TRƯỜNG ĐẠI HỌC SÀI GÒN

Khoa Công nghệ thông tin

BÁO CÁO MÔN HỌC

HỌC PHẦN: CƠ SỞ DỮ LIỆU NÂNG CAO

ĐỀ TÀI: QUARANTINE CAMP

Giáo viên hướng dẫn: Phan Trọng Nhân

Họ và Tên: Mai Nguyễn Trung Kiên

Mã số sinh viên: 3121411112

Lớp: DCT121C3

TP.Hồ Chí Minh, Tháng 12 Năm 2023

Lời mở đầu

Trong bối cảnh dịch bệnh COVID-19 diễn biến phức tạp, việc cách ly người nhiễm bệnh là một biện pháp quan trọng để ngăn chặn sự lây lan của dịch bệnh. Các trại cách ly cần được quản lý một cách khoa học và hiệu quả để đảm bảo sức khỏe và an toàn cho người cách ly.

Cơ sở dữ liệu (CSDL) là một công cụ quan trọng giúp quản lý các thông tin về người cách ly, nhân viên, phương tiện vận chuyển, v.v. của một trại cách ly. CSDL được thiết kế tốt sẽ giúp trại cách ly thực hiện các nhiệm vụ sau một cách hiệu quả:

Quản lý thông tin về người cách ly, bao gồm thông tin cá nhân, thông tin sức khỏe, thông tin lịch sử tiếp xúc, v.v.

Quản lý thông tin về nhân viên, bao gồm thông tin cá nhân, thông tin đào tạo, thông tin lịch sử công tác, v.v.

Quản lý thông tin về phương tiện vận chuyển, bao gồm thông tin phương tiện, thông tin lịch sử hoạt động, v.v.

Môn Cơ sở dữ liệu nâng cao cung cấp cho sinh viên các kiến thức và kỹ năng cần thiết để thiết kế và triển khai một CSDL quan hệ. Các vấn đề đặt ra trong các giai đoạn thiết kế, từ thiết kế cấu trúc quan niệm đến thiết kế cấu trúc vật lý được trình bày rất chi tiết trong môn học này.

Trên cơ sở kiến thức của môn học, nhóm chúng em đã được giao đề tài "Xây dựng CSDL Quản lý Trại Cách ly" để củng cố kiến thức và hiện thực hóa những gì đã được thu nhận thông qua môn học.

Đề tài này sẽ tập trung vào việc xây dựng một CSDL quản lý các thông tin về người cách ly, nhân viên, v.v. của một trại cách ly. CSDL được thiết kế theo mô hình quan hệ, đáp ứng các yêu cầu khai thác, sử dụng của trại cách ly.

Trong quá trình thực hiện, nhóm chúng em đã cố gắng tìm hiểu và áp dụng những kiến thức đã được học để hoàn thành đề tài một cách tốt nhất. Tuy nhiên, vẫn còn có thể tồn tại một số sai sót, nhóm chúng em rất mong nhận được sự góp ý của thầy để bài làm được hoàn thiện hơn.

Mục Lục

I/ Phân tích chung

1) Đặc tả bài tập

Em tiếp nhận đề tài và tiến hành phân tích, đề tài của em có đặc tả như sau:

Due to the Covid-19 outbreak, a quarantine camp has been set up to isolate and monitor people under investigation for 21 days. Those people admitted to the quarantine camp are called “patient”. The camp stores patient information including unique number, full name, identity number, phone, gender, and address. In addition, it wants to record the patient comorbidities (e.g., cancer, chronic lung diseases, diabetes, heart conditions, immunocompromised state) because they will put a patient in a high-risk situation. In parallel, a patient needs to be tracked with his or her symptoms such as fever, dry cough, tiredness, aches and pains, sore throat, diarrhoea, conjunctivitis, headache, loss of taste or smell, a rash on skin, or discoloration of fingers or toes. Some of them may be serious like difficulty breathing or shortness of breath, chest pain or pressure, and loss of speech or movement. Unlike comorbidity, a patient symptom is different from time to time.

The camp has different types of people: managers, doctors, nurses, staffs, and volunteers. One doctor will be designated as the head of the camp. Each has its own responsibility. Besides, the camp has several buildings, each has many floors and rooms. Each room has a limited capacity. There are three types of room: normal room, emergency room, and recuperation room. When admitted by a staff, a patient is assigned into a room based on his or her current condition. Sometimes, a patient is moved from his or her room to the emergency room or the recuperation room. So, it is important to track a patient location history. The camp needs to know the admission date, from where the patient is moved to the camp, the staff information, and the testing information if any. A staff may admit many patients, and a patient is admitted by a staff.

The testing information includes those as described below:

- PCR test: the result is true (positive) or false (negative). In case it is positive, the camp wants to track the corresponding cycle threshold (ct) value.

- Quick test: the result is true (positive) or false (negative). In case it is positive, the camp wants to track the corresponding cycle threshold (ct) value.

• SPO2: which is the percent saturation of oxygen in the blood. The test measures blood oxygen levels, indicated by percentage (%).

• Respiratory rate: it is measured by how many breaths per minute.

A patient may have many testing during his or her stay. If the SPO2 is smaller than 96% and the respiratory rate is larger than 20 breaths per minute, the patient is marked "warning" and needs a healthcare action from the doctors. In case the patient has no clinical sign and the test is either negative or positive whose cycle threshold is larger than 30, he or she will be discharged from the camp. Neither of them, the patient will be tested for every 3 days by Quick test. It is important to track the discharge date for each patient.

A patient can receive treatment from at least one doctor. A doctor can treat many patients at the same time, or sometimes, he has no patients to treat. The camp needs the details of each treatment such as: treatment period (start date and end date), result, and medications. Each patient is taken care of by a nurse; a nurse can take care of many inpatients at the same time. The information of a medication is also stored in the database. This information consists of a unique code, name of the medication, effects, price, and expiration date.

2) Yêu cầu của bài tập

2.1) Yêu cầu chung

1. Design a fully labelled (E)ERD according to your business description. The diagram has to show appropriate entities (with key attributes underlined), relationships, cardinality ratios, and optional & mandatory membership classes (3 points).

2. Mapping your (E)ER diagram above to a relational database schema and identify all constraints not shown in your (E)ER diagram (1 point).

3. Build an application with the following requirements (6 points):

- Programming environment: optional (desktop, web, or mobile application).

- Programming language: optional.

- Students need to prepare data and scripts for demonstration at the reporting session.

2.2) Yêu cầu riêng cho đề tài Quarantine Camp

1. Search patient information: Search results include the name, phone number and information about his/her comorbidities (1 point).

2. Add information for a new patient (1 point).

3. List details of all testing which belong to a patient (1 point).

4. Make a report that provides full information about the patient including demographic information, comorbidities, symptoms, testing, and treatment (1 point).

5. Proving one use-case of indexing efficiency in your scenarios (1 point)

6. Solving one use-case of database security in your scenarios (1 point)

3) Công việc được phân công

-Database:

+Phân tích yêu cầu.

+Xây dựng lược đồ ERD

+Chuyển đổi mô hình ERD thành mô hình ERM

+Tạo bảng trong cơ sở dữ liệu SQL server

+Insert dữ liệu vào các bảng trong cơ sở dữ liệu

-Website:

+Xây dựng mô hình MVC

+Database:

Xây dựng config cho website có thể kết nối với cơ sở dữ liệu.

Kết nối Website với cơ sỡ dữ liệu để truy vấn thông tin.

Xây dựng phương thức thao tác với dữ liệu .

+Font end:

Xây dựng trang báo lỗi để thông báo tới người dùng.

Kiểm tra lỗi font end.

+Back end:

Xử lý url.

Đi đường dẫn url cho từng trang trong website.

Xây dựng models để có thể thao tác với dữ liệu trong bảng.

Xử lý tìm kiếm bệnh nhân theo tên hoặc unique number.

Xử lý dữ liệu trong form thêm bênh nhân.

Xử lý dữ liệu để đẩy lên một số trang như home, patient, detail test table, room, form.

II/ LƯỢT ĐỒ THỰC THỂ KẾT HỢP (ERD)

1. Khái niệm

Mô hình quan hệ thực thể (Entity Relationship model - E-R) [1] là một mô hình được sử dụng rộng rãi trong các bản thiết kế cơ sở dữ liệu ở mức khái niệm, được xây dựng dựa trên việc nhận thức thế giới thực thông qua tập các đối tượng được gọi là các thực thể và các mối quan hệ giữa các đối tượng này.

2. Các thực thể (Entities)

Sau khi phân tích đề tài em có thành lập được những thực thể như sau:

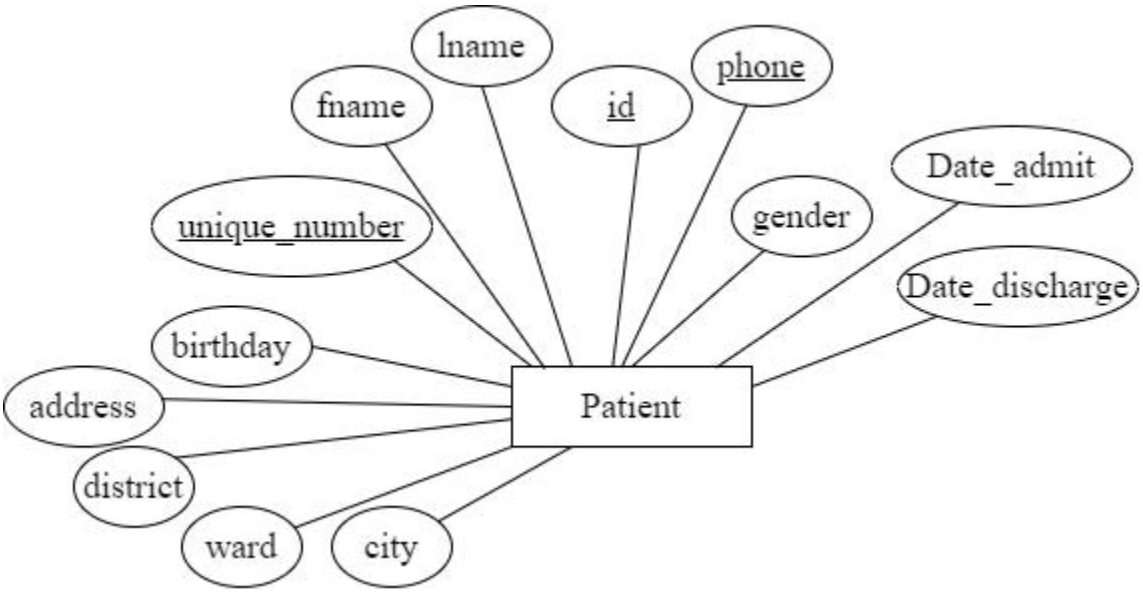
- Patient: Lưu trữ thông tin của bênh nhân được cách ly, bao gồm unique number, first name, last name, indentify number, phone, gender, birthday, address, date admit, date discharge.
- Comorbidity: Lưu trữ thông tin của bệnh nền, bao gồm id, name, describle.
- Symptom: Lưu trữ thông tin của triệu chứng, bao gồm id, name, describle.
- Building: Lưu trữ thông tin tòa nhà, bao gồm id, name, address.
- Floor: Lưu trữ thông tin tầng trong tòa nhà, bao gồm id, name.
- Room: Lưu trữ thông tin phòng của tòa nhà, bao gồm id, name, qty.

- Staff: Lưu trữ thông tin của nhân viên, bao gồm unique number, phone, indentify number, first name, last name, gender, birthday.
- Nurse: Lưu trữ thông tin của y tá, bao gồm unique number, phone, indentify number, first name, last name, gender, birthday.
- Doctor: Lưu trữ thông tin của bác sĩ, bao gồm unique number, phone, indentify number, first name, last name, gender, birthday.
- Medication: lưu trữ thông tin thuốc, bao gồm, id, name, price, effect, expiration date, result.
- Warning: lưu trữ thông tin của bệnh nhân trong tình trành nguy cấp, bao gồm id, date, reason, status.
- Testing: Lưu trữ thông tin xét nghiệm của bệnh nhân, bao gồm, id, date, result, ct_value, spo2_level, respiratory_rate.

3. Phân tích các thực thể và thuộc tính

3.1) Thực thể patient

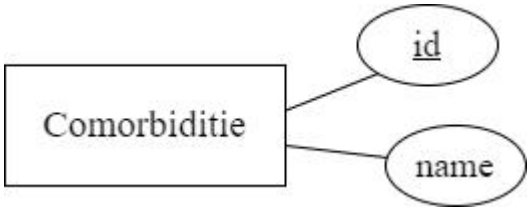
Thực thể Patient có các thuộc tính: *unique_number*, fname, lname, id, phone, gender, birthday, address, dateAdmit, dateDischarge.



thực thể Patient

3.2) Thực thể Comorbidity

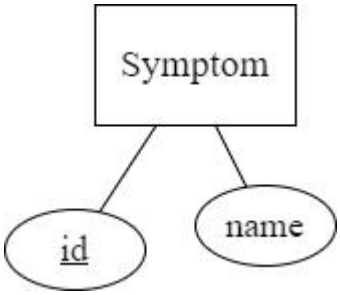
Thực thể Comorbidity có các thuộc tính: *id*, name, describle.



Thực thể Comorbidity

3.3) Thực thể Symptom

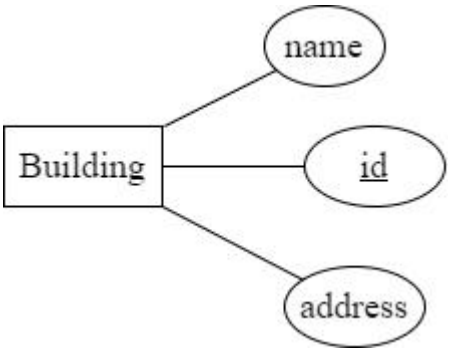
Thực thể Symptom có các thuộc tính: *id*, name, describle.



Thực thể Symptom

3.4) Thực thể Building

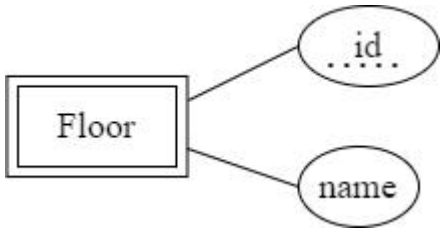
Thực thể Building có các thuộc tính: *id*, name, address



Thực thể Building

3.5) Thực thể Floor

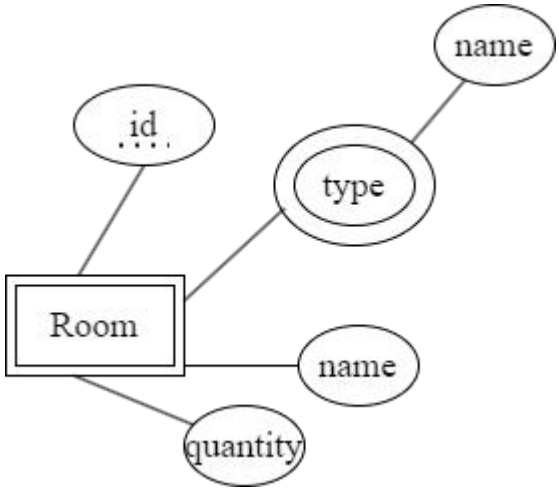
Thực thể Floor có các thuộc tính: *id*, name



Thực thể Floor

3.6) Thực thể Room

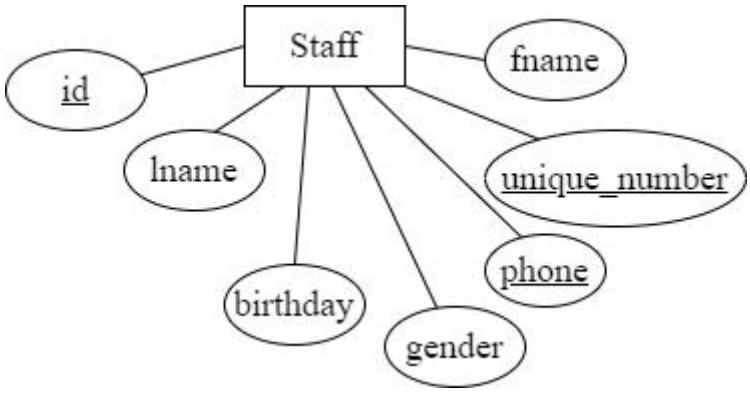
Thực thể Room có các thuộc tính: *id*, name, qty



Thực thể Room

3.7) Thực thể Staff

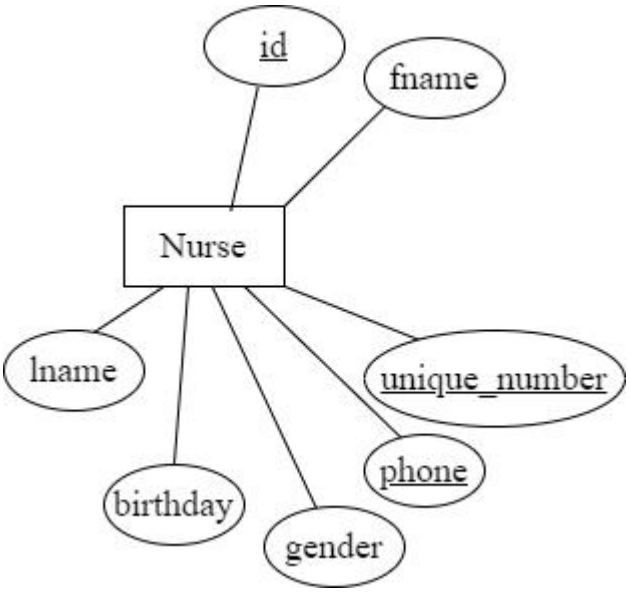
Thực thể Staff có các thuộc tính: *unique_number*, phone, id, fname, lname, gender, birthday



Thực thể Staff

3.8) Thực thể Nurse

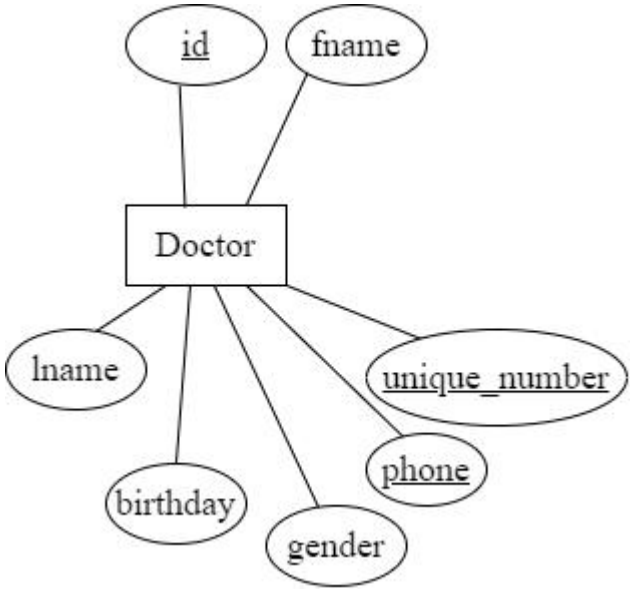
Thực thể Nurse có các thuộc tính: *unique_number*, phone, id, fname, lname, gender, birthday



Thực thể Nurse

3.9) Thực thể Doctor

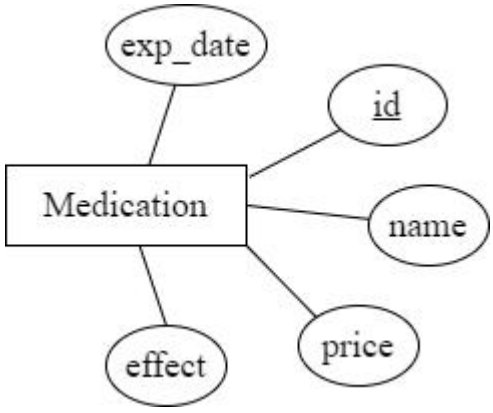
Thực thể Doctor có các thuộc tính: *unique_number*, phone, id, fname, lname, gender, birthday



Thực thể Doctor

3.10) Thực thể Medication

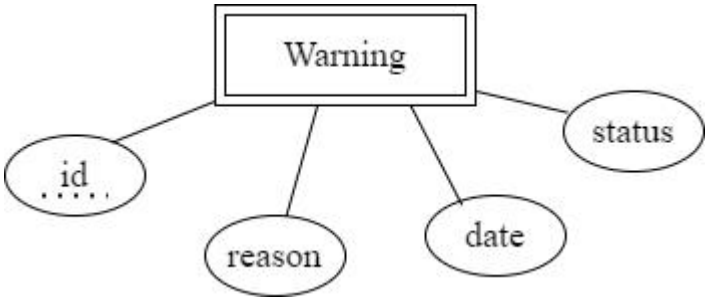
Thực thể Medication có các thuộc tính: *id*, name, price, effect, expiration date, result



