...	2019-08-22	2020-08-04 00:00:00
22	The head is normocephalic and atraumatic. The...	2019-07-11	2020-08-04 00:00:00
23	LUNGS: Clear to auscultation. ABDOMEN: Soft ...	2019-02-14	2020-08-04 00:00:00
24	EXTREMITIES: No cyanosis, clubbing or edema ...	2019-02-15	2020-08-04 00:00:00
25	The patient is pleasant, appears her stated age...	2019-01-18	2020-08-04 00:00:00
26	VITAL SIGNS: Currently stable. The patient is ...	2019-11-12	2020-08-04 00:00:00
27	SKIN AND EXTREMITIES: No skin rashes or lesi...	2019-09-12	2020-08-04 00:00:00
28	HEART: Regular rate and rhythm.LUNGS: Clea...	2019-02-16	2020-08-04 00:00:00
29	GENERAL: The patient is a well-developed, well...	2019-08-05	2020-08-04 00:00:00
30	LUNGS: Clear to auscultation bilaterally. No wh...	2019-09-06	2020-08-04 00:00:00

AppointmentRequest





EmployeeHasPatients

```
1 •  DESCRIBE hospital.employeehaspatients;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
patientID	int unsigned	NO	MUL	NULL	
employeeID	int unsigned	NO	MUL	NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 •  SELECT * FROM hospital.employeehaspatients;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Cont

patientID	employeeID	CREATED_DATE
16	31	2020-07-20 00:00:00
17	32	2020-07-20 00:00:00
18	33	2020-07-20 00:00:00
19	34	2020-07-20 00:00:00
20	35	2020-07-20 00:00:00
21	36	2020-07-20 00:00:00
22	37	2020-07-20 00:00:00
23	38	2020-07-20 00:00:00
24	39	2020-07-20 00:00:00
25	40	2020-07-20 00:00:00
26	41	2020-07-20 00:00:00
27	42	2020-07-20 00:00:00
28	43	2020-07-20 00:00:00
29	44	2020-07-20 00:00:00
30	45	2020-07-20 00:00:00

EmployeeCreateAppointment

```
1 •  DESCRIBE hospital.employeecreateappointment;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
appointmentID	int unsigned	NO	MUL	NULL	
employeeID	int unsigned	NO	MUL	NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 •  SELECT * FROM hospital.employeecreateappointment;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

appointmentID	employeeID	CREATED_DATE
16	31	2020-07-20 00:00:00
17	32	2020-07-20 00:00:00
18	33	2020-07-20 00:00:00
19	34	2020-07-20 00:00:00
20	35	2020-07-20 00:00:00
21	36	2020-07-20 00:00:00
22	37	2020-07-20 00:00:00
23	38	2020-07-20 00:00:00
24	39	2020-07-20 00:00:00
25	40	2020-07-20 00:00:00
26	41	2020-07-20 00:00:00
27	42	2020-07-20 00:00:00
28	43	2020-07-20 00:00:00
29	44	2020-07-20 00:00:00
30	45	2020-07-20 00:00:00

PatientHasAppointment

```
1 •  DESCRIBE hospital.patienthasappointment;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
patientID	int unsigned	NO	MUL	NULL	
appointmentID	int unsigned	NO	MUL	NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 •  SELECT * FROM hospital.patienthasappointment;
```

Result Grid | Filter Rows: | Export: | Wrap

patientID	appointmentID	CREATED_DATE
16	16	2020-07-20 00:00:00
17	17	2020-07-20 00:00:00
18	18	2020-07-20 00:00:00
19	19	2020-07-20 00:00:00
20	20	2020-07-20 00:00:00
21	21	2020-07-20 00:00:00
22	22	2020-07-20 00:00:00
23	23	2020-07-20 00:00:00
24	24	2020-07-20 00:00:00
25	25	2020-07-20 00:00:00
26	26	2020-07-20 00:00:00
27	27	2020-07-20 00:00:00
28	28	2020-07-20 00:00:00
29	29	2020-07-20 00:00:00
30	30	2020-07-20 00:00:00

EmployeeOrdersPrescription

```
1 • DESCRIBE hospital.employeeordersprescription;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
prescriptionID	int unsigned	NO	MUL	NULL	
employeeID	int unsigned	NO	MUL	NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 • SELECT * FROM hospital.employeeordersprescription;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Co

prescriptionID	employeeID	CREATED_DATE
16	31	2020-07-20 00:00:00
17	32	2020-07-20 00:00:00
18	33	2020-07-20 00:00:00
19	34	2020-07-20 00:00:00
20	35	2020-07-20 00:00:00
21	36	2020-07-20 00:00:00
22	37	2020-07-20 00:00:00
23	38	2020-07-20 00:00:00
24	39	2020-07-20 00:00:00
25	40	2020-07-20 00:00:00
26	41	2020-07-20 00:00:00
27	42	2020-07-20 00:00:00
28	43	2020-07-20 00:00:00
29	44	2020-07-20 00:00:00
30	45	2020-07-20 00:00:00

PatientHasPrescription

```
1 •  DESCRIBE hospital.patienthasprescription;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
patientID	int unsigned	NO	MUL	NULL	
prescriptionID	int unsigned	NO	MUL	NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 •  SELECT * FROM hospital.patienthasprescription;
```

Result Grid | Filter Rows: | Export: | Wrap Ce

patientID	prescriptionID	CREATED_DATE
16	16	2020-07-20 00:00:00
17	17	2020-07-20 00:00:00
18	18	2020-07-20 00:00:00
19	19	2020-07-20 00:00:00
20	20	2020-07-20 00:00:00
21	21	2020-07-20 00:00:00
22	22	2020-07-20 00:00:00
23	23	2020-07-20 00:00:00
24	24	2020-07-20 00:00:00
25	25	2020-07-20 00:00:00
26	26	2020-07-20 00:00:00
27	27	2020-07-20 00:00:00
28	28	2020-07-20 00:00:00
29	29	2020-07-20 00:00:00
30	30	2020-07-20 00:00:00

EmployeeWritesNote

```
1 •  DESCRIBE hospital.employeewritesnote;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
employeeID	int unsigned	NO	MUL	NULL	
noteID	int unsigned	NO	MUL	NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 •  SELECT * FROM hospital.employeewritesnote;
```

Result Grid | Filter Rows: | Export:

employeeID	noteID	CREATED_DATE
31	16	2020-07-20 00:00:00
32	17	2020-07-20 00:00:00
33	18	2020-07-20 00:00:00
34	19	2020-07-20 00:00:00
35	20	2020-07-20 00:00:00
36	21	2020-07-20 00:00:00
37	22	2020-07-20 00:00:00
38	23	2020-07-20 00:00:00
39	24	2020-07-20 00:00:00
40	25	2020-07-20 00:00:00
41	26	2020-07-20 00:00:00
42	27	2020-07-20 00:00:00
43	28	2020-07-20 00:00:00
44	29	2020-07-20 00:00:00
45	30	2020-07-20 00:00:00

PatientHasNote

```
1 •  DESCRIBE hospital.patientshasnote;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
patientID	int unsigned	NO	MUL	NULL	
noteID	int unsigned	NO	MUL	NULL	
CREATED_DATE	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

```
1 •  SELECT * FROM hospital.patientshasnote;
```

Result Grid | Filter Rows: | Export:

patientID	noteID	CREATED_DATE
16	16	2020-07-20 00:00:00
17	17	2020-07-20 00:00:00
18	18	2020-07-20 00:00:00
19	19	2020-07-20 00:00:00
20	20	2020-07-20 00:00:00
21	21	2020-07-20 00:00:00
22	22	2020-07-20 00:00:00
23	23	2020-07-20 00:00:00
24	24	2020-07-20 00:00:00
25	25	2020-07-20 00:00:00
26	26	2020-07-20 00:00:00
27	27	2020-07-20 00:00:00
28	28	2020-07-20 00:00:00
29	29	2020-07-20 00:00:00
30	30	2020-07-20 00:00:00

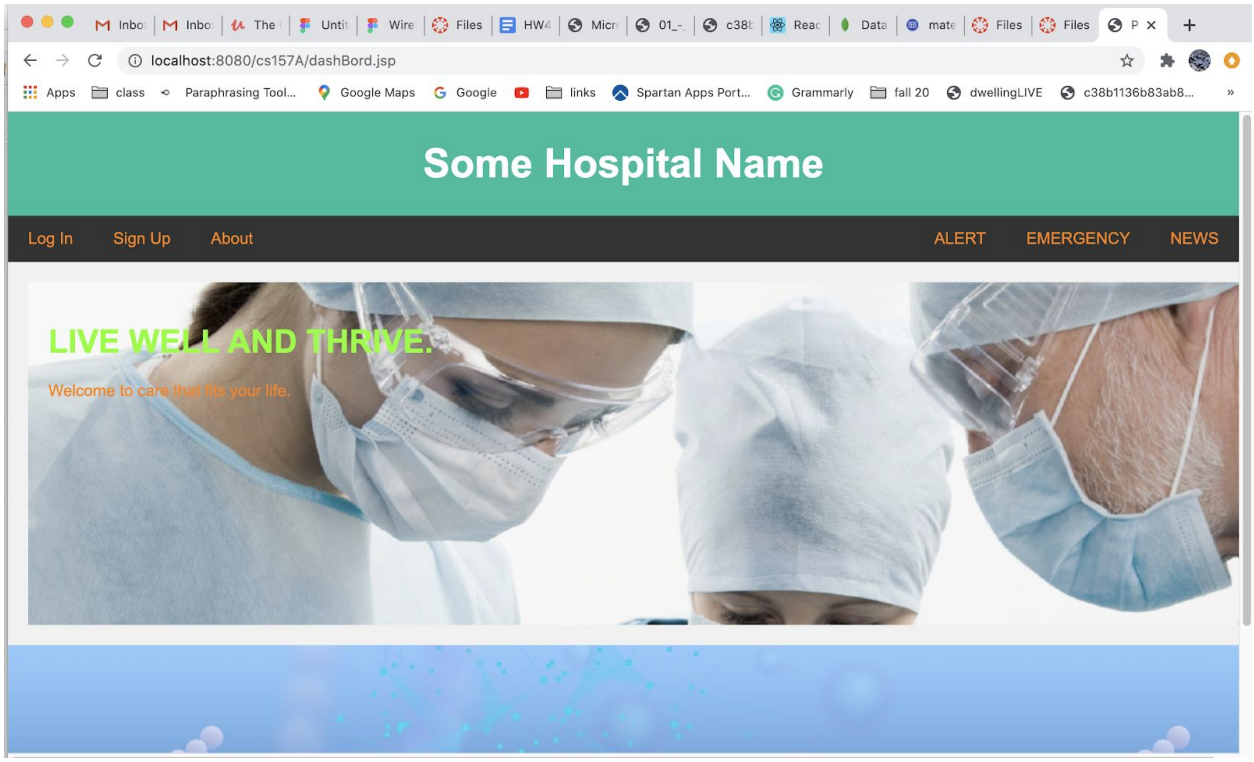
Project Implementation

How to Set up and Run the System

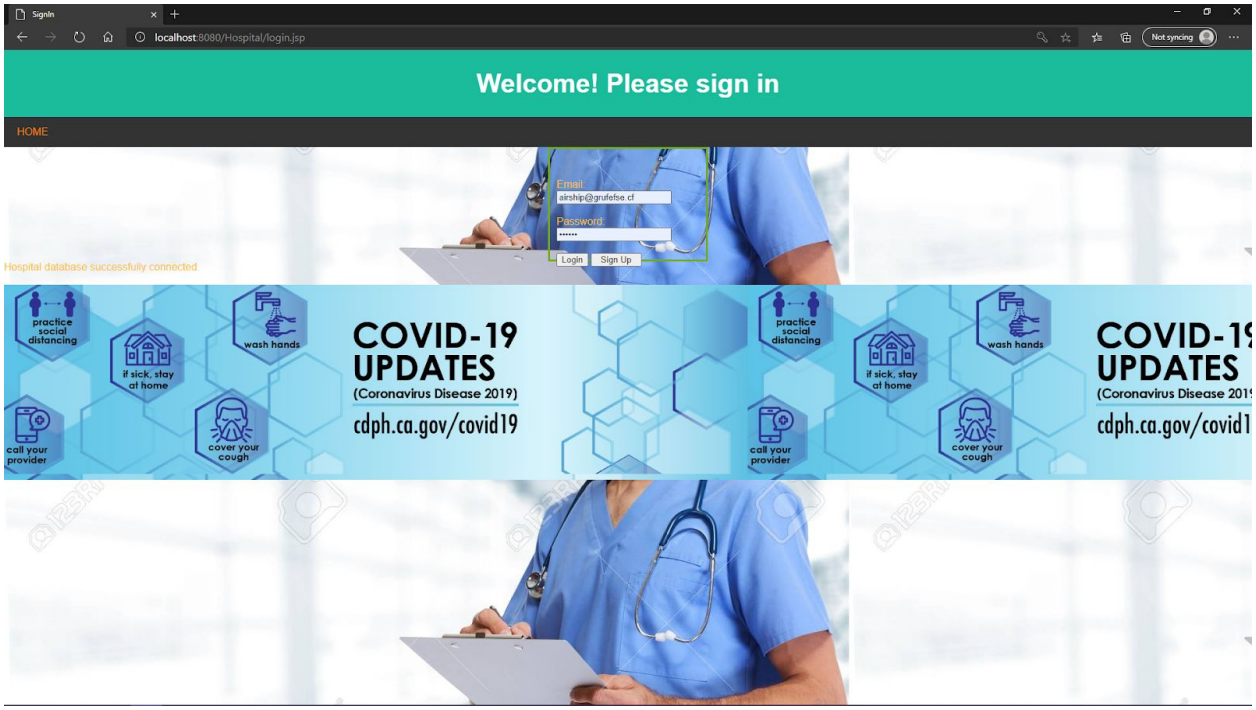
Start the MySQL Workbench, open a connection, and create a schema titled hospital using the SQL statement: CREATE SCHEMA `hospital`;. Double-click on the hospital schema on the left side of the screen to select it (hospital should be in bold). Then download and run hospitalDB.sql into MySQL to load the schema with tables. Then download TableData.jsp from the project code file and run it in Eclipse under a Dynamic Web Project (make sure to enter your own MySQL password in the file on line 32) to fill the tables with data. Then, download and import all .jsp files from the project code folder and all .jpg images from the image folder into Eclipse under a Dynamic Web Project. Open login.jsp and change the password on line 161 to your own MySQL password, then run it with the Apache Tomcat server at localhost:8080/projectName/login.jsp. Make sure that the Apache Tomcat application is installed and running.

