### ****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, CRUD Users

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). Fetch data Serve qua API.

**Day 14:** Sửa lại giao diện danh mục User, thêm phân trang phía Frontend (Sử dụng nz-pagination).

**Day 15:** Thiết kế giao diện login. Đăng nhập thành công qua API login và lưu token xuống localStorage (Review lại code)

**Day 16:**

* Thêm chức năng logout (Xóa token và redirect to Login page).
* Thêm Guard để bảo vệ Route (Kiểm tra có token thì chuyển đến route, không thì về trang admin).

**Day 17:** Thiết kế Modal thêm user (Chưa hoàn thiện, cần thêm input ảnh)

**Day 18:**

+ Hoàn thành Modal add user, đã input được ảnh. Hiện tại đang thiếu các text để show bắt lỗi validation và Toast thông báo khi thêm thành công hoặc error.

+ Đã add thành công User xuống database.

**Day 19:**

+ Thêm phần show text validation bắt input required. Sử dụng Toastr để thông báo thành công và lỗi.

+ Cần sửa đổi các ID trong Database thành dạng Guid và sửa định dạng trả về lỗi cần trong một kiểu cố định để frontend get ra thông báo.

**Day 20:**

+ Đã sửa các yêu cầu ở day 19.

+ Thêm chức năng sửa, xóa User. Cần update modal show để xác nhận xóa thay vì dùng mặc định.

**Day 21:**

+ Update User page hoàn hiện.

+ Hoàn thành CRUD Category page. Cần sử đổi backend nhận vào DTO và bắt lỗi trùng lặp ở backend. Phần table cần độc đáo hơn nếu có thể.

**Day 22:** Sửa Category Controller để nhận DTO

**Day 23:**

+ Design PostPage, CreatePost component. Mới chỉ design cơ bản và get thông tin, cần hoàn thành chức năng sử dụng CKEditor để đăng bài.

**Day 24:**

+ Đã sử dụng hoàn thiện CKEditor để up ảnh ở bài viết (Lưu xuống serve và gọi ngay lên để hiển thị). Giao diện ok

+ Thêm thành công bài đăng xuống serve (Ở mức test và cần hoàn thiện và bắt lỗi thêm).

**Day 25:**

+ Bắt lỗi form nhập thiếu ở bài đăng

+ Thiết kế và styling nội dung bài đăng

**Day 26:**

+ Edit, Delete post

**Day 27:** + Thiết kế header blog (Cần tích hợp chức năng: Khi chưa đăng nhập thì hiện button đăng nhập, khi đăng nhập rồi thì hiện ảnh người dùng)

**Next tasks:**

+ Style thêm phần likes, comments ở view post + thiết kế backend để tích hợp.

**Cấu trúc của JWT**



