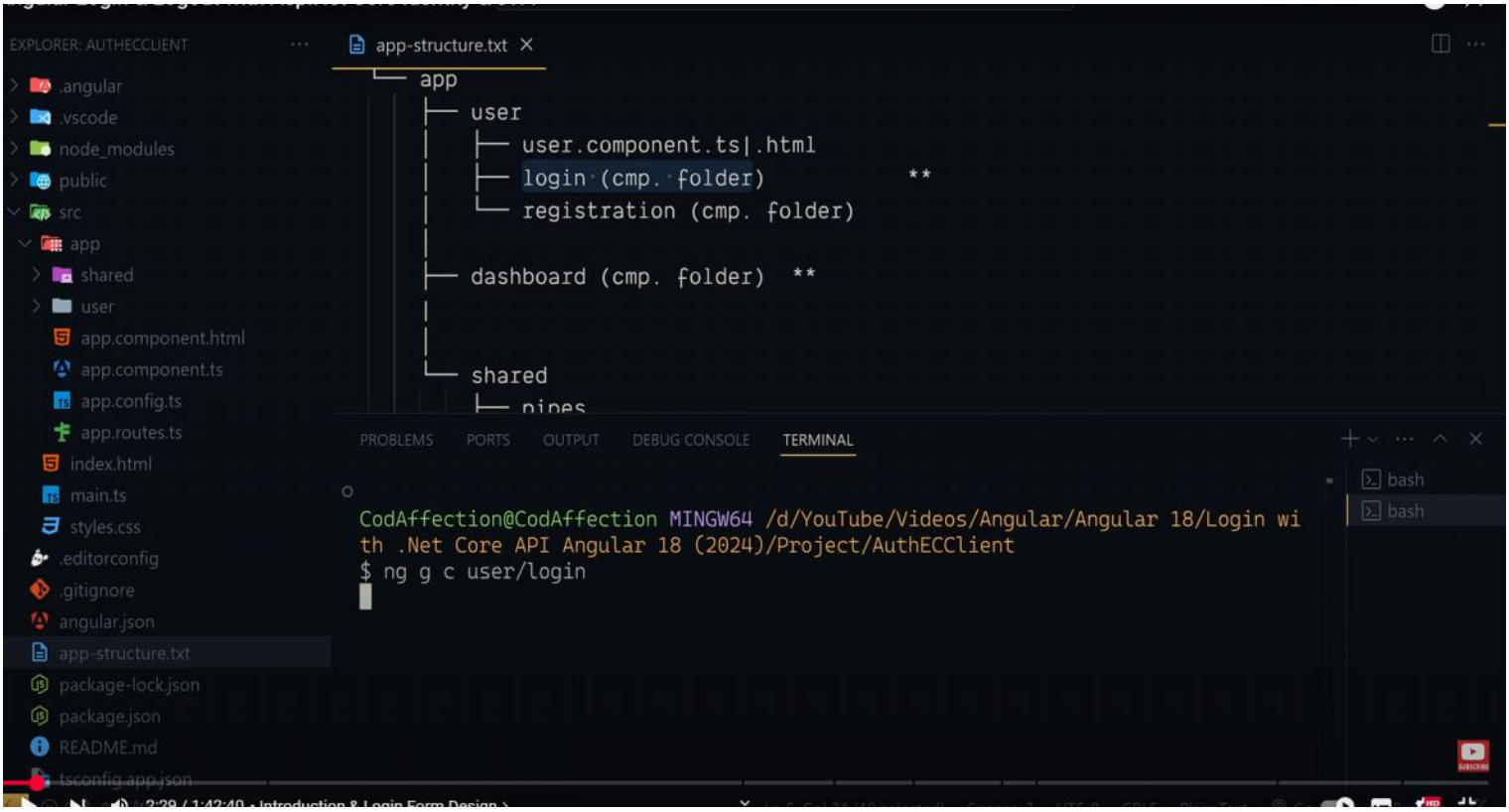
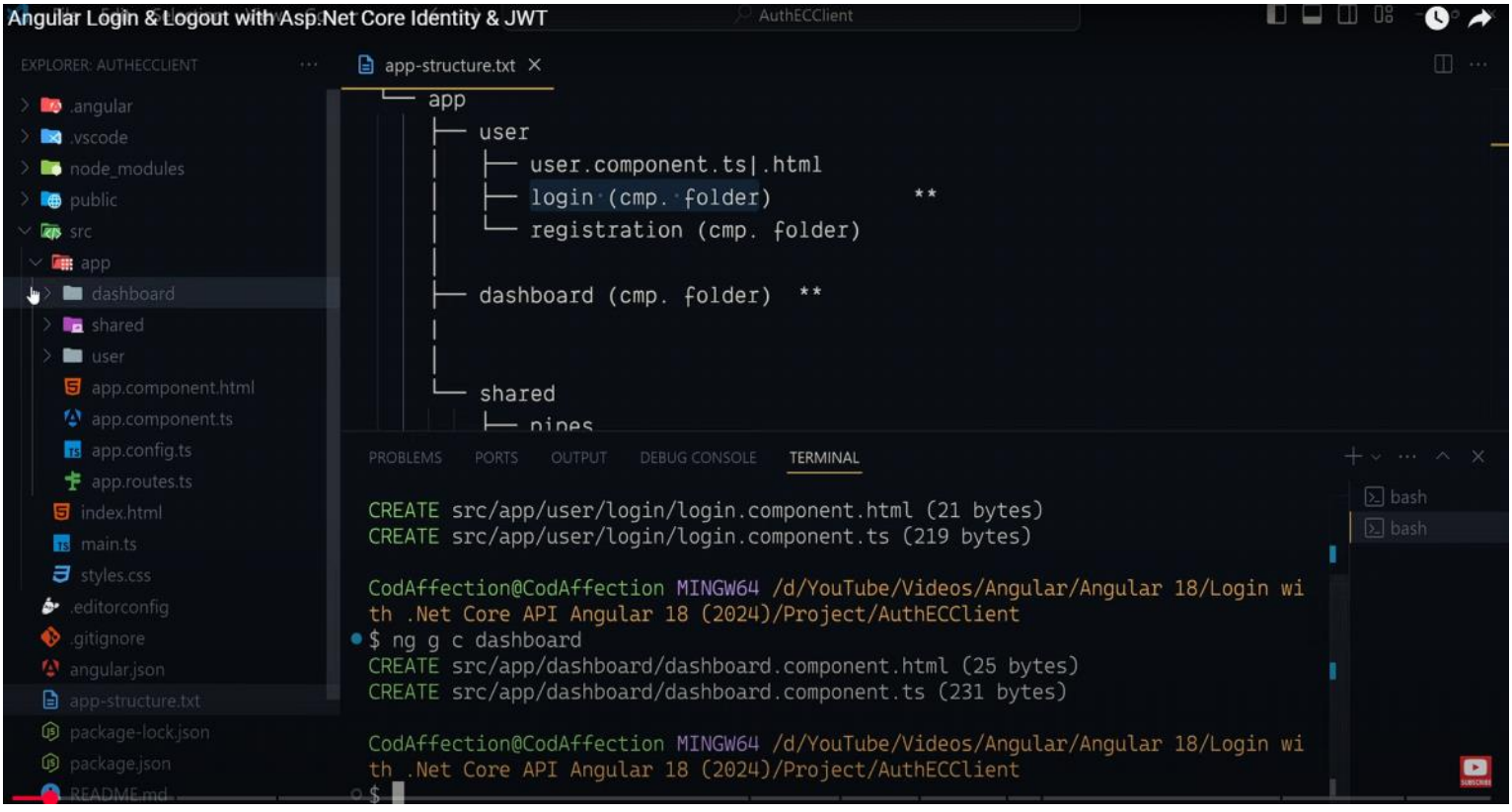
Angular Login & Logout with Asp.Net Core Identity & JWT

