1.Tạo Class AppUserModel (tên gì cũng đc ) trong Model Class đó kế thừa IdentityUser

+nhớ mapping xuống Database ko cần public DbSet<AppUserModel > AppUsers{ get; set; } và thực thi câu lệnh Add-Migration "Tên" → lưu thay đổi giống commit. Update-Database → áp dụng commit đó vào database.

A screenshot of a computer

AI-generated content may be incorrect.

2. trong Class DbContext

+ cài Package



+ A screenshot of a computer program

AI-generated content may be incorrect.

+ trong Class Program .cs

A screen shot of a computer program

AI-generated content may be incorrect.

// 🔹 Đăng ký Identity và liên kết với DbContext

builder.Services.AddIdentity<AppUserModel, IdentityRole>()

.AddEntityFrameworkStores<DataContext>()

.AddDefaultTokenProviders();

builder.Services.Configure<IdentityOptions>(options =>

{

// Password settings. // cấu hình Password

options.Password.RequireDigit = true;

options.Password.RequireLowercase = true;

options.Password.RequireNonAlphanumeric = false;

options.Password.RequireUppercase = false;

options.Password.RequiredLength = 6;

// User settings.

options.User.RequireUniqueEmail = true ;

});

Note : cấu hình UserName mặc định của Asp.net là duy nhất trên toàn hệ thống còn Email thì programmer có thể tự setting

Tạo Class UserModel trong Models

Không cần phải mapping xuống database cũng không cần public DbSet<UserModel> UserModels { get; set; } mục đích chính là để gửi các trường dữ liệu khi người dùng Register Account từ form lên Controller /action

A screenshot of a computer program

AI-generated content may be incorrect.

\* Lưu dữ liệu ngưởi dùng (UserName , Password , Email ) nhập từ form xuống database thông qua Identity

🡺 dùng để Đăng kí tài khoản cho User

A black screen with writing on it

AI-generated content may be incorrect.

Tạo Class LoginModel trong Models

Không cần phải mapping xuống database cũng không cần public DbSet<LoginModel> LoginModels { get; set; } mục đích chính là để gửi các trường dữ liệu khi người dùng Login Account từ form lên Controller /action

* Controller /Action của trang Login

A computer screen with red lines

AI-generated content may be incorrect.

Note : SignInResult result = await \_signInManager.PasswordSignInAsync(

loginVM.UserName, *// 1. Tên đăng nhập*

loginVM.Password, *// 2. Mật khẩu*

false, *// 3. isPersistent - Có ghi nhớ dữ liệu user login vào cookie hay không*

*nếu trả về true dữ liệu sẽ được lưu vào cookie*

false *// 4. lockoutOnFailure - Có khóa tài khoản khi nhập sai?*

);

await \_signInManager.PasswordSignInAsync **= kiểm tra tài khoản + tạo cookie + đánh dấu là đã login.**  
Không cần bạn cấu hình cookie thủ công, trừ khi bạn muốn thay đổi cách lưu (thời gian sống, bảo mật, persistent...).

**Khi bạn login một user mới**

Giả sử bạn đang đăng nhập bằng **User A**,  
sau đó bạn login lại bằng **User B**, ví dụ:

await \_signInManager.PasswordSignInAsync("UserB", "password", true, false);

➡️ Lúc này ASP.NET sẽ **tự động xóa cookie cũ của User A**  
và **tạo một cookie mới** cho **User B** lúc này Browser chỉ có cookie của User B

* Lấy thông tin dữ liệu UserName , Email , Role trong Cookie từ phía Server

var result = User.Identity?.IsAuthenticated ?? false;

var userName = User.Identity .Name ;

var email = User.FindFirstValue(ClaimTypes.Email)

var role = User.FindFirstValue(ClaimTypes.Role)

**💡 Luồng hoạt động của Cookie xác thực (Authentication Cookie)**

**🧩 1. Người dùng đăng nhập**

* Người dùng nhập **username + password** → gửi lên **Server**.
* Server gọi:
* await \_signInManager.PasswordSignInAsync(username, password, isPersistent, false);
* Nếu thông tin đúng:
  + Server **tạo ra một cookie xác thực** (Authentication Cookie).
  + Cookie này chứa dữ liệu mã hóa về user (ID, Name, Roles, thời gian sống, v.v.).
  + Server gửi cookie đó xuống **trình duyệt** bằng header HTTP:
  + Set-Cookie: .AspNetCore.Identity.Application=eyJhbGciOi...; path=/; httponly; secure

**🍪 2. Trình duyệt lưu cookie**

* Trình duyệt (Client) tự động **lưu cookie** này vào bộ nhớ (hoặc tạm thời, tùy cấu hình).
* Cookie này **thuộc domain của ứng dụng web** (vd: localhost:5001).

**🔁 3. Trình duyệt gửi request tiếp theo**

* Mỗi khi bạn **chuyển trang**, **gửi form**, hoặc **reload lại**,  
  trình duyệt **tự động đính kèm cookie** vào header của request:
* Cookie: .AspNetCore.Identity.Application=eyJhbGciOi...

**🖥️ 4. Server nhận request**

* Middleware **Authentication** của ASP.NET chạy trước controller.
* Nó đọc cookie từ header request, giải mã, kiểm tra:
  + Cookie có hợp lệ không?
  + Có hết hạn không?
  + Có bị chỉnh sửa không?
* Nếu hợp lệ → tạo User.Identity với:
* User.Identity.IsAuthenticated = true
* User.Identity.Name = "Tên người dùng"
* Nếu cookie hết hạn hoặc sai → không tạo User.Identity, IsAuthenticated = false.
* Controller/Action trang Logout
* public async Task<IActionResult> Logout()
* {
* await \_signInManager.SignOutAsync(); // thực hiện đăng xuất xóa Cookie ở Client ( trình duyệt )
* return RedirectToAction("Login", "Account");
* }
* Dùng [Authorize] ở Controller Admin

[Area("Admin")]

[Authorize]

* Tức là Phải Login (var result = User.Identity.IsAuthenticated == true ) mới vào được trang (Controller / action ) Admin

Setting ở grogram .cs

builder.Services.ConfigureApplicationCookie(options =>

{

options.LoginPath = "/Account/Login"; // <-- nếu như chưa Login thì nó sẽ đi đến đường dẫn này

});