Thực thể Medication

3.11) Thực thể Warning

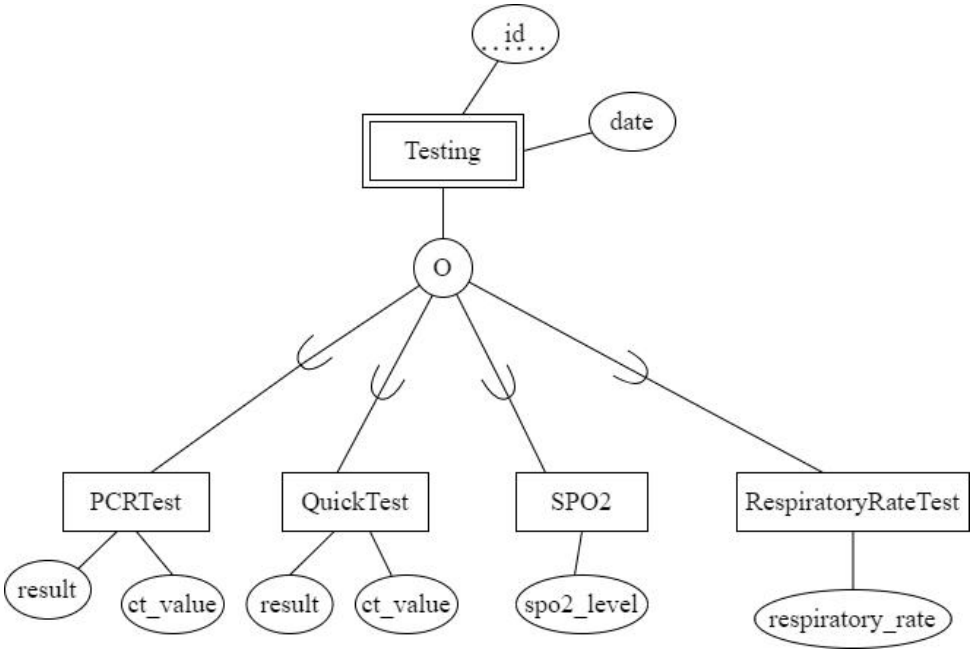
Thực thể Warning có các thuộc tính: *id*, date, reason



Thực thể Warning

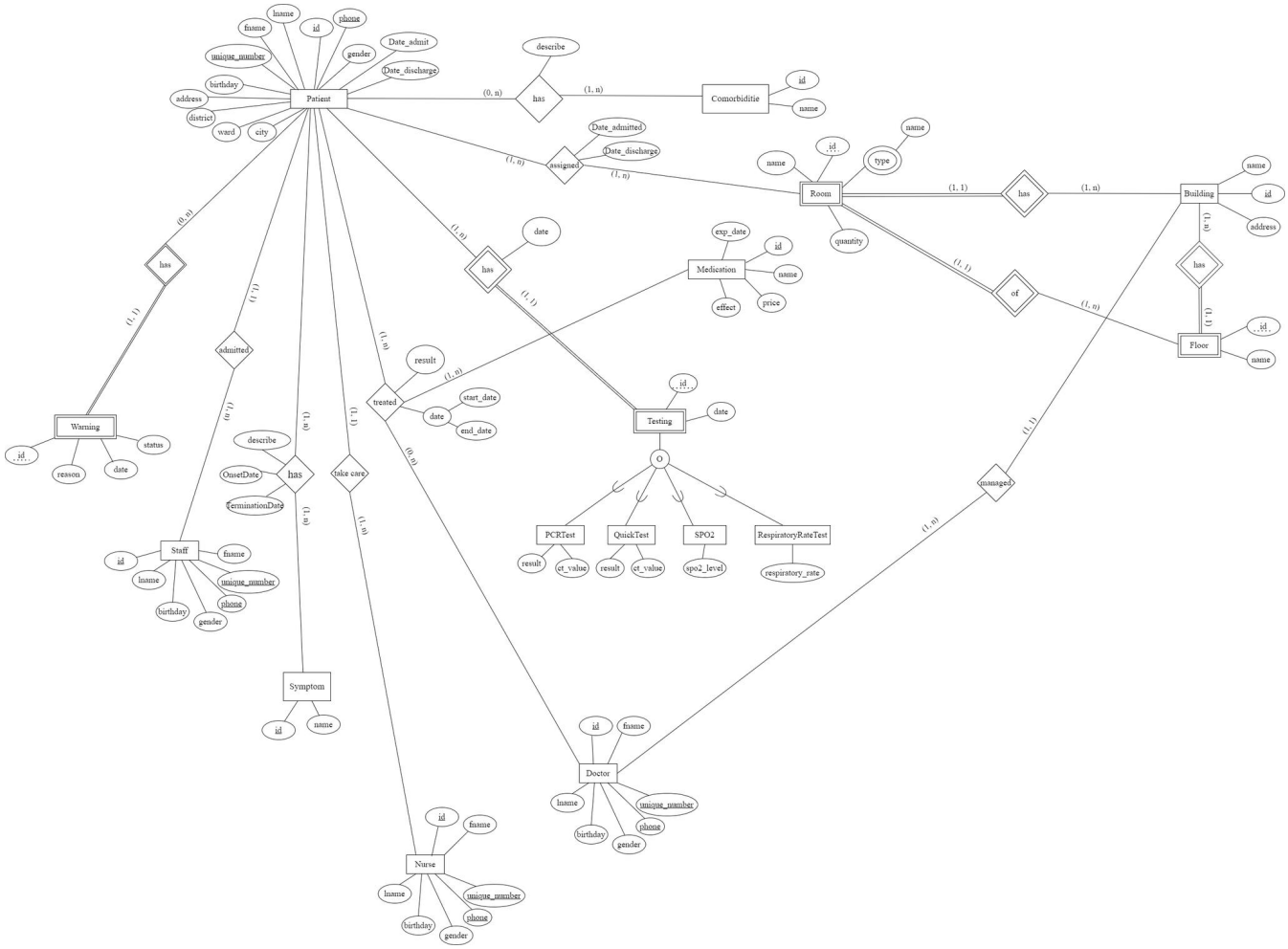
- **3.12) Thực thể Testing**

  Thực thể Testing có các thuộc tính: *id*, date, result, ct_value, spo2_level, respiratory_rate



Thực thể Testing

4.Lượt đồ thực thể kết hợp (ERD) toàn bộ hệ thống



lượt đồ ERD 'Quarantine Camp'.

III/ LƯỢT ĐỒ QUAN HỆ

1. Khái niệm

[2]Lượt đồ quan hệ là một biểu đồ biểu diễn các tập thực thể, thuộc tính và mối quan hệ giữa các tập thực thể. Lượt đồ quan hệ là một công cụ quan trọng trong thiết kế cơ sở dữ liệu quan hệ.

Mỗi tập thực thể trong lượt đồ quan hệ được biểu diễn bằng một hình tròn hoặc hình bầu dục. Mỗi thuộc tính của một tập thực thể được biểu diễn bằng một đường đi từ hình tròn hoặc hình bầu dục đến một nhãn văn bản. Mỗi mối quan hệ giữa hai tập thực thể được biểu diễn bằng một đường thẳng nối hai hình tròn hoặc hình bầu dục.

Lượt đồ quan hệ được sử dụng để mô hình hóa các mối quan hệ giữa các tập thực thể trong một cơ sở dữ liệu quan hệ. Lượt đồ quan hệ giúp các nhà thiết kế cơ sở dữ liệu hiểu rõ mối quan hệ giữa các dữ liệu và xác định các bảng dữ liệu cần thiết để lưu trữ dữ liệu.

2. Lượt đồ quan hệ

Từ lượt đồ ERD ‘Quarantine Camp’ em đã có được lượt đồ quan hệ như sau:

Patient(*unique_number*, phone, id, fname,lname gender, birthday, *nurse_id*, *staff_id*, Address,dateAdmit,dateDischarge)

Comorbidity(*id*, name)

Symptom(*id*, name)

PatientSymptom(***symptom_id,patient_id,*** describe, OnsetDate , TerminationDate)-

PatientComorbidity(***patient_id, comorbidity_id,*** describe*)*

Building(*id*, name, address,*doctor_id*)

Floor(*id*, *building_id*, name)

Room(*id*, *building_id*, *floor_id*, name)

typeRoom(***room_id,*** *id*, name)

Staff(*unique_number,* phone, id, fname,lname, gender, birthday)

PatientHistoryLocation(***patient_id, room_id,*** Date_admitted, Date_discharge)

Nurse(*unique_number,* phone, id, fname,lname, gender, birthday)

Doctor(*unique_number,* phone, id, fname, lname, gender, birthday)

Medication(*id*, name, price, effect, exp_date)

Treatment(***patient_id, doctor_id,medication_id*** start_date, end_date, result)

Warning(*id*, *patient_id*, date,reason)

PCRTest(*id*, *patient_id*, date, result, ct_value)

QuickTest(*id*, *patient_id*, date, result, ct_value)

SPO2Test(*id*, *patient_id*, date,spo2_level)

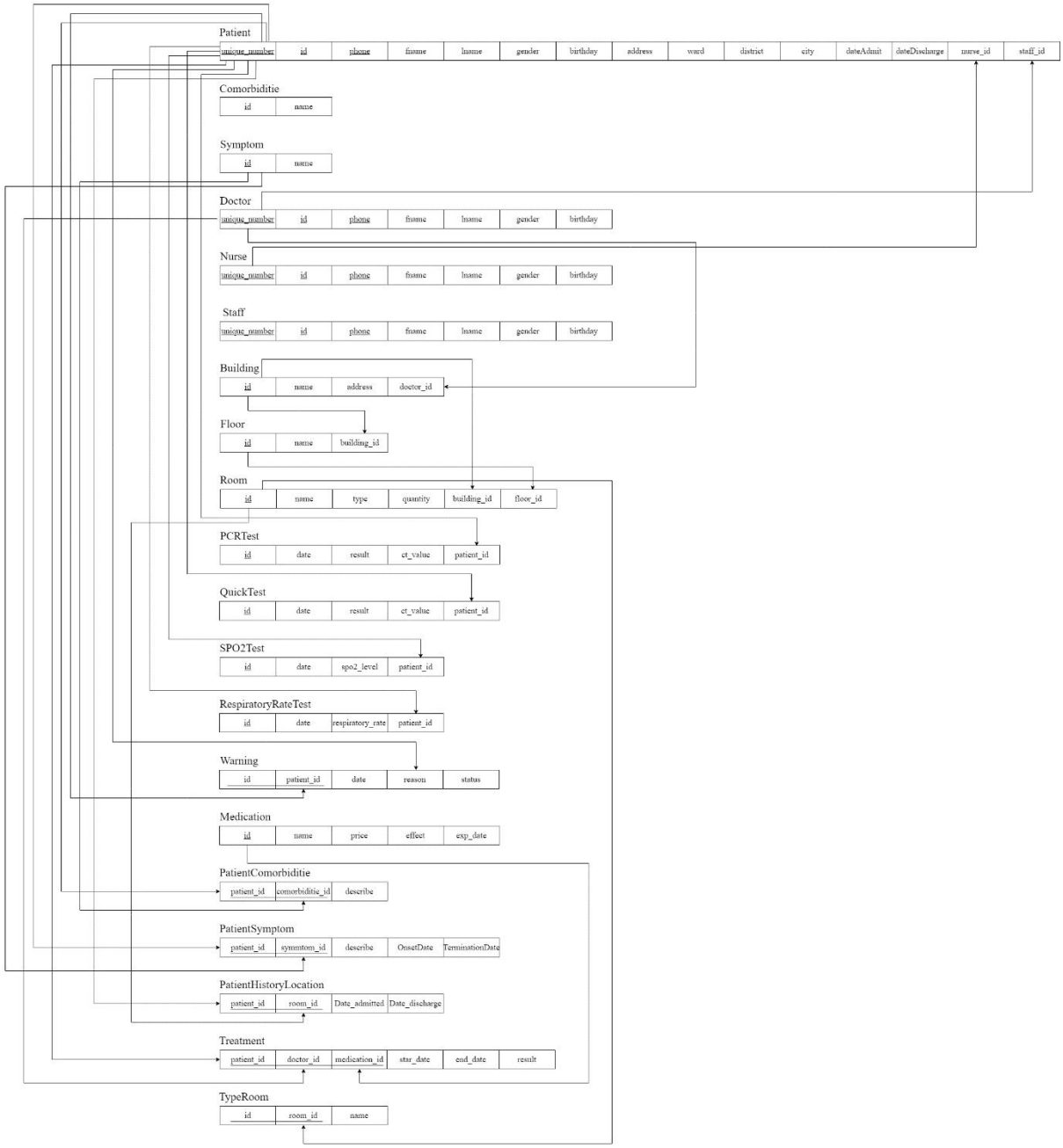
RespiratoryRateTest(*id*, *patient_id*, date,respiratory_rate)

Note:

Text : primary key

Text *:* primary key and foreign key

Text : foreign key


Patient
unique_number
id
phone
fname
lname
gender
birthday
address
ward
district
city
dateAdmit
dateDischarge
nurse_id
staff_id
Comorbiditie
id
name
Symptom
id
name
Doctor
unique_number
id
phone
fname
lname
gender
birthday
Nurse
unique_number
id
phone
fname
lname
gender
birthday
Staff
unique_number
id
phone
fname
lname
gender
birthday
Building
id
name
address
doctor_id
Floor
id
name
building_id
Room
id
name
type
quantity
building_id
floor_id
PCRTest
id
date
result
ct_value
patient_id
QuickTest
id
date
result
ct_value
patient_id
SPO2Test
id
date
spo2_level
patient_id
RespiratoryRateTest
id
date
respiratory_rate
patient_id
Warning
id
patient_id
date
reason
status
Medication
id
name
price
effect
exp_date
PatientComorbiditie
patient_id
comorbiditie_id
describe
PatientSymptom
patient_id
symmtom_id
describe
OnsetDate
TerminationDate
PatientHistoryLocation
patient_id
room_id
Date_admitted
Date_discharge
Treatment
patient_id
doctor_id
medication_id
star_date
end_date
result
TypeRoom
id
room_id
name

Mapping

IV/XÂY DỰNG CƠ SỞ DỮ LIỆU QUARANTINE CAMP

1. Xây dựng bảng trong Microsoft SQL server

1.1) Table Patient

Dùng để lưu trữ thông tin bệnh nhân.

Đây là đoạn code tạo table Patient trong Microsoft SQL server với khóa chính là unique_number, id và phone là hai khóa dự tuyển.

```
CREATE TABLE dbo.Patient (
  unique_number VARCHAR(20) NOT NULL PRIMARY KEY,
 nurse_id VARCHAR(20) NOT NULL ,
 staff_id VARCHAR(20) NOT NULL ,
  id VARCHAR(20) NOT NULL UNIQUE,
  phone VARCHAR(20) NOT NULL UNIQUE,
  fname NVARCHAR(255) NOT NULL,
  lname NVARCHAR(255) NOT NULL,
  gender NVARCHAR(10) NOT NULL CHECK (gender IN ('Male',
'Female')),
  birthday DATE NOT NULL,
  address NVARCHAR(255) NOT NULL,
  ward NVARCHAR(50) NOT NULL,
  district NVARCHAR(50) NOT NULL,
  city NVARCHAR(50) NOT NULL,
  dateAdmit DATE,
 FOREIGN KEY (nurse_id) REFERENCES Nurse(unique_number),
FOREIGN KEY (staff_id) REFERENCES Staff(unique_number)
);
```

Kiểu dữ liệu của các trường dữ liệu của table Patient:

Data Field Name	Data Type
unique_number	VARCHAR(20)
nurse_id	VARCHAR(20)
staff_id	VARCHAR(20)
id	VARCHAR(20)
phone	VARCHAR(20)
fname	NVARCHAR(255)
lname	NVARCHAR(255)
gender	NVARCHAR(10)
birthday	DATE
address	NVARCHAR(255)
ward	NVARCHAR(50)
district	NVARCHAR(50)
city	NVARCHAR(50)
dateAdmit	DATE
dateDischarge	DATE

Ví dụ dữ liệu của bảng:

Results Messages

	unique_number	id	phone	fname	lname	gender	birthday	address	ward	district	city	dateAdmit	dateDischarge	nurse_id	staff_id
1	PT00001	087355369113	0327724061	Mỹ Anh	Trần	Female	2002-07-16	123 Lê Văn Sỹ	10	3	Hồ Chí Minh	2023-11-17	2023-11-27	NR0000	ST0000
2	PT00002	046280794900	0368950878	Minh Anh	Huỳnh	Male	1980-06-11	2/49 Cao Thắng	4	3	Hồ Chí Minh	2023-11-14	2023-11-24	NR0000	ST0000
3	PT00003	046247363471	0372240869	Anh Huy	Huỳnh	Male	1985-01-20	274 An Dương Vương	5	5	Hồ Chí Minh	2023-11-10	NULL	NR0001	ST0001
4	PT00004	024385040439	0583222646	Mỹ Anh	Lê	Female	1991-04-03	42/7 Lê Thánh Tông	Bến Nghé	1	Hồ Chí Minh	2023-10-09	2023-11-16	NR0001	ST0001
5	PT00005	046219629423	0780233385	Trí Tâm	Hồ	Male	1991-07-24	56 Trần Bình Trọng	Tân Định	1	Hồ Chí Minh	2023-11-02	NULL	NR0002	ST0002
6	PT00006	002367531779	0872028042	Thanh Bình	Lê	Male	1995-02-15	126/127 Trần Đình Xu	9	10	Hồ Chí Minh	2023-10-11	2023-10-21	NR0002	ST0002
7	PT00007	079343567401	0920602963	An Khang	Võ	Male	2000-06-29	111/112 Cao Thắng	2	3	Hồ Chí Minh	2023-10-22	2023-11-01	NR0003	ST0003
8	PT00008	042366364941	0336997003	Thịnh Vượng	Nguyễn	Male	2005-10-13	34 An Dương Vương	3	3	Hồ Chí Minh	2023-11-17	NULL	NR0003	ST0003
9	PT00009	020340032325	0354890281	Quang Anh	Trần	Male	2010-01-27	45/7 Lê Văn Sỹ	Cô Giang	1	Hồ Chí Minh	2023-11-17	2023-11-27	NR0004	ST0004
10	PT00010	002318274759	0321413803	Thu Hà	Lê Thị	Female	2015-05-04	Số 1221, Tầng 1, Lô B Chung cư Đào Duy Từ	10	10	Hồ Chí Minh	2023-10-11	2023-10-22	NR0004	ST0004
11	PT00011	067341021710	0764050107	Mỹ Mỹ	Trần Ngọc	Female	2000-11-20	37/9A Cao Thắng	5	3	Hồ Chí Minh	2023-10-04	2023-10-17	NR0005	ST0005
12	PT00012	080217259864	0580310470	Đình Huy	Nguyễn	Male	1997-02-03	12A Hùng Vương	Đa Kao	1	Hồ Chí Minh	2023-11-11	2023-11-21	NR0005	ST0005
13	PT00013	040316320671	0588217963	Mỹ Ngọc	Vũ	Female	1987-12-12	79/12/1 Trần Bình Trọng	11	10	Hồ Chí Minh	2023-11-11	NULL	NR0006	ST0006
14	PT00014	001357101762	0932744345	Ngọc Ánh	Vũ	Female	1999-09-09	112 Lê Đại Hành	14	10	Hồ Chí Minh	2023-10-11	NULL	NR0006	ST0006
15	PT00015	040306572454	0993740804	Mỹ Diệu	Trần Thị	Female	1978-07-01	47 Nguyễn Trí Phương	13	Phú Nhuận	Hồ Chí Minh	2023-11-19	NULL	NR0007	ST0007
16	PT00016	012372790727	0993762903	Mỹ Ngọc	Vũ	Female	1999-06-02	45 Trần Quốc Toản	Cầu Kho	1	Hồ Chí Minh	2023-11-01	2023-11-11	NR0007	ST0007
17	PT00017	052246631761	0777692542	Phát Tài	Xuân Sang	Male	2001-09-09	123/112/12 Hồ Thị Kỷ	3	Gò Vấp	Hồ Chí Minh	2023-10-07	2023-10-27	NR0008	ST0008
18	PT00018	025390089332	0332155652	Như Ý	Mùng Xuân	Female	2003-07-04	273 An Dương Vương	5	5	Hồ Chí Minh	2023-10-06	2023-10-16	NR0008	ST0008
19	PT00019	011316829414	0341210778	Tuấn Lộc	Giáng Sinh	Male	1990-12-24	1212 Trần Đình Xu	3	3	Hồ Chí Minh	2023-11-12	2023-11-22	NR0009	ST0009
20	PT00020	006284716118	0865451945	Phương Tuấn	Trịnh Trần	Male	1997-07-07	123 Lãnh Binh Thăng	6	5	Hồ Chí Minh	2023-10-08	NULL	NR0009	ST0009

1.2) Table Comorbidity

Dùng để lưu trữ thông tin bệnh nền.

Đây là đoạn code tạo table Comorbidity trong Microsoft SQL server với khóa chính là id.

```
CREATE TABLE dbo.Comorbidity (
  id INT PRIMARY KEY,
  name NVARCHAR(255) NOT NULL,
);
```

Kiểu dữ liệu của các trường dữ liệu của table Comorbidity:

Data Field Name	**Data Type**
id	INT
name	NVARCHAR(255)

Ví dụ dữ liệu của bảng:

ID	Name
1	Hypertension
2	Diabetes
3	Asthma
4	Heart Disease
5	Cancer

1.3) Table Symptom

Dùng để lưu trữ thông tin triệu chứng.

Đây là đoạn code tạo table Symptom trong Microsoft SQL server với khóa chính là id.

```
CREATE TABLE dbo.Symptom (
  id INT PRIMARY KEY,
  name NVARCHAR(255) NOT NULL,
);
```

Kiểu dữ liệu của các trường dữ liệu của table Symptom :

Data Field Name	Data Type
id	INT
name	NVARCHAR(255)

Ví dụ dữ liệu của bảng:

ID	Name
1	Fever or chills
2	Nausea or vomiting
3	Diarhea
4	Rash
5	Chest pain

1.4) Table PatientComorbidity

Dùng để lưu trữ thông tin bệnh nền của bệnh nhân.

