TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC



**Khoa Công Nghệ Thông Tin**

****

**Chuyên đề phát triển web 2** | HKI – [2019 – 2020]

PRJ – REPORT

**Tìm hiểu ASP.NET Core 3.1**

**Nhóm M:**

* **Ngô Đình Trí–** 18211TT2467
* **Nguyễn Y Trinh–** 18211TT4551
* **Doãn Hồng Thăng –** 18211TT4557

**BẢNG PHÂN CHIA CÔNG VIỆC**

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ & Tên** | **Công việc** |
| 18211TT4557 | Doãn Hồng Thăng | Hướng dẫn gửi mail MailKit |
| Hướng dẫn tích hợp Api login with google |
|  |
|  |
| Giới thiệu về UserIdentity và User Manager |
| Giới thiệu về RoleIdentity và Role Manager |
| Giới thiệu về SigninManager |
| 18211TT2467 | Ngô Đình Trí | Giới thiệu về EntityFrameworkcore |
| Giới thiệu về Migration |
| Giới thiệu về submit form |
| Giới thiệu về truy vấn dữ liệu bằng linq |
| Giới thiệu về Identity Authentication |
| Hướng dẫn thiết lập Identity |
| Hướng dẫn tích hợp SASS |
| 18211TT4551 | Nguyễn Y Trinh | Cài đặt Asp.net core |
| Cài đặt SDK cho asp.net core 3.1 |
| Cách khởi tạo project |
| Giới thiệu các giao thức web application |
| Giới thiệu mô hình mvc |
| Mô tả cấu trúc main có trong asp.net core |
| Giới thiệu Middleware trong asp.net core |
| Giới thiệu Routing trong asp.net core |
| Giới thiệu về Model Validation |

**Mục lục**

[I. Cài đặt Visual Studio và một số thủ thuật nhỏ 4](#_Toc55301968)

[1. Cài đặt Microsoft Visual Studio 4](#_Toc55301969)

[2. Cài đặt SDK Core 3.1 5](#_Toc55301970)

[3. Cách khởi tạo project 6](#_Toc55301971)

[4. Giới thiệu mô hình MVC 10](#_Toc55301972)

[5. Mô tả cấu trúc main có trong ASP.NET Core 11](#_Toc55301973)

[6. Giới thiệu Middleware trong ASP.NET Core 12](#_Toc55301974)

[7. Giới thiệu Routing trong ASP.NET Core 14](#_Toc55301975)

[8. Giới thiệu về Model Validation 22](#_Toc55301976)

[9. Tích hợp SASS 23](#_Toc55301977)

[II. Khái niệm căn bản trong Asp.net core 28](#_Toc55301978)

[10. Giới thiệu về EntityFrameworkcore 28](#_Toc55301979)

[11. Giới thiệu về Migration 30](#_Toc55301980)

[12. Giới thiệu về submit form 32](#_Toc55301981)

[13. Giới thiệu về truy vấn dữ liệu bằng LINQ 34](#_Toc55301982)

[14. Giới thiệu về Identity Authentication 36](#_Toc55301983)

[15. Hướng dẫn thiết lập Identity 36](#_Toc55301984)

[16. Giới thiệu về UserIdentity và User Manager 37](#_Toc55301985)

[17. Giới thiệu về RoleIdentity và Role Manager 39](#_Toc55301986)

[18. Giới thiệu về SigninManager 39](#_Toc55301987)

[III. Khái niệm nâng cao trong Asp.net core 40](#_Toc55301988)

[19. Hướng dẫn gửi mail MailKit 40](#_Toc55301989)

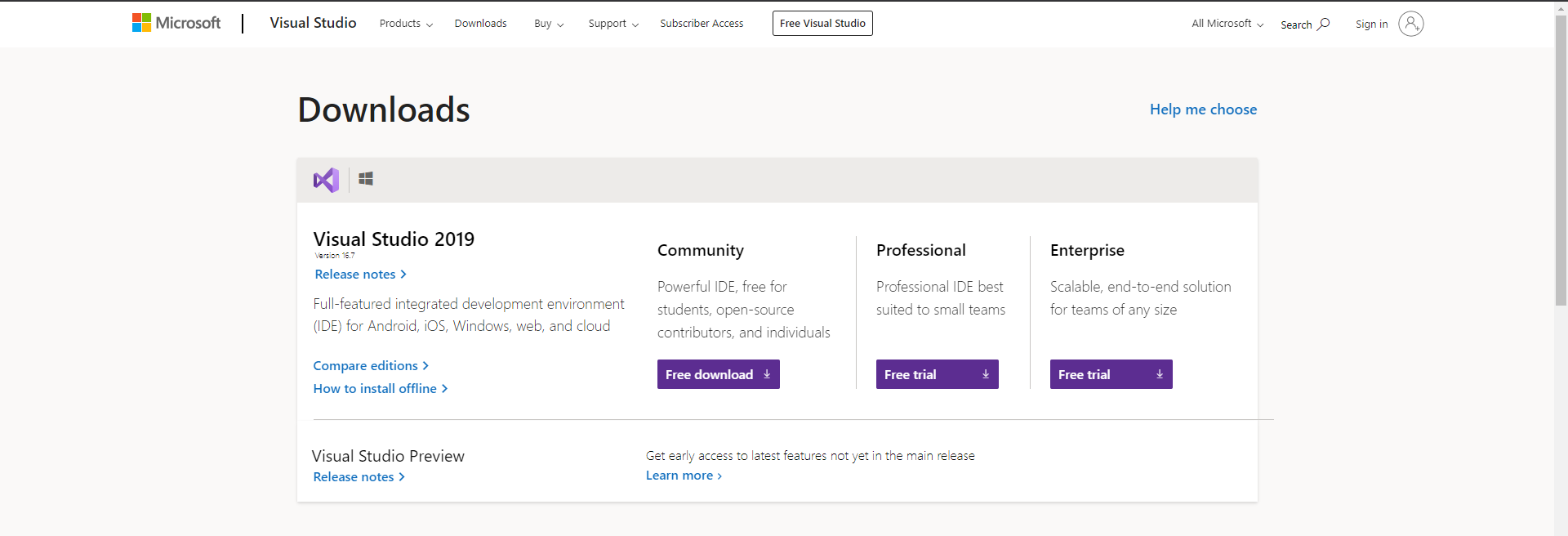
**Tổng quan**

# Cài đặt Visual Studio và một số thủ thuật nhỏ

## Cài đặt Microsoft Visual Studio

Để cài đặt Microsoft Visual Studio

Ta vào trang : <https://visualstudio.microsoft.com/downloads/>



1. Trang download Microsoft Visual Studio

Có 3 sự lựa chon

**Community**: download free đối với cá nhân không phải là các nhà lập xây dựng sản phẩm thực tế và có sự hạn chế khi phát triển dự án nhỏ và lớn

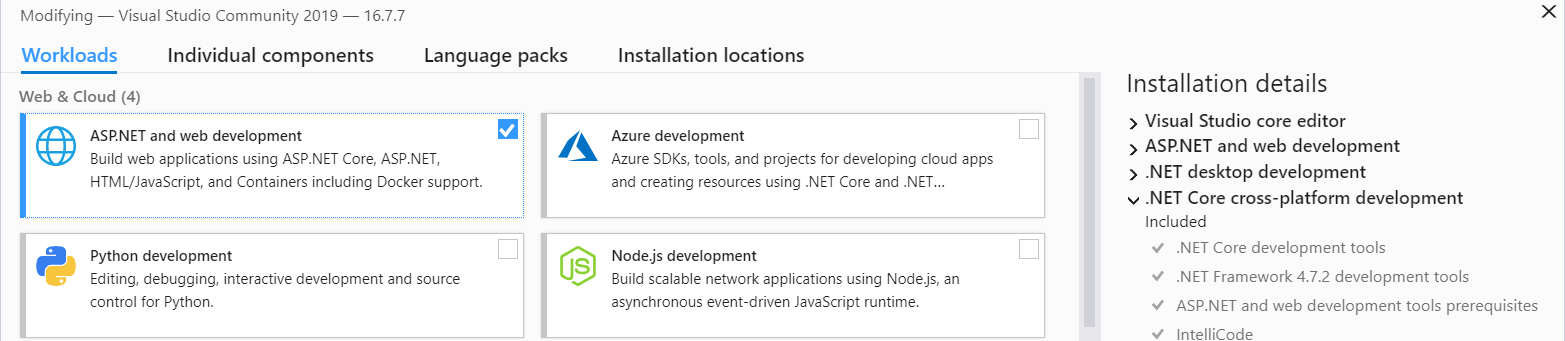
**Professional** :dowload trả phí đối với nhà sán lập sản phẩm ở quy mô nhỏ. Còn sự hạn chế tuy k lớn

**Enterprise**: dowload trả phí đối với nhà sán lập sản phẩm ở quy mô lớn.

-Sau khi tải về bản **Community** , chúng ta tiến hành cài đặt bình thường.

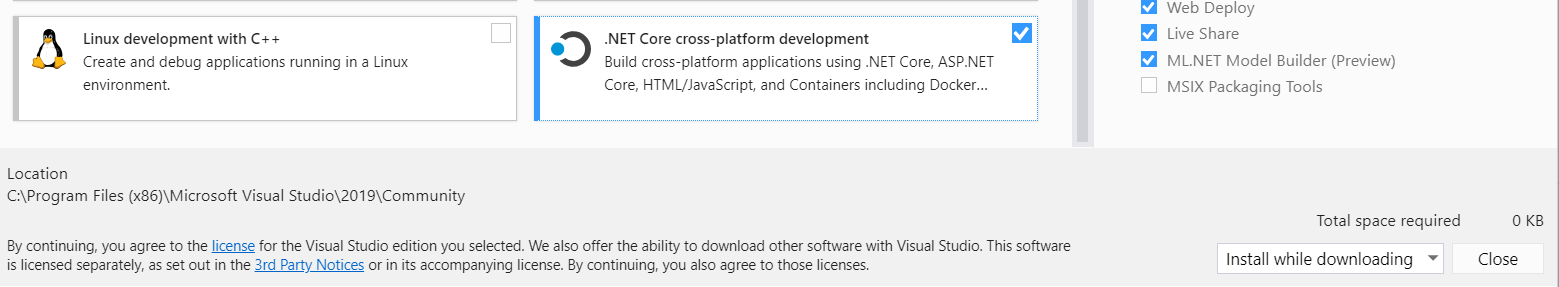
Sau đó khi hộp thoại Visual Studio Installer hiện lên ở phần Workloads trong Visual Studio Installer. Chúng ta cần cài 2 thứ:

* ASP.NET and web development



1. ASP.NET and web development

* .NET Core cross-platform development

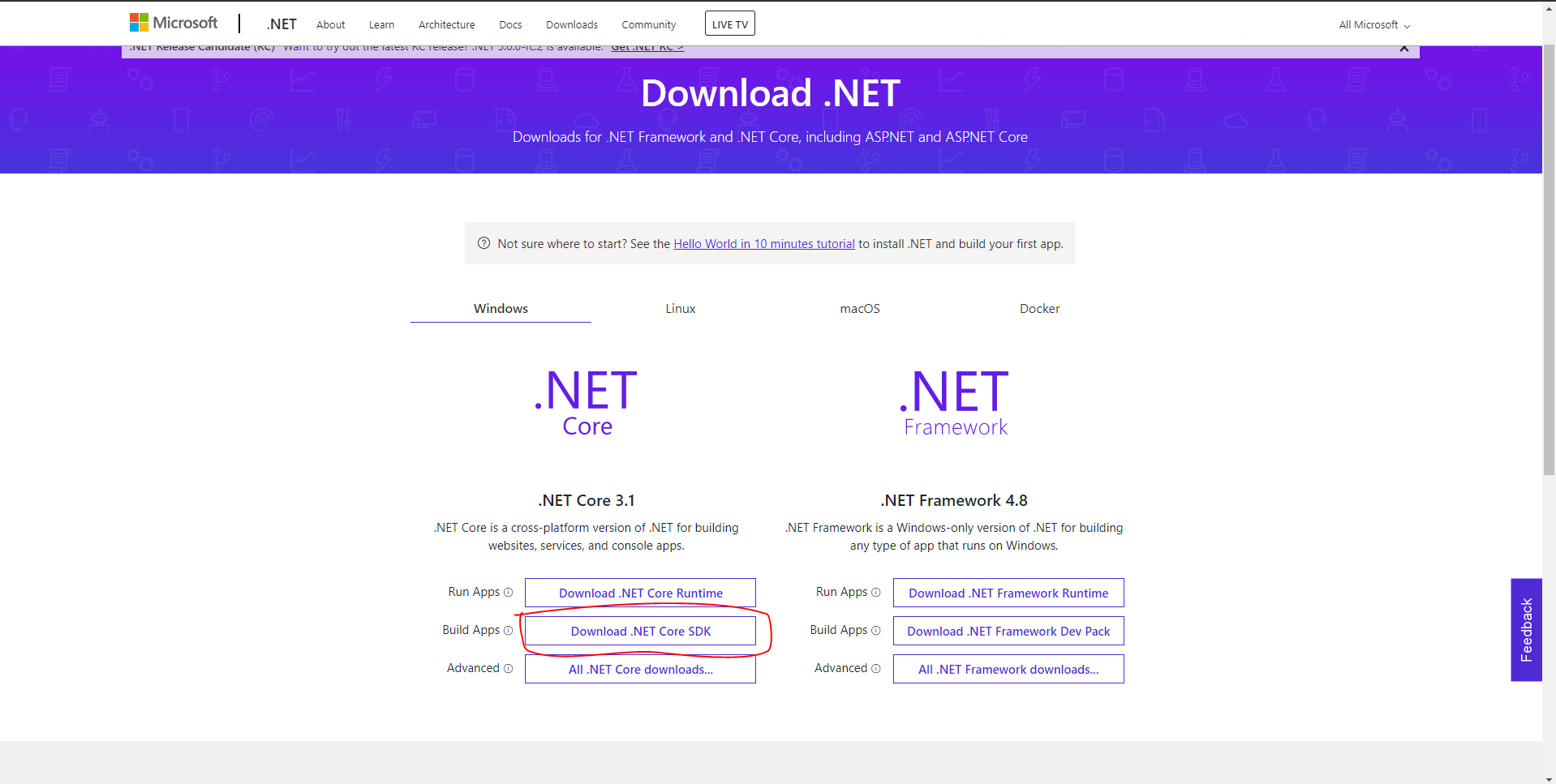


1. Hình 3. .NET Core cross-platform development

🡪Khi cài đặt xong đi tiếp đến Cài đặt SDK ở bên dưới.

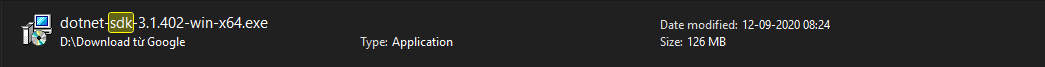
## Cài đặt SDK Core 3.1

***-***Vào trang : <https://dotnet.microsoft.com/download>



1. Trang download .NET Core

-Sau khi tải xuống ta được hình bên dưới

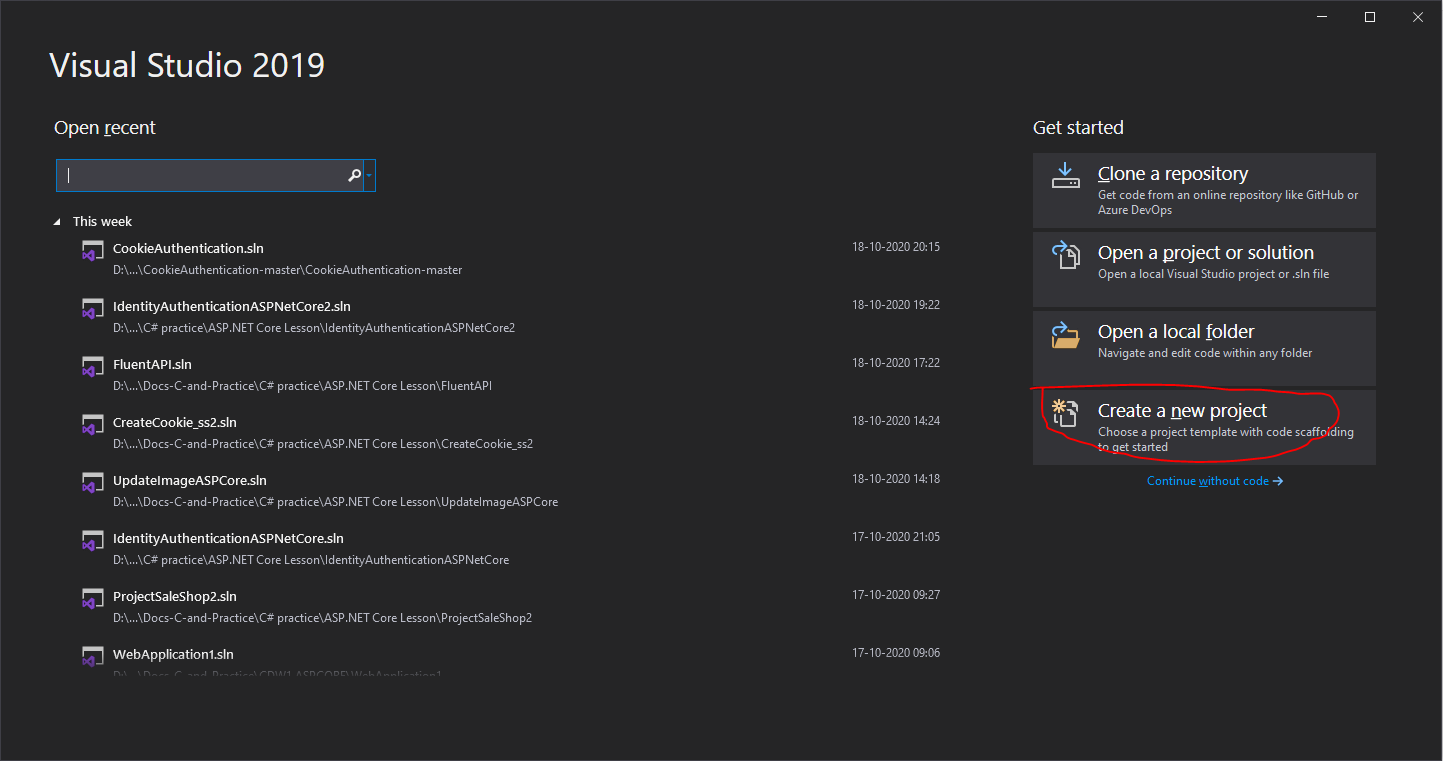


-Tiếp theo nhấn và khởi chạy với đường dẫn lúc đầu khi cài đặt visual

## Cách khởi tạo project

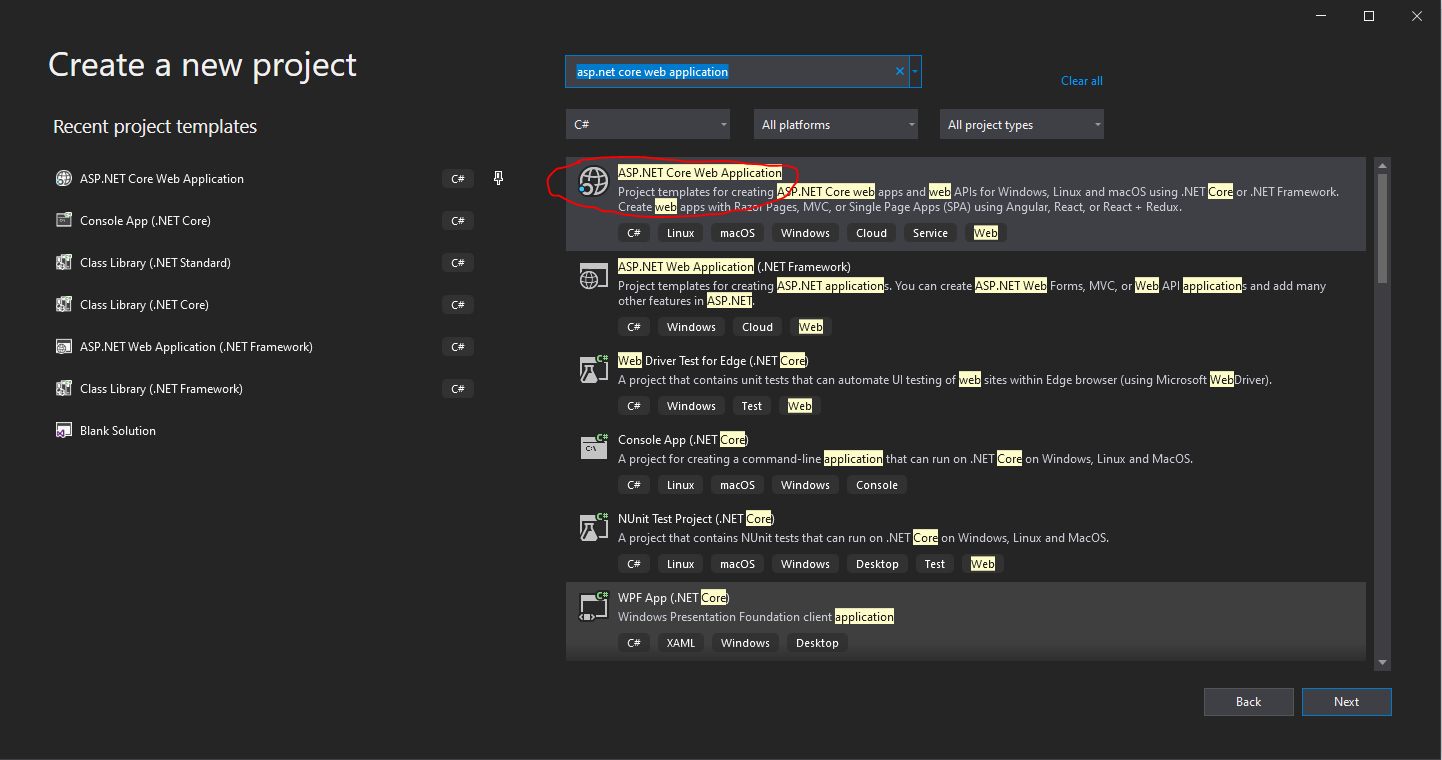
***-***Mở giao diện Microsoft Visual Studio lên

***B1: Nhấn vào nút***



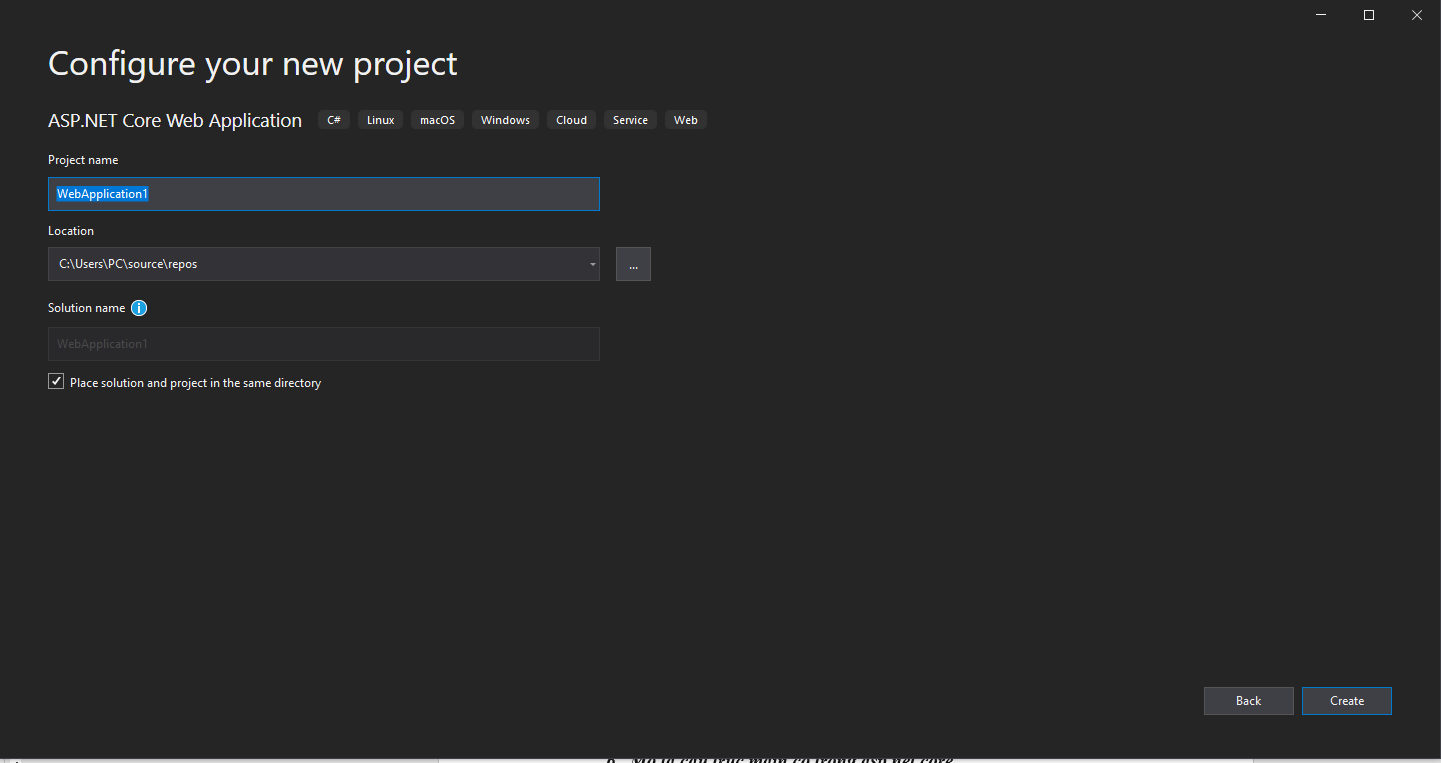
1. Giao diện khởi tạo project của Microsoft Visual Studio.

