MedProvider

Team #8



Project Report

By

Yonatan Greenblum

Mashawn Hall

Suyeob Kim

Table of Contents

**Project Requirements:**

Project Description ……………………………………………………………………3

System Environment ………………………………………………………………….4

Hardware/Software …………………………………………………………...4

RDBMS ………………………………………………………………………4

Application Languages ……………………………………………………….4

Functional Requirements ……………………………………………………………..5

Non-functional Requirements ……………………………………………………….. 6

Graphic User Interface ………………………………………………………. 6

Security ……………………………………………………………………….6

Access Control ………………………………………………………………..6

**Project Design Section:**

ERD…………………………………………………………………………………...7

Schemas……………………………………………………………………………….7

Tables………………………………………………………………………………….9

Screenshots……………………………………………………………………………9

**Project Implementation Section:**

Login/Sign up…………………………………………………………………….…….18

Patient functions…………………………………………………………………….….22

Physician functions………………………………………………………….………….28

Admin functions………………………………………………………………………...31

Project Setup……………………………………………………………………………37

**Lessons Learned**………………………………………………………………………………..38

Project Requirements

Project Description

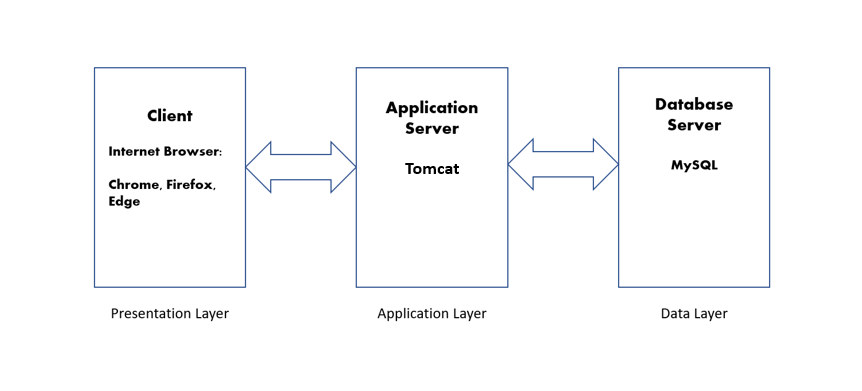
MedProvider is a health management application designed to handle messages, prescriptions, and appointments between patients and their physicians. MedProvider will have separate logins for patients, physicians, and admins for each to view their essential information.

Users will be able to login and see their list of medications, send messages to their physician, and see upcoming appointments. Physicians will have access to all their patient’s records, medications, and messages. Admins will be able to see how many appointments have been scheduled by a physician and how many prescriptions each physician has prescribed.

The goal of MedProvider is to provide a user-friendly application for patients, physicians, and administrators to better manage the healthcare of patients and make patients feel at ease knowing their health is being cared for.

The stakeholders of this project include team members (Yonatan Greenblum, Mashawn Hall, and Suyeob Kim), project advisor (Professor Wu), and virtually any patient and physician who need a healthcare management system. The main target audiences for MedProvider are both patients and physicians which both need to sign up for the application to function. Because MedProvider will handle very discreet and intimate information of our users, MedProvider will apply high level software security, minimum downtime, and functional software to continue gaining the trust of our users.

System Environment



Hardware and Software used:

* Tomcat web server
* Mac/PC

RDBMS:

* MySQL

Application Languages

* HTML/CSS
* PHP, JavaScript
* SQL

Functional Requirements

* Users of the system:
* Patients – people interested in enhancing their experience with their physician and help them manage their health.
* Physicians- physicians who are interested in receiving help to better serve their patients and help them manage their relationships with them.
* Administrators – Help health administrators in tracking their physician’s workload and time.
* How to access the system:
  + Users of the system will need access to a PC with internet capabilities. Patients, physicians, and administrators will have separate logins to the website. First time users will need to sign up first.
* Describe each functionality/features, functional processes, and I/O(s).
  + Patient related functionalities
    1. Home Page
       1. After a patient logs in to their account they will be able to navigate to other sections such as messages, medications, and appointments.
    2. Messages
       1. Patients will be able to send messages to any of their physicians by clicking on the messages tab and filling in a text area with their message.
       2. Patients will be able to see messages from their physicians.
    3. Medications
       1. Patients will be able to see a complete list of their medications including amounts and how often they should be taken.
    4. Appointments
       1. Patients will be able to see their scheduled available appointments.
  + Physician related functionalities
    1. Home Page
       1. After a physician logs into their account, they will be able to navigate to other sections such as messages, prescribe medications and appointments.
    2. Messages
       1. Physicians will be able to send messages to any of their patients by clicking on the messages tab and filling in a text area with their message.
       2. Physicians will be able to see messages from their patients.
    3. Prescribe medications
       1. Physicians will be able to prescribe medications to their patients and will be added to medication list of the patients.
    4. Schedule appointments with patients.
       1. Physicians will be able to schedule appointments with their patients.
  + Administrator related activities
    1. Home Page
       1. After an administrator logs in he will be able to navigate to sections All appointments and All prescriptions.(Overview)

Non-functional Issues

Graphic User Interface

The graphic user interface (GUI) of MedProvider will showcase user-friendliness and a sleek design. The GUI will be compromised of HTML and CSS languages. We will use universal images and icons to help new users operate the application and will be comprised with aesthetic design choices.

Security

Accounts will only be accessible by entering a corresponding username and password combination. To ensure security, the UI will not present any SQL errors to the user and passwords inserted to the UI will be hidden.

Access Control

From a hierarchical perspective, the admins have the highest level of

access to the information held within the system. Admins will be able to see all

physician and patient information that will help them make company decisions.

This information will be artifacts like physicians’ basic information (i.e., name, age, and gender), title and appointment information. And regarding patients,

admins will have access to their basic information as well as their appointment

information. Admins will not have access to patients’ medical information. The

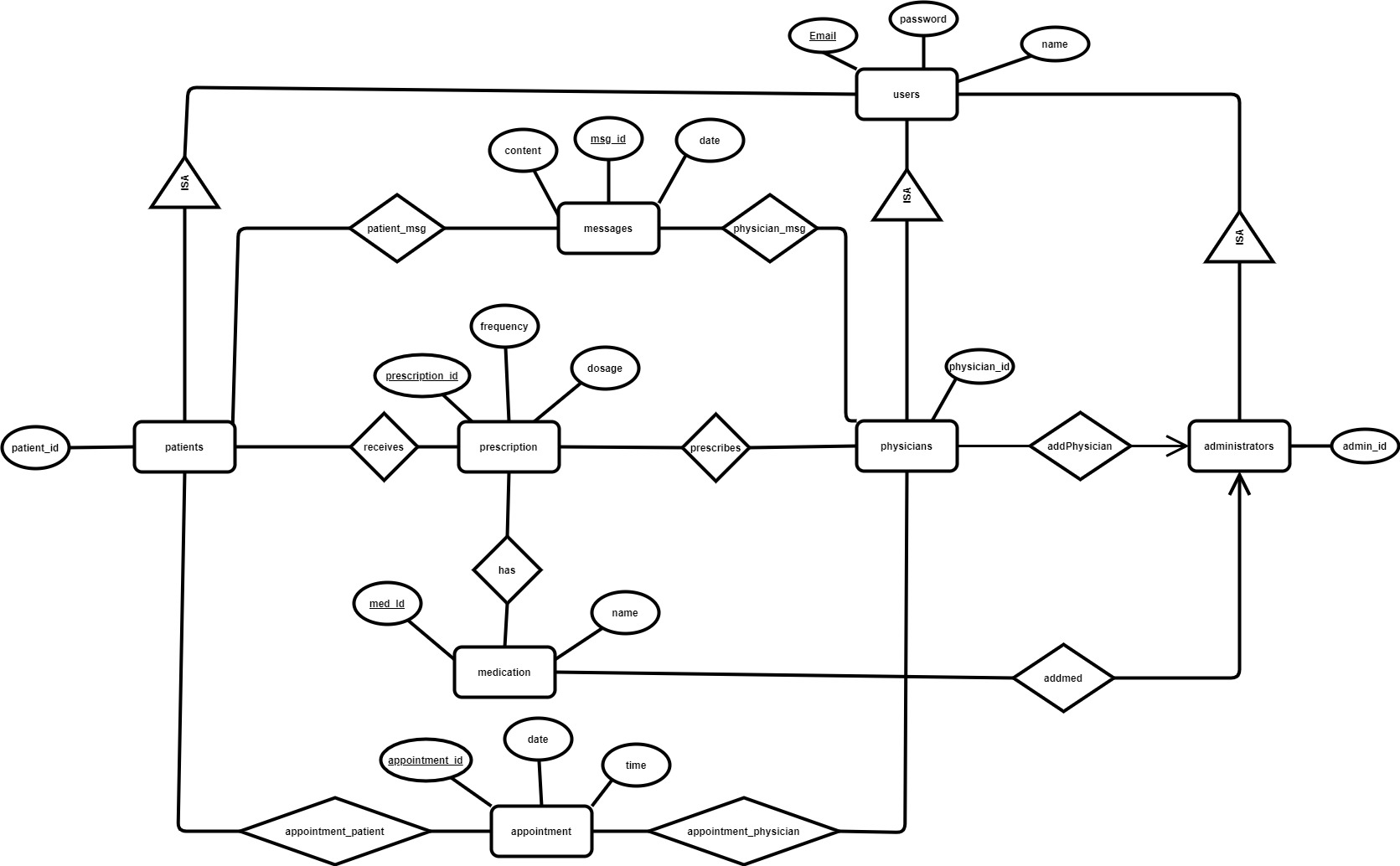
next level of access are the physicians which will only have access to their own

personal information, their appointment information, and the medical information

of their patients. Lastly, patients will only have access to their personal/medical

information and their own appointment information.

Entity-Relationship Diagram



Schemas

users(Email,password,name) – The user entity has a primary key of Email and attributes of password and name. Anyone who creates an account will be added to the database.

patients(Emailpatient,patient\_id) – patients is a subclass of user and therefore has the primary key from user entity. Patients also have a patient\_id.

physicians(Emailphysician,physician\_id) - physicians is a subclass of user and therefore has the primary key from user entity. Physicians also have a physician\_id.

administrators(Emailadmin,admin\_id) - administrators is a subclass of user and therefore has the primary key from user entity. administrators also have an admin\_id.

messages(msg\_id,content,date) – an entity which holds the messages between Physicians and patients. Msg\_id is the key. Content is the message itself and date is the date.