Đây là đoạn code tạo table PatientComorbidity trong Microsoft SQL server với Patient_id và comorbidity vừa là khóa chính và là khóa ngoại.

```
CREATE TABLE dbo.PatientComorbidity (
  patient_id VARCHAR(20) NOT NULL,
  comorbidity_id INT NOT NULL,
  describe NVARCHAR(MAX),
  CONSTRAINT Patient_Comorbidity_PK PRIMARY KEY (patient_id, comorbidity_id),
  FOREIGN KEY (comorbidity_id) REFERENCES Comorbidity(id),
  FOREIGN KEY (patient_id) REFERENCES Patient(unique_number)
);
```

Kiểu dữ liệu của các trường dữ liệu của table PatientComorbidity :

Data Field Name	Data Type
patient_id	VARCHAR(20)
comorbidity_id	INT
describe	NVARCHAR(MAX)

Ví dụ dữ liệu của bảng:

Patient ID	Comorbidity ID	Description
PT00001	1	Patient has hypertension for 5 years
PT00002	2	Patient has diabetes for 10 years
PT00003	3	Patient has asthma since childhood
PT00004	4	Patient has heart disease for 2 years
PT00005	5	Patient has cancer in remission

1.5) Table PatientSymptom

Dùng để lưu trữ triệu chứng của bệnh nhân.

Đây là đoạn code tạo table PatientSymptom trong Microsoft SQL server với là Patient_id và comorbidity vừa là khóa chính và là khóa ngoại.

```
CREATE TABLE dbo.PatientSymptom (
  symptom_id INT NOT NULL,
  patient_id VARCHAR(20) NOT NULL,
  describe NVARCHAR(MAX),
  OnsetDate DATE,
```

```
  TerminationDate DATE,
  CONSTRAINT Patient_Symptom_PK PRIMARY KEY (symptom_id,
patient_id),
  FOREIGN KEY (symptom_id) REFERENCES Symptom(id),
  FOREIGN KEY (patient_id) REFERENCES Patient(unique_number)
);
```

Kiểu dữ liệu của các trường dữ liệu của table PatientSymptom :

Data Field Name	Data Type
symptom_id	INT
patient_id	VARCHAR(20)
describe	NVARCHAR(MAX)
OnsetDate	DATE
TerminationDate	DATE

Ví dụ dữ liệu của bảng:

Symptom ID	Patient ID	Description	Onset Date	Termination Date
1	PT00001	Patient has had a fever for 3 days.	2023-11-14	NULL
2	PT00001	Patient has had a cough for 2 days.	2023-11-16	NULL
3	PT00002	Patient has had shortness of breath for 1 day.	2023-11-17	NULL
4	PT00002	Patient has been feeling fatigued for the past week.	2023-11-10	NULL
5	PT00003	Patient has been experiencing muscle aches for the past 2 days.	2023-11-12	NULL
6	PT00003	Patient has had a headache for the past 3 days.	2023-11-15	NULL

7	PT00004	Patient has recently lost their sense of taste and smell.	2023-11-13	NULL
8	PT00004	Patient has had a sore throat for the past 2 days.	2023-11-15	NULL
9	PT00005	Patient has been experiencing congestion and a runny nose for the past 3 days.	2023-11-11	NULL
10	PT00005	Patient has been nauseated and vomiting for the past day.	2023-11-17	NULL
11	PT00006	Patient has had diarrhea for the past 2 days.	2023-11-16	NULL
12	PT00007	Patient has developed a rash on their chest.	2023-11-14	NULL
13	PT00008	Patient has been experiencing chest pain for the past day.	2023-11-17	NULL
14	PT00009	Patient has been confused and disoriented for the past 2 days.	2023-11-15	NULL
15	PT00010	Patient has bluish lips and face.	2023-11-17	NULL

1.6) Table Building

Dùng để lưu trữ thông tin của tòa nhà.

Đây là đoạn code tạo table Building trong Microsoft SQL server với khóa chính là id và doctor_id là khóa ngoại.

```
CREATE TABLE dbo.Building (
  id INT PRIMARY KEY,
  name NVARCHAR(255) NOT NULL,
  address NVARCHAR(255) NOT NULL
  doctor varchar(20) not null,
 FOREIGN KEY (doctor_id) REFERENCES Doctor(uniquen_number)
);
```

Kiểu dữ liệu của các trường dữ liệu của table Building:

Field Name	Data Type
id	INT
name	NVARCHAR(255)

address	NVARCHAR(255)
Doctor_id	VARCHAR(20)

Ví dụ dữ liệu của bảng:

id	name	Doctor_id	address
1	Main Building	DR0001	123 Main Street, Anytown, USA
2	Outpatient Building	DR0001	456 Elm Street, Anytown, USA
3	Emergency Department Building	DR0001	789 Oak Street, Anytown, USA
4	Laboratory Building	DR0001	123 Maple Street, Anytown, USA
5	Pharmacy Building	DR0001	456 Birch Street, Anytown, USA
6	Administration Building	DR0001	789 Pine Street, Anytown, USA
7	Research Building	DR0001	123 Willow Street, Anytown, USA
8	Library Building	DR0001	456 Cherry Street, Anytown, USA
9	Dining Hall Building	DR0001	789 Walnut Street, Anytown, USA
10	Parking Garage Building	DR0001	123 Cedar Street, Anytown, USA

1.7) Table Floor

Dùng để lưu trữ thông tin của tầng.

Đây là đoạn code tạo table Floor trong Microsoft SQL server với khóa chính là id và building_id là khóa ngoại.

```
CREATE TABLE dbo.Floor (
  id INT PRIMARY KEY,
  building_id INT NOT NULL,
  name NVARCHAR(255) NOT NULL,
  FOREIGN KEY (building_id) REFERENCES Building(id)
);
```

Kiểu dữ liệu của các trường dữ liệu của table Floor:

Field Name	**Data Type**
id	INT
building_id	INT
name	NVARCHAR(255)

Ví dụ kiểu dữ liệu trong bảng:

id	**building_id**	**name**
1	1	First Floor
2	1	Second Floor
3	1	Third Floor
4	1	Fourth Floor
5	1	Fifth Floor
6	1	Sixth Floor

7	2	First Floor
8	2	Second Floor
9	2	Third Floor
10	2	Fourth Floor
11	2	Fifth Floor
12	2	Sixth Floor
13	3	First Floor
14	3	Second Floor
15	3	Third Floor
16	3	Fourth Floor
17	3	Fifth Floor
18	3	Sixth Floor

1.8) Table Room

Dùng để lưu trữ thông tin của phòng.
Đây là đoạn code tạo table Floor trong Microsoft SQL server với khóa chính là id, building_id và floor_id là khóa ngoại.

```
CREATE TABLE dbo.Room (
  id INT PRIMARY KEY,
  building_id INT NOT NULL,
  floor_id INT NOT NULL,
  type NVARCHAR(50) NOT NULL,
  name NVARCHAR(255) NOT NULL,
  FOREIGN KEY (building_id) REFERENCES Building(id),
```

```
  FOREIGN KEY (floor_id) REFERENCES Floor(id)
);
```

Kiểu dữ liệu của các trường dữ liệu của table Room:

Field Name	Data Type
id	INT
building_id	INT
floor_id	INT
type	INT
name	NVARCHAR(255)

Ví dụ dữ liệu của bảng:

id	building_id	floor_id	type	name
1	1	1	1	Room 101
2	1	1	2	Room 102
3	1	1	3	Room 103
4	1	1	4	Room 104
5	1	1	5	Room 105
6	1	2	6	Room 201
7	1	2	7	Room 202
8	1	2	8	Room 203
9	1	2	9	Room 204

10	1	2	10	Room 205
11	1	3	11	Room 301
12	1	3	12	Room 302
13	1	3	13	Room 303
14	1	3	14	Room 304
15	1	3	15	Room 305
16	1	4	16	Room 401
17	1	4	17	Room 402
18	1	4	18	Room 403
19	1	4	19	Room 404
20	1	4	20	Room 405

1.9) Table Staff

Dùng để lưu trữ thông tin của nhân viên.
Đây là đoạn code tạo table Staff trong Microsoft SQL server với khóa chính là unique_number, id và phone là khóa dự tuyển.

```
CREATE TABLE dbo.Staff (
  unique_number VARCHAR(20) NOT NULL PRIMARY KEY,
  id VARCHAR(20) NOT NULL UNIQUE,
  phone VARCHAR(20) NOT NULL UNIQUE,
  fname NVARCHAR(255) NOT NULL,
  lname NVARCHAR(255) NOT NULL,
  gender NVARCHAR(10) NOT NULL CHECK (gender IN ('Male', 'Female')),
  birthday DATE NOT NULL,
```

```
);
```

Kiểu dữ liệu của các trường dữ liệu của table Staff:

Field Name	Data Type
unique_number	VARCHAR(20)
id	VARCHAR(20)
phone	VARCHAR(20)
fname	NVARCHAR(255)
lname	NVARCHAR(255)
gender	NVARCHAR(10)
birthday	DATE

Ví dụ dữ liệu của bảng:

unique_number	id	phone	fname	lname	gender	birthday
ST0000	123456789012	912345678	Staff1	Johnson	Male	1970-01-01
ST0001	134567890123	987654321	Staff2	Anderson	Female	1975-05-12
ST0002	145678901234	976543210	Staff3	Thompson	Male	1980-03-27
ST0003	156789012345	965432109	Staff4	Williams	Female	1985-07-04
ST0004	167890123456	954321098	Staff5	Jones	Male	1990-11-01
ST0005	178901234567	943210987	Staff6	Baker	Female	1995-02-15
ST0006	189012345678	932109876	Staff7	Scott	Male	2000-06-29
ST0007	190123456789	921098765	Staff8	Harris	Female	2005-10-13
ST0008	201234567890	910987654	Staff9	Evans	Male	2010-01-27

ST0009	212345678901	909876543	Staff10	Nelson	Female	2015-05-04

1.10) Table PatientHistoryLocation

Dùng để lưu trữ thông tin lịch sử của bệnh nhân đã ở phòng nào.
Đây là đoạn code tạo table PatientHistoryLocation trong Microsoft SQL server với khóa chính là Patient_id, room_id, và staff_id.

```
CREATE TABLE dbo.PatientHistoryLocation (
  patient_id VARCHAR(20) NOT NULL,
  room_id INT NOT NULL,
  Date_admitted DATE NOT NULL,
  Date_discharge DATE,
  CONSTRAINT Patient_history_location_PK PRIMARY KEY
(patient_id, room_id),
  FOREIGN KEY (patient_id) REFERENCES Patient(unique_number),
  FOREIGN KEY (room_id) REFERENCES Room(id)
  );
```

Kiểu dữ liệu của các trường dữ liệu của table PatientHistoryLocation:

Field Name	Data Type
patient_id	VARCHAR(20)
room_id	INT
Date_admitted	DATE
Date_discharge	DATE

Ví dụ về dữ liệu của bảng:

patient_id	room_id	Date_admitted	Date_discharge
PT00001	1	2023-10-04	2023-10-06
PT00002	2	2023-10-05	2023-10-07
PT00003	3	2023-10-06	2023-10-08
PT00004	4	2023-10-07	2023-10-09
PT00005	5	2023-10-08	2023-10-10
PT00006	6	2023-10-09	NULL
PT00007	7	2023-10-10	NULL
PT00008	8	2023-10-11	NULL
PT00009	9	2023-10-12	NULL
PT00010	10	2023-10-13	NULL

1.11) Table Doctor

Dùng để lưu trữ thông tin của bác sĩ.

Đây là đoạn code tạo table Doctor trong Microsoft SQL server với khóa chính là unique_number, id và phone là khóa dự tuyển.

```
CREATE TABLE dbo.Doctor (
    unique_number VARCHAR(20) NOT NULL PRIMARY KEY,
    id VARCHAR(20) NOT NULL UNIQUE,
    phone VARCHAR(20) NOT NULL UNIQUE,
    fname NVARCHAR(255) NOT NULL,
    lname NVARCHAR(255) NOT NULL,
    gender NVARCHAR(10) NOT NULL CHECK (gender IN ('Male', 'Female')),
    birthday DATE NOT NULL,
```

```
);
```

Kiểu dữ liệu của các trường dữ liệu của table Doctor:

Field Name	Data Type
unique_number	VARCHAR(20)
id	VARCHAR(20)
phone	VARCHAR(20)
fname	NVARCHAR(255)
lname	NVARCHAR(255)
gender	NVARCHAR(10)
birthday	DATE

Ví dụ dữ liệu của bảng:

unique_number	id	phone	fname	lname	gender	birthday
DR0000	418666928076	9795944763	Doctor1	Smith	Female	1973-5-4
DR0001	215226732633	9528609314	Doctor2	Smith	Male	1973-8-2
DR0002	858206134365	1812272154	Doctor3	Smith	Female	1981-11-18
DR0003	844547193332	5640463116	Doctor4	Smith	Female	1989-9-17
DR0004	481320012697	9645070582	Doctor5	Smith	Male	1985-9-3
DR0005	429818678202	6475269022	Doctor6	Smith	Female	1958-1-13
DR0006	612715996147	1801511093	Doctor7	Smith	Female	1979-4-15

DR0007	729577825701	3789665590	Doctor8	Smith	Female	1970-12-20
DR0008	676290615083	5019232200	Doctor9	Smith	Male	1978-7-21
DR0009	239912164270	7447071031	Doctor10	Smith	Male	2000-10-6

1.12) Table Nurse

Dùng để lưu trữ thông tin của y tá.
Đây là đoạn code tạo table Nurse trong Microsoft SQL server với khóa chính là unique_number, id và phone là khóa dự tuyển.

```
CREATE TABLE dbo.Nurse(
    unique_number VARCHAR(20) NOT NULL PRIMARY KEY,
    id VARCHAR(20) NOT NULL UNIQUE,
    phone VARCHAR(20) NOT NULL UNIQUE,
    fname NVARCHAR(255) NOT NULL,
    lname NVARCHAR(255) NOT NULL,
    gender NVARCHAR(10) NOT NULL CHECK (gender IN ('Male',
'Female')),
    birthday DATE NOT NULL,


);
```

Kiểu dữ liệu của table Nurse:

Field Name	Data Type
unique_number	VARCHAR(20)
id	VARCHAR(20)
phone	VARCHAR(20)

fname	NVARCHAR(255)
lname	NVARCHAR(255)
gender	NVARCHAR(10)
birthday	DATE

Ví dụ dữ liệu của bảng:

unique_number	id	phone	fname	lname	gender	birthday
NR0000	347534328975	9795944763	Nurse1	Jones	Female	1973-5-4
NR0001	235426732633	9528609314	Nurse2	Brown	Male	1973-8-2
NR0002	858206134365	1812272154	Nurse3	Williams	Female	1981-11-18
NR0003	844547193332	5640463116	Nurse4	Miller	Female	1989-9-17
NR0004	481320012697	9645070582	Nurse5	Davis	Male	1985-9-3
NR0005	429818678202	6475269022	Nurse6	Rodriguez	Female	1958-1-13
NR0006	612715996147	1801511093	Nurse7	Martinez	Female	1979-4-15
NR0007	729577825701	3789665590	Nurse8	Lopez	Female	1970-12-20
NR0008	676290615083	5019232200	Nurse9	Wilson	Male	1978-7-21
NR0009	239912164270	7447071031	Nurse10	Taylor	Female	2000-10-6

1.13) Table Medication

Dùng để lưu trữ thông tin thuốc.
Đây là đoạn code tạo table Medication trong Microsoft SQL server với khóa chính là id.

```
CREATE TABLE dbo.Medication (
  id VARCHAR(20) PRIMARY KEY,
```

```
  name NVARCHAR(255) NOT NULL,
  price DECIMAL(10,2) NOT NULL,
  effect NVARCHAR(MAX),
  exp_date DATE NOT NULL
);
```

Kiểu dữ liệu của table Medication:

Field Name	**Data Type**
id	VARCHAR(20)
name	NVARCHAR(255)
price	DECIMAL(10,2)
effect	NVARCHAR(MAX)
exp_date	DATE

Ví dụ dữ liệu của bảng:

id	**name**	**price**	**effect**	**exp_date**
CVD001	Paxlovid	500	Treats mild-to-moderate COVID-19	2024-06-30
CVD002	Molnupiravir	300	Treats mild-to-moderate COVID-19	2024-09-15
CVD003	Remdesivir	1000	Treats severe COVID-19	2024-12-25
CVD004	Dexamethasone	200	Reduces inflammation in severe COVID-19	2024-08-01
CVD005	Tocilizumab	800	Reduces inflammation in severe COVID-19	2024-11-30

1.14) Table Treatment

Dùng để lưu trữ thông tin trị liệu của bệnh nhân.
Đây là đoạn code tạo table Treatment trong Microsoft SQL server với patient_id, doctor_id, nurse_id, medication_id vừa là khóa chính và là khóa ngoại.

```
CREATE TABLE dbo.Treatment (
  patient_id VARCHAR(20) NOT NULL,
  doctor_id VARCHAR(20) NOT NULL,
  medication_id VARCHAR(20) NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE,
  result NVARCHAR(255),
  CONSTRAINT Treatment_UK UNIQUE (patient_id, doctor_id, nurse_id, medication_id),
  FOREIGN KEY (patient_id) REFERENCES Patient(unique_number),
  FOREIGN KEY (doctor_id) REFERENCES Doctor(unique_number),
  FOREIGN KEY (medication_id) REFERENCES Medication(id)
);
```

Kiểu dữ liệu của table Treatment:

Field Name	Data Type
patient_id	VARCHAR(20)
doctor_id	VARCHAR(20)

medication_id	VARCHAR(20)
start_date	DATE
end_date	DATE
result	NVARCHAR(255)

Ví dụ dữ liệu của bảng:

patient_id	**doctor_id**	**medication_id**	**start_date**	**end_date**	**result**
PT00001	DR0001	CVD001	2023-10-04	2023-10-06	Recovered
PT00002	DR0002	CVD002	2023-10-05	2023-10-07	Recovered
PT00003	DR0003	CVD003	2023-10-06	2023-10-08	Recovered
PT00004	DR0004	CVD004	2023-10-07	2023-10-09	Recovered
PT00005	DR0005	CVD005	2023-10-08	2023-10-10	Recovered
PT00006	DR0006	CVD001	2023-10-09	NULL	Improving
PT00007	DR0007	CVD002	2023-10-10	NULL	Improving
PT00008	DR0008	CVD003	2023-10-11	NULL	Stable
PT00009	DR0009	CVD004	2023-10-12	NULL	Stable
PT00010	DR0010	CVD005	2023-10-13	NULL	Stable

1.15) Table PCRTest

Dùng để lưu trữ dữ liệu xét nghiệm của bệnh nhân theo kiểu PCR.
Đây là đoạn code tạo table PCRTest trong Microsoft SQL server với khóa chính là id và patient_id là khóa ngoại.

```
CREATE TABLE dbo.PCRTest (
```

```
  id INT PRIMARY KEY,
  patient_id VARCHAR(20) NOT NULL,
  date DATE NOT NULL,
  result NVARCHAR(20) NOT NULL,
  ct_value INT,
  FOREIGN KEY (patient_id) REFERENCES Patient(unique_number)
);
```

Kiểu dữ liệu của table PCRTest:

Field Name	**Data Type**
id	INT
patient_id	VARCHAR(20)
date	DATE
result	NVARCHAR(20)
ct_value	INT

Ví dụ dữ liệu của bảng:

patient_id	**date**	**result**	**ct_value**
PT00001	2023-11-22	Positive	10
PT00001	2023-11-23	Negative	0
PT00001	2023-11-24	Positive	15
PT00002	2023-11-21	Negative	0
PT00002	2023-11-22	Positive	20
PT00002	2023-11-23	Negative	0

PT00003	2023-11-20	Positive	15
PT00003	2023-11-21	Positive	20
PT00003	2023-11-22	Negative	0
PT00004	2023-11-19	Negative	0
PT00004	2023-11-20	Positive	10
PT00004	2023-11-21	Negative	0
PT00005	2023-11-18	Positive	20
PT00005	2023-11-19	Negative	0
PT00005	2023-11-20	Positive	15

1.16) Table QuickTest

Dùng để lưu trữ dữ liệu xét nghiệm của bệnh nhân theo kiểu Quick.
Đây là đoạn code tạo table QuickTest trong Microsoft SQL server với khóa chính là id và patient_id là khóa ngoại.

```
CREATE TABLE dbo.QuickTest(
  id INT PRIMARY KEY,
  patient_id VARCHAR(20) NOT NULL,
  date DATE NOT NULL,
  result NVARCHAR(20) NOT NULL,
  ct_value INT,
  FOREIGN KEY (patient_id) REFERENCES Patient(unique_number)
);
```

Kiểu dữ liệu của table QuickTest:

Field Name	**Data Type**

id	INT
patient_id	VARCHAR(20)
date	DATE
result	NVARCHAR(20)
ct_value	INT

Ví dụ dữ liệu của bảng:

patient_id	date	result	ct_value
PT00001	2023-11-22	Positive	10
PT00001	2023-11-23	Negative	0
PT00001	2023-11-24	Positive	15
PT00002	2023-11-21	Negative	0
PT00002	2023-11-22	Positive	20
PT00002	2023-11-23	Negative	0
PT00003	2023-11-20	Positive	15
PT00003	2023-11-21	Positive	20
PT00003	2023-11-22	Negative	0
PT00004	2023-11-19	Negative	0
PT00004	2023-11-20	Positive	10
PT00004	2023-11-21	Negative	0
PT00005	2023-11-18	Positive	20
PT00005	2023-11-19	Negative	0
PT00005	2023-11-20	Positive	15