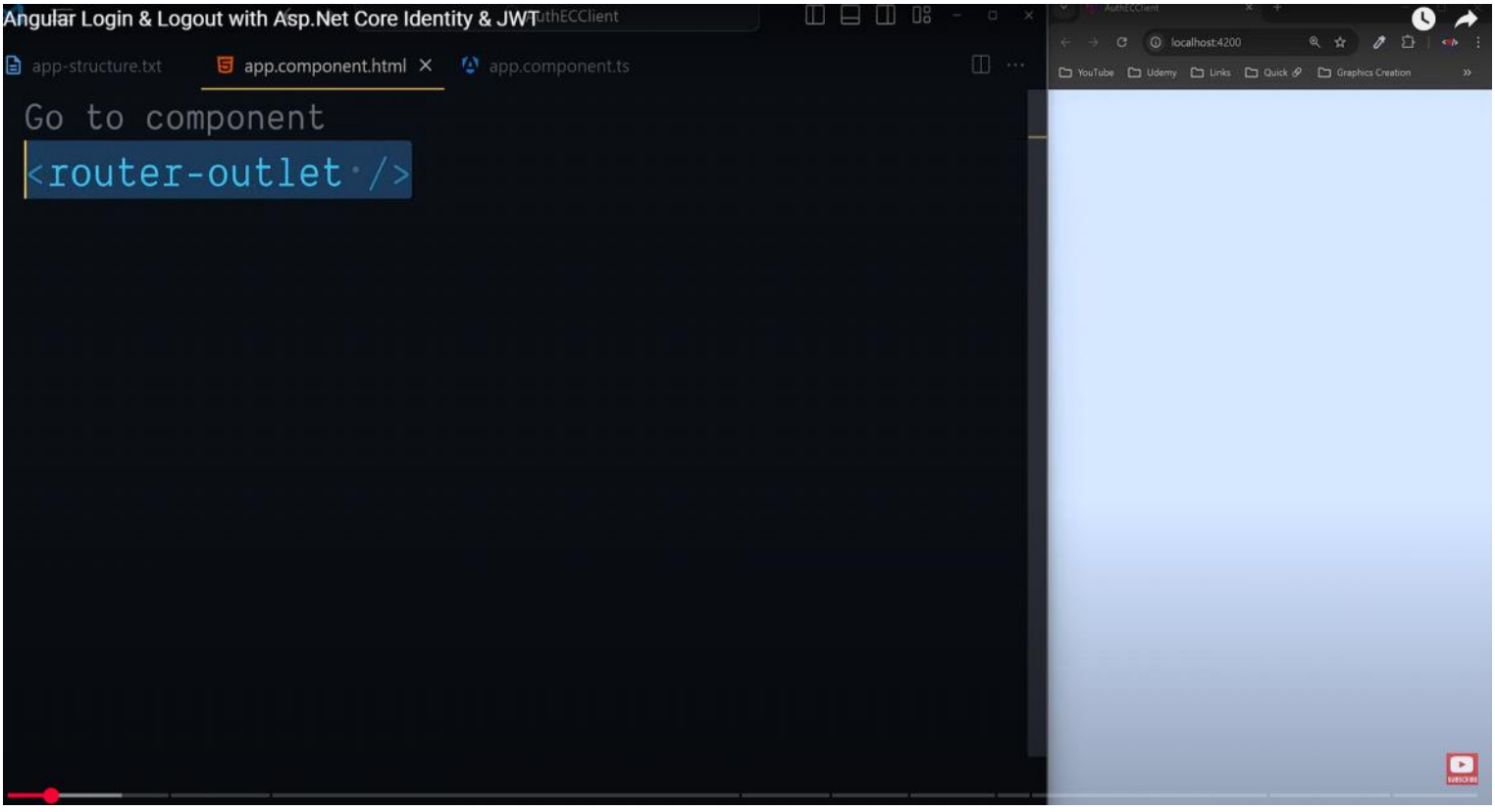
Tuesday, March 18, 2025 6:08 AM

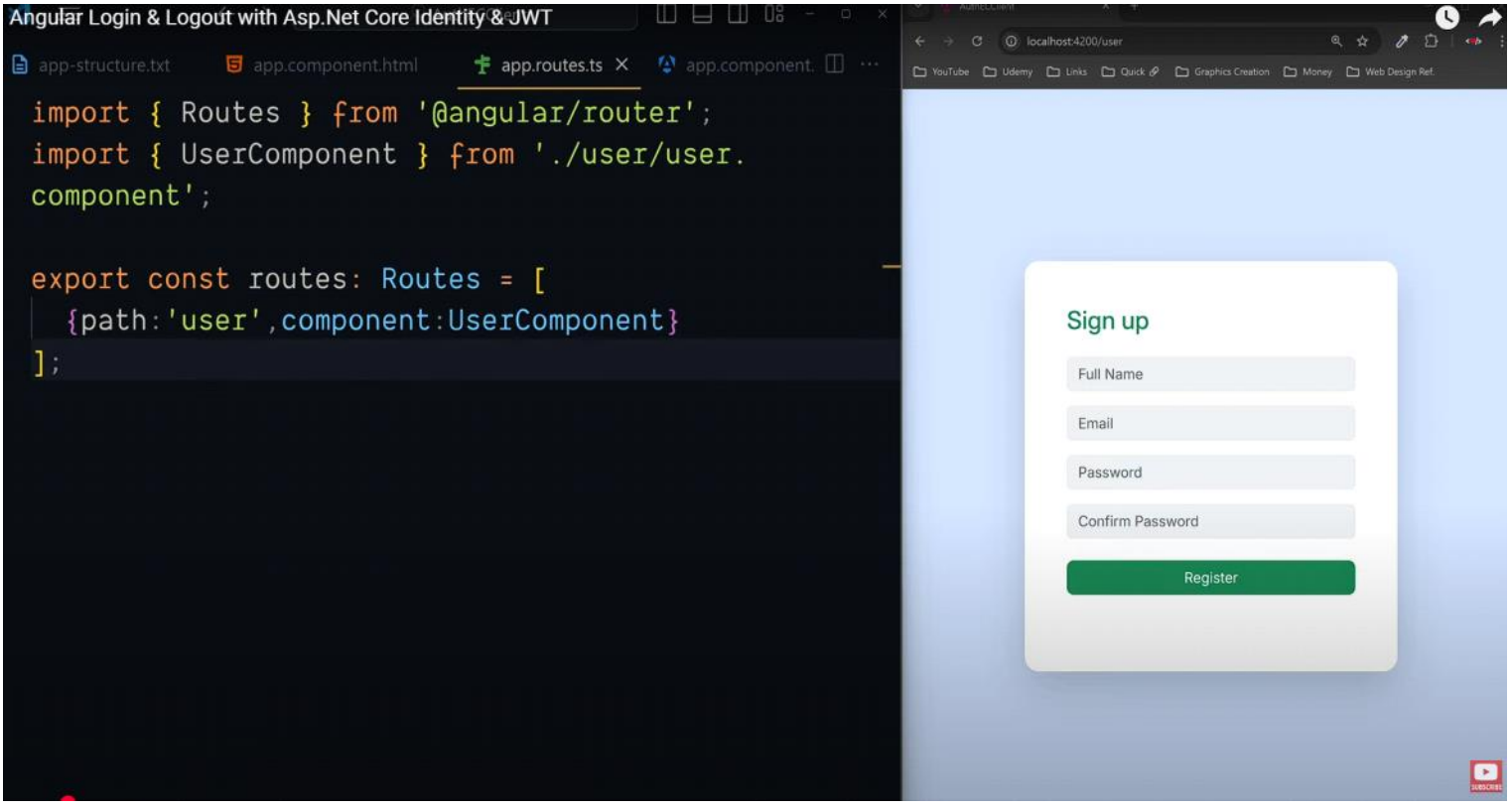


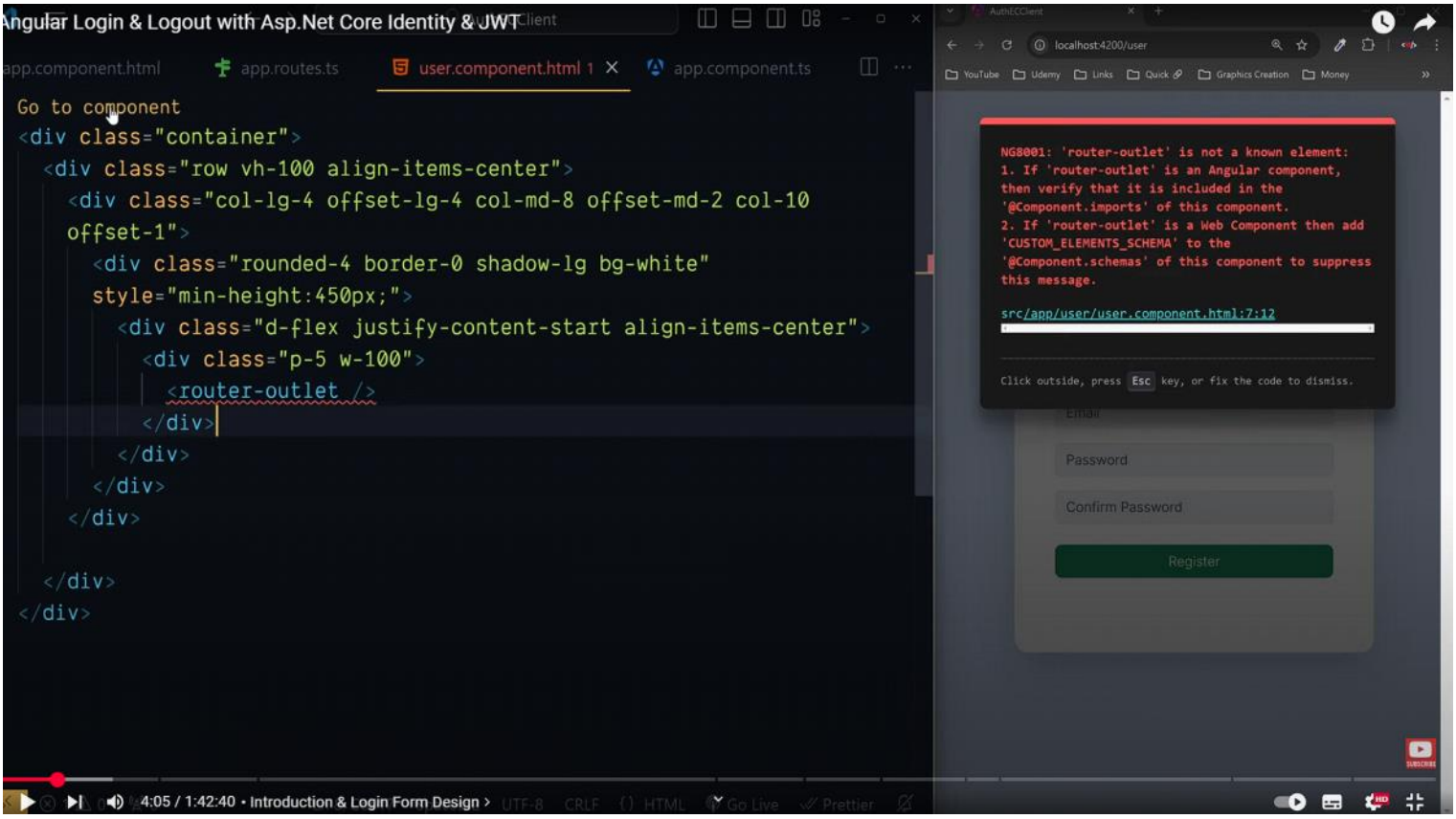
Create dashboard



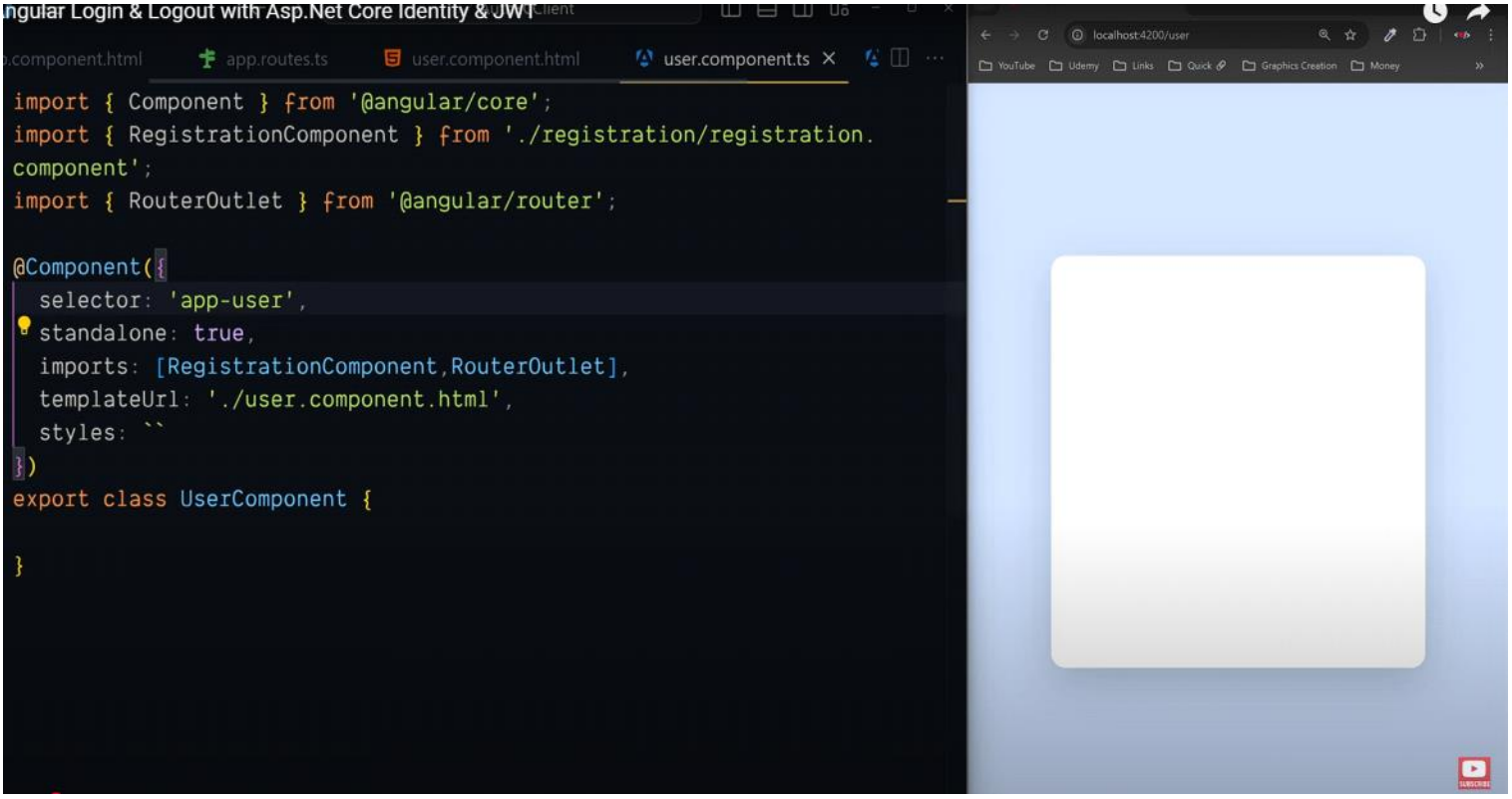
For lots of routes

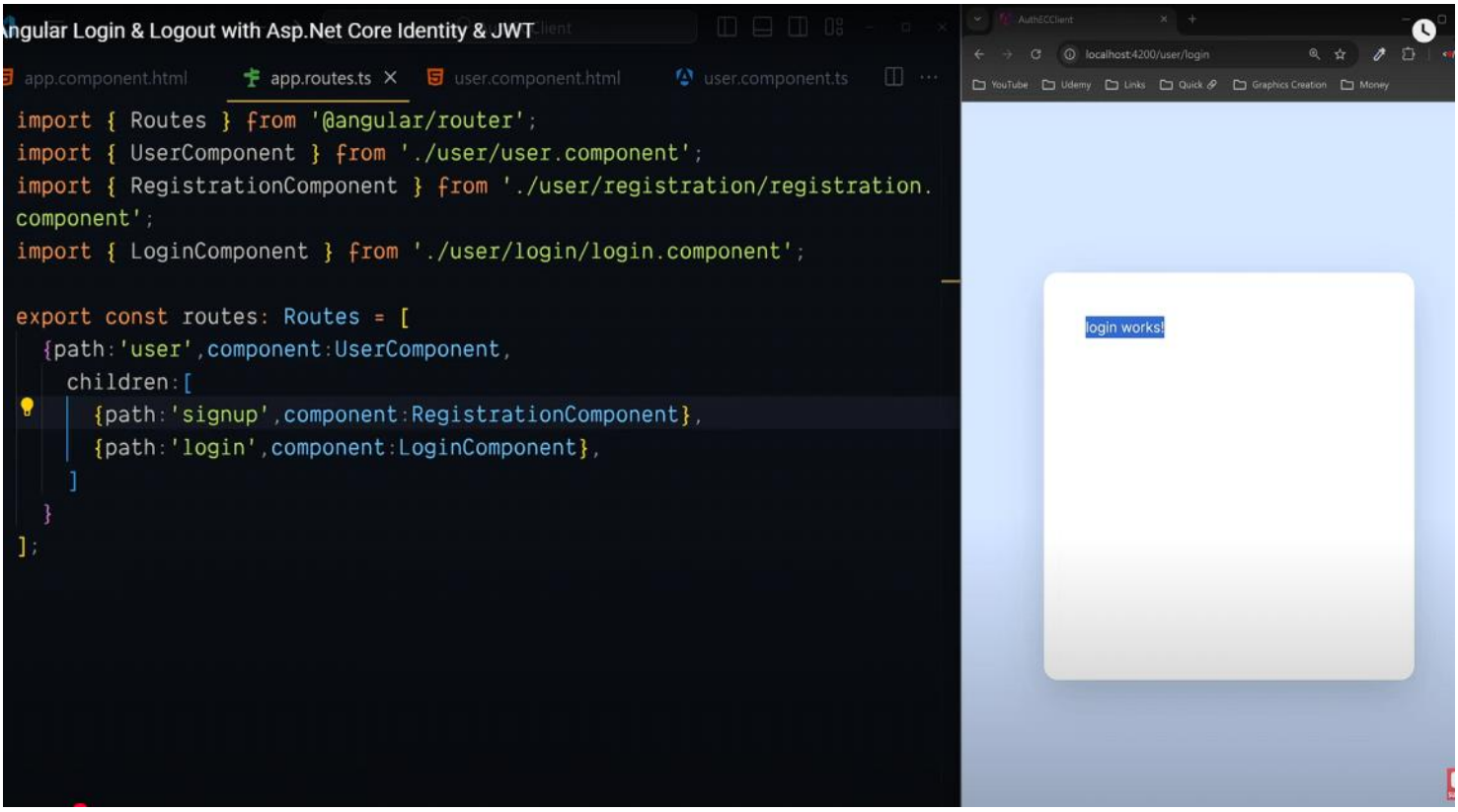


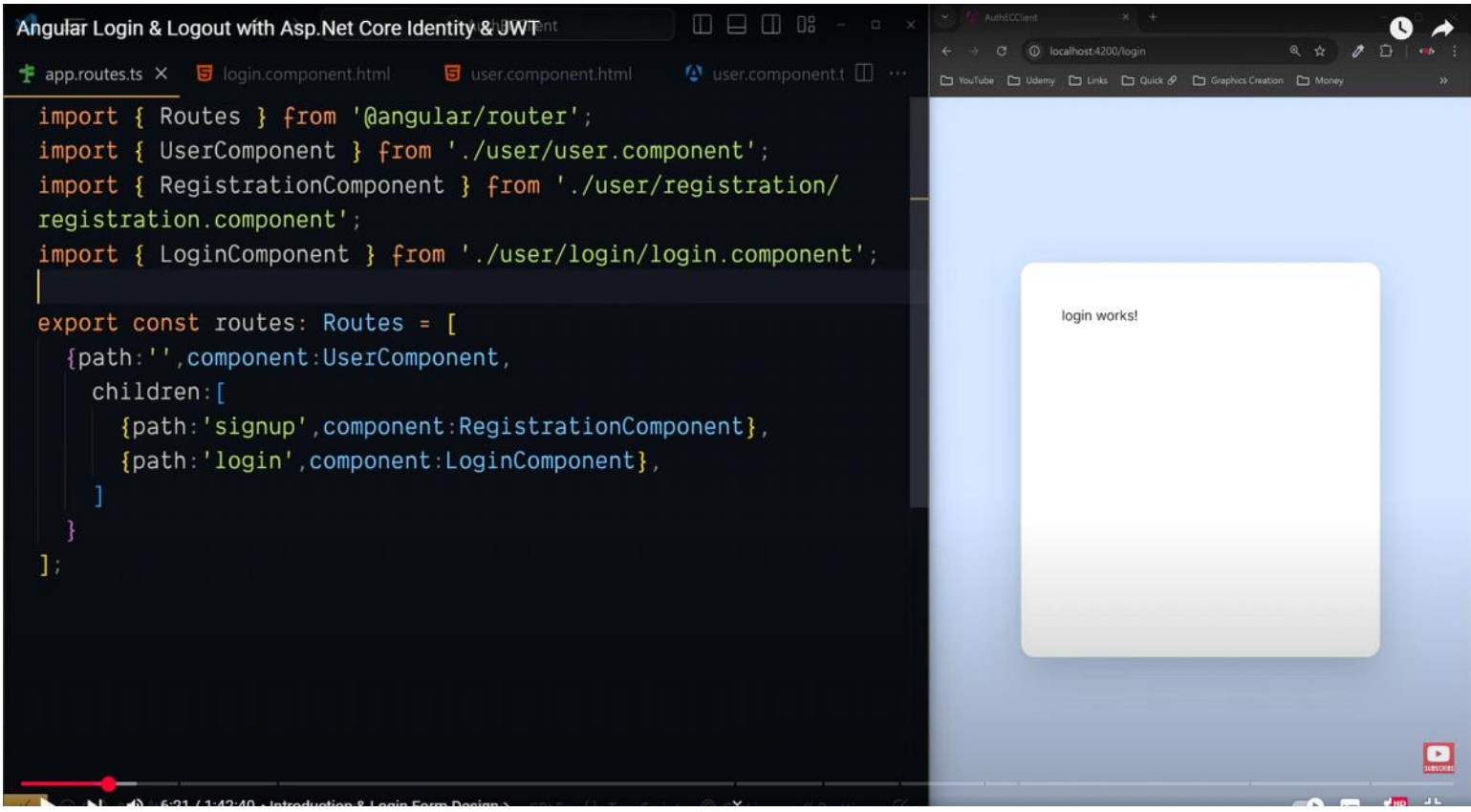


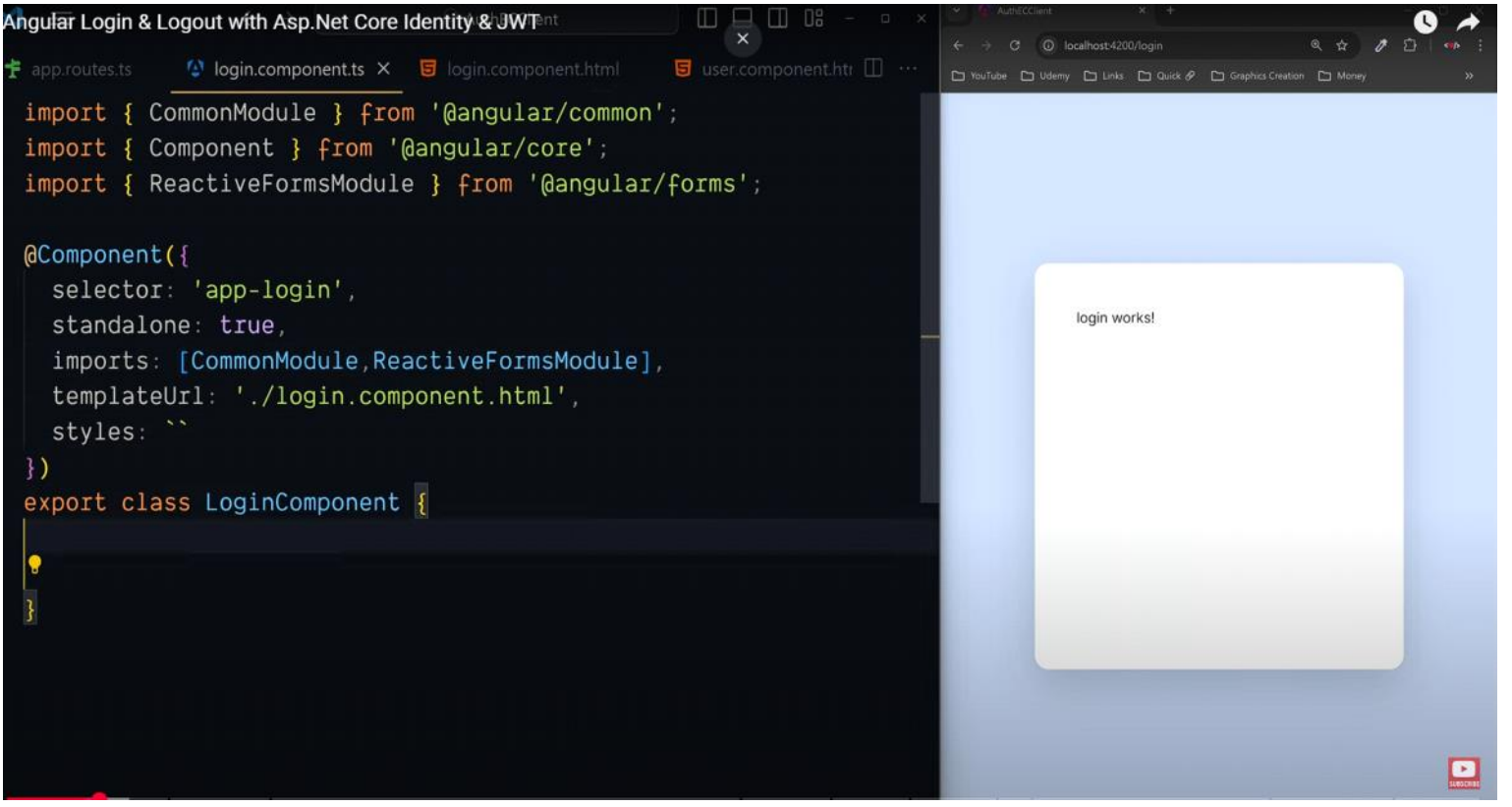


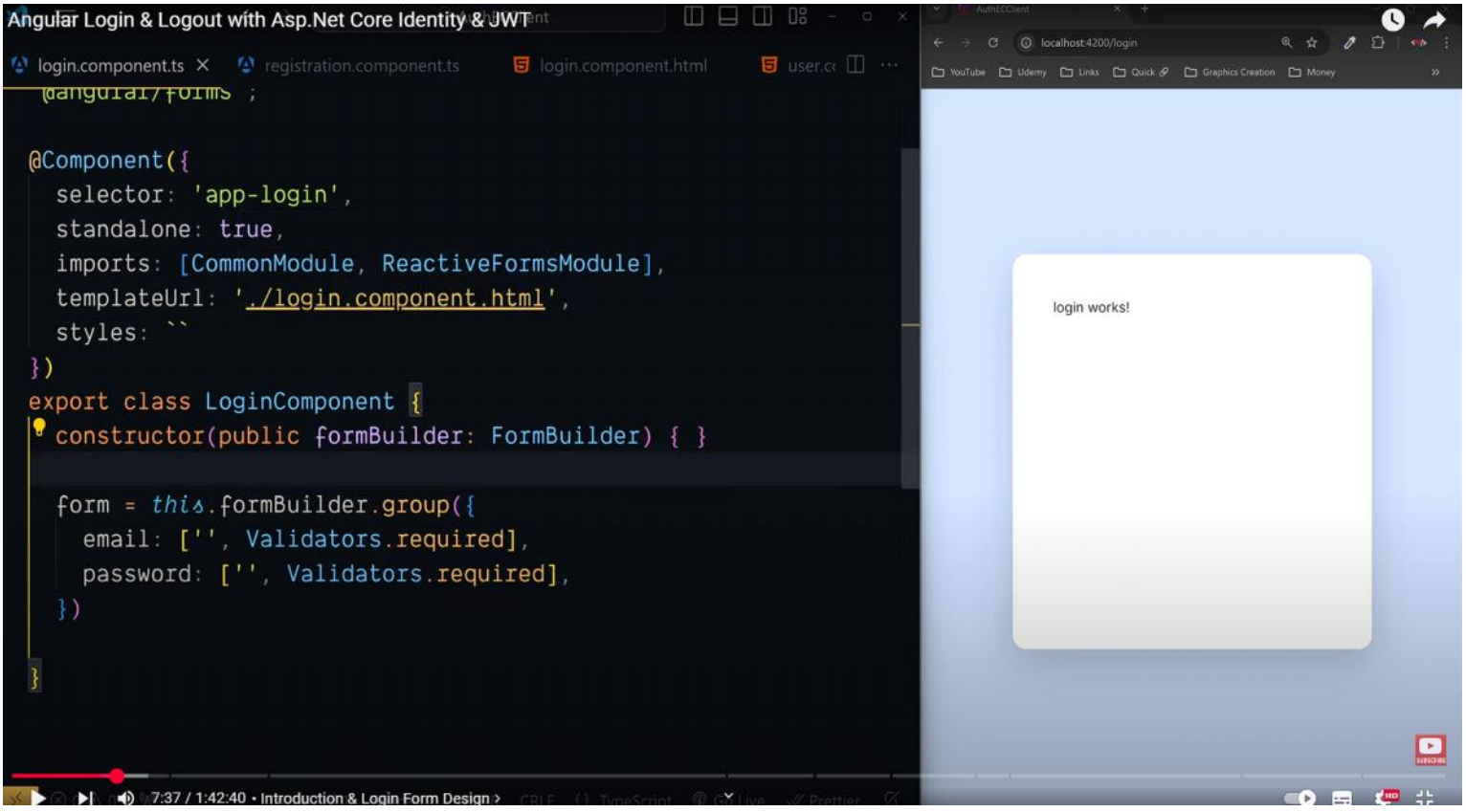
Import router outlet

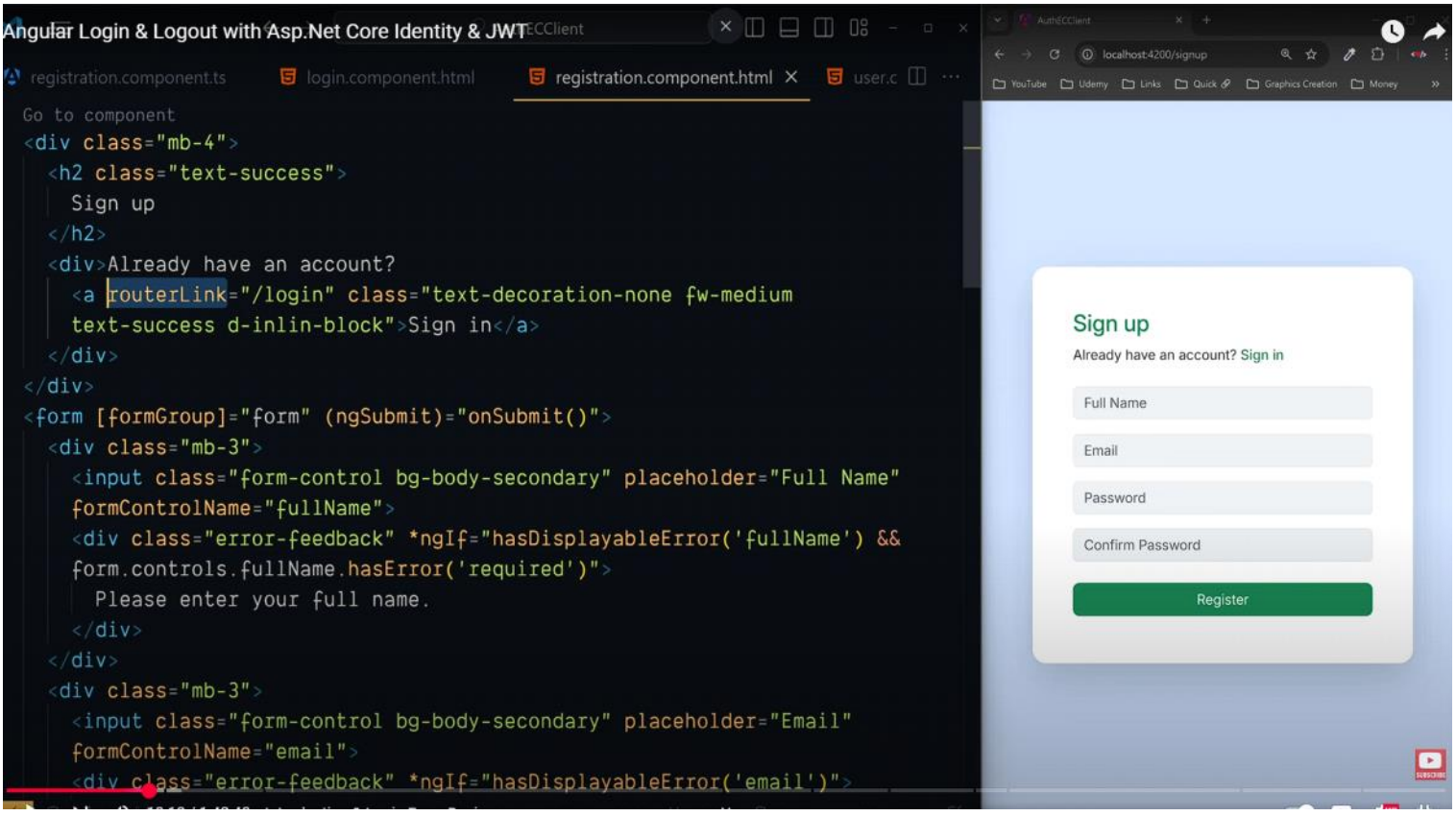



Angular Login & Logout with Asp.Net Core Identity & JWT
app.component.html
app.routes.ts
user.component.html
user.component.ts
import { Routes } from '@angular/router';
import { UserComponent } from './user/user.component';
import { RegistrationComponent } from './user/registration/registration.component';
import { LoginComponent } from './user/login/login.component';
export const routes: Routes = [
{path:'user',component:UserComponent,
children:[
{path:'signup',component:RegistrationComponent},
{path:'login',component:LoginComponent},
]
}
];
AuthECClient
localhost:4200/user/login
YouTube
Udemy
Links
Quick
Graphics Creation
Money
login works!

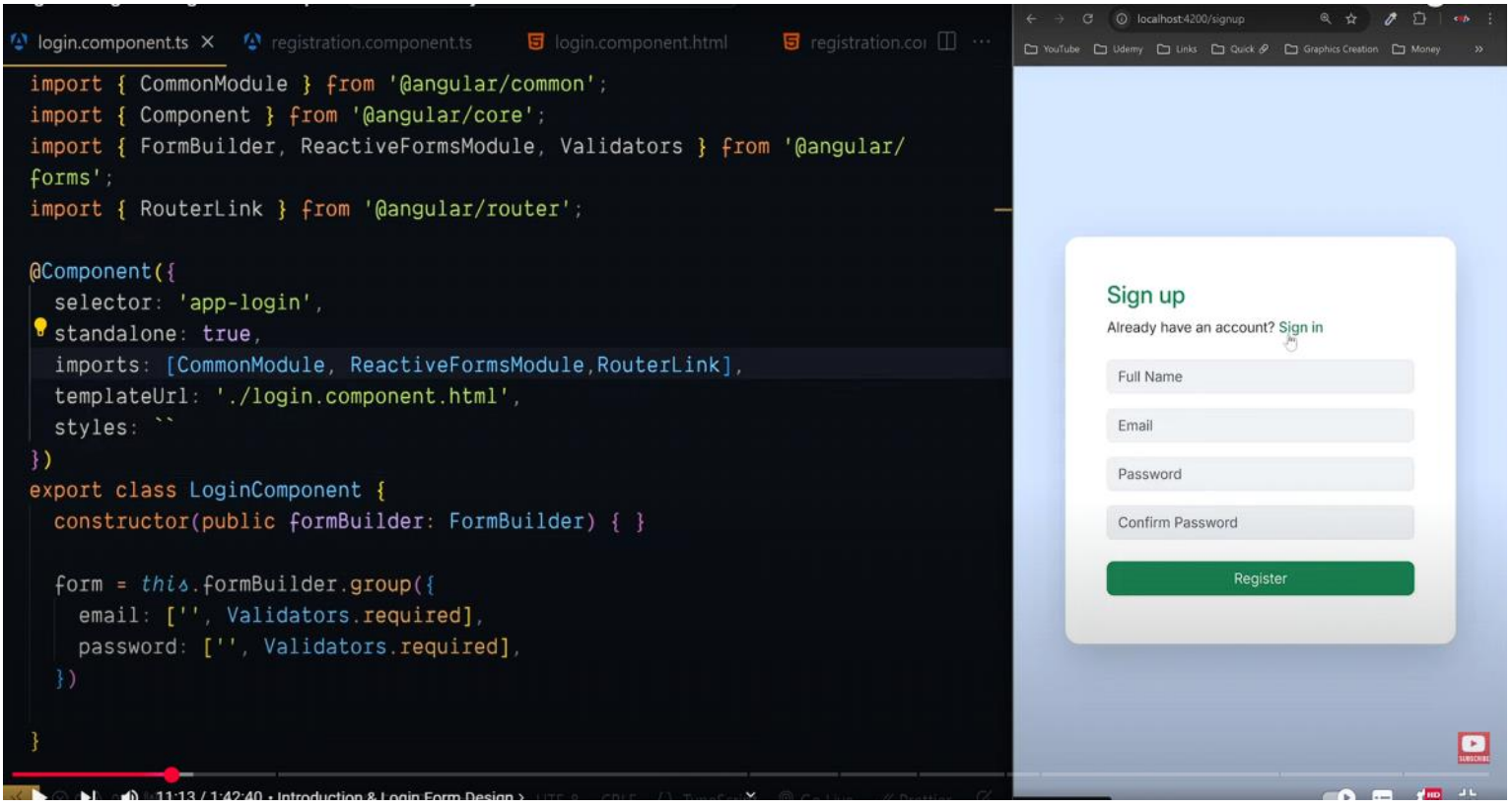

Angular Login & Logout with Asp.Net Core Identity & JWT
app.routes.ts
login.component.html
user.component.html
import { Routes } from '@angular/router';
import { UserComponent } from './user/user.component';
import { RegistrationComponent } from './user/registration/