Physician\_msg(msg\_id\_phys,phys\_email\_msg) a relation between physicians and messages entities

Patient\_msg(msg\_id\_phys,patient\_email\_msg) a relation between patients and messages entities

prescription(prescriptionid,frequency,dosage) – a relation between physician and patient and medication which has all keys from physician, patient and medication and has attributes frequency and dosage. A physician prescribes medication to patients.

prescribes(prescriptionid, Emailphysician) a relation between prescriptions and physician entities

receives(prescriptionid,Emailpatient) a relation between prescriptions and patient entities

has(prescriptionid, med\_id) a relation between prescriptions and medication entities

appointment(appointment\_id,date,time) - an entity which holds the apppointment\_id between physicians and patients. The physician sets an appointment time for the patient and vice versa.

Appointment\_physician(physicianemail\_app, physician\_app\_id) a relation between physician and appointment entities.

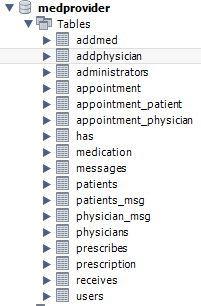
Appointment\_patient(patientemail\_app, patient\_app\_id) a relation between patient and appointment entities.

medication(med\_id,name) – the medication entity is a list of medication id which is the primary key and the name of the medication.

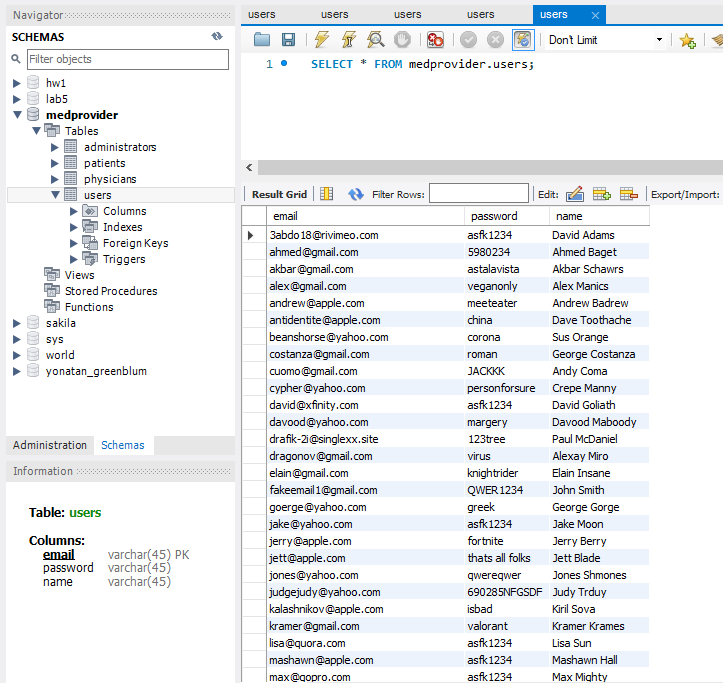
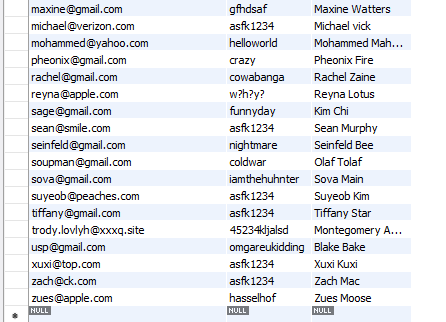
addmed(Emailadmin\_addmed,med\_id\_addmed) – a relation between administrator and medication entities. An admin will be able to add multiple medications to the database.

addPhysician(Emailadmin\_addphys, Emailphysician\_addphys) – a relation between administrator and physician entities. An administrator can add physician to the database. Multiple administrators can add any physician.

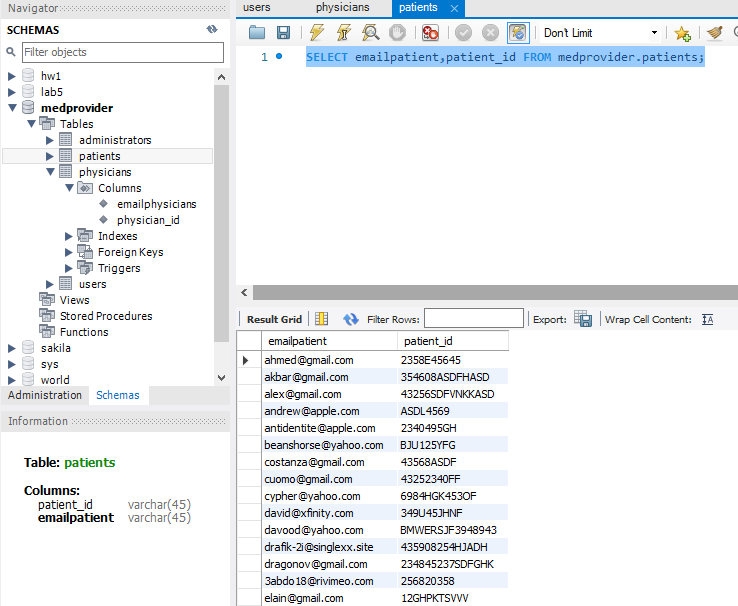
Tables



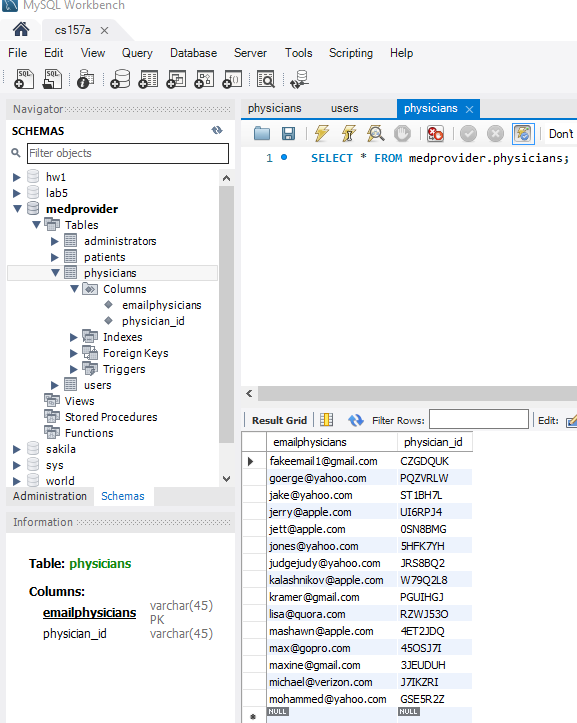
user: (has 45 entries as it will have 3 subclass tables with 15 entries in each)

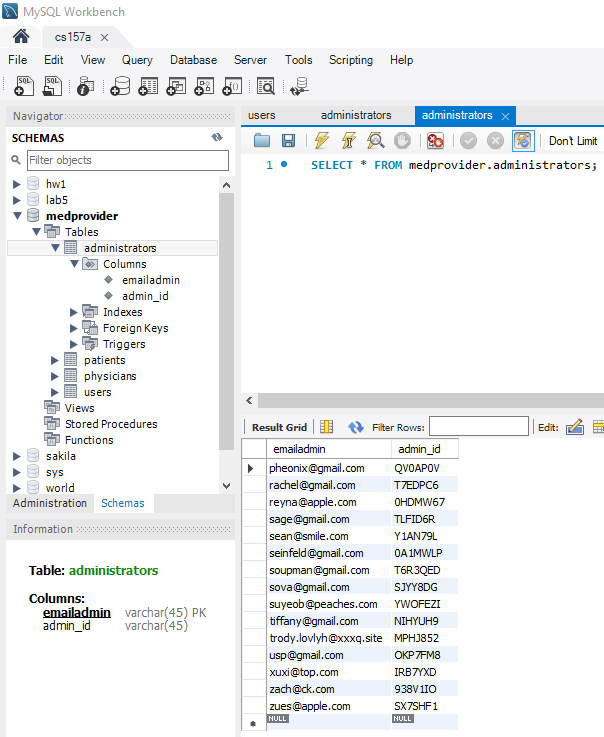
patients: (has 15 entries which also appear in the user table)



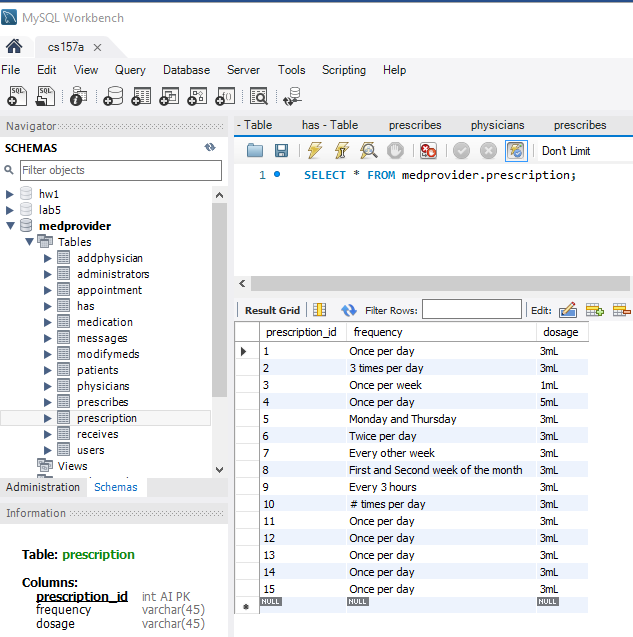
physicians:



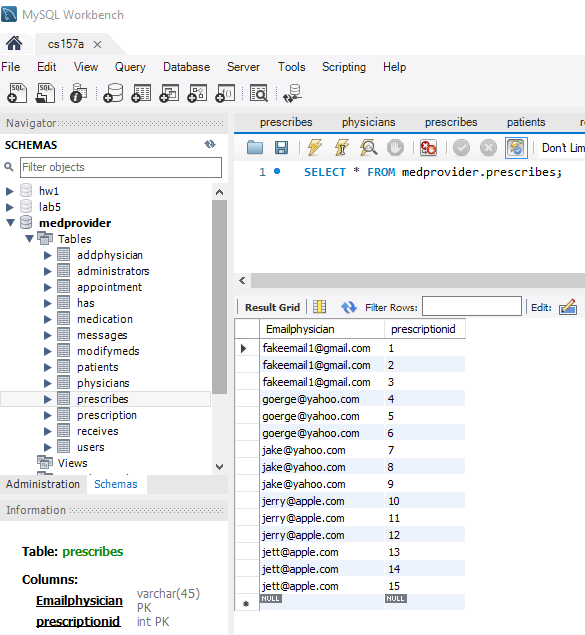
administrators:



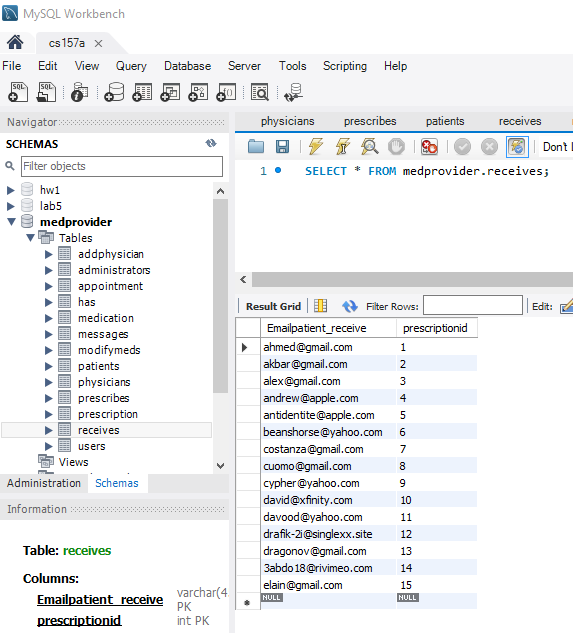
prescription:



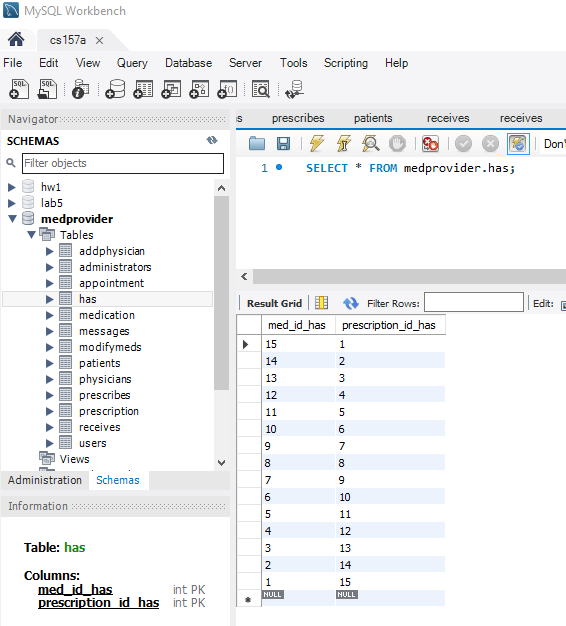
prescribes:



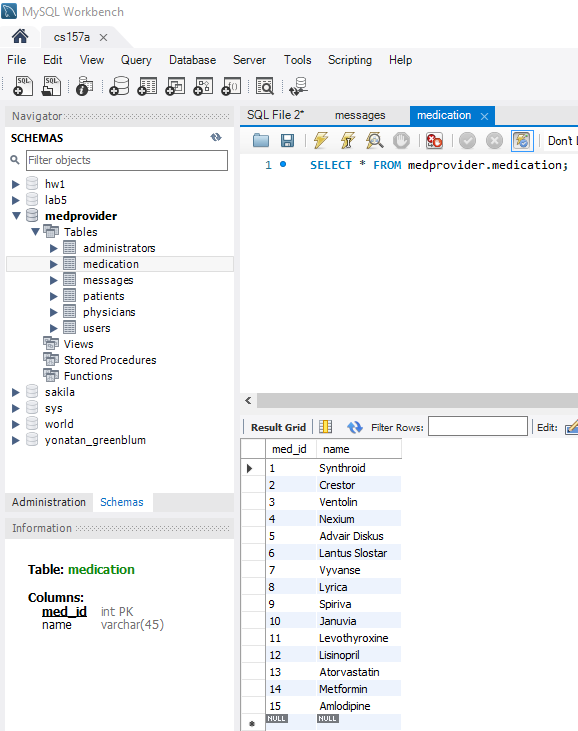
receives:



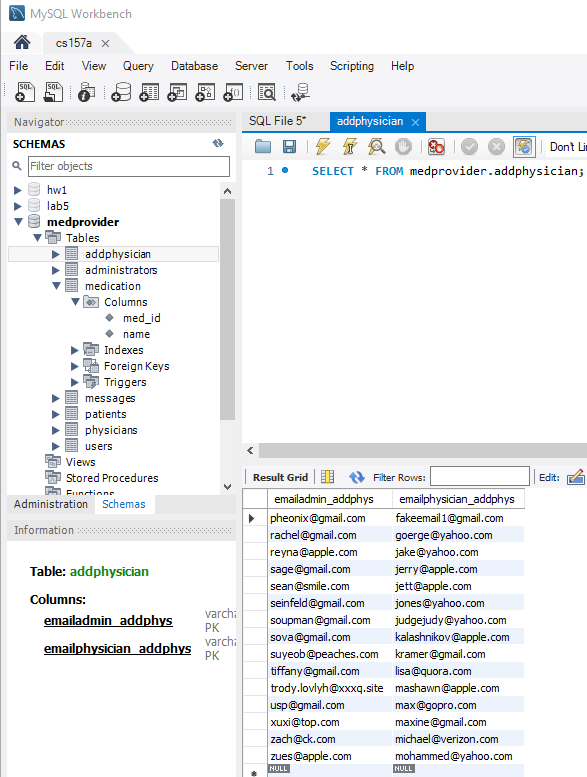
has:



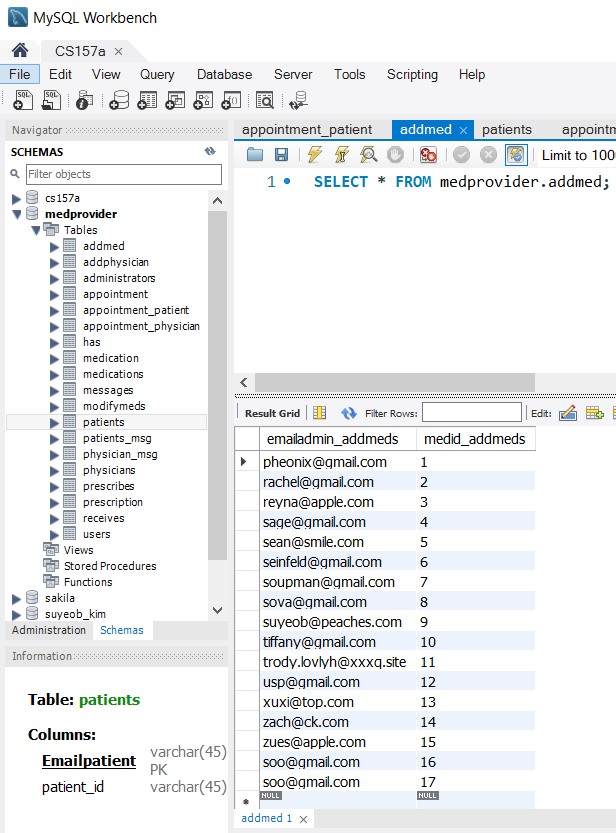
medication:



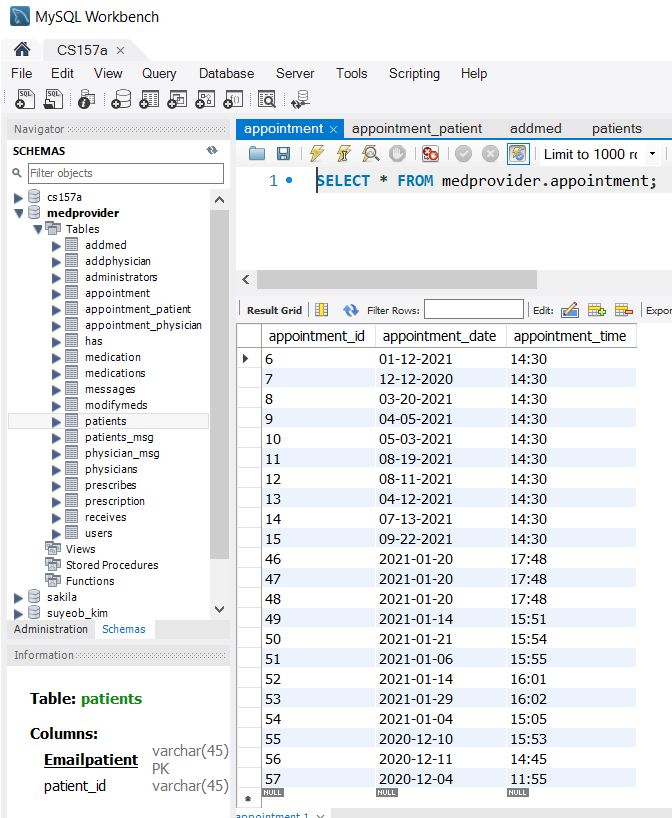
addphysician:



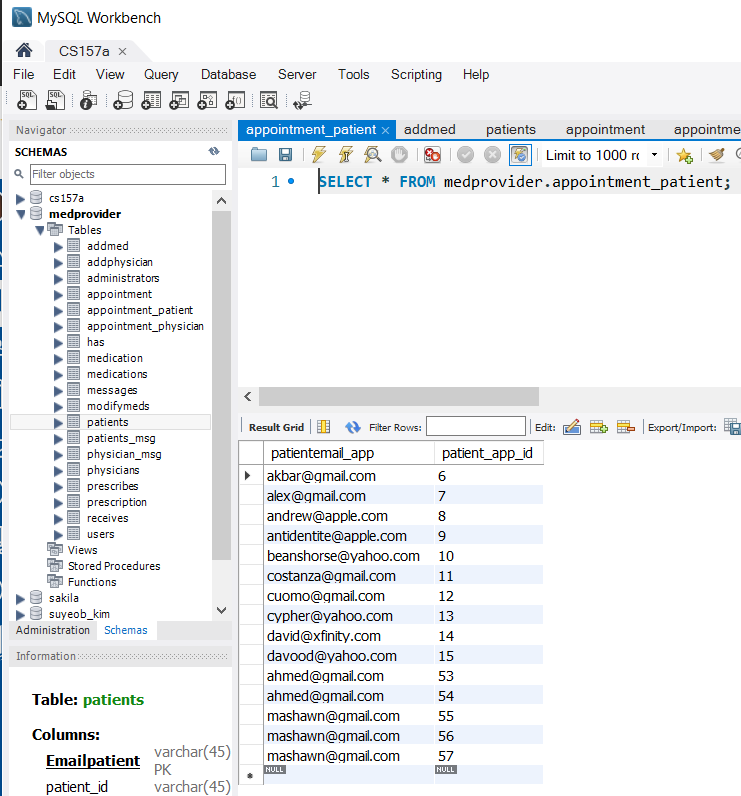
addmed:



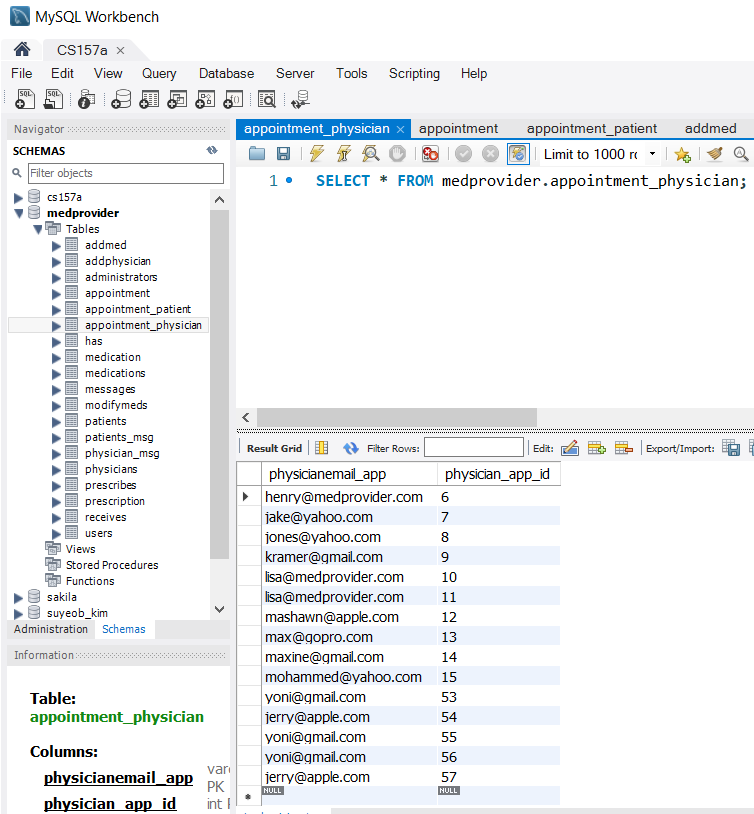
appointment:



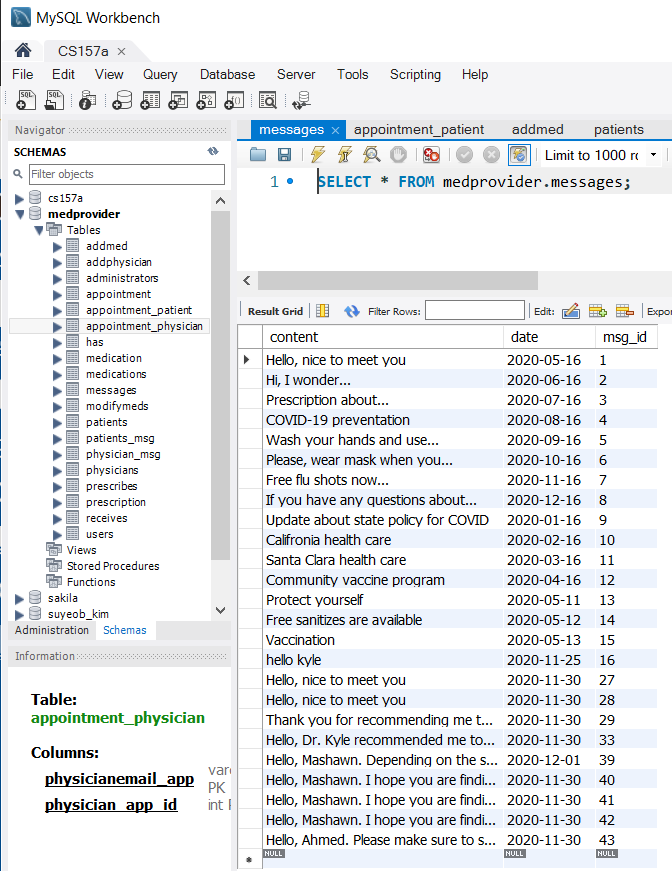
appointment\_patient:



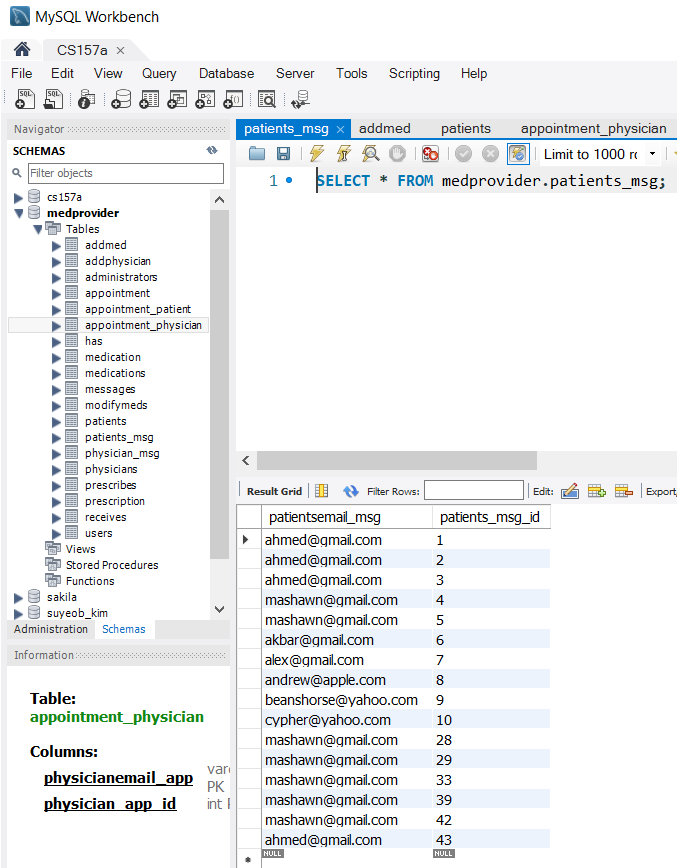
appointment\_physician:



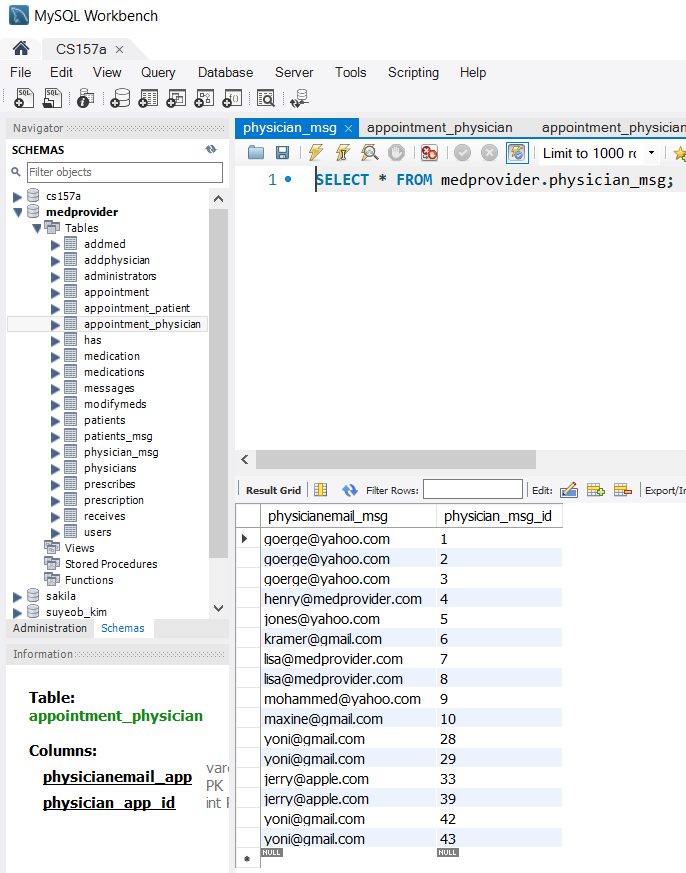
messages:



patient\_message:



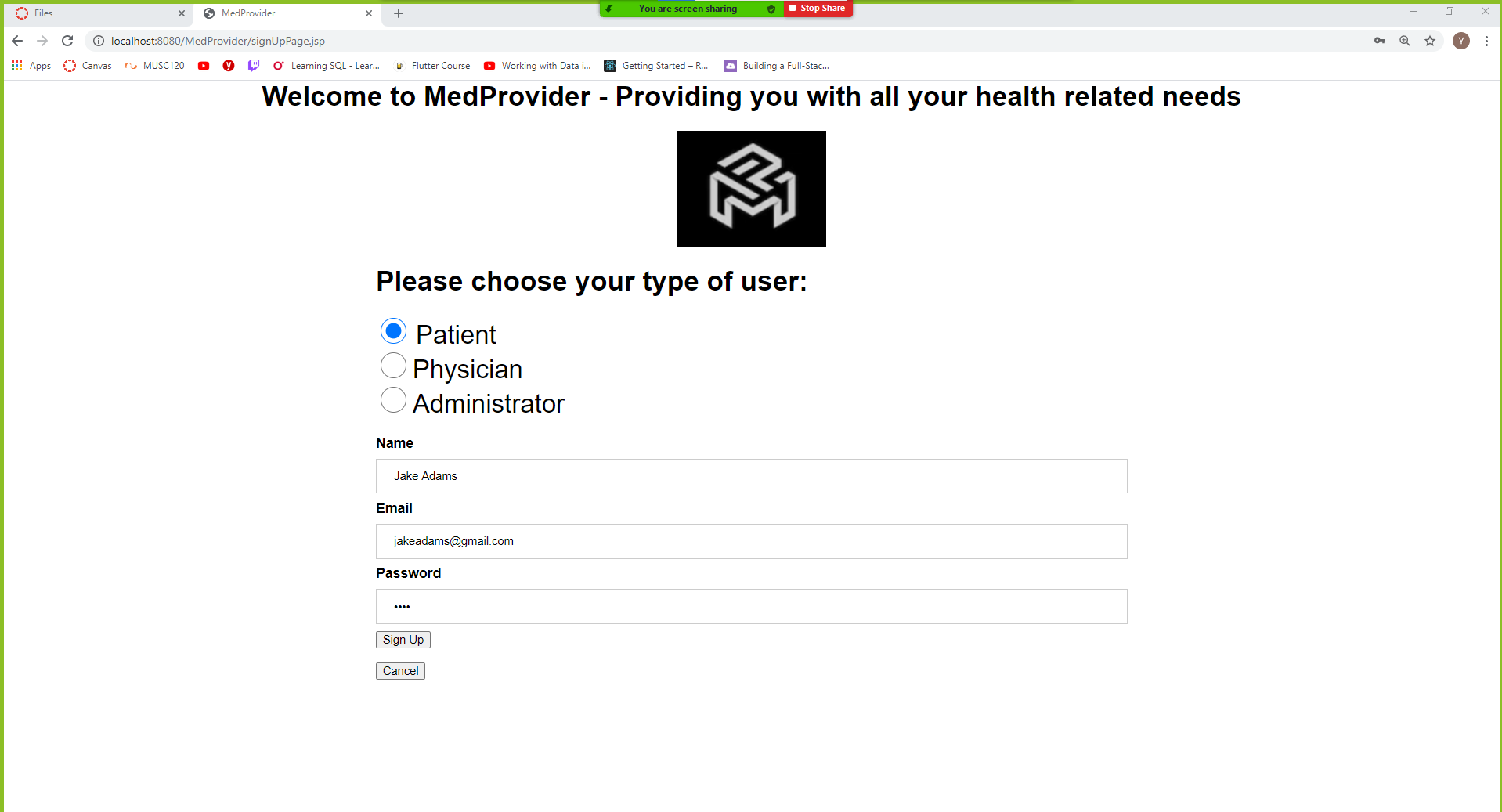
physician\_message:



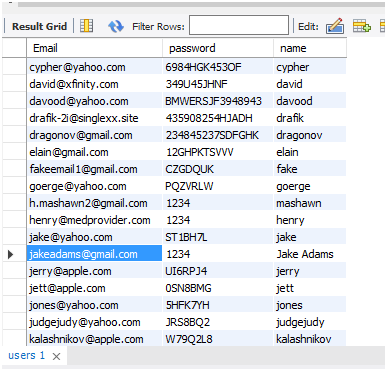
**Project Implementation Section:**

**Signup function** -

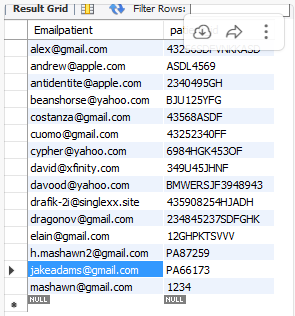
Signup with Name: Jake Adams, Email: [jakeadams@gmail.com](mailto:jakeadams@gmail.com), password: 1234 as a Patient.