***B2: Tìm kiếm “asp.net web application”***

******

1. Tìm mục “asp.net web application”

***B3: Khởi tạo name và đường dẫn***

******

1. Khởi tạo mới project ASP.NET Core Web Application

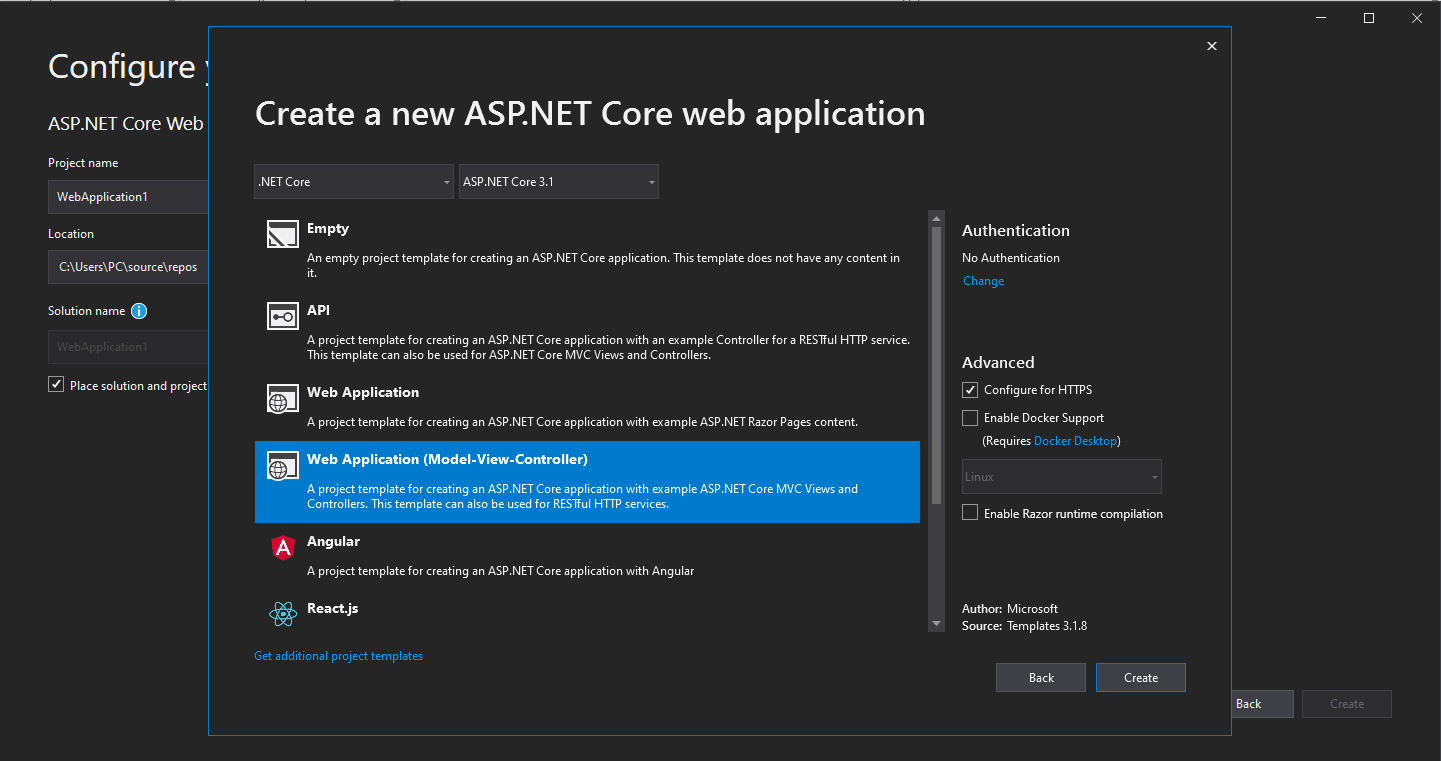
Project name : Khởi tạo lại tên project nếu muốn

Location: Đường dẫn lưu trữ của project

=> Sau khi thao tác xong 2 phần trên nhấn create

***B4: Chọn Project Template***

-Bước tiếp theo về việc khởi tạo project ta nói sơ qua các project templates của nó.



1. Các project templates của project ASP.NET Core Web Application

**Empty**

-Khi khởi tạo mặc đinh nó sẽ không có controller, action ,model,Routing. Người dùng phải xây dựng lại theo cách của mình nếu muốn

=> Dành cho những ví dụ hoặc người thích sáng tạo theo cách của bản thân

**API**

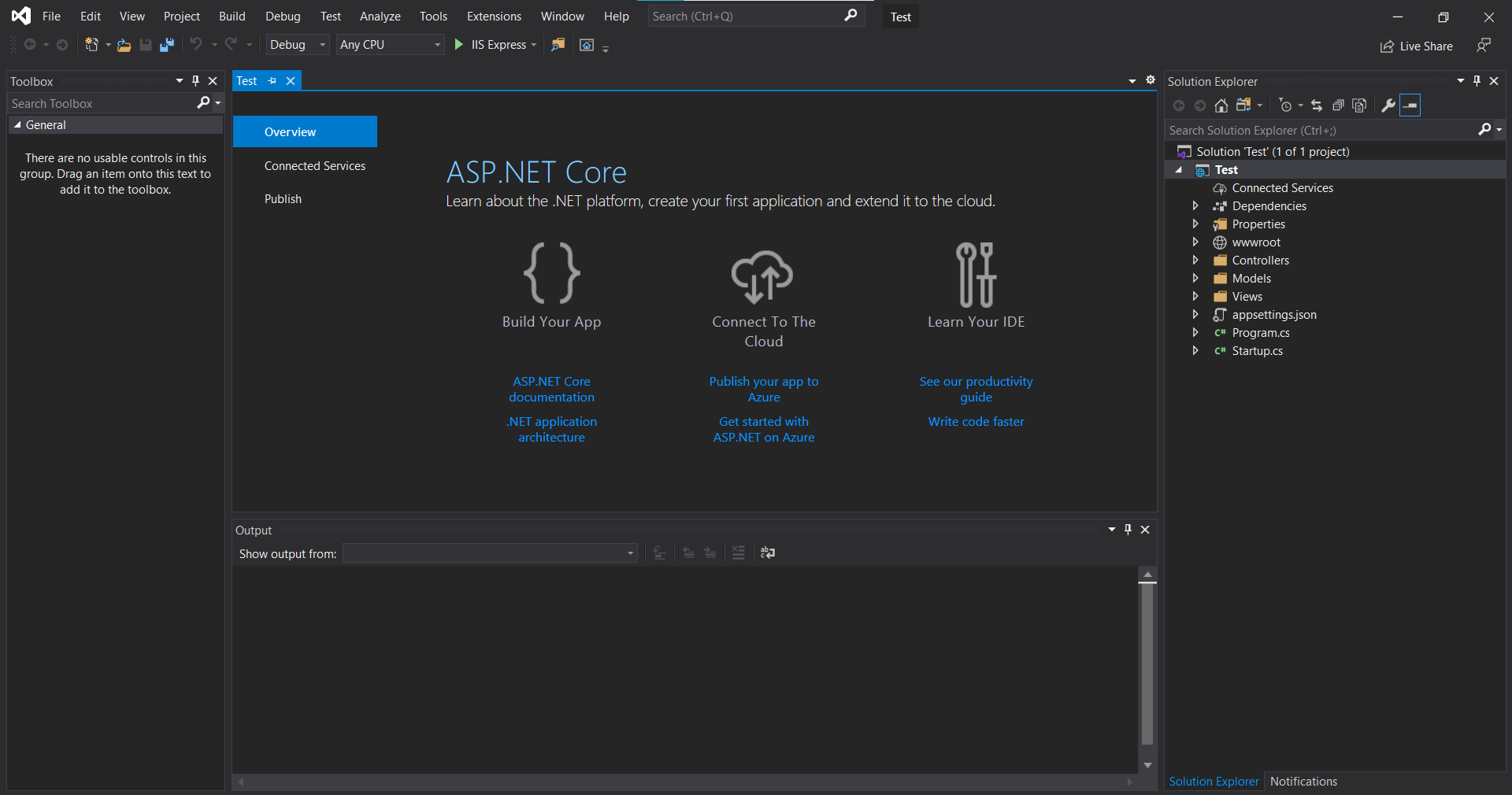
-Khi khởi tạo, mặc định project sẽ nhận được các thư viện liên quan đến API để khởi tạo API cho các framwork như React,Angular,Vue...

**Web Application(Model-View-Controller)**

-Tạo ra model, View, Controller tức là một bản có cấu trúc mvc và routing hoàn chỉnh cho việc phát triển web

Nhấn chọn **Web Application(Model-View-Controller)** để tiếp tục.

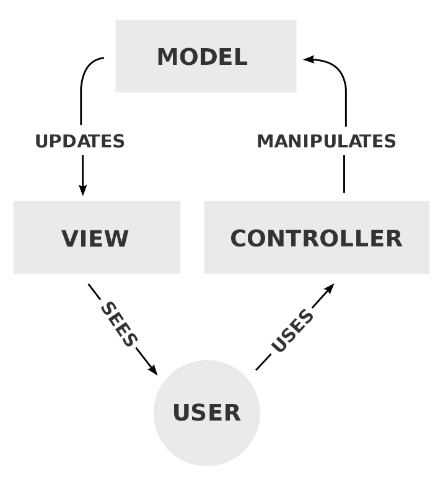
Sau khi project được tạo. Ta được project như sau:



1. Project vừa được tạo mới

## Giới thiệu mô hình MVC

-Mô hình mvc được viết tắt bởi 3 từ model - view – controller. Mô hình được sử dụng trong kỹ thuật phần mềm



1. Mô hình MVC

Model

Là phần chức năng lưu trữ toàn bộ dữ liệu, được kết nối với sql, để tạo các câu truy vấn, truy xuất và xử lý dữ liệu ... Là một cầu nối giữa view và controller

View

Đây là phần giao diện dành cho người xử dụng, nó chứa các template của trình duyệt của phần mềm web. Nơi người dùng lấy và xem các thông tin từ cơ sở dữ liệu

Controller

Bộ phần có nhiệm vụ xử lý yêu cầu người dùng thông qua view. Từ đó lấy ra các dữ liệu phù hợp nhất cho người dùng. Bên cạnh đó Controller có chức năng kết nối với model

Về ưu nhược điểm xem thêm tại: <https://monamedia.co/mvc-la-gi-ung-dung-cua-mo-hinh-mvc-trong-lap-trinh/>

## Mô tả cấu trúc main có trong ASP.NET Core

Program.cs

******

1. Code file Program.cs

-Hàm main ở trên chứa một public static void main() method. Mặc định khi khởi tạo console application (ứng dụng giao diện điều khiển ) trong .net mặc định .Net framework biết rằng đây là nôi thực thi toàn bộ project

* **Tại sao chung ta lại phải có main() method ?**

**-**Ban đầu khi khởi tạo asp.net core, nó sẽ đi vào hàm main với dòng

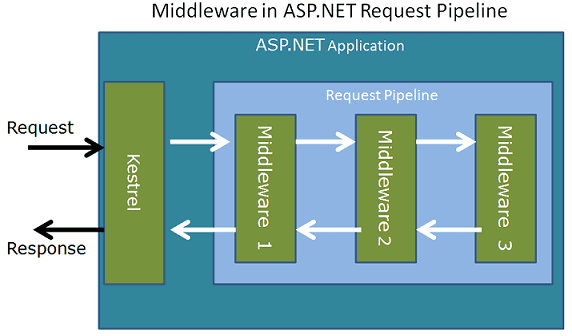
CreateHostBuilder(args).Build().Run();

-CreateHostBuilder() method trả về một lệnh triển khai

một đối tượng thực thi giao diện IHostBuilder. Host là một lớp tĩnh có thể được sử dụng để tạo một phiên bản của IhostBuilder nhằm khởi tạo máy chủ nội bộ

## Giới thiệu Middleware trong ASP.NET Core

***-***Middleware là thành phần các lớp được thực thi theo yêu cầu của ứng dụng ASP.NET Core. Nó là đường ống xử lý yêu cầu “request pipeline”



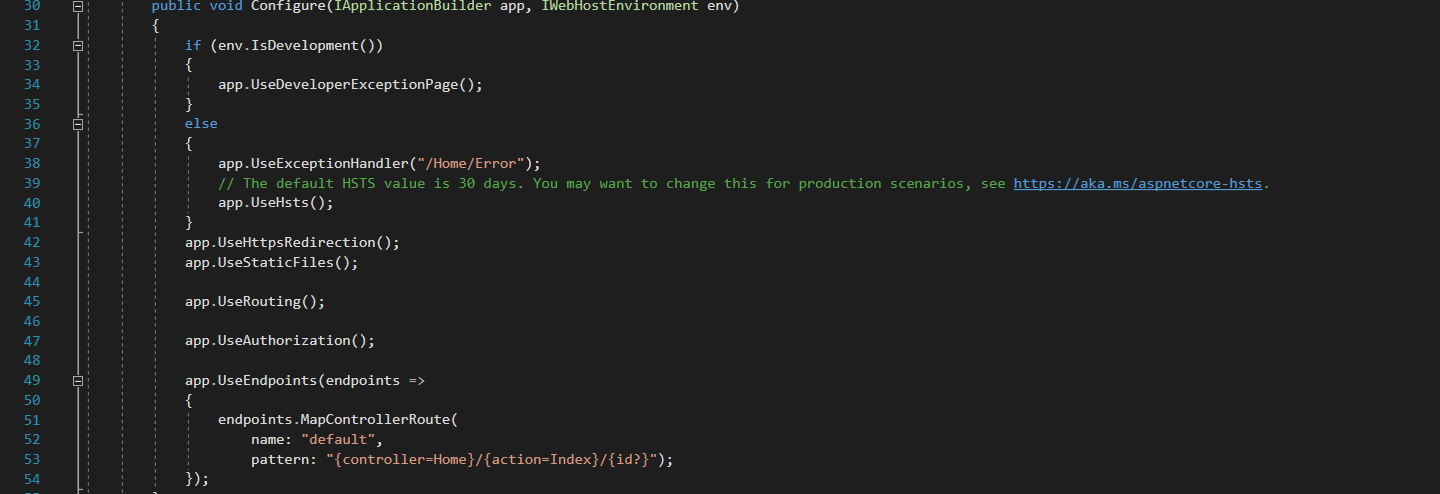
1. Sơ đồ Middleware trong ASP.NET Core

- Có thể thấy ở đây, khi một request được đưa lên nó sẽ đi vào các middleware, khi đó các middlware khác nhau sẽ xử lý phần request và trả về cho thằng đứng sau, khi đến cuối middleware nó sẽ trả lại reponse



1. Demo về middleware

-Khi một middleware được đưa lên, nó sẽ đi qua các app.run để trả về một reponse



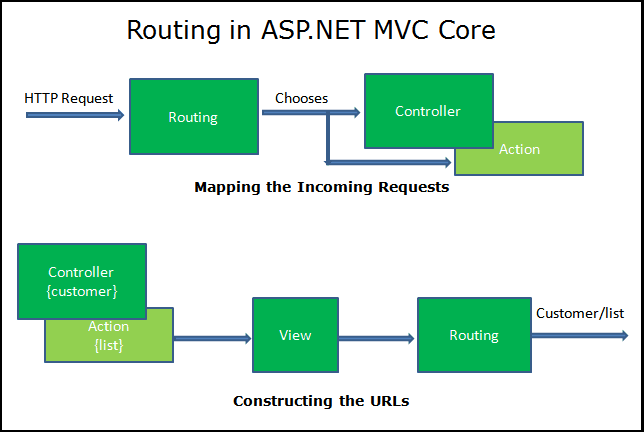
1. Middleware có sẵn.

-Thông thường với web application (mvc) nó có sẵn các middle ware. Về các phương thức Use() “UseHttpsRedirection(),UseStaticFiles(),UseRouting()...”

=> Nó là phương thức mở rộng, tương tự như run() ngoại trừ việc bao gồm các next để gọi middleware kế tiếp trong chuỗi

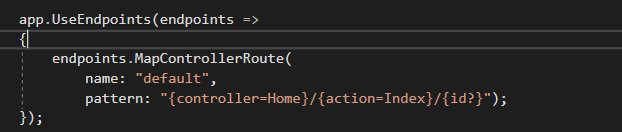
## Giới thiệu Routing trong ASP.NET Core

-Routing là quá trình xem xét các url request gửi đến và chỉ đường cho đến các controller action. Tạo các url đầu ra quá trình này đảm nhiệm bởi routing middleware trong ASP.NET Core.

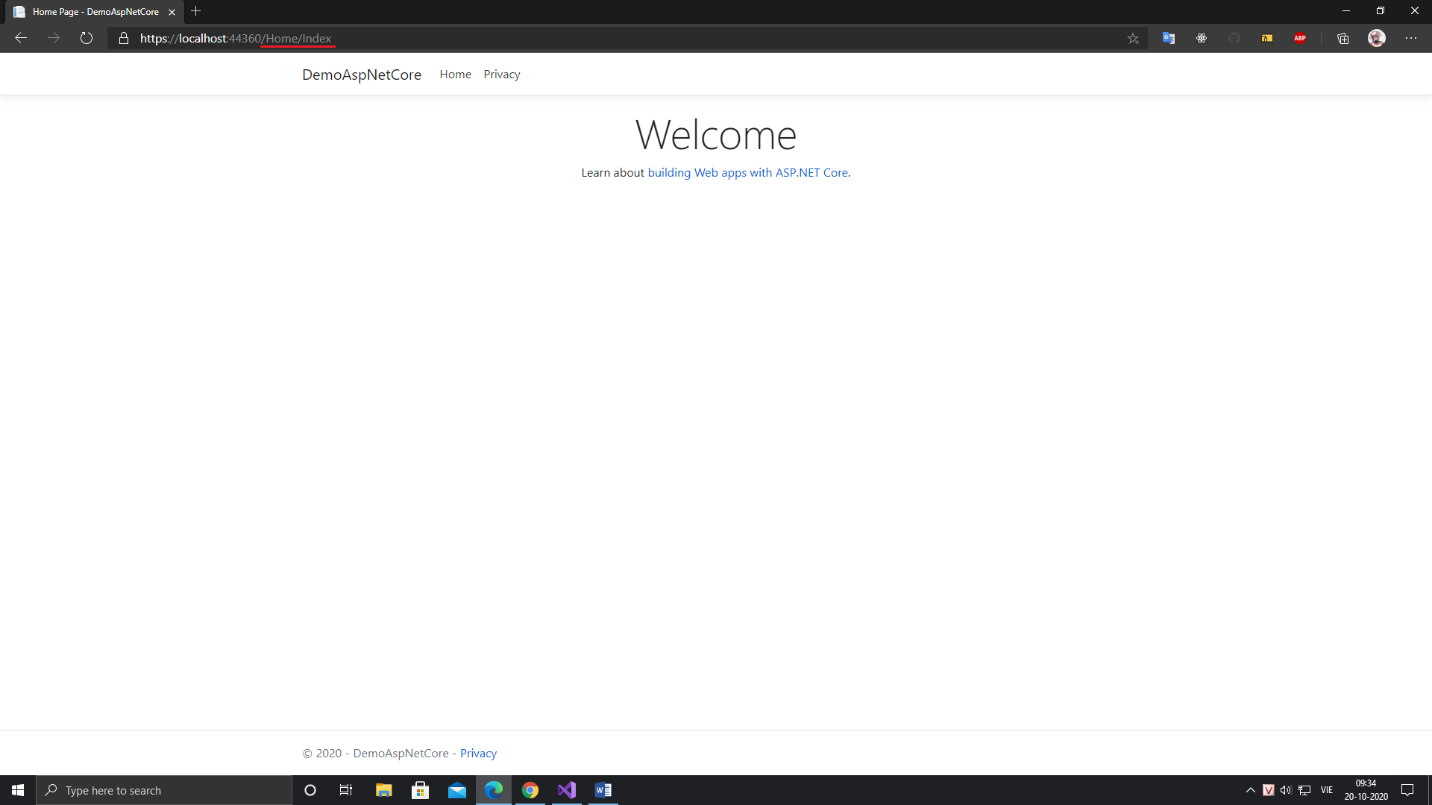


1. Mô hình Routing trong ASP.NET Core

Đối với Routing middleware trong ASP.NET Core như sau:



1. Routing middleware trong ASP.NET Core



1. Trang chủ mới được tạo

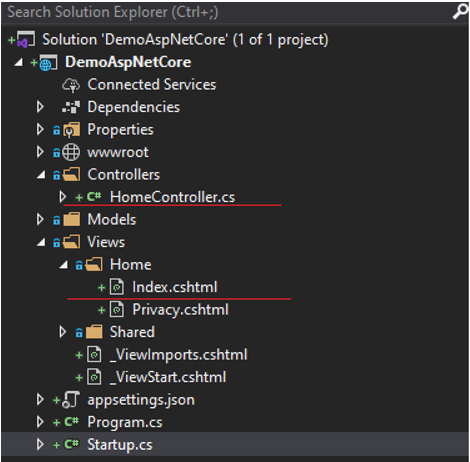
Khi nhấn /Home/Index => Nó tương tự như **pattern** ở phía trên

Controller = “Home”

Action = “Index”

Id = “” lúc này id chưa có nên nó sẽ là rỗng”

=> Sau khi gửi url request lên ta thấy được routing trong middleware sẽ tìm các controller action

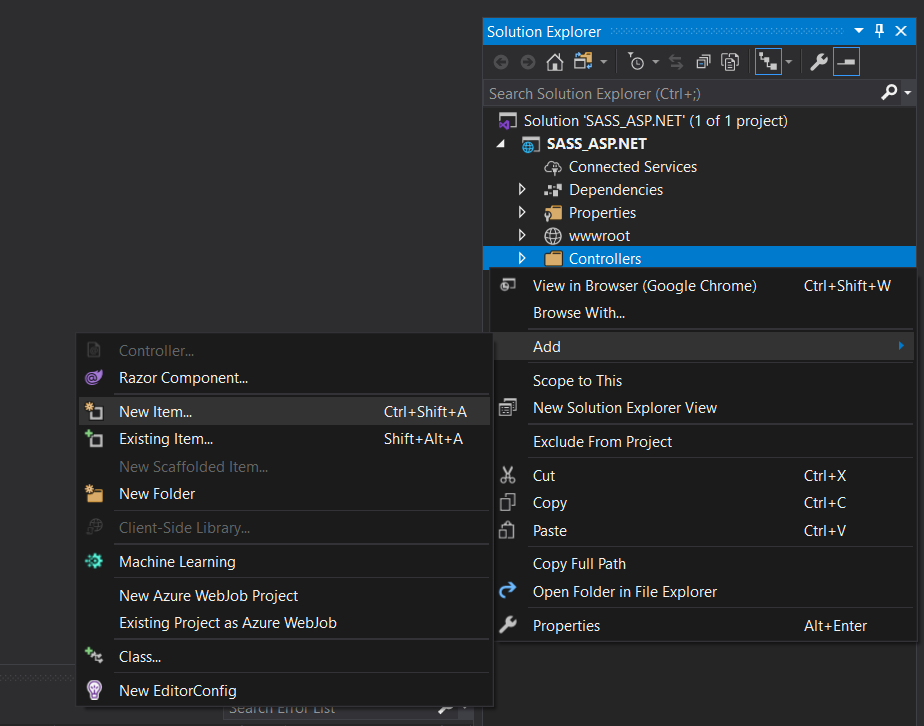


1. Controllers và Views

**Cách thêm 1 Routing.**

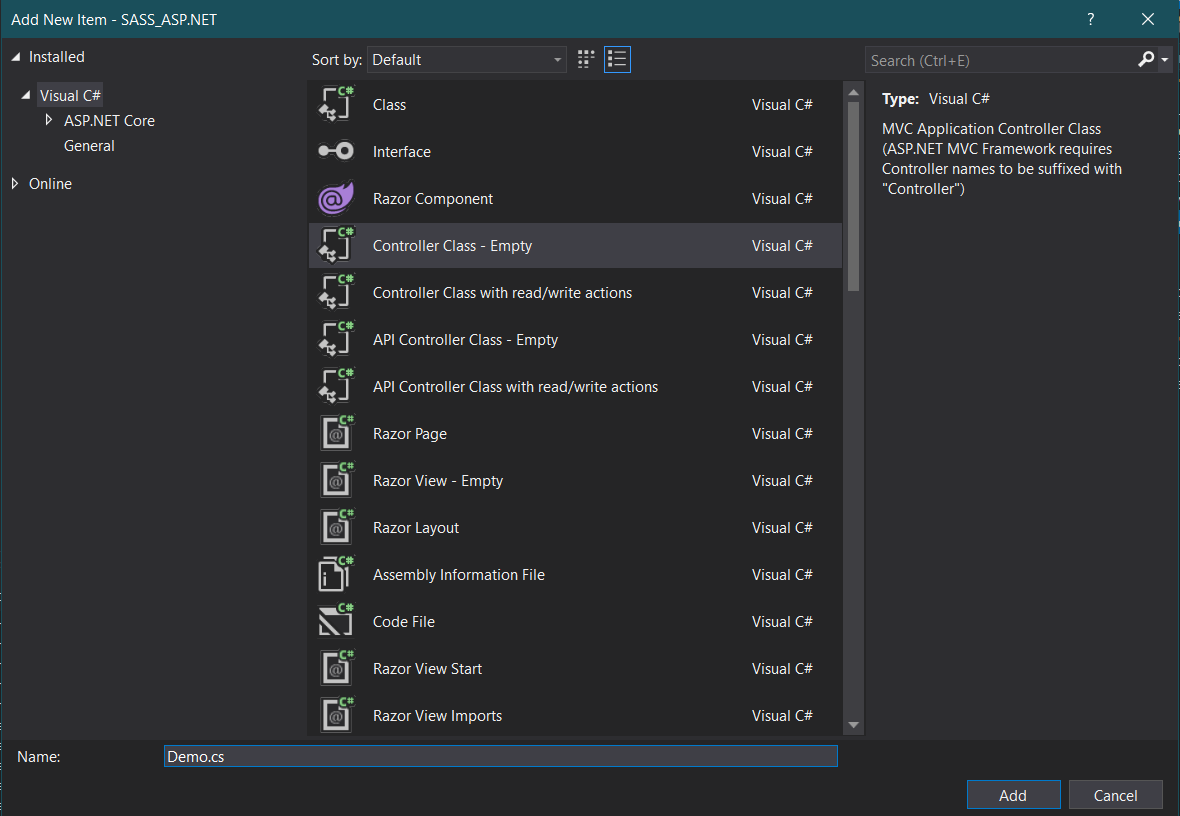
* Click chuột phải vào thư mục Controllers 🡪 Add 🡪 New Item.