registration.component';
import { LoginComponent } from './user/login/login.component';
export const routes: Routes = [
{path:'',component:UserComponent,
children:[
{path:'signup',component:RegistrationComponent},
{path:'login',component:LoginComponent},
]
}
];
localhost:4200/login
login works!

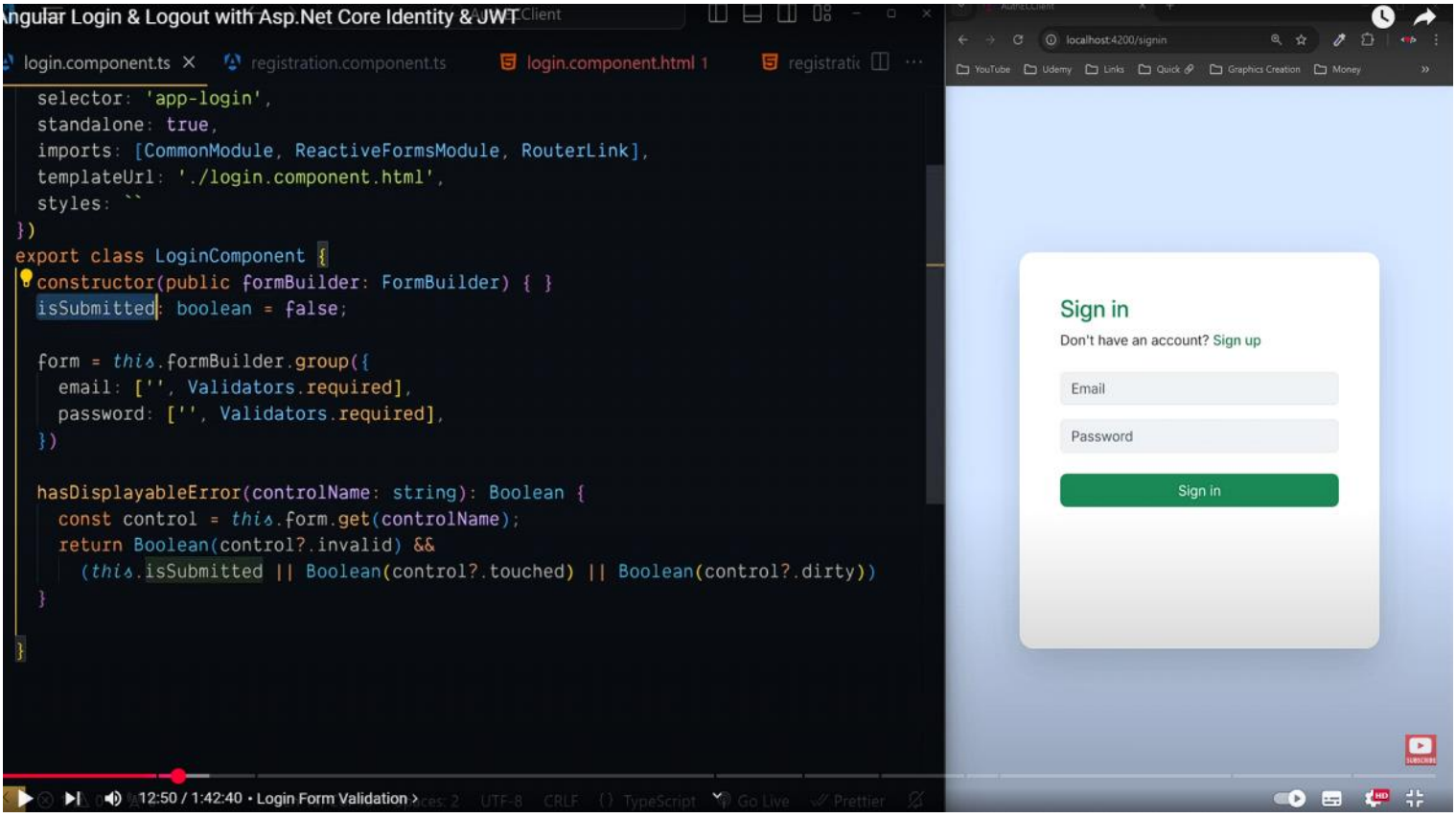

Angular Login & Logout with Asp.Net Core Identity & JWT
app.routes.ts
login.component.ts
login.component.html
import { CommonModule } from '@angular/common';
import { Component } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
@Component({
selector: 'app-login',
standalone: true,
imports: [CommonModule,ReactiveFormsModule],
templateUrl: './login.component.html',
styles: ``
})
export class LoginComponent {
}
localhost:4200/login
login works!

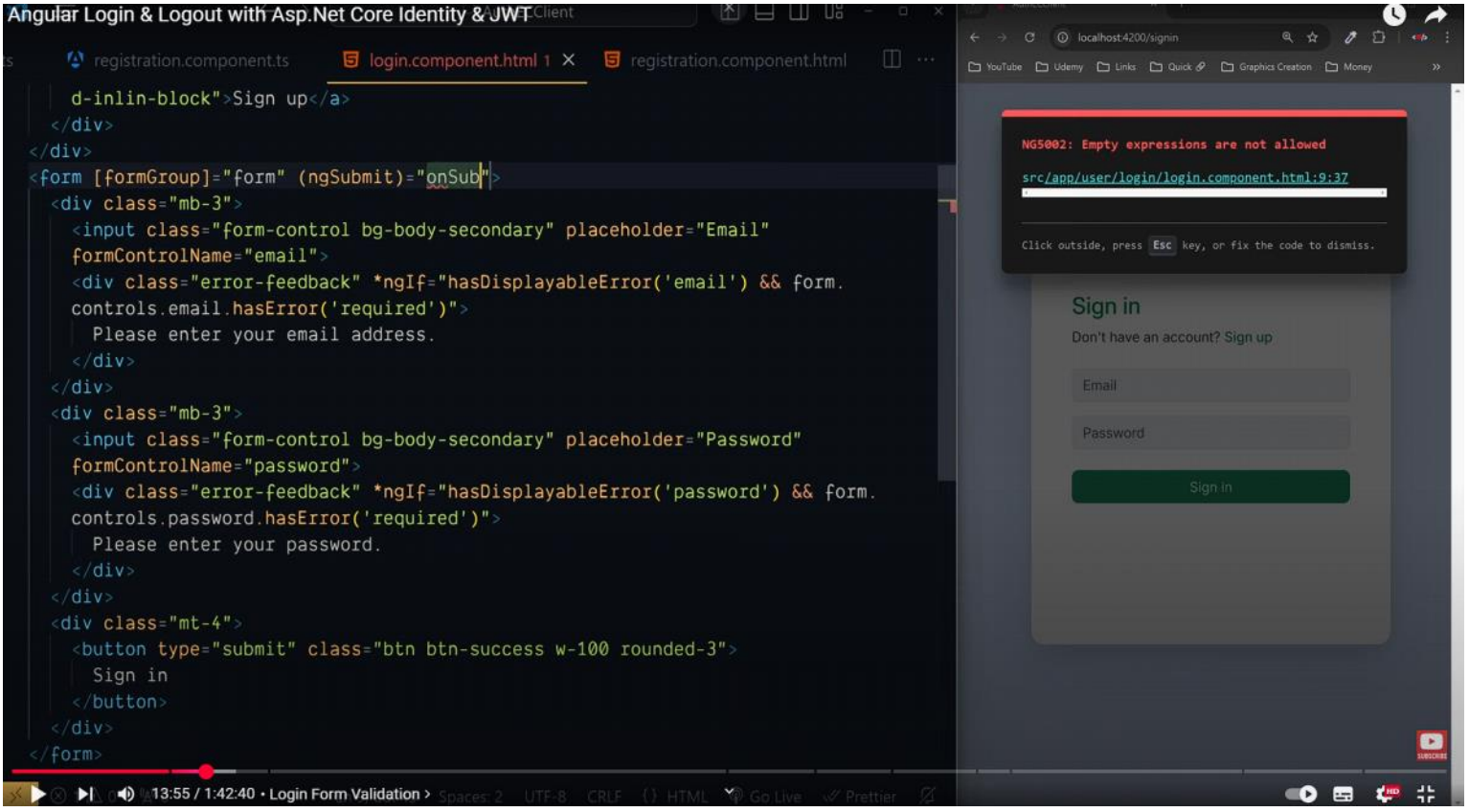

Angular Login & Logout with Asp.Net Core Identity & JWT
login.component.ts
registration.component.ts
login.component.html
@Component({
selector: 'app-login',
standalone: true,
imports: [CommonModule, ReactiveFormsModule],
templateUrl: './login.component.html',
styles: ``
})
export class LoginComponent {
constructor(public formBuilder: FormBuilder) { }
form = this.formBuilder.group({
email: ['', Validators.required],
password: ['', Validators.required],
})
}
7:37 / 1:42:40 • Introduction & Login Form Design
localhost:4200/login
login works!