Screenshots of Site and SQL

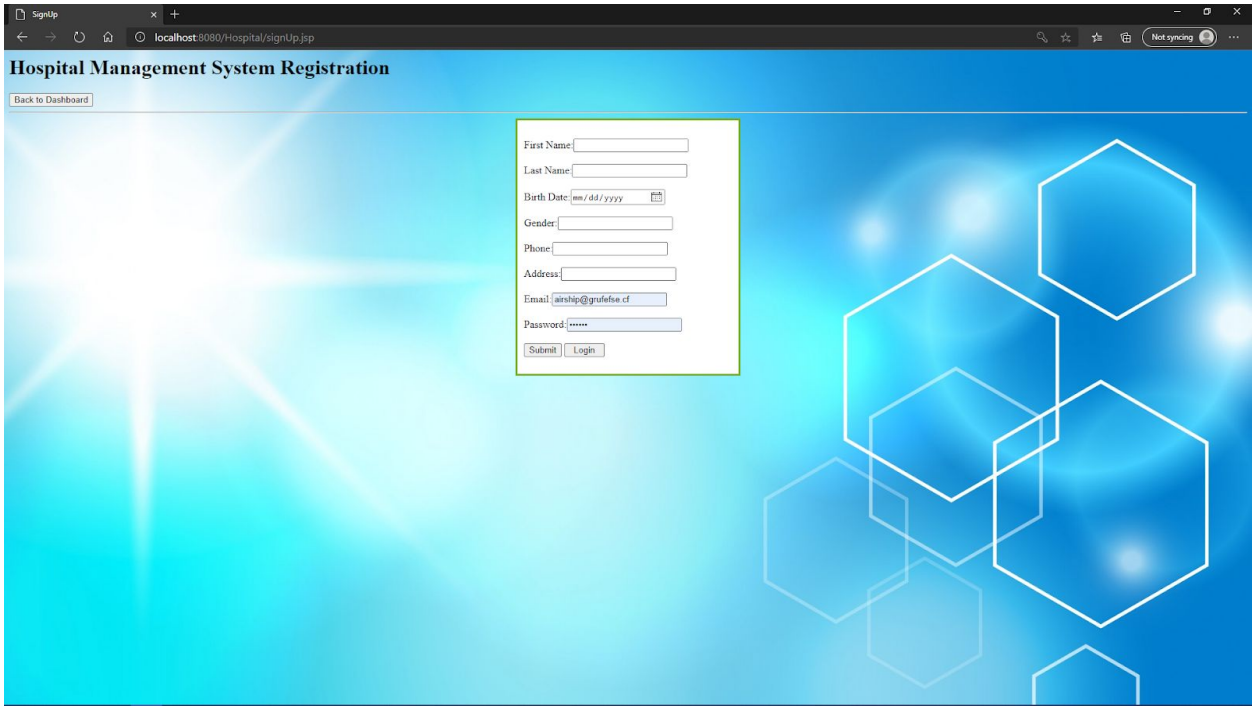
Dashboard



The screenshot shows the main homepage, with options to login and signup.



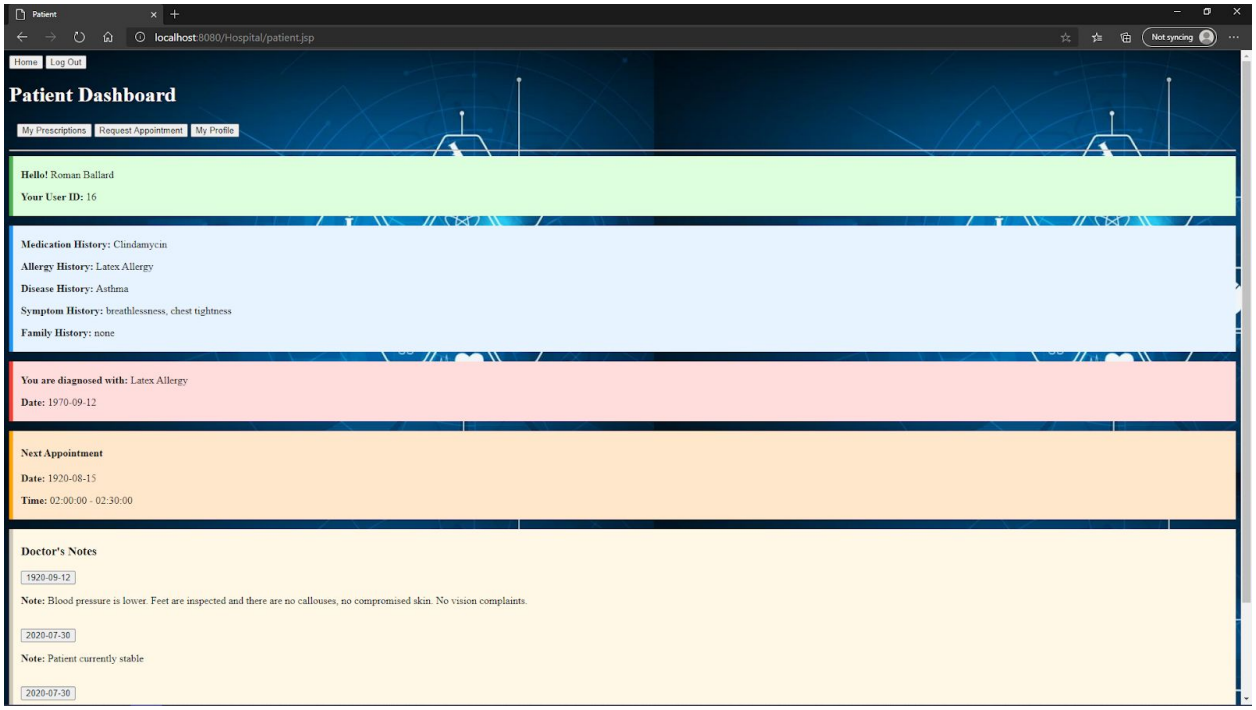
Create account page



Patient Dashboard

Patient dashboard it shows

1. patient information
2. Patient diagnose information
3. prescription detail
4. Patient medical history
5. Patient appointments
6. Patient doctor notes

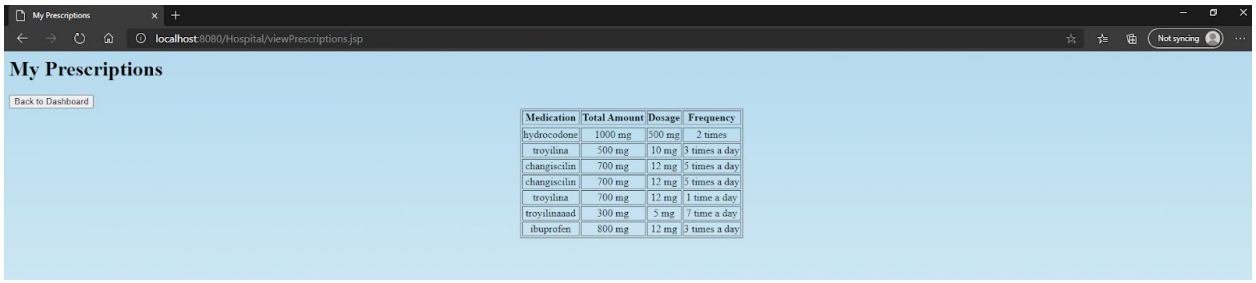


Doctor’s Note is collapsible


```
412     <button type="button" class="collapsible"><%=rs7.getString("date")%></button><br>
413
414          <p>
415              <strong>Note:</strong>
416              <%=rs7.getString("detail")%><br>
417          </p>
418
419
420      <%}}}%>
421
```

```
436 <script>
437 var coll = document.getElementsByClassName("collapsible");
438 var i;
439
440 for (i = 0; i < coll.length; i++) {
441   coll[i].addEventListener("click", function() {
442     this.classList.toggle("active");
443     var content = this.nextElementSibling;
444     if (content.style.display === "block") {
445       content.style.display = "none";
446     } else {
447       content.style.display = "block";
448     }
449   });
450 }
451 </script>
```

Patient Prescriptions



Medication	Total Amount	Dosage	Frequency
hydrocodone	1000 mg	500 mg	2 times
troyilina	500 mg	10 mg	3 times a day
changiscilin	700 mg	12 mg	5 times a day
changiscilin	700 mg	12 mg	5 times a day
troyilina	700 mg	12 mg	1 time a day
troyilinaaad	300 mg	5 mg	7 time a day
ibuprofen	800 mg	12 mg	3 times a day

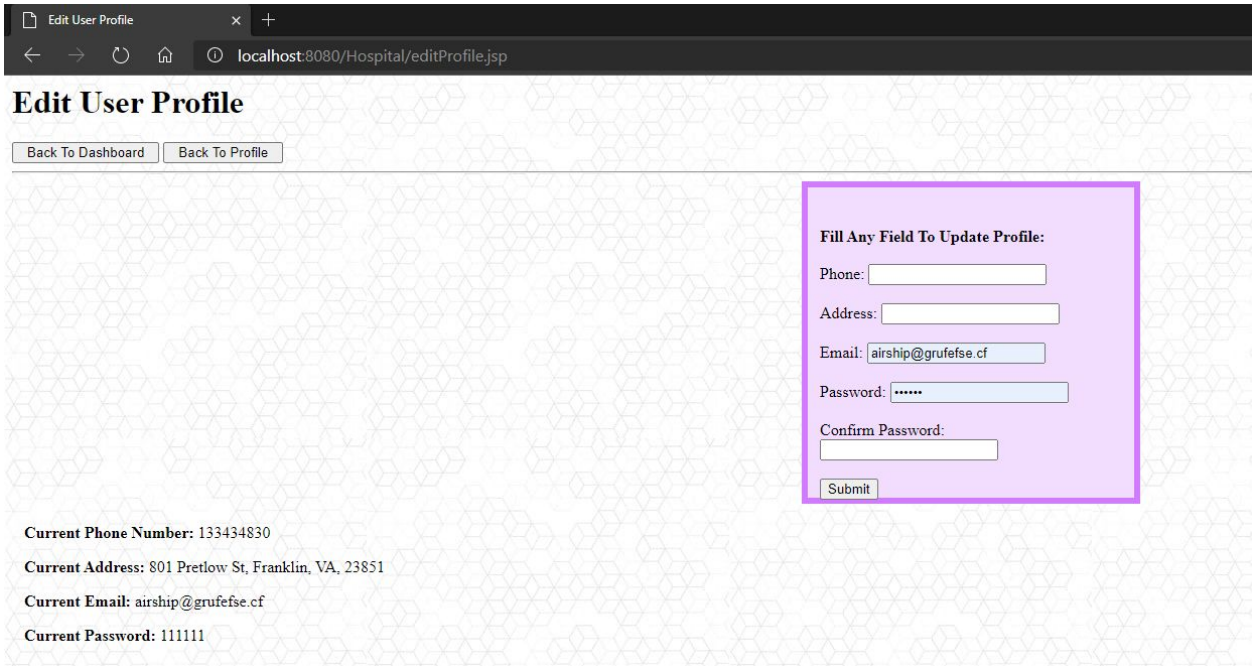
```
38      try{
39          java.sql.Connection con;
40          Class.forName("com.mysql.jdbc.Driver");
41          con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Hospital?serverTimezone=EST5EDT",user, password);
42          dbStatus = (db + " database successfully connected.<br/><br/>");
43          Statement stmt = con.createStatement();
44
45          ResultSet rs = stmt.executeQuery("SELECT *" +
46          " FROM patienthasprescription " +
47          " JOIN prescription ON patienthasprescription.prescriptionID = prescription.prescriptionID " +
48          " WHERE patientID = " + user_id);
49          %>
50              <table border=1 align=center style="text-align:center">
51              <thead>
52                  <tr>
53                      <th>Medication</th>
54                      <th>Total Amount</th>
55                      <th>Dosage</th>
56                      <th>Frequency</th>
57                  <tr>
58              </thead>
59              <tbody>
60              <%
61
62              while(rs.next()){
63                  //if the date is today or later, then print the appointment
64                  %>
65                  <tr>
66                      <td><%=rs.getString("medicationName")%></td>
67                      <td><%=rs.getString("totalAmount")%></td>
68                      <td><%=rs.getString("prescribedAmount")%></td>
69                      <td><%=rs.getString("frequency")%></td>
70
71                  </tr>
72              <%
73              }   %>
74              </tbody>
75          </table>
76
77          <%
78      }catch(Exception e){
79          if(e.getMessage() != "null")
80              out.println("Exception caught: " + e.getMessage());
81      }
```

