### ****1. Chuẩn bị công nghệ & công cụ Backend (.NET 6, Entity Framework Core, SQL Server)****

* **Cài đặt môi trường**

.NET 6 SDK

SQL Server + SQL Server Management Studio (SSMS)

Visual Studio hoặc VS Code (có cài đặt .NET & các extension hỗ trợ ASP.NET Core)

* **Các thư viện cần thiết**

**Entity Framework Core** (EF Core) → để làm việc với cơ sở dữ liệu

**Identity (hoặc JWT)** → để xác thực và phân quyền

**Automapper** → để map dữ liệu giữa DTOs và Models

**Serilog hoặc NLog** → để logging

**Swagger** → để test API

* **Cấu trúc database**

**Bảng Users** → Lưu thông tin admin

**Bảng Categories** → Lưu danh mục bài viết

**Bảng Posts** → Lưu bài viết

**Bảng Comments (tuỳ chọn)** → Lưu bình luận nếu cần

#### ****Frontend (HTML, CSS, JavaScript - Fetch API để gọi API từ Backend)****

**Cài đặt môi trường**

Cần một thư viện UI như Bootstrap hoặc Tailwind CSS để thiết kế giao diện

Cân nhắc dùng một framework JS (React/Vue) nếu muốn xây dựng UI mạnh mẽ hơn

* **Thư viện hỗ trợ**

**CKEditor** → để viết bài với nội dung phong phú

**Axios hoặc Fetch API** → để gọi API

### ****2. Các bước triển khai dự án****

#### ****Bước 1:** Thiết kế cơ sở dữ liệu**

Xây dựng sơ đồ database với bảng Users, Categories, Posts

Viết các migration với **Entity Framework Core**

#### ****Bước 2: Xây dựng API với ASP.NET Core****

Tạo dự án ASP.NET Core Web API

Viết các API:

**Authentication API** (Đăng nhập, đăng xuất, quản lý session/token)

**Category API** (Lấy danh sách, tìm kiếm, phân trang, thêm, sửa, xóa)

**Post API** (Tương tự Category nhưng có thêm tính năng upload ảnh)

#### ****Bước 3: Xây dựng giao diện Frontend****

Trang chính: Hiển thị danh sách bài viết mới nhất, phân trang

Trang chi tiết bài viết

Trang quản trị: CRUD Categories, CRUD Posts

Form đăng nhập

#### ****Bước 4: Tích hợp Frontend với Backend****

Dùng Fetch API/Axios để gọi API từ frontend

Hiển thị dữ liệu nhận được từ API

Xử lý đăng nhập, lưu token, phân quyền

#### ****Bước 5: Kiểm thử và triển khai****

Test API với Postman

Test giao diện trên nhiều trình duyệt

Deploy lên Azure hoặc VPS

### ****3. Các tính năng có thể mở rộng****

Hỗ trợ bình luận (comments) cho bài viết

Tạo trang dashboard thống kê số lượng bài viết, danh mục

Thêm chức năng gửi email thông báo khi có bài viết mới

Thêm chức năng đăng nhập bằng liên kết mạng xã hội (Facebook, Google, …)

Thêm tính năng Like/Dislike bài viết

**Process:**

**Day 1:** Vẽ sơ đồ cơ sở dữ liệu

**Day 2:** Tạo Model, connect Database, Add Migration

**Day 3:** Tạo Controller Category

**Day 4:** Tạo Controller User, Post, Comment

**Day 5:** Thêm Upload ảnh ở Post

**Day 6:** Tạo Contoller Authentication, Authorization với JWT

**Day 7:** Thêm Upload ảnh ở User

**Day 8:**

+ Cập nhập API login, register, logout (Thực tế không cần API logout vì chỉ cần xóa token ở LocalStorage phía frontend).

+ Cập nhập thêm chức năng xác thực Email khi đăng ký tài khoản:

Luồng hoạt động cơ bản:

1. Người dùng đăng ký tài khoản
2. Tạo ra một token xác nhận email (Dùng Guid để sinh token ngẫu nhiên)
3. Gửi email chứa link xác nhận đến địa chỉ email người dùng
4. Khi người dùng nhấp vào link, backend xử lý xác thực:

+ Kiểm tra token hợp lệ

+ Nếu hợp lệ: đánh dấu email đã xác thực (Update cột IsEmailConfirm)

\* Cách làm: Tạo EmailSerivce để gửi email (sử dụng smtp, mật khẩu của email host sử dụng App Password), Thêm API xác nhận email (gửi kèm token để so sánh với token lưu ở database).

**Day 9:** Sửa lại phần đăng nhập thì cần kiểm tra xem email đã xác nhận chưa, tạo project Frotend với Angular 16.

**Day 10:** Xây dựng các template cho giao diện, lên kế hoạch xây dựng frontend.

<https://modernize-angular-main.netlify.app/dashboards/dashboard1>

**Day 11:** Thêm header cho trang admin

**Day 12:** Thêm sidebar cho trang admin (Tìm hiểu router trong Angular)

**Day 13:** Thêm danh mục user trang admin (Cần sửa thêm các phần hiệu ứng user, admin, true false). Đã dùng API để nhận data từ Serve.

**Cấu trúc của JWT**