Import router link




Angular Login & Logout with Asp.Net Core Identity & JWT
login.component.ts
registration.component.ts
login.component.html 1
selector: 'app-login',
standalone: true,
imports: [CommonModule, ReactiveFormsModule, RouterLink],
templateUrl: './login.component.html',
styles: ``
})
export class LoginComponent {
constructor(public formBuilder: FormBuilder) { }
isSubmitted: boolean = false;
form = this.formBuilder.group({
email: ['', Validators.required],
password: ['', Validators.required],
})
hasDisplayableError(controlName: string): Boolean {
const control = this.form.get(controlName);
return Boolean(control?.invalid) &&
(this.isSubmitted || Boolean(control?.touched) || Boolean(control?.dirty))
}
}
12:50 / 1:42:40 • Login Form Validation
localhost:4200/signin
YouTube
Udemy
Links
Quick
Graphics Creation
Money
Sign in
Don't have an account? Sign up
Email
Password
Sign in


Angular Login & Logout with Asp.Net Core Identity & JWT
registration.component.ts
login.component.html 1
registration.component.html
d-inlin-block">Sign up

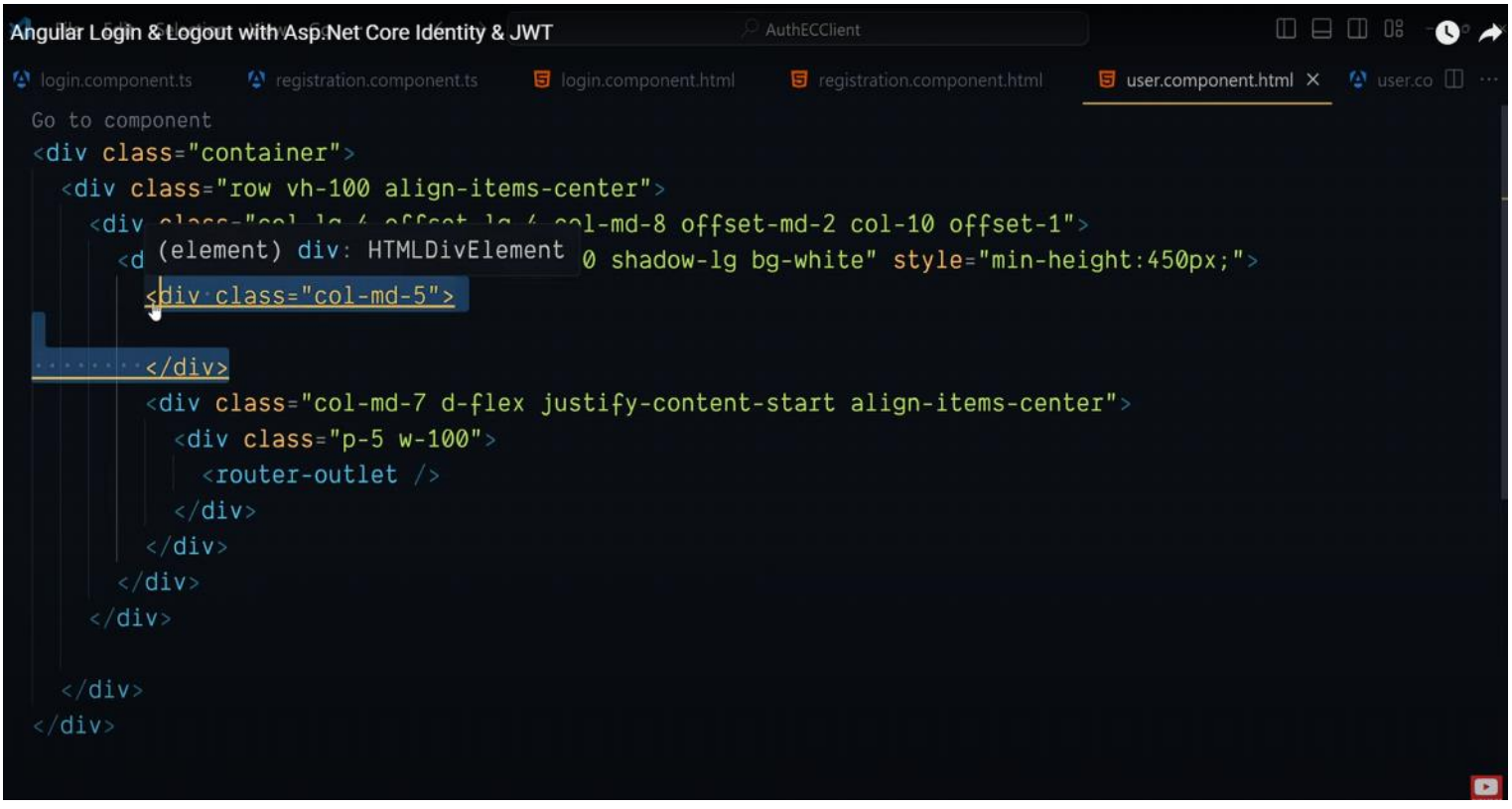
<form [formGroup]="form" (ngSubmit)="onSub">

<input class="form-control bg-body-secondary" placeholder="Email"
formControlName="email">
<div class="error-feedback" *ngIf="hasDisplayableError('email') && form.
controls.email.hasError('required')">
Please enter your email address.

<input class="form-control bg-body-secondary" placeholder="Password"
formControlName="password">
<div class="error-feedback" *ngIf="hasDisplayableError('password') && form.
controls.password.hasError('required')">
Please enter your password.

<button type="submit" class="btn btn-success w-100 rounded-3">
Sign in
</button>

</form>
13:55 / 1:42:40 • Login Form Validation
localhost:4200/signin
NG5002: Empty expressions are not allowed
src/app/user/login/login.component.html:9:37
Click outside, press Esc key, or fix the code to dismiss.
Sign in
Don't have an account? Sign up
Email
Password
Sign in

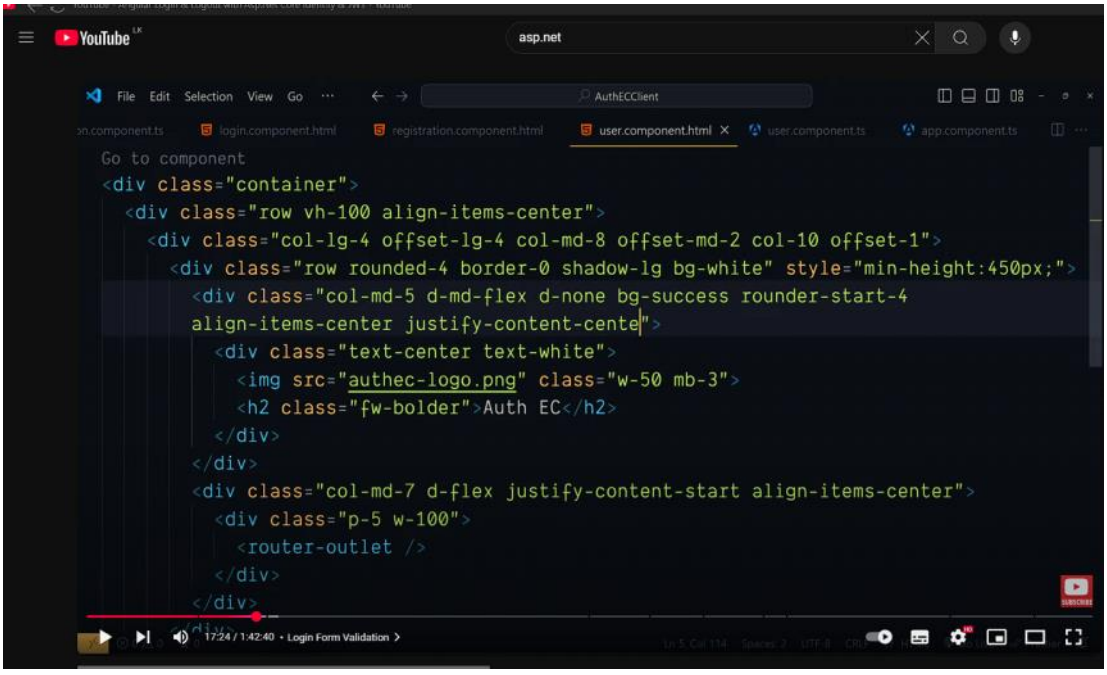

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECClient
login.component.ts
registration.component.ts
login.component.html
registration.component.html
user.component.html
Go to component

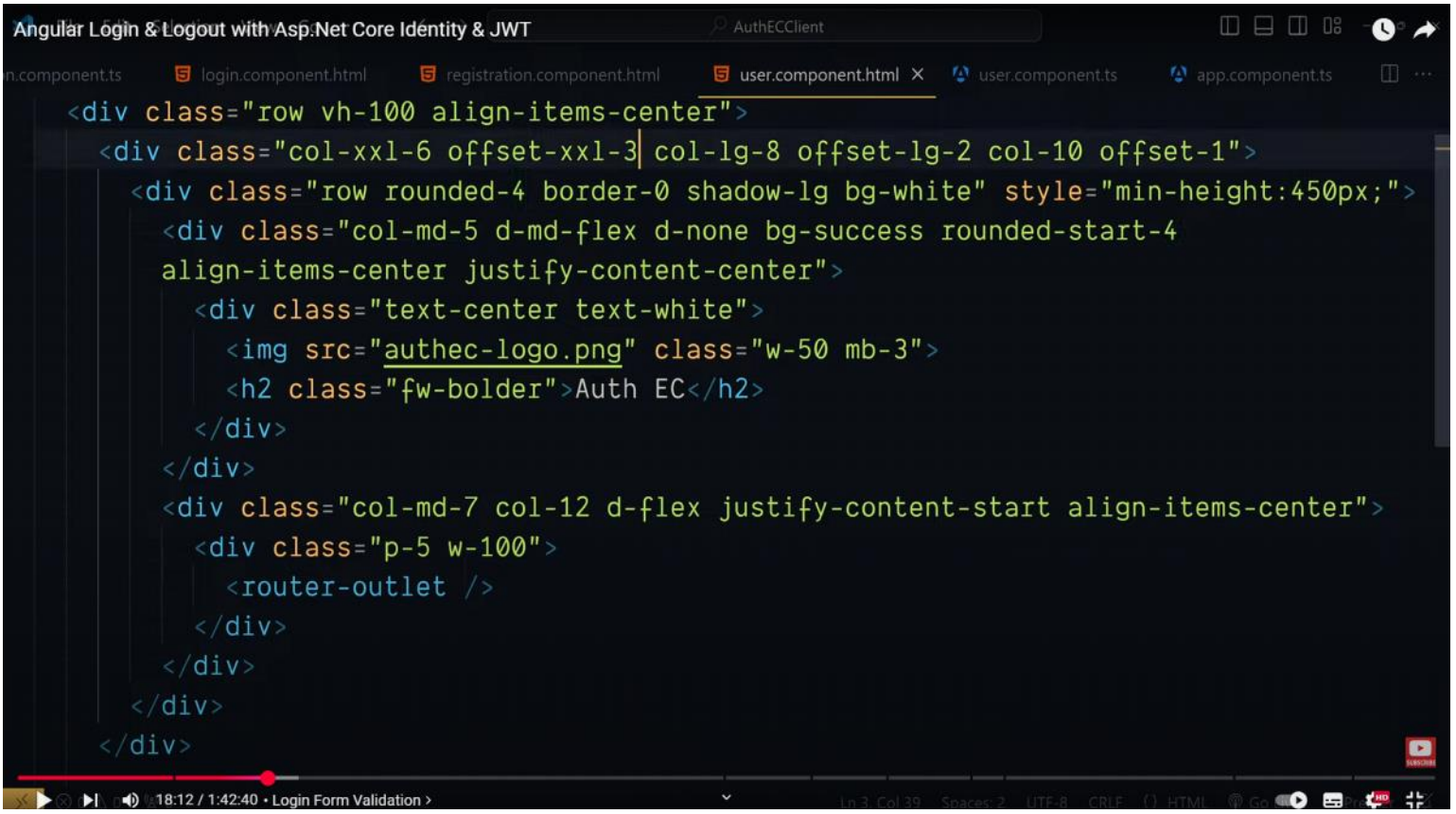
(element) div: HTMLDivElement
0 shadow-lg bg-white" style="min-height:450px;">

<router-outlet />

Should put this image inside the

Shard/public folder



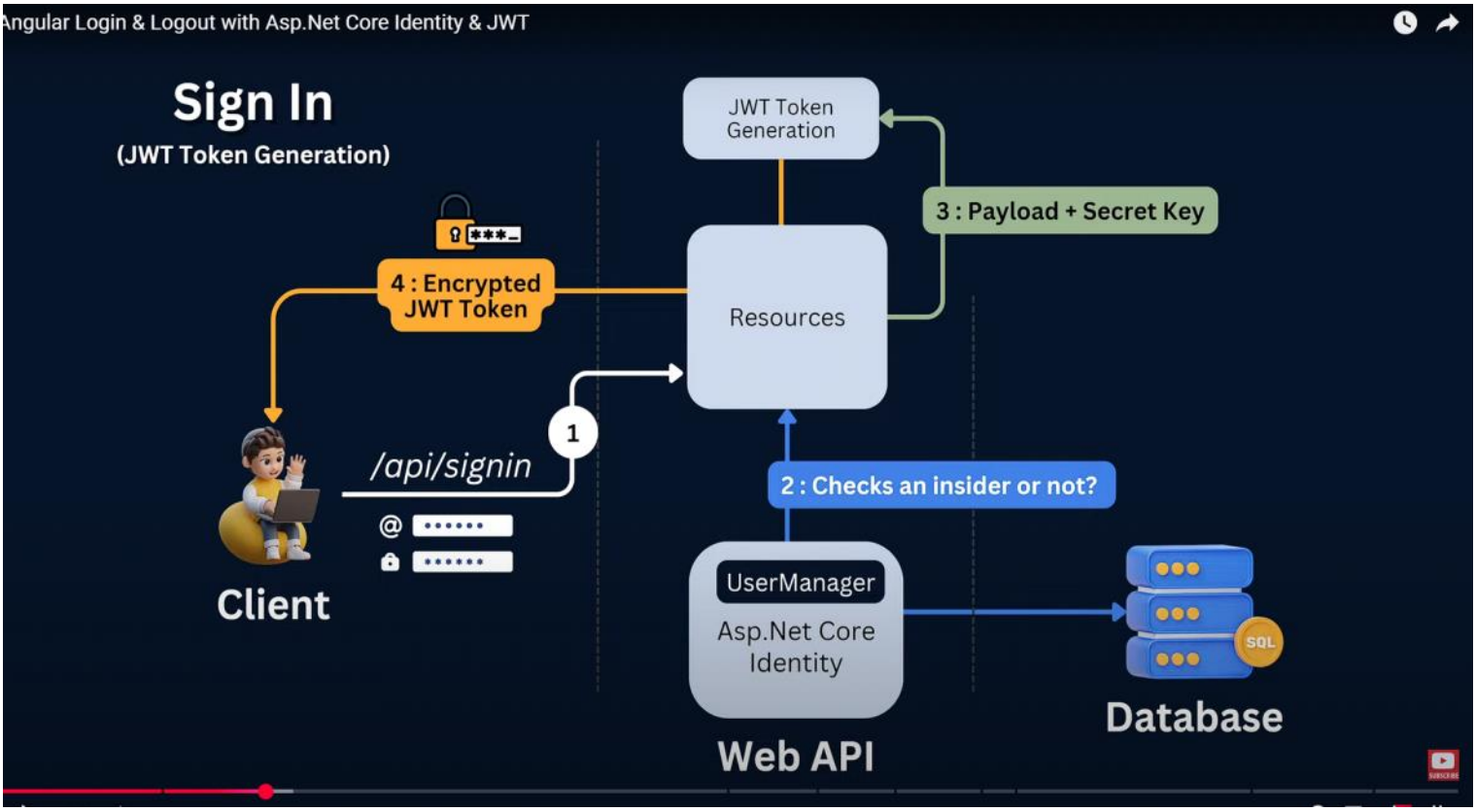

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECClient
login.component.html
registration.component.html
user.component.html
user.component.ts
app.component.ts

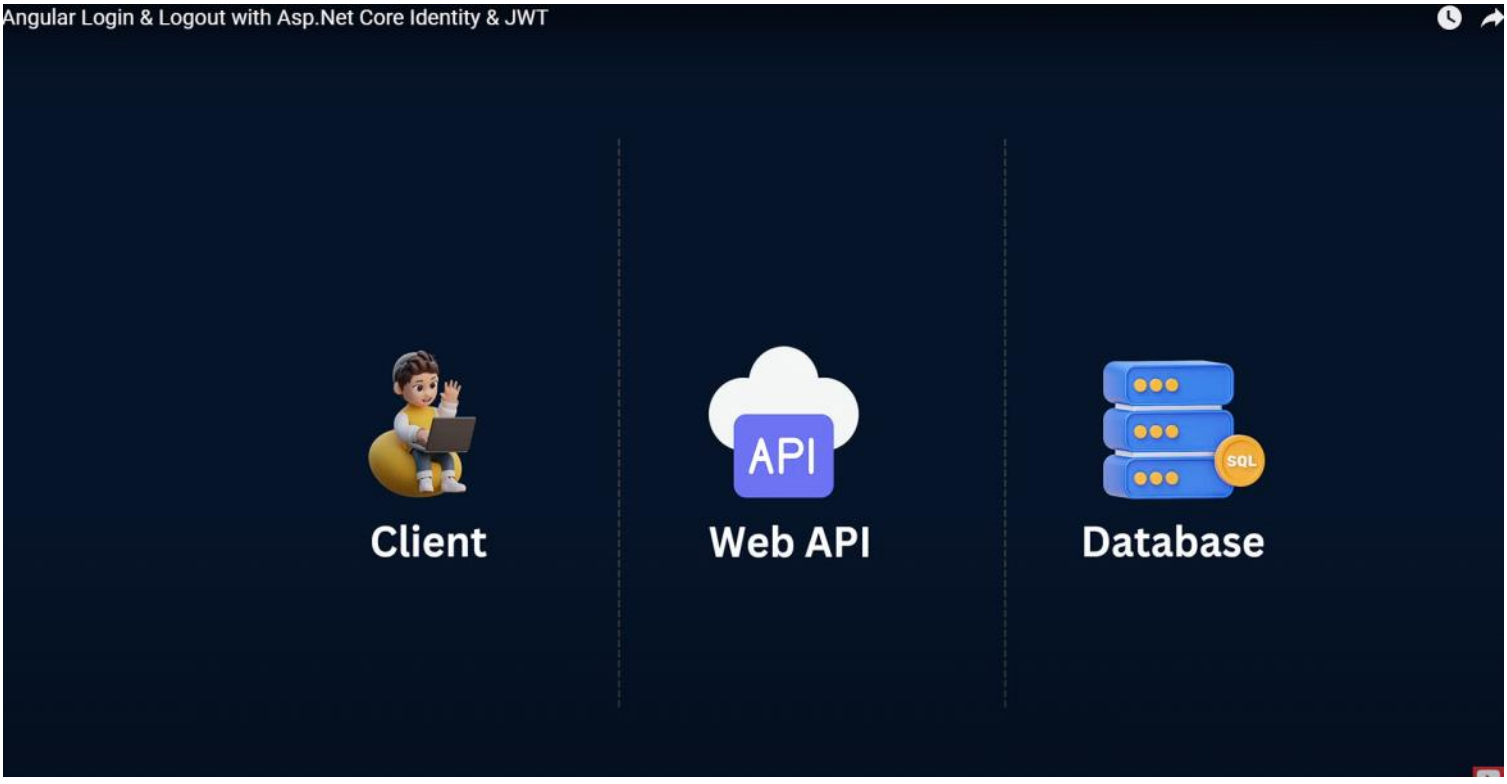
<div class="col-md-5 d-md-flex d-none bg-success rounded-start-4
align-items-center justify-content-center">