Patient View Account Info

patient.jsp | signUp.jsp | login.jsp | dashBord.jsp | viewPrescriptions.jsp | appointment.jsp | profile.jsp

```
try {

    java.sql.Connection con;
    Class.forName("com.mysql.jdbc.Driver");
    con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Hospital?serverTimezone=EST5EDT",user, password);
    Statement stmt = con.createStatement();

    java.util.Date now = new java.util.Date();
    java.sql.Date sqlDate = new java.sql.Date(now.getTime());
    //find the patient id using the patient id
    ResultSet rs = stmt.executeQuery("SELECT * FROM user ");

    while(rs.next()) {
        if(rs.getInt(1) == (user_id) ){
            firstName = rs.getString("firstName");
            lastName = rs.getString(3);
            dateOfBirth = rs.getString(4);
            gender = rs.getString(5);
            phoneNumber = rs.getString(6);
            address = rs.getString(7);
            email = rs.getString(8);
            userPassword = rs.getString(9);
            passwordLength = userPassword.length();
            userPassword = "";
               for (int i = 0; i < passwordLength; i++) {
                   censoredPassword = censoredPassword + "*";
               }
           break;
        }
    }
    stmt.close();
    con.close();
} catch(SQLException e) {
    out.println("SQLException caught: " + e.getMessage());
}
%>


<div class="profileInfo">
  <p><strong>User:</strong> <%=firstName%> <%=lastName%></p>
   <p><strong>User ID:</strong> <%=user_id%></p>
    <p><strong>Birth Date:</strong> <%=dateOfBirth%></p>
     <p><strong>Gender:</strong> <%=gender%></p>
      <p><strong>Phone Number:</strong> <%=phoneNumber%></p>
       <p><strong>Address:</strong> <%=address%></p>
        <p><strong>Email:</strong> <%=email%></p>
         <p><strong>Password:</strong> <%=censoredPassword%></p>

</div>

<div class="edit">
  <button><a class="button" href="editProfile.jsp">Edit User Profile</a></button>
```

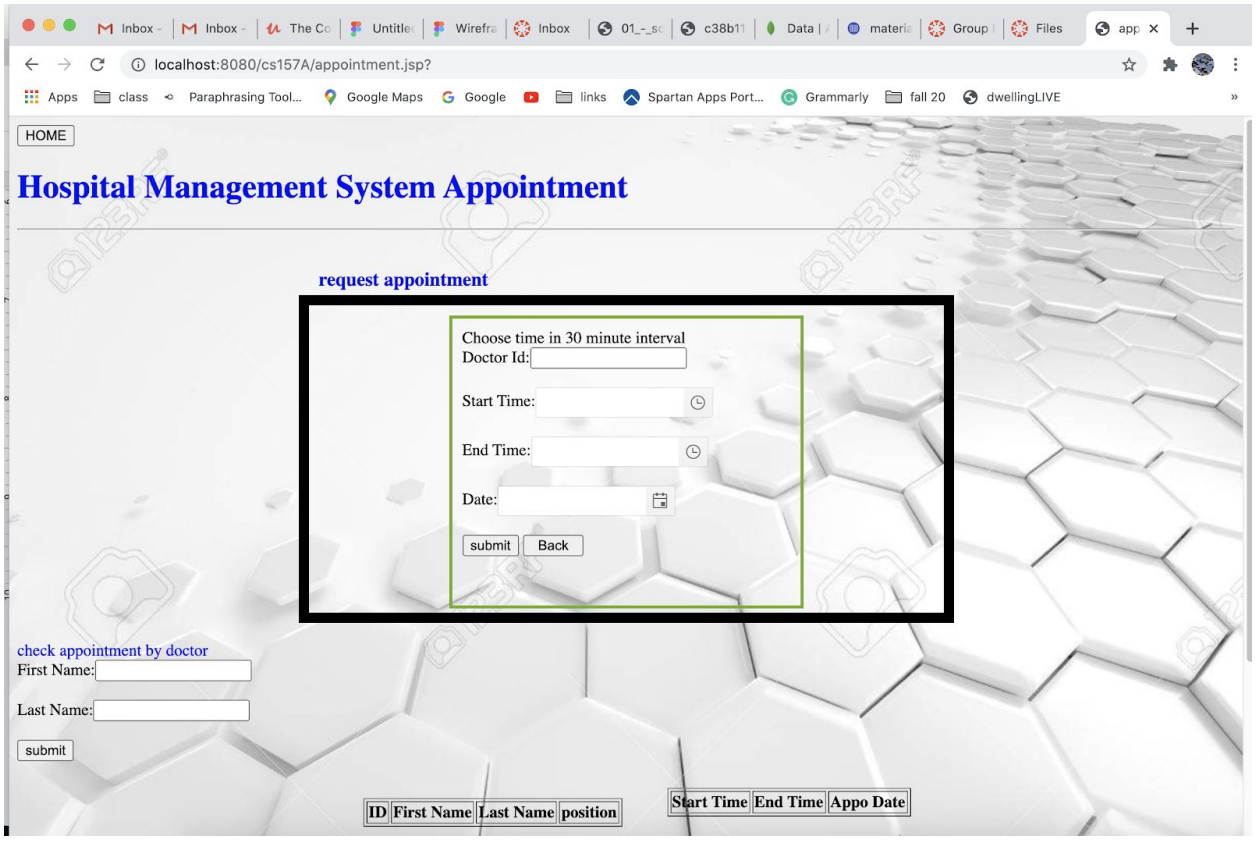
Patient Edit Account Info



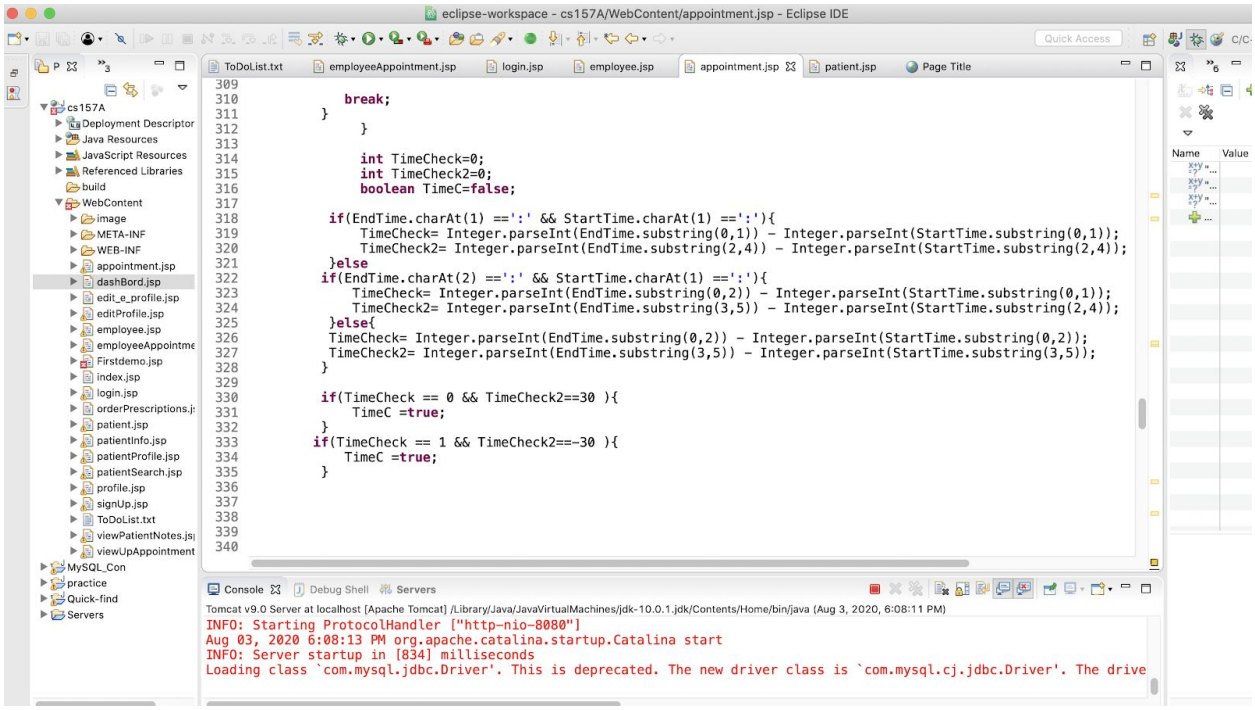
patient.jsp | signUp.jsp | login.jsp | dashBord.jsp | viewPrescriptions.jsp | appointment.jsp | profile.jsp | editProfile.jsp

```
84      try {
85
86          java.sql.Connection con;
87          Class.forName("com.mysql.jdbc.Driver");
88          con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Hospital?serverTimezone=EST5EDT",user, password);
89          Statement stmt = con.createStatement();
90
91          java.util.Date now = new java.util.Date();
92          java.sql.Date sqlDate = new java.sql.Date(now.getTime());
93
94          ResultSet rs = stmt.executeQuery("SELECT * FROM user ");
95
96          boolean formComplete = false;
97          if (phoneNumber != null || address != null || email != null || userPassword != null){
98             formComplete = true;
99          }
100
101         if (formComplete && !userPassword.equals(confirmedUserPassword)) {
102             out.print("Passwords must match! Try again");
103         } else {
104         while(rs.next()) {
105             if(rs.getInt(1) == (user_id) ){
106                 currentPhoneNumber = rs.getString(6);
107                 if (phoneNumber == "") {
108                     phoneNumber = currentPhoneNumber;
109                 }
110                 currentAddress = rs.getString(7);
111                 if (address == "") {
112                     address = currentAddress;
113                 }
114                 currentEmail = rs.getString(8);
115                 if (email == "") {
116                     email = currentEmail;
117                 }
118                 currentUserPassword = rs.getString(9);
119                 if (userPassword == "") {
120                     userPassword = currentUserPassword;
121                 }
122                 break;
123             }
124         }
125         if (formComplete) {
126         String insertSql = "UPDATE user"
127                 + "SET user.phoneNumber ='"+ phoneNumber + "', user.address = '" + address + "', user.email = '" + email + "', user.password = '" + userPassword + "'"
128                 + "WHERE user.userID = " + user_id;
129         stmt.execute(insertSql);
130         //response.sendRedirect("patient.jsp");
131         }
132         //rs.close();
133         }
134         stmt.close();
135         con.close();
136     } catch(SQLException e) {
```