**(Hoặc Ctrl + Shift + A)**



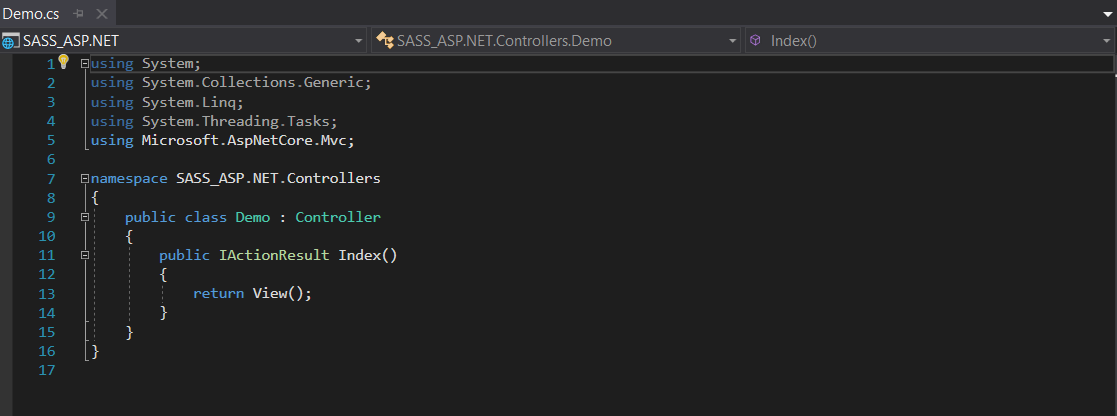
1. Tạo “New Item”

* Chọn Controller Class – Empty. Đặt tên file rồi nhấn “Add”.



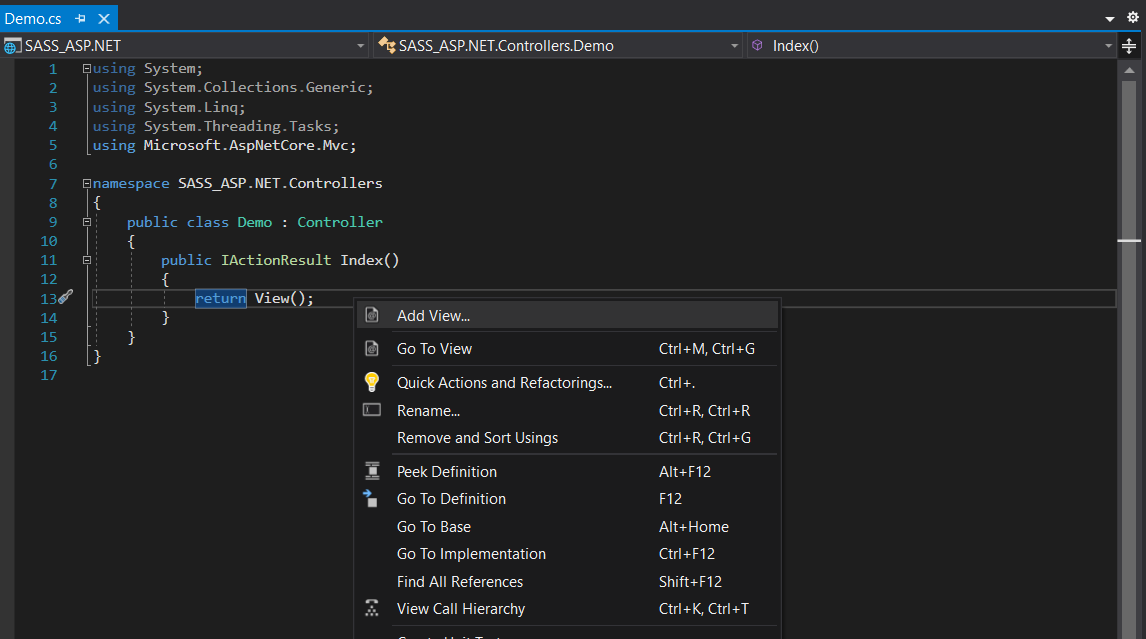
1. Chọn kiểu Controller Class - Empty

* Ta được 1 file Demo.cs như sau. Có phương thức IActionResult trả về View().



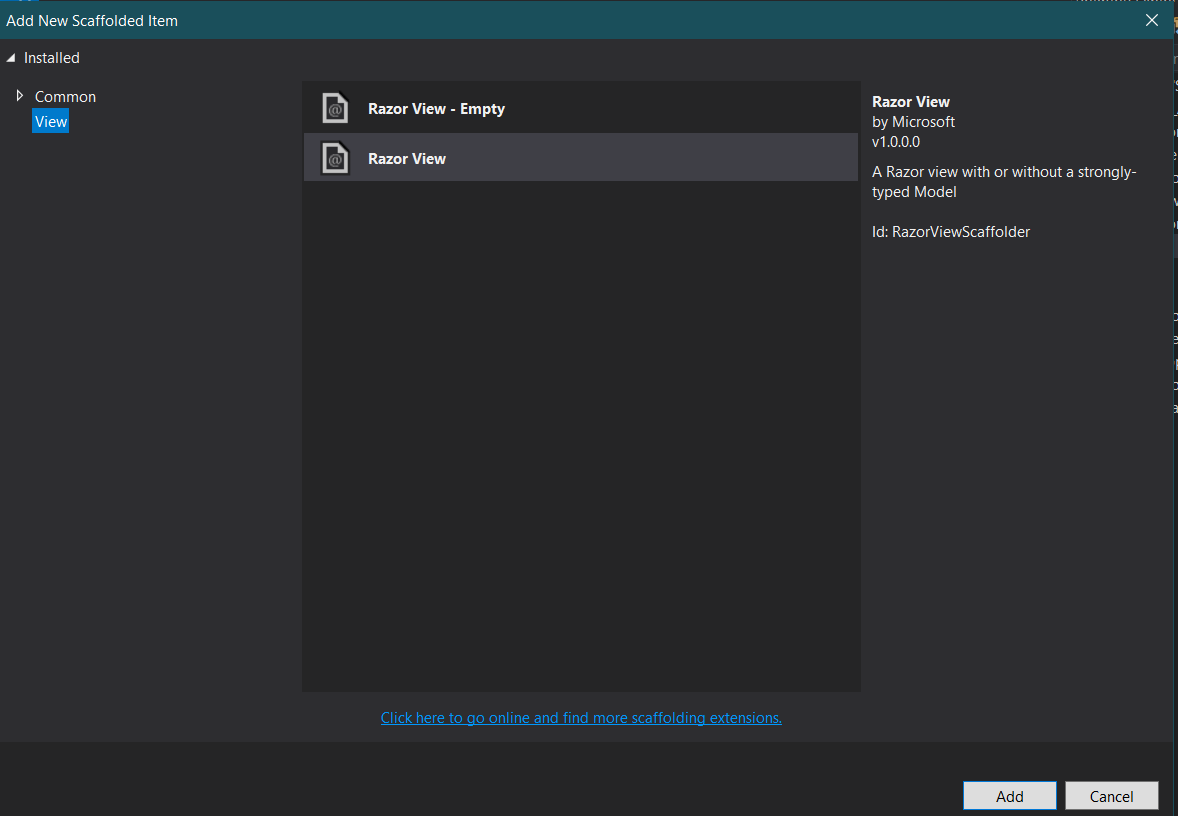
1. Code file Controller, IActionResult trả về View().

* Ta trỏ chuột vào View() và click chuột phải. Chọn Add View…



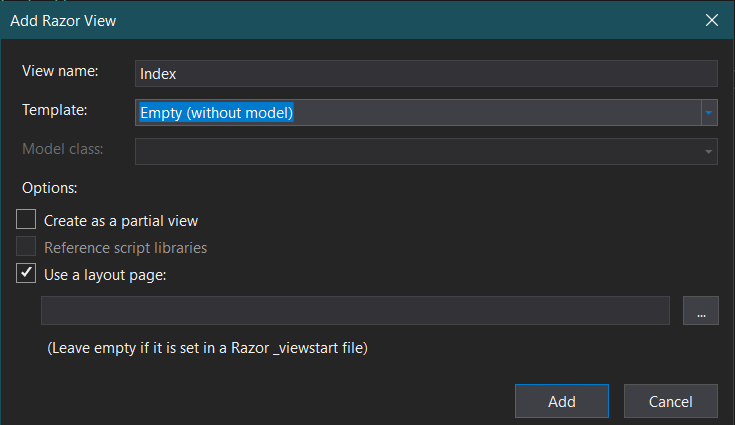
1. Add View() được trả về

* Chọn Razor View.



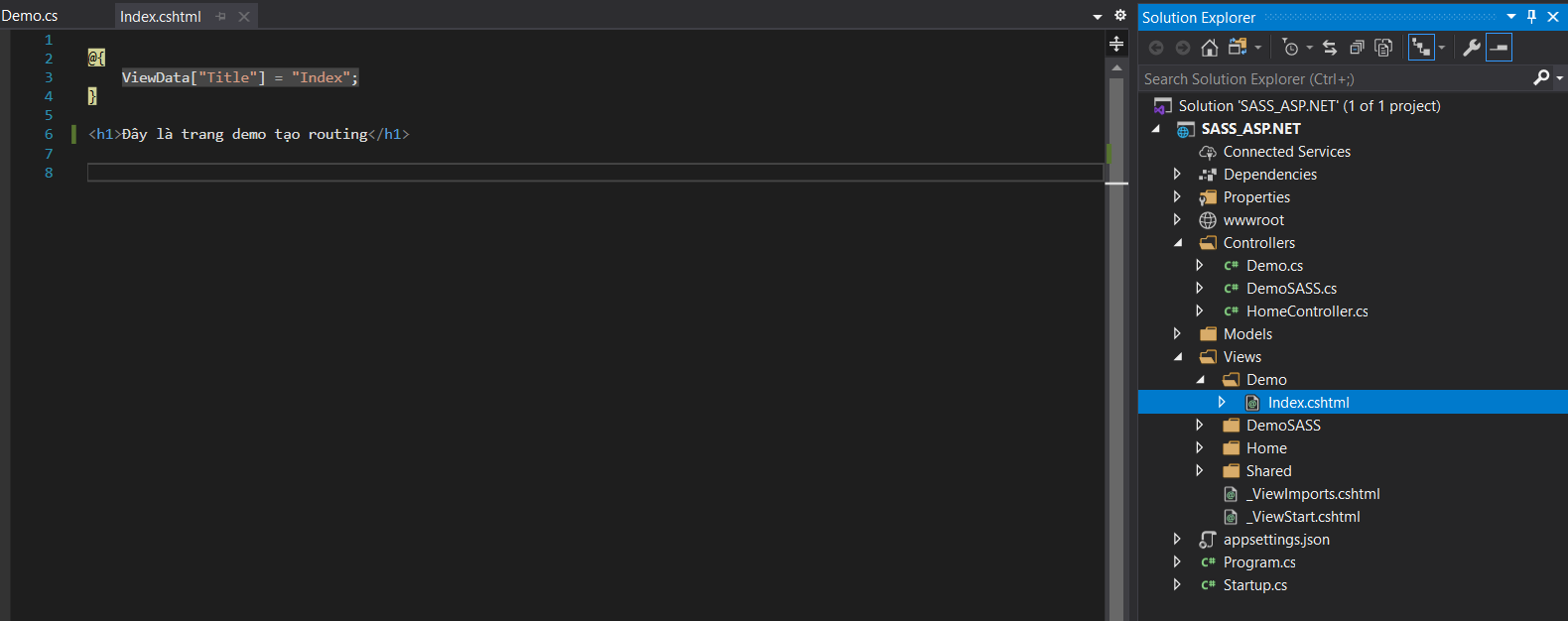
1. Razor View

* Tùy chỉnh một số thuộc tính. Ở đây mình để mặc định. Rồi nhấn “Add”



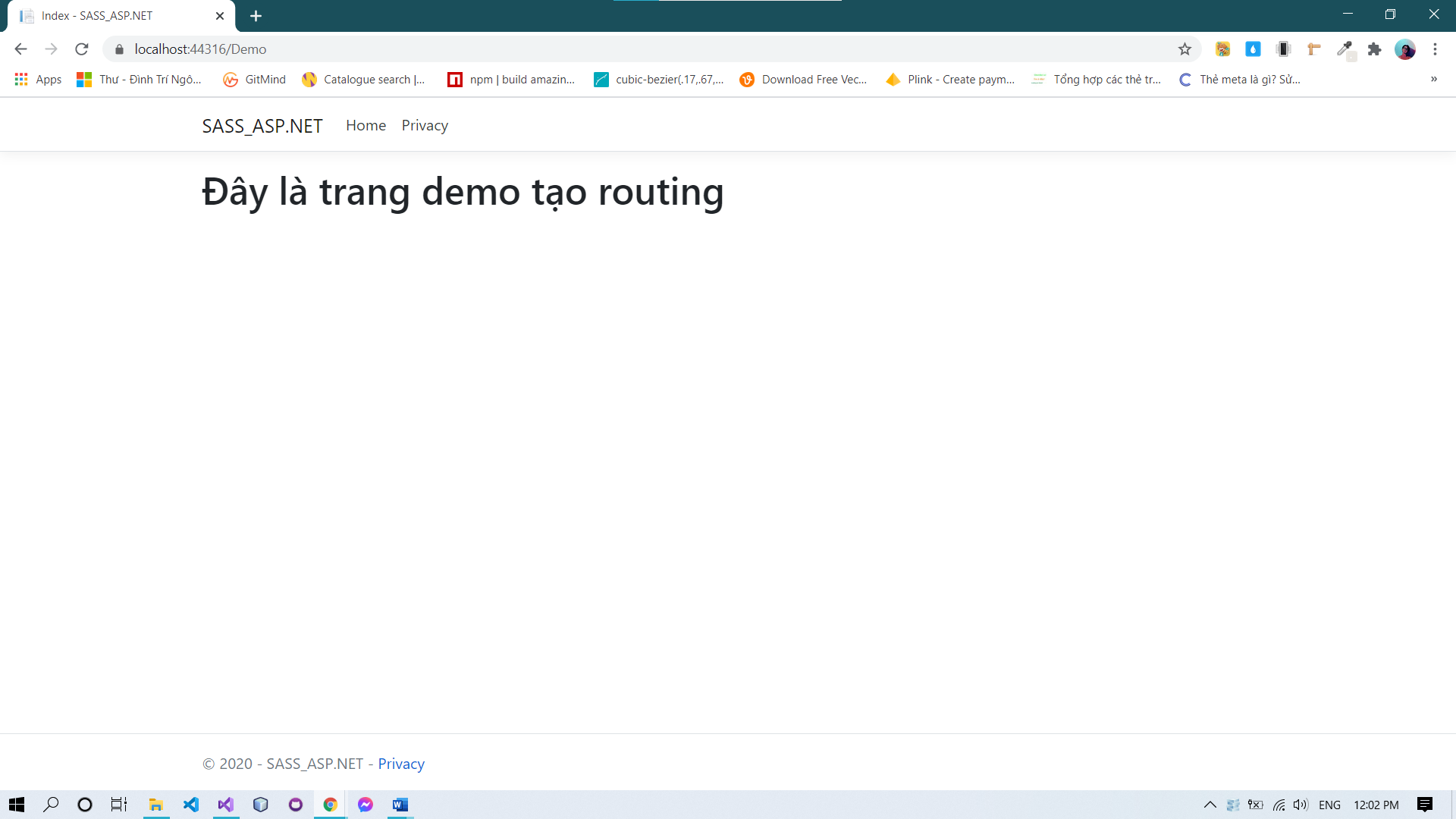
1. Cấu hình Razor View

* Một file Index.cshtml sẽ được tự động tạo trong folder Demo trong Views.



1. . Index.cshtml được trả về

* Chạy chương trình vào đường dẫn localhost:xxxx/Demo. Ta sẽ truy cập tới trang vừa tạo.

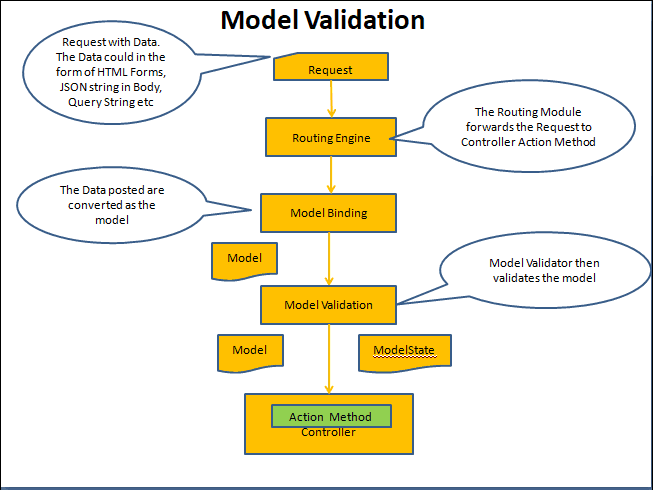


1. Trang Web vừa được tạo

## Giới thiệu về Model Validation

Tự động được map data vào các tham số của action bởi Model Binder

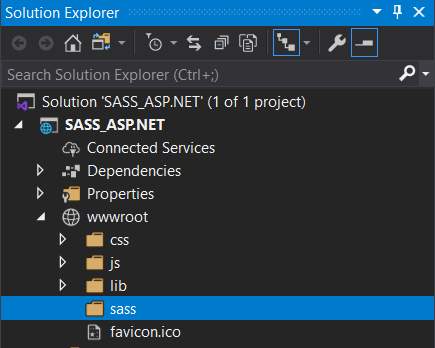
hợp lệ không. Quá trình kiểm tra này có thể được hoàn thành bởi client trước khi gửi lên server hoặc server kiểm tra sau khi nhận được từ client



1. Mô hình Model Validation

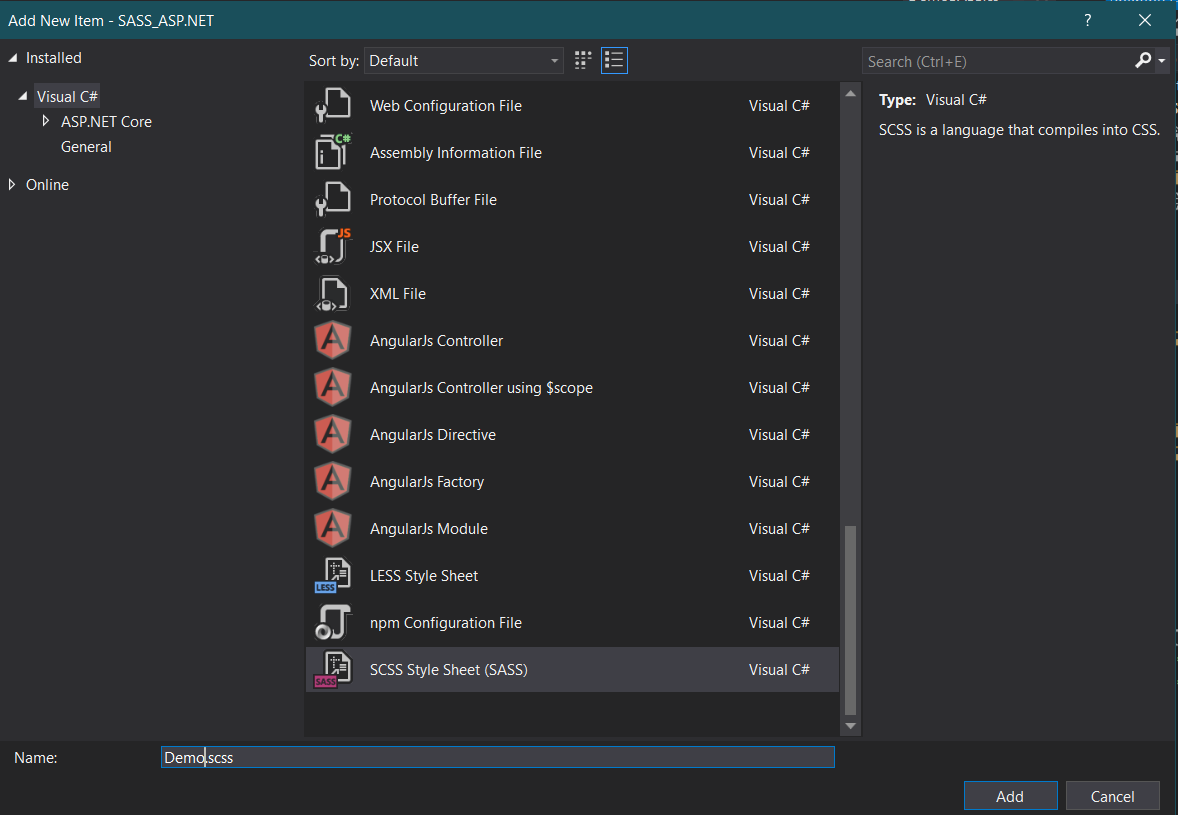
## Tích hợp SASS

* Tạo folder sass trong wwwroot.



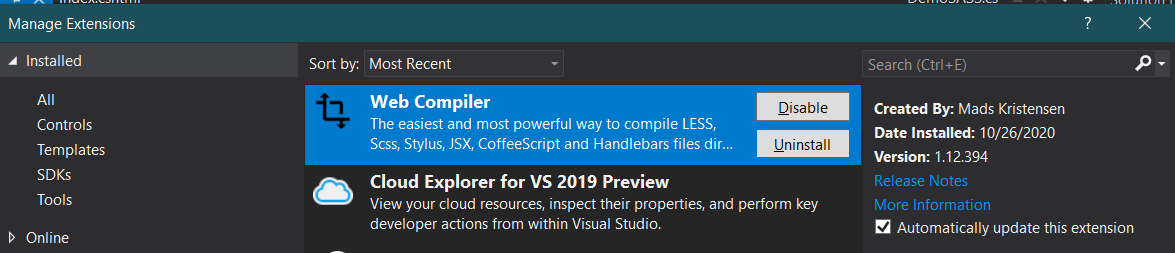
1. wwwroot

* Chọn thư mục sass rồi nhấn tổ hợp phím Ctrl + Shift + A để thêm item mới. Chọn SCSS Style Sheet (SASS). Đặt tên file rồi tạo mới.