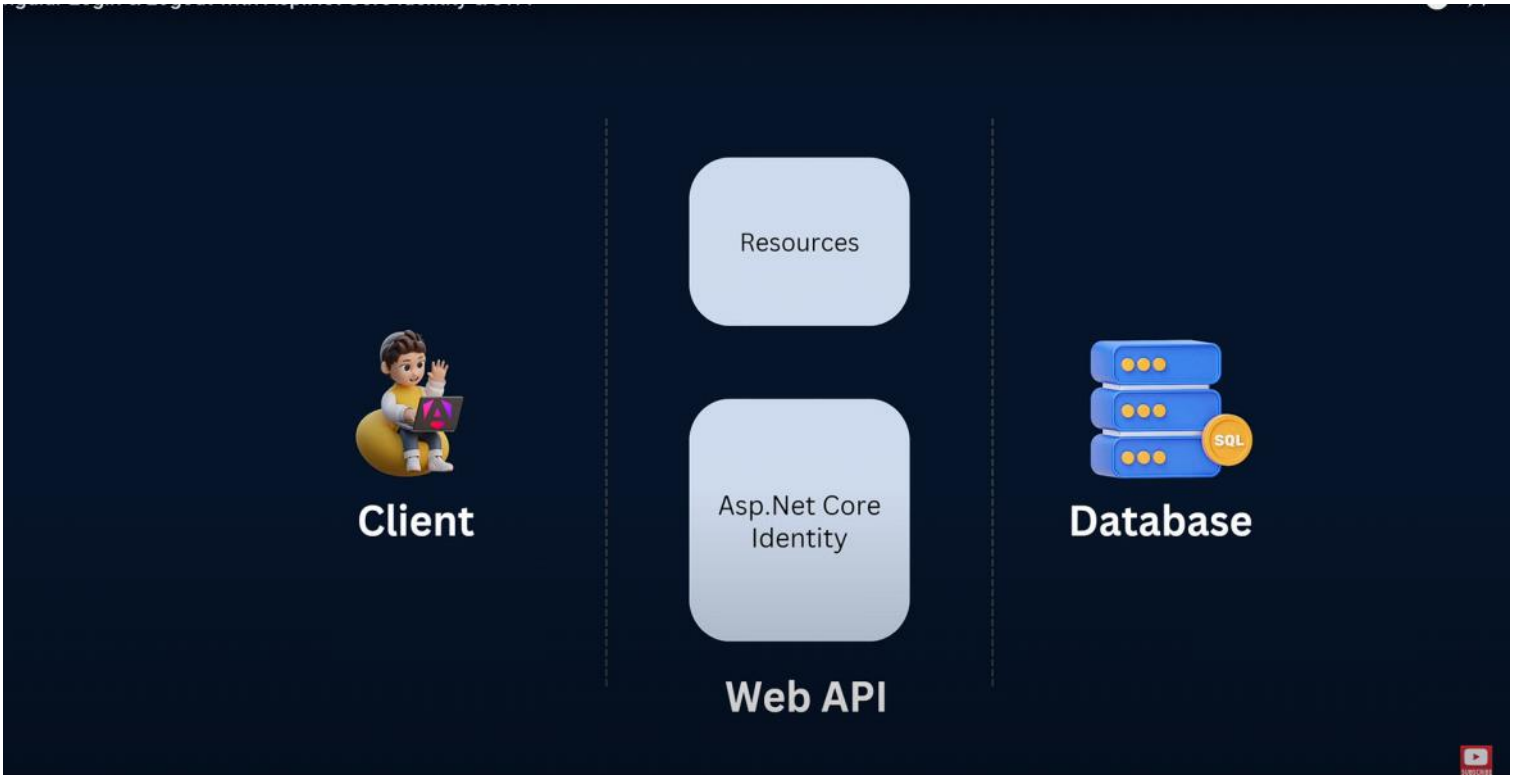
<img src="authec-logo.png" class="w-50 mb-3">
<h2 class="fw-bolder">Auth EC</h2>

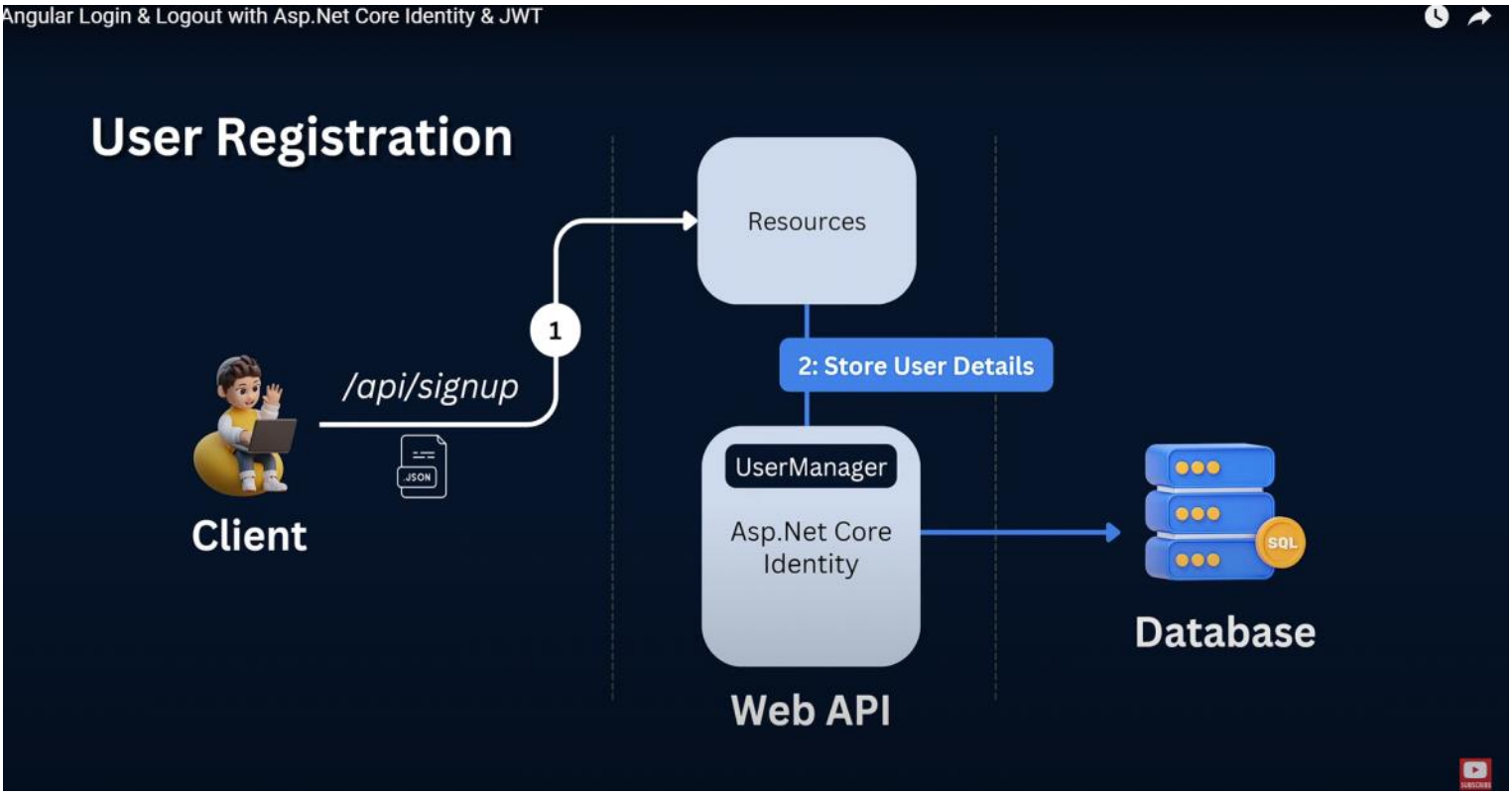
<router-outlet />

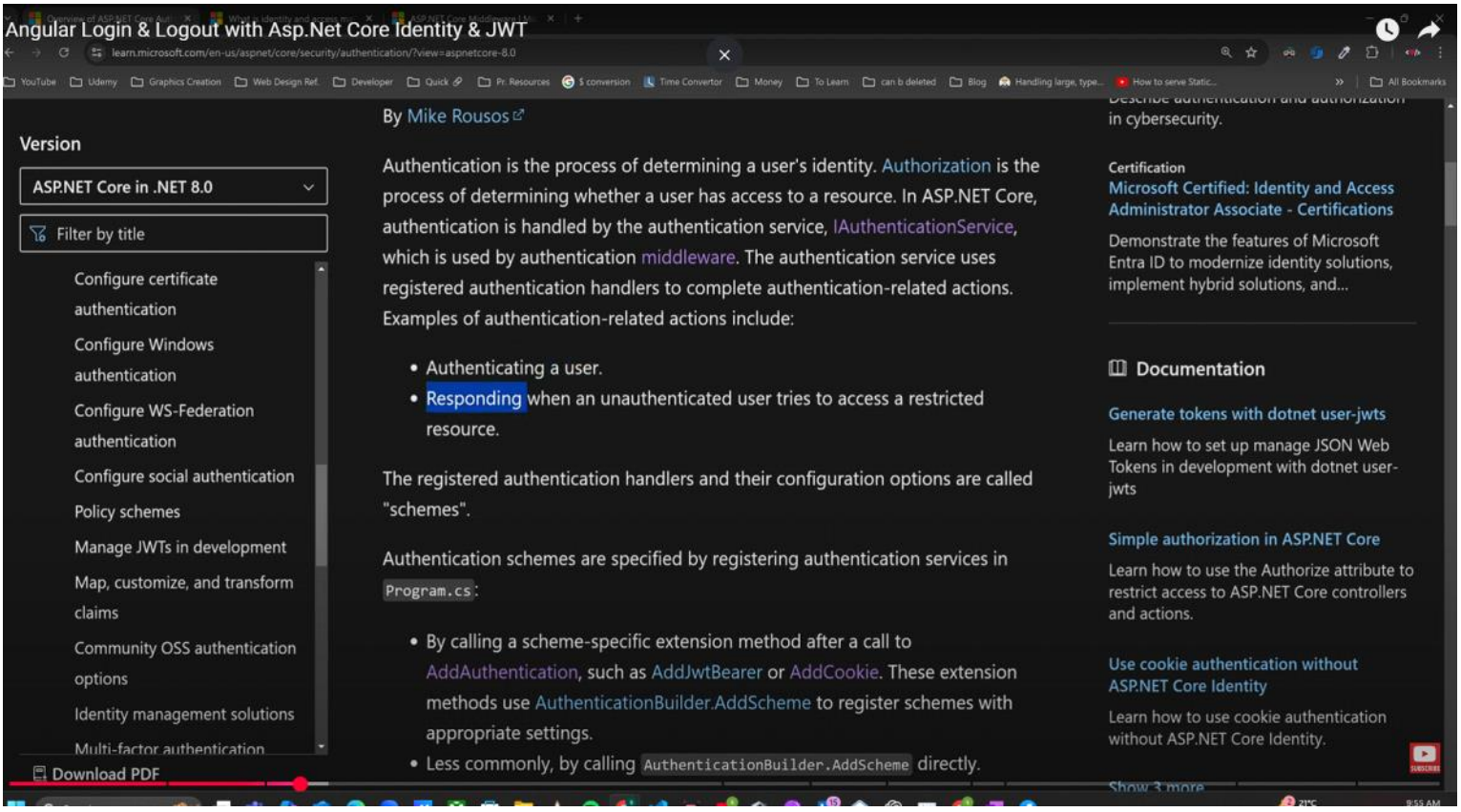
18:12 / 1:42:40 • Login Form Validation ›

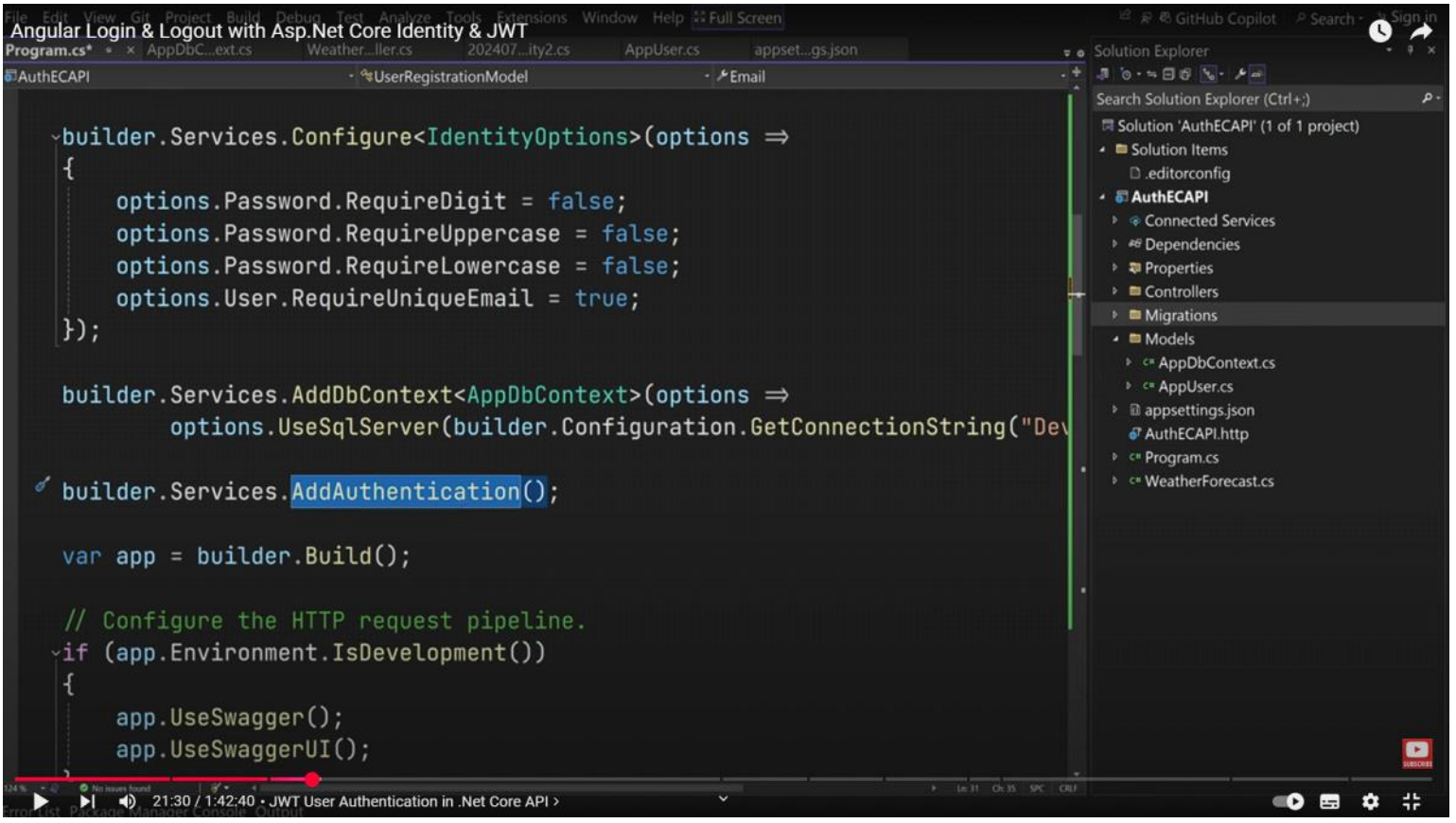

Angular Login & Logout with Asp.Net Core Identity & JWT
Sign In
(JWT Token Generation)
JWT Token Generation
3 : Payload + Secret Key
4 : Encrypted JWT Token
Resources
1
/api/signin
2 : Checks an insider or not?
Client
UserManager
Asp.Net Core Identity
Web API
SQL
Database

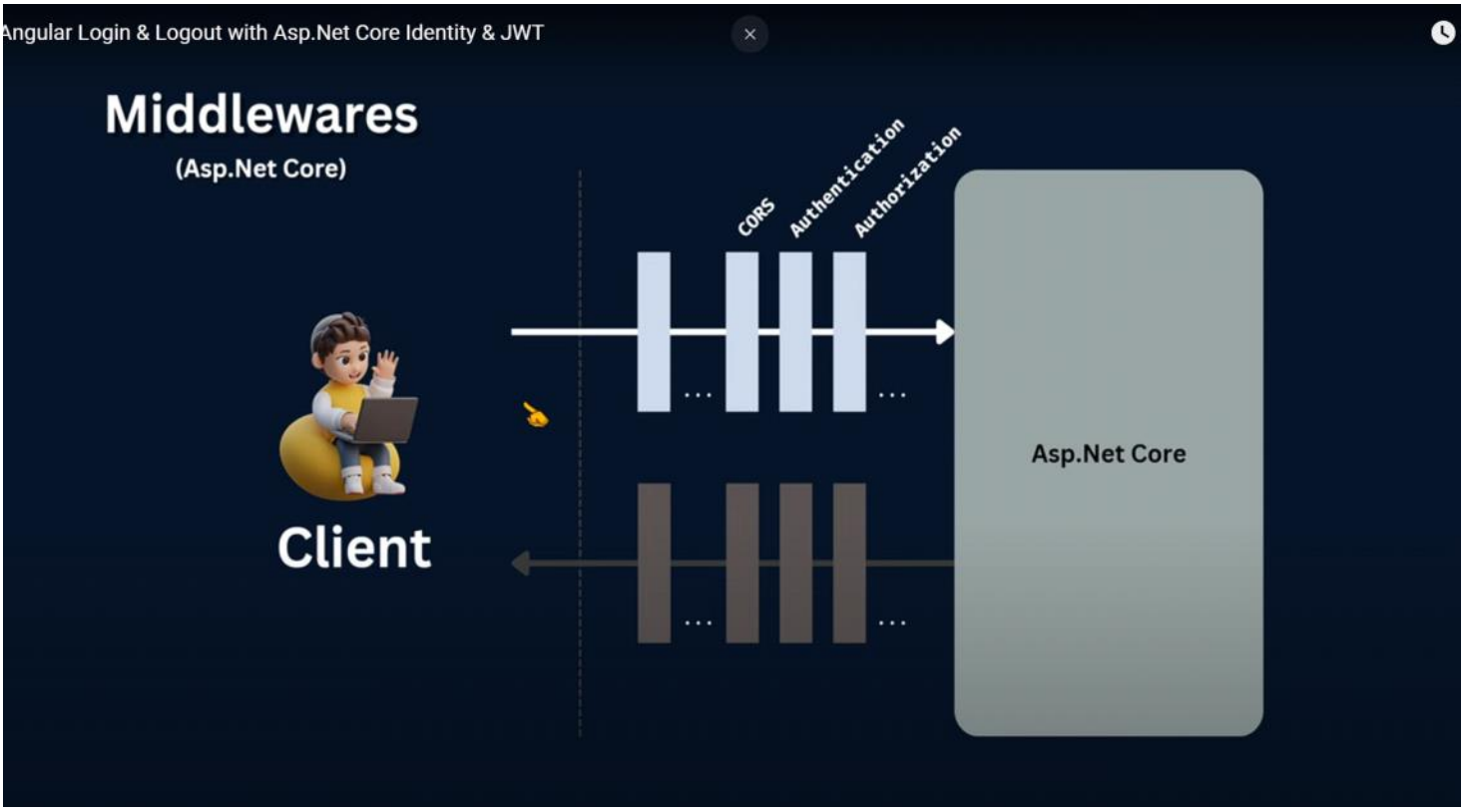

Angular Login & Logout with Asp.Net Core Identity & JWT
API
SQL
Client
Web API
Database