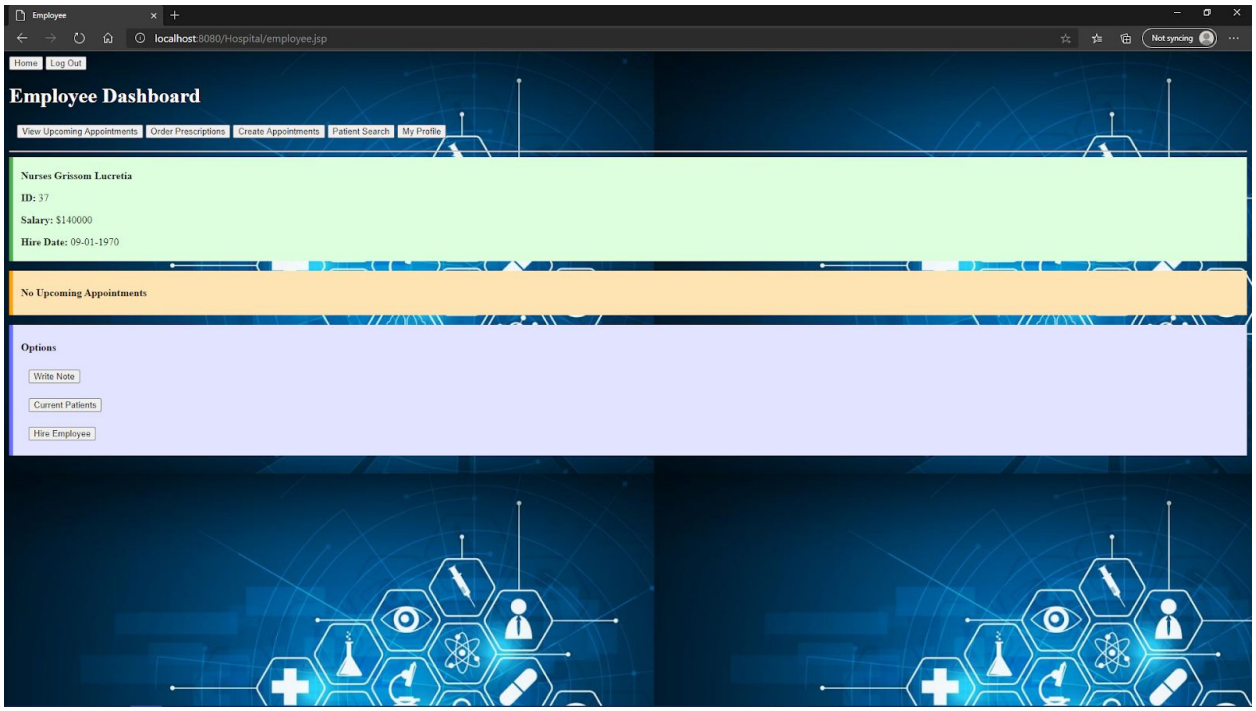
Patient request appointment



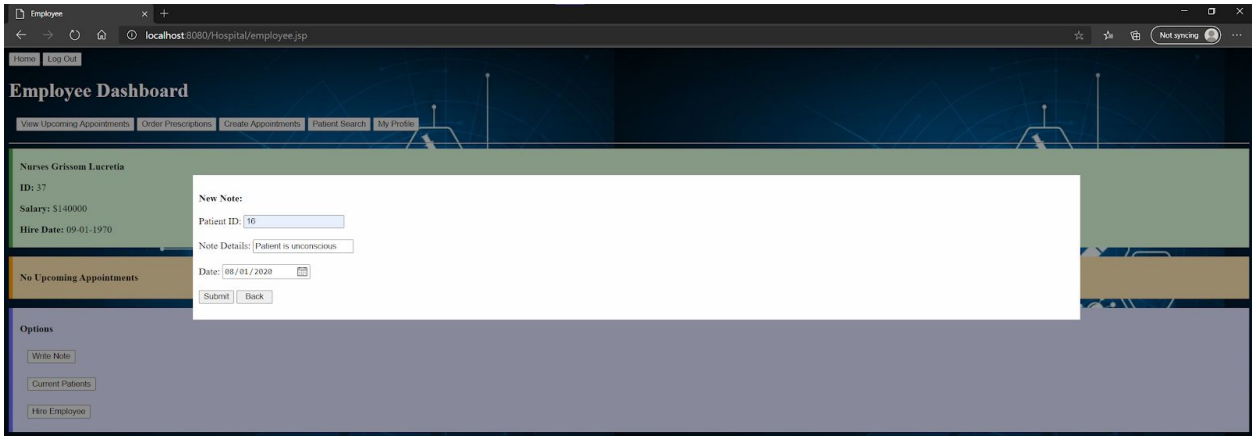
This code below restricts the time interval user selected to create appointment by subtracting the star time and end time



Employee Dashboard

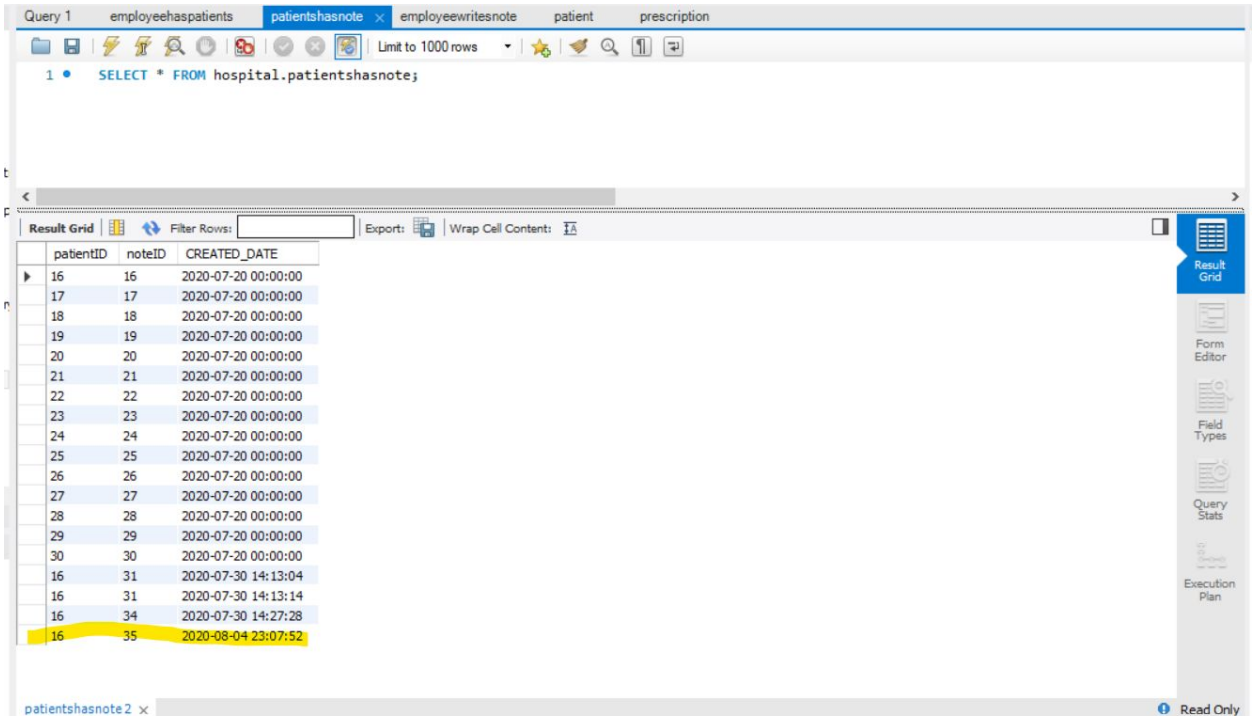


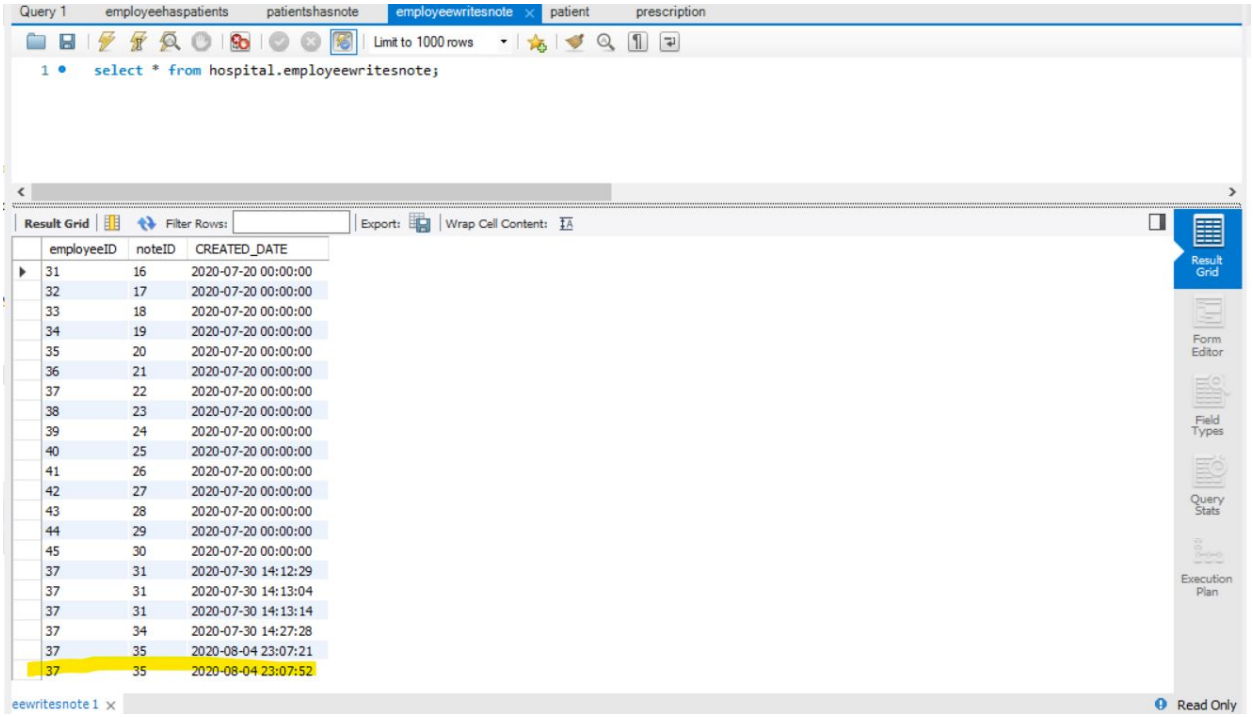
Employee Write Note

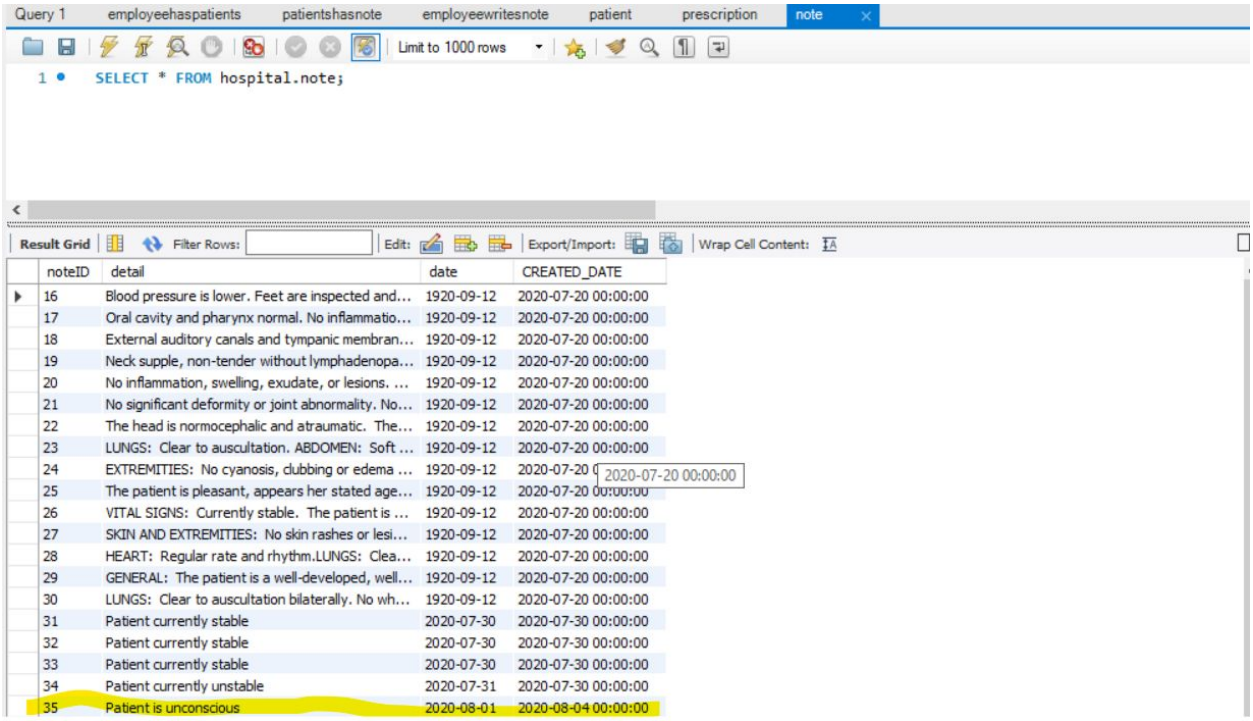


patient.jsp | signUp.jsp | login.jsp | dashBord.jsp | viewPrescriptions.jsp | appointment.jsp | profile.jsp | editProfile.jsp | employee.jsp

