TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

****

BÁO CÁO MÔN HỌC
THỰC HÀNH CƠ SỞ DỮ LIỆU

ĐỀ TÀI: Xây dựng một hệ thống web đơn giản quản lý dữ liệu phòng khám

ĐIỂM	*NHẬN XÉT VÀ CHỮ KÝ CỦA GIẢNG VIÊN*

Giảng viên hướng dẫn: T.S Vũ Tuyết Trinh

Sinh viên thực hiện:

Nguyễn Thanh Huyền – 20184122

Vũ Tuấn Đạt – 20184067

Hà Thị Hạnh – 20184091

Mã lớp: 121258

Hà Nội, tháng 1 năm 2021

MỤC LỤC

I. Lý do & mục đích chọn đề tài

Cách mạng 4.0 đang diễn ra và đem lại ảnh hưởng to lớn trên nhiều mặt của đời sống. Số hóa các tài liệu và văn bản là một trong những việc quan trọng để tối ưu hóa quy trình làm việc, đồng thời có thể trích xuất, sử dụng những dữ liệu đó phục vụ cho mục đích nghiên cứu và dự đoán những xu hướng, biến động trong tương lai. Trong đó, số hóa dữ liệu y tế cũng là một việc rất quan trọng. Dữ liệu y tế được lưu lại có thể giúp các bác sĩ nhanh chóng nắm được tiểu sử bệnh tật, các thuốc đã dùng cũng như cơ địa, thể trạng của bệnh nhân, từ đó nhanh chóng đề ra được phương án điều trị, góp phần nâng cao chất lượng khám chữa bệnh. Vì vậy, nhóm chúng em quyết định chọn đề tài : "*Xây dựng ứng dụng quản lí dữ liệu phòng khám tư nhân*."

Hi vọng qua ứng dụng này, có thể giúp ích các phòng khám quản lí dữ liệu của họ một cách thuận tiện và nhanh chóng nhất, giúp ích cho cả phòng khám lẫn người bệnh về thời gian và công sức.

II. Phân tích & thiết kế cơ sở dữ liệu

A. Mô tả các tính năng của hệ thống

Người sử dụng hệ thống: Người quản lý phòng khám

1. Quản lý về bác sĩ (/doctors …)

1.1 Quản lý danh sách bác sĩ, bao gồm mã số, tên tuổi, chuyên khoa chuyên ngành

1.2 Tìm kiếm bác sĩ theo tên

1.3 Xem bảng lương theo tháng/ năm của các bác sĩ và đồng thời cho phép tìm kiếm lương của bác sĩ theo năm và tháng cụ thể nhập vào

1.4 Cho phép thay đổi tiểu sử của bác sĩ (bổ sung về học vấn, học vị …)

1.5 Cho phép thêm 1 bác sĩ mới

2. Quản lý về bệnh nhân (/patients …)

2.1 Cho phép hiển thị toàn bộ danh sách bệnh nhân cùng các thông tin như ngày sinh, tên bệnh nhân, giới tính và số điện thoại liên hệ

2.2 Cho phép thêm mới vào 1 bệnh nhân

2.3 Cho phép tìm kiếm bệnh nhân dựa theo tên

2.4 Lưu trữ hồ sơ bệnh án, có chứa những cuộc hẹn khám bệnh và chi tiết về cuộc hẹn khám bệnh ấy (như mã đơn khám bệnh, bác sĩ nào kê đơn, đã dùng những thuốc gì và sử dụng qua những dịch vụ gì của phòng khám, cũng như chi tiết về các khoản tiền thanh toán)

2.5 Cho phép chỉnh sửa lại các thông tin về cuộc hẹn (khi cuộc hẹn vẫn chưa diễn ra, trong trường hợp bác sĩ hoặc bệnh nhân có việc đột xuất)

2.6 Cho phép thêm 1 đơn thuốc mới (trong trường hợp người bệnh đến nhiều lần, sẽ có nhiều mã đơn thuốc riêng ứng với nhiều đơn thuốc khác nhau). Khi một loại thuốc được kê N hộp thì số lượng tồn kho (stock_quantity) tự động giảm đi N (sử dụng trigger)

2.7 Trong đơn thuốc, ngoài chẩn đoán sơ bộ của bác sĩ (mục Ghi chú/chẩn đoán) ra, có thể thêm danh mục các thuốc được kê và danh mục các dịch vụ sử dụng tại phòng khám. Khi nhấn "Đã thanh toán" thì thuộc tính is_paid (đã chi trả hay chưa) trong bảng "histories" (lịch sử khám chữa bệnh) sẽ chuyển thành TRUE và không thể thay đổi bất cứ thông tin gì được nữa. Một khi thuộc tính is_paid chuyển thành TRUE thì số lượng bệnh nhân của bác sĩ thực hiện ca khám sẽ tự động tăng lên 1 và lương tháng cũng tự động cập nhật theo công thức

Lương = 6.000.000 + 20%.(tổng giá tiền đã thanh toán = total_price) nếu lương = 0 (ca bệnh đầu tiên) Lương = Lương + 20%(tổng giá tiền đã thanh toán = total_price) nếu lương khác = 0

(6,000.000 (sáu triệu đồng) là lương cứng của bác sĩ) (sử dụng trigger)

3. Quản lý về lịch khám bệnh (đặt lịch từ xa)

3.1 Cho phép đặt lịch khám với bác sĩ chọn trước, hoặc tùy chọn bởi phòng khám. Trong trường hợp bệnh nhân yêu cầu ngày cụ thể và bác sĩ cụ thể, hệ thống sẽ tự động kiểm tra xem bác sĩ được yêu cầu có làm việc vào ngày mà bệnh nhân yêu cầu không. Nếu không thì báo lỗi và nếu có thì xác nhận cuộc hẹn sẽ được diễn ra.

3.2 Trong trường hợp bác sĩ đã có quá số bệnh nhân cố định (số slot cố định) theo ngày thì hệ thống sẽ từ chối những cuộc hẹn sẽ được đặt vào ngày đó.

3.3 Cuộc hẹn đã diễn ra sẽ có sự xác nhận (UPDATE lại trạng thái của cuộc hẹn, cũng chính là "histories")

4. Quản lý về thuốc (/medicines …)

4.1 Cho phép hiển thị trên màn hình danh sách các loại thuốc hiện có trong kho

4.2 Cho phép tìm kiếm theo 'mã số thuốc', 'tên thuốc', 'các loại thuốc có giá cao hơn giá nhập vào' và 'các loại thuốc có giá thấp hơn giá nhập vào'

4.3 Cho phép hiển thị 'các loại thuốc thấp hơn số bất kỳ nhập vào'

4.4 Cho phép thêm 1 loại thuốc mới

4.5 Cho phép chỉnh sửa tên thuốc (cập nhật thông tin) cũng như cập nhật về giá thuốc

5. Quản lý về dịch vụ khám chữa bệnh (/services …)

5.1 Cho phép hiển thị danh sách các dịch vụ (các hình thức khám chữa bệnh) cũng như giá thành tham khảo cho bệnh nhân (khi đặt lịch từ xa chẳng hạn)

5.2 Cho phép tìm kiếm dịch vụ theo tên

6. Quản lý về thu chi (/income_expense …)

6.1 Cho phép hiển thị tiền khám bệnh (thu từ dịch vụ), tiền bán thuốc (doanh thu từ thuốc) và tổng giá tiền thu được của phòng khám theo mỗi tháng hoặc theo năm. Đồng thời cũng hiển thị tiền lương (khoản phải chi) cho các bác sĩ theo tháng và năm

6.2 Cho phép tìm kiếm lương cũng như số bệnh nhân mà bác sĩ đã khám theo năm và theo tháng của 1 năm cụ thể (trường hợp sau yêu cầu nhập đầy đủ cả năm và cả tháng)

7. Quản lý về lịch hẹn khám (/appointments...)

7.1 Hiển thị tất cả các lịch hẹn bao gồm:mã cuộc hẹn, tên bệnh nhân, bác sĩ được đặt lịch và ngày tới khám.

7.2 Cho phép tìm kiếm các cuộc hẹn theo tên bệnh nhân và tên bác sĩ

Công nghệ sử dụng để xây dựng hệ thống trên :

- Frond - end : HTML/CSS/Javascript/jQuery

  Phụ trách : Hà Thị Hạnh

- Back - end : Framework Flask (Python)

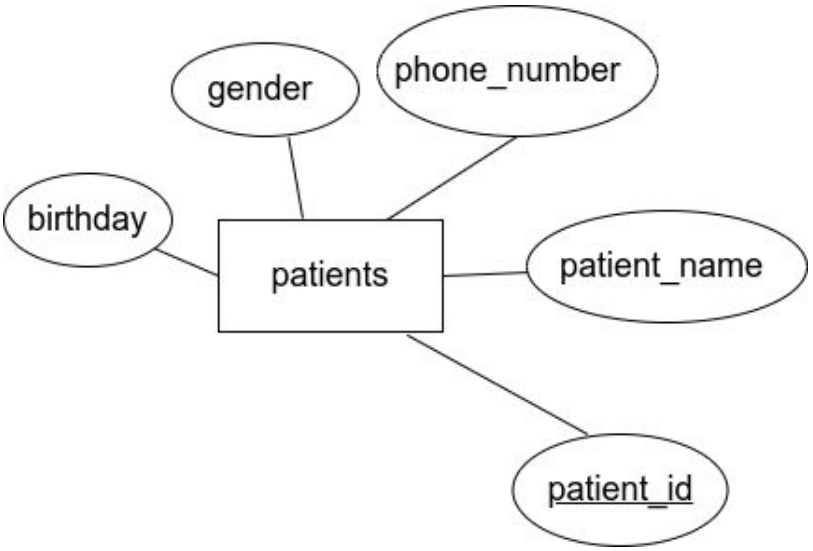
  Phụ trách : Nguyễn Thanh Huyền

- Cơ sở dữ liệu : PostgreSQL

  Phụ trách xây dựng CSDL ảo + thiết kế & vẽ sơ đồ: Vũ Tuấn Đạt

B. Mô tả các thực thể và liên kết

1. Thực thể patients (bệnh nhân)



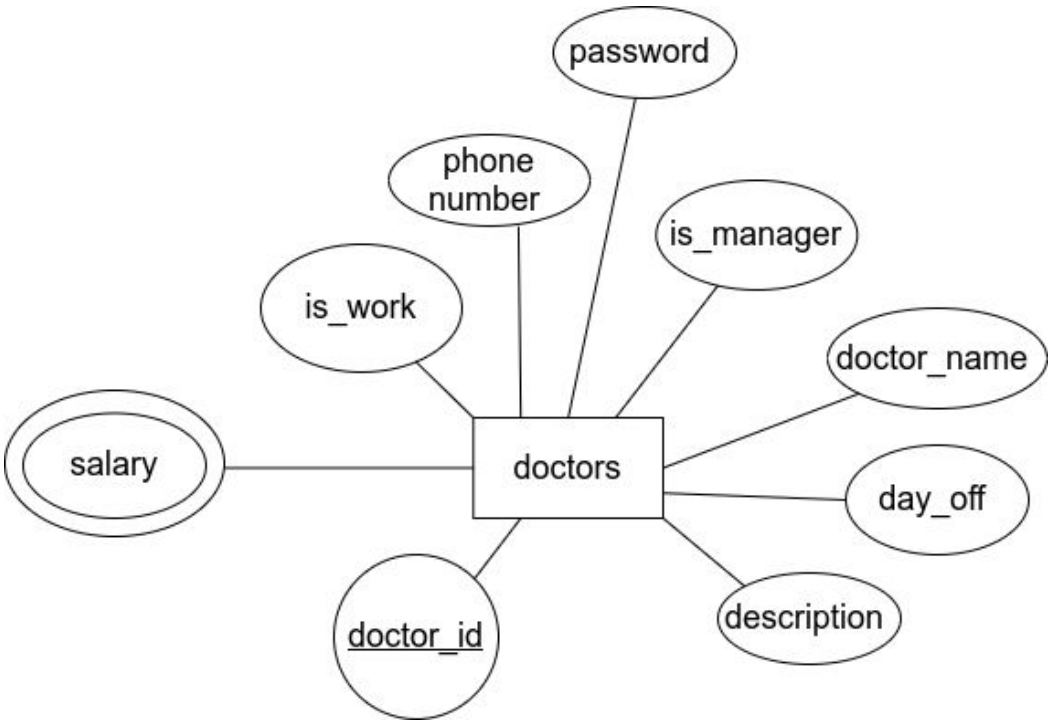
Mục đích: Lưu trữ những thông tin cá nhân của người bệnh phục vụ khám chữa bệnh.