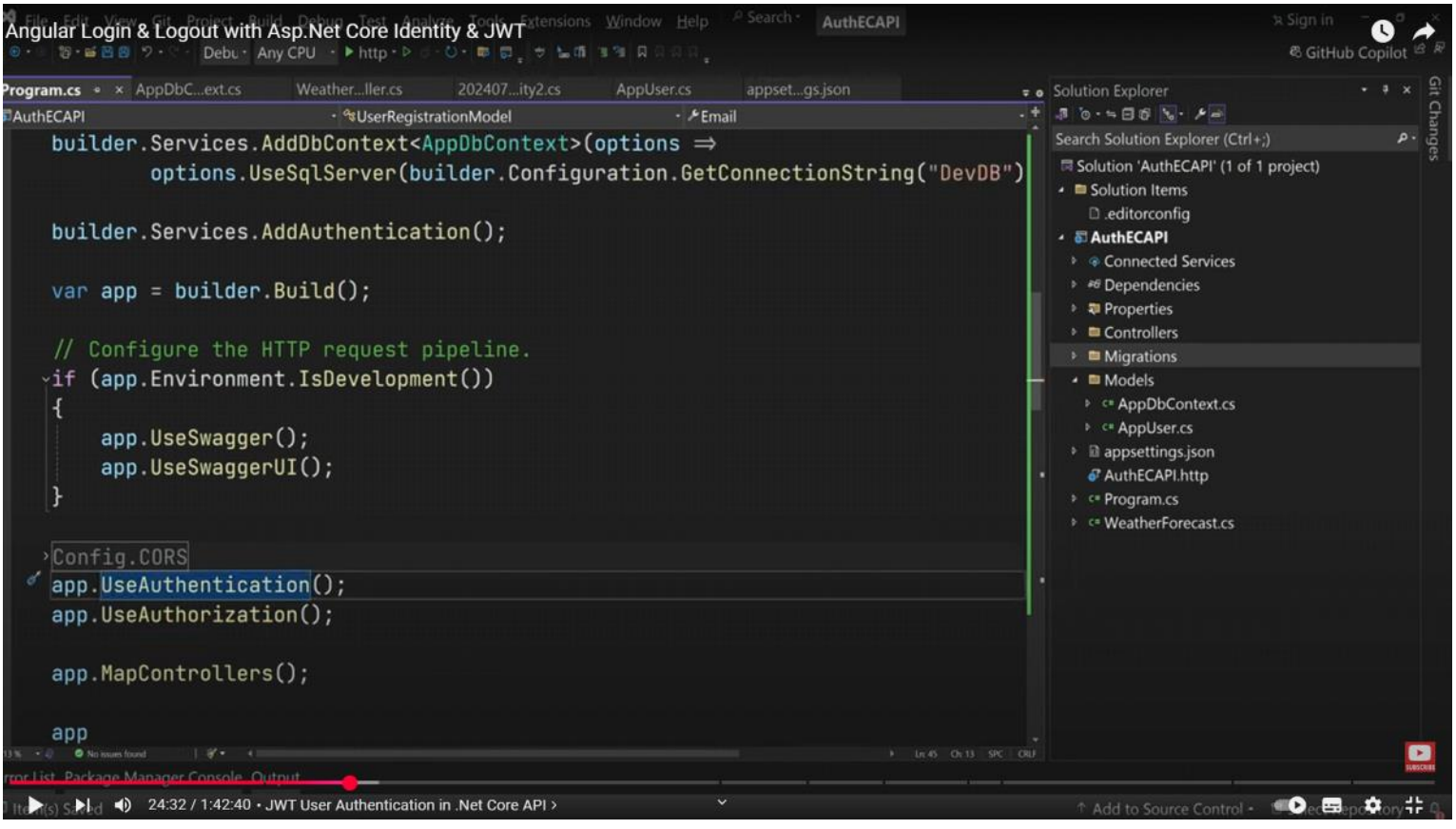

Resources
SQL
Client
Asp.Net Core Identity
Database
Web API

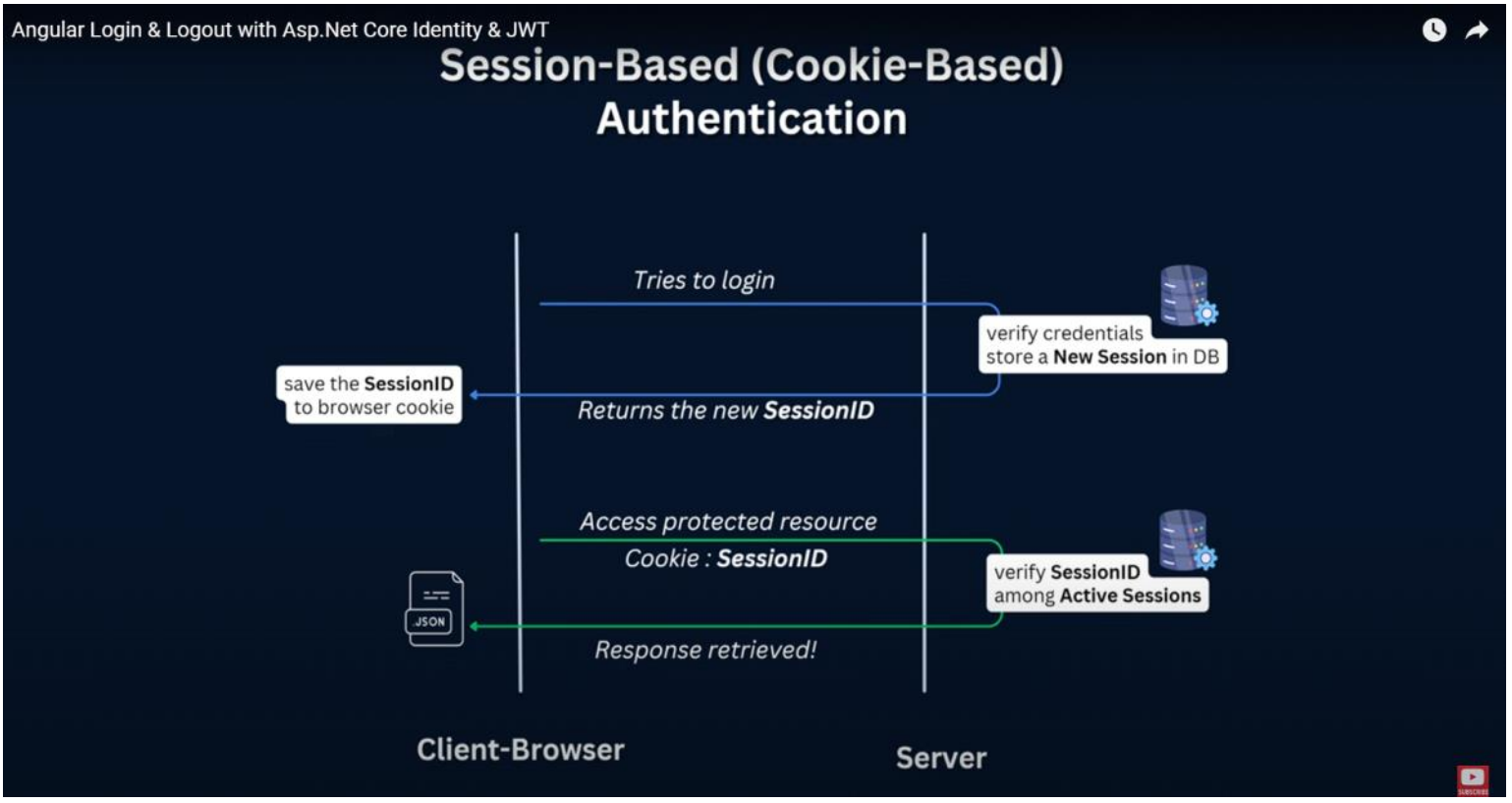

Angular Login & Logout with Asp.Net Core Identity & JWT
User Registration
Resources
1
/api/signup
JSON
Client
2: Store User Details
UserManager
Asp.Net Core Identity
Web API
SQL
Database

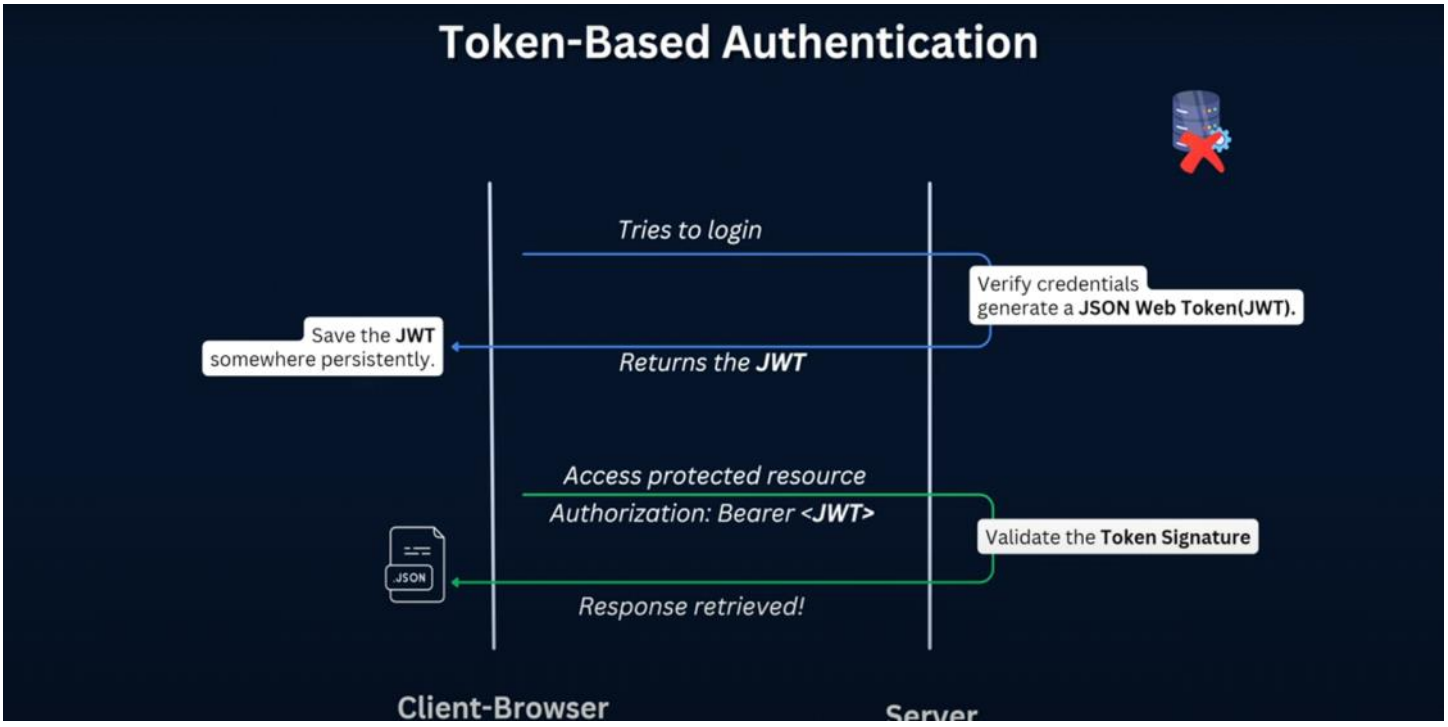

Angular Login & Logout with Asp.Net Core Identity & JWT
Version
ASP.NET Core in .NET 8.0
Filter by title
Configure certificate authentication
Configure Windows authentication
Configure WS-Federation authentication
Configure social authentication
Policy schemes
Manage JWTs in development
Map, customize, and transform claims
Community OSS authentication options
Identity management solutions
Multi-factor authentication
Download PDF
By Mike Rousos
Authentication is the process of determining a user's identity. Authorization is the process of determining whether a user has access to a resource. In ASP.NET Core, authentication is handled by the authentication service, IAuthenticationService, which is used by authentication middleware. The authentication service uses registered authentication handlers to complete authentication-related actions. Examples of authentication-related actions include:
Authenticating a user.
Responding when an unauthenticated user tries to access a restricted resource.
The registered authentication handlers and their configuration options are called "schemes".
Authentication schemes are specified by registering authentication services in Program.cs:
By calling a scheme-specific extension method after a call to AddAuthentication, such as AddJwtBearer or AddCookie. These extension methods use AuthenticationBuilder.AddScheme to register schemes with appropriate settings.
Less commonly, by calling AuthenticationBuilder.AddScheme directly.
in cybersecurity.
Certification
Microsoft Certified: Identity and Access Administrator Associate - Certifications
Demonstrate the features of Microsoft Entra ID to modernize identity solutions, implement hybrid solutions, and...
Documentation
Generate tokens with dotnet user-jwts
Learn how to set up manage JSON Web Tokens in development with dotnet user-jwts
Simple authorization in ASP.NET Core
Learn how to use the Authorize attribute to restrict access to ASP.NET Core controllers and actions.
Use cookie authentication without ASP.NET Core Identity
Learn how to use cookie authentication without ASP.NET Core Identity.
Show 3 more

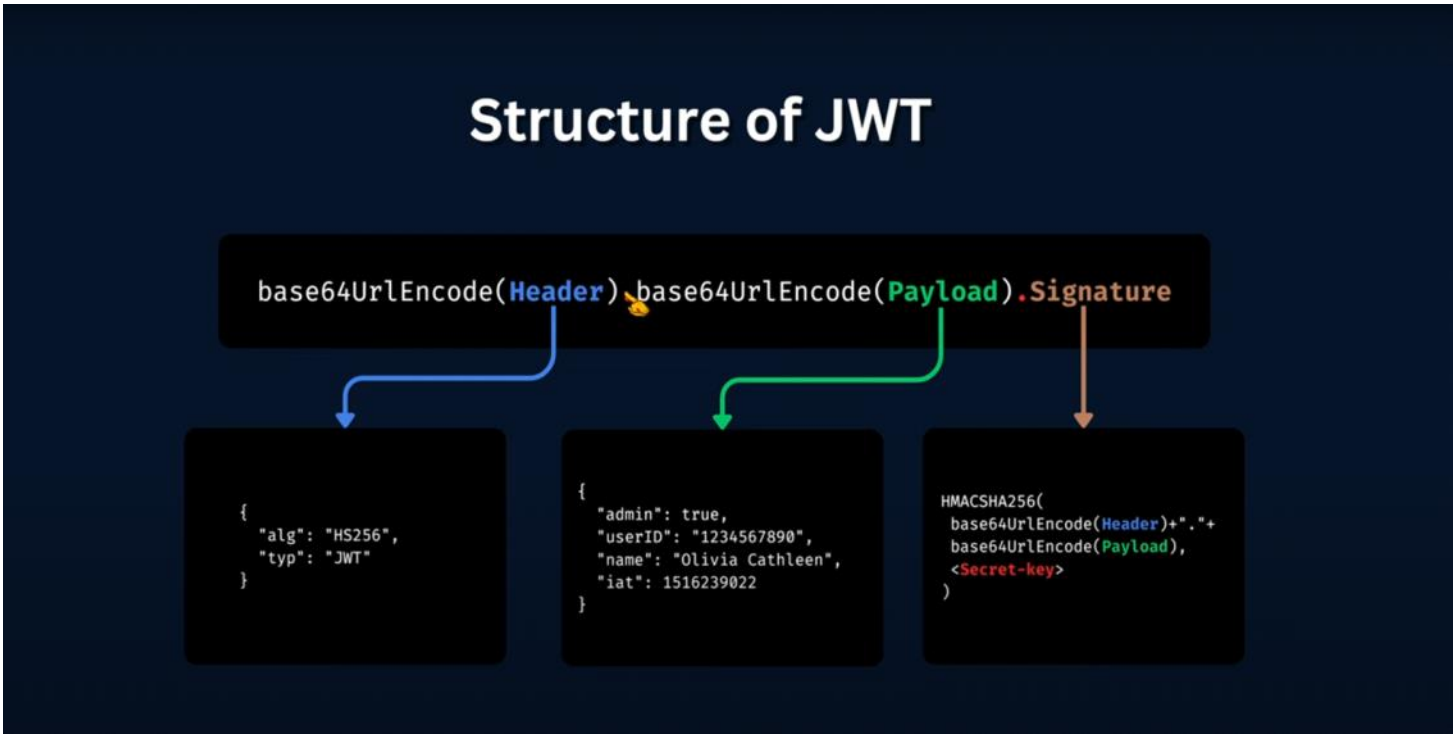

Angular Login & Logout with Asp.Net Core Identity & JWT
builder.Services.Configure<IdentityOptions>(options =>
{
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("Dev
builder.Services.AddAuthentication();
var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
Solution Explorer
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
.editorconfig
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Migrations
Models
AppDbContext.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
21:30 / 1:42:40 • JWT User Authentication in .Net Core API

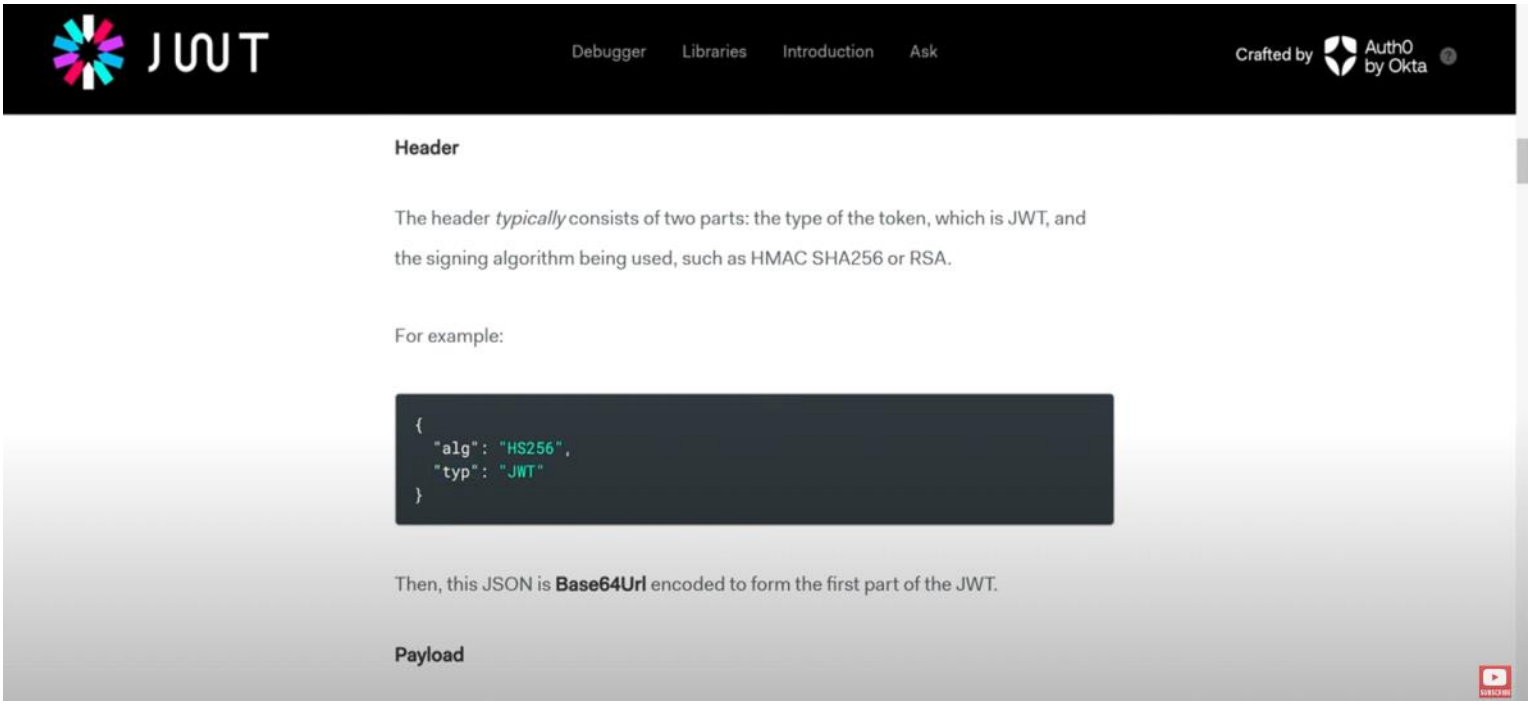

Angular Login & Logout with Asp.Net Core Identity & JWT
Middlewares
(Asp.Net Core)
CORS
Authentication
Authorization
Client
Asp.Net Core

