1. **Explain the MVC pattern. Describe the workflow when a user enters an URL in the browser in Rails (as detailed as possible).**

- Mô hình MVC (model, view, controller).
- Chức năng của từng phần:
  Controller: Đây là nơi nhận, và nơi xử lý chính các yêu cầu khi người dùng thực hiện 1 request nào đó, thông qua các giao thức HTTP: Get, post, delete…
  Model: Là nơi chịu trách nhiệm làm việc trực tiếp với database. Tiếp nhận yêu cầu từ controller và thực hiện lấy dữ liệu, xử lý phù hợp và trả lại cho controller. Sau đó Controller gửi data từ model đó sang view.
  View: Là nơi hiển thị chính ra giao diện, tiếp nhận data đc xử lý từ controller và hiển thị cho người dùng ở browser.

Mô tả luồng hoạt động khi mà người dùng nhập một URL trên trình duyệt rails.

Khi người dùng nhập một URL vào trình duyệt, yêu cầu đó sẽ được gửi đến server.

Đầu tiên phải định tuyến, url khớp với định tuyến trong file router.rb. file. Định tuyến đến action cụ thể trong controller.

Khi router được khớp, action trong controller được gọi. Controller chịu trách nhiệm xử lý yêu cầu và chuẩn bị dữ liệu để gửi đến view.

Trong action của controller, controller có thể tương tác với model để lấy data từ DB.

Sau khi xử lý data, controller được hiển thị view tương ứng. View được xác định dựa trên name convenience.

Cuối cùng, Nội dung HTML cuối cùng được tạo bởi view . Browser nhận phản hồi này và hiển thị trang đã được hiển thị cho người dùng.

2. **What is the difference between User.all.first(2) and User.all.limit(2) ?**
   Đây là 2 thực hiện việc lấy dữ liệu tương ứng từ database. Tuy nhiên,
- **User.all.first(2):** Lấy 2 bản ghi đầu tiên từ bảng users và trả về một mảng chứa chúng.
- **User.all.limit(2):** Lấy 2 bản ghi ngẫu nhiên từ bảng users và trả về một mảng chứa chúng. limit không quan tâm đến thứ tự.

3. **Sự khác nhau của delete và destroy**
   delete: thực hiện thao tác xóa dữ liệu nhưng mà nó sẽ k xóa được các bản ghi con, và nó k thực hiện đc thao tác xóa chính nó khi mà có bảng khác tham chiếu đến nó. Phải xóa hết bản con mới xóa đc bản cha đó.
   destroy thì nó xóa được table cha và xóa luôn đc bảng ghi con khi chúng ta thực hiện dependent: :destroy ngay trên model cha.

4. **How to check if a record is valid or not? Show all errors or that record if it invalid.**

   Theo em tìm hiểu thì mình có thể sử dụng valid? để kiểm tra, nếu nó là true thì hợp lệ. Ngược lại false thì không hợp lệ.

   Và muốn in ra tất cả các lỗi trong bản ghi nếu không hợp lệ thì mình có thể sử dụng errors.full_messages

5. What is the difference between Validation (in Active Record) and Constraint in database?

Theo em hiểu:

- Validation: Là việc thực hiện kiểm tra dữ liệu của mình có đúng, với phù hợp với yêu cầu trước khi lưu nó vào database. Thực hiện trong **model**.
- Constraint: Nó như là 1 ràng buộc, mục đích duy trì tính toàn vẹn, ví dụ ràng buộc các quy tắc như khóa chính, khóa ngoại. Được thực hiện ở trong **database**.

6. How do you understand the **strong parameters** in Rails?

Theo như em hiểu nó được sử dụng để tránh sự tấn công bảo mật, khi mà người dùng nhập dữ liệu vào

em ví dụ việc truyền tham số từ bên ngoài vào:

Thì sử dụng **strong parameters** ở bên dưới

```
  private
  def comment_params
    params.require(:comment).permit(:commenter, :body, :status)
  end
end
```

7. What is the difference between has_many :through and has_and_belongs_to_many?

Theo em hiểu:

2 cái này dùng để thực hiện việc liên quan mối quan hệ nhiều nhiều (many-to-many): Nhưng nó sẽ khác nhau như sau:

- **has_many :through:** Thực hiện liên kết gián tiếp, Tạo ra 1 table trung gian chứa khóa chính của các table trong liên kết đó. Bên cạnh đó cx tạo ra luôn 1 model đó. Và việc sử dụng cách này chúng ta có thể thực hiện thao tác trên bảng trung gian đó.
- **has_and_belongs_to_many:** Thực hiện liên kết trực tiếp. cũng tạo ra 1 table trung gian, nhưng chỉ chứa duy nhất khóa chính của các bảng thôi chứ k có các field khác. Bên cạnh đó theo như.

8. Should we remove or edit old migration files? Why?

Migration nó là nơi giúp mình có thể thực hiện quản lý cấu trúc cơ sở dữ liệu của mình. Mình có thể sử dụng migration để thực hiện thao tác liên quan đến các table, create table, add column, remove…Nó cx có thể được quản lý các phiên bản của csdl.

Mỗi migration đc hiểu như là 1 version của csdl. Giúp mapping model với database.

Vậy có nên xóa hay không ?

Khi mình xóa 1 migration thì khi chạy db:migrate thì nó sẽ hiển thị không tìm thấy file đó. Và theo em thì tùy thuộc vào yêu cầu của ứng dụng. Theo như em nói ở trên thì nếu như mình xóa 1 migration old thì mình phải cân nhắc xem có ảnh hưởng gì đến ứng dụng đang chạy không, hoặc muốn duy trì lịch sử csdl. Nếu không ảnh hưởng gì hay là k cần thiết nữa thì mình có thể xóa để cấu trúc gọn gàng và hiệu suất tốt hơn

9. Re-write these callbacks in the correct order when we save a new record:

a. after_save
b. before_save
c. before_commit
d. after_create
e. Before_validation

Trước tiên phải hiểu callback trong rails nó là 1 ActiveRecord. Thì nó là cơ chế xảy ra trong **model**, callback can thiệp vào 1 sự kiện trong quá trình diễn ra vòng đời (ở đây các sự kiện đó có thể là create, update, delete). Nó thực thi 1 login trước hoặc sau khi đối tượng đó thay đổi và lưu vào csdl.

Thì khi tạo mới một record. Theo thứ tự với yêu cầu trên thì THEO EM sẽ như sau:

Before_validation -> before_save → after_create → before_commit → after_save

11. With User and Post in previous question, to get an array of **post id** and **author name**, I wrote like this:

```
Post.all.map do |post|
  [post.id, post.author.name]
end
```

a. **How many times does our code hit the database?**
   Với đoạn code trên thì sẽ lấy tất cả bài post rồi trả về mảng theo post id và name của author.
   Mã truy cập vào cơ sở dữ liệu N+1 lần, trong đó N là số lượng bài post. Đối với mỗi bài post, nó thực hiện truy vấn để tìm name author cho bài post đó. Vì vậy, đối với N bài post, sẽ có N truy vấn để tìm nạp bài post và thêm N truy vấn để tìm nạp name author, dẫn đến tổng số 2N truy vấn.
b. **Do you have any ideas to optimize it?**
   Để tối ưu được điều đó thì có thể sử dụng **include(:author)** rails sẽ tìm tất cả bài post và data của tác giả được liên kết. Thay vì cách trên. Điều này giảm số lượng truy vấn và cải thiện được hiệu suất.

```
posts_with_authors = Post.includes(:author).map do |post|
  [post.id, post.author.name]
end
```