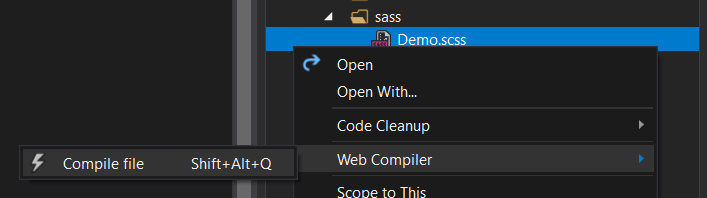
1. Chọn loại SASS để tạo mới

* Cài extension “Web Compiler”. Có thể cài trong Manage Extensions của Microsoft Visual Studio. Hoặc tải online ở [link](https://marketplace.visualstudio.com/items?itemName=MadsKristensen.WebCompiler) này.



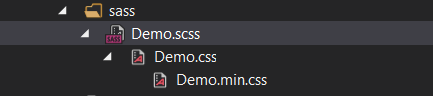
1. Web Compiler

* Chọn file .scss cần dùng rồi complie sang .css.



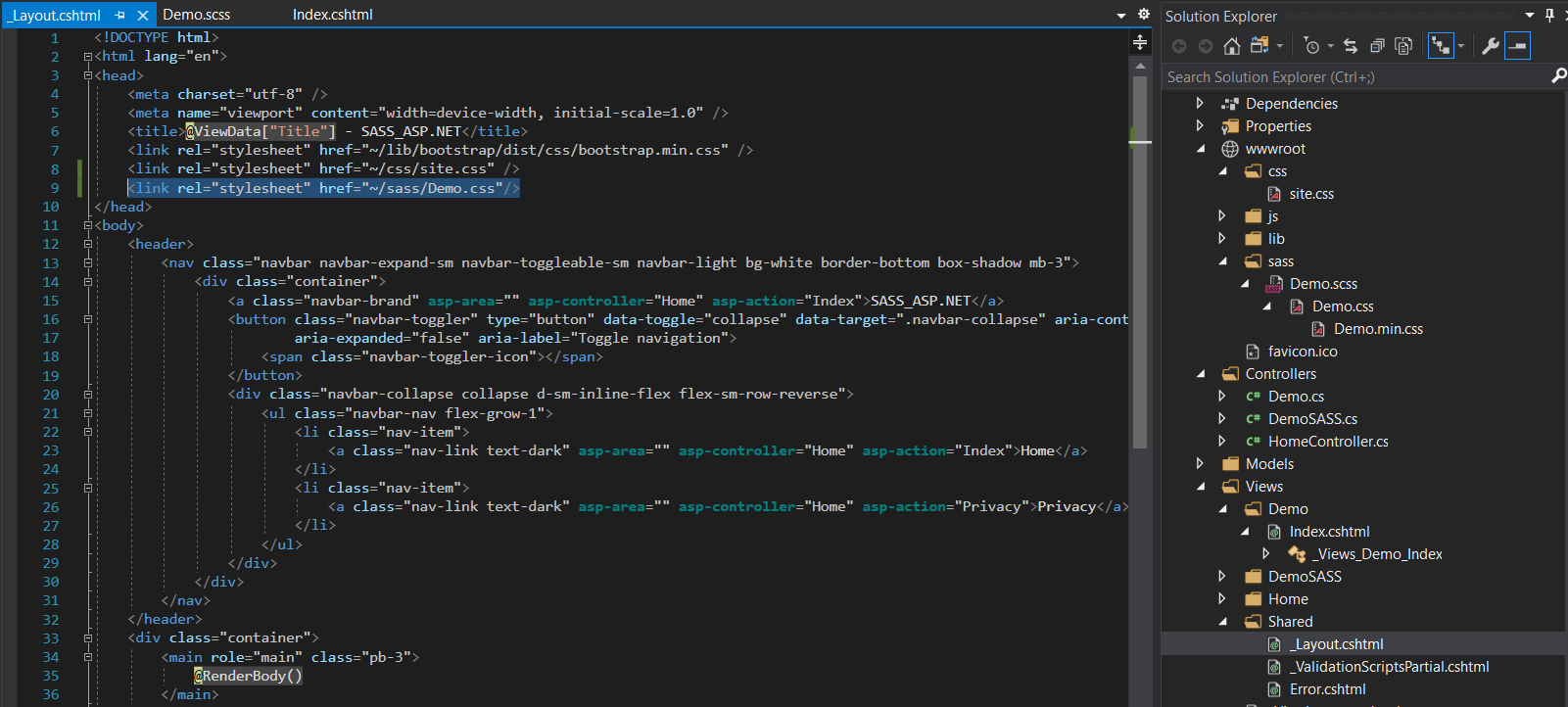
1. Compile .scss 🡪 .css

* Ta thu được file .css và .min.css. Extension “Web compiler” còn giúp đồng bộ hóa code scss sang css theo thời gian thực.



1. Kết quả sau khi compile

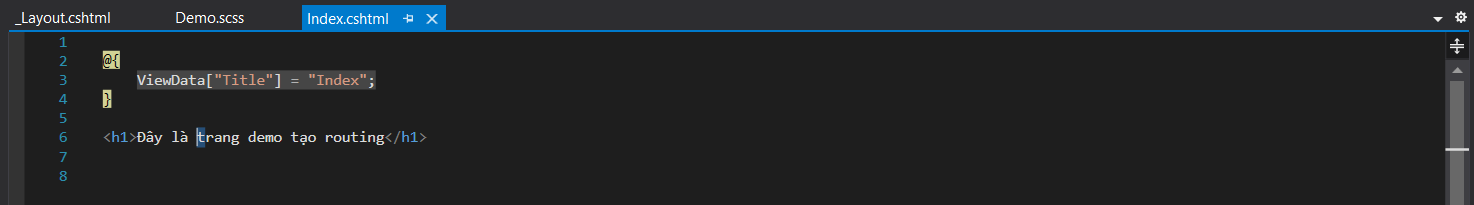
* Do ở đây trang demo của mình dùng file \_Layout.cshtml trong folder Shared. Nên ta sẽ bỏ file css vừa được complie vào trong header.



1. Thêm .css

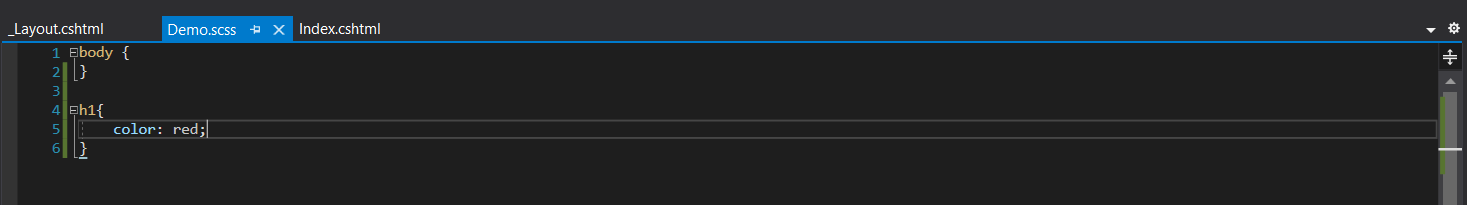
* Sau đó ta code SASS như bình thường.

*File Index.cshtml*

**

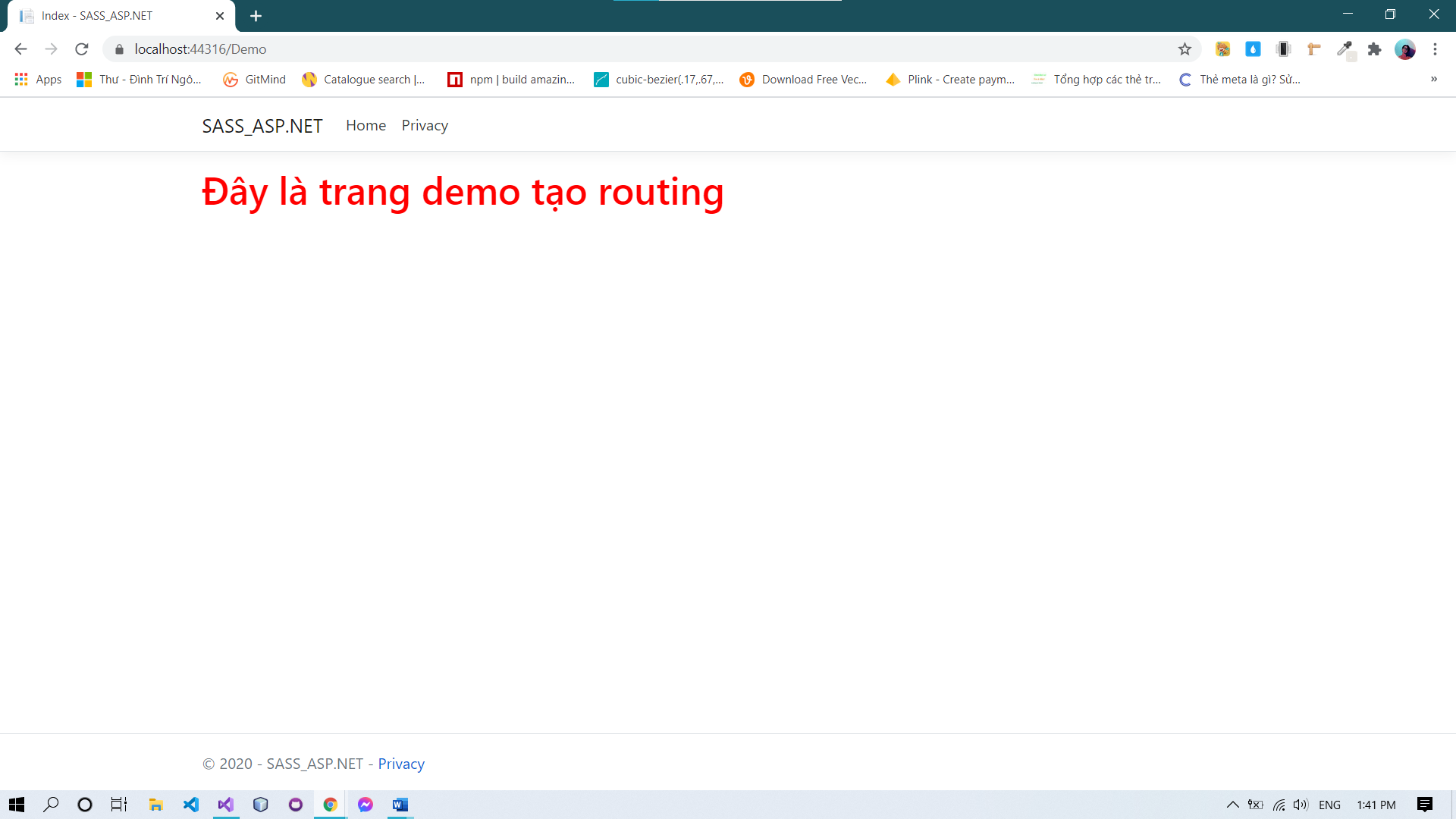
1. Index.cshtml

*File SCSS*

**

1. File SASS

*Kết quả:*

**

1. Kết quả đạt được

# Khái niệm căn bản trong Asp.net core

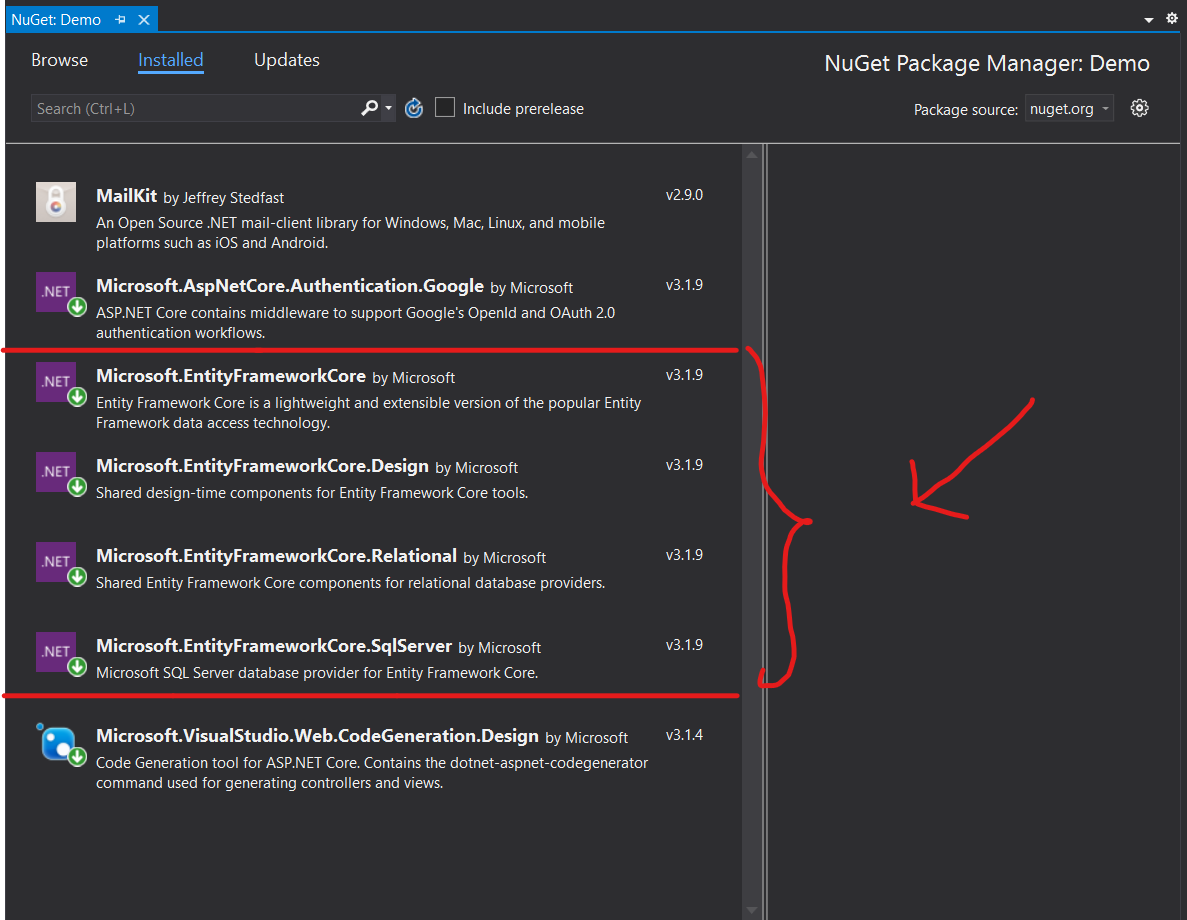
## Giới thiệu về EntityFrameworkcore

EF(Core) là một library giúp chúng ta có thể kết nối với cơ sở dữ liệu (SQL server).Với EF(Core) truy cập dữ liệu được thực hiện bằng các sử dụng model. Model được tạo thành các class giúp ánh xạ và truy xuất hoặc nhập dữ liệu vào trong database

Sau đây là các cách kết nối database bằng EF(Core)

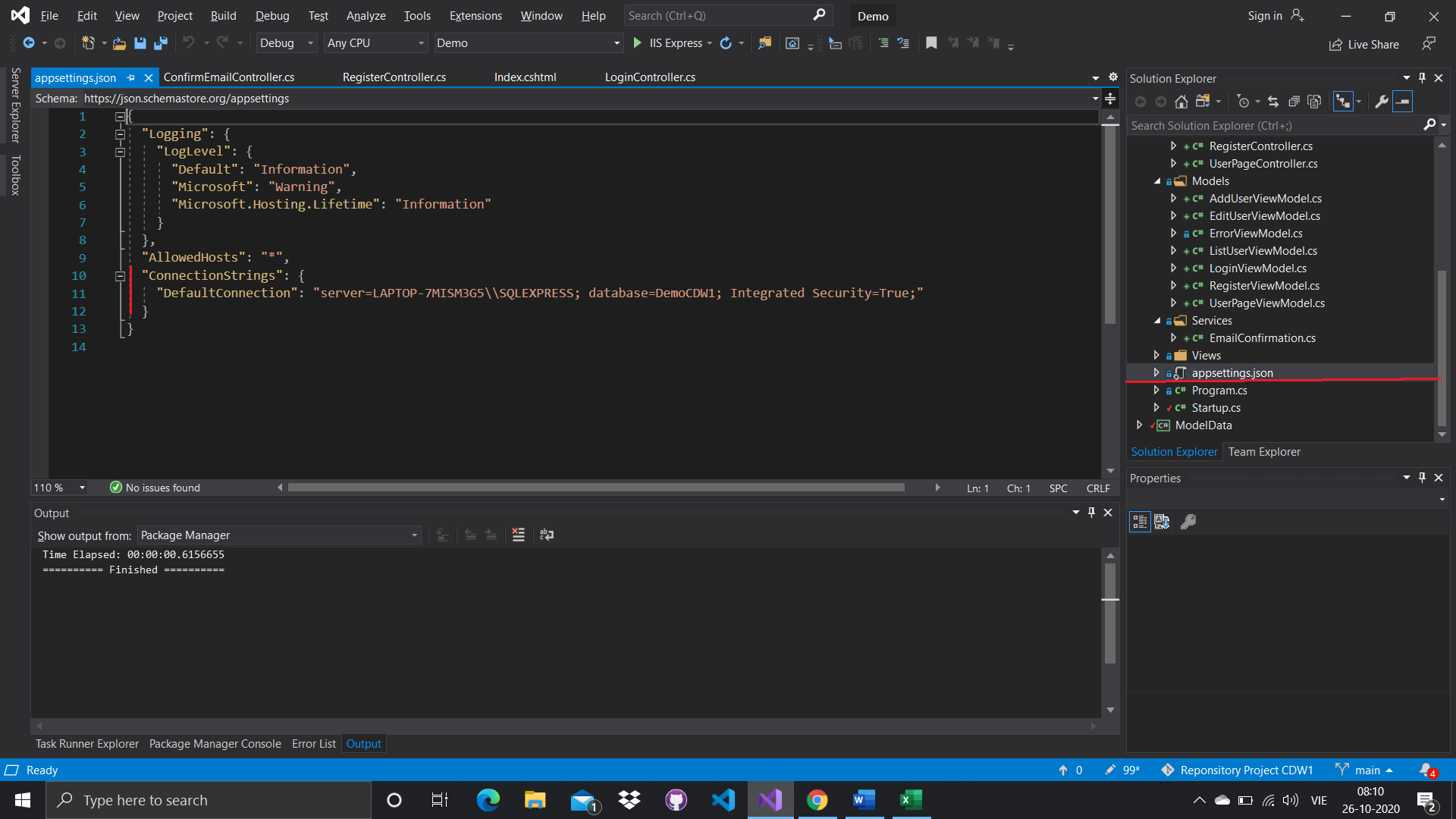
Trước tiên bạn cần cài các NuGet Package hổ trợ EF(Core). Click vào tên

Project ở ô Solution Explorer 🡪 Chọn Manage NuGet Package. Sau đó cài các NuGet Package sau:



1. NuGet Package hổ trợ EF(Core)

**B1:** Mở project của bạn sau đó truy cập vào file appsetting.json



1. Cấu hình kết nối với Database

* Khởi tạo chuỗi json như trên
* server = “something”

Truy cập vào cơ sở dữ liệu SQL Server và lấy địa chỉ của database

* database = “something”

Tên database cần phải truy cập

* Integrated Security = true

Mở tính năng bảo mật cho database

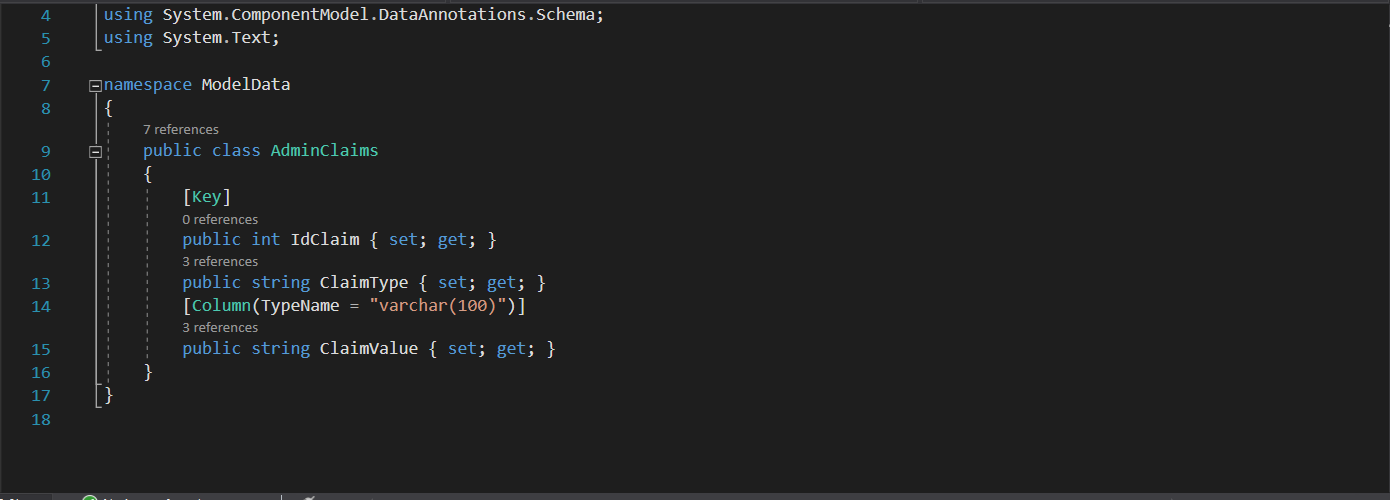
**B2:** Tạo ra một class tên là ApplicationDbContext dùng để ảnh xạ database thông qua Dbcontext



1. Ánh xạ Database qua Dbcontext

B3: Tạo một đối tượng để anh xạ, ở đây ta tạo adminClaim

Có các primarykey như IdClaim và loại varchar(100)



1. Đối tượng ánh xạ

## Giới thiệu về Migration

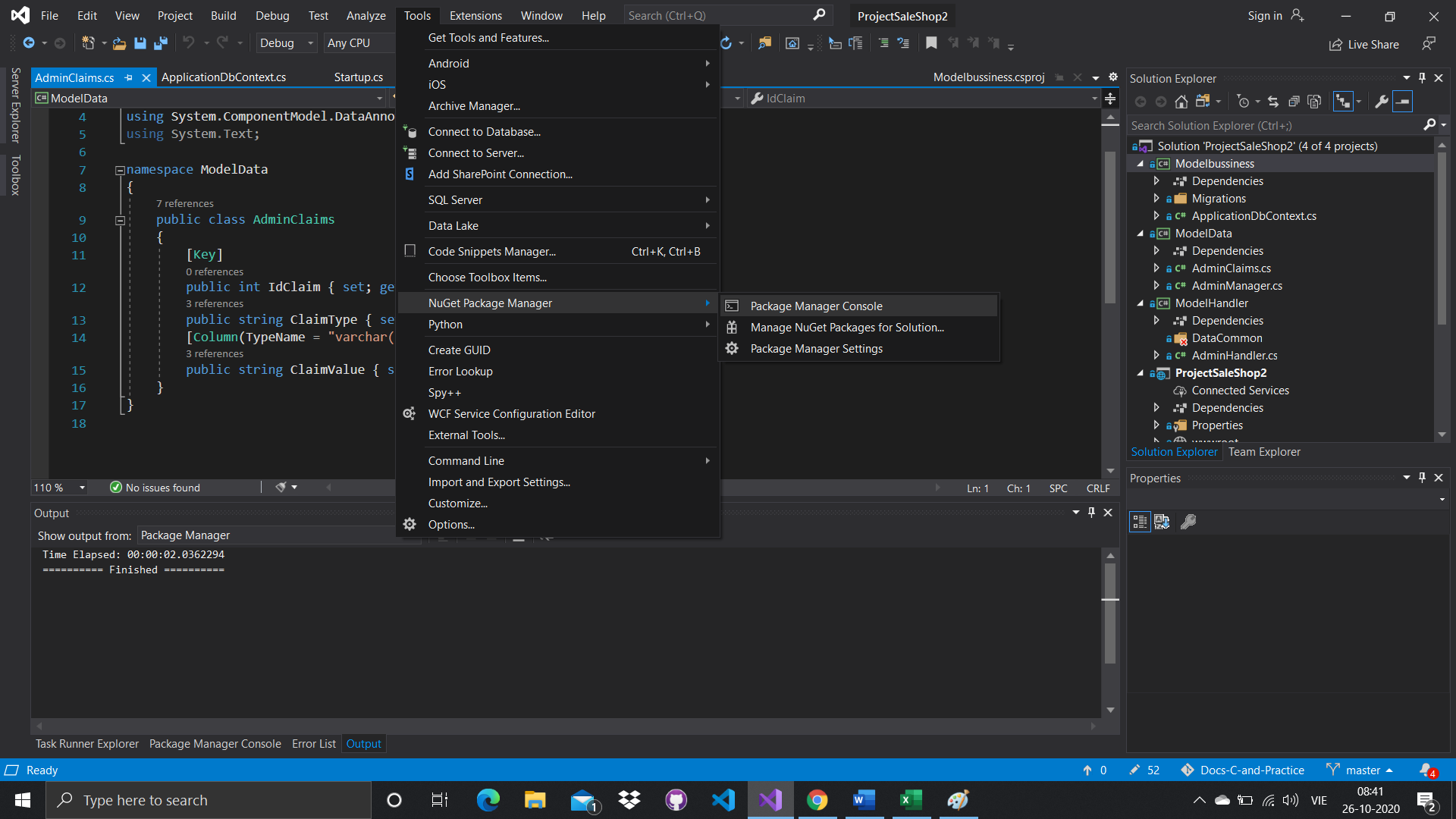
Migration là cơ chế đặc biệt trong Entity ,cho phép cập nhật các model class vào database nếu có 1 class nào bị thay đổi