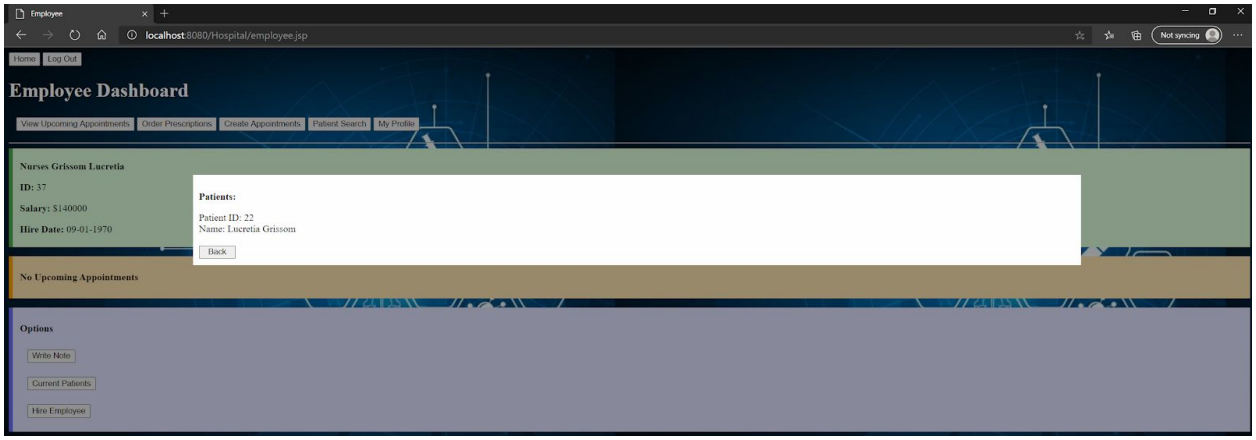
```
<div class="note">
 <button id="noteButton">Write Note</button>
</div>
<%
            String patientID = request.getParameter("patientID");
            String details = request.getParameter("details");
            String date = request.getParameter("date");
            int noteID = 0;
            boolean noteFormCompleted = false;

            try {

            java.sql.Connection con;
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Hospital?serverTimezone=EST5EDT",user, password);
            Statement stmt = con.createStatement();

            java.util.Date now = new java.util.Date();
            java.sql.Date sqlDate = new java.sql.Date(now.getTime());

            if (patientID != null && details != null && date != null) {
                noteFormCompleted = true;
            }
            if (noteFormCompleted) {
            String insertSql = "INSERT INTO note (detail, date, CREATED_DATE) "
                    + "VALUES ('" + details + "', '" + date + "','" + sqlDate + "')";

            stmt.execute(insertSql);

            ResultSet rs = stmt.executeQuery("SELECT * FROM note");

            while(rs.next()) {
                if(rs.getString(2).equals(details) && rs.getString(3).equals(date)){
                    noteID= rs.getInt(1);
                   break;
                }
            }

            insertSql = "INSERT INTO employeeWritesNote (employeeID, noteID)"
                    + "VALUES ('" + user_id + "','" + noteID + "')";

            stmt.execute(insertSql);

            insertSql = "INSERT INTO patientHasNote (patientID, noteID)"
                    + "VALUES ('" + patientID + "','" + noteID + "')";

            stmt.execute(insertSql);
            }
            stmt.close();
            con.close();
        } catch(SQLException e) {
            out.println("SQLException caught: " + e.getMessage());
```


Query 1
employeehaspatients
patientshasnote
employeewritesnote
patient
prescription
Limit to 1000 rows
1 • SELECT * FROM hospital.patientshasnote;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
patientID noteID CREATED_DATE
16 16 2020-07-20 00:00:00
17 17 2020-07-20 00:00:00
18 18 2020-07-20 00:00:00
19 19 2020-07-20 00:00:00
20 20 2020-07-20 00:00:00
21 21 2020-07-20 00:00:00
22 22 2020-07-20 00:00:00
23 23 2020-07-20 00:00:00
24 24 2020-07-20 00:00:00
25 25 2020-07-20 00:00:00
26 26 2020-07-20 00:00:00
27 27 2020-07-20 00:00:00
28 28 2020-07-20 00:00:00
29 29 2020-07-20 00:00:00
30 30 2020-07-20 00:00:00
16 31 2020-07-30 14:13:04
16 31 2020-07-30 14:13:14
16 34 2020-07-30 14:27:28
16 35 2020-08-04 23:07:52
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan
patientshasnote 2
Read Only


Query 1
employeehaspatients
patientshasnote
employeewritesnote
patient
prescription
Limit to 1000 rows
1 • select * from hospital.employeewritesnote;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
employeeID noteID CREATED_DATE
31 16 2020-07-20 00:00:00
32 17 2020-07-20 00:00:00
33 18 2020-07-20 00:00:00
34 19 2020-07-20 00:00:00
35 20 2020-07-20 00:00:00
36 21 2020-07-20 00:00:00
37 22 2020-07-20 00:00:00
38 23 2020-07-20 00:00:00
39 24 2020-07-20 00:00:00
40 25 2020-07-20 00:00:00
41 26 2020-07-20 00:00:00
42 27 2020-07-20 00:00:00
43 28 2020-07-20 00:00:00
44 29 2020-07-20 00:00:00
45 30 2020-07-20 00:00:00
37 31 2020-07-30 14:12:29
37 31 2020-07-30 14:13:04
37 31 2020-07-30 14:13:14
37 34 2020-07-30 14:27:28
37 35 2020-08-04 23:07:21
37 35 2020-08-04 23:07:52
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan
eewritesnote 1
Read Only

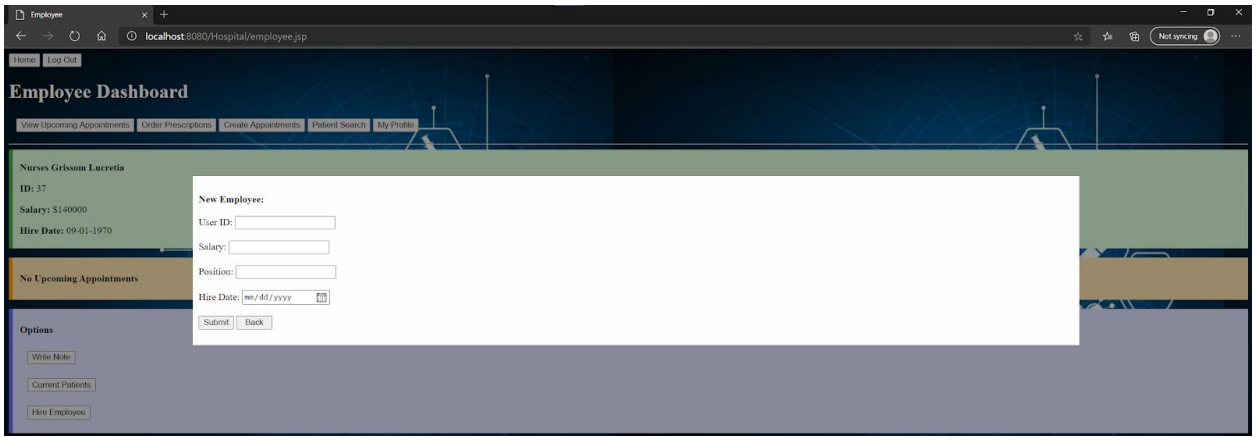

Query 1
employeehaspatients
patientshasnote
employeewritesnote
patient
prescription
note
Limit to 1000 rows
1 • SELECT * FROM hospital.note;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
noteID detail date CREATED_DATE
16 Blood pressure is lower. Feet are inspected and... 1920-09-12 2020-07-20 00:00:00
17 Oral cavity and pharynx normal. No inflammatio... 1920-09-12 2020-07-20 00:00:00
18 External auditory canals and tympanic membran... 1920-09-12 2020-07-20 00:00:00
19 Neck supple, non-tender without lymphadenopa... 1920-09-12 2020-07-20 00:00:00
20 No inflammation, swelling, exudate, or lesions. ... 1920-09-12 2020-07-20 00:00:00
21 No significant deformity or joint abnormality. No... 1920-09-12 2020-07-20 00:00:00
22 The head is normocephalic and atraumatic. The... 1920-09-12 2020-07-20 00:00:00
23 LUNGS: Clear to auscultation. ABDOMEN: Soft ... 1920-09-12 2020-07-20 00:00:00
24 EXTREMITIES: No cyanosis, clubbing or edema ... 1920-09-12 2020-07-20 0
2020-07-20 00:00:00
25 The patient is pleasant, appears her stated age... 1920-09-12 2020-07-20 00:00:00
26 VITAL SIGNS: Currently stable. The patient is ... 1920-09-12 2020-07-20 00:00:00
27 SKIN AND EXTREMITIES: No skin rashes or lesi... 1920-09-12 2020-07-20 00:00:00
28 HEART: Regular rate and rhythm.LUNGS: Clea... 1920-09-12 2020-07-20 00:00:00
29 GENERAL: The patient is a well-developed, well... 1920-09-12 2020-07-20 00:00:00
30 LUNGS: Clear to auscultation bilaterally. No wh... 1920-09-12 2020-07-20 00:00:00
31 Patient currently stable 2020-07-30 2020-07-30 00:00:00
32 Patient currently stable 2020-07-30 2020-07-30 00:00:00
33 Patient currently stable 2020-07-30 2020-07-30 00:00:00
34 Patient currently unstable 2020-07-31 2020-07-30 00:00:00
35 Patient is unconscious 2020-08-01 2020-08-04 00:00:00

Employee View Current Patients



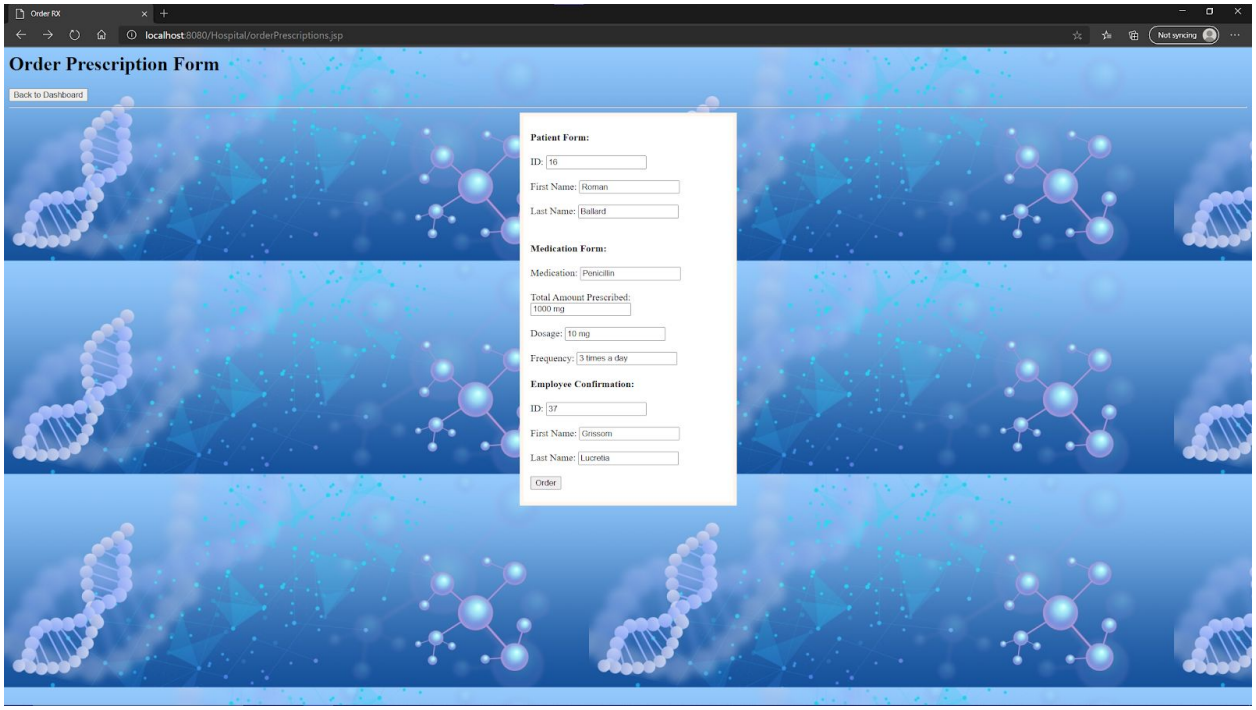

```
341
342 try {
343
344     java.sql.Connection con;
345     Class.forName("com.mysql.jdbc.Driver");
346     con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Hospital?serverTimezone=EST5EDT",user, password);
347     Statement stmt = con.createStatement();
348
349     java.util.Date now = new java.util.Date();
350     java.sql.Date sqlDate = new java.sql.Date(now.getTime());
351     //find the patient id using the patient id
352
353     ResultSet rs = stmt.executeQuery("SELECT * FROM user");
354
355
356     ResultSet rs2 = stmt.executeQuery("SELECT * FROM employeeHasPatients");
357
358     while(rs2.next()) {
359         if(rs2.getInt("employeeID") == user_id){
360             patientIDs.add(rs2.getString("patientID"));
361            break;
362         }
363     }
364
365     ResultSet rs3 = stmt.executeQuery("SELECT * FROM user");
366
367     while(rs3.next()) {
368         for (String ID : patientIDs) {
369             if(rs3.getString(1).equals(ID)){
370                 ArrayList<String> patient = new ArrayList<String>();
371                 patient.add(rs3.getString(1));
372                 patient.add(rs3.getString(2));
373                 patient.add(rs3.getString(3));
374                 patients.add(patient);
375                break;
376             }
377         }
378     }
379
380     stmt.close();
381     con.close();
382 } catch(SQLException e) {
383     out.println("SQLException caught: " + e.getMessage());
384 }
385
386 %>
```