Gồm có các thuộc tính:

STT	Tên thuộc tính	Ý nghĩa	Ghi chú	Kiểu dữ liệu
1	patient_id	Mã số bệnh nhân	**Khóa chính (primary key)**	SERIAL
2	patient_name	Họ tên bệnh nhân		varchar(50)
3	birthday	Ngày sinh của bệnh nhân		date
4	gender	Giới tính bệnh nhân		varchar(1)
5	phone_number	Số điện thoại của bệnh nhân		Varchar

Ràng buộc (**constraint**): Giới tính bệnh nhân phải thuộc 1 trong 2 giá trị ‘M’ (male) hoặc ‘F’ (female)

b. Thực thể doctors (bác sĩ)



Mục đích: Lưu thông tin cá nhân của bác sĩ, nhằm bố trí việc khám bệnh và truy xuất lịch sử khám bệnh.

Gồm các thuộc tính:

STT	Tên thuộc tính	Ý nghĩa	Ghi chú	Kiểu dữ liệu
1	doctor_id	Mã số bác sĩ	**Khóa chính (primary key)**	SERIAL
2	doctor_name	Họ tên bác sĩ		varchar(50)
3	phone_number	Số điện thoại bác sĩ		varchar(15)
4	day_off	Ngày nghỉ trong tuần		varchar (7)
5	description	Mô tả (tiểu sử & học vấn của bác sĩ)		varchar (255)
6	is_manager	Có phải quản lý không		boolean
7	is_work	Có đang làm việc không		boolean
8	password	Mật khẩu		varchar(20)
9	salary	Mức lương của bác sĩ	**thuộc tính đa trị**	integer

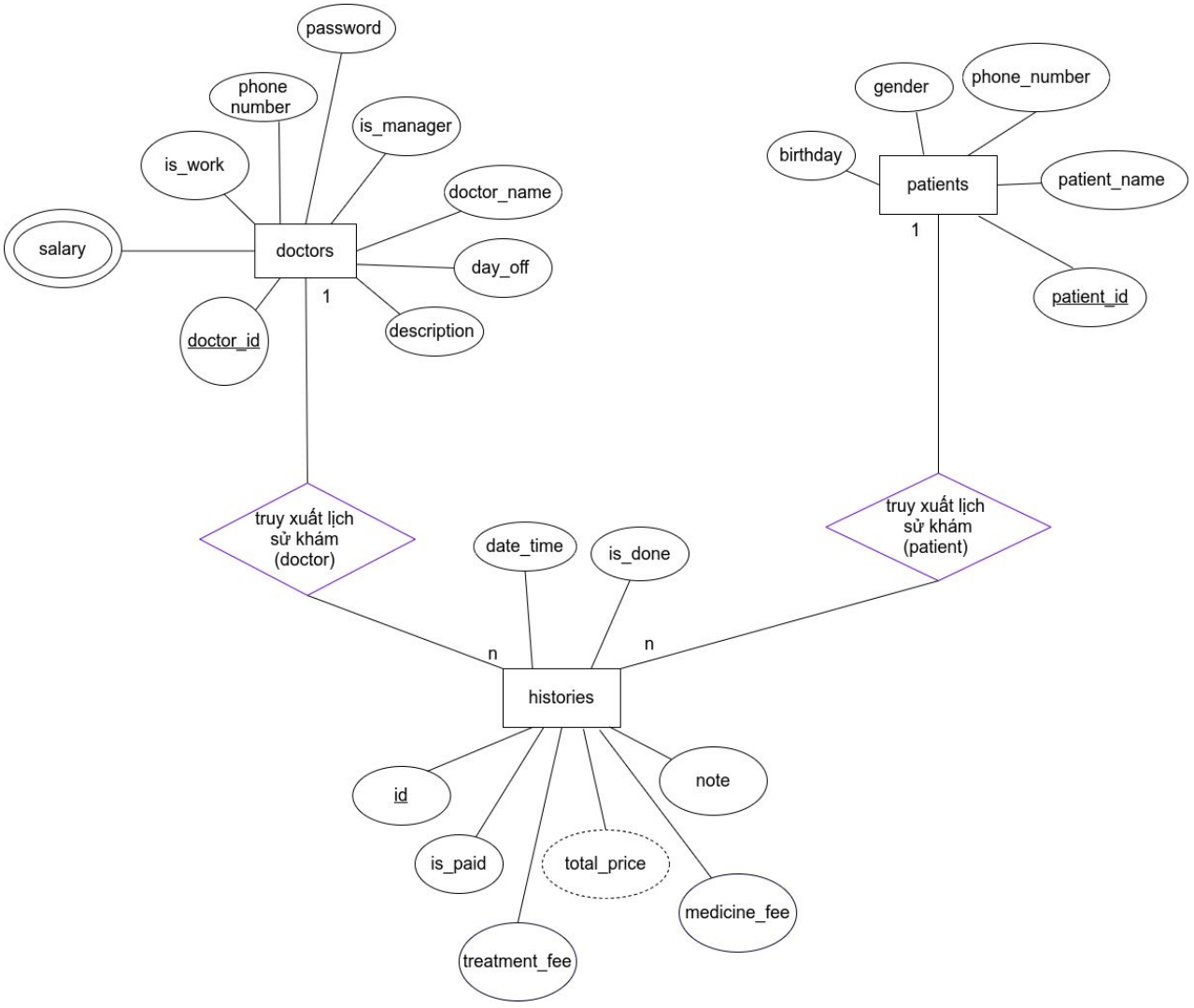
(Lý do thêm thuộc tính **is_work**: Khi 1 bác sĩ từng làm việc tại phòng khám, thì sẽ có đơn thuốc lưu **doctor_id** của bác sĩ đó lại, nếu tiến hành DELETE toàn bộ dữ liệu về bác sĩ ấy khi người đó nghỉ việc rồi thì việc truy xuất đơn thuốc sẽ gặp rắc rối lớn (vì phải JOIN bảng **doctors** với bảng **histories** (chung **doctor_id**)). Nên khi 1 bác sĩ nghỉ việc thì chỉ tiến hành UPDATE **is_work** của người đó từ **TRUE** thành **FALSE**)

c. Thực thể histories (lịch sử khám chữa bệnh)

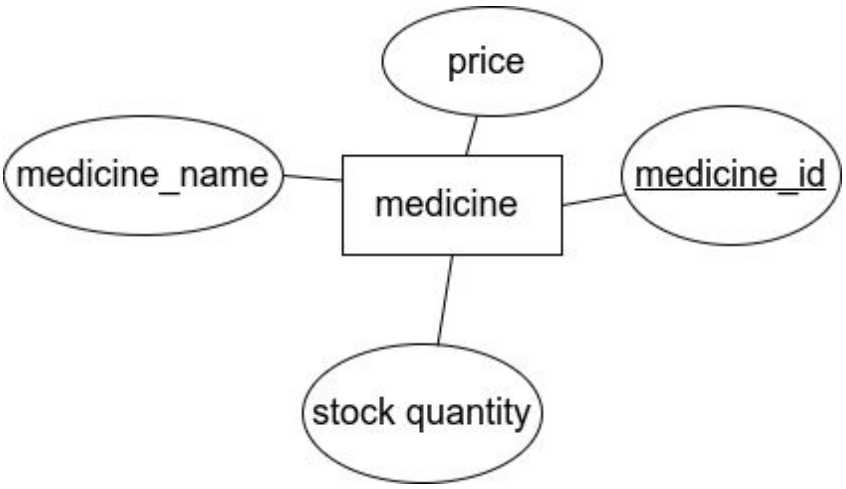
Mục đích: Truy xuất bác sĩ khám bệnh, bệnh nhân, thuốc đã dùng và ngày giờ khám bệnh

STT	Tên thuộc tính	Ý nghĩa	Ghi chú	Kiểu dữ liệu
1	id	mã lịch sử (mã đơn thuốc)	**Khóa chính (primary key)**	SERIAL
2	note	ghi chú		varchar(255)
3	date_time	ngày giờ diễn ra ca khám		date
4	is_done	ca khám đã diễn ra chưa		boolean
5	medicine_fee	tiền thuốc		integer
6	treatment_fee	tiền phí điều trị (tiền dịch vụ)		integer
7	total_price	tổng chi phí	**Là thuộc tính suy diễn**	integer
8	is_paid	bệnh nhân đã chi trả chưa		boolean

(thuộc tính ***total_price*** = ***medicine_fee*** + ***treatment_fee***)



d. Thực thể medicines (thuốc)



Mục đích: Lưu lại danh mục thuốc, phục vụ truy xuất lịch sử dùng thuốc cũng như thu chi

Gồm các thuộc tính:

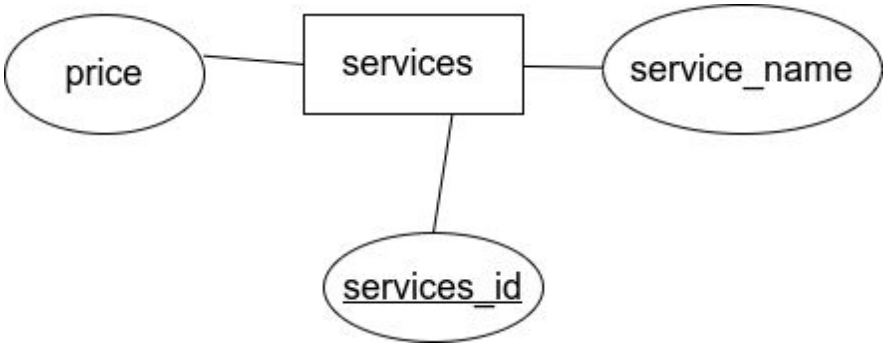
STT	Tên thuộc tính	Ý nghĩa	Ghi chú	Kiểu dữ liệu
1	medicine_id	mã số thuốc	**Khóa chính (primary key)**	SERIAL
2	medicine_name	tên thuốc		varchar(250)
3	unit_price	giá thành (theo hộp – vỉ)		integer
4	stock_quantity	số lượng tồn kho		integer

e. Thực thể services (dịch vụ)

Mục đích: Lưu thông tin và giá tiền các loại hình dịch vụ (thăm khám) trong phòng khám

Gồm các thuộc tính:

STT	Tên thuộc tính	Ý nghĩa	Ghi chú	Kiểu dữ liệu
1	service_id	Mã số dịch vụ	**Khóa chính (primary key)**	SERIAL
2	service_name	Tên dịch vụ		varchar(50)
3	price	Giá dịch vụ		integer