1.17) Table SPO2Test

Dùng để lưu trữ dữ liệu xét nghiệm của bệnh nhân theo kiểu SPO2.
Đây là đoạn code tạo table QuickTest trong Microsoft SQL server với khóa chính là id và patient_id là khóa ngoại.

```
CREATE TABLE dbo.SPO2Test (
  id INT PRIMARY KEY,
  patient_id VARCHAR(20) NOT NULL,
  date DATE NOT NULL,
  spo2_level INT NOT NULL,
  FOREIGN KEY (patient_id) REFERENCES Patient(unique_number)
);
```

Kiểu dữ liệu của table SPO2Test:

Field	**Data Type**
id	INT
patient_id	VARCHAR(20)
date	DATE
spo2_level	INT

Ví dụ dữ liểu của bảng:

ID	**Patient ID**	**Date**	**SPO2 Level**
1	PT00001	2023-11-22	95
2	PT00002	2023-11-21	98
3	PT00003	2023-11-20	97
4	PT00004	2023-11-19	96

5	PT00005	2023-11-18	99
6	PT00006	2023-11-17	98
7	PT00007	2023-11-16	97
8	PT00008	2023-11-15	96
9	PT00009	2023-11-14	99
10	PT00010	2023-11-13	98
11	PT00001	2023-11-23	96
12	PT00002	2023-11-22	99
13	PT00003	2023-11-21	98
14	PT00004	2023-11-20	97
15	PT00005	2023-11-19	96
16	PT00006	2023-11-18	99
17	PT00007	2023-11-17	98
18	PT00008	2023-11-16	97
19	PT00009	2023-11-15	96
20	PT00010	2023-11-14	99

1.18) Table RespiratoryRateTest

Dùng để lưu trữ dữ liệu xét nghiệm của bệnh nhân theo kiểu Respiratory Rate

Đây là đoạn code tạo table RespiratoryRateTest trong Microsoft SQL server với khóa chính là id và patient_id là khóa ngoại.

```
CREATE TABLE dbo.RespiratoryRateTest (
```

```
  id INT PRIMARY KEY,
  patient_id VARCHAR(20) NOT NULL,
  date DATE NOT NULL,
  respiratory_rate INT,
  FOREIGN KEY (patient_id) REFERENCES Patient(unique_number)
);
```

Kiểu dữ liệu của table RespiratoryRateTest:

Field	**Data Type**
id	INT
patient_id	VARCHAR(20)
date	DATE
respiratory_rate	INT

Ví dụ dữ liệu của bảng:

ID	**Patient ID**	**Date**	**Respiratory Rate**
1	PT00001	2023-11-22	16
2	PT00002	2023-11-21	18
3	PT00003	2023-11-20	17
4	PT00004	2023-11-19	16
5	PT00005	2023-11-18	19
6	PT00006	2023-11-17	18
7	PT00007	2023-11-16	17
8	PT00008	2023-11-15	16

9	PT00009	2023-11-14	19
10	PT00010	2023-11-13	18
11	PT00001	2023-11-23	17
12	PT00002	2023-11-22	19
13	PT00003	2023-11-21	18
14	PT00004	2023-11-20	17
15	PT00005	2023-11-19	16
16	PT00006	2023-11-18	19
17	PT00007	2023-11-17	18
18	PT00008	2023-11-16	17
19	PT00009	2023-11-15	16
20	PT00010	2023-11-14	19

1.19) Table Warning

Dùng để lưu trữ thông tin bệnh đang trong tình trạn cảnh báo và cần được bác sĩ thăm khám.

Đây là đoạn code tạo table Warning trong Microsoft SQL server với khóa chính là id và patient_id là khóa ngoại.

```
CREATE TABLE dbo.Warning (
  id INT PRIMARY KEY,
  patient_id VARCHAR(20) NOT NULL,
  date DATE NOT NULL,
  reason NVARCHAR(255),
```

```
  CONSTRAINT Warning_FK1 FOREIGN KEY (patient_id) REFERENCES
Patient(unique_number)
);
```

Kiểu dữ liệu của table Warning:

Field	**Data Type**
id	INT
patient_id	VARCHAR(20)
date	DATE
reason	NVARCHAR(255)

Ví dụ dữ liệu của bảng:

ID	**Patient ID**	**Date**	**Reason**
1	PT00001	2023-11-22	Low SpO2 level (90%)
2	PT00001	2023-11-23	Low SpO2 level (93%)
3	PT00001	2023-11-24	Low SpO2 level (95%)
4	PT00002	2023-11-21	High respiratory rate (22 breaths/minute)
5	PT00002	2023-11-22	High respiratory rate (20 breaths/minute)
6	PT00002	2023-11-23	High respiratory rate (18 breaths/minute)
7	PT00003	2023-11-20	Low SpO2 level (92%)
8	PT00003	2023-11-21	Low SpO2 level (94%)
9	PT00003	2023-11-22	Low SpO2 level (96%)

10	PT00004	2023-11-19	High respiratory rate (24 breaths/minute)
11	PT00004	2023-11-20	High respiratory rate (21 breaths/minute)
12	PT00004	2023-11-21	High respiratory rate (19 breaths/minute)
13	PT00005	2023-11-18	Low SpO2 level (88%)
14	PT00005	2023-11-19	Low SpO2 level (87%)
15	PT00005	2023-11-20	Low SpO2 level (86%)
16	PT00006	2023-11-17	High respiratory rate (26 breaths/minute)
17	PT00006	2023-11-18	High respiratory rate (27 breaths/minute)
18	PT00006	2023-11-19	High respiratory rate (28 breaths/minute)
19	PT00007	2023-11-16	Low SpO2 level (91%)
20	PT00007	2023-11-17	Low SpO2 level (90%)
21	PT00007	2023-11-18	Low SpO2 level (89%)

1.20) Table Account

Dùng để lưu trữ thông tin tài khoản của nhân viên.
Đây là đoạn code tạo table Account trong Microsoft SQL server với khóa chính là phone.

```
CREATE TABLE dbo.Account (
  phone VARCHAR(20) NOT NULL PRIMARY KEY,
  password VARCHAR(255) NOT NULL,
```

```
  role VARCHAR(10) NOT NULL CHECK (role IN ('patient',
'admin'))
);
```

Kiểu dữ liệu của bảng Account:

Field	**Data Type**
phone	VARCHAR(10)
password	NVARCHAR(255)
role	VARCHAR(10)

Ví dụ dữ liệu của bảng:

Phone	**password**	**role**
9795944763	admin###	admin

1.21) Table TypeRoom

Dùng để lưu trữ thông tin kiểu phòng của phòng.
Đây là đoạn code tạo table Account trong Microsoft SQL server với khóa chính là phone.

```
CREATE TABLE dbo.TypeRoom(
  id INT NOT NULL PRIMARY KEY,
  room_id INT NOT NULL,
  name VARCHAR(255) NOT NULL,
 );
```

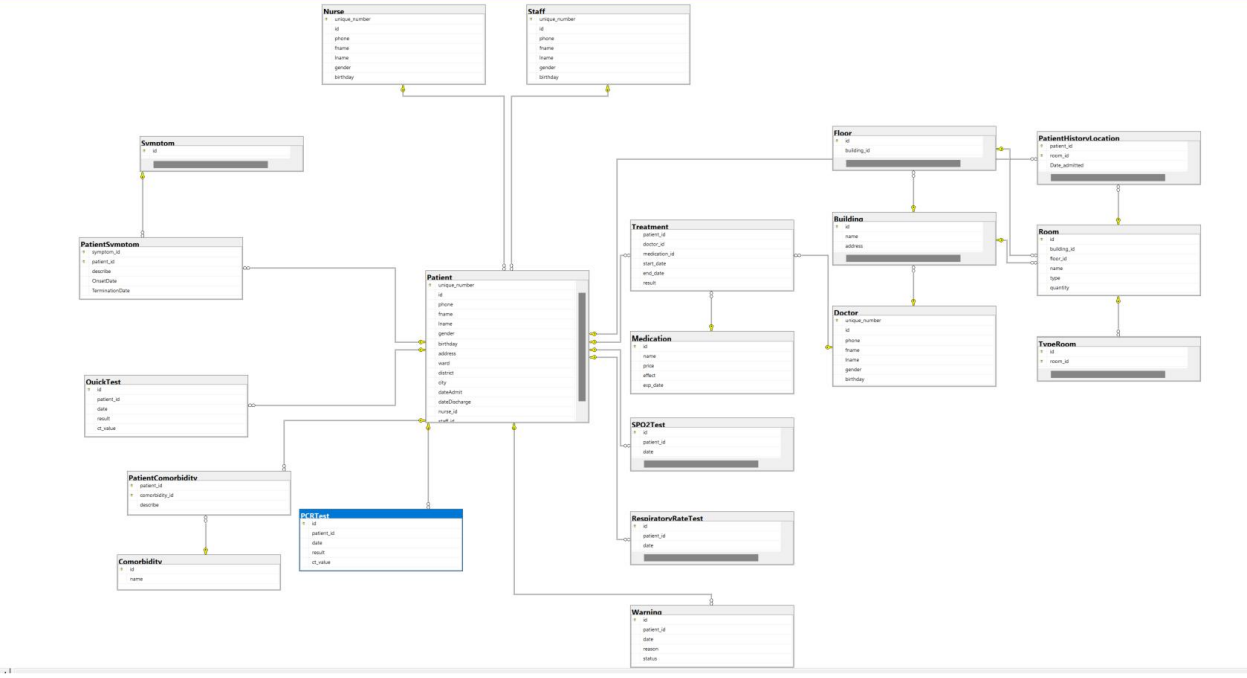
Kiểu dữ liệu của bảng Account:

Field	Data Type
id	INT
room_id	INT
name	VARCHAR(255)

Ví dụ dữ liệu của bảng:

id	room_id	name
1	1	Phòng cấp cứu

2) Quan hệ giữa các bảng



Quan hệ giữa các bảng

3) Index

3.1) Khái niệm

[3]Index là một cấu trúc dữ liệu được dùng để định vị và truy cập nhanh nhất vào dữ liệu trong các bảng database.

Index là một cách tối ưu hiệu suất truy vấn database bằng việc giảm lượng truy cập vào bộ nhớ khi thực hiện truy vấn.

Nói đơn giản, index trỏ tới địa chỉ dữ liệu trong bảng, giống như mục lục của một cuốn sách(gồm tên đề mục và số trang), nó giúp truy vấn trở nên nhanh chóng như việc bạn xem mục lục và tìm đúng trang cần đọc.

3.2) Đánh giá hiệu quả của indexing

Vấn đề:

Trong hệ thống quản lý bệnh viện, việc tìm kiếm thông tin bệnh nhân là một yêu cầu quan trọng. Tuy nhiên, nếu hệ thống có nhiều bệnh nhân, việc tìm kiếm theo câu truy vấn sẽ tốn rất nhiều thời gian. Nguyên nhân là do hệ thống phải duyệt qua toàn bộ bảng dữ liệu để tìm kết quả phù hợp.

Giải pháp:

Để giải quyết vấn đề này, có thể sử dụng kỹ thuật indexing (chỉ mục). Indexing là một cấu trúc dữ liệu giúp hệ thống tìm kiếm thông tin nhanh hơn bằng cách chỉ định vị trí của các bản ghi dữ liệu trong bảng.

Thực nghiệm:

Để chứng minh hiệu quả của indexing, chúng ta sẽ thực hiện một thí nghiệm trên bảng "Patient" với 113988 records với dữ liệu ngẫu nhiên. Thí nghiệm sẽ so sánh thời gian tìm kiếm thông tin bệnh nhân với và không sử dụng indexing.

Thực hiện trên SSMS

Thực hiện đoạn code sau:

```
SET STATISTICS IO ON;

select * from dbo.Patient where fname = 'FirstName9989';

SET STATISTICS IO OFF;
```

khi không sử dụng indexing cho colum fname

Thời gian truy vấn một bệnh nhân có fname là ‘FirstName9989’ là 55 giây, với 1238 logical reads.

khi sử dụng indexing cho colum fname

Thời gian truy vấn một bệnh nhân có fname là ‘FirstName9989’ xấp xỉ 0 giây, với 3 logical reads.

Lý thuyết

Tệp dữ liệu được sắp xếp theo thứ tự với r = 113988 bản ghi được lưu trên ổ đĩa với block size B = 4096 bytes

Data Field Name	**Data Type**
unique_number	VARCHAR(20)
nurse_id	VARCHAR(20)
staff_id	VARCHAR(20)
id	VARCHAR(20)
phone	VARCHAR(20)
fname	NVARCHAR(255)
lname	NVARCHAR(255)
gender	NVARCHAR(10)
birthday	DATE
address	NVARCHAR(255)
ward	NVARCHAR(255)
district	NVARCHAR(255)
city	NVARCHAR(255)
dateAdmit	DATE

dateDischarge	DATE

Bảng kiểu dữ liệu của bệnh nhân (dbo.Patient)

Từ kiểu table trên ta biết được 1 dòng có R= 3204 bytes.

Chi phí tìm kiếm một bản ghi sử dụng binary search

Số lượng record trên một block là

bfr = ⌊ (B/R)⌋ = ⌊ (4096/3204)⌋ = 1 record 1 block

Tổng số block

b = ⌈ (r/bfr)⌉ = ⌈ (113988/1)⌉ = 113988 blocks

Việc tìm kiếm binary search này sẽ cần truy cập xấp xỉ $\log_2 b = \log_2 113988 = 16$ blocks

Chi phí tìm kiếm bản ghi sử dụng indexing

Ta có độ dài khóa V là 510 bytes với fname có kiểu dữ liệu là nvarchar(255), độ dài mỗi index Ri là V + P = 510 + 6 = 516 bytes. Do đó, mỗi block có thể chứa bfri = B/Ri = 4096/516 = 7 records.

Tổng số index entries ri bằng với số lượng blocks trong tệp dữ liệu, là 113988. Do đó, số lượng index blocks là bi = ⌈ (ri / bfri)⌉ = ⌈ (113988/7)⌉ = 16 blocks.

Tìm kiếm nhị phân trên tệp chỉ mục sẽ cần ⌈ (log2 bi)⌉ = ⌈ (log2 16)⌉ = 4 block accesses.

Tổng số block accesses cần thiết để tìm kiếm một bản ghi trong tệp dữ liệu là

4 + 1 = 5 block accesses.

Kết luận

Từ kết quả thực nghiệm và phân tích trên, có thể thấy rằng việc sử dụng indexing có thể cải thiện đáng kể hiệu suất của các câu truy vấn tìm kiếm trong tệp dữ liệu lớn.

Trong trường hợp không sử dụng indexing, câu truy vấn tìm kiếm một bệnh nhân có fname là 'FirstName9989' sẽ cần truy cập 1238 logical reads và mất 55 giây. Điều này là do cơ sở dữ liệu phải đọc toàn bộ tệp dữ liệu để tìm bản ghi mong muốn.

Trong trường hợp sử dụng indexing, câu truy vấn chỉ cần truy cập 3 logical reads và mất xấp xỉ 0 giây. Điều này là do cơ sở dữ liệu có thể sử dụng tệp chỉ mục để tìm kiếm bản ghi mong muốn một cách nhanh chóng.

Vậy, việc sử dụng indexing có thể mang lại những lợi ích sau:

Tăng tốc độ truy vấn: Tìm kiếm bản ghi trong tệp dữ liệu lớn sẽ nhanh hơn nhiều so với không sử dụng indexing.

Giảm tải cho hệ thống: Cơ sở dữ liệu không phải đọc toàn bộ tệp dữ liệu, do đó giảm tải cho hệ thống.

Tăng tính ổn định của hệ thống: Hệ thống sẽ ổn định hơn vì không phải đọc toàn bộ tệp dữ liệu mỗi khi có truy vấn tìm kiếm.

Tuy nhiên, việc sử dụng indexing cũng có một số nhược điểm sau:

Tăng kích thước cơ sở dữ liệu: Tệp chỉ mục sẽ làm tăng kích thước cơ sở dữ liệu.

Tăng thời gian cập nhật dữ liệu: Cập nhật dữ liệu trong tệp chỉ mục sẽ mất nhiều thời gian hơn so với không sử dụng indexing.

Tóm lại, việc sử dụng indexing là một giải pháp hiệu quả để cải thiện hiệu suất của các câu truy vấn tìm kiếm trong tệp dữ liệu lớn. Tuy nhiên, cần cân nhắc kỹ lưỡng các ưu nhược điểm của indexing trước khi đưa ra quyết định sử dụng.

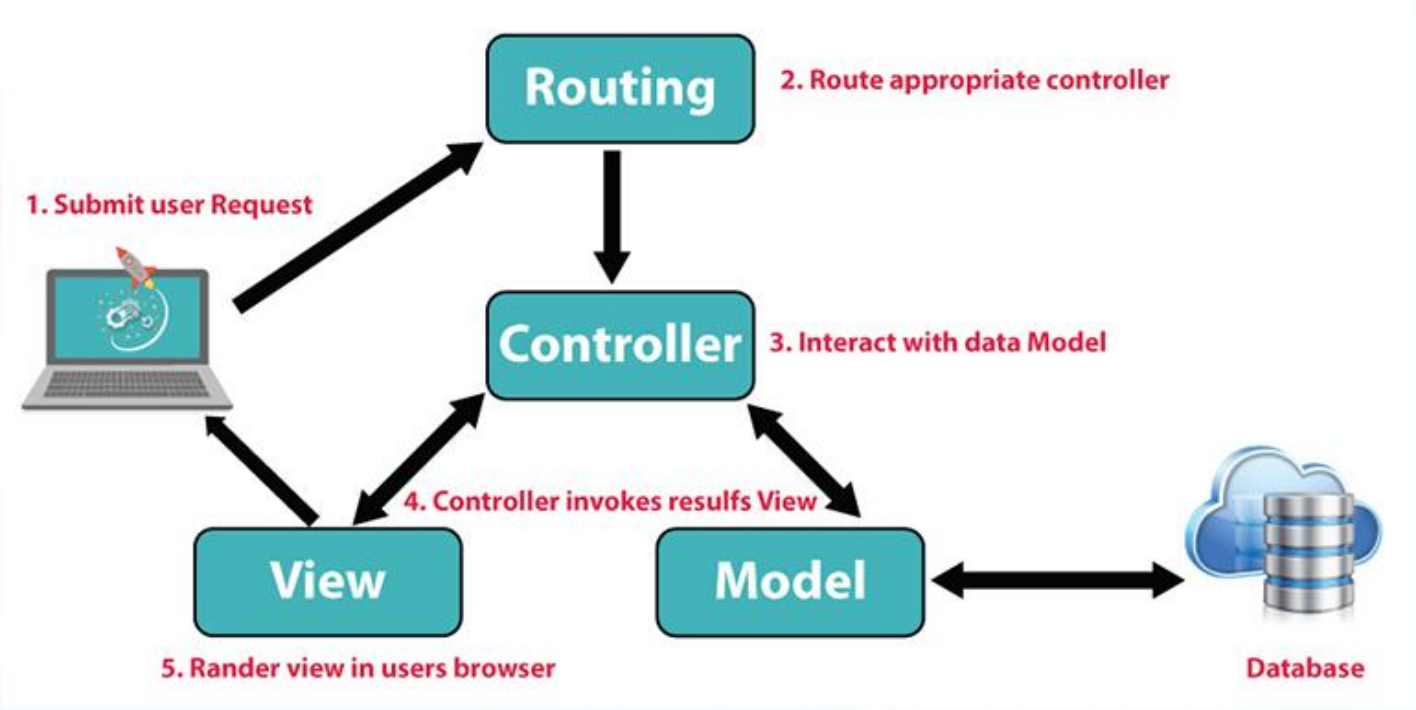
V/XÂY DỰNG WEBSITE QUARANTINE CAMP

1) Kiến thức cần có.

1.1) Mô hình MVC (Model-View-Controller)

1.1.1) Khái niệm

[4]Mô hình MVC là một mẫu kiến trúc phân tách một ứng dụng thành ba phần logic chính Model, View, Controller. Mỗi thành phần kiến trúc được xây dựng để xử lý kía cạnh phát triển cụ thể của một ứng dụng. MVC tách lớp logic nghiệp vụ và lớp hiển thị ra riêng biệt.



Mô hình MVC

1.1.2) Kiến trúc MVC

Mô hình MVC bao gồm:

Model:

+Có nhiệm vụ thao tác với Database.

+Nó chứa tất cả các hàm, các phương thức truy vấn trực tiếp với dữ liệu.

+Controller sẽ thông qua các hàm, phương thức đó để lấy dữ liệu rồi gửi qua View.

Controller:

+Là thành phần trung gian giữa Model và View.