* Để bắt đầu vế Migration hãy cài đặt NuGet Package:



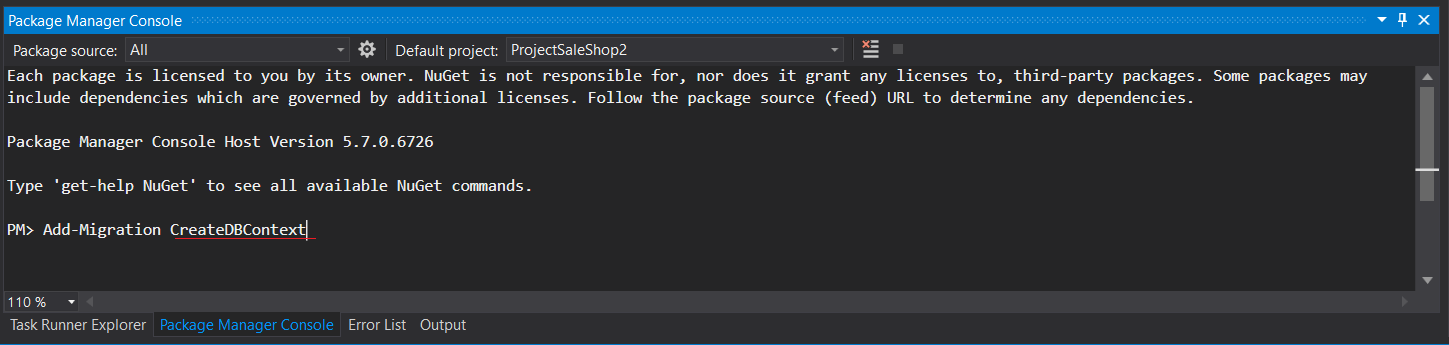
1. Microsoft.EnityFrameworkCore.Tools

Sau khi thực hiện các bước trên Entity(Core) ở trên hãy mở Package Manager Console như sau:



1. Tools 🡪 NuGet Package Manager 🡪 Package Manager Console

* Sau đó nhập

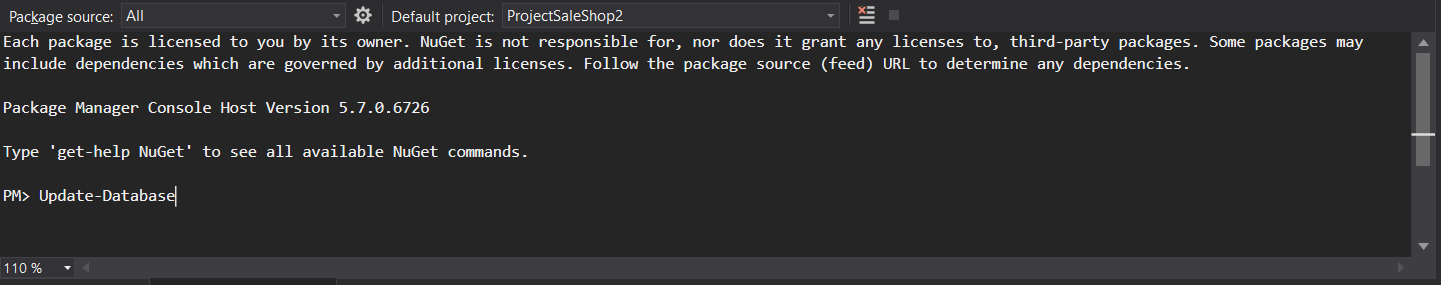


1. Thêm Migration

* Ở dấu gặt đỏ nó là tên của migrate

VD: Bạn có class Product có các properties là id,price

Sau đó bạn thêm name vào trong product, Khi them xong để update lên database ta dùng lệnh add và đặt tên là AddNameToProductTable



1. Cập nhật lên Database

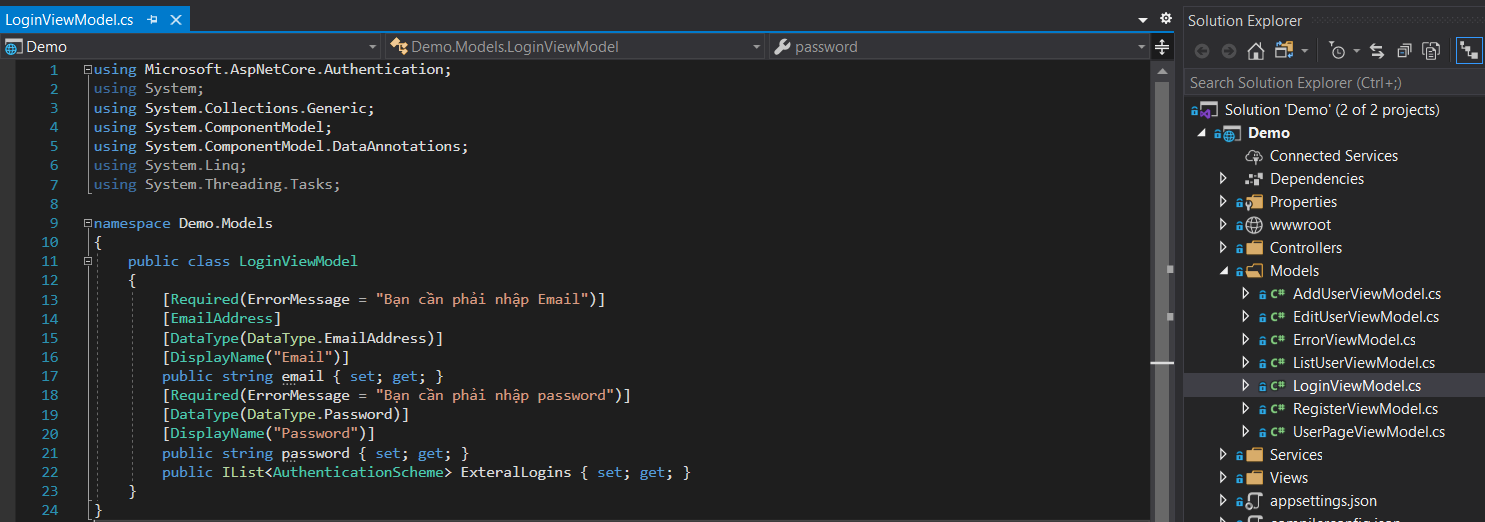
Dùng lệnh Update-Database để đưa lên SQL

## Giới thiệu về submit form

Submit form: Khi bạn muốn client gửi lên các dữ liệu của họ lên server, chúng ta dùng thẻ form trên view để đưa về cho controller

* Để submit form ta hãy làm như sau

**B1:** Tạo LoginViewmodel trong folder Models



1. Login Viewmodel

* Trong model có email và password
* Tạo thêm 1 controller tên Login



1. Login Controller

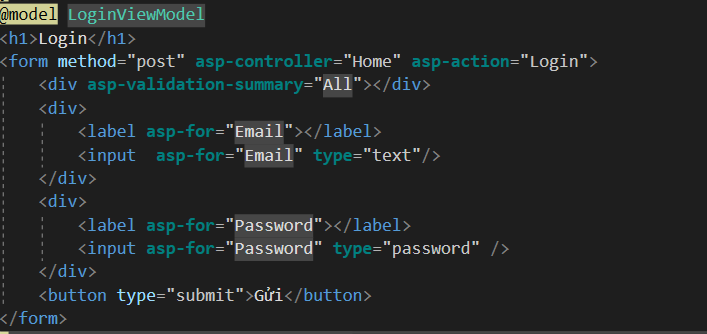
* Với HttpGet Được xem như url đầu vào của Login

Method :get

* Với HttpPost được xem như url có dữ liệu đưa vào

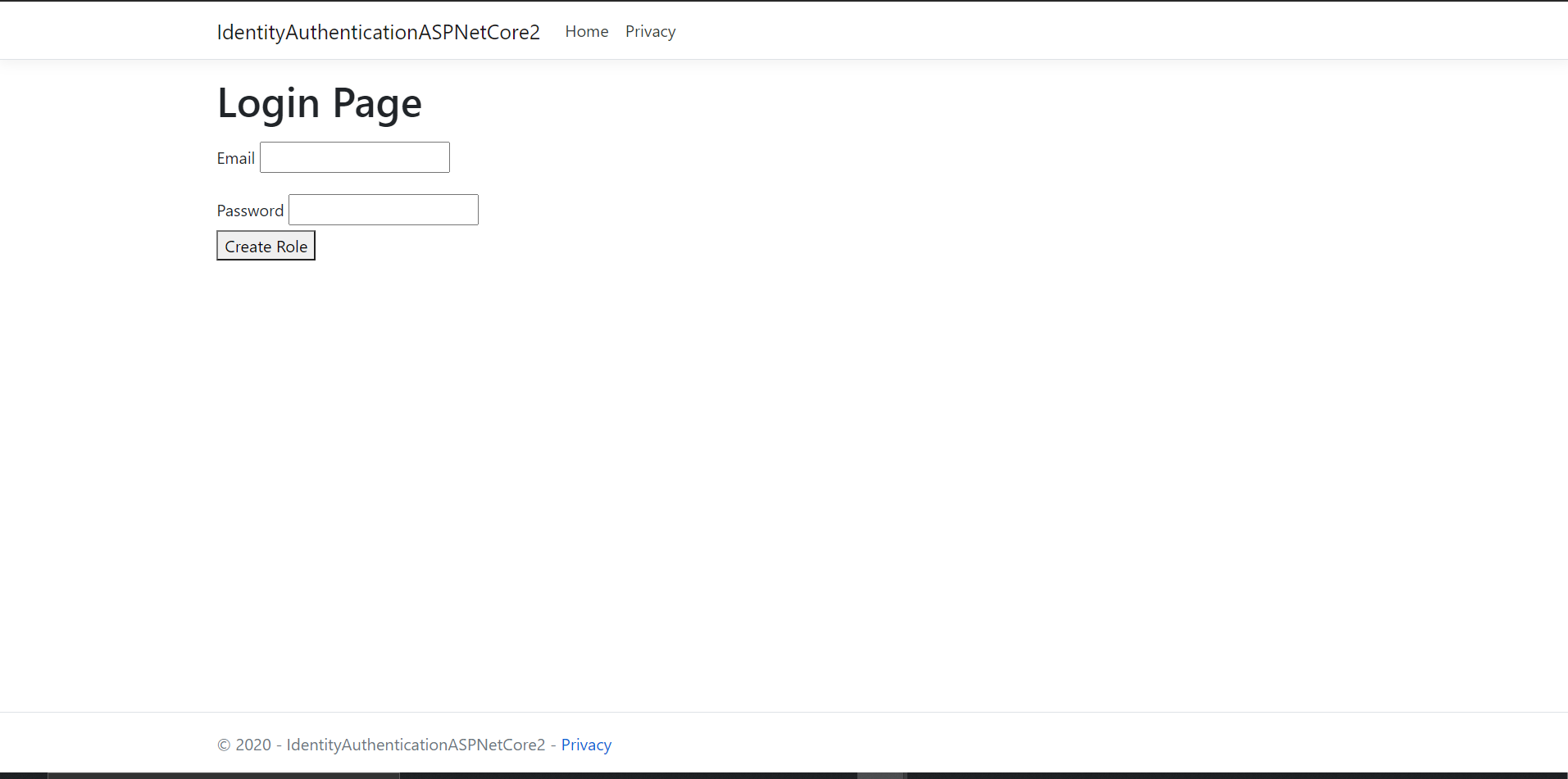
Method:post

Bước tiếp theo bạn hãy vào trong view Login và add thông tin ở dưới vào



1. Tạo form trong Login.cshtml

* Run project



1. Trang login sau khi chạy

* Sau đó nhập thông tin vào ta được kết quả như sau



1. Kết quả khi nhập vào

## Giới thiệu về truy vấn dữ liệu bằng LINQ

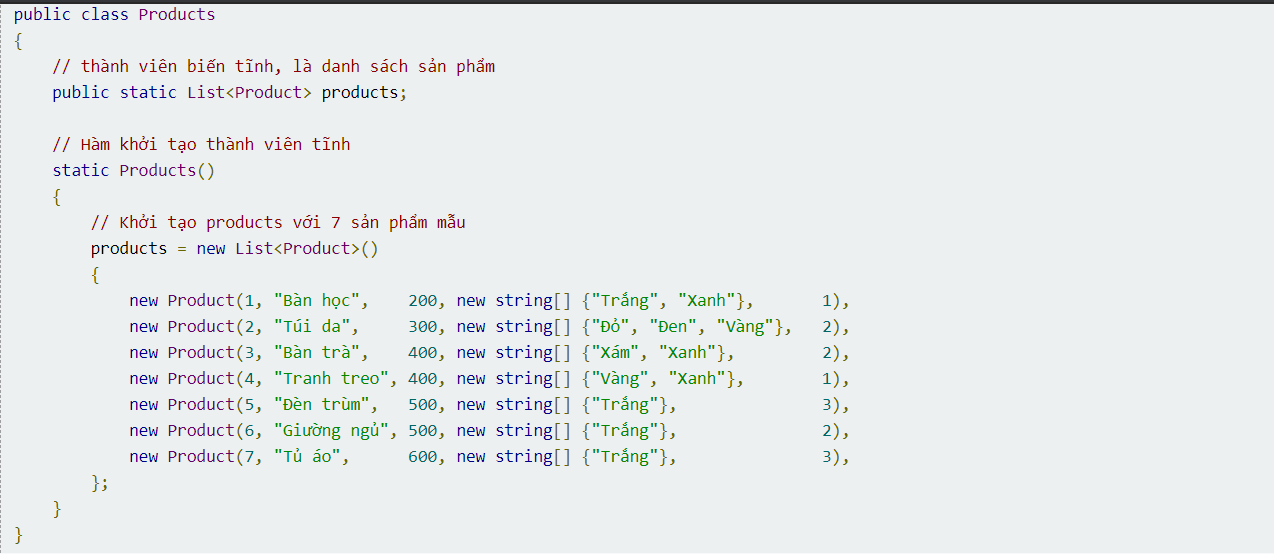
Ngôn ngữ truy vấn LINQ cho phép truy vấn,chuyển truy vấn thành các câu lệnh trên đối tượng để truy vấn vào Database

* VD truy vấn LINQ:

Tạo một đối tượng tên là Product



1. Đối tượng product viết bằng LINQ



1. Thêm sản phẩm



1. Demo

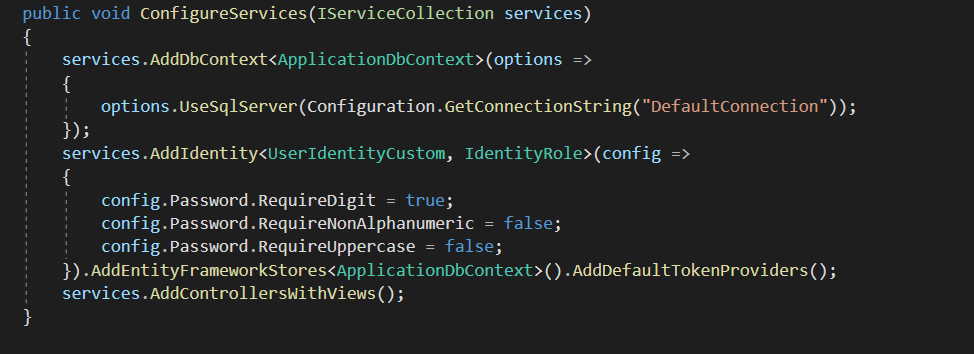
Tham khảo tại [Link](https://docs.microsoft.com/en-us/dotnet/api/system.linq?view=netcore-3.1)

## Giới thiệu về Identity Authentication

Identity Authentication là là model (buil-in) nó cung cấp cho người dùng các tính năng đầy đủ và đa dạng về authentication và authorization, cho phép tạo tài khoản, đăng xuất cũng như là thêm, xóa user... Hỗ trợ tích hợp với external login bên ngoài như Google, Facebook, Twitter... Và được lưu vào hệ sinh thái.

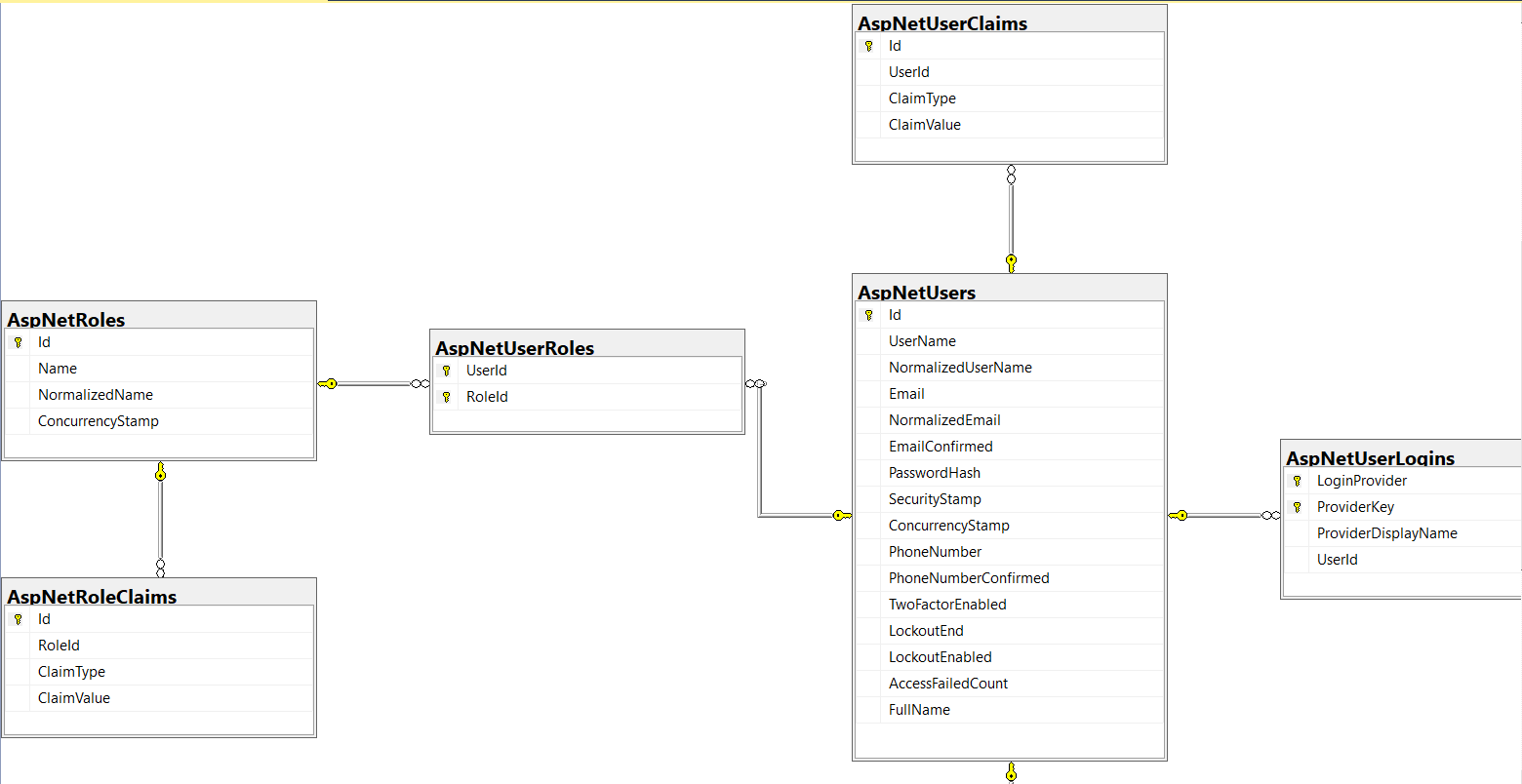
## Hướng dẫn thiết lập Identity

B1: Vào startup.css



1. custom mật khẩu cho phía identity

* Khi này bạn có thể sử dụng identity authentication, ở config => Đây là nơi bạn có thể custom mật khẩu cho phía identity
* Sau khi thêm bước này dùng migration để update-database



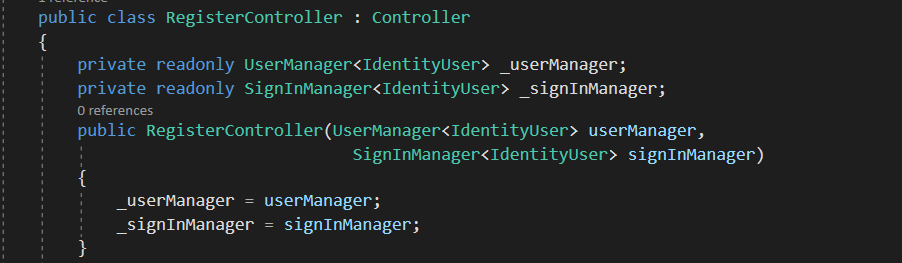
1. Mô hình database sau khi thiết lập identity

* Trong database, identity đã phân user role và thông tin cho người dùng lưu trữ

## Giới thiệu về UserIdentity và User Manager

* UserIdentity : Dùng để lưu trữ thông tin người dùng
* UserManager: Dùng để khởi tạo người dùng trong database, Hãy coi một số vd sau

B1:Khởi tạo trong Register controller



1. Khởi tạo đối tượng quản ly người dùng và đăng nhập

* Khi này bạn có thể dùng \_userManager để thêm người dùng

B2:



1. Khởi tạo thông tin usertrong database và đăng nhập

\_userManager.CreateAsync ()

* Khởi tạo một user trong Database,Khi người dùng gửi request kèm thông tin, lúc này trong database sẽ lập tức thêm vào trong ASPUser table trong database

## Giới thiệu về RoleIdentity và Role Manager

**RoleIdentity**: Dùng để thiết lập thông tin role có trong ứng dụng

Nếu bạn muốn ứng dụng có những trang admin mà người có vai trò admin vào dc thì bạn có thể dùng nó bằng [Authorization(Roles = “something”)]

**RoleManager**: Dùng để quản lý role, thiết lập và khởi tạo role vào trong database

vd sau

* Vì đây là phần khó, và càng khó hiểu cho với các bạn mới bắt đầu
* Đây là các đường dẫn liên quan các bạn có thể tham khảo

<https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.identity.rolemanager-1?view=aspnetcore-3.1>

<https://www.yogihosting.com/aspnet-core-identity-roles/#all-roles>

## Giới thiệu về SigninManager

* Dùng để kiểm tra người dùng có trong database hay ko, mặt khác nó có thể tạo ra cookie khi nhận biết ng dùng có trong database
* Vd:

B1 vào trong loginController

 public class LoginController : Controller

    {

        private readonly UserManager<IdentityUser> \_userManager;

        private readonly SignInManager<IdentityUser> \_signInManager;

        public LoginController(UserManager<IdentityUser> userManager,

                                    SignInManager<IdentityUser> signInManager)

        {

            \_userManager = userManager;

            \_signInManager = signInManager;

        }

        [HttpGet]

        public IActionResult Index()

        {

            return View();

        }

        [HttpPost]

        public async Task<IActionResult> Index(LoginViewModel model)

        {

            if (ModelState.IsValid)

            {

                var result = await \_signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);

                if (result.Succeeded) // thành công

                {

                    return RedirectToAction("index", "home");

                }

                    ModelState.AddModelError("", "Invalid Login Attempt");

            }

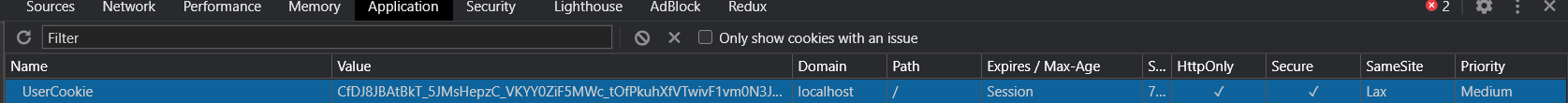
            return View();

        }

    }

1. Kiểm tra password của user đăng nhập có phù hợp trong database

await \_signInManager.PasswordSignInAsync Giúp kiểm tra và tạo cookie.