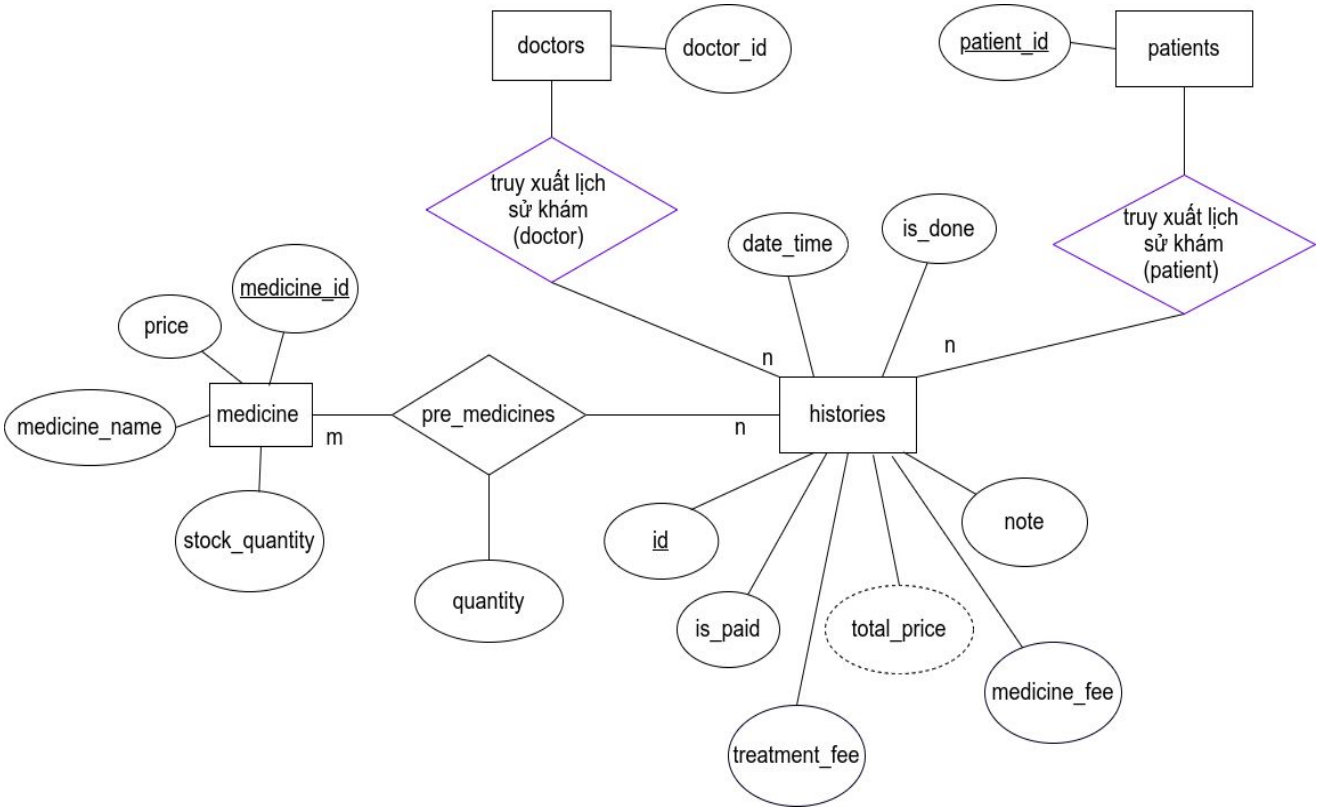


Vấn đề : Nếu bây giờ chúng ta lưu tất cả các thuốc đã được kê, các dịch vụ đã được khám vào cùng bảng histories, sẽ xảy ra dư thừa dữ liệu trầm trọng. Sẽ có thể xảy ra hiện tượng dị thường dữ liệu khi thêm, khi xóa và sửa đổi.

Từ đó, nhóm chúng em đề xuất xây dựng 2 bảng con : pre_medicines và pre_services.

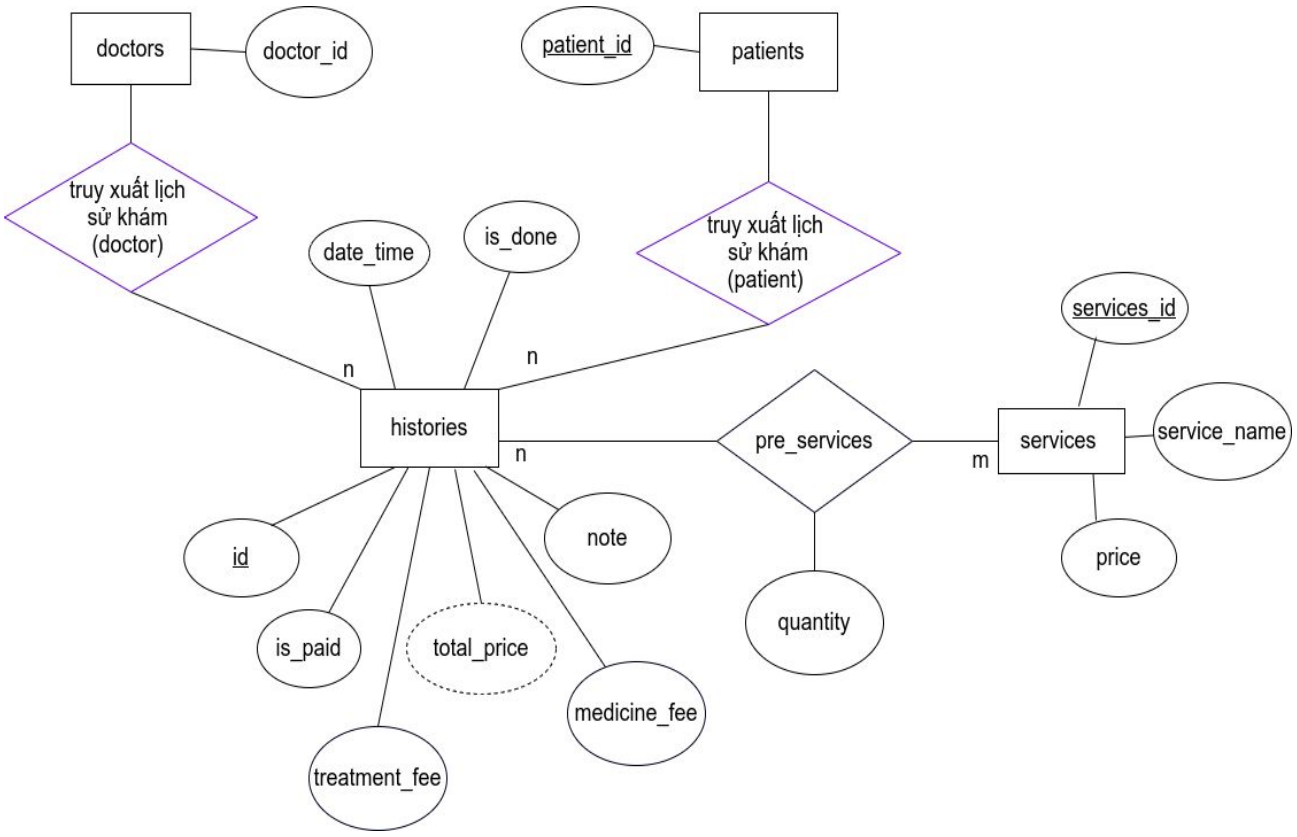
f. Liên kết pre_medicine (prescription: đơn thuốc & medicines: thuốc)

Mục đích: Ghi chú rõ ràng loại thuốc nào thuộc đơn nào, số lượng bao nhiêu:



f. Liên kết pre_services

Mục đích: Lưu thông tin về các loại dịch vụ dùng trong từng đơn, cũng như số lần sử dụng dịch vụ đó.



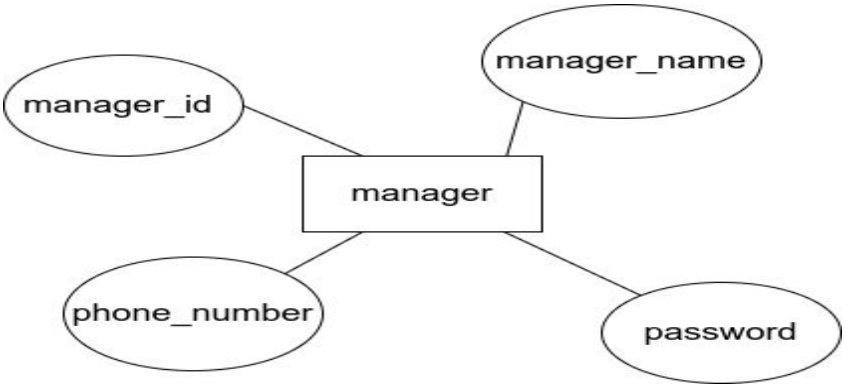
(Ví dụ: khi nhổ răng khôn bác sĩ chụp lần thứ nhất xác định xem răng khôn mọc lệch thế nào. Nhổ xong, trước khi khâu chỉ nha khoa cần chụp lần thứ 2 xem còn mảnh răng nào còn sót lại không - tức là lưu số lần dùng dịch vụ khám chữa bệnh – quantity)

h. Thực thể manager (quản lí)

Mục đích: Lưu trữ thông tin về những người có thể đăng nhập vào hệ thống và quản lí hoạt động của phòng khám. (chỉ phục vụ cho thao tác đăng nhập)

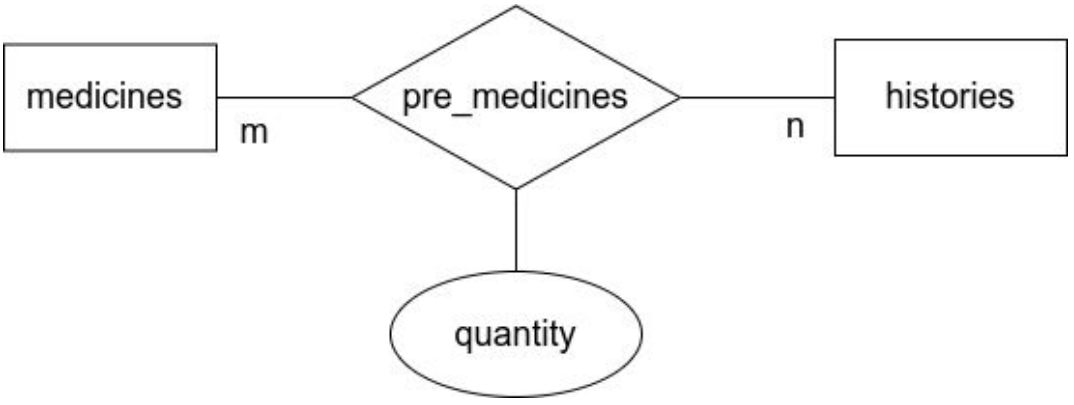
Gồm các thuộc tính:

STT	Tên thuộc tính	Ý nghĩa	Ghi chú	Kiểu dữ liệu
1	manager_id	Mã số giám đốc	**Khóa chính (primary key)**	SERIAL
2	manager_name	Tên giám đốc		varchar(50)
3	phone_number	Số điện thoại giám đốc		varchar(15)
4	password	Mật khẩu		varchar(20)



C. Chuyển đổi sang mô hình quan hệ

1. Chuyển đổi liên kết **pre_medicines** giữa bảng **medicines** (thuốc) và **histories** (lịch sử thăm khám) (liên kết nhiều – nhiều, do 1 thuốc có thể nằm trong nhiều đơn, và 1 đơn cũng có thể có nhiều loại thuốc), dùng để ghi lại loại thuốc nào, nằm trong đơn thuốc nào với số lượng là bao nhiêu thành 1 bảng mới, bảng **pre_medicines.** Bảng này có thể không có khóa chính, hoặc mượn khóa chính của 2 bảng **medicines** và **histories** làm khóa chính (biết được mã đơn và mã thuốc ta suy ra được chính xác số lượng thuốc loại ấy)