SQL Workbench inserted into users table:



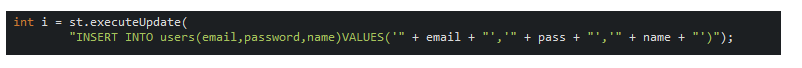
SQL Workbench inserted into patients table:



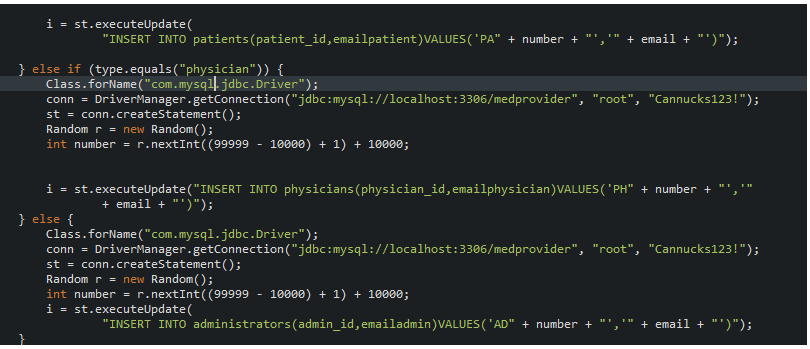
The query for checking if you are trying to create a user that already exists is:



And if this returns not zero. Then the insert query is executed:

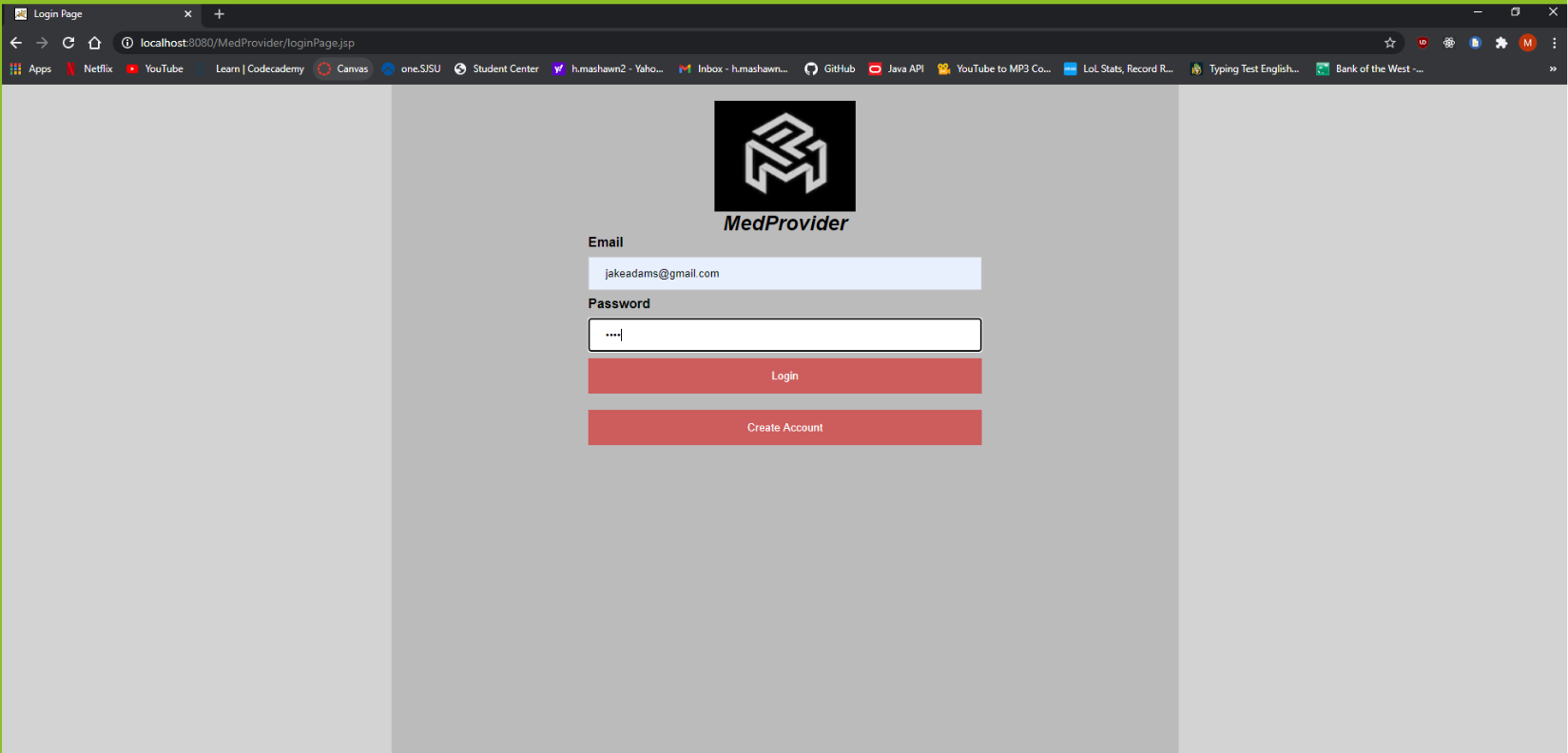


Based on the button chosen when creating the account – this query is executed:

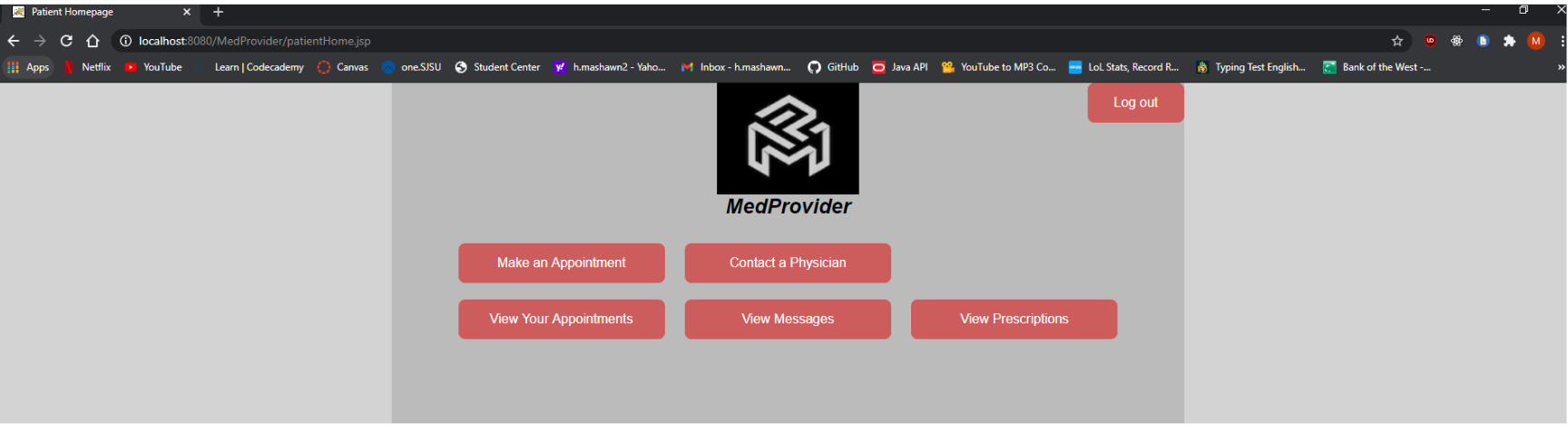


**Login function** -

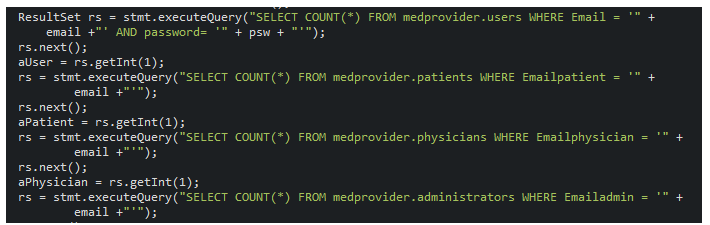
Login with same username and password from sign up



Which then if done correctly takes you to this page: (the patient home page)



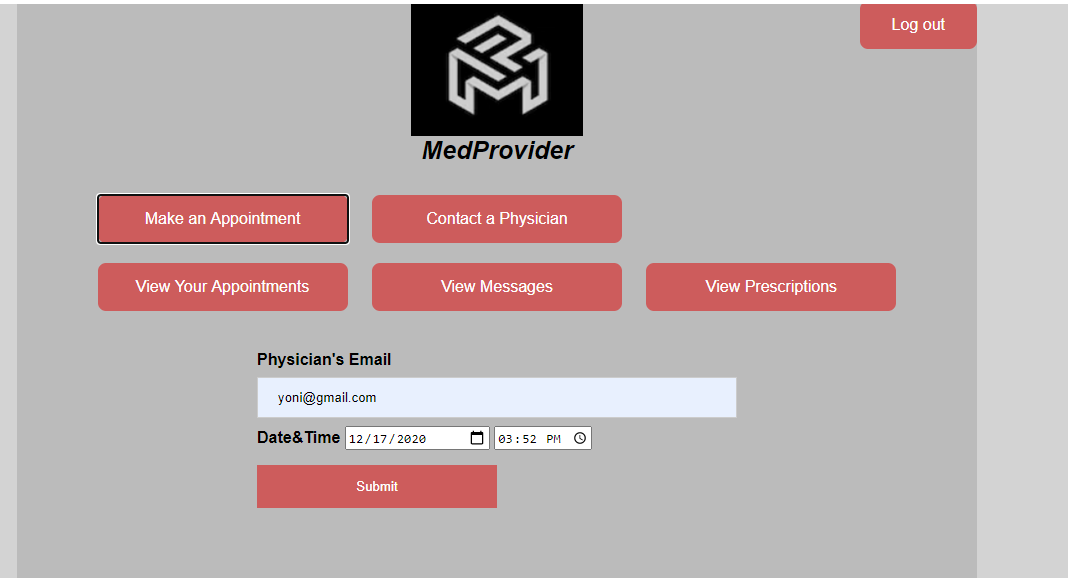
This query is the check to make sure that a certain user is logging in based on the type they are being either a patient/physician/admin.