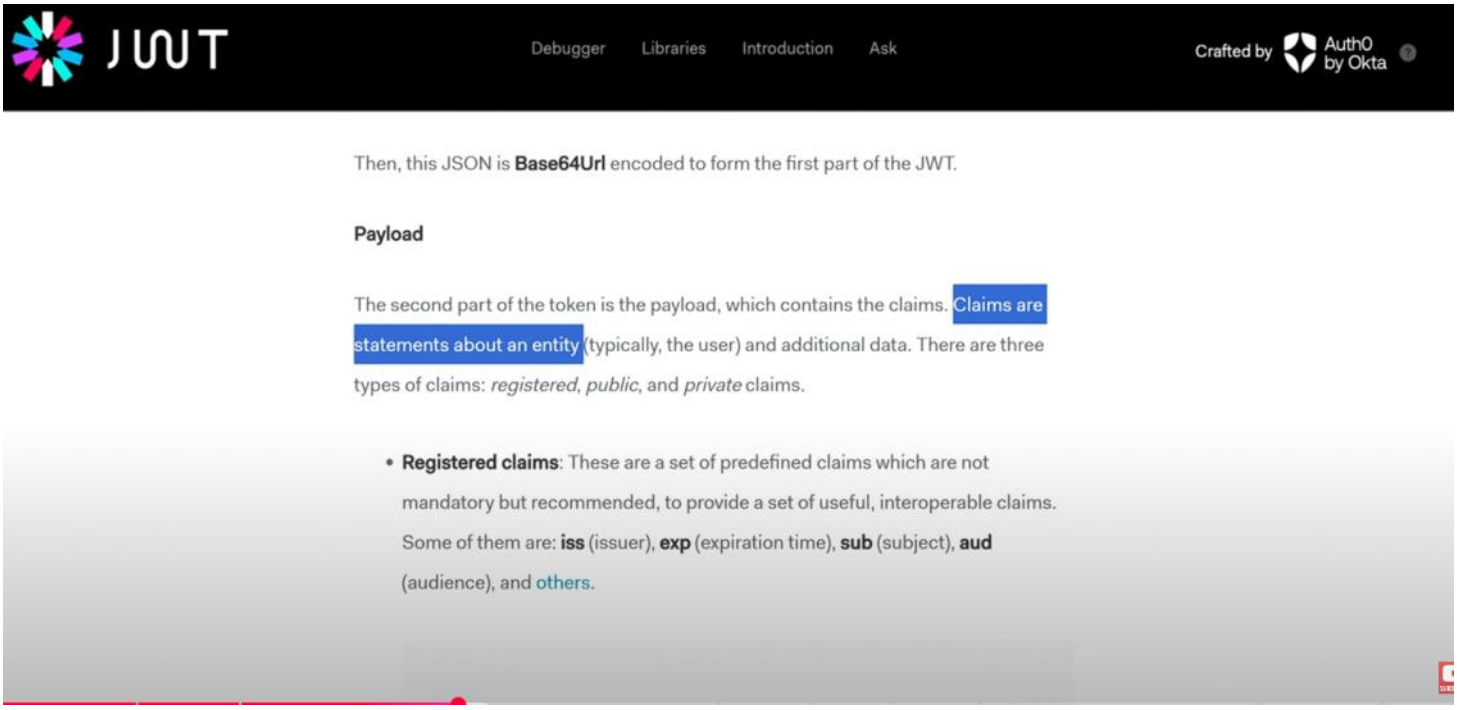

Angular Login & Logout with Asp.Net Core Identity & JWT
builder.Services.AddDbContext<AppDbContext>(options ⇒
options.UseSqlServer(builder.Configuration.GetConnectionString("DevDB")
builder.Services.AddAuthentication();
var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}
Config.CORS
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();
app
Solution Explorer
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
.editorconfig
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Migrations
Models
AppDbContext.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
24:32 / 1:42:40 • JWT User Authentication in .Net Core API ›

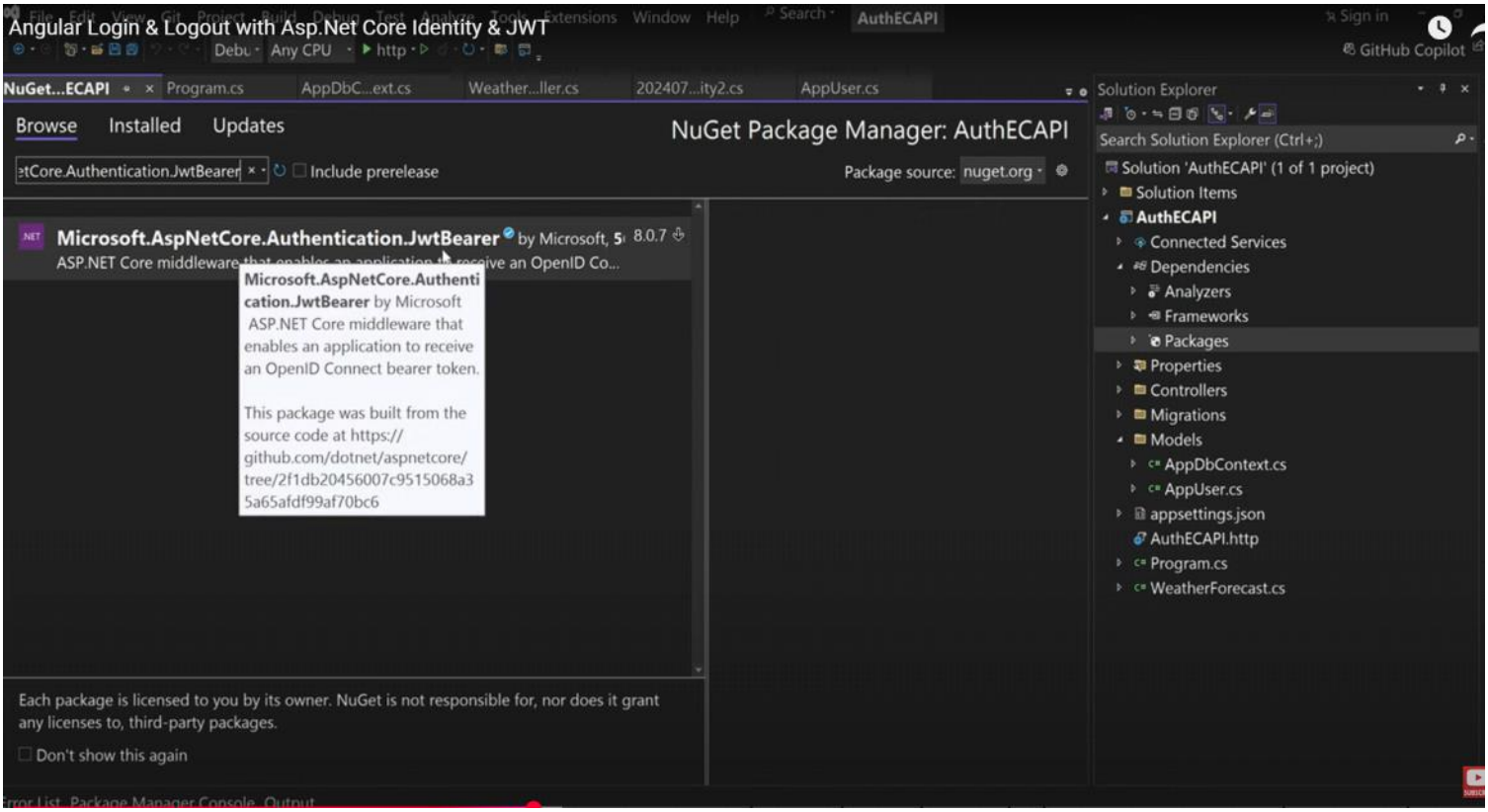

Angular Login & Logout with Asp.Net Core Identity & JWT
Session-Based (Cookie-Based) Authentication
Tries to login
verify credentials
store a New Session in DB
save the SessionID
to browser cookie
Returns the new SessionID
Access protected resource
Cookie : SessionID
verify SessionID
among Active Sessions
JSON
Response retrieved!
Client-Browser
Server

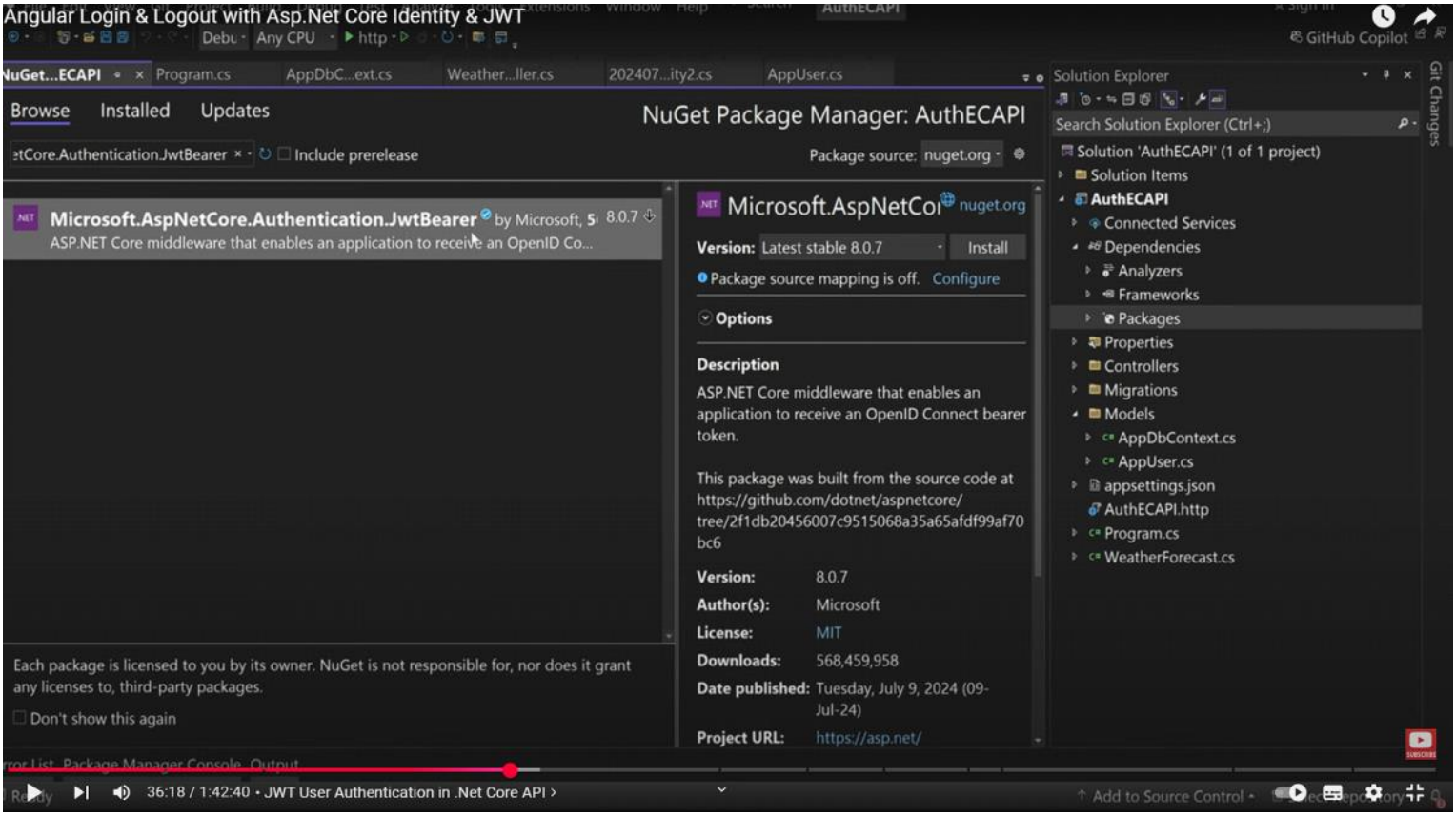

Token-Based Authentication
Tries to login
Verify credentials
generate a JSON Web Token(JWT).
Save the JWT
somewhere persistently.
Returns the JWT
Access protected resource
Authorization: Bearer <JWT>
Validate the Token Signature
JSON
Response retrieved!
Client-Browser
Server

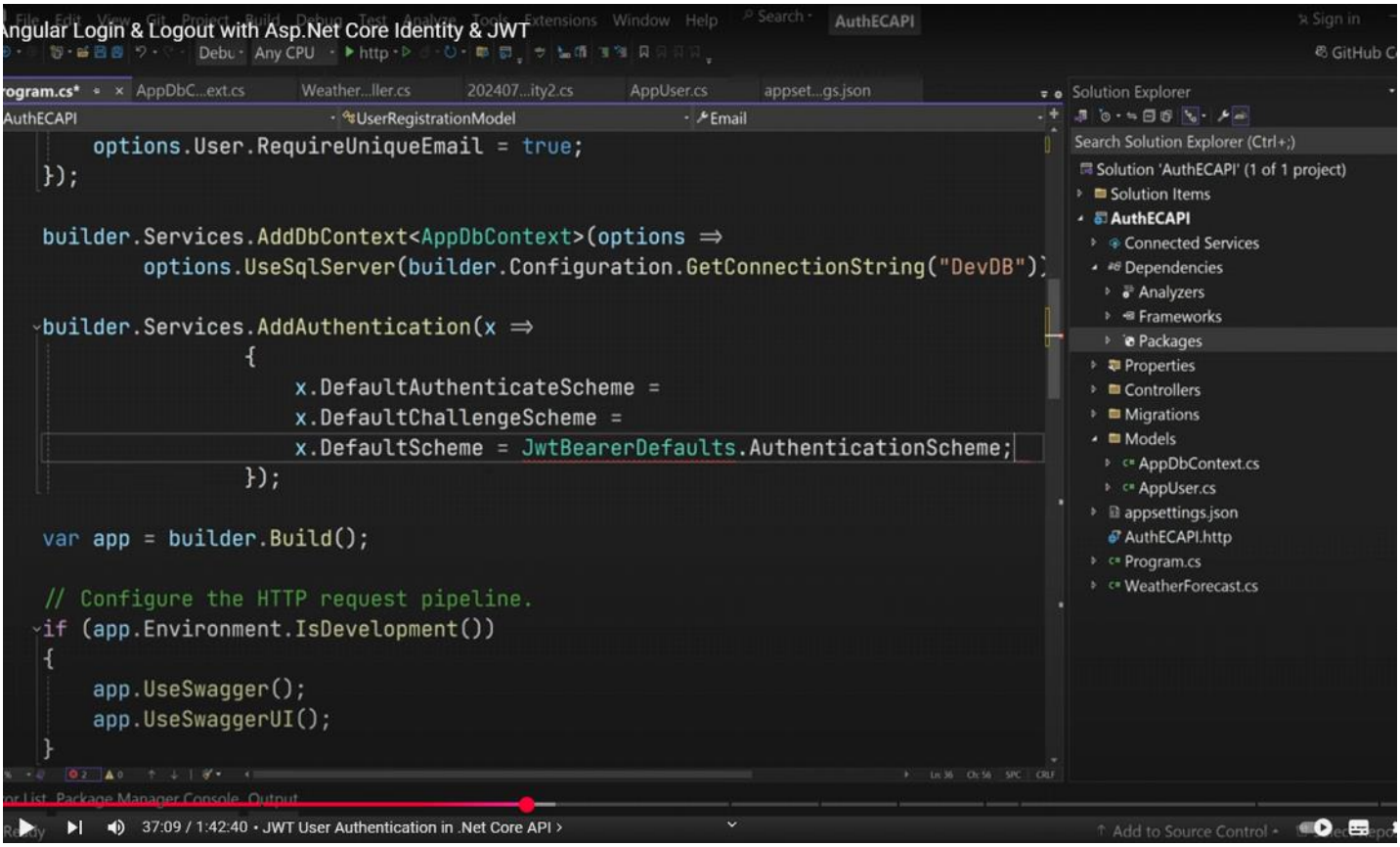

Structure of JWT
base64UrlEncode(Header).base64UrlEncode(Payload).Signature
{
"alg": "HS256",
"typ": "JWT"
}
{
"admin": true,
"userID": "1234567890",
"name": "Olivia Cathleen",
"iat": 1516239022
}
HMACSHA256(
base64UrlEncode(Header)+"."+
base64UrlEncode(Payload),
<Secret-key>
)

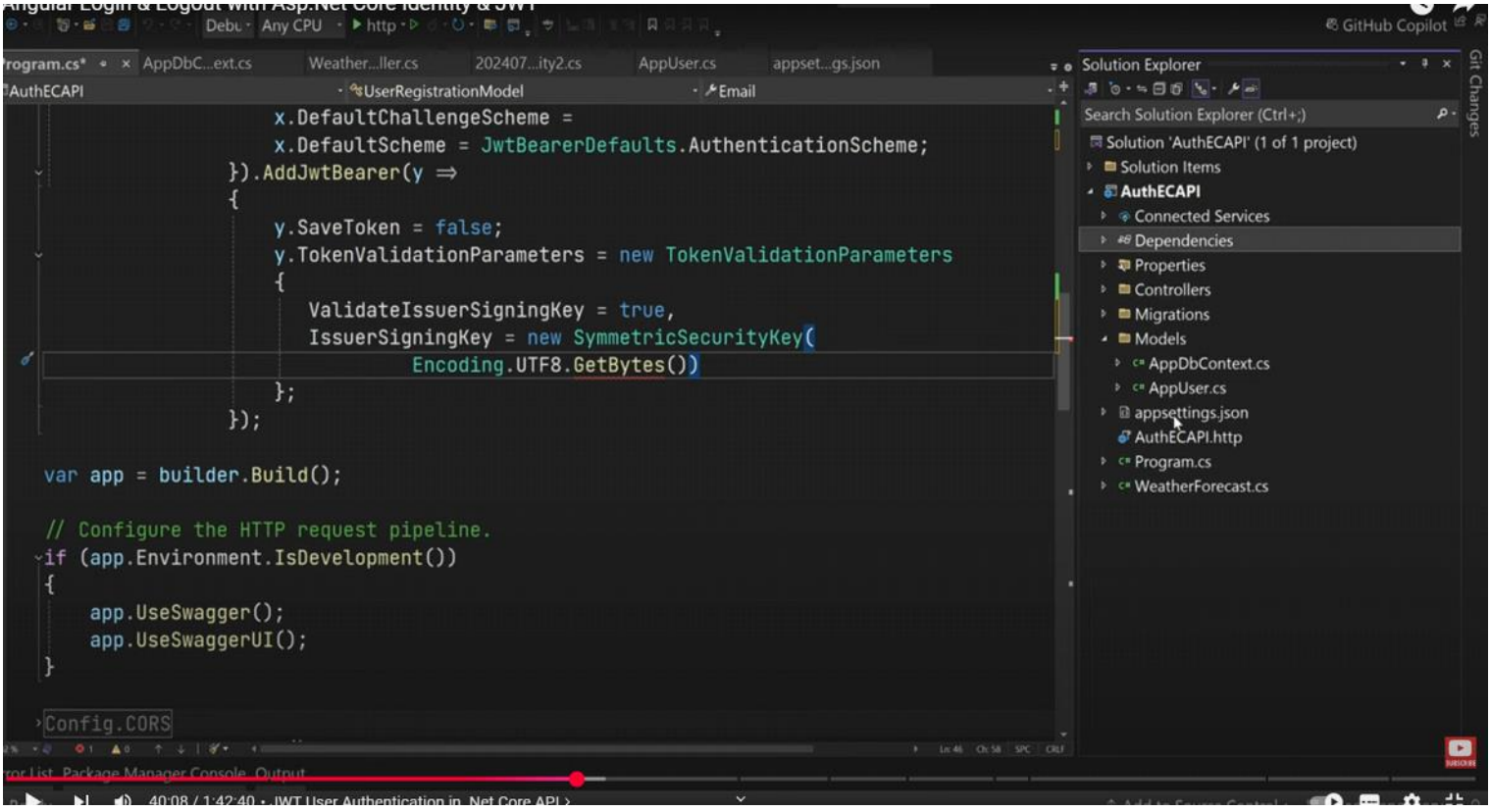

JWT
Debugger
Libraries
Introduction
Ask
Crafted by
Auth0 by Okta
Header
The header typically consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA.
For example:
{
"alg": "HS256",
"typ": "JWT"
}
Then, this JSON is Base64Url encoded to form the first part of the JWT.
Payload