1. Nếu thành công tạo cookie

# Khái niệm nâng cao trong Asp.net core

## Hướng dẫn gửi mail MailKit

B1: Dowload <https://www.nuget.org/packages/MailKit/> Mailkit package

B2:Truy cập vào <https://www.hostinger.vn/huong-dan/lam-the-nao-de-su-dung-google-smtp-server-mien-phi/>. Đây là cách đăng ký một smtp server để bạn có thể gửi email và các dịch vụ khác

B3: 

1. Mô tả hình thức gửi message qua email

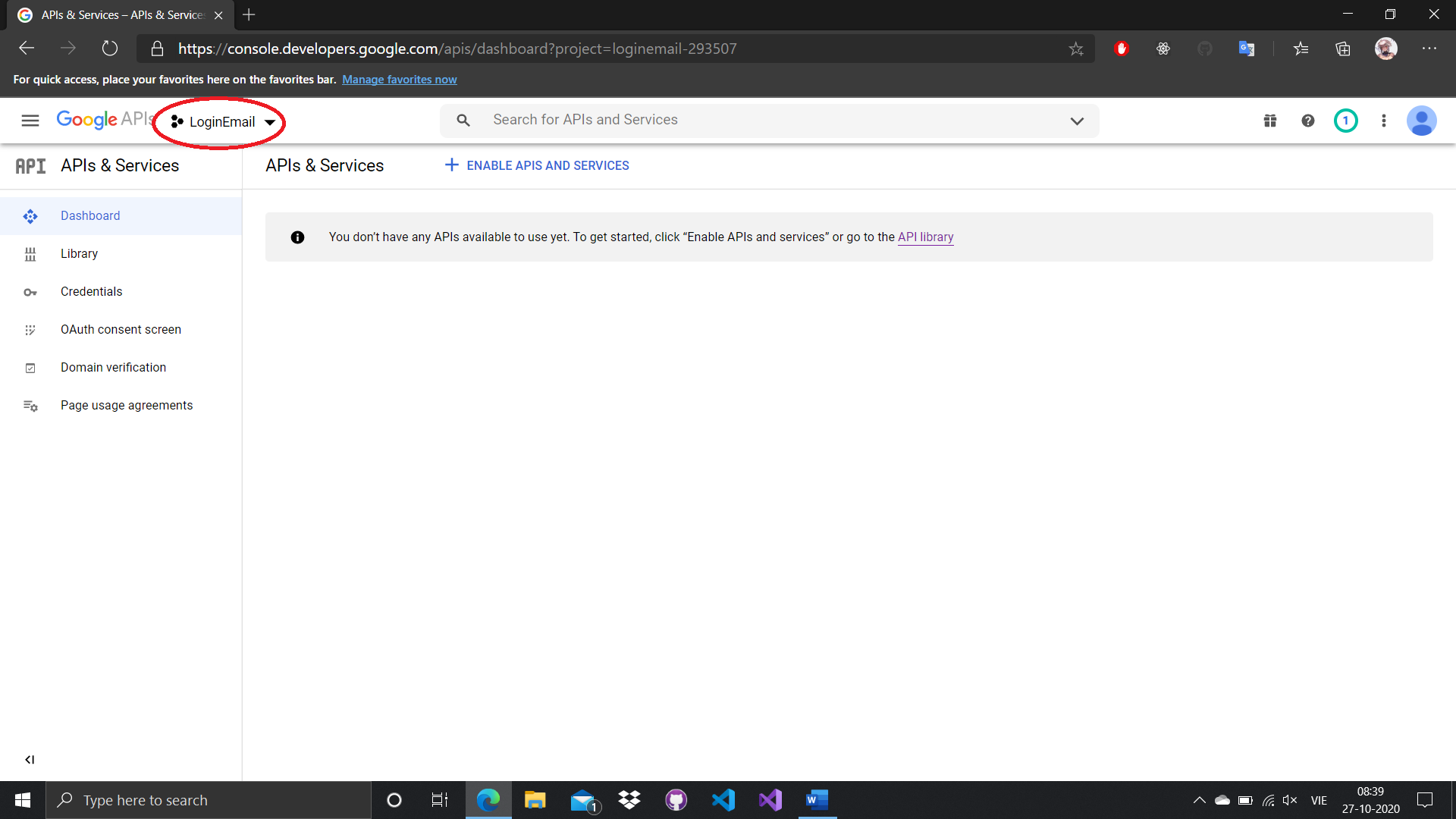
* MimeMessage => Tạo ra một đối tượng message
* From.Add() => Đây là email người gửi
* From.To()=> Đây là email người nhận
* Subject() => Tiêu đề gửi
* Body()=> Là kiểu gì, với “plain ” định dạng kiểu text văn bản
* SmtpClient()=>Khởi tạo một đối tượng smtp để gửi
* Authentication xác nhận email người gửi
* Send(đây là tin nhắn bạn gửi)
* Sau đó dùng lệnh Disconnect để kết thúc việc gửi, nếu k đóng lại sẽ gây ra lỗi

## Hướng dẫn tích hợp API login with google

* Để tích hợp API google login qua web application, phải có căn bản của Identity Authentication của ASP.NET CORE
* Biết sử dụng signinManager methods and userManager method nhé
* Bài chỉ hướng dẫn cách tích hợp ứng dụng web của bạn với phần đăng nhập bên phía google còn về phần ứng dụng desktop thì bạn nên nghiên cứu thêm
* Biết phát triển web bằng asp.net core

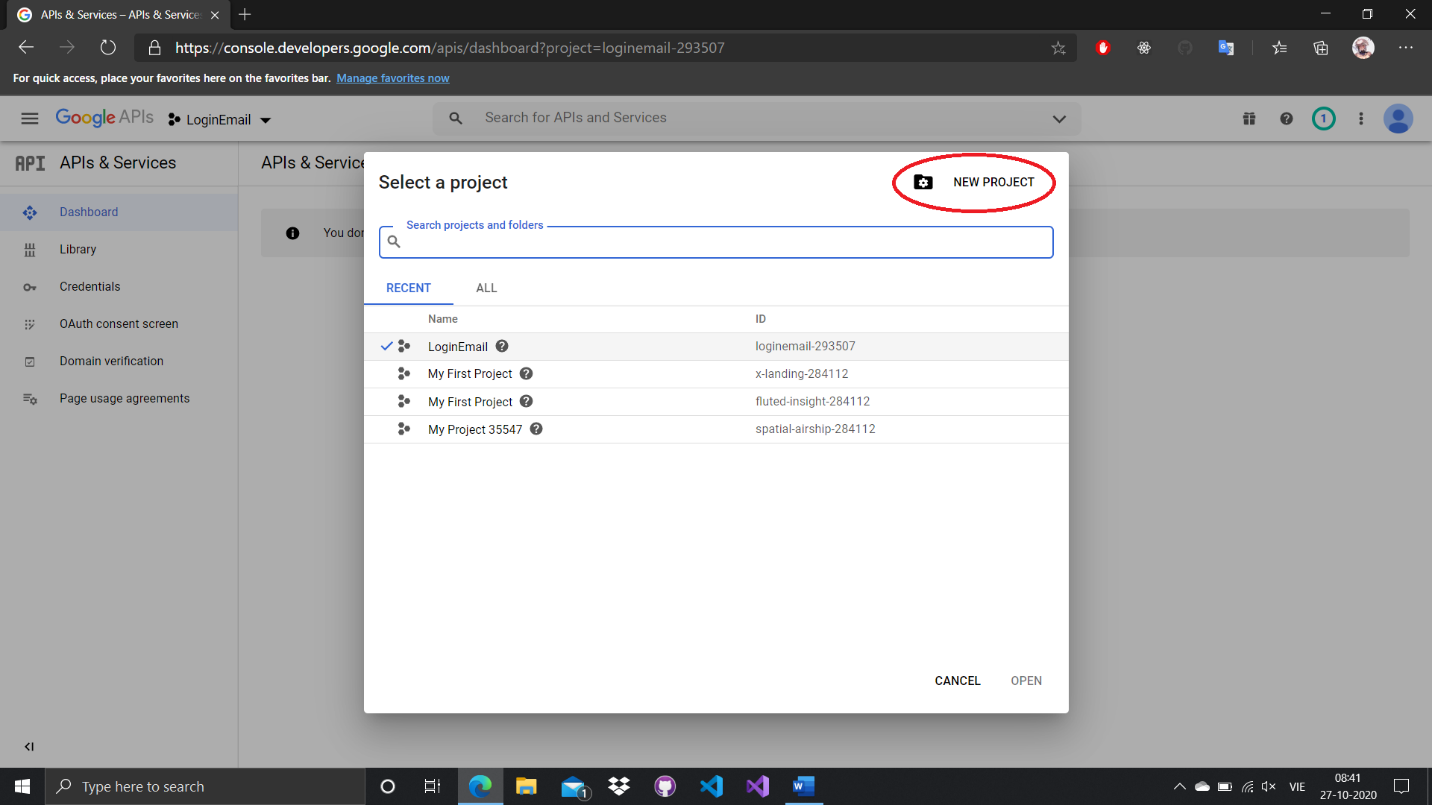
B1: Vào trang console.developer.com

Link:<https://console.developers.google.com/apis/dashboard?project=loginemail-293507>



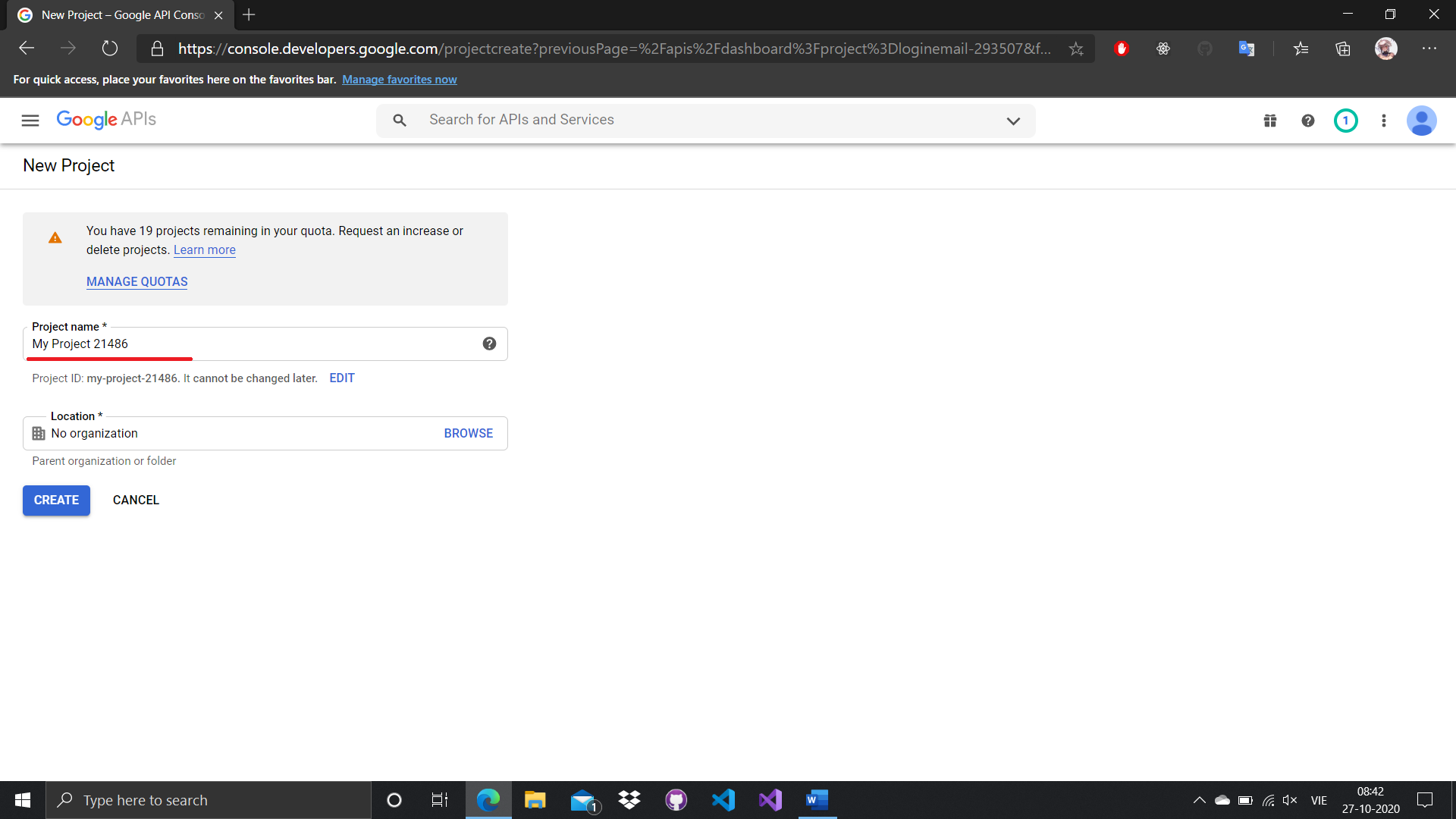
1. Vào console.developer.google.com

* Sau đó nó sẽ hiện lên bản Select Project



1. Tạo project

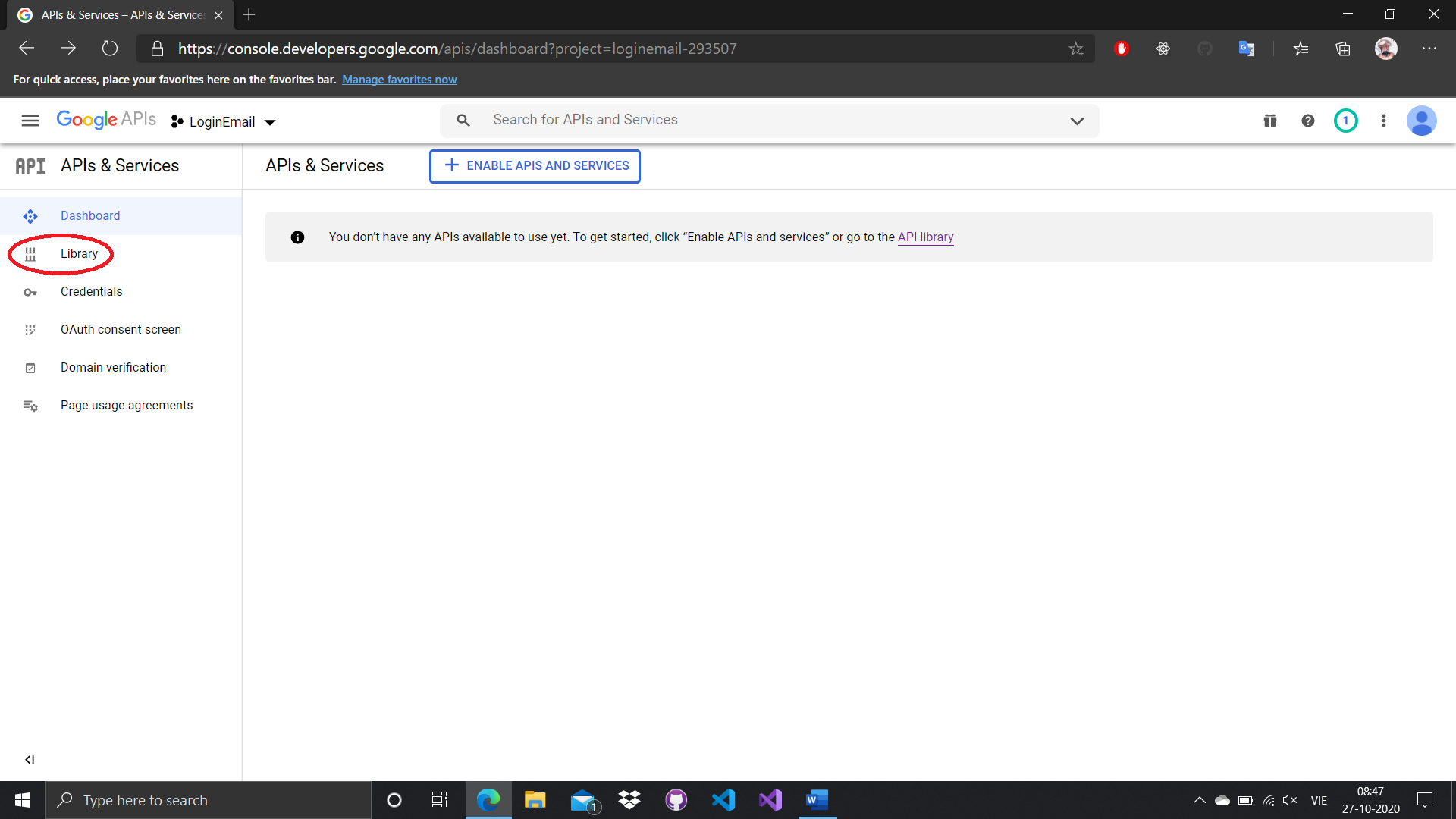
* Khi nhấn Newproject



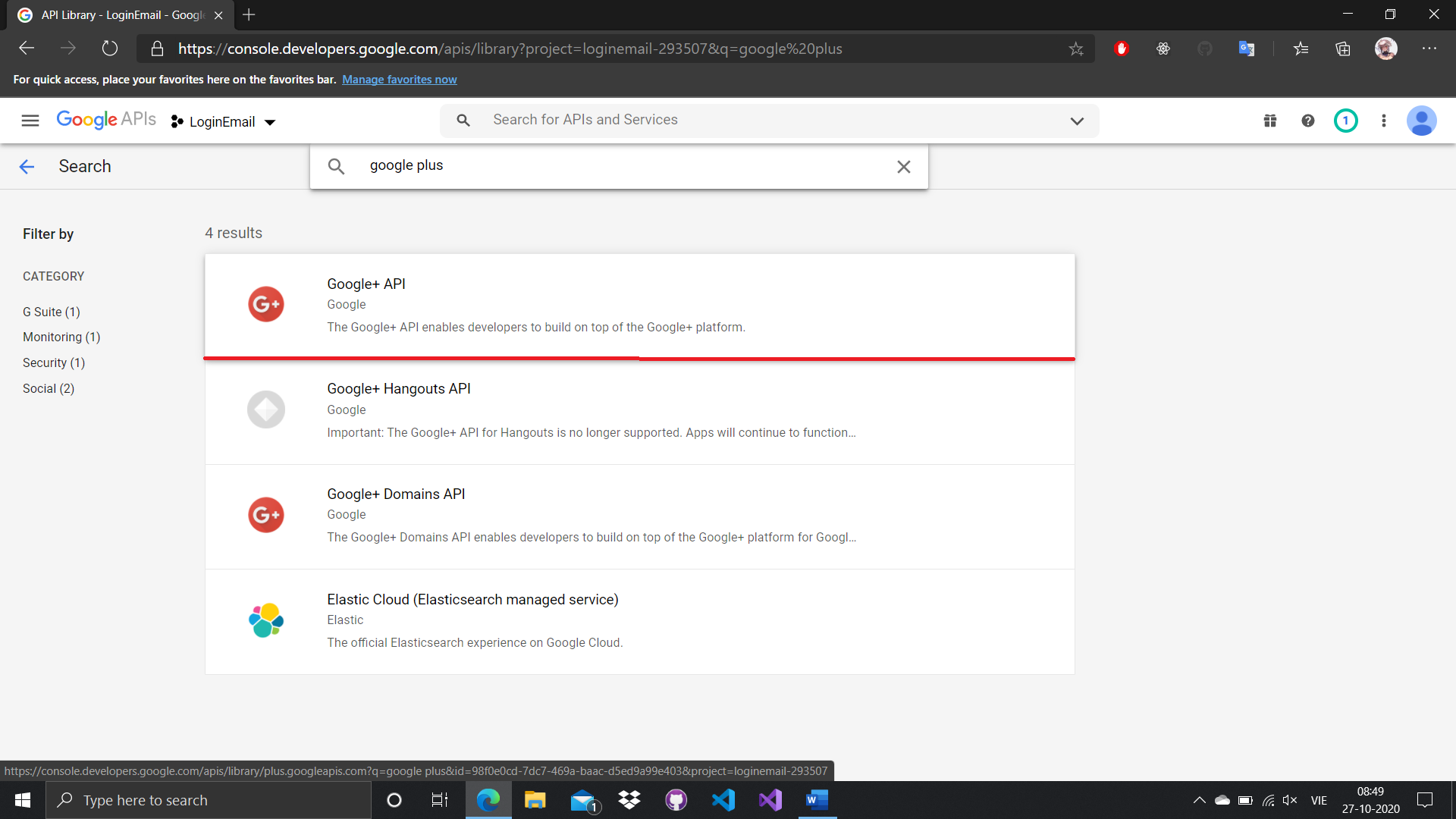
1. Cập nhật thông tin để sử dụng API

* Khi này bạn chú ý ở đây có phần điền thông tin Project name
* Ở đây chính làm tên của project bạn đã khởi tạo
* Ví dụ, khi bạn nhấn create project web nó sẽ có mục sửa tên project của bạn, lấy tên đó điền vào đây
* Sau đó nhấn CREATE

