1. Tìm hiểu về Backend Development

- Back-end có nghĩa là làm việc trên phần mềm phía máy chủ, tập trung vào mọi thứ bạn không thể thấy trên trang web. Các nhà phát triển back-end đảm bảo trang web hoạt động chính xác, tập trung vào cơ sở dữ liệu, logic back-end, giao diện lập trình ứng dụng (API), kiến trúc và máy chủ.
- Trong một nhóm, các nhà phát triển back-end cộng tác với các nhà phát triển front-end, quản lý sản phẩm, kiến trúc sư chính và người kiểm tra trang web để xây dựng cấu trúc của trang web hoặc ứng dụng di động. Các nhà phát triển back-end phải quen thuộc với nhiều loại công cụ và khuôn khổ, bao gồm các ngôn ngữ như Python, Java và Ruby. Họ đảm bảo back-end hoạt động nhanh chóng và phản hồi các yêu cầu của người dùng front-end.

2. Cài đặt Git, tìm hiểu Git(Branch, Rebase, Merge, Push, Commit, Add, Checkout)

- Branch :

  Là một nhánh phụ riêng biệt của kho lưu trữ chính.

  Xem các branch hiện có: git branch

  Tạo một branch mới:git branch <tên-branch>

  Tạo và chuyển sang branch mới:git checkout -b <tên-branch>

  Chuyển sang một branch đã có:git checkout <tên-branch>

  Xóa một branch:git branch -d <tên-branch>

  Liệt kê các remote branch::git branch -r

  Đẩy một branch mới lên remote:git push origin <tên-branch>

  Xóa một branch trên remote:git push origin --delete <tên-branch>

- Rebase :

  Git Rebase là một thao tác mạnh mẽ trong Git dùng để thay đổi lịch sử commit. Thay vì tạo một merge commit như khi sử dụng git merge, git rebase sẽ tái áp dụng các commit của bạn lên một branch khác, giúp giữ lịch sử commit sạch sẽ và tuyến tính.

- Merge:

  Dùng để kết hợp các thay đổi từ một branch này vào branch khác. Khi bạn muốn gộp các commit từ một branch

- Push:

Sử dụng để đẩy các thay đổi (commits) từ local repository (repo cục bộ) lên remote repositoryCommit

- Add:

  Sử dụng để thêm các thay đổi (như các file mới hoặc các thay đổi trong file đã có) vào staging area (vùng chuẩn bị) trước khi thực hiện commit.

- Checkout

  Sử dụng để chuyển đổi giữa các branch hoặc phục hồi các file trong thư mục làm việc của bạn.

3. Cài đặt Node.js, TypeScript, MongoDB, Redis
4. Setup một project NestJS từ đầu
5. Cấu trúc thư mục trong NestJS
   - **src/**: Chứa toàn bộ source code của ứng dụng.
   - **modules/**: Mỗi module sẽ có controller, service, entity, dto riêng.
   - **common/**: Chứa các thành phần có thể tái sử dụng trong toàn bộ ứng dụng.
   - **config/**: Chứa các file cấu hình như database, JWT, môi trường .env.
   - **database/**: Quản lý database, migrations, seed data.
   - **test/**: Chứa các file test (unit test, e2e test).
6. Sử dụng eslint & prettier để format code
7. Tìm hiểu package.json và dependency management

   Package.json là file quan trọng trong một dự án Node.js/NestJS, dùng để:

- Quản lý thông tin dự án (tên, phiên bản, mô tả, tác giả).
- Liệt kê các package (dependencies & devDependencies).
- Chứa scripts để chạy dự án.
- Cấu hình cho một số công cụ
- Dependencies là các package cần thiết khi chạy ứng dụng.
  DevDependencies là các package chỉ dùng trong phát triển.

8. Backend Development là gì và tại sao nó quan trọng trong phát triển ứng dụng?

- Là quá trình xây dựng và duy trì các thành phần phía máy chủ (server-side) của một ứng dụng hoặc trang web. Phần backend là tất cả những gì người dùng không thấy, bao gồm việc xử lý logic nghiệp vụ, quản lý cơ sở dữ liệu, xác thực người dùng, xử lý API, và quản lý các dịch vụ khác. Mặc dù người dùng chỉ tương tác với frontend (giao diện người dùng), nhưng backend đóng vai trò quan trọng trong việc đảm bảo ứng dụng hoạt động đúng và hiệu quả.

9. Các thành phần chính của một hệ thống backend?
- Máy chủ (Server): Là nơi chứa mã nguồn của ứng dụng và dữ liệu, nhận yêu cầu từ frontend và trả kết quả.
- Cơ sở dữ liệu (Database): Lưu trữ dữ liệu của ứng dụng như thông tin người dùng, đơn hàng, bài viết, và các dữ liệu khác.
- API (Application Programming Interface): Giúp frontend và backend giao tiếp với nhau. Các API cung cấp các endpoint mà frontend có thể gọi để lấy dữ liệu từ backend.
- Xử lý logic nghiệp vụ (Business Logic): Là phần xử lý các yêu cầu và quyết định cách ứng dụng sẽ hoạt động, như tính toán giá trị, xác thực dữ liệu, hay quản lý các trạng thái của ứng dụng.
- Bảo mật (Security): Backend đảm bảo tính bảo mật cho dữ liệu người dùng, bao gồm các phương pháp mã hóa, xác thực, phân quyền người dùng, và phòng chống các tấn công như SQL injection, XSS, CSRF.