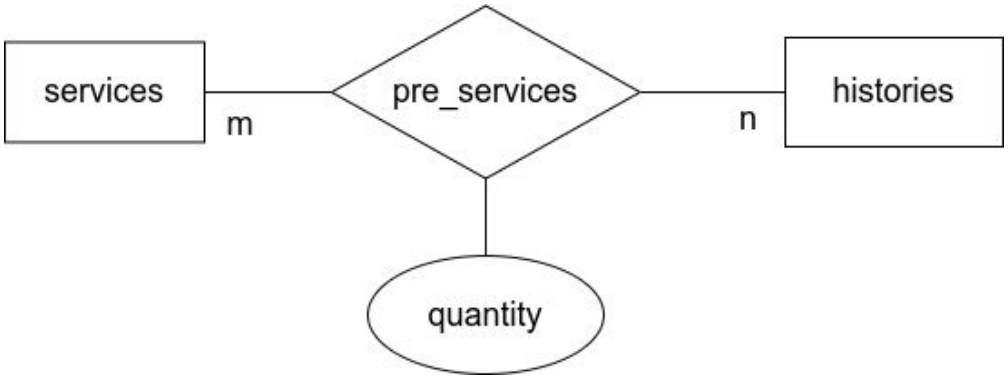


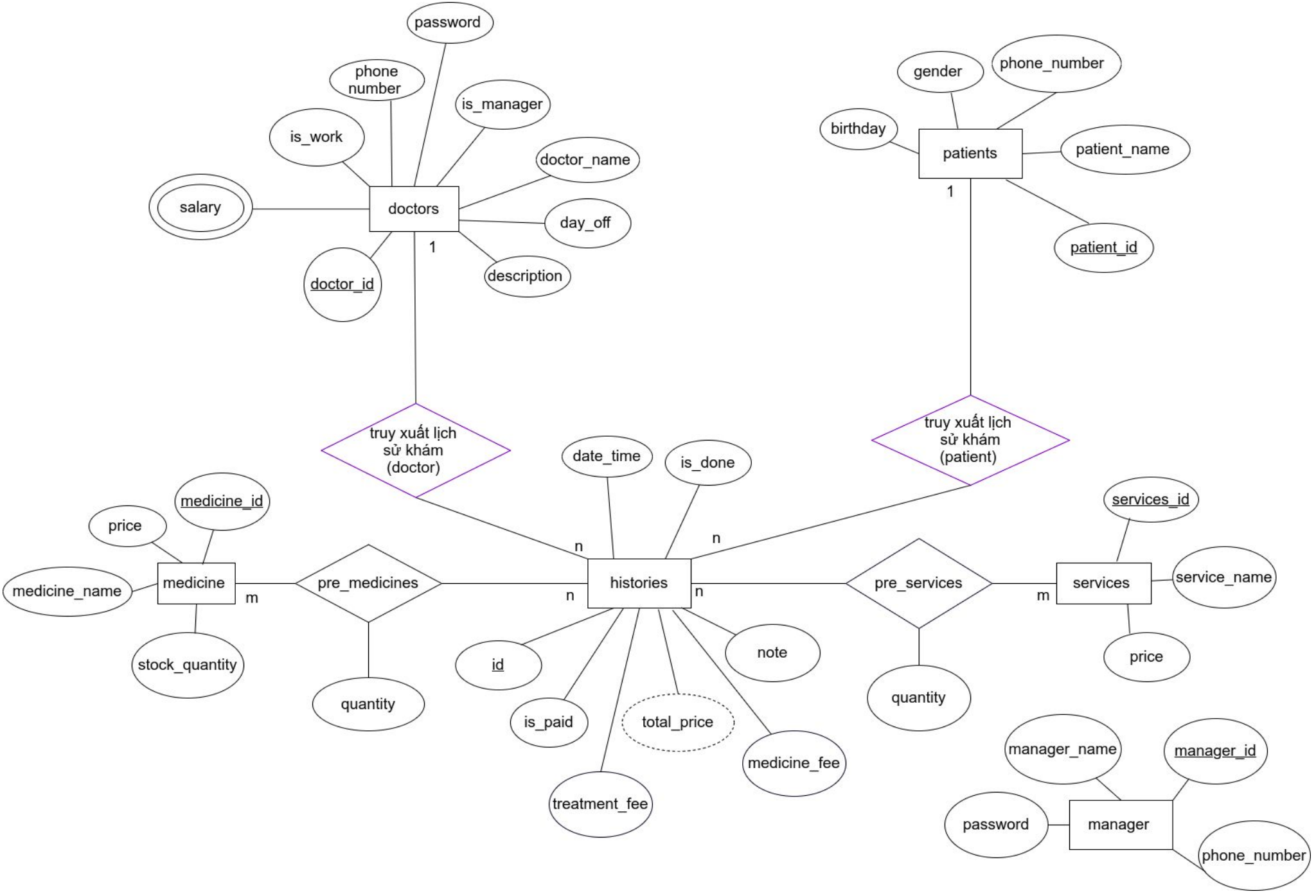
Bảng pre_medicines có 3 thuộc tính:

STT	Tên thuộc tính	Ý nghĩa	Ghi chú	Kiểu dữ liệu
1	prescription_id	Mã số đơn thuốc	tham chiếu **id** của **histories**	integer
2	medicine_id	Mã số thuốc	tham chiếu **medicine_id** của **medicines**	integer
3	quantity	Số lượng thuốc theo đơn		integer

2. Chuyển đổi tương tự với liên kết **pre_services** (ghi lại dịch vụ nào dùng trong đơn nào). Bảng pre_services cũng có 3 thuộc tính.

STT	Tên thuộc tính	Ý nghĩa	Ghi chú	Kiểu dữ liệu
1	prescription_id	Mã số đơn thuốc	tham chiếu **id** của **histories**	integer
2	service_id	Mã số dịch vụ khám	tham chiếu **service_id** của **services**	integer
3	quantity	Số lượng thuốc theo đơn		integer




password
phone number
is_manager
is_work
doctor_name
salary
doctors
day_off
1
doctor_id
description
gender
phone_number
birthday
patients
patient_name
1
patient_id
truy xuất lịch sử khám (doctor)
truy xuất lịch sử khám (patient)
date_time
is_done
n
n
medicine_id
price
medicine_name
medicine
m
pre_medicines
n
histories
n
pre_services
m
services
services_id
service_name
price
stock_quantity
quantity
id
is_paid
total_price
treatment_fee
note
medicine_fee
quantity
manager_name
manager_id
password
manager
phone_number

3. Tạo bảng lương (salary_table)

Nhận thấy hiện tại trong thực thể **doctors** có thuộc tính **salary** là đa trị (1 bác sĩ, ở cùng trong 1 năm có thể có rất nhiều mức lương khác nhau) (vi phạm dạng chuẩn 1). Để xác định 1 mức lương 1 cách chính xác, chúng ta cần biết 3 yếu tố:

+ Đó là lương của bác sĩ nào (cần doctor_id)

+ Đó là lương trong năm nào (which_year)

+ Đó là lương của tháng nào trong năm ấy (which_month)

Bảng salary_table được tạo ra, với các thuộc tính:

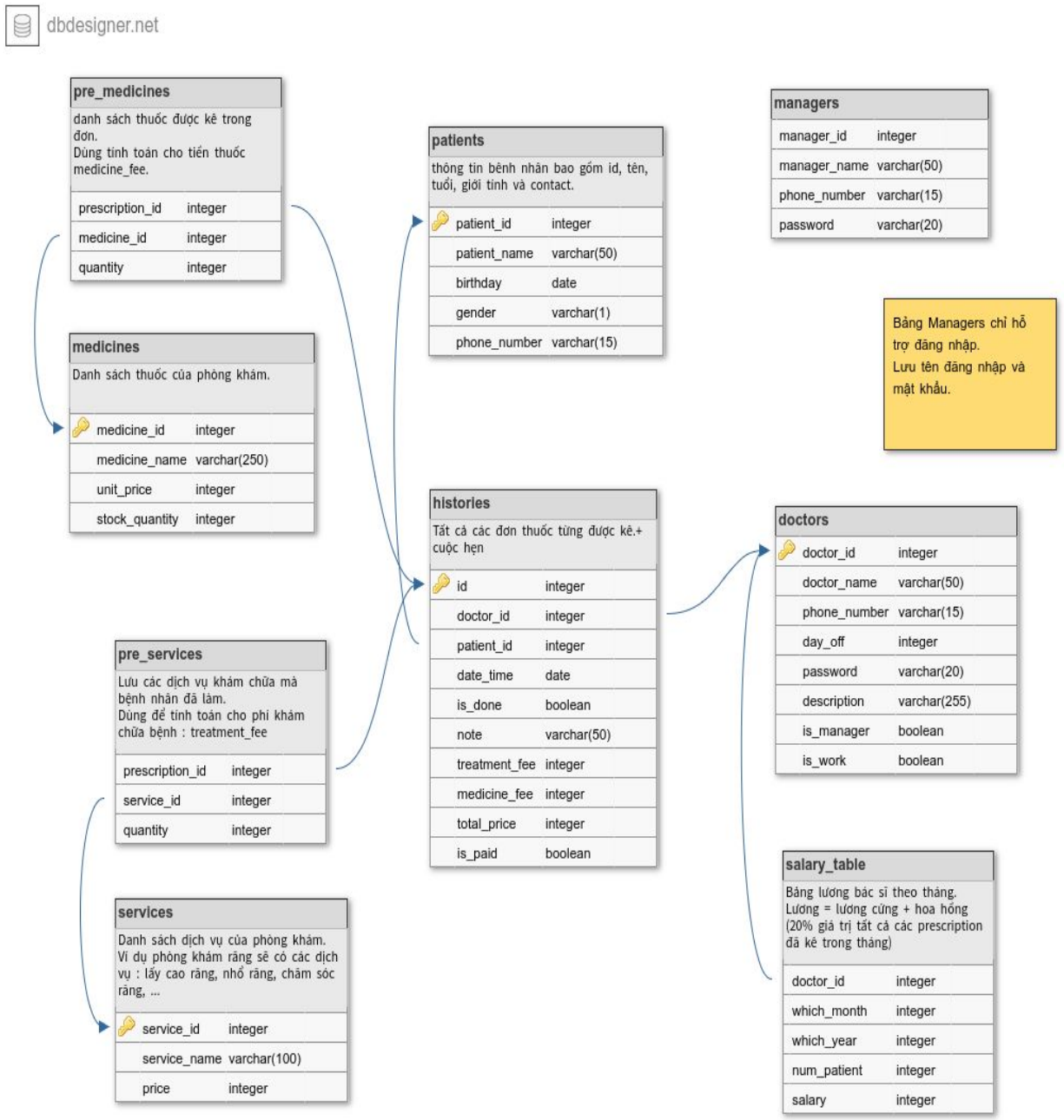
STT	**Tên thuộc tính**	**Ý nghĩa**	**Ghi chú**	**Kiểu dữ liệu**
1	doctor_id	mã số bác sĩ	**tham chiếu đến doctors**	integer
2	which_month	tháng nào		integer
3	which_year	năm nào		integer
4	num_patient	số lượng bệnh nhân		integer
5	salary	lương của bác sĩ		integer

4. Các liên kết **1 - n** giữa bảng **histories** với **doctors** chuyển thành ***khóa ngoài***, tức là **doctor_id** trong bảng **histories** tham chiếu đến **doctor_id** trong bảng **doctors**. Tương tự như vậy với liên kết **1 – n** giữa bảng **patients** và bảng **histories** (mục đích là để lưu lại mã số của bác sĩ và bệnh nhân đã thực hiện cái đơn khám khi truy xuất lịch sử khám chữa bệnh - **histories**)

Cuối cùng, chúng ta có các bảng như sau:

(nếu không nhìn rõ xin cô hãy vào liên kết sau)

https://dbdesigner.page.link/Lc7foJjjxQzqWE8NA



D. Câu lệnh tạo bảng và các ràng buộc

1. Bảng doctors

```
DROP TABLE  doctors CASCADE;
CREATE TABLE doctors (
     doctor_id serial,
     doctor_name varchar(50) ,
     phone_number varchar(15) ,
     day_off integer ,
     description varchar(255) ,
     password varchar (20),
     is_work boolean,
     is_manager boolean,
     CONSTRAINT doctors_pk PRIMARY KEY (doctor_id)
);
```