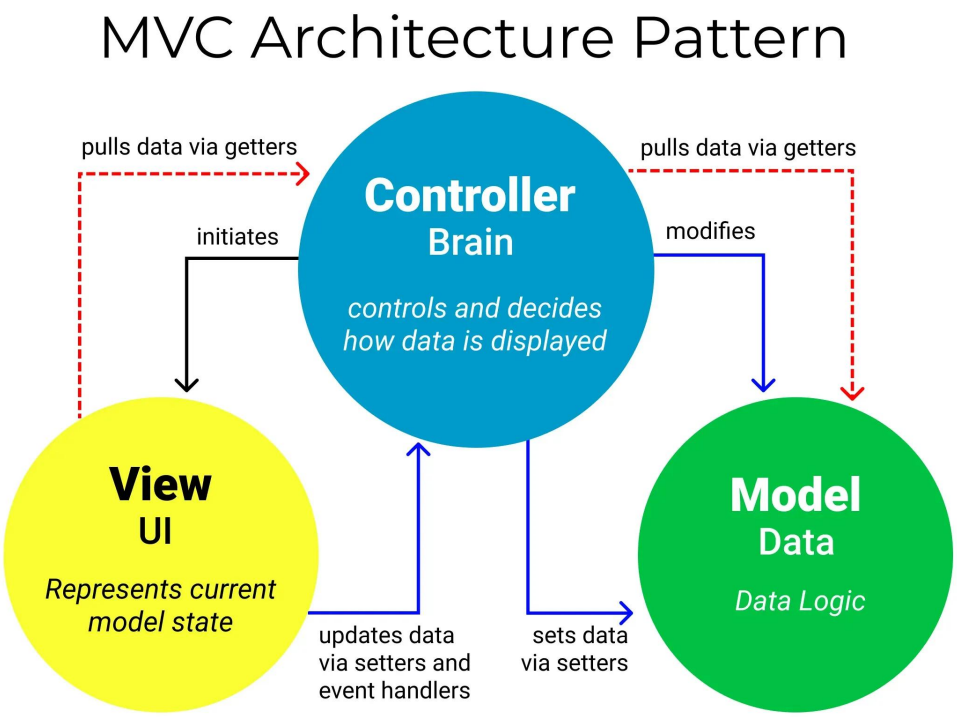
+Đảm nhận vai trò tiếp nhận yêu cầu từ người dùng, thông qua Model để lấy dữ liệu sau đó thông qua View để hiển thị cho người dùng.

View:

+Là giao diện người dùng (User Interface).

+Chứa các thành phần tương tác với người dùng như menu, button, image, text,...

+Nơi nhận dữ liệu từ Controller và hiển thị.



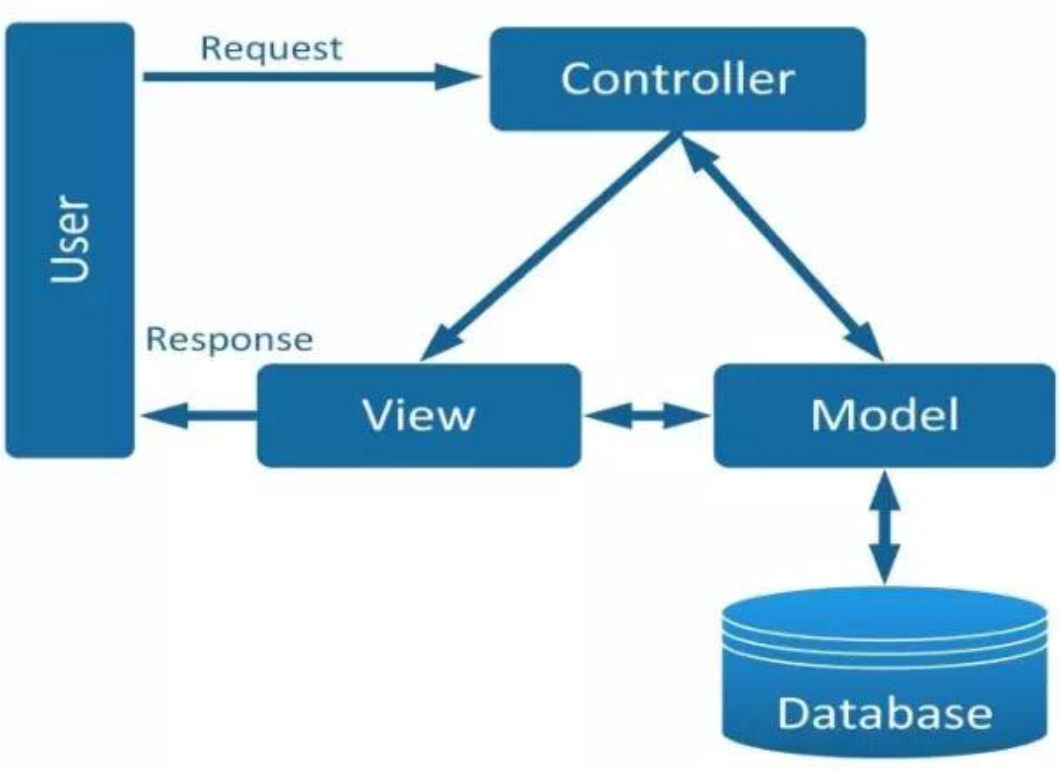
Kiến trúc mô hình mvc

1.1.3) Sự tương tác giữa các thành phần

Controller tương tác với qua lại với View.

Controller tương tác qua lại với Model.

Model và View không có sự tương tác với nhau trực tiếp mà nó tương tác với nhau thông qua Controller.



Sự tương tác của các thành phần trong mô hình mvc

1.2) PHP

1.2.1) Khái niệm

[5]Ngôn ngữ PHP là từ viết tắt của Personal Home Page nay đã chuyển thành Hypertext Preprocessor. Thuật ngữ này là một dạng mã lệnh hoặc một chuỗi ngôn ngữ kịch bản được dùng để phát triển các ứng dụng web chạy trên máy chủ. Ki các lập trình viên PHP viết chương trình, chuỗi lệnh sẽ được xử lý trên server sau đó sinh ra mã HTML trên client. Dựa vào đó, các ứng dụng trên website sẽ hoạt động một cách dễ dàng.
Ngôn ngữ PHP thường được dùng trong việc xây dựng và phát triển website bởi nó có thể kết nối dễ dàng với các website khác có sử dụng HTML. PHP cũng là ngôn ngữ lập trình có mã nguồn mở, tương thích với nhiều nền tảng khác nhau như MacOS, Linux, Windows,… PHP được nhiều người dùng đánh giá là dễ đọc nên đa số các lập trình viên sẽ lựa chọn học PHP trước khi bắt đầu vào nghề.

1.2.2) Ứng dụng của ngôn ngữ lập trình

Ngôn ngữ lập trình PHP thường tập trung vào việc thiết lập chương trình cho máy chủ, tạo các cơ sở dữ liệu, xây dựng nội dung website, nhận dữ liệu cookie. Chưa hết, bạn còn có thể thực hiện được nhiều thao tác, công năng khác khi sử dụng ngôn ngữ này.

Một số ứng dụng phổ biến của PHP trong ngành IT:

- *Thiết lập chương trình cho hệ thống máy chủ:* Đây là một ứng dụng chủ yếu nhất của PHP. Các PHP Developer sẽ phải thực hiện các thao tác như phân tích ngôn ngữ lập trình PHP, xây dựng máy chủ web và trình duyệt web.
- Tạo các dòng tập lệnh: Các lập trình viên sẽ tạo ra một dòng tập lệnh để vận hành chương trình PHP mà không cần đến máy chủ. Kiểu lập trình này được sử dụng trên các hệ điều hành phổ biến như Linux hay Windows.
- *Xây dựng các ứng dụng làm việc:* Bạn có thể ứng dụng những điểm mạnh vốn có của PHP để xây dựng ứng dụng phần mềm. Các lập trình viên thường dùng PHP – GTK làm nền tảng xây dựng phần mềm vì đây là

nhánh mở rộng của ngôn ngữ lập trình này và không có sẵn trong các bản phân phối chính thức hiện nay.

- *Hỗ trợ cho mọi loại cơ sở dữ liệu khác nhau:* Khi một website có hỗ trợ cơ sở dữ liệu tốt sẽ giúp ích cho việc vận hành, sao lưu và đặc biệt là backup dữ liệu đề phòng trường hợp xảy ra an ninh mạng.

1.3) Localhost

1.3.1) Khái niệm

[6] Localhost được sử dụng để truy cập vào các dịch vụ mạng đang vận hành trên máy tính đó bằng một cổng mạng loopback. Bằng cách này, nó không sử dụng bất kỳ cổng mạng vật lý nào để thực hiện kết nối tới chính nó. Máy tính giờ đây hoạt động dưới dạng một hệ thống mạng ảo, chạy ngay bên trong nó.

Localhost cơ bản nó như một webserver bao gồm: Apache, MySQL, PHP và PHPmyadmin. Chúng được cài đặt và sử dụng trên chính chiếc máy tính của bạn, dùng chính ổ cứng máy tính để làm không gian lưu trữ và cài đặt trang web. Mục đích chính để giúp bạn học tập và thực hành trên đó mà chưa cần mua host.

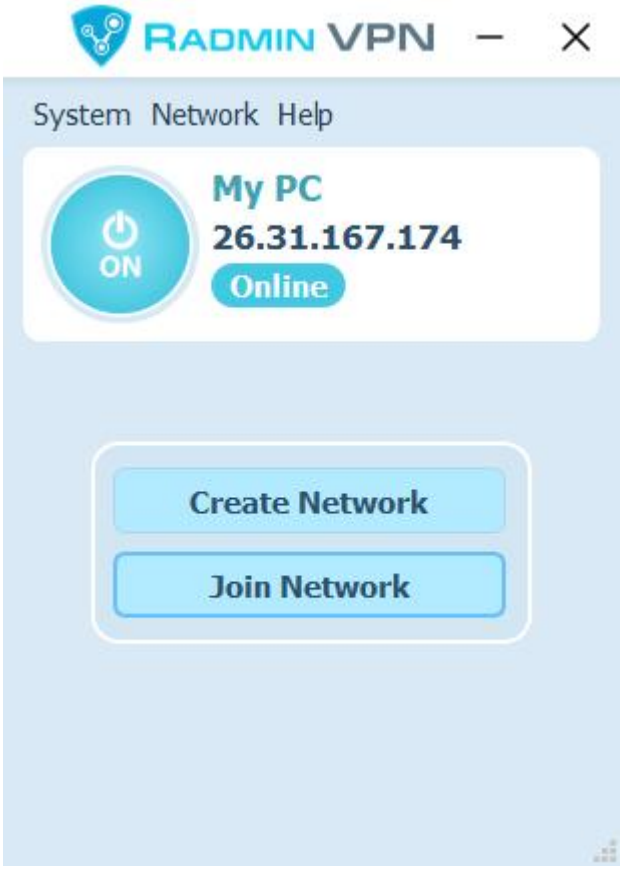
Ứng dụng Localhost có chức năng cài đặt và thử nghiệm các website trên máy tính. Giúp cho việc thao tác cũng như xử lý dữ liệu nhanh hơn. Không mất quá nhiều công sức là không lo mất kết nối như online hosting. Vì nó được đặt trên chính chiếc máy tính của bạn nên chỉ bạn mới có thể xem được trang web mà bạn cài đặt trên localhost mà người khác không thể xem được.

2) Phần mềm sử dụng

2.1) Radmin VPN

2.1.1) Giới thiệu

[7]Radmin VPN là một giải pháp VPN mạnh mẽ và dễ sử dụng, được thiết kế để tạo ra các kết nối an toàn giữa các thiết bị từ xa.



Giao diện Radmin VPN

2.1.2) Tính năng

[7]Radmin VPN là một phần mềm miễn phí giúp tạo một mạng LAN ảo giữa nhiều máy tính. Điều này giúp người dùng kết nối với các máy tính khác thông qua mạng internet như thể họ đang ở trong cùng một mạng LAN.

2.1.3) SQL server

2.1.4) Khái niệm

[8]SQL Server (hay Microsoft SQL Server) là một hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) được phát triển bởi Microsoft.

SQL Server cung cấp cho người dùng các công cụ và tính năng để quản lý, lưu trữ, xử lý các truy vấn dữ liệu, kiểm soát truy cập, xử lý giao dịch và hỗ trợ tích hợp dữ liệu từ nhiều nguồn khác nhau.

Ngoài ra, SQL Server cũng cung cấp các công cụ để tạo báo cáo, phân tích và quản lý cơ sở dữ liệu trực quan thông qua giao diện người dùng hoặc các script lệnh SQL.

SQL Server được xây dựng dựa trên SQL, một ngôn ngữ lập trình tiêu chuẩn để tương tác với cơ sở dữ liệu quan hệ. SQL Server được liên kết với Transact-SQL hoặc T-SQL, triển khai SQL của Microsoft có bổ sung một tập hợp các cấu trúc lập trình độc quyền.

SQL Server hoạt động độc quyền trên môi trường Windows trong hơn 20 năm. Vào năm 2016, Microsoft đã cung cấp SQL Server trên Linux. SQL Server 2017 ra mắt vào tháng 10 năm 2016 chạy được trên cả Windows và Linux.

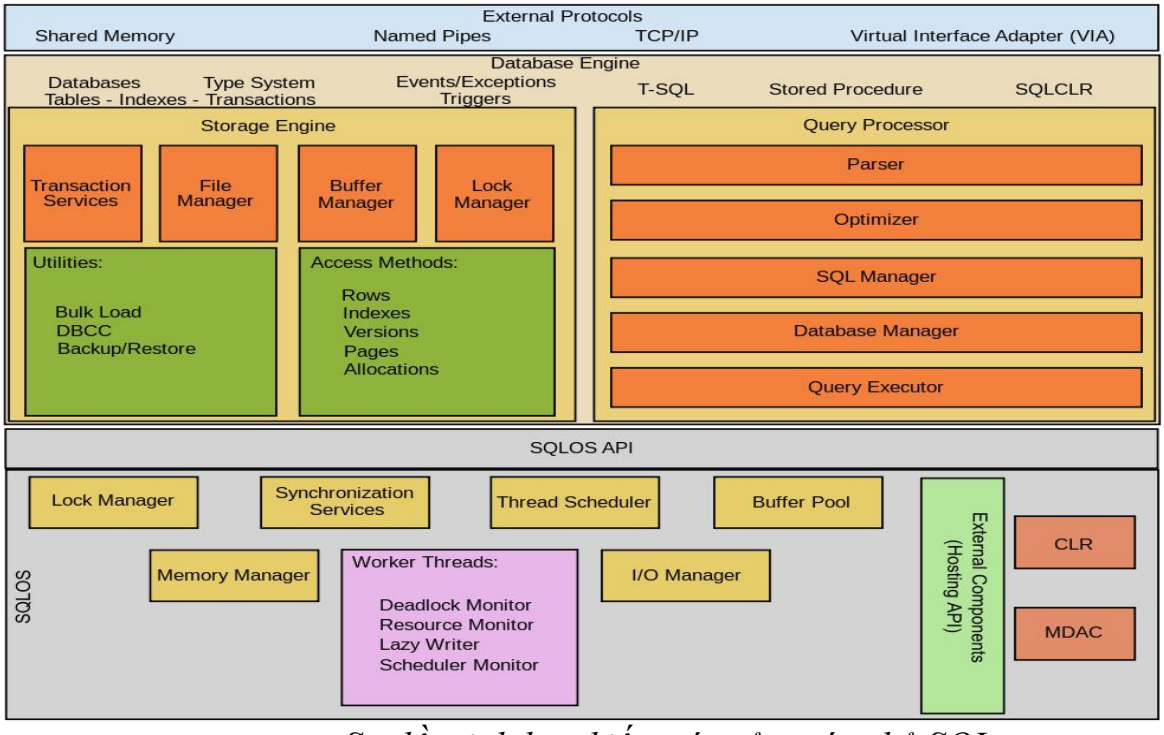
SQL Server thường đi kèm với việc thực hiện riêng các ngôn ngữ SQL, T-SQL,... Cụ thể như sau:

T-SQL là một trong những loại ngôn ngữ thuộc quyền sở hữu của Microsoft và được gọi với cái tên Transact-SQL. Nó thường cung cấp thêm rất nhiều cho các khả năng khai báo biến, thủ tục lưu trữ và xử lý ngoại lệ,...

SQL Server Management Studio là một loại công cụ giao diện chính cho máy chủ cơ sở của chính dữ liệu SQL, thông thường thì nó hỗ trợ cho cả môi trường 64 bit và 32 bit.

SQL server

2.1.5) Cấu trúc của SQL server



Sơ đồ minh họa kiến trúc của máy chủ SQL

SQL Server bao gồm hai thành phần chính:

+Database Engine:

[8]Thành phần cốt lõi của SQL Server là database engine. Thành phần này bao gồm một công cụ quan hệ có chức năng xử lý các lệnh và truy vấn, một công cụ lưu trữ quản lý các tệp, bảng, trang, index, bộ đệm và giao dịch cơ sở dữ liệu.

Các nhiệm vụ, trigger, trình xem và các đối tượng dữ liệu lưu trữ khác cũng được Database Engine khởi tạo và xử lý.

+SQLOS:

[8]SQLOS là tầng cuối cùng trong kiến trúc tổng thể của SQL Server. SQLOS cung cấp nhiều hệ điều hành như quản lý bộ nhớ và I/O. Ngoài ra, còn có các dịch vụ khác như dịch vụ xử lý ngoại lệ và đồng bộ hóa.

2.1.6) Các thành phần chính của SQL server

Các thành phần của SQL server

Các thành phần chính của SQL Server bao gồm:

+SQL Server Database Engine: [8]Là thành phần cốt lõi của SQL Server, nó quản lý và lưu trữ dữ liệu trong các cơ sở dữ liệu, cung cấp tính năng như truy vấn dữ liệu, xử lý giao dịch và kiểm soát truy cập.

+Integration Services (SSIS): [8]Là công cụ dùng để tích hợp dữ liệu từ nhiều nguồn khác nhau vào SQL Server. Nó cho phép bạn xử lý, chuyển đổi và chuyển dữ liệu giữa các hệ thống khác nhau.

+Analysis Services (SSAS): [8]Là công cụ phân tích dữ liệu cho phép bạn tạo các cube dữ liệu để phân tích. Nó cung cấp các công cụ để tìm hiểu mối quan hệ và xu hướng trong dữ liệu.

+Reporting Services (SSRS): [8]Là công cụ tạo báo cáo dữ liệu, cho phép người dùng tạo các báo cáo trực quan và dễ đọc từ các dữ liệu được lưu trữ trong SQL Server.

+SQL Server Management Studio (SSMS): [8]Là công cụ quản lý SQL Server, cho phép người dùng quản lý cơ sở dữ liệu, truy vấn và thực thi các tác vụ khác liên quan đến SQL Server.

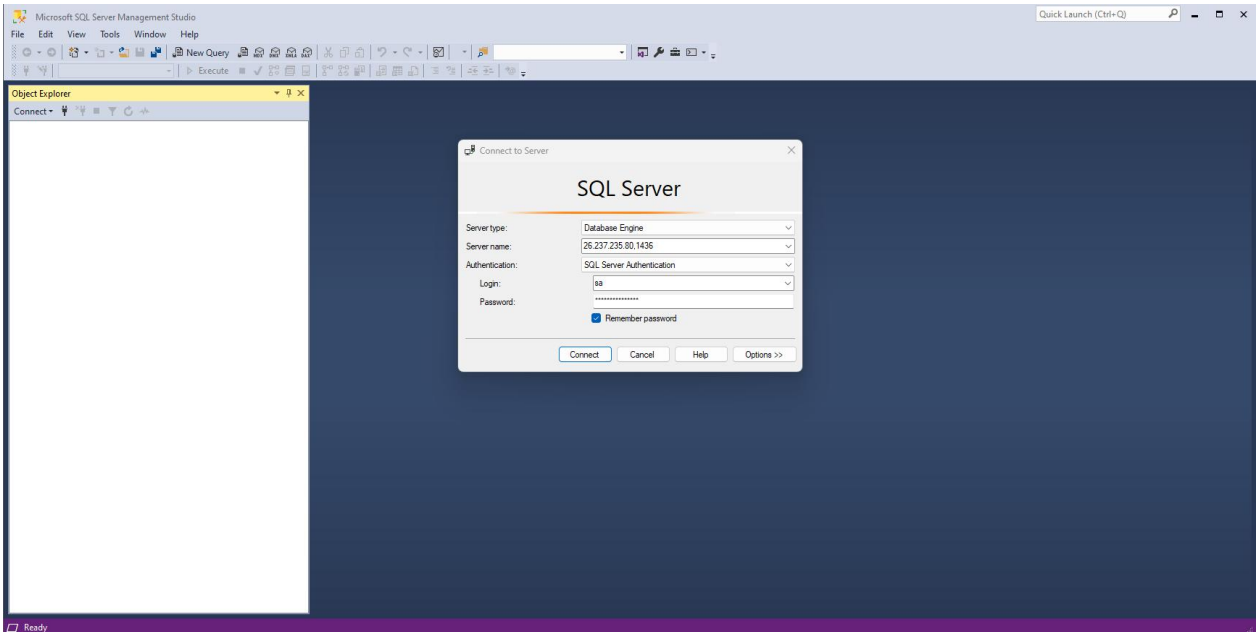
+SQL Server Data Tools (SSDT):[8] Là công cụ phát triển ứng dụng, cho phép người dùng tạo, thiết kế và triển khai các ứng dụng liên quan đến SQL Server.

+Azure SQL Database:[8] Là phiên bản SQL Server được đưa lên cloud của Microsoft, cho phép người dùng quản lý cơ sở dữ liệu và các ứng dụng của họ trên đám mây.

2.2) SQL Server Management Studio

1.4.1) Giới thiệu

SQL Server Management Studio (SSMS) [8] là một ứng dụng phần mềm được giới thiệu lần đầu thông qua Microsoft SQL Server 2005, được sử dụng để cấu hình, quản lý tất cả các thành phần trong Microsoft SQL Server.