Employee Hire Employee




```jsp
<div class="addEmployee">
 <button id="addEmployeeButton">Hire Employee</button>
</div>

<%
    String employeeID = request.getParameter("userID");
    String salaryS = request.getParameter("salary");
    String positionS = request.getParameter("position");
    String hireDate = request.getParameter("hireDate");
    boolean employeeFormCompleted = false;

        try {

            java.sql.Connection con;
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Hospital?serverTimezone=EST5EDT",user, password);
            Statement stmt = con.createStatement();

            java.util.Date now = new java.util.Date();
            java.sql.Date sqlDate = new java.sql.Date(now.getTime());

            if (employeeID != null && salaryS != null && salaryS != null && hireDate != null) {
                employeeFormCompleted = true;
            }
            String insertSql = "INSERT INTO employee (employeeID, salary, position, hireDate, CREATED_DATE) "
                    + "VALUES ('" + employeeID + "', '" + salaryS + "','" + positionS + "','" + hireDate + "','" + sqlDate + "')";
            if (employeeFormCompleted) {
                stmt.execute(insertSql);
            } stmt.close();
            con.close();
        } catch(Exception e){
            if(e.getMessage() != "null")
                out.println(e.getMessage());
        }
%>
<%-- ADD EMPLOYEE END --%>

  <%-- ADD EMPLOYEE MODAL START --%>
<div id="addEmployeeModal" class="modal">
<div class="modal-content">
    <form action="employee.jsp" method="post">
    <br /><strong> New Employee:</strong><br/>
    <br /> User ID: <input type="text" name="userID" required/><br />
    <br /> Salary: <input type="text" name="salary" required/><br />
    <br /> Position: <input type="text" name="position" required/><br />
    <br /> Hire Date: <input type="date" name="hireDate" required/><br />
    <br /> <input type="submit" value="Submit" />
     <button><a class="button" href="employee.jsp">Back</a></button>
    </form>
    </div>
</div>

<script>
```

Employee Orders Prescription

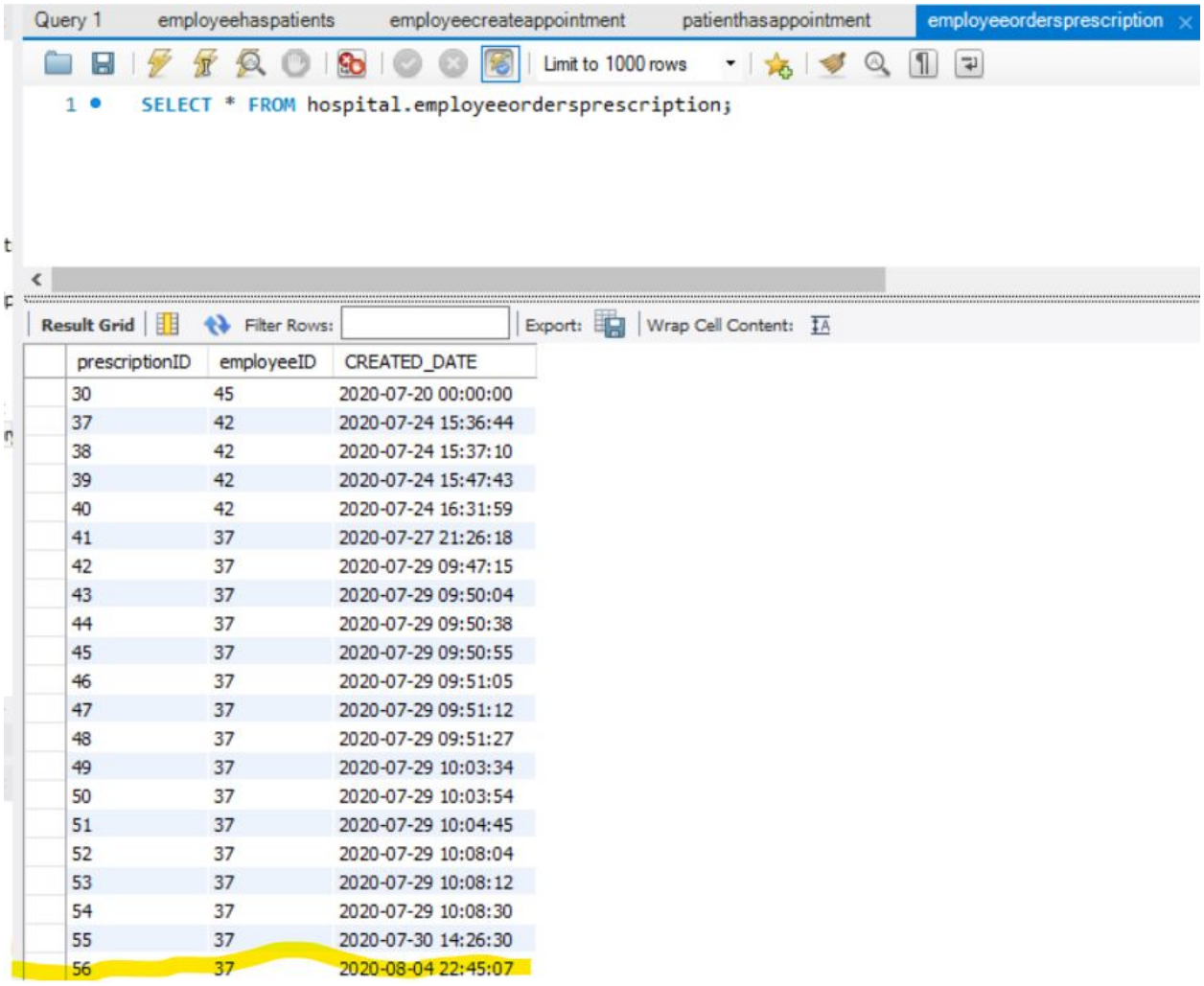


patient.jsp | signUp.jsp | login.jsp | dashBord.jsp | viewPrescriptions.jsp | appointment.jsp | profile.jsp | editProfile.jsp | employee.jsp | orderPrescriptions.jsp

```
94          String dose = request.getParameter("dose");
95          String freq = request.getParameter("frequency");
96
97          int user_id = (int)session.getAttribute("user_id");
98          int e_id = Integer.parseInt(request.getParameter("e_id"));
99          String e_fname = request.getParameter("e_fname");
100         String e_lname = request.getParameter("e_lname");
101
102         //check if the employee confirmation and the email logged in with matches
103         ResultSet rs = stmt.executeQuery("SELECT userID, firstName, lastName" +
104                 " FROM user" +
105                 " JOIN employee ON userID = employeeID" +
106                 " WHERE userID = " + e_id +
107                 " AND firstName = '" + e_fname + "'" +
108                 " AND lastName = '" + e_lname + "'");
109
110
111         if(rs.next() && user_id==e_id);
112         else
113             throw new Exception("Invalid employee confirmation");
114
115         //check if the patient exists in the table
116         rs = stmt.executeQuery("SELECT userID, firstName, lastName" +
117                 " FROM user" +
118                 " JOIN patient ON userID = patientID" +
119                 " WHERE userID = " + p_id +
120                 " AND firstName = '" + p_fname + "'" +
121                 " AND lastName = '" + p_lname + "'");
122
123         if(rs.next());
124         else
125             throw new Exception("Invalid patient information");
126
127         //generate and format the timestamp
128         java.util.Calendar calendar = java.util.Calendar.getInstance();
129         java.sql.Timestamp current_timestamp = new java.sql.Timestamp(calendar.getTime().getTime());
130         SimpleDateFormat noSeconds = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
131         String currTimeStr = noSeconds.format(current_timestamp);
132
133         //insert prescription information
134         stmt.execute("INSERT INTO prescription(medicationName,totalAmount,prescribedAmount,frequency, CREATED_DATE)" +
135                 " VALUES ('" + medication + "', '" + med_amt + "', '" + dose + "', '" + freq + "', '" + currTimeStr + "')");
136         //get the prescription id associated with the prescription information
137         rs = stmt.executeQuery("SELECT prescriptionID" +
138                 " FROM prescription" +
139                 " WHERE medicationName = '" + medication +
140                 "' AND totalAmount = '" + med_amt +
141                 "' AND prescribedAmount = '" + dose +
142                 "' AND frequency = '" + freq +
143                 "' AND CREATED_DATE = '" + currTimeStr +"'");
144
145         int prescription_id = 0;
146         if(rs.next())
```

The code validates employees and patient information before inserting the medication

This appears when the prescription order is valid



prescriptionID	employeeID	CREATED_DATE
30	45	2020-07-20 00:00:00
37	42	2020-07-24 15:36:44
38	42	2020-07-24 15:37:10
39	42	2020-07-24 15:47:43
40	42	2020-07-24 16:31:59
41	37	2020-07-27 21:26:18
42	37	2020-07-29 09:47:15
43	37	2020-07-29 09:50:04
44	37	2020-07-29 09:50:38
45	37	2020-07-29 09:50:55
46	37	2020-07-29 09:51:05
47	37	2020-07-29 09:51:12
48	37	2020-07-29 09:51:27
49	37	2020-07-29 10:03:34
50	37	2020-07-29 10:03:54
51	37	2020-07-29 10:04:45
52	37	2020-07-29 10:08:04
53	37	2020-07-29 10:08:12
54	37	2020-07-29 10:08:30
55	37	2020-07-30 14:26:30
56	37	2020-08-04 22:45:07

```
select * from hospital.patienthasprescription;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

patientID	prescriptionID	CREATED_DATE
30	30	2020-07-20 00:00:00
16	37	2020-07-24 15:36:44
16	38	2020-07-24 15:37:10
16	39	2020-07-24 15:47:43
16	40	2020-07-24 16:31:59
16	41	2020-07-27 21:26:18
17	42	2020-07-29 09:47:15
17	43	2020-07-29 09:50:04
17	44	2020-07-29 09:50:38
17	45	2020-07-29 09:50:55
17	46	2020-07-29 09:51:05
17	47	2020-07-29 09:51:12
17	48	2020-07-29 09:51:27
17	49	2020-07-29 10:03:34
17	50	2020-07-29 10:03:54
17	51	2020-07-29 10:04:45
17	52	2020-07-29 10:08:04
17	53	2020-07-29 10:08:12
17	54	2020-07-29 10:08:30
16	55	2020-07-30 14:26:30
16	56	2020-08-04 22:45:07

Result Grid, Form Editor, Field Types, Query Stats, Execution Plan