2. Bảng patients

```
DROP TABLE IF EXISTS patients CASCADE;
CREATE TABLE patients (
     patient_id serial,
     patient_name varchar(50) ,
     birthday date ,
     gender varchar(1) ,
     phone_number varchar(15) ,
     CONSTRAINT patients_pk PRIMARY KEY (patient_id),
     CONSTRAINT patients_chk CHECK(gender in ('M','F'))
);
```

Ràng buộc: giới tính của bệnh nhân thuộc 1 trong 2 giá trị ‘M’,’F’ (Male và Female)

3. Bảng histories

```
DROP TABLE IF EXISTS histories CASCADE;
CREATE TABLE histories (
    id serial ,
    doctor_id int ,
    patient_id integer ,
    note varchar(50) ,
    date_time date ,
    is_done BOOLEAN ,

    medicine_fee integer default 0,
    treatment_fee integer default 0,
    total_price integer default 0,
    is_paid BOOLEAN ,
    CONSTRAINT histories_pk PRIMARY KEY (id)
);
```

Ràng buộc:

Thuộc tính ***doctor_id*** và ***patient_id*** trong bảng ***histories*** lần lượt tham chiếu đến ***doctor_id*** trong bảng ***doctors*** và ***patient_id*** trong bảng ***patients***

```
ALTER TABLE histories ADD CONSTRAINT histories_fk0 FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id) ON UPDATE CASCADE;
ALTER TABLE histories ADD CONSTRAINT histories_fk1 FOREIGN KEY (patient_id) REFERENCES patients(patient_id) ON UPDATE CASCADE;
```

4. Bảng medicines

```
DROP TABLE IF EXISTS medicines CASCADE;
CREATE TABLE medicines (
    medicine_id serial ,
    medicine_name varchar(250) ,
    unit_price integer ,
    stock_quantity integer ,
    CONSTRAINT medicines_pk PRIMARY KEY (medicine_id)
```

```
) ;
```

5. Bảng pre_medicines

```
DROP TABLE IF EXISTS pre_medicines CASCADE;
CREATE TABLE pre_medicines (
    prescription_id integer ,
    medicine_id integer ,
    quantity integer
);
```

Ràng buộc:

prescription_id (mã số đơn thuốc) tham chiếu đến thuộc tính ***id*** trong bảng ***histories***

medicine_id (mã số thuốc) tham chiếu đến thuộc tính ***medicine_id*** trong bảng ***medicines***

```
ALTER TABLE pre_medicines ADD CONSTRAINT pre_medicines_fk0 FOREIGN KEY (prescription_id) REFERENCES histories(id) ON UPDATE CASCADE;
ALTER TABLE pre_medicines ADD CONSTRAINT pre_medicines_fk1 FOREIGN KEY (medicine_id) REFERENCES medicines(medicine_id) ON UPDATE CASCADE;
```

6. Bảng services

```
DROP TABLE IF EXISTS services CASCADE;
CREATE TABLE services (
    service_id serial ,
    service_name varchar(50) ,
    price integer ,
    CONSTRAINT services_pk PRIMARY KEY (service_id)
);
```

7. Bảng pre_services

```
DROP TABLE IF EXISTS pre_services CASCADE;
CREATE TABLE pre_services (
    prescription_id integer ,
    service_id integer ,
    quantity integer
);
```

Ràng buộc: prescription_id (mã số đơn thuốc) phải tham chiếu đến khóa chính ***id*** của bảng ***histories***

service_id (mã số dịch vụ) phải tham chiếu đến khóa chính ***service_id*** của bảng ***services***

```
ALTER TABLE pre_services ADD CONSTRAINT pre_services_fk0 FOREIGN KEY (prescription_id) REFERENCES histories(id) ON UPDATE CASCADE;
```

```
ALTER TABLE pre_services ADD CONSTRAINT pre_services_fk1 FOREIGN KEY (service_id) REFERENCES services(service_id) ON UPDATE CASCADE;
```

8. Bảng salary_table

Câu lệnh tạo bảng:

```
DROP TABLE IF EXISTS salary_table CASCADE;
CREATE TABLE salary_table (
    doctor_id integer ,
    which_month integer ,
    which_year integer ,
    num_patient integer ,
    salary integer
);
```

Ràng buộc: thuộc tính ***doctor_id*** phải tham chiếu đến khóa chính ***doctor_id*** của bảng ***doctors***

```
ALTER TABLE salary_table ADD CONSTRAINT salary_table_fk0 FOREIGN KEY (doctor_id) REFERENCES doctors(doctor_id) ON UPDATE CASCADE;
```

9. Bảng managers

```
DROP TABLE IF EXISTS managers CASCADE;
CREATE TABLE managers (
     manager_id serial ,
     manager_name varchar(50) ,
     phone_number varchar(15) ,
     password varchar(20)
);
```

III. Danh sách truy vấn, cùng với chú thích mô tả về mục đích, cách thực hiện

Thành viên: Hà Thị Hạnh (tổng cộng: 15 câu)

A. Quản lý services (3 câu)

1. Khi vào **/services** thì hệ thống sẽ hiển thị toàn bộ danh sách các dịch vụ gồm id dịch vụ, tên dịch vụ và giá dịch vụ có trong bảng services theo thứ tự tăng dần id dịch vụ , sử dụng câu truy vấn:

```
SELECT * FROM services
```

(Gọi hàm **service_page** trong file **routes.py** để thực thi câu truy vấn trên, liệt kê mọi loại dịch vụ)

2. Tìm kiếm dịch vụ theo tên. Ví dụ tìm dịch vụ có tên "'Nhổ răng sữa (trẻ em)"

```
Nhổ răng sữa (trẻ em)
```

Sử dụng câu truy vấn:

```
SELECT *FROM services WHERE service_name = 'Nhổ răng sữa
(trẻ em)'
```

3. Dịch vụ được làm nhiều nhất (không demo trên web)

```
select service_id, service_name, count(service_id)

from pre_services as ps JOIN services using(service_id)

JOIN histories as h ON (h.id = ps.prescription_id)

where h.date_time between '2020-12-31' and '2021-4-1'

group by service_id,service_name

order by count(service_id) DESC
```

```
limit 2
```

B. Quản lý medicines (10 câu)

1. Khi vào **/medicines** thì hệ thống sẽ hiển thị toàn bộ danh sách các thuốc gồm id thuốc, tên thuốc, giá thuốc và số lượng thuốc còn trong kho có trong bảng medicines theo thứ tự tăng dần id thuốc , sử dụng câu truy vấn:

Gọi hàm **query_medicines (medicine_id=None, medicine_name=None, from_price=None, to_price=None, stock_quantity=None)** trong file **models.py** để thực hiện truy vấn:

```
SELECT * FROM medicines ORDER BY medicine_id ASC
```

2,3. Tìm kiếm thuốc theo mã Id thuốc, tên thuốc.

Gọi hàm **query_medicines (medicine_id=None, medicine_name=None, from_price=None, to_price=None, stock_quantity=None)** trong file **models.py** để thực hiện truy vấn:

Ví dụ tìm thuốc có id thuốc là 3, sử dụng câu truy vấn:

```
SELECT * FROM medicines WHERE medicine_id = 3
```

Ví dụ tìm thuốc có tên chứa chữ 'viêm', sử dụng câu truy vấn:

```
SELECT * FROM medicines WHERE medicine_name like
'%viêm%' or medicine_name ~ 'Viêm'
```

4,5. Tìm thuốc có giá lớn hơn giá cho trước, hoặc nhỏ hơn giá cho trước.

Ví dụ thuốc giá lớn hơn 20.000 (đồng), sử dụng câu truy vấn:

```
SELECT * FROM medicines WHERE unit_price>20000
```

Ví dụ thuốc giá nhỏ hơn 110.000 (đồng) sử dụng câu truy vấn

```
SELECT * FROM medicines WHERE unit_price<110000
```

Gọi hàm **query_medicines (medicine_id=None, medicine_name=None, from_price=None, to_price=None, stock_quantity=None)** trong file **models.py** để thực hiện truy vấn.

6. Tìm thuốc có giá nằm trong khoảng cho trước

```
SELECT * FROM medicines WHERE unit_price>20000 and
unit_price<110000
```

Gọi hàm **query_medicines (medicine_id=None, medicine_name=None, from_price=None, to_price=None, stock_quantity=None)** trong file **models.py** để thực hiện truy vấn:

7. Tìm kiếm những thuốc có số lượng nhỏ hơn số lượng nhập vào (kiểm tra số lượng tồn kho. VD: 30)

Gọi hàm **query_medicines (medicine_id=None, medicine_name=None, from_price=None, to_price=None, stock_quantity=None)** trong file **models.py** để thực hiện truy vấn:

```
SELECT * FROM medicines WHERE stock_quantity<30
```

8. Cập nhật, chỉnh sửa thông tin thuốc (tên thuốc và giá thuốc):

Hệ thống gọi đến hàm **medicine_update(medicine_id)** trong file **routes.py** để thực thi câu lệnh:

```
cur.execute("UPDATE medicines SET medicine_name = (%s),
unit_price = (%s), stock_quantity = (%s) "
"WHERE medicine_id = (%s)",
(request.form['medicine_name'],
request.form['unit_price'],
request.form['stock_quantity'],medicine_id))
```

9. Thêm thuốc mới: Khi thêm thuốc mới, người dùng cần nhập vào tên thuốc, giá và số lượng:

medicine_name

unit_price

quantity

Sau khi form này được xác nhận (Confirm) thì hệ thống sẽ gọi đến hàm **insert_new_medicine** trong file **routes.py** để thực hiện truy vấn.

Sử dụng câu lệnh:

```
INSERT INTO medicines (medicine_name, unit_price,
stock_quantity ) "
"VALUES(%s,%s,%s)",
(form.medicine_name.data,form.unit_price.data,
form.stock_quantity.data)
```

10. Đưa ra thuốc được kê nhiều nhất (không demo trên web)

```
select medicine_id, medicine_name, count(medicine_id)

from pre_medicines as pm JOIN medicines using(medicine_id)

JOIN histories as h ON (h.id = pm.prescription_id)

where extract(month from h.date_time) = 1 and extract(year
from h.date_time) = 2021

group by medicine_id,medicine_name

order by count(medicine_id) DESC

limit 1
```

C. Quản lý các cuộc hẹn đã được đặt lịch (2 câu)

Hệ thống gọi hàm

query_appointment(doctor_name=doctor_name,patient_name=patient_name) trong file **models.py** để thực hiện các câu truy vấn:

1. Hiển thị các cuộc hẹn đã được đặt lịch

```
Select h.id, p.patient_name, d.doctor_name, h.date_time,
h.disease_name, p.patient_id

from histories as h, doctors as d, patients as p

WHERE is_done = False and h.doctor_id = d.doctor_id and
h.patient_id = p.patient_id
```

2. Tìm kiếm cuộc hẹn theo tên bác sĩ được đặt lịch hoặc bệnh nhân được đặt lịch:

Ví dụ tìm cuộc hẹn của bệnh nhân tên có chữ 'anh':

```
select h.id, p.patient_name, d.doctor_name, h.date_time,
p.patient_id

from histories as h, doctors as d, patients as p

WHERE (p.patient_name like '%anh%' or p.patient_name ~ 'Anh')
and is_done = False and h.doctor_id = d.doctor_id and
h.patient_id = p.patient_id
```

Ví dụ tìm cuộc hẹn của bác sĩ tên có chữ 'binh'

```
select h.id, p.patient_name, d.doctor_name, h.date_time,
p.patient_id from histories as h, doctors as d, patients as p

WHERE (d.doctor_name like '%binh%' or d.doctor_name ~ 'Binh')
and is_done = False and h.doctor_id = d.doctor_id and
h.patient_id = p.patient_id
```

Thành viên : Nguyễn Thanh Huyền (tổng cộng: 25 câu)

A. Quản lí bác sĩ (Bảng doctors) (4 câu)

Các hàm trong source code web : doctor_page, doctor_detail, insert_new_doctor, doctor_update

1. Hiển thị toàn bộ danh sách bác sĩ theo mã số bác sĩ và tên của bác sĩ

*SELECT * FROM doctors ORDER BY doctor_id ASC*

2. Tìm kiếm theo tên bác sĩ

SELECT doctor_id, doctor_name

*FROM doctors WHERE doctor_name **like** '%{doctor_name}%' or doctor_name ~ '{doctor_name}'*

Kết quả trả về dãy tên các bác sĩ, có chứa tên là An, cũng có chứa họ Tran (Trần), do trong xâu "Tran" cũng có chứa xâu "an".