**Patient Related Functions** -

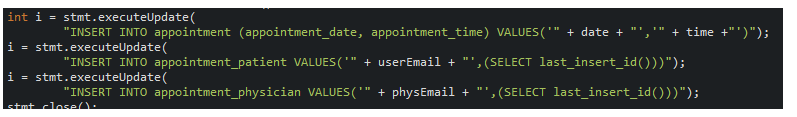
**Create appointment function** – we are signed in as Jake Adams with the email [jakeadams@gmail.com](mailto:jakeadams@gmail.com)

This appointment is for the date 12/17/2020 at 3:52PM.



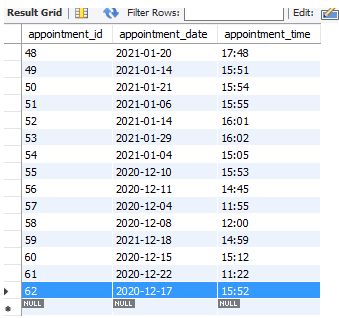
The check for physician email is the same check we used beforehand.

The insertion query for appointment:

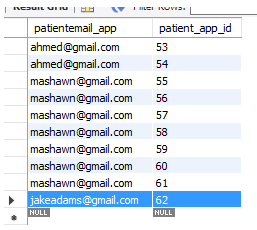


The tables that are updated are the appointment table,

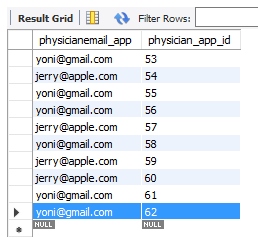
Appointment table:



Appointment\_patient table:

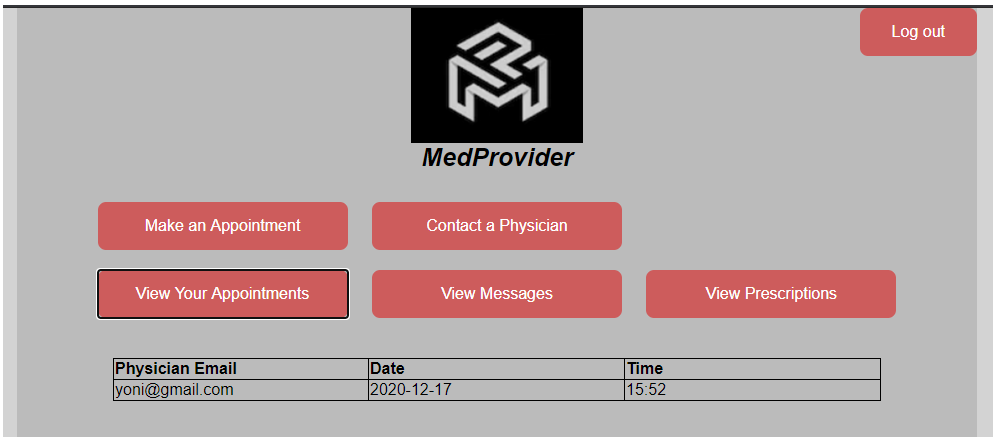


Appointment\_physician table:

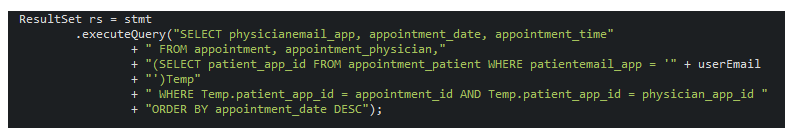


**View appointment function** –

Signed in as jakeadams@gmail.com



The query for viewing is:



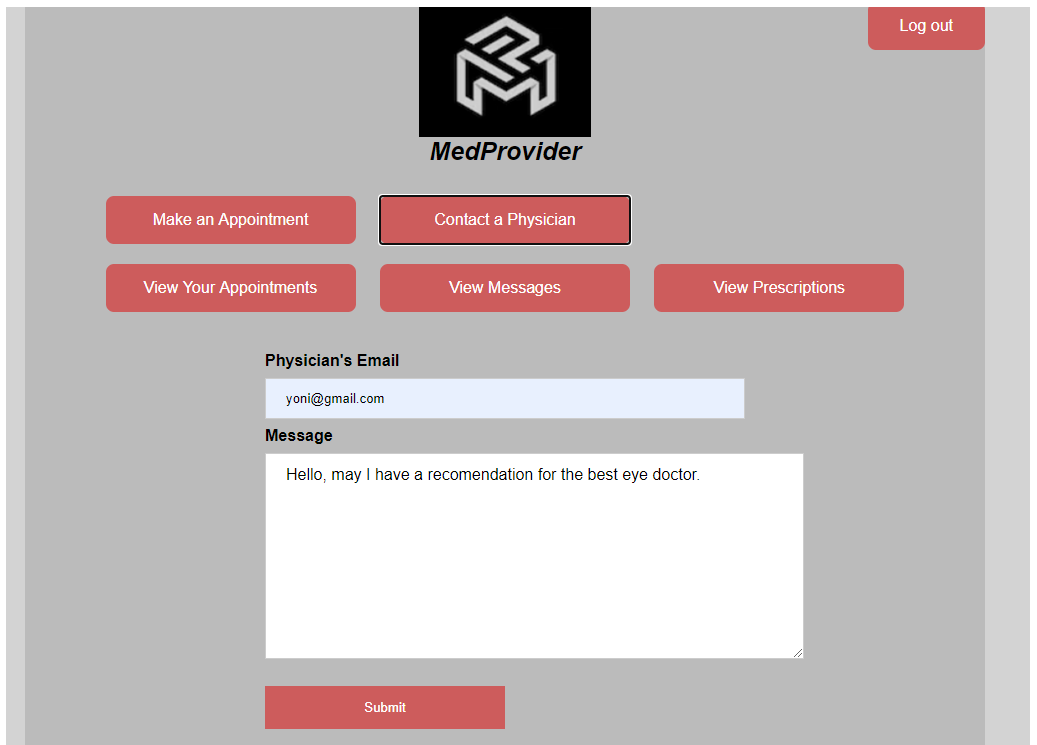
(userEmail is the email we are currently signed in with)

Result in sql table is:

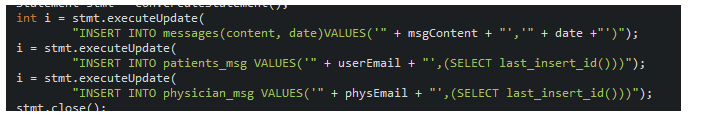


**Message a Physician function –**

(using the same user jakeadams@gmail.com)

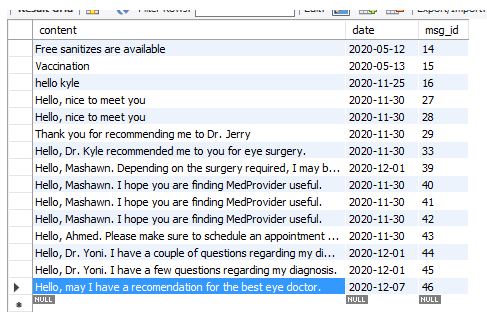
****

Insert query –

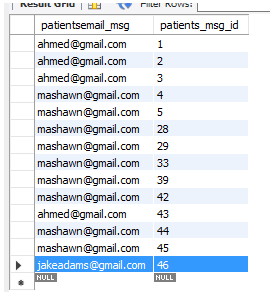


SQL tables that are affected:

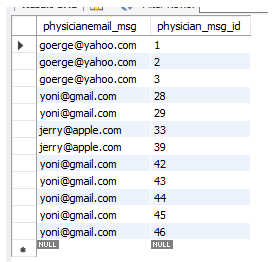
Messages table –



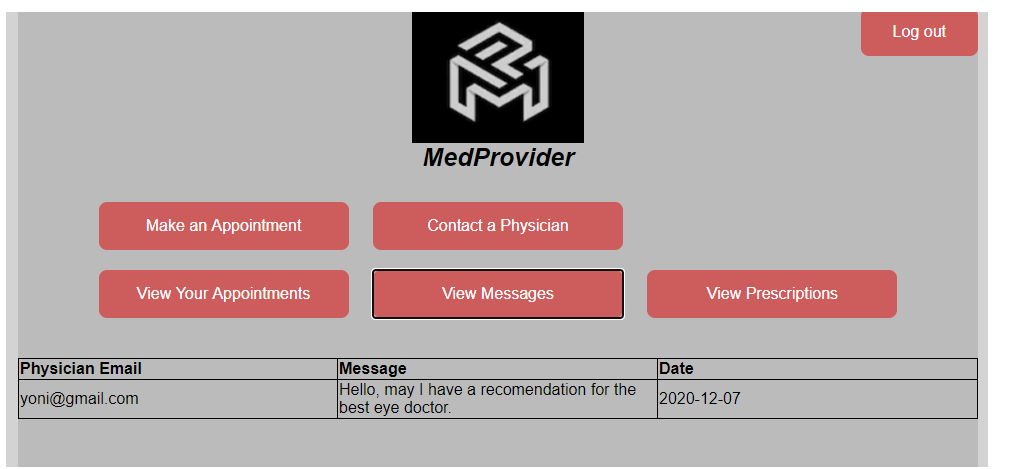
Patients\_msg table:

****

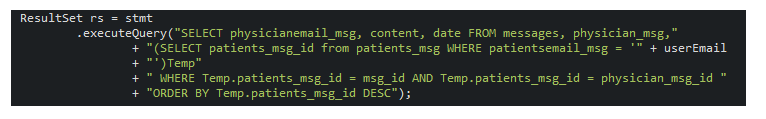
Physician\_msg table:



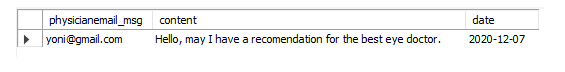
**View messages function -**



Query to view all the messages –



Query result from Mysql workbench –

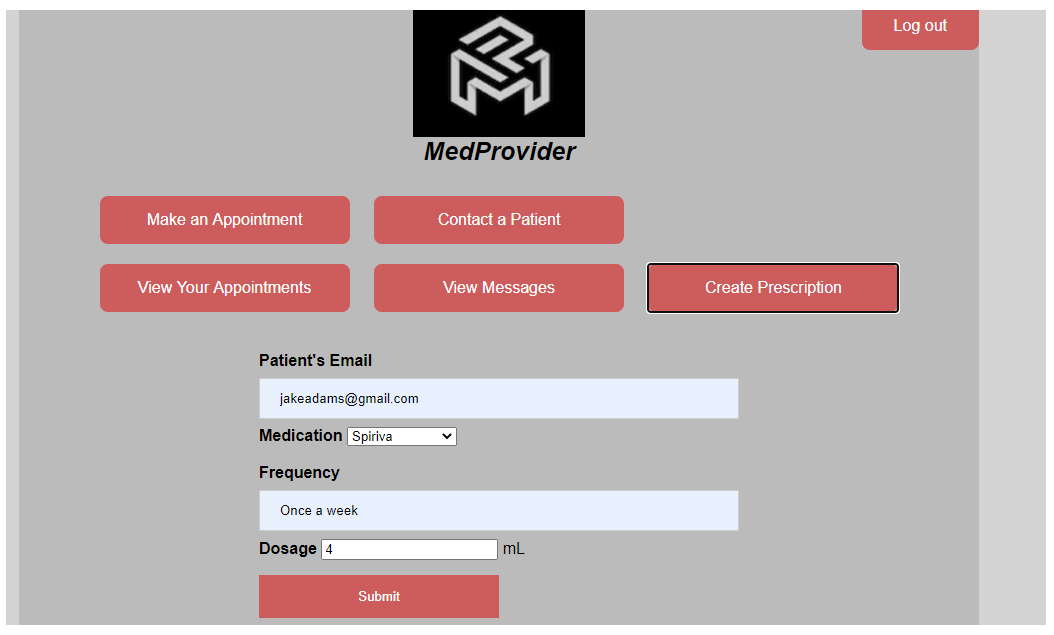


Now we will show the create prescription function as a physician for this user and then return to view prescription inside the patient home page.

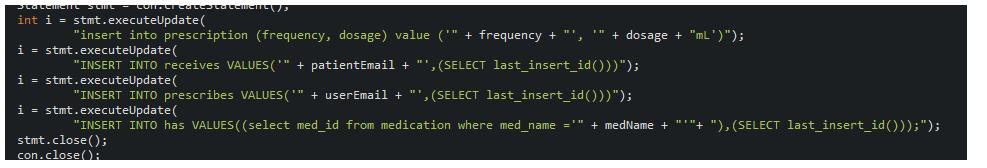
**Physician related functions:**

**Create prescription function –**

Using [jakeadams@gmail.com](mailto:jakeadams@gmail.com) as the patient choosing Spiriva as the medication and frequencey as Once a week with Dosage value of 4.

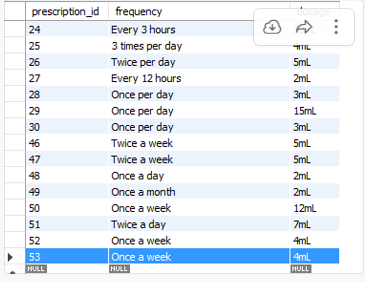


Inserting a prescription to the database:

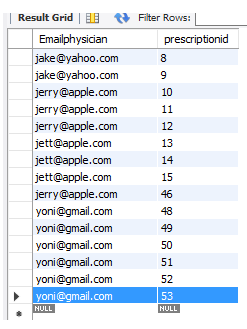


Tables affected:

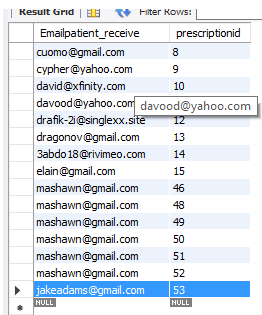
Prescription table:



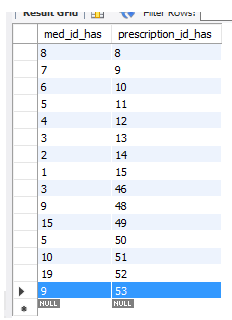
Prescribes table:



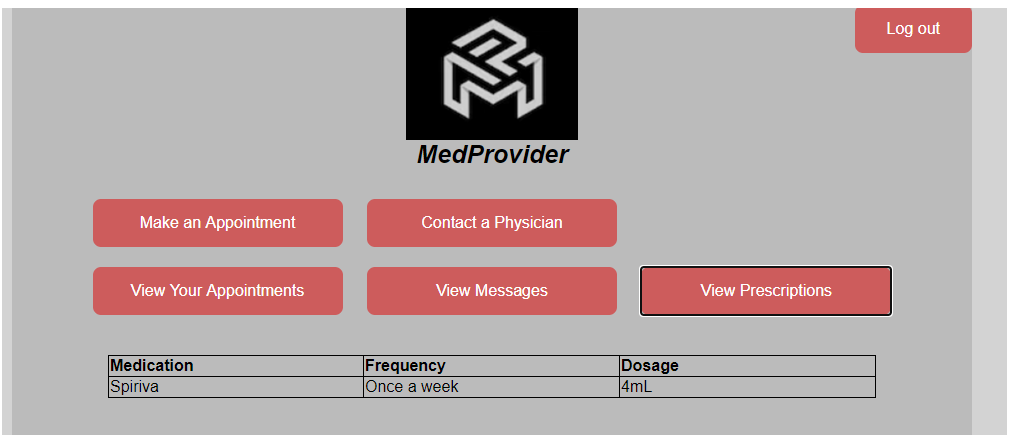
Receives table:



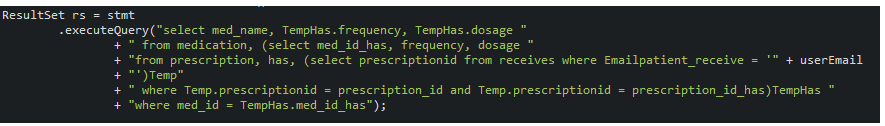
Has table:



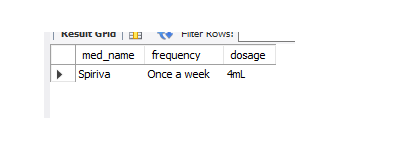
**View Prescription function –**



Query to view prescriptions:

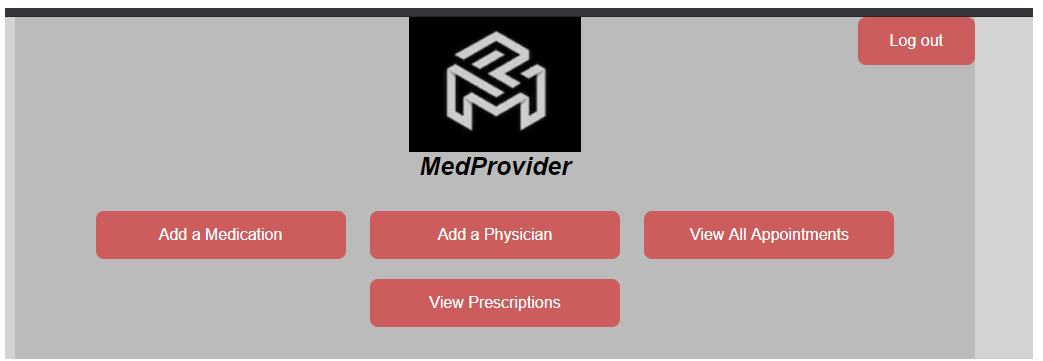


Result in sql workbench table:

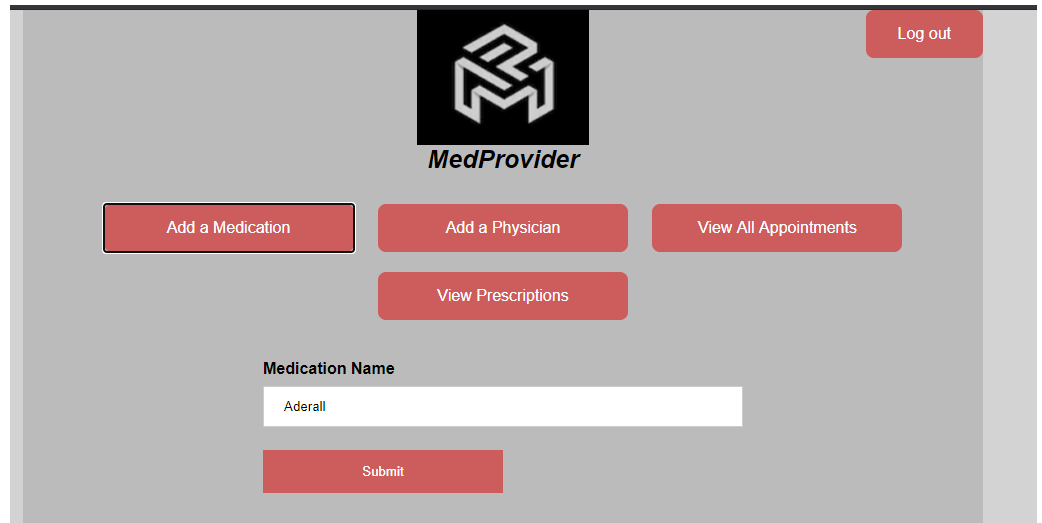


**The appointment/message/view appointment/view messages functions in the physician home page follow the same structure and logic in the patient home page. The only new function in the physician home page is create prescriptions which is detailed above.**

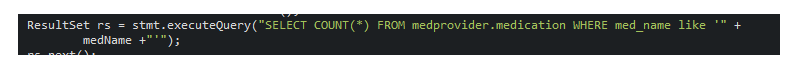
**Admin related functions:**

****

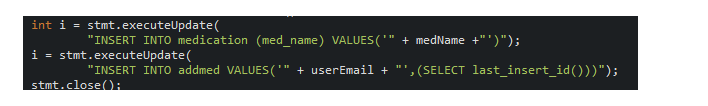
**Add medication function –** adding in Aderall to the database.