10. Sự khác biệt giữa backend và frontend?
- Frontend: Là phần giao diện người dùng (UI) của ứng dụng hoặc trang web, mà người dùng có thể thấy và tương tác. Frontend bao gồm tất cả các yếu tố mà người dùng nhìn thấy trên màn hình như các nút bấm, biểu mẫu, hình ảnh, màu sắc, văn bản, v.v.
- Backend: Là phần máy chủ (server-side) của ứng dụng, nơi xử lý các yêu cầu từ người dùng, quản lý cơ sở dữ liệu, thực hiện logic nghiệp vụ và gửi dữ liệu trở lại cho frontend. Backend là những gì người dùng không nhìn thấy nhưng rất quan trọng để đảm bảo ứng dụng hoạt động hiệu quả.

11. TypeScript là gì?
- TypeScript là một ngôn ngữ lập trình mã nguồn mở được phát triển bởi Microsoft, là một sự mở rộng của JavaScript. TypeScript thêm các tính năng mạnh mẽ, đặc biệt là tính năng kiểm tra kiểu dữ liệu (type checking), giúp việc phát triển các ứng dụng phức tạp trở nên dễ dàng hơn và ít lỗi hơn.

12. Redis là gì ?
- là một cơ sở dữ liệu lưu trữ dữ liệu theo dạng key-value (khóa-giá trị) rất nhanh và mạnh mẽ. Redis là in-memory database, có nghĩa là dữ liệu được lưu trữ trực tiếp trong bộ nhớ (RAM) thay vì ổ đĩa, giúp truy xuất và xử lý dữ liệu cực kỳ nhanh chóng.

13. NestJS là gì?

    NestJS là một framework Node.js được xây dựng để phát triển các ứng dụng server-side (phía máy chủ) hiệu quả và dễ bảo trì. Nó được phát triển bằng TypeScript và sử dụng các nguyên lý lập trình hướng đối tượng, lập trình chức năng và lập trình phản ứng để giúp tạo ra các ứng dụng backend mạnh mẽ.

14. ESLint là gì?

- Là một công cụ linter (kiểm tra mã nguồn) cho JavaScript và TypeScript, giúp phát hiện và sửa các lỗi trong mã nguồn, tuân thủ các quy chuẩn lập trình, và cải thiện chất lượng mã. Linter là công cụ giúp kiểm tra mã nguồn theo các quy tắc xác định, giúp phát hiện lỗi, cảnh báo, và cải tiến mã nguồn trong quá trình phát triển.

15. Prettier là gì?

- Là một công cụ formatting (định dạng mã nguồn) tự động giúp bạn đảm bảo rằng mã nguồn của mình tuân thủ một phong cách nhất quán về cách thức viết mã, giúp mã dễ đọc và dễ bảo trì hơn.

16. Sự khác biệt giữa git merge và git rebase?

- Git Merge:
- Mục đích: Dùng để kết hợp các nhánh lại với nhau mà không thay đổi lịch sử của các commit đã có.
- Cách hoạt động: Khi thực hiện git merge, Git sẽ tạo một commit mới (gọi là merge commit) để kết hợp hai nhánh. Merge giữ nguyên lịch sử commit của các nhánh và thêm một commit hợp nhất vào cây commit.
- Git Rebase:
- Mục đích: Dùng để di chuyển hoặc "replay" các commit từ nhánh này lên trên nhánh khác, giúp làm sạch lịch sử commit.
- Cách hoạt động: Khi thực hiện git rebase, Git sẽ áp dụng lại các commit từ nhánh hiện tại lên trên các commit của nhánh đích, tạo ra lịch sử commit mới và làm cho nó "gọn gàng" hơn. Rebase thay đổi lịch sử commit vì nó tạo ra các commit mới cho các thay đổi.

17. Sự khác nhau giữa git reset, git checkout và git revert?

Lệnh	Mục đích	Cách hoạt động	Lịch sử commit	Thư mục làm việc / Chỉ mục
git reset	Quay lại một commit trước đó, thay đổi lịch sử commit	Di chuyển `HEAD`, có thể thay đổi thư mục làm việc hoặc chỉ mục	Thay đổi	Có thể thay đổi hoặc xóa bỏ
git checkout	Chuyển nhánh hoặc khôi phục file	Chuyển nhánh hoặc khôi phục file về trạng thái của commit hiện tại	Không thay đổi	Có thể thay đổi nếu khôi phục file
git revert	Đảo ngược thay đổi của commit mà không thay đổi lịch sử commit	Tạo commit mới đảo ngược thay đổi của commit đã chỉ định	Không thay đổi	Không thay đổi