3. Thêm bác sĩ mới

INSERT INTO doctors (doctor_name, phone_number, day_off, password, description, is_manager, is_work) VALUES (...)

4. Chỉnh sửa doctors bao gồm chỉnh sửa tiểu sử (description), ngày nghỉ (day_off), số điện thoại liên hệ (phone_number) và tình trạng làm việc (is_work)
Khi một bác sĩ nghỉ việc, cập nhật tình trạng làm việc về False

UPDATE doctors SET is_work = False WHERE doctor_id = {doctor_id}

B. Quản lý về bệnh nhân (Bảng Patients) (5 câu)

Các hàm trong source code web : patient_page, patient_detail, insert_new_patient

1. Hiển thị toàn bộ danh sách bệnh nhân cùng các thông tin như ngày sinh, tên bệnh nhân, giới tính và số điện thoại liên hệ. Cho phép tìm kiếm bệnh nhân dựa theo tên

*SELECT * FROM patient ;*

*SELECT * FROM patients WHERE patient_name like '%*patient_name*%'*

2. Thêm mới vào 1 bệnh nhân

INSERT INTO patients (patient_name, birthday, gender, phone_number) VALUES (...)

3. Truy xuất hồ sơ bệnh án thông qua patient_id. Trong hồ sơ gồm : danh sách những cuộc hẹn khám bệnh (đã diễn ra, lịch hẹn trước - chưa diễn ra). Chi tiết trong một hồ sơ bao gồm mã đơn khám bệnh, ngày giờ khám bệnh, bác sĩ kê đơn, chẩn đoán của bác sĩ, thuốc đã kê, dịch vụ đã dùng, chi tiết về các khoản thanh toán, đã thanh toán hay chưa.

SELECT d.doctor_name, note, total_price, medicine_fee, treatment_fee, h.id, date_time, is_paid, is_done FROM histories as h, doctor as d

WHERE h.patient_id = {patient_id} and h.doctor_id = d.doctor_id

4,5. Lấy ra các thuốc được kê trong đơn và các dịch vụ điều trị

- Lấy ra các thuốc được kê trong đơn :

SELECT pm.prescription_id, medicine_name, pm.quantity

FROM medicines as m, histories as h, pre_medicines as pm

WHERE h.patient_id = {patient_id} and m.medicine_id = pm.medicine_id

and pm.prescription_id = h.id

- Tương tự lấy ra các dịch vụ đã điều trị :

SELECT pm.prescription_id, service_name, pm.quantity

FROM services as s, histories as h, pre_services as ps

WHERE h.patient_id = {patient_id} and s.service_id = ps.service_id

and ps.prescription_id = h.id

<u>C. Quản lí lịch sử khám bệnh của phòng khám (bảng histories)</u> (9 câu)

Các hàm trong source code web : appoinment_page, histories_insert, histories_update,

1,2,3. Thêm cuộc hẹn mới và xem danh sách những cuộc hẹn chưa diễn ra.

- Thêm cuộc hẹn mới : (khi bệnh nhân đặt lịch trước, hoặc lên lịch hẹn tái khám)

INSERT INTO histories (doctor_id, patient_id, note, date_time, treatment_fee, medicine_fee, total_price, is_done, is_paid)

VALUES ({doctor_id}, {patient_id}, {note}, {date_time}, 0, 0, 0, False, False)

- Thêm cuộc hẹn mới (bệnh nhân tới trực tiếp, không đặt lịch) :

INSERT INTO histories (doctor_id, patient_id, note, date_time, treatment_fee, medicine_fee, total_price, is_done, is_paid)

VALUES ({doctor_id}, {patient_id}, {note}, current_date, 0, 0, 0, True, False)

- Xem danh sách những cuộc hẹn chưa diễn ra :

*SELECT * FROM histories WHERE is_done = False;*

4. Cập nhật các thông tin về cuộc hẹn (khi cuộc hẹn vẫn chưa diễn ra, trong trường hợp bác sĩ bận việc, hoặc bệnh nhân muốn đổi ngày, đổi bác sĩ).

UPDATE histories SET doctor_id = {doctor_id} WHERE id = {id}

UPDATE histories SET date_time = '{date_time}' WHERE id = {id}

Sau khi cuộc hẹn diễn ra:

Các hàm gọi đến trong source code : histories_confirm, prescription_medicine_update, prescription_service_update, cal_total_price.

Gọi đến hàm **histories_confirm** - xác nhận cuộc hẹn diễn ra (is_done=True) : lúc này quản lí có thể cập nhật thuốc được kê, các dịch vụ bệnh nhân đã dùng.

- Cập nhật medicine_fee

INSERT INTO pre_medicines (prescription_id, medicine_id, quantity) VALUES (...)
UPDATE histories
*SET medicine_fee = (SELECT SUM(m.unit_price * pm.quantity)*
FROM medicines as m, pre_medicines as pm, histories as h

WHERE pm.prescription_id = h.id and h.id = {id}

and m.medicine_id = pm.medicine_id)

WHERE id = {id};

- Cập nhật treatment_fee

INSERT INTO pre_services (prescription_id, service_id, quantity) VALUES (...)

UPDATE histories

*SET treatment_fee = (SELECT SUM(m.unit_price * pm.quantity)*

FROM services as s, pre_services as ps, histories as h

WHERE ps.prescription_id = h.id and h.id = {id}

and s.services_id = ps.service_id)

WHERE id = {id};

- Cập nhật total_price :

UPDATE histories

SET total_price = (SELECT (medicine_fee+treatment_fee)

FROM histories WHERE id = {id})

where id = {id};

+ Lúc cập nhật thuốc sẽ được gợi ý các thuốc trong kho có số lượng tồn kho lớn hơn 0.

*SELECT * FROM medicines WHERE stock_quantity > 0 ;*

+ Tuy nhiên, khi cập nhật các thuốc đã kê vào bảng pre_medicines có thể xảy ra vấn đề : KHÔNG đủ số lượng thuốc tồn kho để bán cho bệnh nhân.

→ Sử dụng **TRIGGER**. (phần Vũ Tuấn Đạt)

5,6. Kiểm tra số lượng bệnh nhân và ngày nghỉ của bác sĩ

- Một lưu ý khi đặt lịch hẹn và chọn bác sĩ :

Trường hợp 1 : Bác sĩ đó đã quá số lượng bệnh nhân trong ngày (tối đa 5 slots đặt trước)

Kiểm tra số lượng bệnh nhân ngày hôm đó của bác sĩ :

SELECT count(*) FROM histories

WHERE is_done = False and doctor_id = {doctor_id} and date_time = '{date_time}'

Trường hợp 2 : Kiểm tra xem bác sĩ có làm việc ngày hôm đó hay không bằng cách xem ngày nghỉ của bác sĩ đó.

SELECT day_off FROM doctors WHERE doctor_id = {doctor_id}

(Phần kiểm tra còn lại được xử lí trong backend)

D. Một số câu truy vấn khác (không demo trên web) (7 câu)

- Hiển thị tất cả các cuộc hẹn từng có :

*SELECT * FROM histories;*

- Hiển thị tất cả những cuộc hẹn trong một tháng nào đó :

*SELECT * FROM histories WHERE (EXTRACT(month from date_time)) = '{month}'*

- Hiển thị bệnh nhân nào chưa trả tiền và số tiền còn nợ:

```
SELECT p.patient_name, sum(total_price)
FROM patients as p, histories as h
WHERE p.patient_id = h.patient_id and is_paid = False
GROUP BY p.patient_name;
```

- Hiển thị bệnh nhân trong độ tuổi nào nhiều nhất :

```
SELECT concat(DIV(cast(extract(year from age(birthday)) as integer),10)*10,' - ',
((DIV(cast(extract(year from age(birthday)) as integer),10)+1) * 10)) as Age,
count(DIV(cast(extract(year from age(birthday)) as integer),10))
FROM patients
GROUP BY DIV(cast(extract(year from age(birthday)) as integer),10)
ORDER BY count(extract(year from age(birthday))) DESC
```

- Hiển thị bác sĩ có nhiều bệnh nhân nhất trong tháng nào đó :

SELECT doctor_id, doctor_name, num_patient, which_month, which_year

FROM doctors as d, salary_table as s

WHERE (num_patient >= (select max(num_patient) from s)

AND which_month = 1 and which_year = 2021

and d.doctor_id = s.doctor_id)

- Hiển thị bác sĩ được hẹn trước nhiều nhất:

SELECT h.doctor_id, d.doctor_name, count(*) as nop

FROM doctors as d, histories as h

WHERE d.doctor_id = h.doctor_id

and h.is_done = False

GROUP BY h.doctor_id, d.doctor_name

ORDER BY nop DESC LIMIT 1

- Hiển thị bác sĩ thường xuyên thực hiện trồng răng sứ nhất :

SELECT d.doctor_name, count(*) as num

FROM doctors as d, histories as h, pre_services as ps, services as s

WHERE d.doctor_id = h.doctor_id and h.id = ps.prescription_id

and ps.service_id = s.service_id and s.service_name = 'Trồng răng sứ (Implant)'

GROUP BY d.doctor_name

Thành viên : Vũ Tuấn Đạt (tổng cộng: 15 câu)

A. Quản lý bảng salary_table và histories (7 câu)

1,2. Truy cập vào danh sách thu chi

Hệ thống gọi đến hàm **income_expense()** trong file **routes.py**. Hàm này lại gọi đến hàm **query_income(which_month=None,which_year=None)** và hàm

query_expense(which_month=None, which_year=None) trong file **models.py** để thực hiện đồng thời 2 câu truy vấn:

```
SELECT EXTRACT(year from date_time),EXTRACT(month from
date_time),
SUM(medicine_fee),SUM(treatment_fee),SUM(total_price)
FROM histories
GROUP BY EXTRACT(year from date_time), EXTRACT(month from
date_time)
ORDER BY EXTRACT(year from date_time), EXTRACT(month from
date_time) ASC
```

và:

```
SELECT doctor_id,doctor_name,which_year,which_month,salary
FROM salary_table NATURAL JOIN doctors
ORDER BY doctor_id ASC
```

Câu truy vấn thứ nhất gom nhóm và tính tổng số tiền phòng khám thu được từ các khoản + tổng tiền thu được từ khám chữa bệnh (chi phí điều trị), tức là **SUM(treatment_fee)**

+ tổng tiền thu được từ bán thuốc, tức là **SUM(medicine_fee)**

+ tổng cộng số tiền từ tổng cộng của hai loại trên, tức là **SUM(total_price)**

theo tháng và theo năm, đồng thời sắp xếp kết quả theo thứ tự tăng dần (ASC), ưu tiên năm rồi đến tháng.

Câu truy vấn thứ hai giống câu truy vấn hiện ra bảng lương của các bác sĩ, nhưng trong web có thêm chức năng tính tổng cộng số lương này, tùy theo kết quả trả về:

```
total_salary = sum([i['salary'] for i in expense])
```

(trong hàm **income_expense** của file **routes.py**)

3,4. Khi tìm kiếm theo năm hoặc theo tháng

Hệ thống gọi đến hàm **income_expense()** trong file **routes.py**. Hàm này lại gọi đến hàm **query_income(which_month=None,which_year=None)** và hàm

query_expense(which_month=None, which_year=None) trong file **models.py,** nhưng ở đây các đối số which_month và which_year đã khác None.