****

Query to test if the medication is already in the system:



Query to insert:

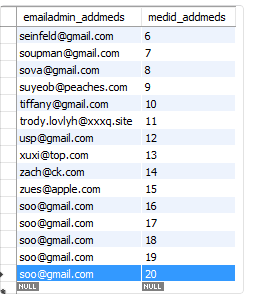


Tables affected:

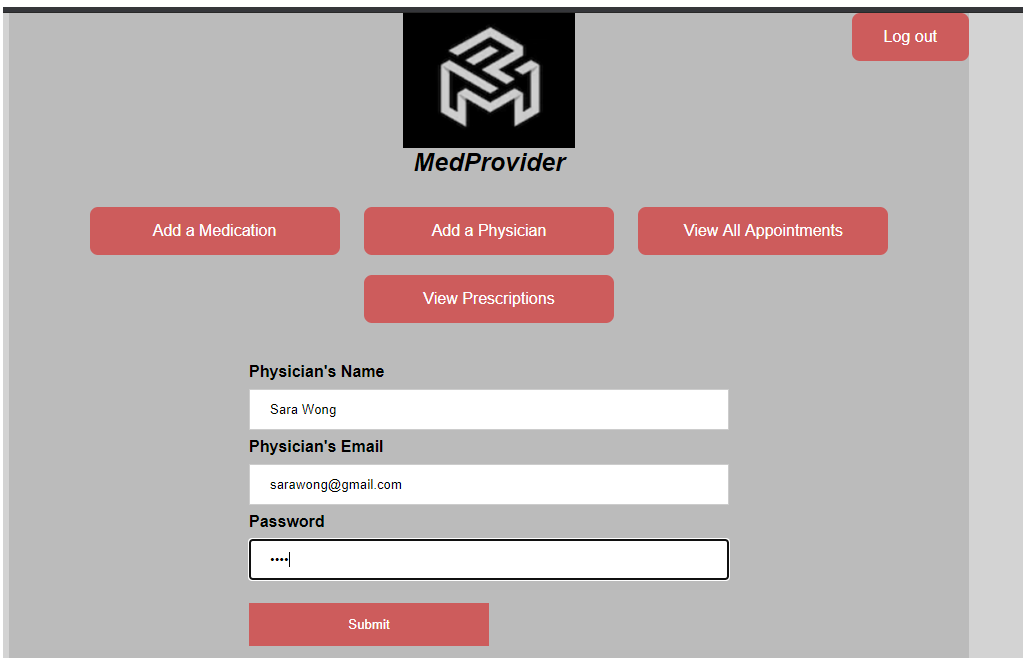
Medication table:



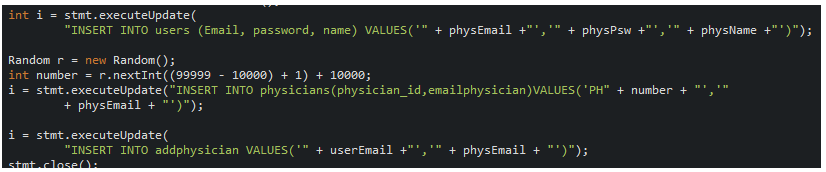
Add\_med table:

****

**Add physician function –** name: Sara Wong, email: [sarawong@gmail.com](mailto:sarawong@gmail.com), password 1234

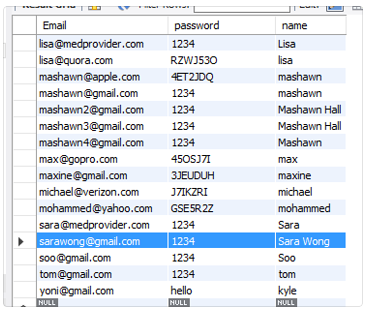


First we make a query to make sure the added physician is not already in the system:

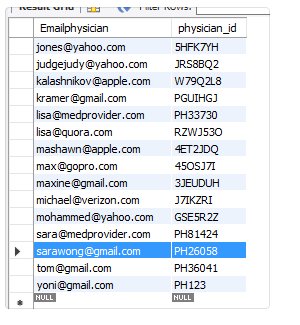


Tables affected:

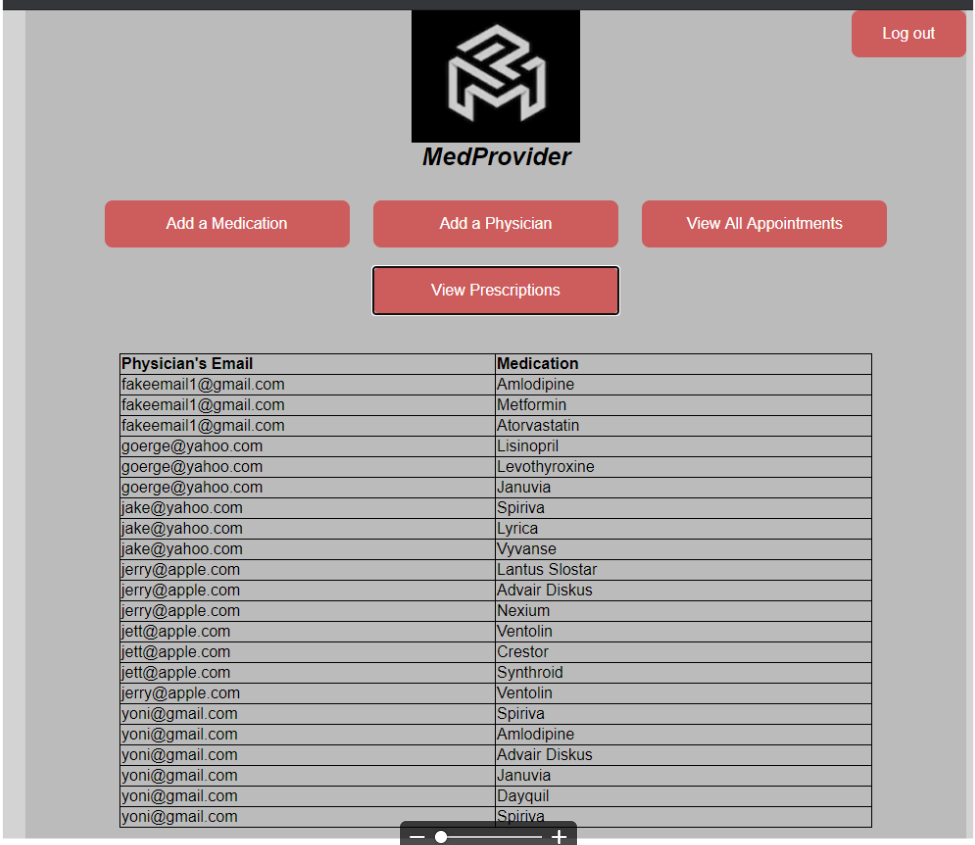
Users table:



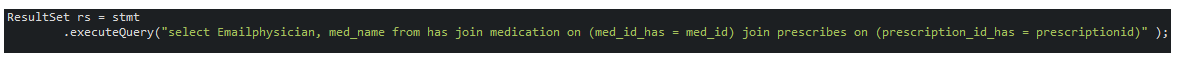
Physicians table:



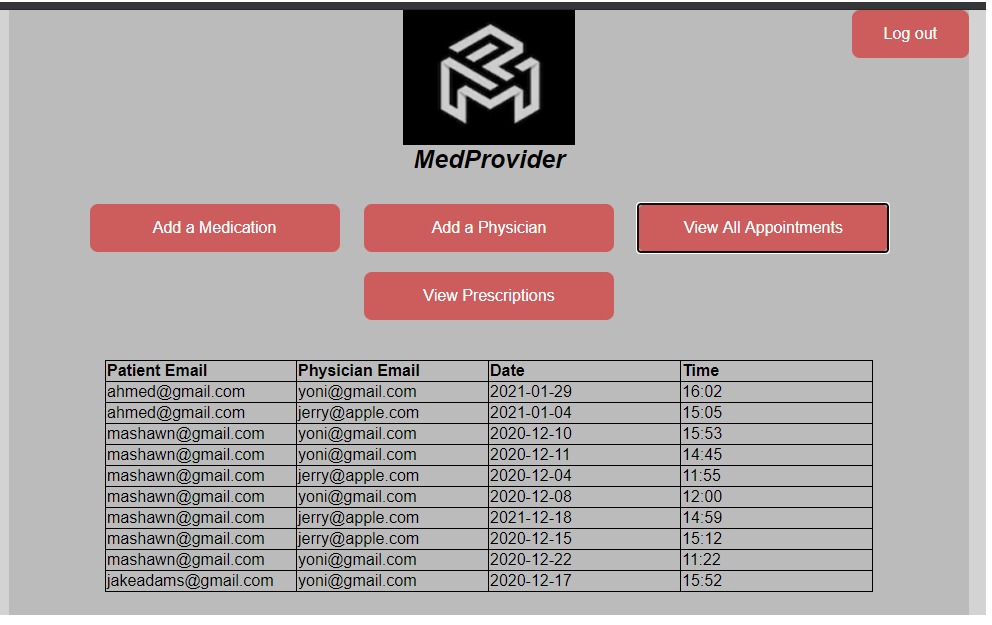
**View all prescriptions function –**

****

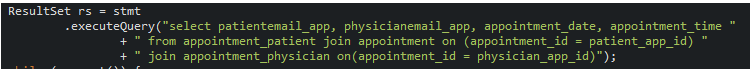
SQL query:



**View all appointments function –**

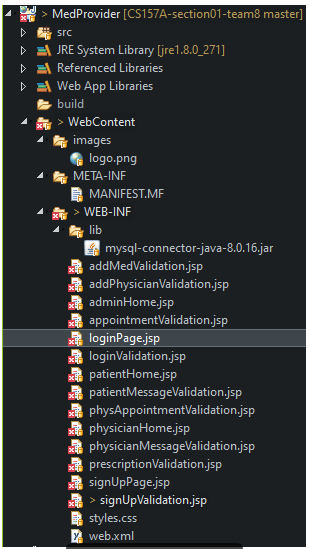


SQL query:



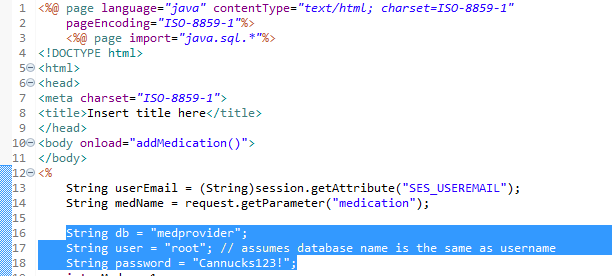
**Project Setup:**

Import the project files to eclipse. The file structure should be as follows:



To run the application correctly on your system you need to change the username and password credentials based on your MySQL workbench in many of the files.

For example, in the addMedValidation.jsp file, you need to change the following marked in blue:



The part highlighted in blue should be your user and the password for that user on your system.

This process needs to be completed for every page and in some pages needs to be completed multiple times. This process does not need to be completed on loginPage.jsp, signUpPage.jsp, styles.css

To run the project, you need to right click on the loginPage.jsp file and choose Run As and the choose Run on Server. (Tomcat Server)

**Lessons Learned:**

Yonatan Greenblum:

This project has taught me a lot on how a website would create a database for the information that it stores. My first thought was that a database would not need as many tables as expected but to create an efficient database it is better to have multiple tables in a normalized form. I learned in the past how to connect a project to a database, but never did I need as much functionality which included joining multiple tables to retrieve a table I wanted to show in the GUI.

With creating parts of the physician homepage and the sign-up page, I gained more experience using JavaScript and session management and creating complex sql queries to retrieve data.