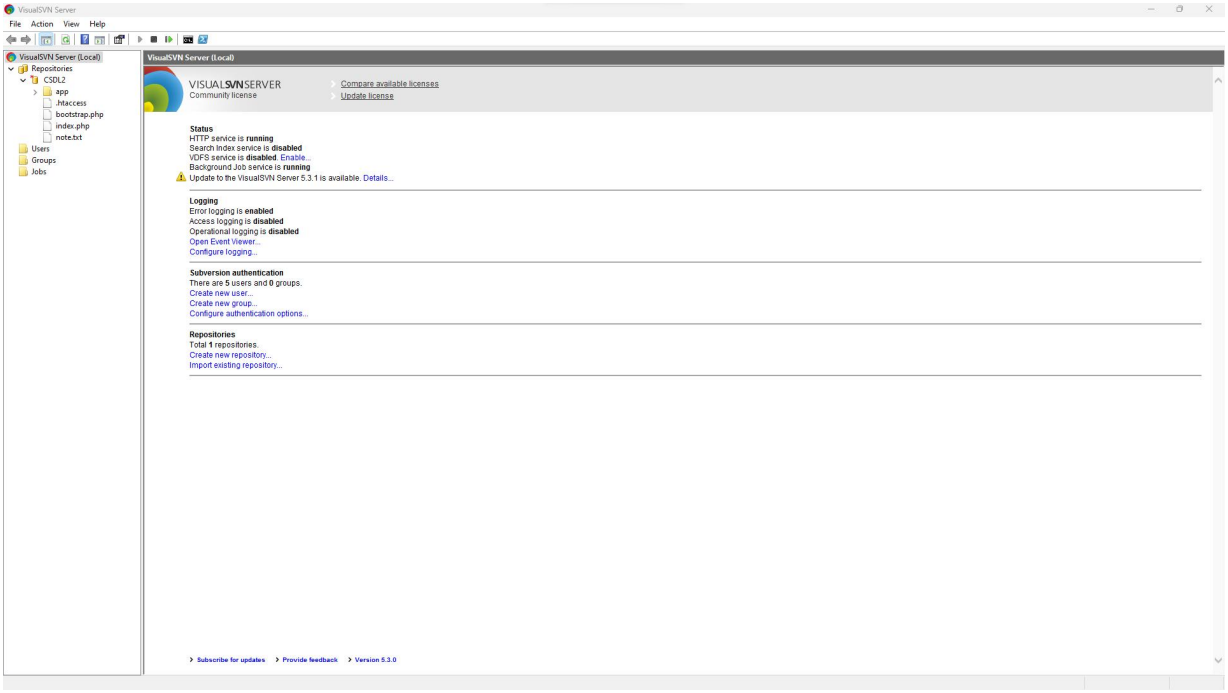
Giao diện ssms

Công cụ này cho phép người dùng truy vấn, quản lý và thiết kế database trên một máy tính cục bộ hay trên cloud.

2.3) VisualSVN server

2.3.1) Giới thiệu

Subversion(viết tắt là SVN) [9]là một hệ thống quản lí phiên bản được giới thiệu vào năm 2000 bởi công ty CollabNet. Đây là hệ thống hỗ trợ làm việc theo nhóm rất hiệu quả. Khi một nhóm làm việc cùng trên một project, việc nhiều người cùng chỉnh sửa nội dung của một file là điều không thể tránh khỏi. SVN cung cấp các chức năng để có thể thực hiện việc này một cách đơn giản và an toàn.



Giao diện VisualSVN server

2.3.2) Tính năng

Điểm đặc biệt của SVN [9] là nó lưu lại tất cả những gì thay đổi trên hệ thống file: file nào đã bị thay đổi, thay đổi lúc nào, thay đổi như thế nào, và ai đã thay đổi nó. SVN cũng cho phép recover lại những version cũ một cách chính xác. Các chức năng này giúp cho việc làm việc nhóm trở nên trơn tru và an toàn hơn rất nhiều.

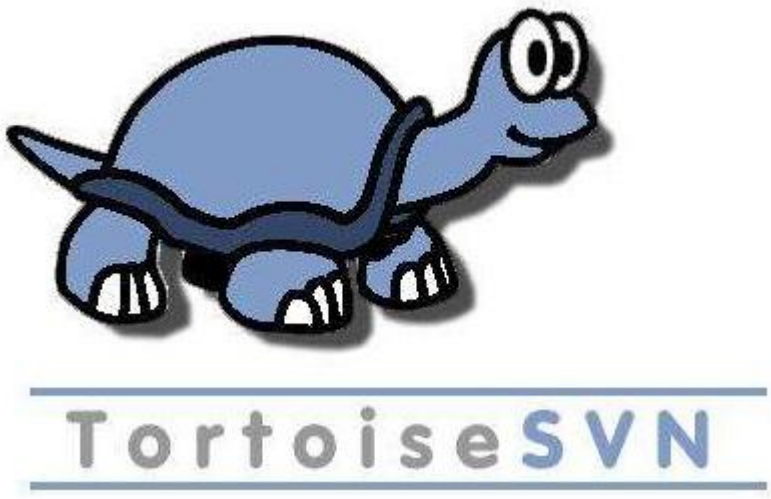
2.4) TortoiseSVN

1.6.1) Giới thiệu

TortoiseSVN [9]là một trình máy khách Windows mã nguồn mở miễn phí cho hệ thống kiểm soát phiên bản *Apache™ Subversion®* . TortoiseSVN quản lý tập tin và thư mục theo thời gian. Các tập tin được lưu trữ trong một kho trung tâm. Kho lưu trữ phần nhiều giống như một máy chủ tập tin bình thường, ngoại trừ việc nó nhớ tất cả các hay đổi đã từng được thực hiện trên các tập tin và thư mục của bạn. Điều này cho phép bạn khôi phục lại phiên bản cũ của tập tin của bạn và kiểm tra lịch sử của việc dữ liệu của bản đã được thay đổi như thế nào và khi nào, và ai đã thay đổi nó. Đây là lý do

tại sao nhiều người nghĩ rằng Subversion và các hệ thống kiểm soát phiên bản nói chung là một loại của " cỗ máy thời gian " .

Một số hệ thống kiểm soát phiên bản cũng là phần mềm quản lý cấu hình (SCM). Các hệ thống này được thiết kế đặc biệt để quản lý cây mã nguồn, và có nhiều tính năng cụ thể cho phát triển phần mềm - chẳng hạn như sự hiểu biết bản xứ về ngôn ngữ lập trình, hoặc cung cấp các công cụ để xây dựng phần mềm. Subversion, tuy nhiên, không phải là một trong những hệ thống này, nó là một hệ thống tổng quát có thể được sử dụng để quản lý *bất kỳ* bộ sưu tập các tập tin, bao gồm cả mã nguồn.



TortoiseSVN

1.6.2) Tính năng của TortoiseSVN

Kiểm tra và cam kết: TortoiseSVN cho phép người dùng kiểm tra các tập tin và thư mục từ kho lưu trữ Subversion và cam kết các thay đổi của họ trở lại kho lưu trữ.

Xem lịch sử: TortoiseSVN cho phép người dùng xem lịch sử của các thay đổi đã được thực hiện đối với một dự án Subversion.

Giải quyết xung đột: Nếu hai người dùng thực hiện các thay đổi đối với cùng một tập tin, TortoiseSVN có thể giúp giải quyết xung đột.

Các tính năng khác: TortoiseSVN cũng cung cấp một số tính năng khác, bao gồm:

+Khóa và mở khóa các tập tin để ngăn người khác chỉnh sửa chúng.

+Sao chép và di chuyển các tập tin và thư mục trong dự án.

+Tạo các báo cáo về lịch sử của dự án.

2.5) Visual studio code

1.7.1) Giới thiệu

Visual Studio Code (VS Code) [10]là một trình soạn thảo mã nguồn nhẹ nhưng mạnh mẽ, chạy trên máy tính và hỗ trợ Windows, macOS và Linux. Nó đi kèm với hỗ trợ tích hợp sẵn cho JavaScript, TypeScript và Node.js và có một hệ sinh thái mở rộng phong phú cho các ngôn ngữ khác (chẳng hạn như C ++, C#, Java, Python, PHP và Go) và thời gian chạy (chẳng hạn như .NET và Unity). Visual Studio Code được các chuyên gia công nghệ thông tin đánh giá cao, nó là sự kết hợp hoàn hảo giữa IDE và CODE Editor.

Visual Studio Code hỗ trợ chức năng debug, đi kèm với git, có syntax highlighting, tự hoàn thành mã thông minh, snippets, và cái tiến mã nguồn. Nhờ tính năng tùy chỉnh, Visual Studio Code cũng cho phép người dùng thảy đổi theme, phím tắt, và các tùy chọn khác.

Visual studio code

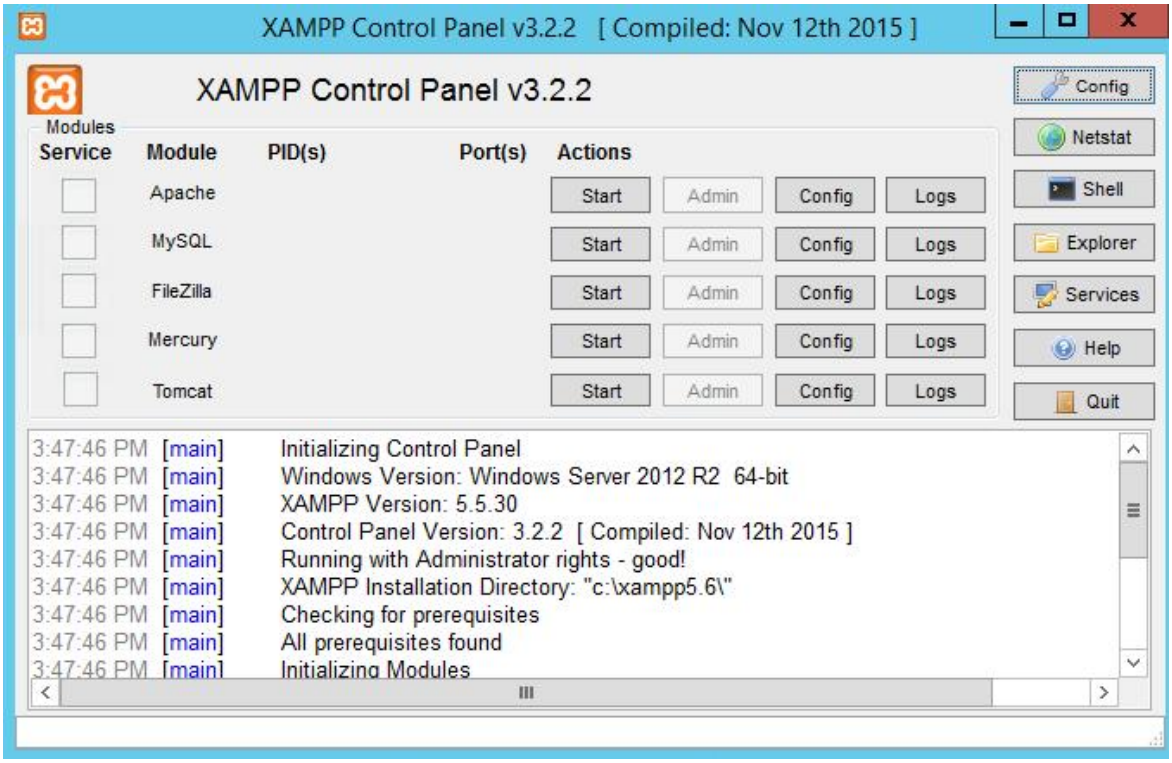
2.6) Xampp

2.6.1) Giới thiệu

XAMPP là [11]viết tắt của 5 module được tích hợp bên trong nó bao gồm là Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) và Perl (P). XAMPP là một phần mềm nguồn mở và miễn phí dùng để tạo web server trên máy tính cá nhân (Localhost), XAMPP tương thích với các hệ điều hành phổ biến như : Linux, MacOS, Windows,..

Ưu điểm lớn nhất của XAMPP là mã nguồn mở và tính dễ sử dụng, tương đối đơn giản, gọn nhẹ nên được sử dụng ngày càng phổ biến hiện nay.

XAMPP được ứng dụng rộng rãi từ người dùng phổ thông đến lập trình viên, nhằm để vận hành cũng như phát triển các website dùng ngôn ngữ lập trình PHP như: WordPress, Joomla!, Magento, Drupal, OpenCart, phpBB,..



Giao diện xampp

3) Yêu cầu khi xây dựng website

Yêu cầu về mô hình MVC:

Mô hình MVC cần được thiết kế rõ ràng và khoa học: Mô hình MVC cần được thiết kế rõ ràng để các thành viên trong nhóm dễ dàng hiểu và phát triển.

Mô hình MVC cần được triển khai đúng cách: Mô hình MVC cần được triển khai đúng cách để đảm bảo website hoạt động hiệu quả.

Yêu cầu về Radmin VPN:

Radmin VPN cần được cài đặt và cấu hình đúng cách: Radmin VPN cần được cài đặt và cấu hình đúng cách để đảm bảo các thành viên trong nhóm có thể sử dụng để dùng chung mạng LAN.

Yêu cầu về TortoiseSVN:

TortoiseSVN cần được cài đặt và cấu hình đúng cách: TortoiseSVN cần được cài đặt và cấu hình đúng cách để đảm bảo các thành viên trong nhóm có thể sử dụng để commit code lên máy chủ và update từ máy chủ.

Các thành viên trong nhóm cần được hướng dẫn sử dụng TortoiseSVN: Các thành viên trong nhóm cần được hướng dẫn sử dụng TortoiseSVN để đảm bảo họ có thể sử dụng để commit code lên máy chủ một cách hiệu quả.

4) Xây dựng website

4.1) Xây dựng mô hình mvc

Mô hình MVC trong hình được cấu trúc như sau:

+Thư mục configs: chứa các file cấu hình của ứng dụng, chẳng hạn như cấu hình cơ sở dữ liệu, cấu hình xác thực người dùng.

+Thư mục controllers: chứa các lớp controller của ứng dụng. Các lớp controller xử lý các yêu cầu của người dùng và trả về các phản hồi.

+Thư mục core: chứa các thành phần cốt lõi của ứng dụng, bao gồm controllers, connection, database, route

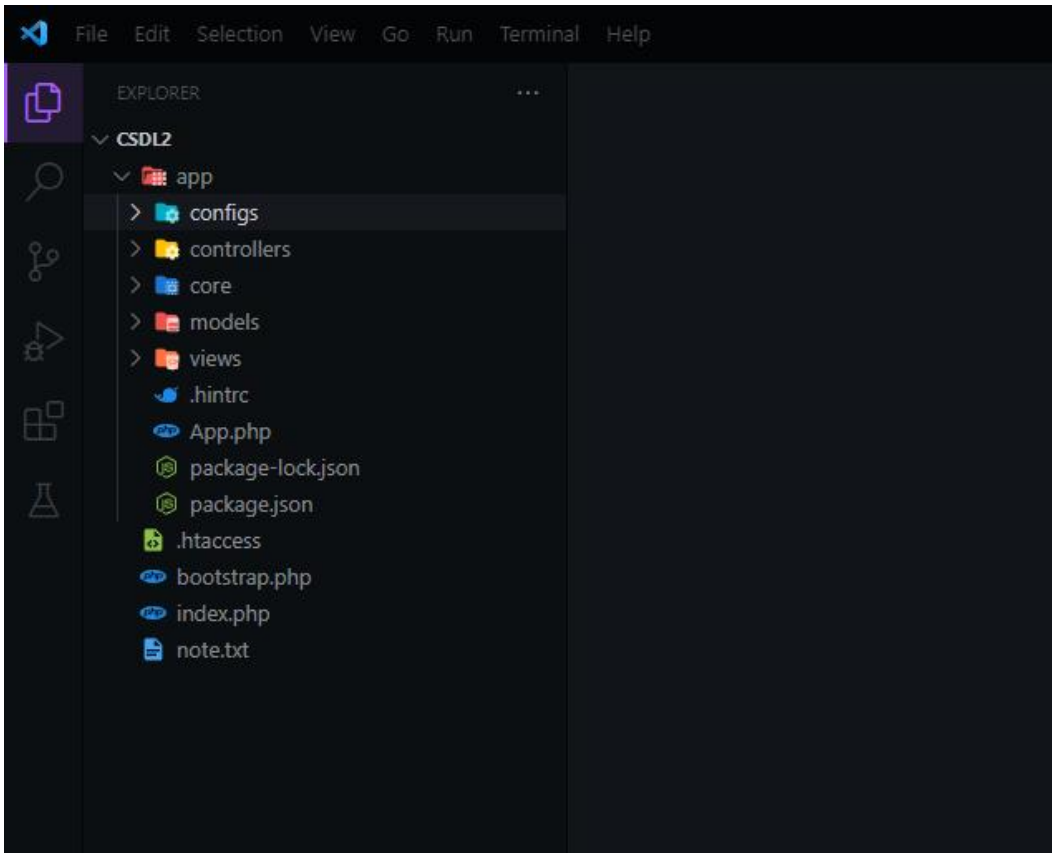
+Thư mục models: chứa các lớp đại diện cho dữ liệu của ứng dụng. Các lớp này thường được sử dụng bởi các controller để truy cập và thao tác dữ liệu.

+Thư mục views: chứa các file HTML hiển thị giao diện người dùng của ứng dụng. Các file này thường được sử dụng bởi các controller để hiển thị dữ liệu cho người dùng.

+Các file khác:

Ngoài các thư mục trên, mô hình MVC này còn chứa một số file khác, bao gồm:

- App.php: File khởi tạo ứng dụng.
- .htaccess: File cấu hình cho web server Apache.
- bootstrap.php: File có nhiệm vụ khởi tạo ứng dụng và chuẩn bị các thành phần cần thiết cho ứng dụng hoạt động.
- index.php: File có nhiệm vụ khởi tạo ứng dụng và xử lý các yêu cầu của người dùng.
- note.txt: File ghi chú.



Mô hình mvc website quarantine camp

Mô hình này tuân theo các nguyên tắc của mô hình MVC, cụ thể:

+Controllers: Các controller xử lý các yêu cầu của người dùng và trả về các phản hồi.

+Models: Các models đại diện cho dữ liệu của ứng dụng.

+Views: Các views hiển thị giao diện người dùng của ứng dụng.

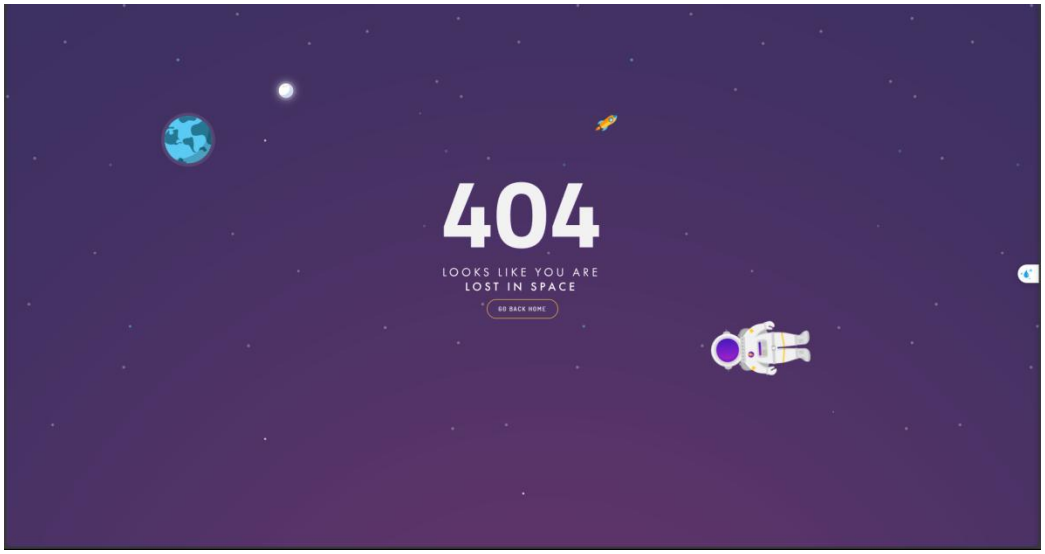
4.2) Font end

4.2.1) ERRORS

Trong mô hình MVC, folder errors nằm trong folder views và chứa các file hiển thị các lỗi cho người dùng. Các file này thường được sử dụng khi ứng dụng gặp lỗi và không thể xử lý yêu cầu của người dùng.