Giả sử nhấn which_year = 2020. Các câu truy vấn sau sẽ được thực thi:

```
select EXTRACT(year from date_time),
SUM(medicine_fee),SUM(treatment_fee),SUM(total_price) from
histories
group by EXTRACT(year from date_time)
having EXTRACT(year from date_time) = 2020
```

```
SELECT doctor_id,doctor_name,which_year,which_month,salary
FROM salary_table NATURAL JOIN doctors WHERE which_year =
2020
```

Lúc này, các khoản mục tổng số tiền phải trả cho các bác sĩ trong năm 2020 cũng sẽ hiện ra, tiện lợi cho việc đối chiếu.

Trường hợp nhấn tháng thì yêu cầu nhấn cả năm (Vì tháng được xác định trong năm cụ thể).

Ví dụ tháng 12 năm 2020

```
select EXTRACT(year from date_time),EXTRACT(month from
date_time),
SUM(medicine_fee),SUM(treatment_fee),SUM(total_price)
from histories
group by EXTRACT(year from date_time), EXTRACT(month from
date_time)
having EXTRACT(year from date_time) = 2020 and EXTRACT(month
from date_time) = 12
```

```
SELECT doctor_id,doctor_name,which_year,which_month,salary
FROM salary_table NATURAL JOIN doctors
WHERE which_year = 2020 and which_month = 12
```

4,5. Thêm và tự động trừ số lượng tồn kho tương ứng các thuốc được bác sĩ kê

Nhấn vào "Thêm các thuốc được kê trong đơn", hệ thống hiển thị form

Các thuốc đã được kê trong đơn số …

Thuốc được kê

Số lượng

Confirm

Sau khi form được điền và nhấn nút **Confirm (xác nhận)**, hệ thống gọi đến hàm **prescription_medicine_update(*patient_id*, *id*)** trong file **routes.py** để chèn và cập nhật lại bảng **pre_medicines**.

Kết quả: Số tiền dịch vụ (tiền điều trị), tức **treatment_fee** và số tiền tổng cộng **total_price** được cập nhật chính xác và hiển thị vào trong đơn thuốc, cũng như cập nhật vào cơ sở dữ liệu

Khi sự kiện insert loại thuốc & số lượng thuốc vào bảng **pre_medicines** được **kích hoạt** bởi nút confirm, trigger ***quantity_change*** cũng được kích hoạt theo

```
CREATE OR REPLACE FUNCTION change_stock_quantity()
```

```
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS
$$
BEGIN
IF ( ((select stock_quantity from medicines where medicine_id = NEW.medicine_id) - NEW.quantity) < 0) THEN
DELETE FROM pre_medicines WHERE (prescription_id = NEW.prescription_id AND medicine_id = NEW.medicine_id);
RAISE notice 'Not enough! ';
ELSE
UPDATE medicines
SET stock_quantity = stock_quantity - NEW.quantity
WHERE medicine_id = NEW.medicine_id;

END IF;
RETURN NEW;
END;
$$;

CREATE TRIGGER quantity_change
AFTER INSERT ON pre_medicines
FOR EACH ROW
EXECUTE PROCEDURE change_stock_quantity();
```

Trường hợp thứ nhất: Số thuốc nhập vào vượt quá số thuốc hiện có trong kho, hệ thống xóa ngay bản ghi vừa mới được chèn vào (tránh sai sót) và hiển thị thông báo:

```
RAISE notice 'Not enough! ';
```

Trường hợp thứ hai: Số thuốc nhập vào không vượt quá số thuốc hiện có trong kho, hệ thống tiến hành cập nhật

số thuốc tồn kho mới = số thuốc tồn kho hiện tại - số lượng hộp/ vỉ thuốc kê cho bệnh nhân

```
UPDATE medicines
SET stock_quantity = stock_quantity - NEW.quantity
WHERE medicine_id = NEW.medicine_id;
```

Trong trường hợp thứ 2, khi mọi thứ hoạt động trơn tru, sau khi kê đơn, chúng ta vào lại trang **/medicines** sẽ thấy số lượng tồn kho (stock_quantity) giảm đi 1 lượng đúng bằng số hộp/vỉ đã kê

6,7. Xác nhận thanh toán và cập nhật lương cho bác sĩ

Trong mục "xem thông tin chi tiết về Hồ sơ bệnh án", nhấn vào ô Thanh toán, dòng chữ "**Chưa thanh toán"** sẽ biến mất, thay thế bằng dòng chữ "**Đã thanh toán"**, đồng thời đơn thuốc không thể chỉnh sửa được nữa

Khi nhấn nút **Đã thanh toán** này, hệ thống chuyển đổi thuộc tính is_paid từ FALSE thành TRUE. Nó sẽ gọi đến trigger **num_patient_update**

```
CREATE OR REPLACE FUNCTION increase_num_patients()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS
$$
BEGIN
IF (OLD.is_paid = FALSE AND NEW.is_paid = TRUE) THEN
IF(EXTRACT(year from NEW.date_time) NOT IN (SELECT DISTINCT
which_year from salary_table)) THEN

 INSERT INTO salary_table VALUES
```

```
(NEW.doctor_id,EXTRACT(month from NEW.date_time),EXTRACT(year
from NEW.date_time),1,6000000+0.2*NEW.total_price);

ELSIF (EXTRACT(month from NEW.date_time) NOT IN (SELECT
DISTINCT which_month from salary_table WHERE which_year =
EXTRACT(year from NEW.date_time))) THEN
INSERT INTO salary_table VALUES
(NEW.doctor_id,EXTRACT(month from NEW.date_time),EXTRACT(year
from NEW.date_time),1,6000000+0.2*NEW.total_price);
ELSIF (NEW.doctor_id NOT IN (SELECT DISTINCT doctor_id from
salary_table)) THEN
INSERT INTO salary_table VALUES
(NEW.doctor_id,EXTRACT(month from NEW.date_time),EXTRACT(year
from NEW.date_time),1,6000000+0.2*NEW.total_price);
ELSE
UPDATE salary_table
SET num_patient = num_patient + 1,
salary = salary + 0.2*NEW.total_price
WHERE (doctor_id = NEW.doctor_id AND
which_year = EXTRACT(year from NEW.date_time) AND
which_month = EXTRACT(month from NEW.date_time));
END IF;
END IF;
RETURN NEW;
END;
$$;

CREATE TRIGGER num_patient_update
AFTER UPDATE OF is_paid ON histories
FOR EACH ROW
```

```
EXECUTE PROCEDURE increase_num_patients();
```

Trường hợp thứ nhất: Năm diễn ra ngày khám bệnh chưa từng xuất hiện → Hệ thống tự động chèn 1 bản ghi vào bảng salary_table

Trường hợp thứ hai: Năm diễn ra ngày khám bệnh đã diễn ra rồi, nhưng tháng diễn ra ngày khám bệnh chưa từng diễn ra → Hệ thống tự động chèn 1 bản ghi mới vào salary_table

Trường hợp thứ ba: Năm và tháng diễn ra ngày khám bệnh đã diễn ra rồi nhưng bác sĩ thực hiện ca khám bệnh NEW lại thực hiện ca khám bệnh đầu tiên trong tháng đó → Hệ thống tự động chèn 1 bản ghi mới vào salary_table

(Lý do là bởi bảng salary_table phải đủ 3 thuộc tính (doctor_id, which_month, which_year) này mới có thể xác định được mọi thuộc tính trong bảng salary_table)

Trường hợp thứ tư: Khi đã có đủ 3 thuộc tính **doctor_id, which_month, which_year** thì tiến hành update, tăng số bệnh nhân trong tháng của bác sĩ lên thêm 1, đồng thời tăng mức lương của bác sĩ lên 20% nhân total_price của đơn bệnh trên.

Sau khi trở về trang thông tin bệnh nhân, vào lại trang doctors/salary chúng ta nhận thấy có sự thay đổi, về bản ghi mới được thêm vào, hoặc về số lượng bệnh nhân cũng như mức lương của bác sĩ.

B. Một số câu truy vấn viết thêm và không demo trên web (8 câu)

1,2. Lựa chọn N tháng có thu nhập vào cao nhất, thấp nhất

```
select EXTRACT(month from date_time), EXTRACT(year from date_time), SUM(total_price)
from histories
group by EXTRACT(month from date_time), EXTRACT(year from date_time)
order by SUM(total_price) DESC/ASC
limit N
```

--DESC: dùng khi thu nhập cao nhất, ASC: dùng khi thu nhập thấp nhất

3,4. Lựa chọn N tháng có lương trả cho các bác sĩ nhiều nhất, ít nhất

```
select which_month, which_year, sum(salary)
from salary_table
group by which_month, which_year
order by sum(salary) DESC/ASC
limit N
```

5. Số bệnh nhân đến khám vì bệnh ... (VD: sâu răng) (note: ghi chú khám bệnh)

```
select count(patient_id)
from histories
where note like '%sâu răng%' or note like '%Sâu răng%'
```

6. Tên bệnh nhân, mã số đơn thuốc và số điện thoại của N bệnh nhân mà đã hoàn thành thanh toán, và có tỉ lệ tiền chi trả cho thuốc lớn hơn 50% trong tổng số tiền đã trả

```
select distinct patient_name,id,medicine_fee
from patients as p NATURAL JOIN histories
where (is_paid = TRUE) AND medicine_fee/total_price >= 0.5
order by medicine_fee DESC
limit 5
```

7. Chính sách đề xuất: Trong số những bệnh nhân đã thanh toán, người nào có tổng tiền thanh toán lớn hơn bằng 20 triệu thì chiết khấu 10%, nhỏ hơn 20 triệu lớn hơn 10 triệu chiết khấu 5%. Nêu tên, số lần khám bệnh và số điện thoại của người đó. (dùng view …)

(discount (v) chiết khấu, giảm giá)

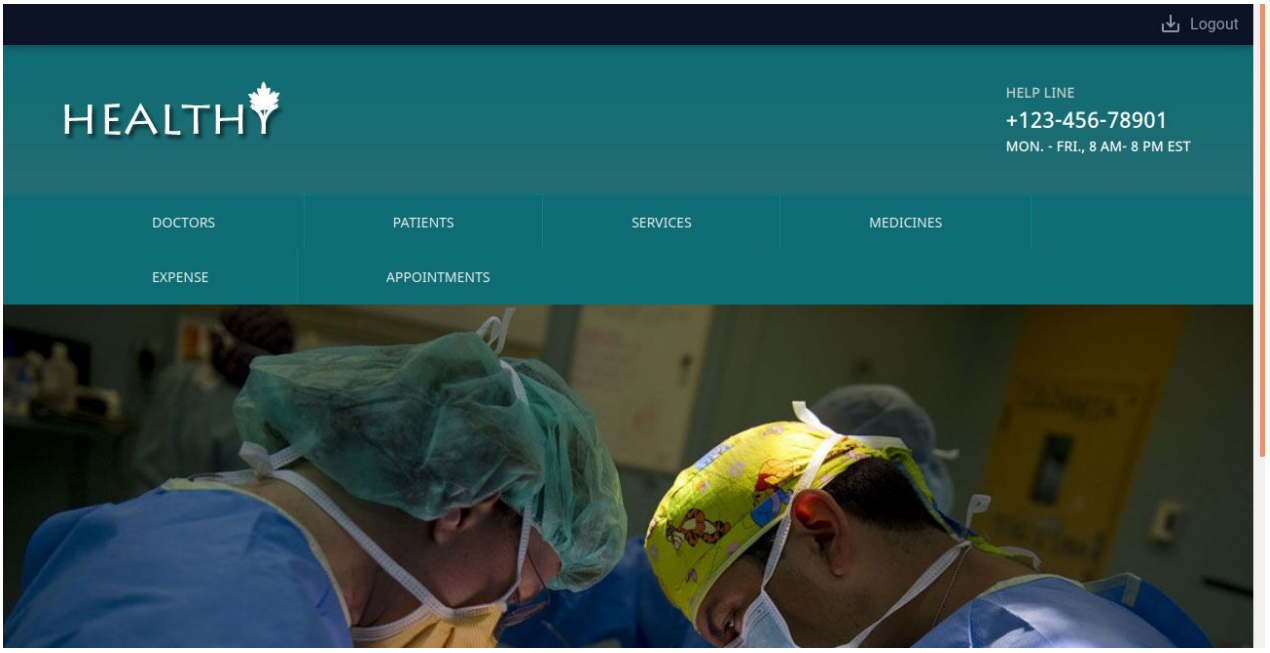
```
create view discount as
    select distinct patient_name,total_price,phone_number,
    CASE
            WHEN total_price >= 20000000 THEN 0.1
```

```
                WHEN total_price < 20000000 AND total_price >=
10000000 THEN 0.05
            ELSE 0
        END discount_rate
    from patients NATURAL JOIN histories
    where is_paid = true;
select * from discount;
```