B2: Bạn quay trở lại trang chính lúc nãy và nhấn vào Library

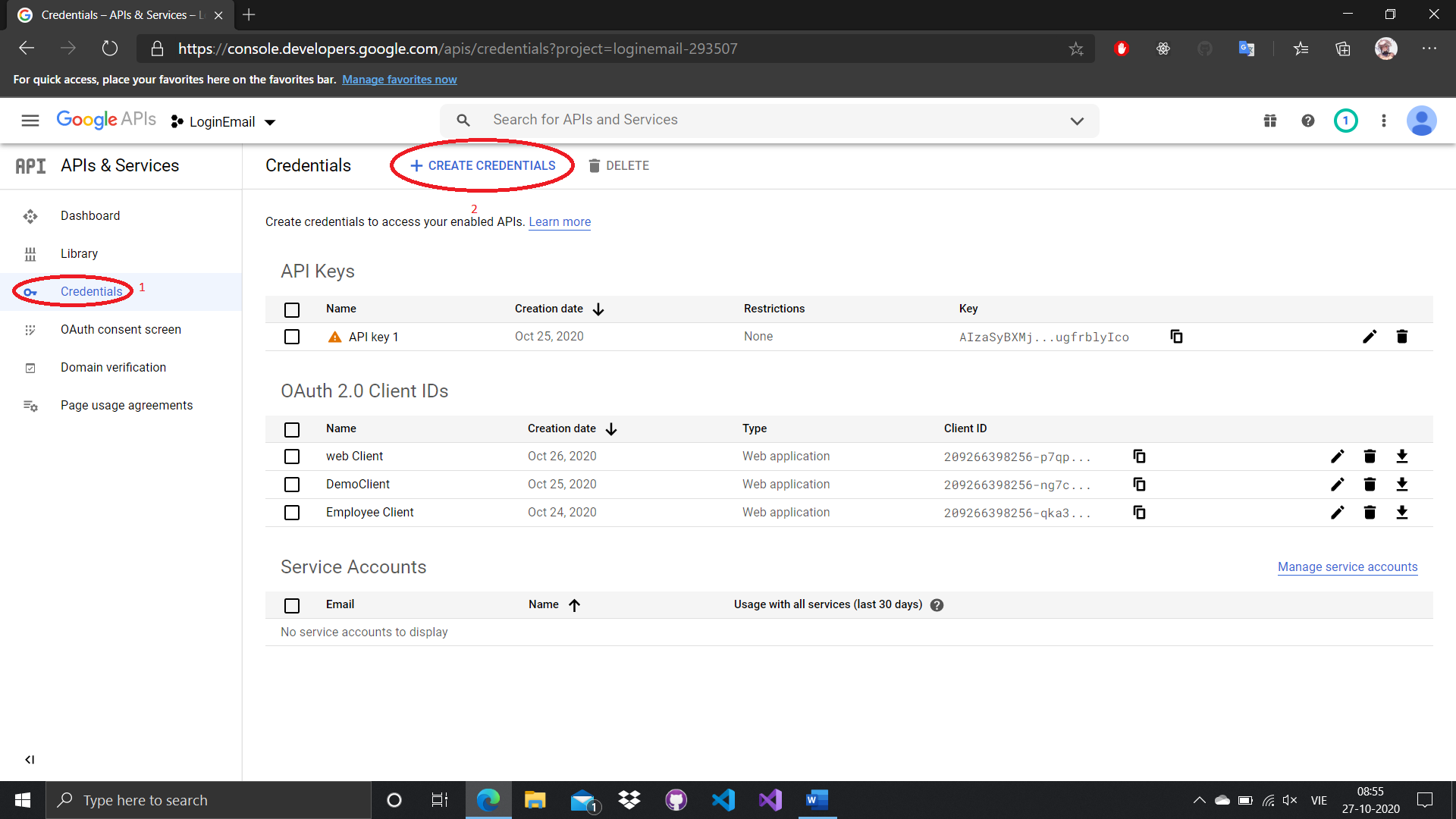


1. Tiếp tục vào library

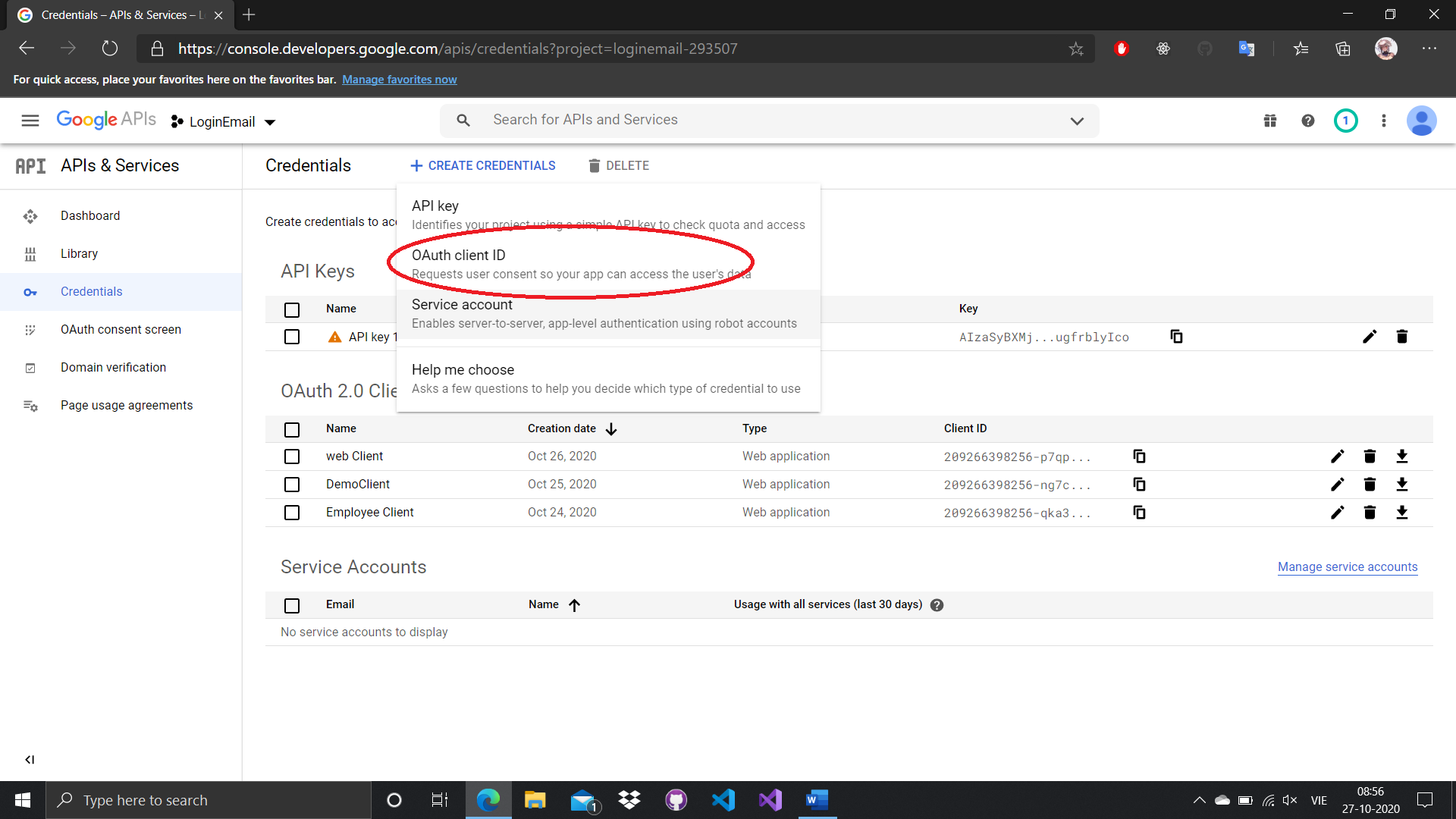


1. Tìm Google+ API

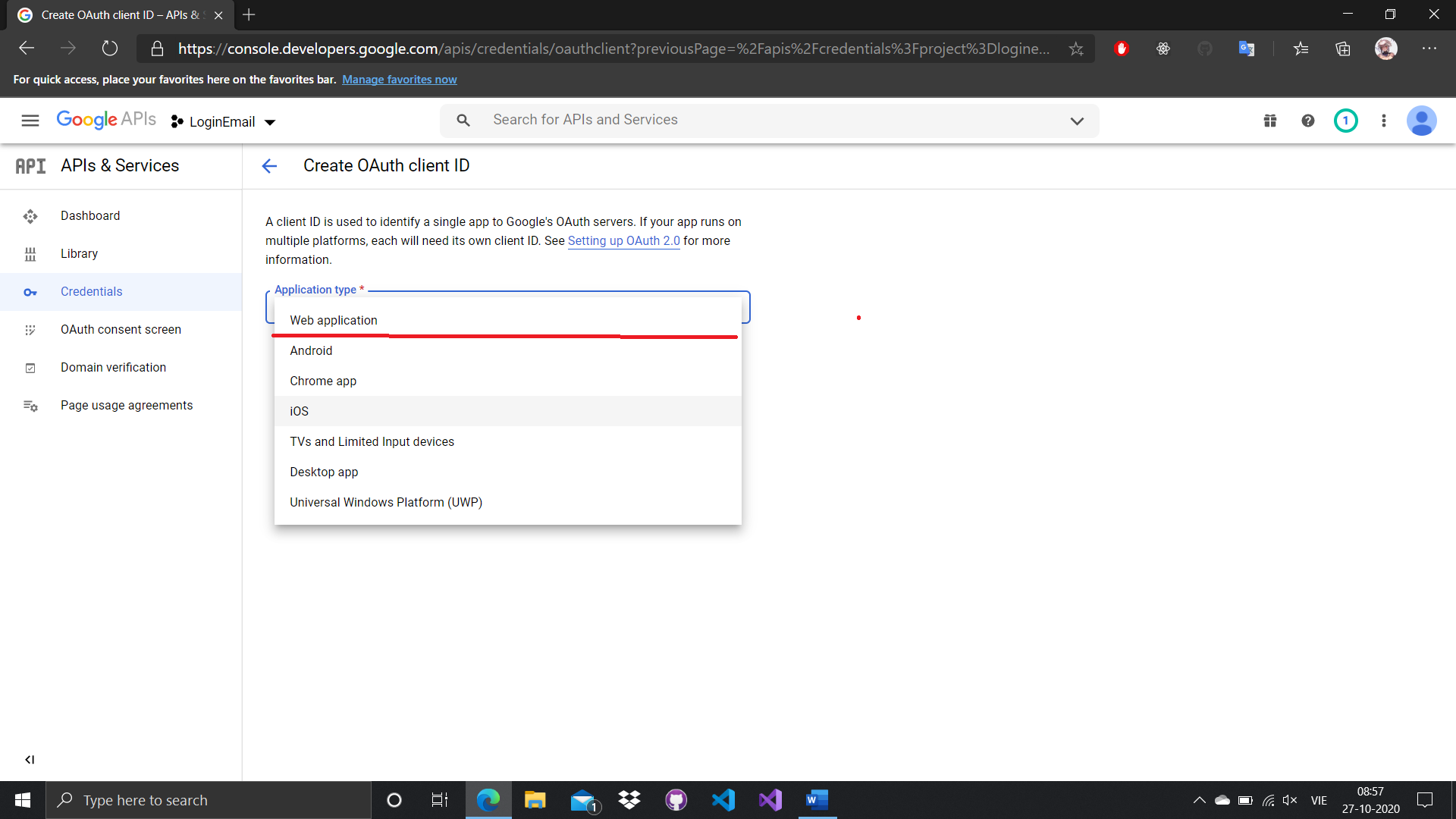
* Sau đó nhấn vào sử dụng nó



1. Khởi tạo credentials

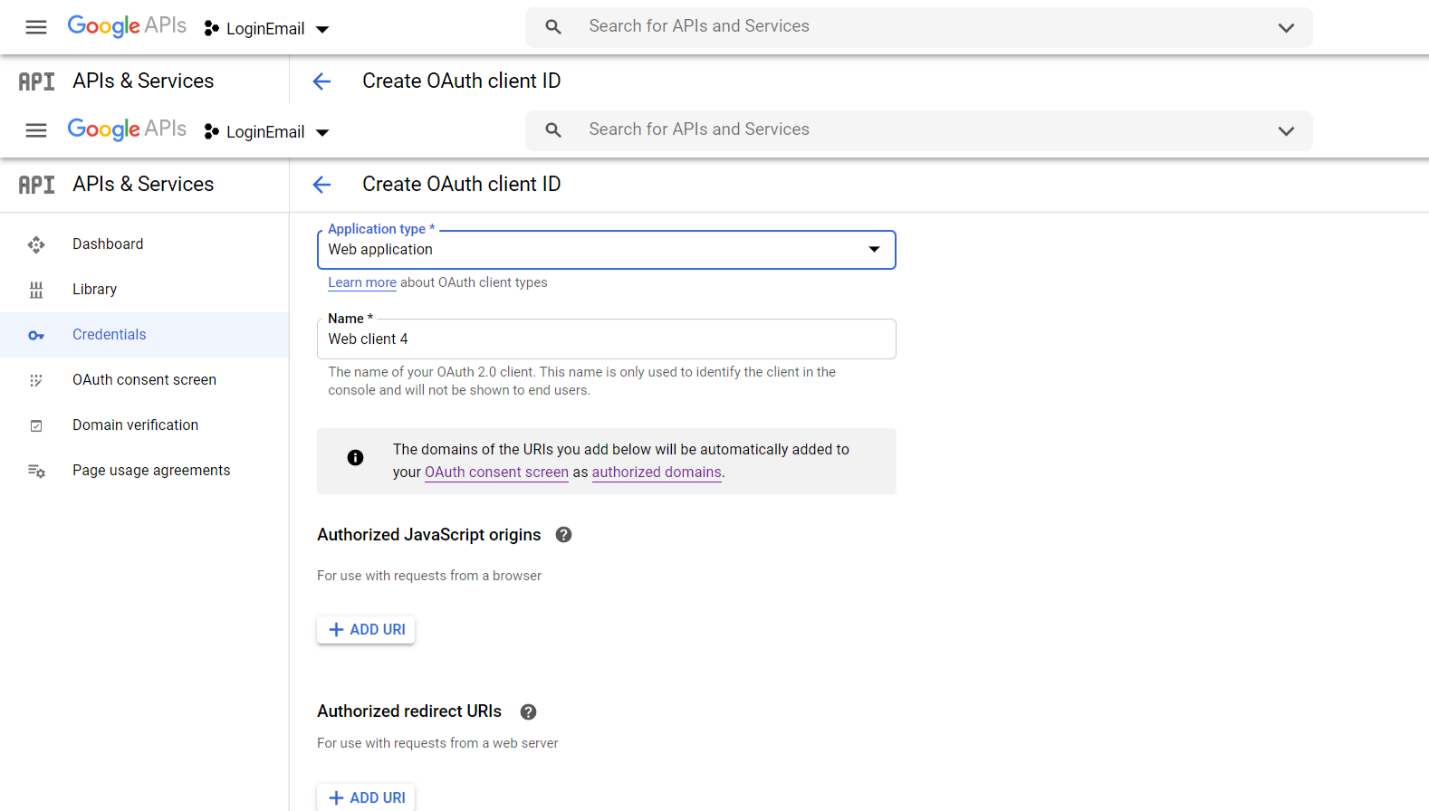


1. Vào Oauth client ID



1. Chỉnh loại công cụ để dùng API

* Ở đây có nhiều nền tảng phát triển, ở đây nhấn web application vì mình đang làm web mà :V



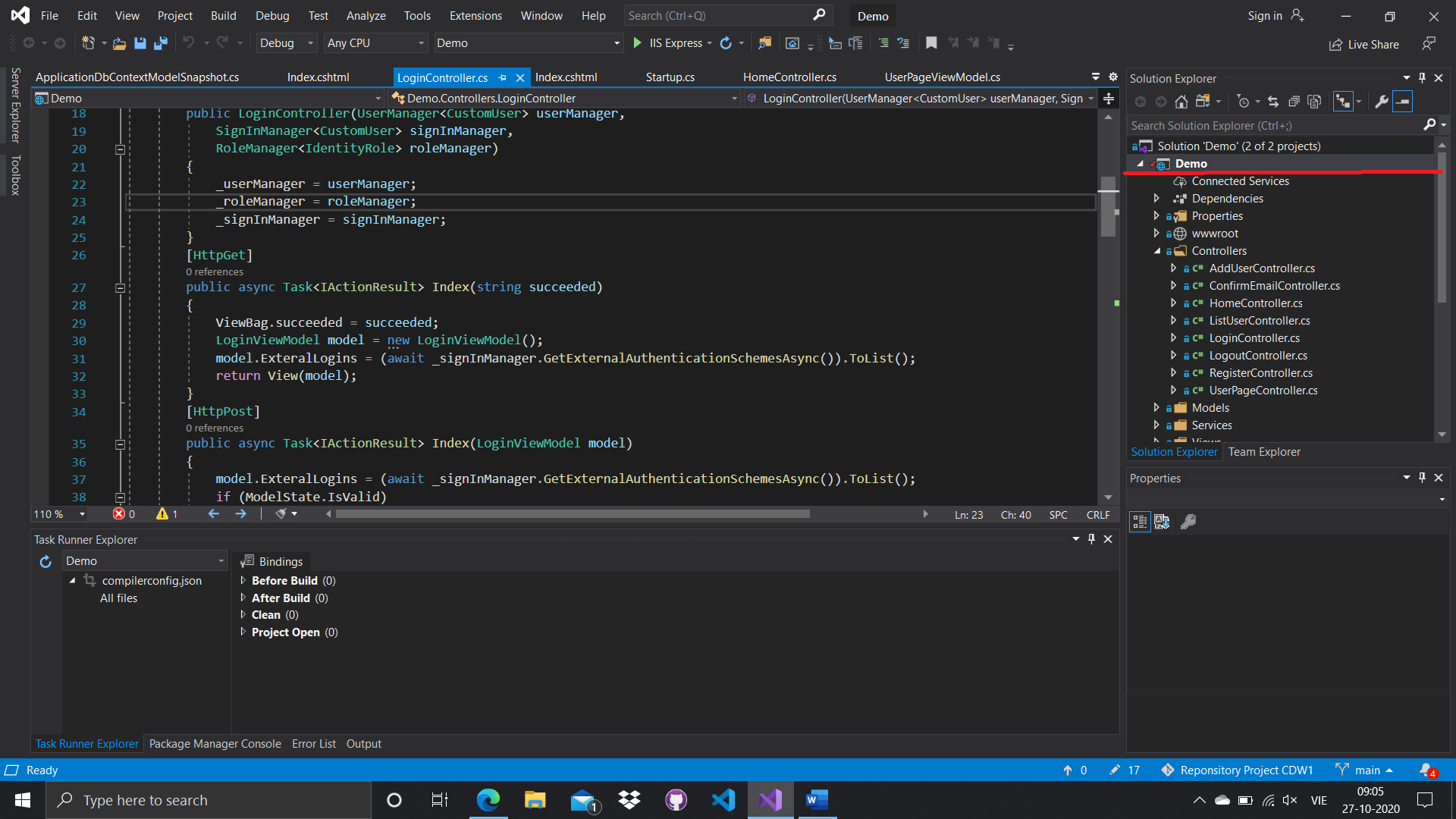
1. Đặt tên Name cho Oauth Client

Đầu tiên đặt lại tên Name: cái này là tên client k đổi cũng k sao

Nhìn xuống phía **Authorized JavaScript origins**

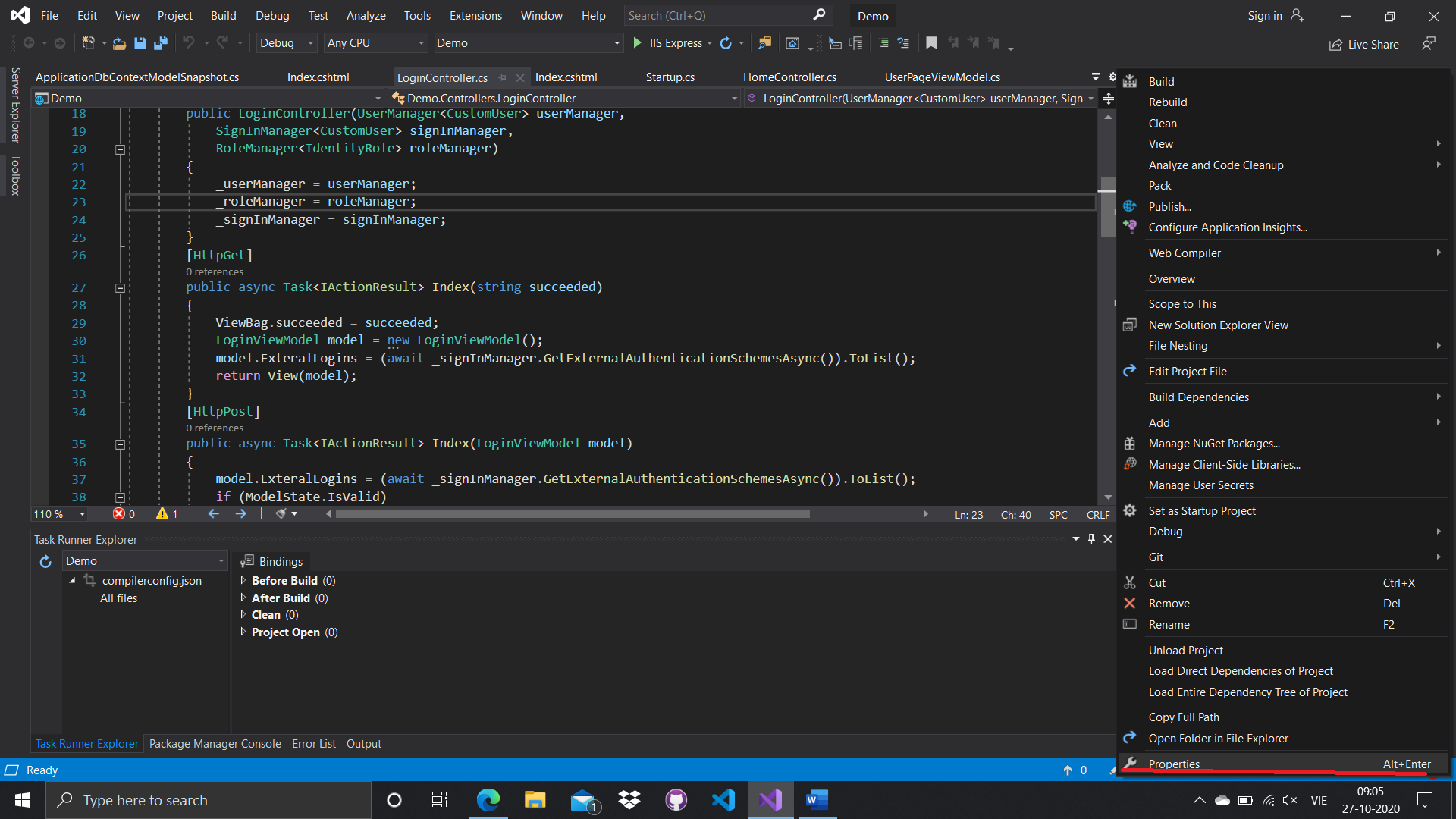
Sau đó nhấn add URL sau đó mở project của bạn lên