```
SELECT * FROM hospital.prescription;
```

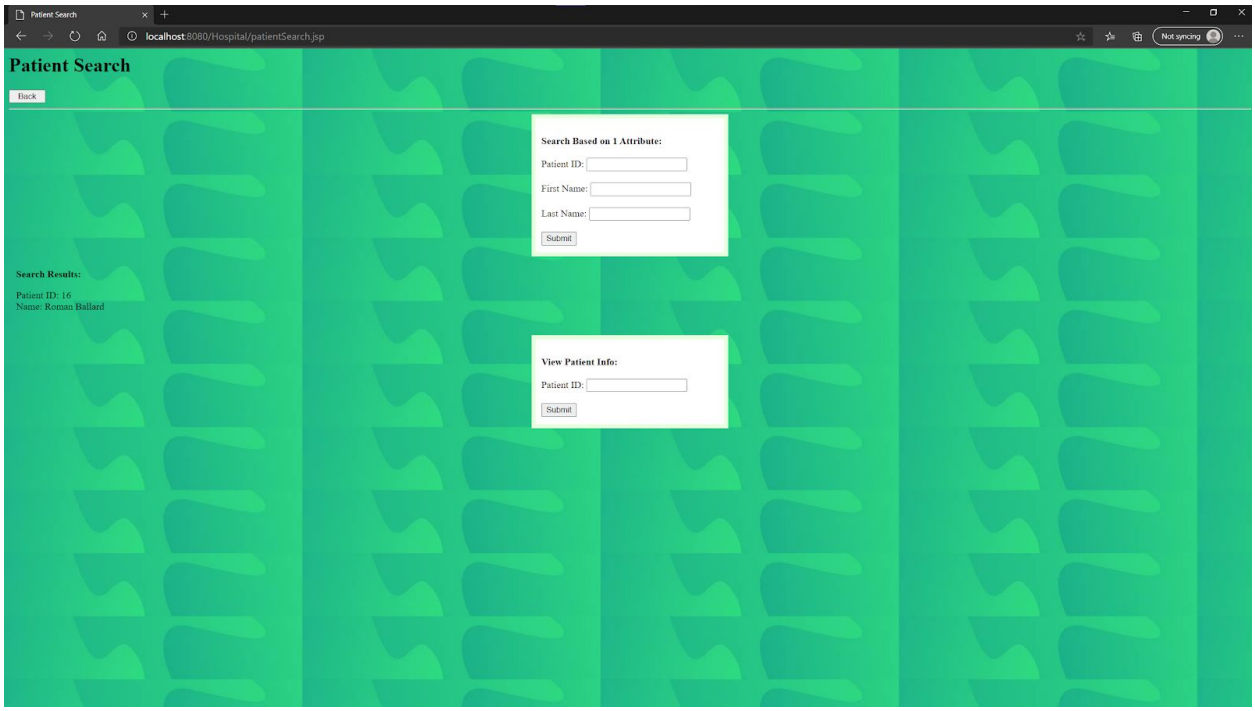
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

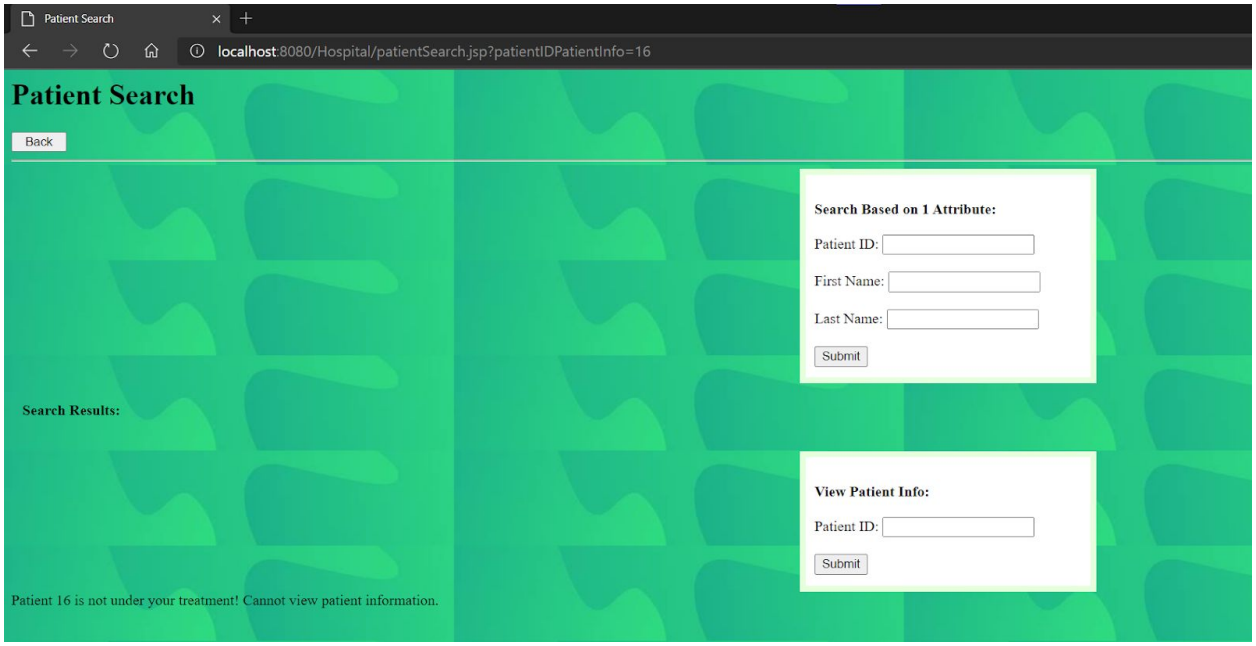
prescriptionID	medicationName	totalAmount	receivedByPatient	prescribedAmount	frequency	CREATED_DATE
37	troyilina	500 mg	0	10 mg	3 times a ...	2020-07-24 15:36:44
38	changiscilin	700 mg	0	12 mg	5 times a ...	2020-07-24 15:37:10
39	changiscilin	700 mg	0	12 mg	5 times a ...	2020-07-24 15:47:43
40	troyilina	700 mg	0	12 mg	1 time a ...	2020-07-24 16:31:59
41	troyilinaaad	300 mg	0	5 mg	7 time a ...	2020-07-27 21:26:18
42	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 09:47:15
43	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 09:50:04
44	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 09:50:38
45	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 09:50:55
46	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 09:51:05
47	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 09:51:12
48	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 09:51:27
49	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 10:03:34
50	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 10:03:54
51	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 10:04:45
52	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 10:08:04
53	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 10:08:12
54	Penicillin	800 mg	0	12 mg	1 time a ...	2020-07-29 10:08:30
55	ibuprofen	800 mg	0	12 mg	3 times a ...	2020-07-30 14:26:30
56	Penicillin	1000 mg	0	10 mg	3 times a ...	2020-08-04 22:45:07
NULL	NULL	NULL	NULL	NULL	NULL	NULL

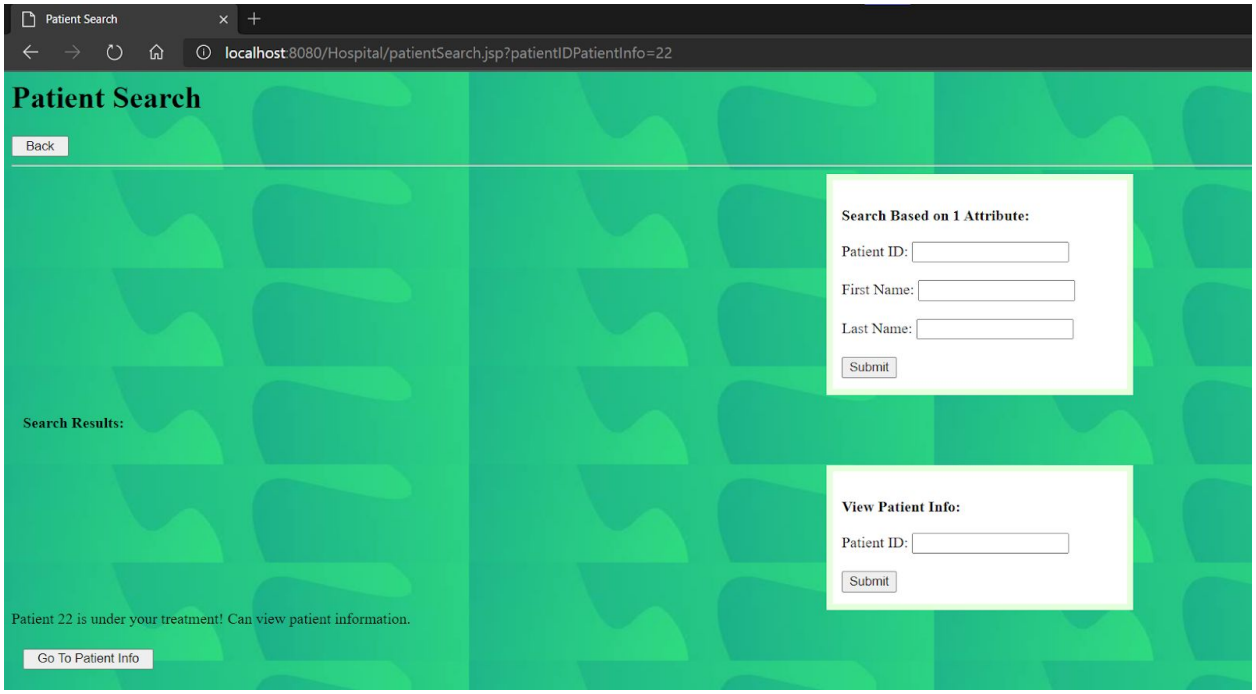
Result Grid, Form Editor, Field Types, Query Stats, Execution Plan

prescription 1 | Apply | Revert

Employee Search Patient





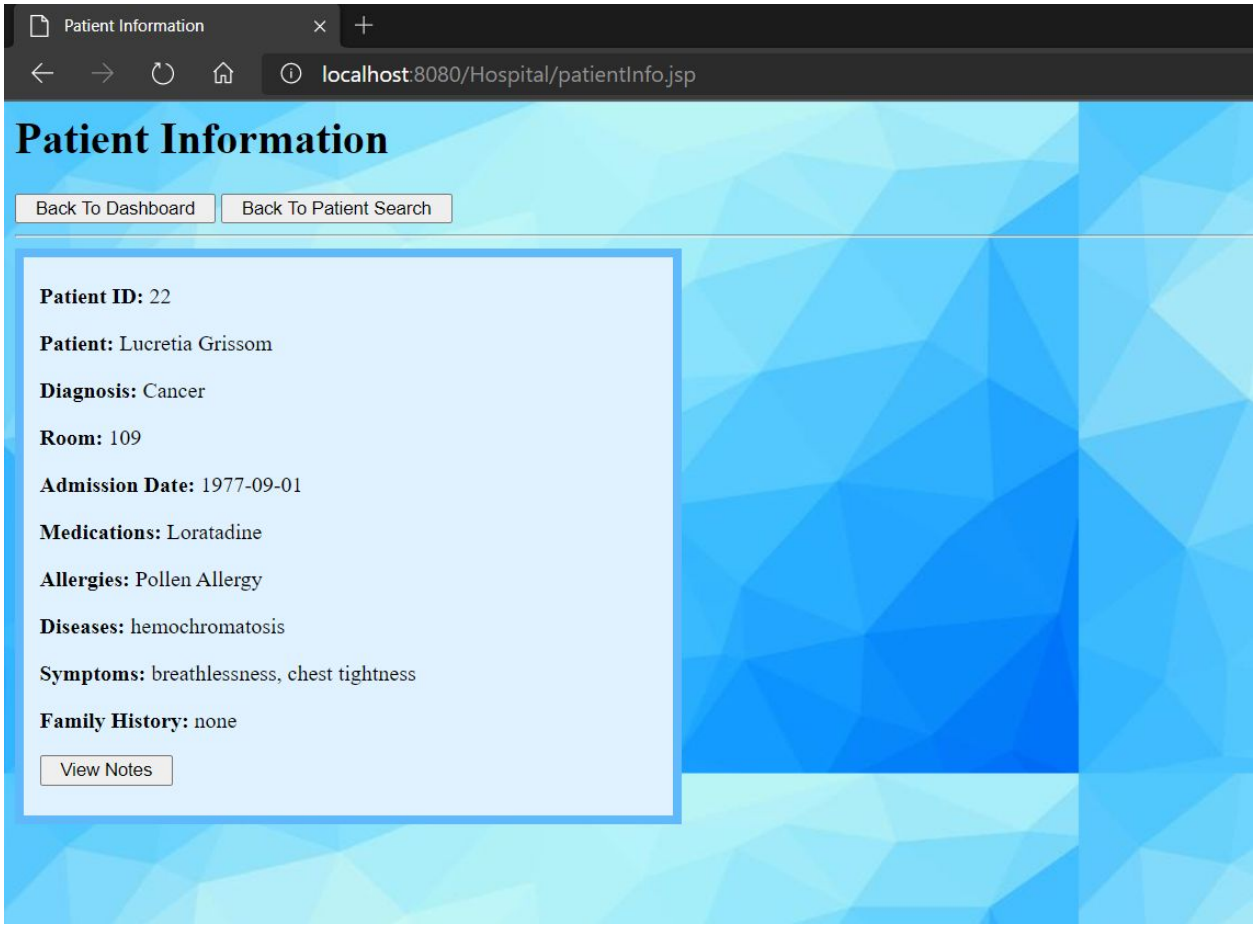


This is the code for the search

patient.jsp | signUp.jsp | login.jsp | dashBord.jsp | viewPrescriptions.jsp | appointment.jsp | profile.jsp | editProfile.jsp | employee.jsp | orderPrescriptions.jsp | patientSearch.jsp

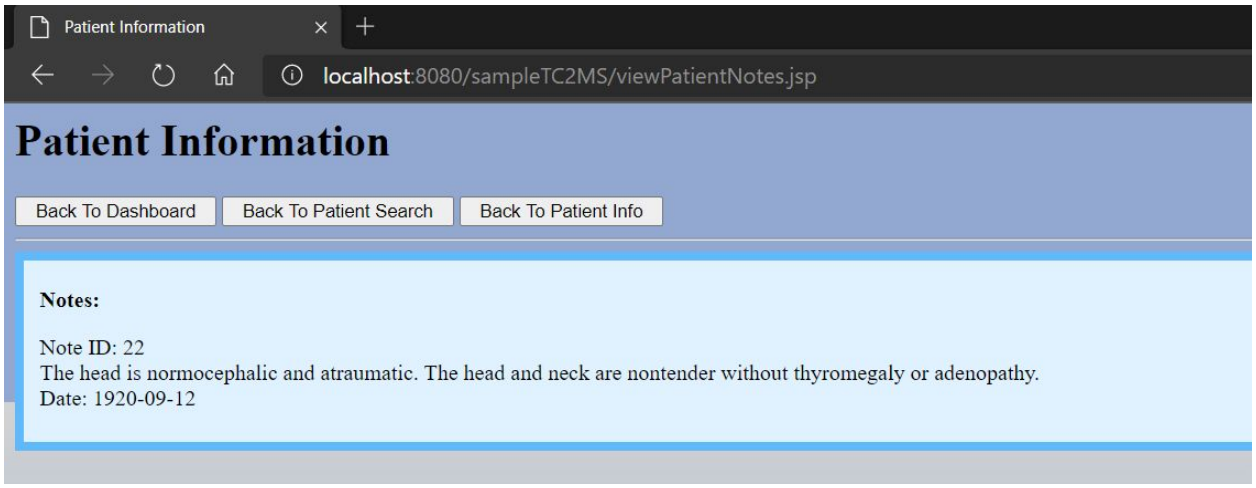
```
        java.util.Date now = new java.util.Date();
        java.sql.Date sqlDate = new java.sql.Date(now.getTime());