8. Phân loại bệnh nhân (Classification): Đưa ra tên, ngày sinh, tuổi và phân loại của bệnh nhân, theo tiêu chí: 0-14 tuổi: Kids, 15-30: Young adults, 30-60: Middle ages, lớn hơn 60: elderly

```
SELECT patient_name,birthday,extract(year from age(birthday)),
       CASE
           WHEN extract(year from age(birthday)) > 0
                   AND extract(year from age(birthday)) <= 14
THEN 'Kids'
           WHEN extract(year from age(birthday)) > 14
                   AND extract(year from age(birthday)) <= 30
THEN 'Young Adults'
           WHEN extract(year from age(birthday)) > 30
         AND extract(year from age(birthday)) <= 60 THEN
'Middle Ages'
        ELSE 'Elderly'
       END classification
FROM patients
```

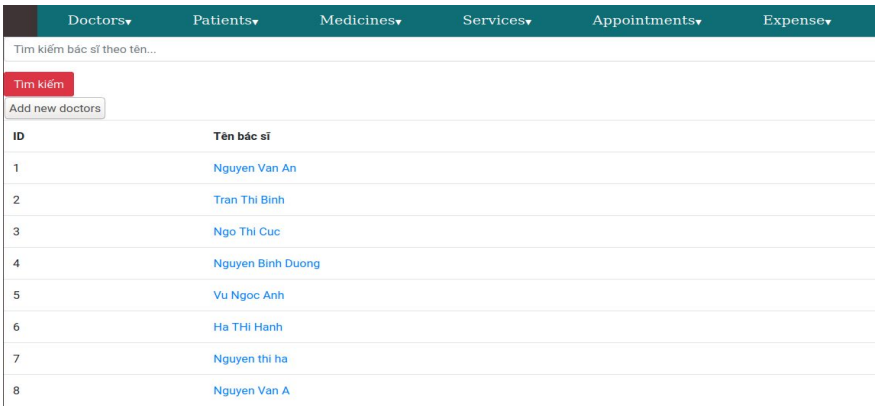
IV. Giao diện web



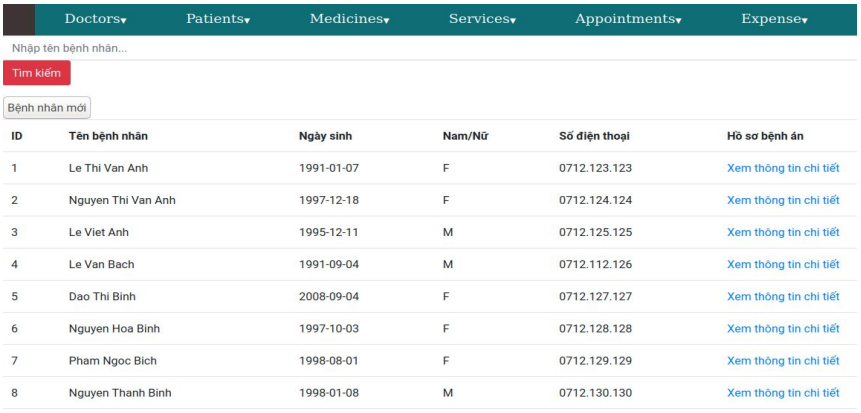
Hình 3.1: Giao diện trang chủ

Hình 3.2: Đăng nhập

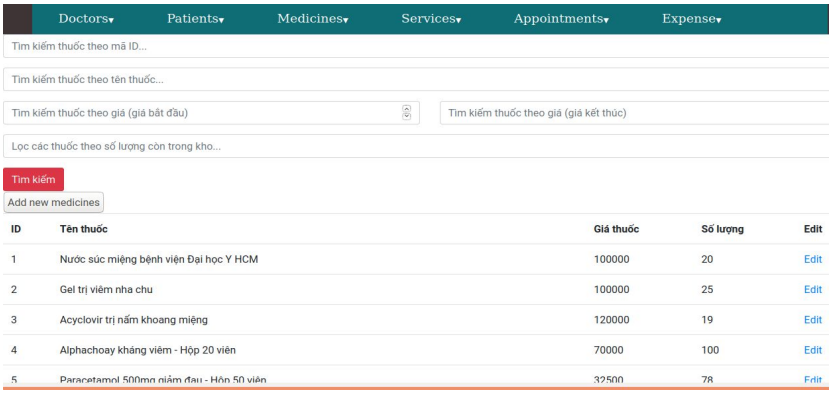
Hình 3.3: Đăng ký



Hình 3.4: Quản lý bác sĩ

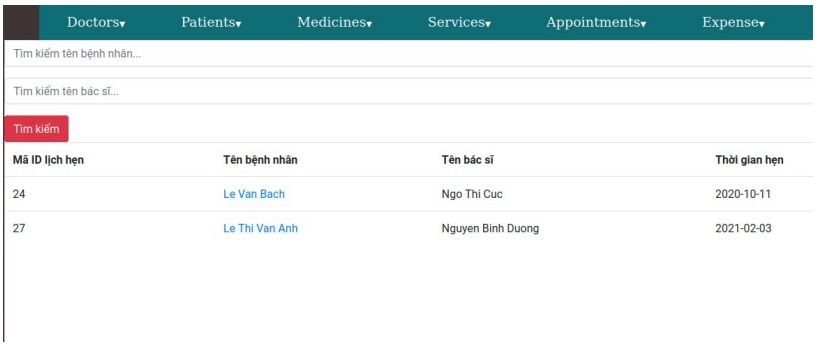


Hình 3.5: Quản lý bệnh nhân



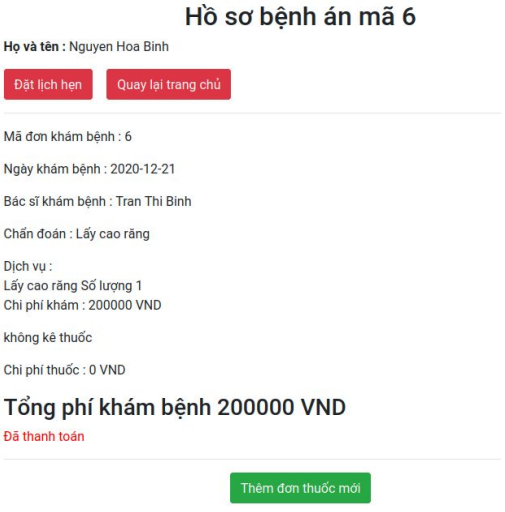
Hình 3.6: Quản lý thuốc

Hình 3.7: Quản lý dịch vụ

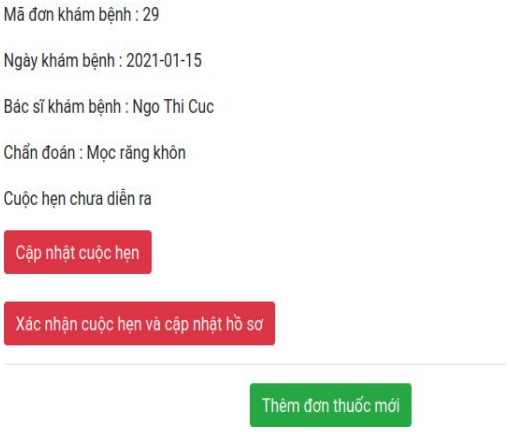


Hình 3.8: Quản lý lịch hẹn

Hình 3.9: Quản lý thu chi

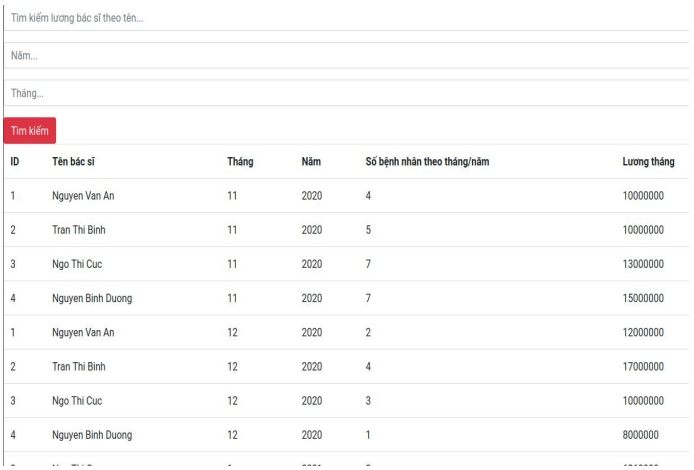


Hình 3.10: Hồ sơ bệnh án



Hình 3.11: Lịch khám bệnh

Hình 3.12: Đặt lịch



Hình 3.13: Bảng lương của bác sĩ

V. Đánh giá và nhận xét

1. Về những gì đã làm được

+ Thông qua mini project môn thực hành cơ sở dữ liệu lần này, chúng em đã có thể hiểu được cách thiết kế và xử lý một bài toán trong thực tế, dùng những thuộc tính để mô tả chính xác và đầy đủ một thực thể, một liên kết.

Đồng thời sử dụng các chuẩn (mà cụ thể ở trường hợp đưa thuộc tính salary đa trị ban đầu về chuẩn 1, loại bỏ quan hệ nhiều nhiều giúp tăng tốc độ truy xuất bằng cách tạo ra hai bảng pre_medicines và pre_services) và các kiến thức về mối liên kết 1-nhiều, nhiều - nhiều để đưa từ sơ đồ thực thể liên kết về sơ đồ quan hệ. Từ đó áp dụng được những kiến thức về thao tác dữ liệu (data manipulation) để thực hiện truy vấn.

+ Đồng thời, về back-end, chúng em đã học được nhiều về ngôn ngữ python, cú pháp cơ bản & 1 số thư viện hỗ trợ và framework hỗ trợ làm web của nó, **Flask.** Về front-end, chúng em có được kiến thức cơ bản về html cũng như javascript.

2. Về hạn chế

+ Chưa thực hiện được quản lý phiên đăng nhập cho người sử dụng hệ thống (ở đây là người quản lý)

+ Chưa thực hiện được việc phân quyền cho người dùng trong trường hợp người dùng là nhiều đối tượng khác nhau. Ở đây, chúng em mới chỉ dừng lại ở việc người dùng là người quản lý phòng khám, có quyền thêm nhập sửa xóa đến tất cả các bảng.

3. Về hướng phát triển và hoàn thiện thêm

+ Phát triển phiên đăng nhập cho người dùng (nhiều đối tượng)

+ Thực hiện phân quyền cho người dùng (sử dụng INVOKE và GRANT) (ví dụ như bác sĩ có quyền đổi lịch hẹn cho 1 bác sĩ khác (quyền UPDATE) trong trường hợp người bác sĩ đó có tang ma hiếu hỉ đột xuất nhưng bác sĩ lại không được quyền chỉnh sửa đơn thuốc đã kê hay tự chỉnh sửa tiểu sử & học vấn của chính mình, …)