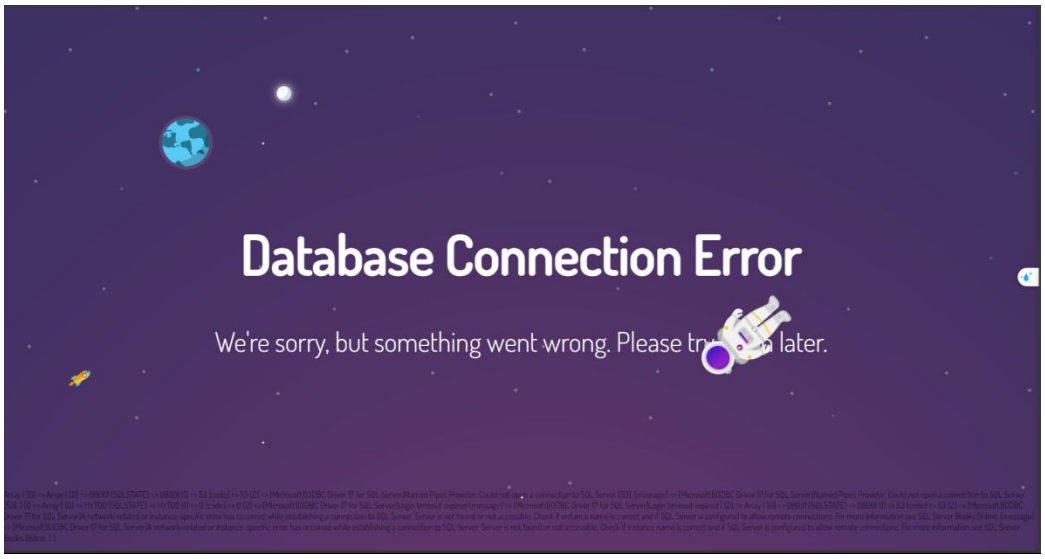
Folder errors chứa các file sau:

+404.php: File này được hiển thị khi người dùng cố gắng truy cập một không tồn tại. File này thường chứa thông báo lỗi "Trang không tồn tại" cùng với liên kết trở lại trang chủ.



lỗi trang không tồn tại.

+database.php: File này được hiển thị khi ứng dụng gặp lỗi khi truy cập cơ sở dữ liệu.



Lỗi database

Các file trong folder errors có thể được tùy chỉnh để hiển thị các thông tin lỗi theo cách phù hợp với ứng dụng. Ví dụ, file 404.php có thể được hiển thị thông báo lỗi "Trang không tồn tại" cùng với liên kết trở lại trang chủ.

4.3) Database

4.3.1) Config

Trong config.php khai báo một mảng có tên là $config. Mảng này chứa các thông tin cấu hình cho một cơ sở dữ liệu, bao gồm:

serverName: Tên máy chủ cơ sở dữ liệu. Trong trường hợp này, máy chủ cơ sở dữ liệu có địa chỉ IP là 26.237.235.80 và cổng là 1436.

uid: Tên người dùng cơ sở dữ liệu.

pass: Mật khẩu cơ sở dữ liệu.

db: Tên cơ sở dữ liệu.

Đoạn code sử dụng cú pháp sau để khai báo mảng:

```
$config["database"] = [
  'serverName' => '26.237.235.80,1436',
  'uid'=> 'sa',
  'pass'=> '12345',
```

```
  'db'=> 'QuarantineCampDatabase'
];
```

Trong đó, mỗi phần tử của mảng được khai báo bằng dấu =>. Giá trị của mỗi phần tử được đặt trong dấu ngoặc kép.

4.3.2) Connection

Đoạn code trong Connection.php:

```
class Connection extends Controller{
    private static $instance = null;
    private Controller $controller;
    public static function connection($config){
        $serverName = $config["serverName"];
        $database = $config["db"];
        $uid = $config["uid"];
        $pass = $config["pass"];
        $connection = [
            "Database" => $database
            ,"Uid"=> $uid
            ,"PWD"=> $pass
        ];
        $conn = sqlsrv_connect($serverName, $connection);
        if ($conn === false) {
            require_once 'app/views/errors/database.php';
            die(print_r(sqlsrv_errors(), true));
        }
        return $conn;
```

```
    }
```

Đoạn code này định nghĩa một lớp Connection trong PHP. Lớp này có chức năng thiết lập và quản lý kết nối tới cơ sở dữ liệu SQL Server.

Phân tích chi tiết:

public static function connection($config): Đây là phương thức tĩnh của lớp, chịu trách nhiệm thiết lập kết nối dựa trên cấu hình được truyền vào ($config).

$serverName và $database: Hai biến này trích xuất thông tin tên máy chủ và tên database từ config .

$uid và $pass: Hai biến này trích xuất thông tin của user id và password từ config

$connection: Mảng này lưu trữ các thông tin kết nối gồm tên database, uid, và pass.

sqlsrv_connect: Hàm này được sử dụng để thực hiện việc kết nối tới database với các thông tin đã chuẩn bị.

Kiểm tra kết nối: Nếu kết nối thất bại ($conn === false), mã sẽ yêu cầu file lỗi database và hiển thị các lỗi cụ thể bằng sqlsrv_errors().

Trả về kết nối: Nếu kết nối thành công, phương thức sẽ trả về đối tượng $conn để sử dụng cho các truy vấn database sau này.

Mục đích sử dụng:

Lớp Connection được dùng để đơn giản hóa việc thiết lập kết nối với database SQL Server trong ứng dụng PHP. Nó cung cấp một phương thức tĩnh connection để lấy một đối tượng kết nối dựa trên cấu hình, xử lý các lỗi kết nối, và trả về đối tượng kết nối để sử dụng trong các lớp khác.

4.3.3) Database

Đoạn code trong Database.php:

```
<?php
class Database{
    private $__conn;
    public function __construct(){
```

```
        global $db_config;
        $this->__conn =
Connection::connection($db_config);
    }
    public function create($table, $data) {
        // Build SQL statement dynamically using $data
array
        $sql = "INSERT INTO " . $table . " (";
        $columns = [];
        $values = [];
              foreach ($data as $column => $value) {
          $columns[] = $column;
                if ($value === null) {
            $values[] = "NULL";
          } else {
            $value = str_replace('\'', '\\\'', $value);
            $values[] = "N'" . $value . "'";
          }
        }

        $sql .= implode(", ", $columns) . ") VALUES (" .
implode(", ",          $values) . ")";
        // Prepare and execute SQL statement
        $stmt = sqlsrv_prepare($this->__conn, $sql);
        if (sqlsrv_execute($stmt) === false) {
          die(print_r(sqlsrv_errors(), true));
        }

        // Close connection
        sqlsrv_close($this->__conn);
      }
      public function update($table, $data, $condition)
{
        // Build SQL statement dynamically using $data
and $condition              arrays
        $sql = "UPDATE " . $table . " SET ";
```

```
        $updates = [];
        foreach ($data as $column => $value) {
          if ($value === null) {
            $updates[] = $column . " = NULL";
          } else {
            // Thêm tiền tố N cho giá trị chuỗi
            $value = str_replace('\'', '\\\'', $value);
// Escape dấu               nháy đơn
            $updates[] = $column . " = N'" . $value .
"'";
          }
        }
        $sql .= implode(", ", $updates);
        $sql .= " WHERE " . $condition;
        // Prepare and execute SQL statement
        $stmt = sqlsrv_prepare($this->__conn, $sql);

        if (sqlsrv_execute($stmt) === false) {
          die(print_r(sqlsrv_errors(), true));
        }
        // Close connection
        sqlsrv_close($this->__conn);
      }
    public function delete($table, $condition) {
        // Build SQL statement dynamically using
$condition array
        $sql = "DELETE FROM " . $table . " WHERE " .
$condition;
            // Prepare and execute SQL statement
        $stmt = sqlsrv_prepare($this->__conn, $sql);
        if (sqlsrv_execute($stmt) === false) {
            die(print_r(sqlsrv_errors(), true));
        }
        // Close connection
        sqlsrv_close($this->__conn);
    }
```

```
    public function query($sql, $params = []) {
        // Execute the provided SQL statement with
parameters
        $stmt = sqlsrv_query($this->__conn, $sql,
$params);
            if (!$stmt) {
            die(print_r(sqlsrv_errors(), true));
        }
            // Fetch all results as an associative array
        $results = [];
        while ($row = sqlsrv_fetch_array($stmt,
SQLSRV_FETCH_ASSOC)) {
            $results[] = $row;
        }
            // Free resources associated with the
statement
        sqlsrv_free_stmt($stmt);
            return $results;
    }
    public function countRows($table) {
        $sql = "SELECT COUNT(*) AS num_rows FROM $table";
        $results = $this->query($sql);
        return $results[0]['num_rows']; // Lấy giá trị
'num_rows' từ         kết quả đầu tiên
      }
```

Đoạn code này định nghĩa một lớp Database trong PHP có chức năng thực hiện các truy vấn cơ bản trên cơ sở dữ liệu SQL Server.

Phân tích chi tiết:

private $__conn: Biến riêng lưu trữ đối tượng kết nối với database, được thiết lập trong hàm khởi tạo.

__construct: Hàm khởi tạo kết nối với database bằng cách sử dụng lớp Connection và tham số cấu hình toàn cục $db_config.

create: Hàm này thực hiện việc thêm mới bản ghi vào một bảng cụ thể.

Nó xây dựng câu lệnh INSERT dựa trên mảng dữ liệu được truyền vào.

Xử lý đặc biệt cho giá trị null và escape ký tự đơn để đảm bảo an toàn.

Chuẩn bị và thực thi câu lệnh bằng sqlsrv_prepare và sqlsrv_execute.

Sau đó đóng kết nối với database.

update: Hàm này cập nhật một hoặc nhiều bản ghi trong bảng dựa trên mảng dữ liệu và điều kiện.

Tương tự create, nó xây dựng câu lệnh UPDATE động.

Xử lý giá trị null và escape ký tự đơn.

Đóng kết nối sau khi thực thi.

delete: Hàm này xóa một hoặc nhiều bản ghi dựa trên điều kiện.

Xây dựng câu lệnh DELETE dựa trên điều kiện được truyền vào.

Chuẩn bị và thực thi câu lệnh, đóng kết nối sau khi thực hiện.

query: Hàm này thực thi một câu lệnh SQL bất kỳ với tham số tùy chọn.

Nó sử dụng sqlsrv_query để thực thi câu lệnh.

Kiểm tra lỗi và giải phóng tài nguyên liên quan tới câu lệnh.

Trả về kết quả dưới dạng mảng các bản ghi kết hợp (associative array).

countRows: Hàm này tính tổng số bản ghi trong một bảng.

Nó thực thi câu lệnh SELECT COUNT(*) để đếm số bản ghi.

Sử dụng query để lấy kết quả và trả về giá trị 'num_rows' từ bản ghi đầu tiên.

4.4) Back end

4.4.1) App

Đoạn code trong App.php:

```
<?php
class App{

    private $__controller, $__action, $__params,
$__routes;
    static public $app;
    function __construct(){
                global $routes, $config;
        self::$app = $this;
        $this->__routes = new Route() ;
        if(!empty($routes['default_controller'])){
            $this->__controller =
$routes['default_controller'];
        }
               $this->__action ='index';
        $this->__params =[];

       $this->handleUrl();
    }
    function getUrl(){
        if(!empty($_SERVER['PATH_INFO'])){
            $url = $_SERVER['PATH_INFO'];
        }else{
            $url ='/';
        }
        return $url;
    }
    public function handleUrl(){
                $url = $this ->getUrl();
        $url = $this->__routes->handleRoute($url);
        $urlArr = array_filter(explode('/',$url));
        $urlArr = array_values($urlArr);
        //xu ly controller
```

```
        if(!empty($urlArr[0])){
            $this->__controller = ucfirst($urlArr[0]);
        }else{
            $this->__controller = ucfirst($this-
>__controller);
        }
        if(file_exists('app/controllers/'.($this-
>__controller).'.php')){
            require_once 'app/controllers/'.($this-
>__controller).'.php';
            if(class_exists($this->__controller)){
                $this->__controller = new $this-
>__controller();
                unset($urlArr[0]);
            }else{
                $this->loadError();
            }

        }else{
            $this->loadError();
        }

        //xu ly action
        if(!empty($urlArr[1])){
            $this->__action = $urlArr[1];
            unset($urlArr[1]);
        }

        //xu ly params
        $this->__params = array_values($urlArr);

        // check  method
        if(method_exists($this->__controller, $this-
>__action)){
            call_user_func_array([$this->__controller,
$this->__action], $this->__params);
```

```
        }else{
            $this->loadError();
        }



    }
    public function loadError($name = '404', $data= []){
        extract($data);
        require_once 'app/views/errors/'.$name.'.php';
    }
}
```

Đoạn code này định nghĩa một lớp App đóng vai trò chính trong mô hình MVC. Mục đích chính của lớp này là:

Xử lý URL và định hướng yêu cầu đến controller và action phù hợp.

Quản lý các routes (đường dẫn) và ánh xạ chúng đến các controller và action cụ thể.

Tạo các instance của controller và gọi các action của chúng dựa trên URL.

Xử lý các lỗi xảy ra trong quá trình xử lý URL và hiển thị trang lỗi phù hợp.

Phân tích chi tiết:

Thuộc tính:

$__controller: Lưu trữ tên của controller được yêu cầu.

$__action: Lưu trữ tên của action được yêu cầu.

$__params: Lưu trữ các tham số truyền qua URL.

$__routes: Lưu trữ một instance của lớp Route để quản lý các routes.

$app: Biến static lưu trữ instance của lớp App (singleton pattern).

Phương thức:

__construct(): Khởi tạo lớp, thiết lập các giá trị mặc định và instance của Route.

getUrl(): Lấy URL từ server.

handleUrl(): Xử lý URL theo các bước:

Định hướng URL dựa trên routes.

Phân tích URL thành các phần: controller, action, params.

Kiểm tra và nạp controller.

Kiểm tra và gọi action của controller.

Xử lý lỗi nếu không tìm thấy controller, action hoặc method.

loadError(): Hiển thị trang lỗi tùy theo tên lỗi và dữ liệu truyền vào.

4.4.2) Routes

Đoạn code trong Routes.php:

```
$routes['default_controller'] = 'home';
$routes['home'] = 'home/index';
$routes['patient'] = 'patient/list1';
$routes['patient/detail'] = 'patient/detail';
$routes['form'] = 'form/index';
$routes['room'] = 'room/index';
$routes['patient/test'] = 'patient/test';
$routes['add'] ='form/add';
$routes['search'] ='patient/search';
```

Đoạn code này định nghĩa một bảng định tuyến (routing table) trong PHP. Mục đích của nó là ánh xạ các URL của ứng dụng tới các controller và action cụ thể để xử lý yêu cầu của người dùng.

Phân tích chi tiết:

Mỗi dòng trong bảng là một cặp key-value. Key là một URL hoặc một phần của URL, còn value là một chuỗi định dạng "controller/action".

Key $routes['default_controller'] thiết lập controller mặc định của ứng dụng là home. Nếu URL không khớp với bất kỳ key nào khác, controller home và action index sẽ được gọi.

Các key khác như $routes['patient'] và $routes['patient/detail'] ánh xạ các URL cụ thể tới các controller và action tương ứng. Ví dụ, truy cập /patient sẽ gọi controller patient và action list1, còn /patient/detail sẽ gọi controller patient và action detail.

Key $routes['add'] ánh xạ URL /add tới controller form và action add, cho biết đây là chức năng thêm mới.

Key $routes['search'] ánh xạ URL /search tới controller patient và action search, cho biết đây là chức năng tìm kiếm bệnh nhân.

Mục đích sử dụng:

Routing table giúp tổ chức code của ứng dụng rõ ràng hơn bằng cách tách biệt logic xử lý URL khỏi logic xử lý nghiệp vụ.

Nó cho phép định nghĩa các URL thân thiện với người dùng và dễ nhớ.

Giúp điều hướng người dùng tới các chức năng phù hợp dựa trên URL họ truy cập.

Ví dụ:

Nếu người dùng truy cập /patient/detail/PT00001, đoạn mã này sẽ ánh xạ URL tới controller patient, action detail, và truyền tham số "PT00001" để hiển thị thông tin chi tiết về bệnh nhân có mã đó.

4.4.3) Controller

Đoạn code trong Controller.php:

```
<?php
class Controller{
    public function model($model){
        if(file_exists(_DIR_ROOT.'/app/models/'.$model.'.
php')){
            require_once
_DIR_ROOT.'/app/models/'.$model.'.php';
            // if(class_exists($model)){
            //     $model = new $model();
            //     return $model;
            // }
        }
```

```
        return false;
    }
      public function render($view, $data =[]){
        extract($data);
         if(file_exists(_DIR_ROOT.'/app/views/'.$view.'.
php')){
            require_once
_DIR_ROOT.'/app/views/'.$view.'.php';
        }
    }
}
```

Đoạn code này định nghĩa một lớp Controller cơ bản trong PHP, cung cấp hai chức năng chính:

Khởi tạo model:

Phương thức model nhận tên của một model làm tham số.

Nó kiểm tra xem file model tương ứng có tồn tại trong thư mục /app/models/ hay không.

Nếu file tồn tại, nó sẽ sử dụng require_once để include file đó.

Bình luận: Khối lệnh if(class_exists($model)) bị chú thích, vì nó không cần thiết. Nếu file model được include thành công, lớp sẽ được định nghĩa sẵn.

Trả về: Phương thức trả về false nếu file model không tồn tại, nếu không thì không trả về gì.

Hiển thị view:

Phương thức render nhận tên của một view và một mảng dữ liệu làm tham số.

Nó sử dụng extract để giải nén các key-value trong mảng dữ liệu thành các biến riêng lẻ trong scope hiện tại.

Nó kiểm tra xem file view tương ứng có tồn tại trong thư mục /app/views/ hay không.

Nếu file tồn tại, nó sẽ sử dụng require_once để include file đó.

Bình luận: Mục đích của extract là cho phép truy cập trực tiếp các biến trong view mà không cần truy cập qua mảng $data.

Mục đích sử dụng:

Lớp Controller này đóng vai trò trung gian giữa các URL và logic xử lý nghiệp vụ của ứng dụng.

Nó giúp tải và khởi tạo các model, cung cấp quyền truy cập tới các phương thức và thuộc tính của chúng.

Nó giúp hiển thị các view tương ứng với URL được truy cập, truyền dữ liệu cần thiết vào view để hiển thị nội dung động.

Bằng cách tách biệt logic model, controller, và view, mã code trở nên rõ ràng, dễ bảo trì và mở rộng hơn.

4.4.3)Controllers

4.4.3.1)Home

Đoạn code trong Home.php:

```
<?php
class Home extends Controller{
    public $model;
    public function __construct(){
            $this->model('PatientModel');
       $this->model('WarningModel');
    }
    public function index(){
       $data =[];
       $data['p'] = PatientModel::CountRows();
       $data['pd'] = PatientModel::PD();
       $data['pw']= WarningModel::PW();
       $data['warning'] = WarningModel::getAllWarning();
```

```
        $this->render("home/index", $data);
    }
}
```

Đoạn code này định nghĩa một lớp Home kế thừa lớp Controller trong PHP. Lớp này có chức năng hiển thị trang chủ của ứng dụng.

Phân tích chi tiết:

public $model: Biến public này được khai báo để lưu trữ các model được sử dụng.

__construct: Hàm khởi tạo thực hiện việc tải hai
model: PatientModel và WarningModel sử dụng phương thức model của lớp Controller.

index: Phương thức này chịu trách nhiệm hiển thị trang chủ.

$data: Mảng này lưu trữ dữ liệu cần truyền vào view.

PatientModel::countRows(): Dòng này gọi phương thức countRows của model PatientModel để lấy tổng số bệnh nhân.

PatientModel::countDischarged(): Gọi phương thức countDischarged của model PatientModel để lấy số bệnh nhân xuất viện.

WarningModel::countCritical(): Gọi phương thức countCritical của model WarningModel để lấy số bệnh nhân đang ở tình trạng nguy cấp.

WarningModel::getAllWarnings(): Gọi phương thức getAllWarnings của model WarningModel để lấy danh sách tất cả cảnh báo.

$this->render: Cuối cùng, phương thức render được gọi để hiển thị view "home/index" với dữ liệu đã chuẩn bị trong mảng $data.

Mục đích sử dụng:

Lớp Home thực hiện các chức năng sau:

Tải các model cần thiết cho trang chủ.

Lấy dữ liệu từ các model: tổng số bệnh nhân, số bệnh nhân xuất viện, số bệnh nhân đang ở tình trạng nguy cấp, danh sách tất cả cảnh báo.

Chuẩn bị mảng dữ liệu để truyền vào view.

Hiển thị view "home/index" với nội dung động dựa trên dữ liệu lấy được.

4.4.3.2)Room

Đoạn code trong Room.php:

```
class Room extends Controller{
    public $model;
    public function __construct(){

       $this->model('RoomModel');
    }
    public function index(){
        $data =[];
        $data['room'] = RoomModel::getAllRoom();
        $this->render("room/room", $data);
    }
}
```

Đoạn code này định nghĩa một lớp Room kế thừa lớp Controller trong PHP. Lớp này có chức năng hiển thị danh sách các phòng.

Phân tích chi tiết:

public $model: Biến public này được khai báo để lưu trữ model được sử dụng.

__construct: Hàm khởi tạo thực hiện việc tải model RoomModel sử dụng phương thức model của lớp Controller.

index: Phương thức này chịu trách nhiệm hiển thị danh sách phòng.

$data: Mảng này lưu trữ dữ liệu cần truyền vào view.

RoomModel::getAllRoom(): Dòng này gọi phương thức getAllRoom của model RoomModel để lấy danh sách tất cả các phòng.

$this->render: Cuối cùng, phương thức render được gọi để hiển thị view "room/room" với dữ liệu các phòng đã chuẩn bị trong mảng $data.

Mục đích sử dụng:

Lớp Room thực hiện các chức năng sau:

Tải model quản lý phòng.

Lấy danh sách tất cả các phòng từ model.

Chuẩn bị dữ liệu danh sách phòng vào mảng.

Hiển thị view "room/room" với danh sách phòng động.