        if (patientIDInput != null || firstNameInput != null || lastNameInput != null) {
            formCompleted = true;
        }

        if (patientIDInput != null) {
            ResultSet rs2 = stmt.executeQuery("SELECT * FROM user");
            while(rs2.next()) {
                if(rs2.getString(1).equals(patientIDInput) ){
                    ArrayList<String> patient = new ArrayList<String>();
                    patient.add(patientIDInput);
                    patient.add(rs2.getString(2));
                    patient.add(rs2.getString(3));
                    results.add(patient);
                }
            }
        }

        if (firstNameInput != null) {
            ResultSet rs3 = stmt.executeQuery("SELECT * FROM user");
            while(rs3.next()) {
                if(rs3.getString(2).equals(firstNameInput) ){
                ArrayList<String> possiblePatientByFirstName = new ArrayList<String>();
                possiblePatientByFirstName.add(rs3.getString(1));
                possiblePatientByFirstName.add(rs3.getString(2));
                possiblePatientByFirstName.add(rs3.getString(3));
                possiblePatientsByFirstName.add(possiblePatientByFirstName);
                }
            }

            for (int i = 0; i < possiblePatientsByFirstName.size(); i++) {
                ResultSet rs4 = stmt.executeQuery("SELECT * FROM patient");
                while(rs4.next()) {
                    if(rs4.getString(1).equals(possiblePatientsByFirstName.get(i).get(0))){
                    results.add(possiblePatientsByFirstName.get(i));
                    }
                }
            }
        }

        if (lastNameInput != null) {
            ResultSet rs5 = stmt.executeQuery("SELECT * FROM user");
            while(rs5.next()) {
                if(rs5.getString(3).equals(lastNameInput) ){
                ArrayList<String> possiblePatientByLastName = new ArrayList<String>();
                possiblePatientByLastName.add(rs5.getString(1));
                possiblePatientByLastName.add(rs5.getString(2));
                possiblePatientByLastName.add(rs5.getString(3));
                possiblePatientsByLastName.add(possiblePatientByLastName);
                }
            }

            for (int i = 0; i < possiblePatientsByLastName.size(); i++) {
                ResultSet rs6 = stmt.executeQuery("SELECT * FROM patient");
```

This is the page linked when clicking “Go To Patient Info”



Query the database for a single patient information

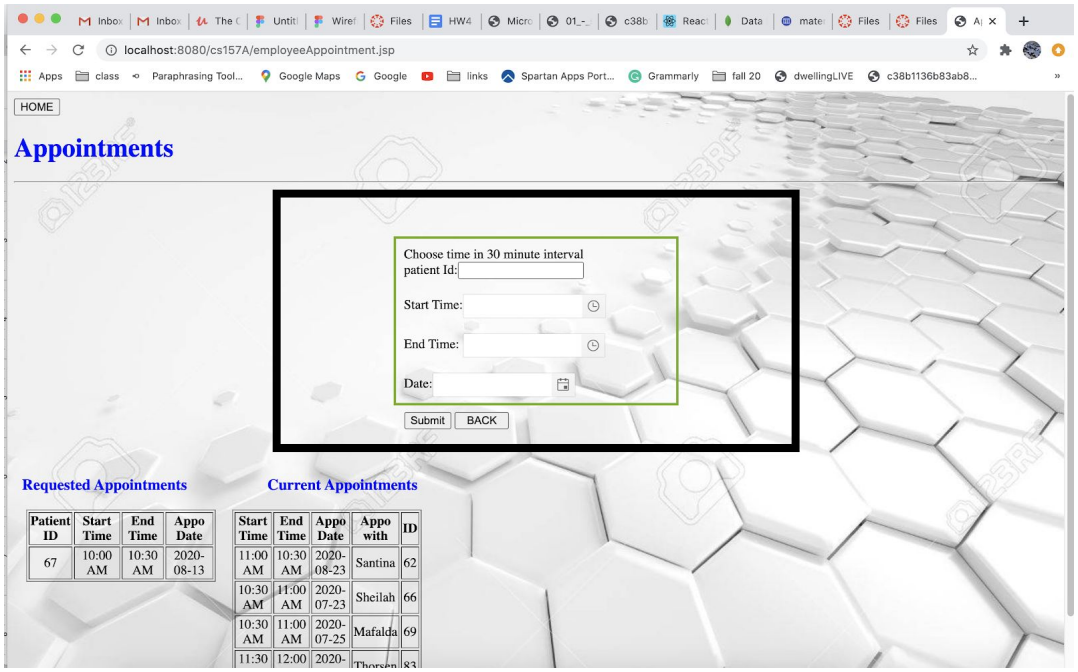
```
try {

    java.sql.Connection con;
    Class.forName("com.mysql.jdbc.Driver");
    con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Hospital?serverTimezone=EST5EDT",user, password);
    Statement stmt = con.createStatement();

    java.util.Date now = new java.util.Date();
    java.sql.Date sqlDate = new java.sql.Date(now.getTime());
    //find the patient id using the patient id
    ResultSet rs = stmt.executeQuery("SELECT * FROM user;");

    while(rs.next()) {
        if(rs.getString(1).equals(patientID) ){
            firstName = rs.getString(2);
            lastName = rs.getString(3);
           break;
        }
    }

    ResultSet rs2 = stmt.executeQuery("SELECT * FROM patient;");

    while(rs2.next()) {
        if(rs2.getString(1).equals(patientID) ){
            diagnosis = rs2.getString(2);
            room = rs2.getString(3);
            admissionDate = rs2.getString(4);
           break;
        }
    }

   ResultSet rs3 = stmt.executeQuery("SELECT * FROM patientMedicalHistory;");

    while(rs3.next()) {
        if(rs3.getString(2).equals(patientID) ){
            medications = rs3.getString(3);
            allergies = rs3.getString(4);
            diseases = rs3.getString(5);
            symptoms = rs3.getString(6);
            familyHistory = rs3.getString(7);
           break;
        }
    }

    stmt.close();
    con.close();
} catch(SQLException e) {
    out.println("SQLException caught: " + e.getMessage());
}
%>
```




```
59  try {
60
61      java.sql.Connection con;
62      Class.forName("com.mysql.jdbc.Driver");
63      con = DriverManager.getConnection("jdbc:mysql://localhost:3306/Hospital?serverTimezone=EST5EDT",user, password);
64      Statement stmt = con.createStatement();
65
66      java.util.Date now = new java.util.Date();
67      java.sql.Date sqlDate = new java.sql.Date(now.getTime());
68      //find the patient id using the patient id
69      ResultSet rs = stmt.executeQuery("SELECT * FROM patientsHasNote;");
70
71      while(rs.next()) {
72          if(rs.getString(1).equals(patientID) ){
73              ArrayList<String> note = new ArrayList<String>();
74              note.add(rs.getString(2));
75              notes.add(note);
76             break;
77          }
78      }
79
80     ResultSet rs2 = stmt.executeQuery("SELECT * FROM note;");
81
82      while(rs2.next()) {
83          for (int i = 0; i < notes.size(); i++) {
84              if(rs2.getString(1).equals(notes.get(i).get(0)) ){
85                  notes.get(i).add(rs2.getString(2));
86                  notes.get(i).add(rs2.getString(3));
87              }
88          }
89      }
90      stmt.close();
91      con.close();
92  } catch(SQLException e) {
93      out.println("SQLException caught: " + e.getMessage());
94  }
95  %>
96
97
98  <div class="notes">
99   <p><strong>Notes: </strong></p>
100  <%for (int i = 0; i < notes.size(); i++) {
101      out.println("Note ID: " + notes.get(i).get(0) + "\n"); %> <br>
102      <%out.println(notes.get(i).get(1) + "\n"); %> <br>
103      <%out.println("Date: " + notes.get(i).get(2) + "\n"); %> <br><br> <%
104  }
105  %>
106 </div>
```

Employee creates appointment

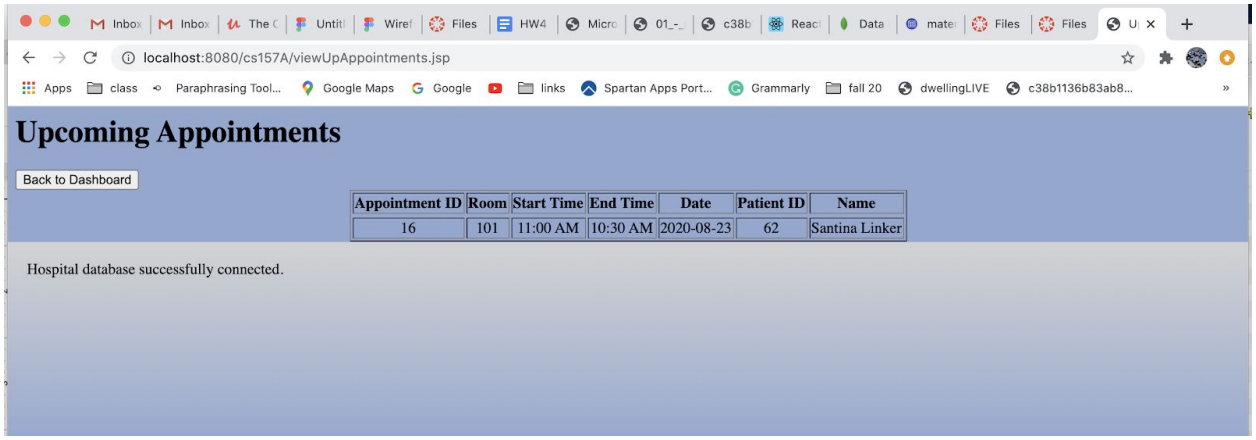
-when a patient requests an appointment it will show on the employee dashboard. Employees see only requested appointments to them. As soon as an employee creates the appointment based on the information based on the requested appointment , the appointment in the requested table will be removed or deleted by itself.



Patient ID	Start Time	End Time	Appo Date
67	10:00 AM	10:30 AM	2020-08-13

Start Time	End Time	Appo Date	Appo with	ID
11:00 AM	10:30 AM	2020-08-23	Santina	62
10:30 AM	11:00 AM	2020-07-23	Sheilah	66
10:30 AM	11:00 AM	2020-07-25	Mafalda	69
11:30	12:00	2020-	Thorsen	83

Employee can see the upcoming appointment by clicking on the button “view upcoming appointment”



Lesson Learned

Girma Jembere

From this project, I learned that SQL is one of the most ***important*** skills to learn for any programmer. I learned developing database applications using DBMS and the SQL language, Effectively use the Entity-Relationship Diagram for the representation of conceptual schemas, identify functional dependencies and apply normalization process, construct data retrieval procedures using the Data Manipulation Language and data definition language. In addition, most importantly it helps me to visualize and get a full understanding of how any application works. I also learned how SQL as a back end will integrate other languages and have a big impact on the performance of any application. a good Query and effective Entity-Relationship will help to get a good performance on a large scale database application.

Jade Webb

This project was very crucial in solidifying my understanding of the usefulness of DBMS in everyday applications. This project allowed me to learn a lot from using MySQL, Eclipse, and Apache Tomcat to produce a working web application based on a database. There were some struggles when I was installing MySQL and Apache Tomcat on my device, but I learned a lot from finally resolving the issues. I had never worked with an application like Apache Tomcat, so my time spent learning how it works with a web application was very beneficial. From this project, I learned SQL, HTML, CSS, and Javascript, which I had not known prior to taking this course. This was my first time working with .jsp files, so there was a learning curve but I gained a lot of knowledge and experience through configuration of Eclipse and writing the code within. I had also never used GitHub before, so it was a good experience learning how to use it in a group project setting. I learned how to create branches, push files, and merge our code together. I am very proud of the hard work I contributed to this project and I am proud of how our project came out.

Arlan Prado

Before this project, I only knew how to use MySQL using PHP. I previously did not know about .jsp files and how it integrates Java into websites. I only thought that Java could be used for applications that are app-based, not web-based. I understood how GitHub works but I haven't worked on a group project where everyone contributes equally.

Through this project, I have learned how to work with a team as a leader, coding using jsp files, and extensively using GitHub. With GitHub I learned that merging branches and joining code together is very tedious and confusing because all the potential differences it could have: deletions, additions, and modifications. It takes a while in that the merge must be perfect or else there might be a roll back to the previous version. While it was difficult, I acknowledge learning how to merge branches together in a project is essential to working in projects. Learning .jsp wasn't hard as it mostly was Java and MySQL exercise. HTML and CSS was still newer to me so I had to look that up how to organize certain things, especially the CSS. I have worked in large group projects of 3, but the situation was different in that the project allowed for merging easier because parts of the code didn't interact with each other. In this situation, integration is especially important, requiring more communication and meetings than my previous project.