* ở đây mình đã tạo sẵn demo



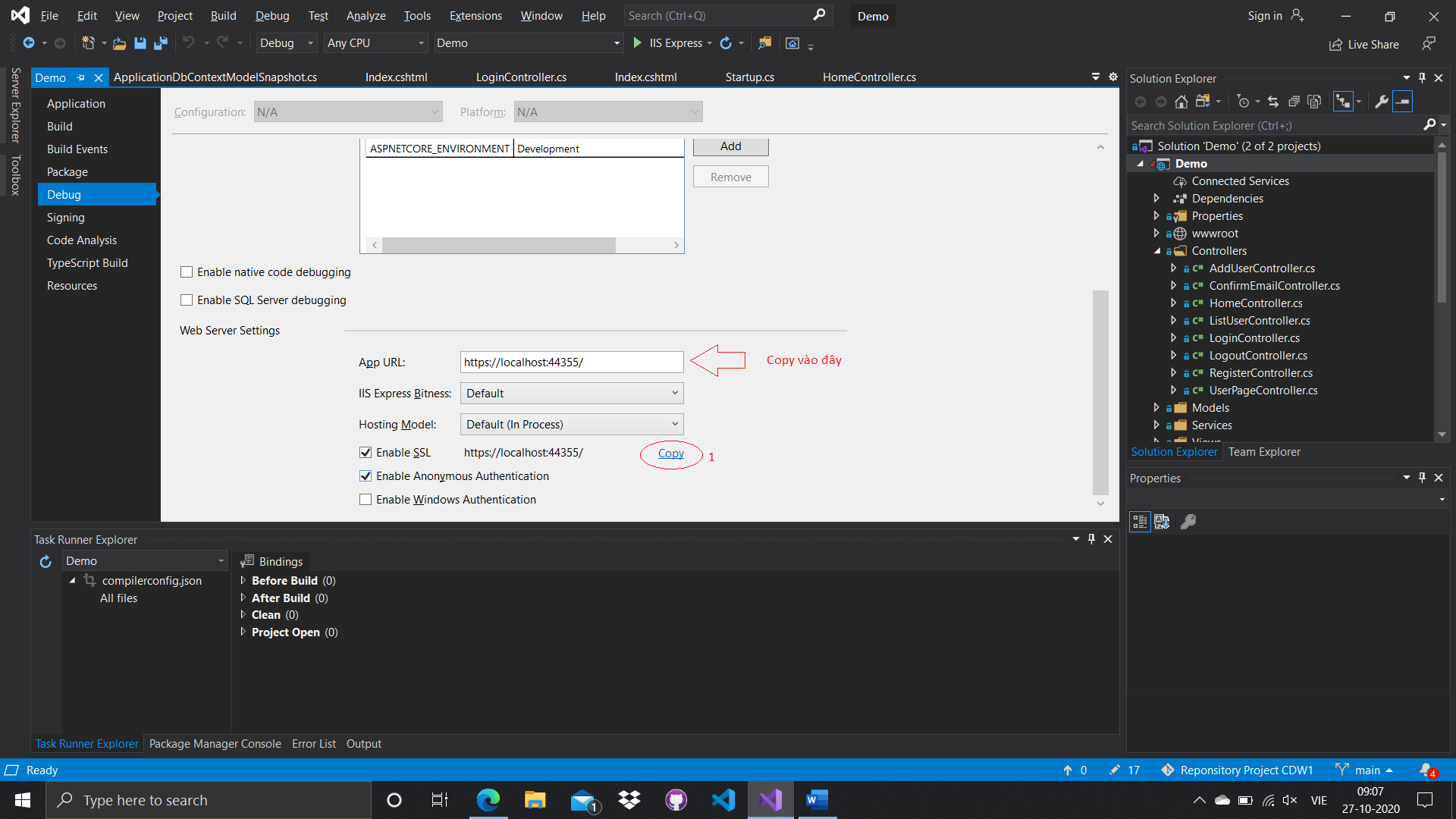
1. Chọn tên project

* Nhấn chuột phải vào tên project



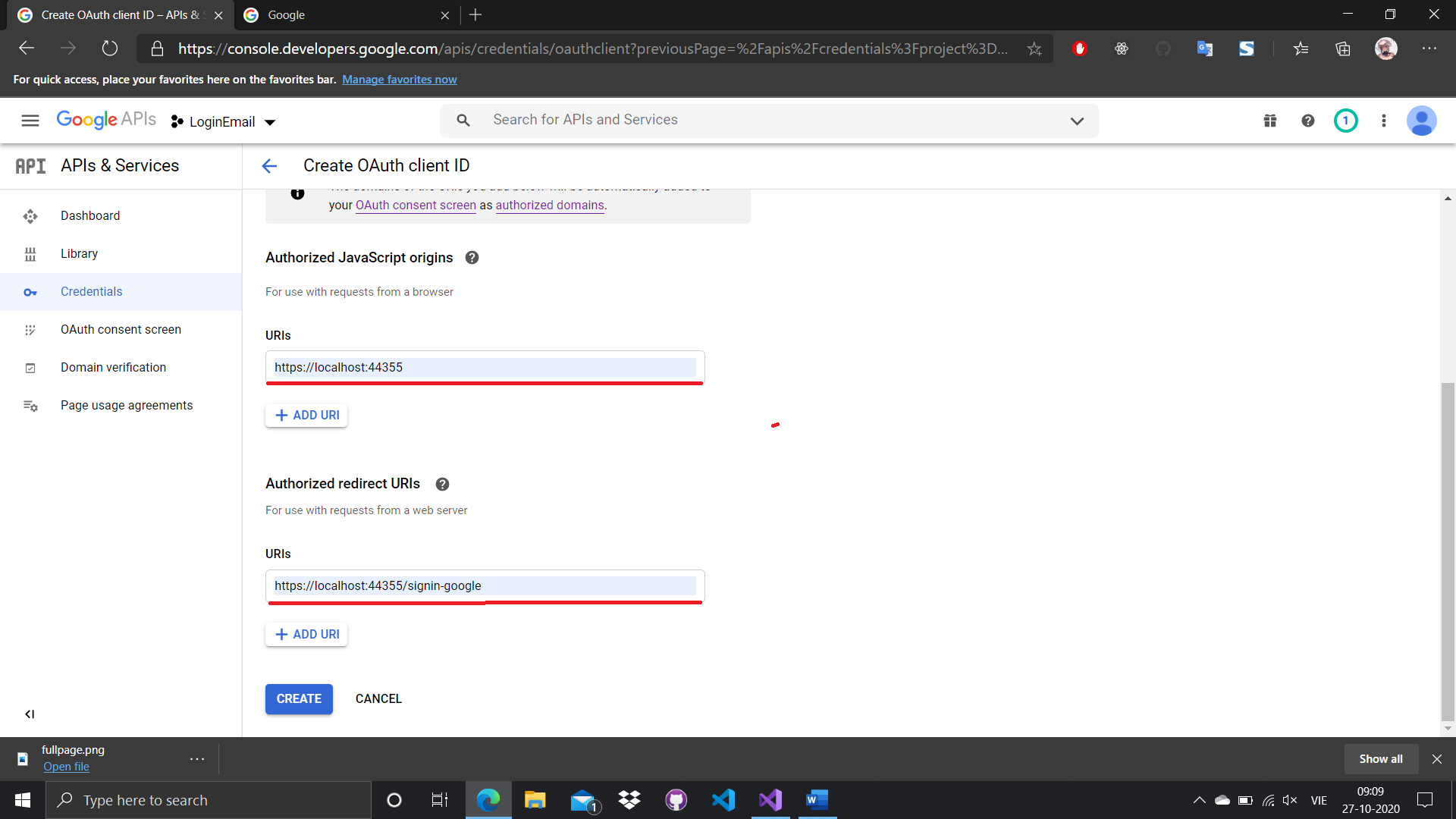
1. Chọn properties

* Sau đó nhấn vào Properties của project của bạn



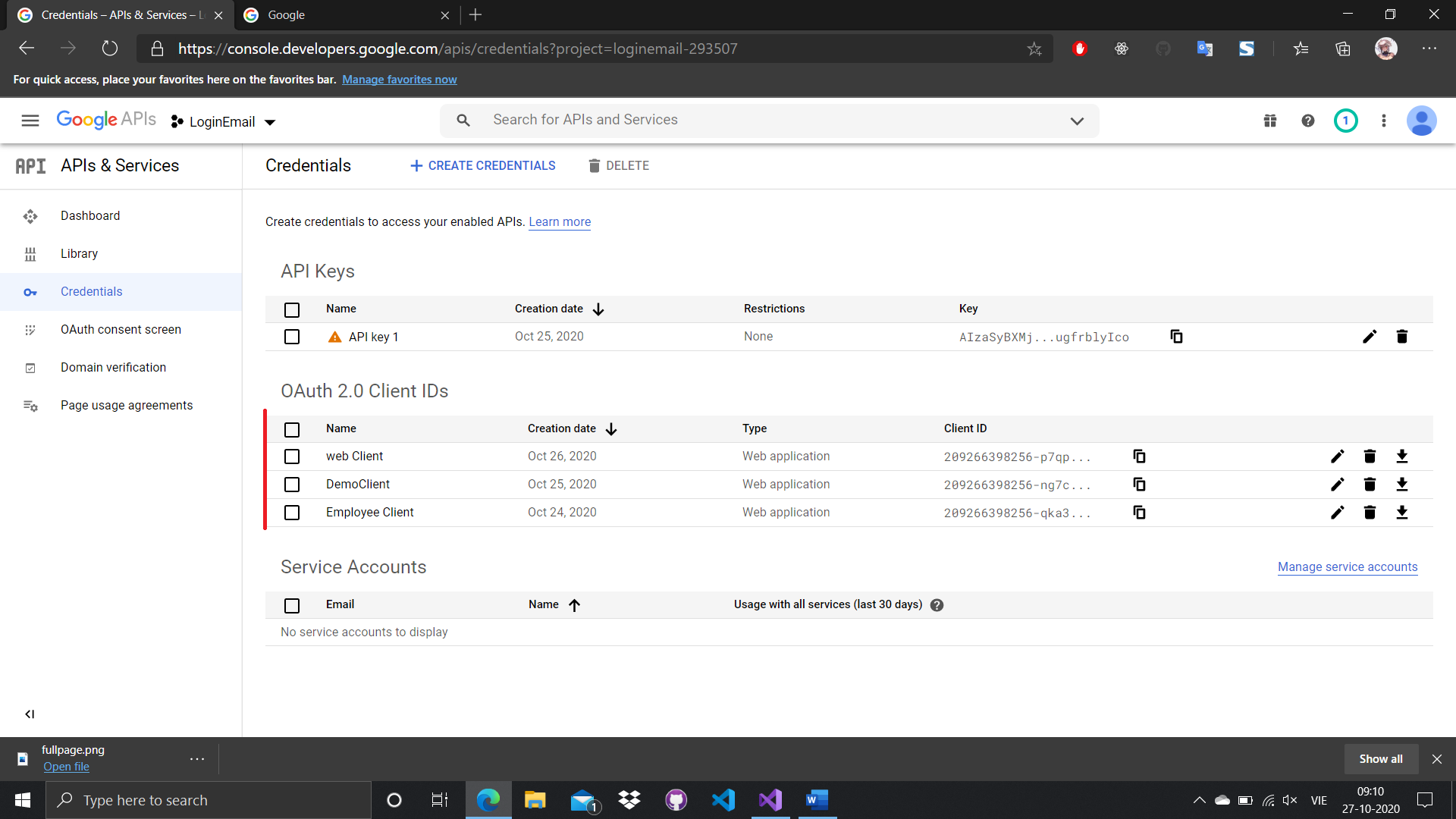
1. Tìm và copy App URL

* Sau đó lấy phần copy đó add vào URL bên trang web google api



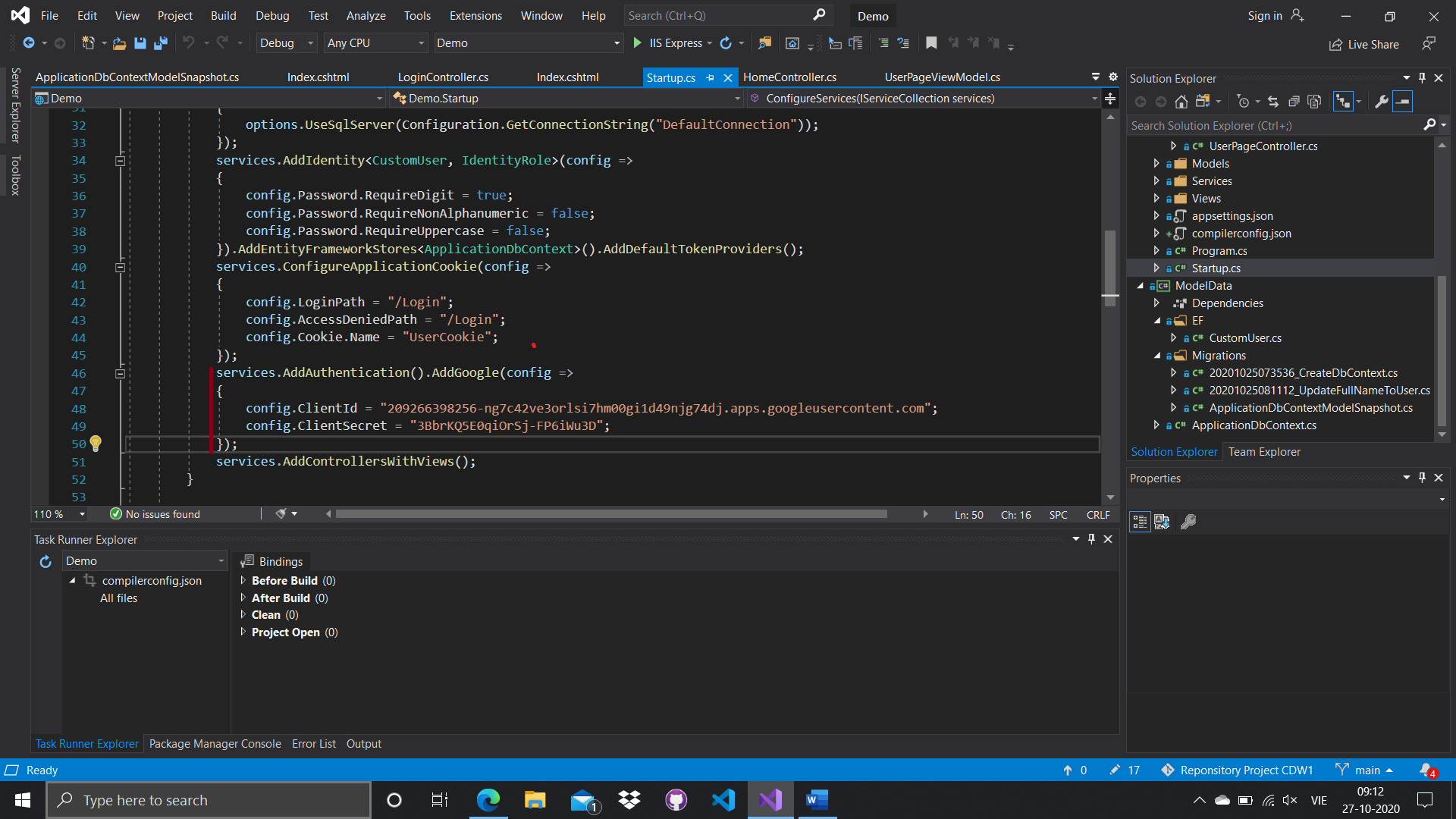
1. Thêm App URL của project vào đây

* Nhấn create
* Quay lại bên phía Credentials bạn sẽ thấy tên ứng dụng bạn đã đăng ký



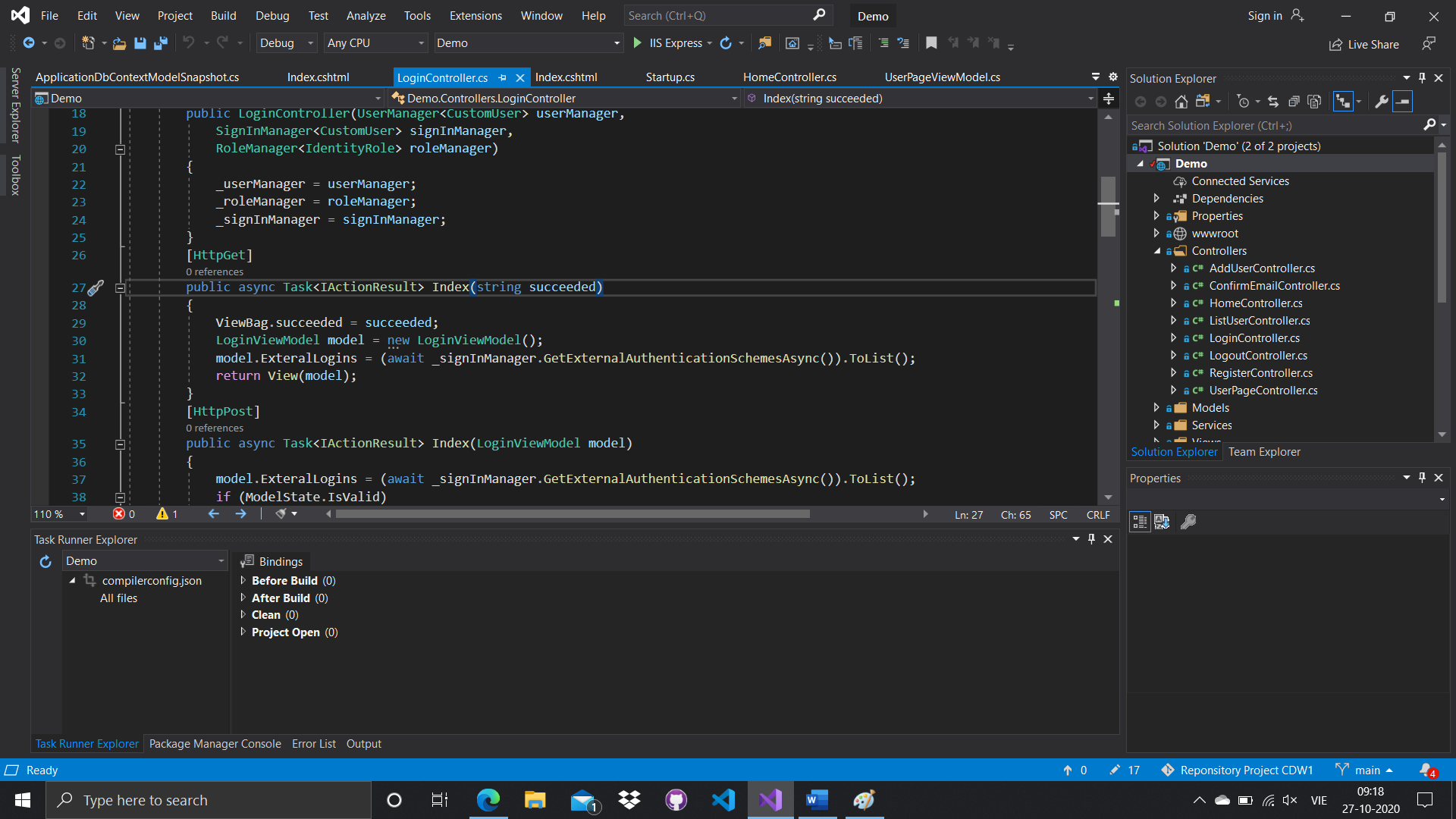
1. Nếu thành công tạo ra một tài khoản OAuth

* Nhấn vào tên name của bạn ta sẽ có client id và client secret
* Sau đó mở startup.cs
* Cài đặt gới add google <https://www.nuget.org/packages/Microsoft.AspNetCore.Authentication.Core/>



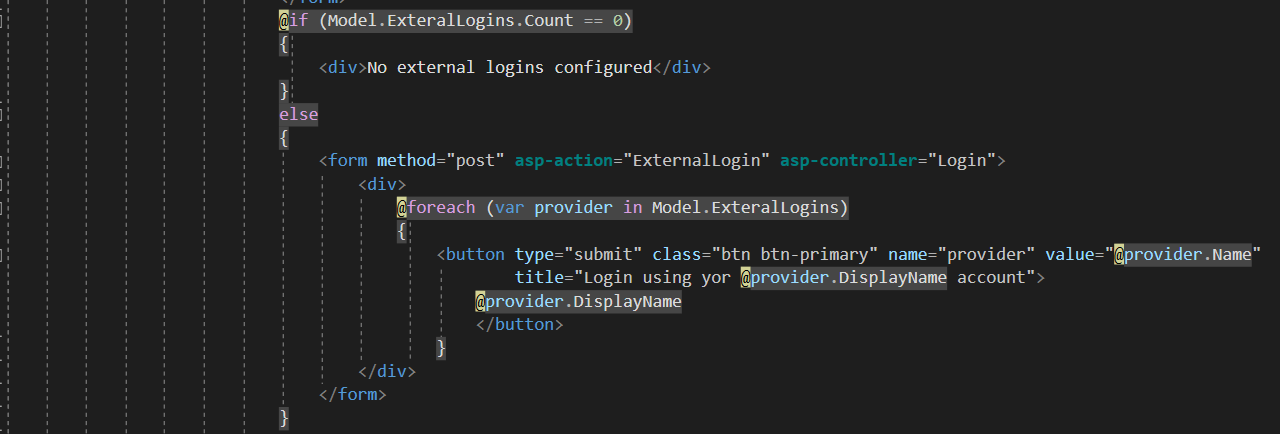
1. Thêm cột màu đỏ vào file startup.cs

* Khi này bạn đã kết nói thành công, để kiểm tra kết nối thành công hay k dùng signinManager có trong IdentityAuthentication
* await \_signInManager.GetExternalAuthenticationSchemesAsync()).ToList();



1. Kiểm tra bằng hàm GetExternalAuthenticationSchemesAsync

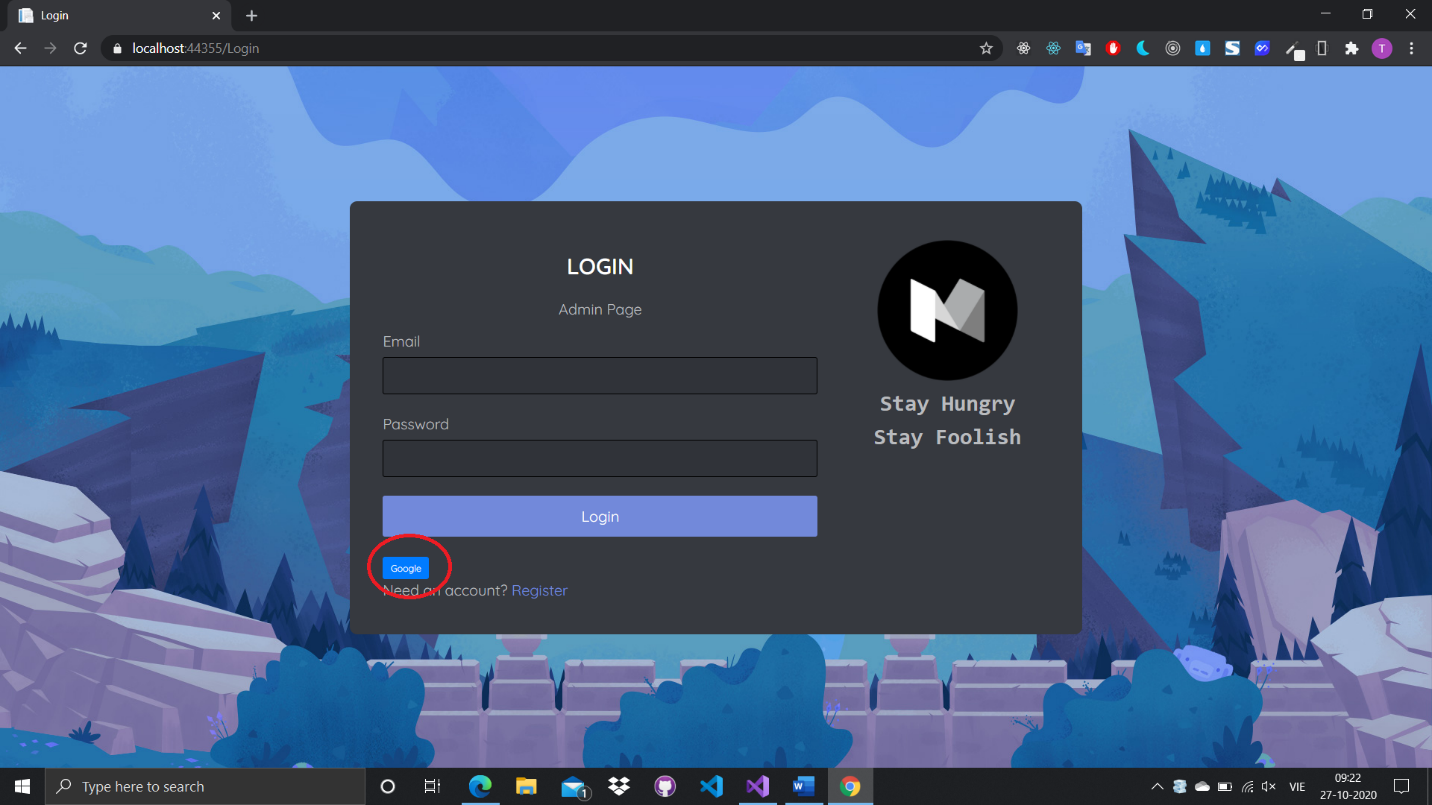
Ở đây ngay chỗ LoginController với action Index ta thấy được câu lệnh ở trên, lúc này thêm nó vào một model và truyền qua View Index phía Login



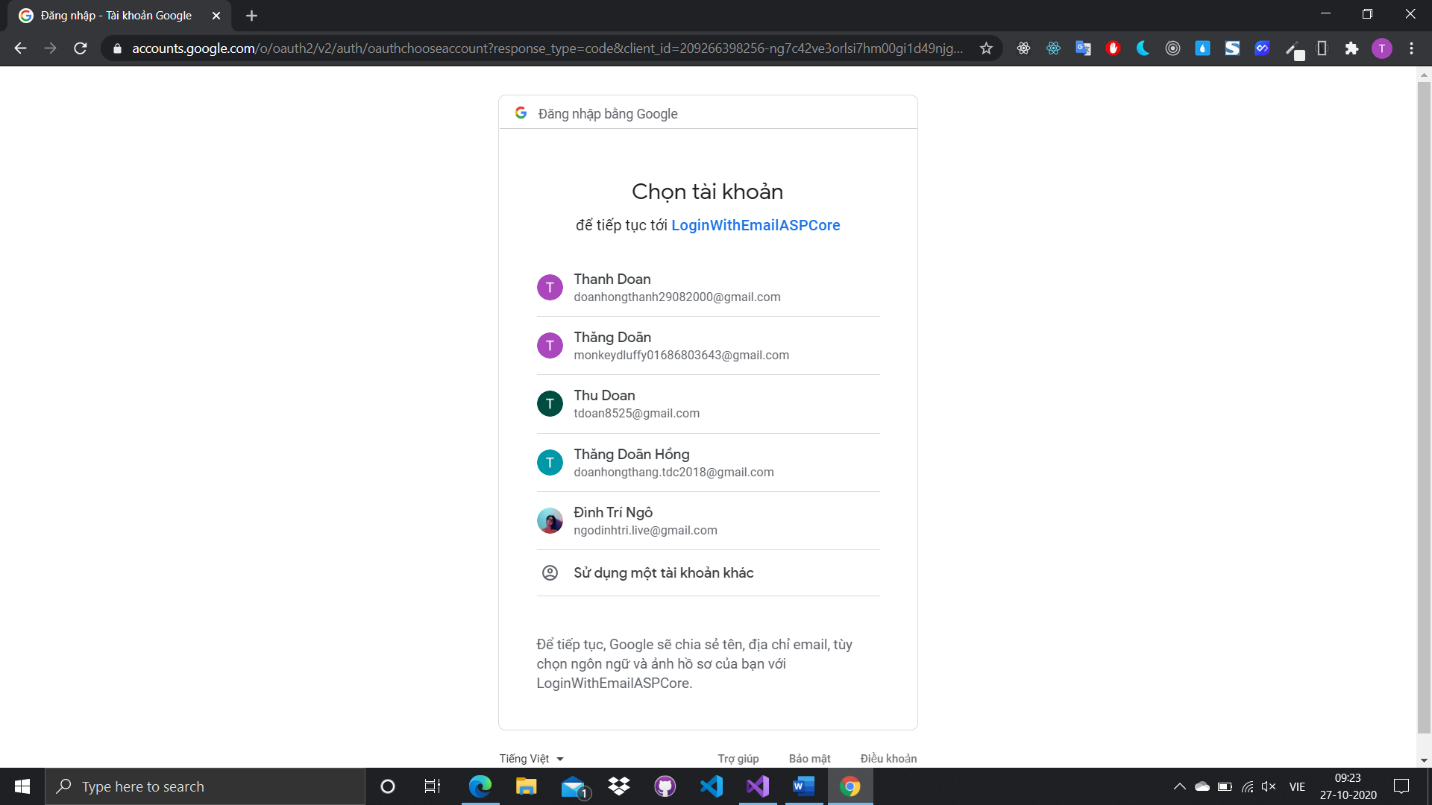
1. Kiểm tra nếu có tích hợp sẽ hiện button

Với câu lệnh Model.ExternalLogins.Count nó dc truyền qua trong Index action của LoginController. Nếu nó ==0 nghĩa là có lỗi trong việc khởi tạo

Nếu nó != 0 ta thấy được phần button đc cấu trúc như sau:



1. Hiển thị trang login

* Nhấn vào nút google ta được
* 

1. Hiển thị trang google login

* Vậy là xong phần login nhé, còn về việc lấy thông tin như thế nào. Vào
* <https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.identity.signinmanager-1?view=aspnetcore-3.1>
* <https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.identity.usermanager-1?view=aspnetcore-3.1>