4.4.3.3)Patient

Đoạn code trong Patient.php:

```
<?php
class Patient extends Controller{
    public function __construct(){
        $this->model('PatientModel');
        $this->model('PatientSymptomModel');
        $this->model('PatientComorbidityModel');
        $this->model('PCRTestModel');
        $this->model('QuickTestModel');
        $this->model('SPO2TestModel');
        $this->model('RespiratoryRateTestModel');
        $this->model('WarningModel');
        $this->model('TreatmentModel');
        $this->model('PatientHistoryLocationModel');
     }
     public function index(){
     }
        public function list1(){
        $Patients = PatientModel::getAllPatient();
```

```
        $this->render('patients/patient', $Patients);
    }
    public function test(){
        $data = [];
        $id = base64_decode($_REQUEST['id']);
        $data['PCR'] =  PCRTestModel::getPCRById($id);
        $data['QUICK']
     =  QuickTestModel::getQuickById($id);
        $data['SPO2'] =  SPO2TestModel::getSPO2ById($id);
        $data['RR']
     =  RespiratoryRateTestModel::getRRById($id);
        $this->render('patients/test_table',$data);
    }
    public function detail(){
        $data = [];
        $id = base64_decode($_REQUEST['id']);
        // $result =
PatientModel::getPatientById('PT00001');
        // //$result = PatientModel::getPatientById();
        // echo '<pre>';
        // print_r($result);
        // echo '</pre>';
        $data['patient'] =
PatientModel::getPatientById($id);
        $data['symptom'] =
PatientSymptomModel::getPatientSymptomById($id);
```

```
        $data['comorbidity'] =
PatientComorbidityModel::getPatientComorbidityById($id);
        $data['PCR'] =  PCRTestModel::getPCRById($id);
        $data['QUICK']
=  QuickTestModel::getQuickById($id);
        $data['SPO2'] =  SPO2TestModel::getSPO2ById($id);
        $data['RR']
=  RespiratoryRateTestModel::getRRById($id);
        $data['warning'] =
WarningModel::getWarningById($id);
        $data['treatment'] =
TreatmentModel::getTreatmentById($id);
        $data['phl'] =
PatientHistoryLocationModel::getPHLById($id);


        $this->render('patients/detail',$data);
    }


    public function search() {
    $data = [];
    $base_url = ( isset($_SERVER['HTTPS']) &&
$_SERVER['HTTPS']=='on' ? 'https' :
'http'://' .  $_SERVER['HTTP_HOST'];
    $url = $base_url . $_SERVER["REQUEST_URI"];
    // $search = $_REQUEST('search');
    // $option = $_REQUEST('options');
    parse_str(parse_url($url)['query'], $params);
    $search = $params['search'];
```

```
    $option = $params['options'];
      $data = PatientModel::search($search,$option);
      $this->render('patients/patient', $data);
  }
}
```

Lớp Patient này kế thừa từ lớp Controller và đóng vai trò điều khiển các chức năng liên quan đến bệnh nhân trong ứng dụng PHP.

Phân tích chi tiết:

Hàm khởi tạo:

Tải 10 model liên quan đến bệnh nhân:

PatientModel (thông tin bệnh nhân cơ bản)

PatientSymptomModel (triệu chứng bệnh nhân)

PatientComorbidityModel (bệnh nền của bệnh nhân)

PCRTestModel (kết quả xét nghiệm PCR)

QuickTestModel (kết quả xét nghiệm nhanh)

SPO2TestModel (kết quả đo SPO2)

RespiratoryRateTestModel (kết quả đo nhịp thở)

WarningModel (cảnh báo liên quan đến bệnh nhân)

TreatmentModel (liệu pháp điều trị)

PatientHistoryLocationModel (lịch sử di chuyển của bệnh nhân)

Phương thức list1:

Lấy danh sách tất cả bệnh nhân từ model PatientModel.

Truyền danh sách bệnh nhân vào view patients/patient để hiển thị.

Phương thức test:

Lấy ID bệnh nhân từ URL đã giải mã base64.

Sử dụng các model liên quan để lấy kết quả xét nghiệm
PCR, Quick, SPO2, nhịp thở theo ID bệnh nhân.

Truyền các kết quả xét nghiệm vào view patients/test_table để hiển thị.

Phương thức detail:

Tương tự test, lấy ID bệnh nhân từ URL.

Lấy thông tin chi tiết bệnh nhân, triệu chứng, bệnh nền, các kết quả xét nghiệm, cảnh báo, liệu pháp điều trị, lịch sử di chuyển từ các model tương ứng.

Truyền tất cả dữ liệu vào view patients/detail để hiển thị thông tin tổng quan về bệnh nhân.

Phương thức search:

Lấy các tham số tìm kiếm từ URL (search, options) bằng cách phân tích URL.

Gọi phương thức search của model PatientModel với các tham số tìm kiếm để lấy kết quả.

Truyền kết quả tìm kiếm vào view patients/patient để hiển thị danh sách bệnh nhân phù hợp.

Mục đích sử dụng của lớp Patient là để quản lý thông tin bệnh nhân trong ứng dụng PHP. Cụ thể, lớp này cung cấp các chức năng sau:

Lấy danh sách tất cả bệnh nhân

Hiển thị thông tin chi tiết về một bệnh nhân

Lấy kết quả xét nghiệm của một bệnh nhân

Tìm kiếm bệnh nhân dựa trên các tiêu chí nhất định

Lớp Patient có thể được sử dụng trong các ứng dụng quản lý bệnh viện, theo dõi sức khỏe, v.v.

4.4.3.4)Form

Đoạn code trong Form.php:

```
<?php
class Form extends Controller{
    public $model;
    public function __construct(){
       $this->model('BuildingModel');
       $this->model('FloorModel');
       $this->model('RoomModel');
       $this->model('SymptomModel');
       $this->model('ComorbidityModel');
       $this->model('NurseModel');
       $this->Model('PatientModel');
       $this->model('PatientHistoryLocationModel');
       $this->model("PatientSymptomModel");
       $this->model("PatientComorbidityModel");
    }
    public function index(){
        $data = [];
        $data['nurse'] = NurseModel::getAllNurse();
        $data['building'] =
BuildingModel::getAllBuilding();
        $data['floor'] = FloorModel::getAllFloor();
        $data['room'] = RoomModel::getAllRoomAvailable();
        // print_r($data['nurse'][0]->getUniqueNumber());
```

```
        $data['symptom'] = SymptomModel::getAllSymptom();
        $data['comorbidity'] =
ComorbidityModel::getAllComorbidity();
        $this->render("Form/form", $data);
    }
    public function add(){
        $uq = PatientModel::CountRows() + 1;
        $timestamp = time();
        $currentDate = gmdate('Y-m-d', $timestamp);
        //patient
        $unique_number = 'PT'.$uq; //xu ly ma
        $id = $_REQUEST['id'];
        $phone = $_REQUEST['phone'];
        $fName = $_REQUEST['fname'];
        $lName = $_REQUEST['lname'];
        $gender = $_REQUEST['gender'];
        $birthday = $_REQUEST['birthday'];
        $address = $_REQUEST['address'];
        $ward = $_REQUEST['ward'];
        $district = $_REQUEST['district'];
        $city = $_REQUEST['city'];
        $dateAdmit = $currentDate;
        $dateDischarge = null;
        $nurse_id = $_REQUEST['nurse_id'];
        $staff_id = $_REQUEST['staff_id'];
```

```
        $patient = new PatientModel($unique_number, $id,
$phone, $fName, $lName, $gender,
$birthday, $address,$ward,$district,$city, $dateAdmit,
$dateDischarge, $nurse_id, $staff_id);

        $patient->create($patient);

               // // PHL

        $room = $_REQUEST['room'];

        $phl = new
PatientHistoryLocationModel($unique_number,$room,$curren
tDate,null);

        $phl->create($phl);

        //patient symptom

        if(isset($_REQUEST['symptom'])){

            $symptomId = $_REQUEST['symptom'];

            foreach($symptomId as $symptom){

                $sd = 'Sdescription'.$symptom;

                $od = 'OnsetDate'.$symptom;

                $td = 'TerminationDate'.$symptom;

                $description = $_REQUEST[$sd];

                $onsetDate = $_REQUEST[$od];

                $terminationDate = $_REQUEST[$td];

         $ps = new PatientSymptomModel($symptom,
        $unique_number,$description,$onsetDate,$terminati
        onDate);

                $ps->create($ps);

            }
```

```
        }


        // // //patient comorbidity
        if(isset($_REQUEST['comorbidity'])){
            $ComorbidityId = $_REQUEST['comorbidity'];
            foreach($ComorbidityId as $comorbidity){
                $sd = 'Cdescription'.$comorbidity;
                                $description =
    $_REQUEST[$sd];
                $pc=newPatientComorbidityModel( $unique_
number,$comorbidity,$de scription);
                $pc->create($pc);
            }
        }
                if(isset($_REQUEST['Sdiff'])){
                $symptomId =SymptomModel::countRows()+1;
                $name = $_REQUEST['diff1'];
                $description = $_REQUEST['Sdescription'];
                $onsetDate = $_REQUEST['OnsetDate'];
                $terminationDate =
$_REQUEST['TerminationDate'];
                $symptom = new SymptomModel($symptomId,
$name);
       $ps = new PatientSymptomModel($symptomId,
      $unique_number,$description,$onsetDate,$terminatio
      nDate);
                $symptom->create($symptom);
```

```
                $ps->create($ps);


        }


        if(isset($_REQUEST['Cdiff'])){
            $ComorbidityId =
ComorbidityModel::countRows()+1;
            $name = $_REQUEST['diff2'];
            $description = $_REQUEST['Cdescription'];
            $Comorbidity = new
ComorbidityModel($ComorbidityId, $name);
            $pc = new PatientComorbidityModel
          ($unique_number,$ComorbidityId,$description);
            $Comorbidity->create($Comorbidity);
            $pc->create($pc);


    }
        header("Location: /CSDL2/form");
}
}
```

Lớp Form này kế thừa từ lớp Controller và có chức năng chính là xử lý việc thêm mới bệnh nhân vào hệ thống.

Phân tích chi tiết:

Hàm khởi tạo: Tương tự lớp Patient, nó tải các model liên quan đến bệnh nhân, nhân viên y tế, phòng bệnh, triệu chứng, bệnh nền, v.v.

Phương thức index:

Lấy danh sách các nhân viên y tế, tòa nhà, tầng, phòng bệnh khả dụng.

Lấy danh sách các triệu chứng và bệnh nền.

Truyền dữ liệu vào view Form/form để hiển thị form nhập thông tin bệnh nhân mới.

Phương thức add:

Xử lý logic thêm mới bệnh nhân:

Lấy thông tin bệnh nhân từ request ($_REQUEST).

Tạo một object PatientModel mới với các thông tin lấy được.

Gọi phương thức create của model để lưu thông tin bệnh nhân vào database.

Xử lý tương tự cho PatientHistoryLocationModel (lịch sử di chuyển bệnh nhân).

Kiểm tra và xử lý các trường hợp thêm mới triệu chứng hoặc bệnh nền mới:

Nếu có danh sách triệu chứng mới (Sdiff), tạo
object SymptomModel và PatientSymptomModel mới để thêm vào database.

Tương tự với bệnh nền mới (Cdiff).

Cuối cùng, chuyển hướng người dùng về trang form.

Mục đích sử dụng của lớp Form là để xử lý việc thêm mới bệnh nhân vào hệ thống. Cụ thể, lớp này cung cấp các chức năng sau:

Hiển thị form nhập thông tin bệnh nhân mới.

Xử lý logic thêm mới bệnh nhân, bao gồm:

Lấy thông tin bệnh nhân từ request ($_REQUEST).

Tạo một object PatientModel mới với các thông tin lấy được.

Gọi phương thức create của model để lưu thông tin bệnh nhân vào database.

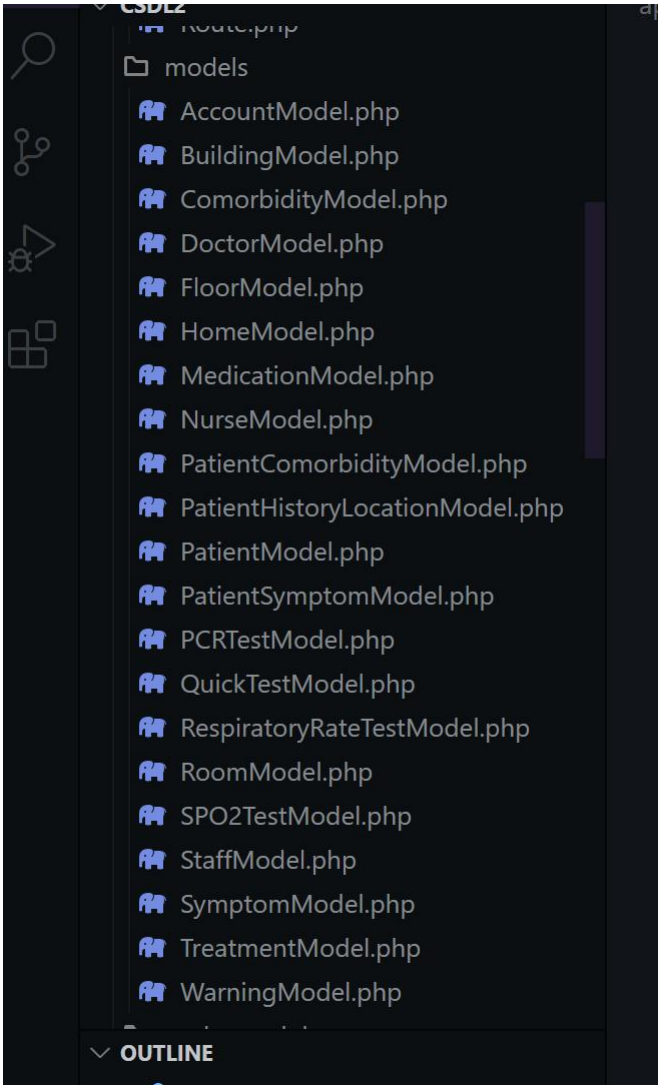
Xử lý tương tự cho PatientHistoryLocationModel (lịch sử di chuyển bệnh nhân).

Kiểm tra và xử lý các trường hợp thêm mới triệu chứng hoặc bệnh nền mới.

4.4.4) Models

Trong folder models chứa những file model giúp việc lưu trữ và truy xuất dữ liệu từ cơ sở dữ liệu.

Trong folder models có:



Các file trong folder model

Các file model đại điện cho các bảng trong cơ sở dữ liệu:

- AccountModel
- BuildingModel

- ComorbidityModel
- DoctorModel
- FloorModel
- MedicationModel
- NurseModel
- PatientComorbidityModel
- PatientHistoryLocationModel
- PatientSymptomModel
- PCRTestModel
- QuickTestModel
- RespiratoryRateTestModel:
- RoomModel
- SPO2TestModel
- StaffModel
- SymptomModel
- TreatmentModel
- WarningModel
- PatientModel

Phân tích cụ thể một model:

File PatientModel.php

File này định nghĩa lớp PatientModel, đại diện cho một bệnh nhân trong hệ thống bệnh viện. Lớp này có các thuộc tính sau:

uniqueNumber: Số hiệu bệnh nhân

id: ID bệnh nhân

phone: Số điện thoại của bệnh nhân

fname: Tên của bệnh nhân

lname: Họ của bệnh nhân

gender: Giới tính của bệnh nhân

birthday: Ngày sinh của bệnh nhân

address: Địa chỉ của bệnh nhân

ward: Khu vực bệnh nhân đang nằm

district: Quận/huyện của bệnh nhân

city: Thành phố của bệnh nhân

dateAdmit: Ngày nhập viện của bệnh nhân

dateDischarge: Ngày xuất viện của bệnh nhân

nurse_id: ID của y tá phụ trách bệnh nhân

staff_id: ID của nhân viên y tế phụ trách bệnh nhân

Lớp này cũng có các phương thức sau:

__construct(): Hàm khởi tạo lớp

getUniqueNumber(): Phương thức trả về số hiệu bệnh nhân

getId(): Phương thức trả về ID bệnh nhân

getPhone(): Phương thức trả về số điện thoại của bệnh nhân

getFirstName(): Phương thức trả về tên của bệnh nhân

getLastName(): Phương thức trả về họ của bệnh nhân

getGender(): Phương thức trả về giới tính của bệnh nhân

getBirthday(): Phương thức trả về ngày sinh của bệnh nhân

getAddress(): Phương thức trả về địa chỉ của bệnh nhân

getWard(): Phương thức trả về khu vực bệnh nhân đang nằm

getDistrict(): Phương thức trả về quận/huyện của bệnh nhân

getCity(): Phương thức trả về thành phố của bệnh nhân

getDateAdmit(): Phương thức trả về ngày nhập viện của bệnh nhân

getDateDischarge(): Phương thức trả về ngày xuất viện của bệnh nhân

getNurseId(): Phương thức trả về ID của y tá phụ trách bệnh nhân

getStaffId(): Phương thức trả về ID của nhân viên y tế phụ trách bệnh nhân

create(): Phương thức tạo mới một bệnh nhân

update(): Phương thức cập nhật thông tin của một bệnh nhân

delete(): Phương thức xóa một bệnh nhân

getPatientById(): Phương thức lấy thông tin của một bệnh nhân theo ID

getAllPatient(): Phương thức lấy thông tin của tất cả bệnh nhân

getAllPatientCount(): Phương thức lấy tổng số bệnh nhân

CountRows(): Phương thức đếm số hàng trong bảng bệnh nhân

PND(): Phương thức đếm số bệnh nhân đang nằm viện

PD(): Phương thức đếm số bệnh nhân đã xuất viện

search(): Phương thức tìm kiếm bệnh nhân theo một tiêu chí

Các file khác

Các file khác trong thư mục models đại diện cho các mô hình khác nhau trong hệ thống bệnh viện, chẳng hạn như:

AccountModel: Mô hình tài khoản người dùng

BuildingModel: Mô hình tòa nhà

ComorbidityModel: Mô hình bệnh lý đi kèm

DoctorModel: Mô hình bác sĩ

FloorModel: Mô hình tầng

HomeModel: Mô hình nhà

MedicationModel: Mô hình thuốc men

NurseModel: Mô hình y tá

PatientComorbidityModel: Mô hình bệnh lý đi kèm của bệnh nhân

PatientHistoryLocationModel: Mô hình lịch sử vị trí của bệnh nhân

PatientSymptomModel: Mô hình triệu chứng của bệnh nhân

PCRTestModel: Mô hình xét nghiệm PCR

QuickTestModel: Mô hình xét nghiệm nhanh

RespiratoryRateTestModel: Mô hình xét nghiệm nhịp thở

RoomModel: Mô hình phòng bệnh

SPO2TestModel: Mô hình xét nghiệm oxy trong máu

StaffModel: Mô hình nhân viên y tế

SymptomModel: Mô hình triệu chứng

TreatmentModel: Mô hình điều trị

WarningModel: Mô hình cảnh báo

VI/ Kết luận

Trong bài báo cáo này, chúng em đã trình bày việc xây dựng cơ sở dữ liệu cho trại cách ly. Cơ sở dữ liệu này được thiết kế để lưu trữ thông tin của các bệnh nhân, nhân viên y tế và các hoạt động trong trại.

Cơ sở dữ liệu được xây dựng dựa trên mô hình MVC, với các thành phần chính là:

Model: Lưu trữ dữ liệu của các đối tượng trong trại.

View: Hiển thị dữ liệu cho người dùng.

Controller: Điều khiển việc truy cập và xử lý dữ liệu.

Cơ sở dữ liệu được thiết kế theo hướng mở, có thể mở rộng để đáp ứng nhu cầu của các trại cách ly khác nhau.

Dưới đây là một số ưu điểm của cơ sở dữ liệu này:

Tính nhất quán: Dữ liệu được lưu trữ tập trung trong một cơ sở dữ liệu, giúp đảm bảo tính nhất quán của dữ liệu.

Tính bảo mật: Dữ liệu được bảo vệ bằng các biện pháp bảo mật, giúp ngăn chặn truy cập trái phép.

Tính khả dụng cao: Cơ sở dữ liệu được thiết kế để có thể hoạt động ổn định và không bị gián đoạn.

Tuy nhiên, cơ sở dữ liệu này cũng có một số hạn chế cần được khắc phục trong tương lai, chẳng hạn như:

Tính hiệu quả: Cơ sở dữ liệu có thể được tối ưu hóa để cải thiện hiệu suất truy vấn dữ liệu.

Tính mở rộng: Cơ sở dữ liệu có thể được mở rộng để đáp ứng nhu cầu của các trại cách ly có quy mô lớn hơn.

Nhìn chung, cơ sở dữ liệu này là một giải pháp hiệu quả để quản lý thông tin trong trại cách ly. Cơ sở dữ liệu này có thể được sử dụng trong các trại cách ly khác nhau, giúp cải thiện hiệu quả quản lý và vận hành trại.

VII)/ Tài liệu tham khảo

[1]:Mô Hình Quan Hệ Thực Thể (Entity Relationship Model)

[2]:Chuong1_2_ER_MohinhQuanhe(20189101629).pdf

[3]:Những Điều Cần Biết Về Index Trong Database

[4]:Tất tần tật về mô hình MVC

[5]:Tất tần tật những điều bạn cần biết về PHP

[6]:Localhost là gì?

[7]:Radmin VPN

[8]:SQL Server và SSMS

[9]:Hướng-dẫn-cài-đặt-và-sử-dụng-VisualSVN-Server-và-TortoiseSVN.pdf

[10]:Visual Studio Code là gì?

[11]:Xampp là gì?