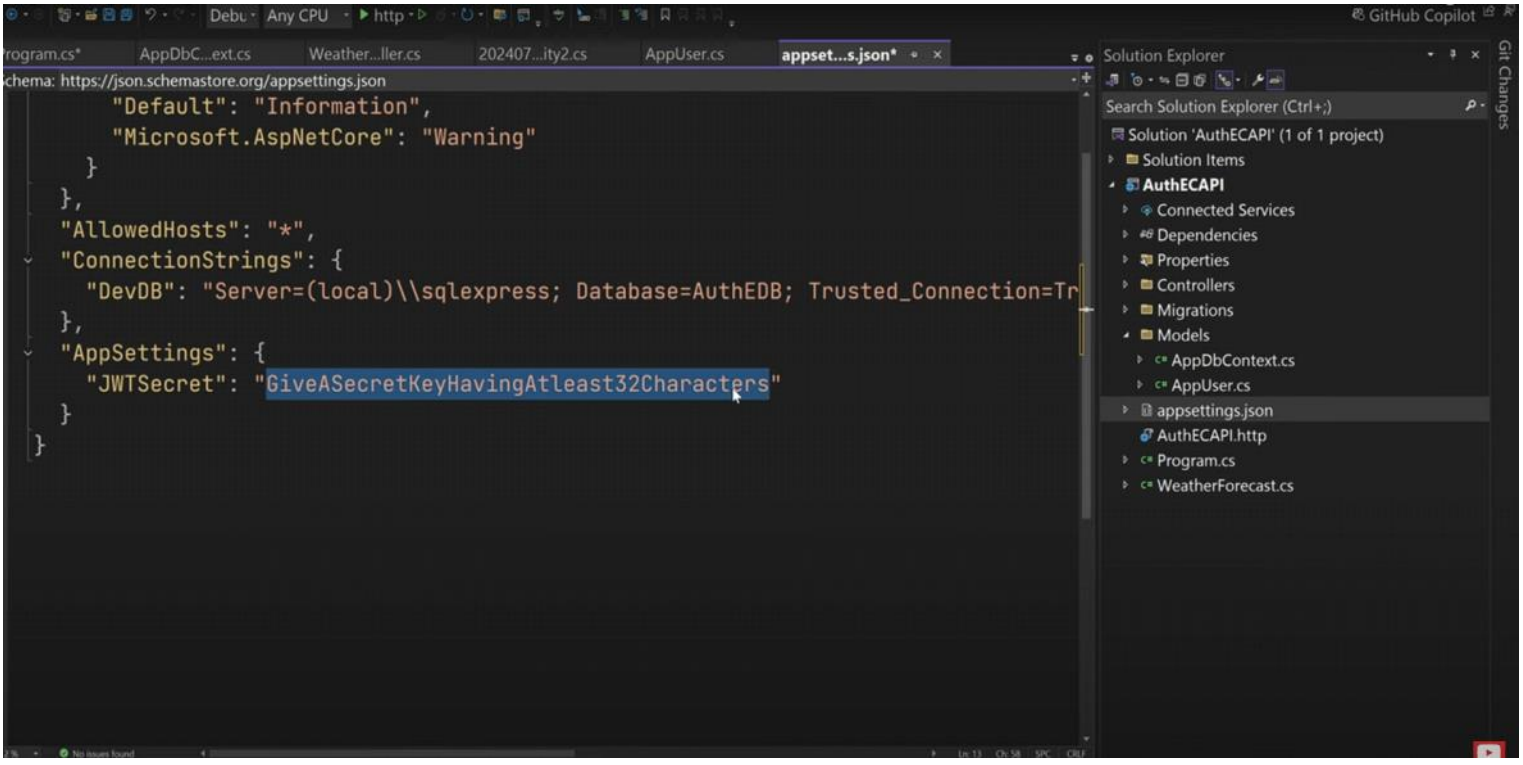

JWT
Debugger
Libraries
Introduction
Ask
Crafted by Auth0 by Okta
Then, this JSON is Base64Url encoded to form the first part of the JWT.
Payload
The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims: registered, public, and private claims.
Registered claims: These are a set of predefined claims which are not mandatory but recommended, to provide a set of useful, interoperable claims. Some of them are: iss (issuer), exp (expiration time), sub (subject), aud (audience), and others.

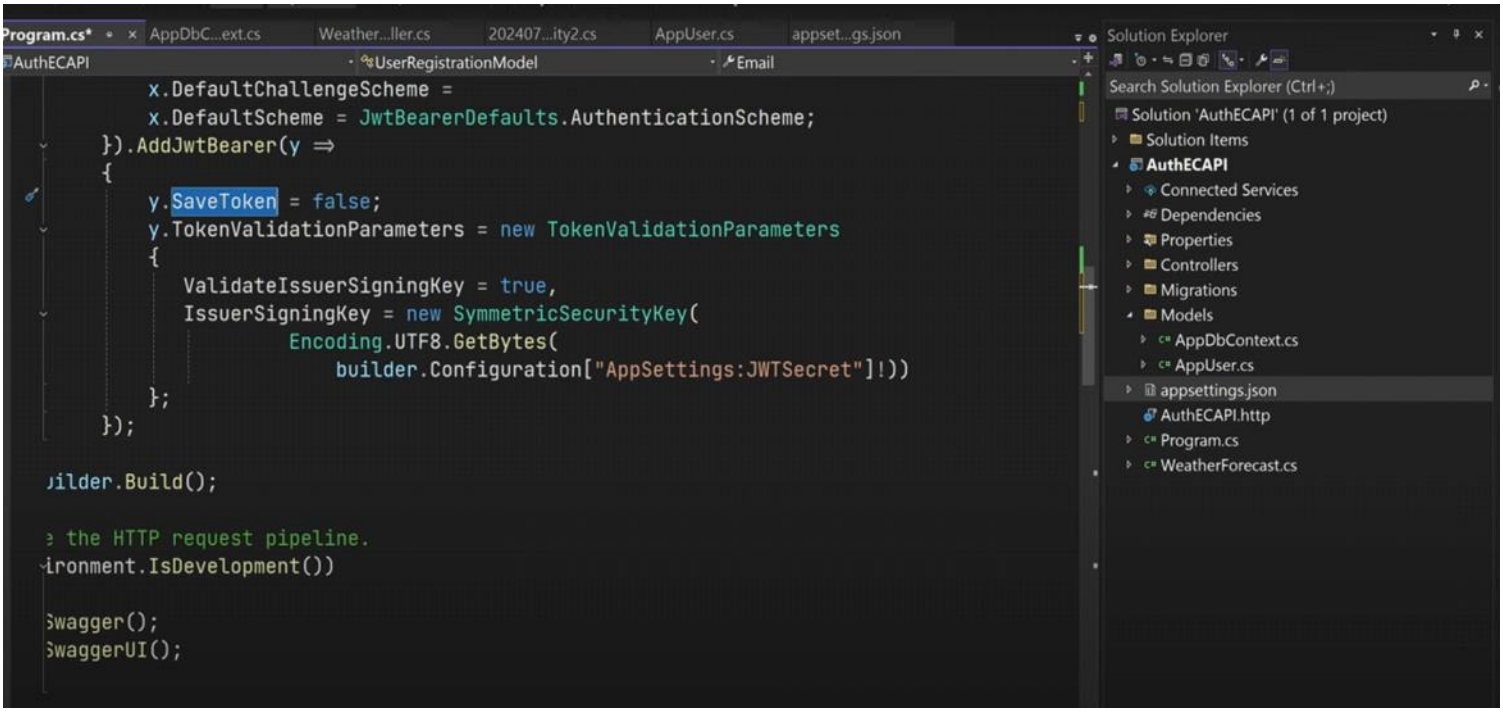

Angular Login & Logout with Asp.Net Core Identity & JWT
NuGet Package Manager: AuthECAPI
Browse
Installed
Updates
Include prerelease
Package source: nuget.org
Microsoft.AspNetCore.Authentication.JwtBearer by Microsoft
ASP.NET Core middleware that enables an application to receive an OpenID Co...
Microsoft.AspNetCore.Authentication.JwtBearer by Microsoft
ASP.NET Core middleware that enables an application to receive an OpenID Connect bearer token.
This package was built from the source code at https://github.com/dotnet/aspnetcore/tree/2f1db20456007c9515068a35a65afdf99af70bc6
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
Don't show this again
Solution Explorer
Solution 'AuthECAPI' (1 of 1 project)
AuthECAPI
Connected Services
Dependencies
Analyzers
Frameworks
Packages
Properties
Controllers
Migrations
Models
AppDbContext.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs


Angular Login & Logout with Asp.Net Core Identity & JWT
NuGet Package Manager: AuthECAPI
Browse
Installed
Updates
Include prerelease
Package source: nuget.org
Microsoft.AspNetCore.Authentication.JwtBearer by Microsoft, 5 8.0.7
ASP.NET Core middleware that enables an application to receive an OpenID Co...
Microsoft.AspNetCor nuget.org
Version: Latest stable 8.0.7
Install
Package source mapping is off. Configure
Options
Description
ASP.NET Core middleware that enables an application to receive an OpenID Connect bearer token.
This package was built from the source code at https://github.com/dotnet/aspnetcore/tree/2f1db20456007c9515068a35a65afdf99af70bc6
Version: 8.0.7
Author(s): Microsoft
License: MIT
Downloads: 568,459,958
Date published: Tuesday, July 9, 2024 (09-Jul-24)
Project URL: https://asp.net/
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
Don't show this again
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Analyzers
Frameworks
Packages
Properties
Controllers
Migrations
Models
AppDbContext.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
36:18 / 1:42:40 • JWT User Authentication in .Net Core API >


Angular Login & Logout with Asp.Net Core Identity & JWT
options.User.RequireUniqueEmail = true;
});
builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DevDB"));
builder.Services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
});
var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Analyzers
Frameworks
Packages
Properties
Controllers
Migrations
Models
AppDbContext.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
37:09 / 1:42:40 • JWT User Authentication in .Net Core API >

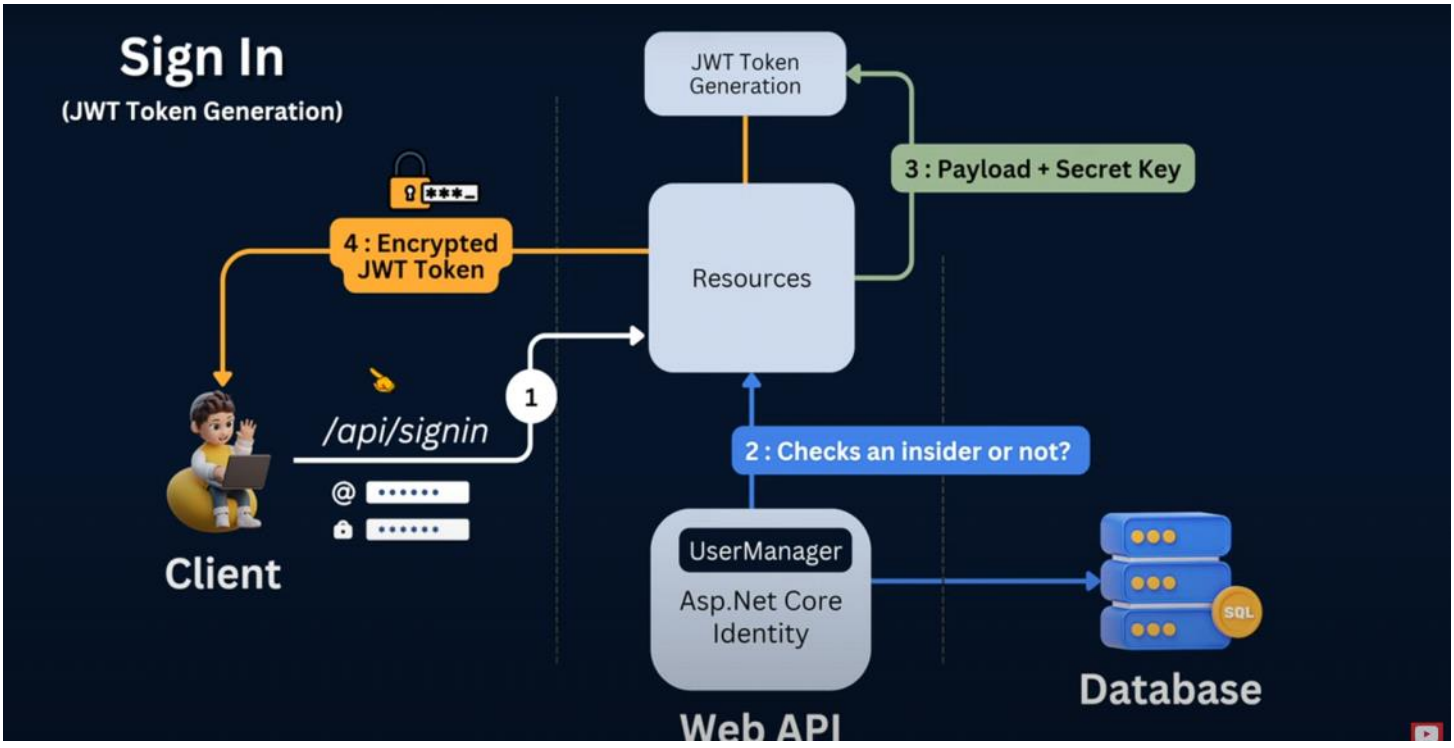

Angular Login & Logout with Asp.Net Core Identity & JWT
GitHub Copilot
Program.cs*
AppDbC...ext.cs
Weather...ller.cs
202407...ity2.cs
AppUser.cs
appset...gs.json
AuthECAPI
UserRegistrationModel
Email
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Encoding.UTF8.GetBytes())
};
});
var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}
Config.CORS
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Migrations
Models
AppDbContext.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
Git Changes
Error List
Package Manager Console
Output
40:08 / 1:42:40 • JWT User Authentication in .Net Core API


GitHub Copilot
Program.cs*
AppDbC...ext.cs
Weather...ller.cs
202407...ity2.cs
AppUser.cs
appset...s.json*
Schema: https://json.schemastore.org/appsettings.json
"Default": "Information",
"Microsoft.AspNetCore": "Warning"
}
},
"AllowedHosts": "*",
"ConnectionStrings": {
"DevDB": "Server=(local)\\sqlexpress; Database=AuthEDB; Trusted_Connection=Tr
},
"AppSettings": {
"JWTSecret": "GiveASecretKeyHavingAtleast32Characters"
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Migrations
Models
AppDbContext.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
Git Changes



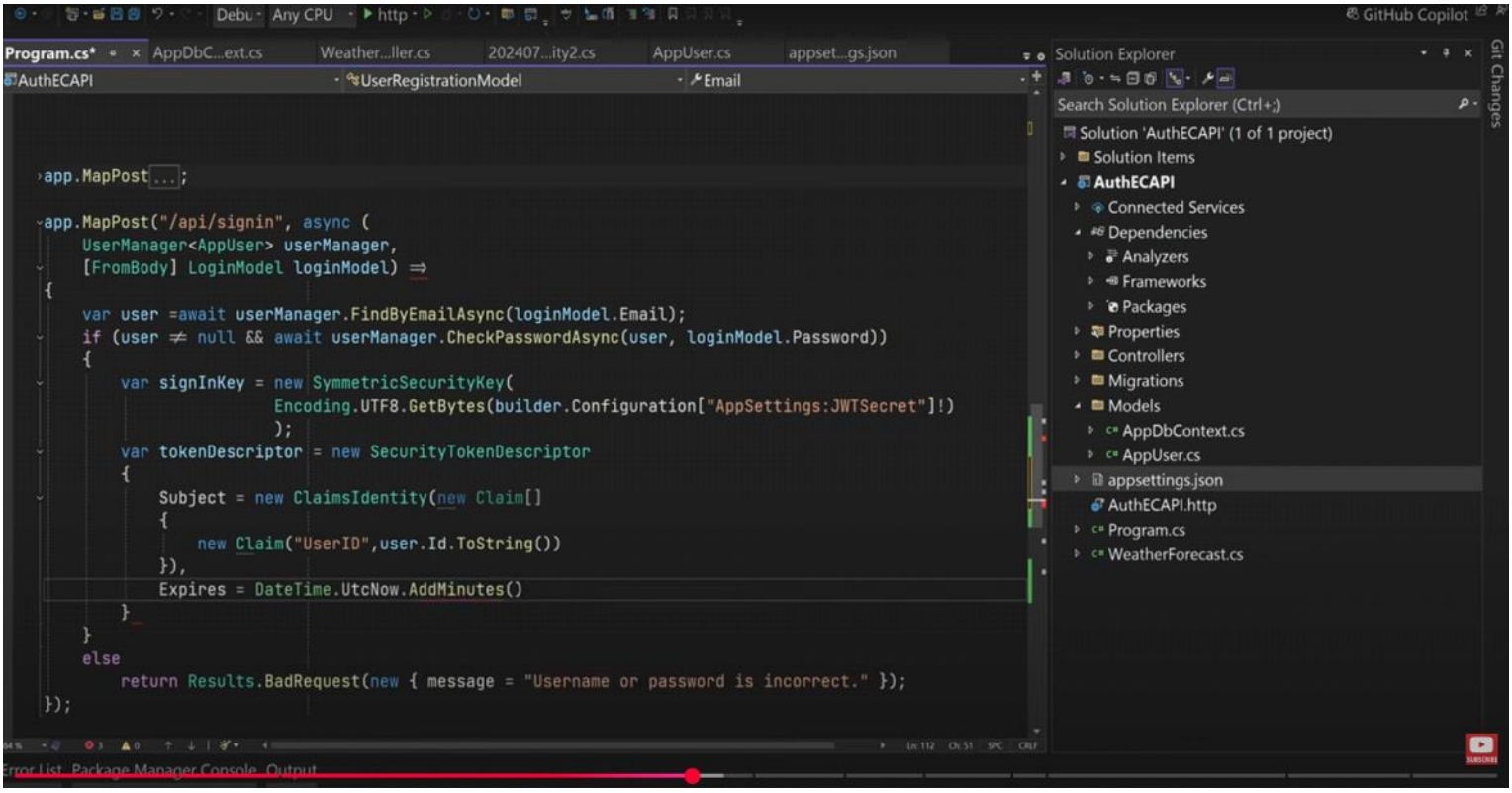
.

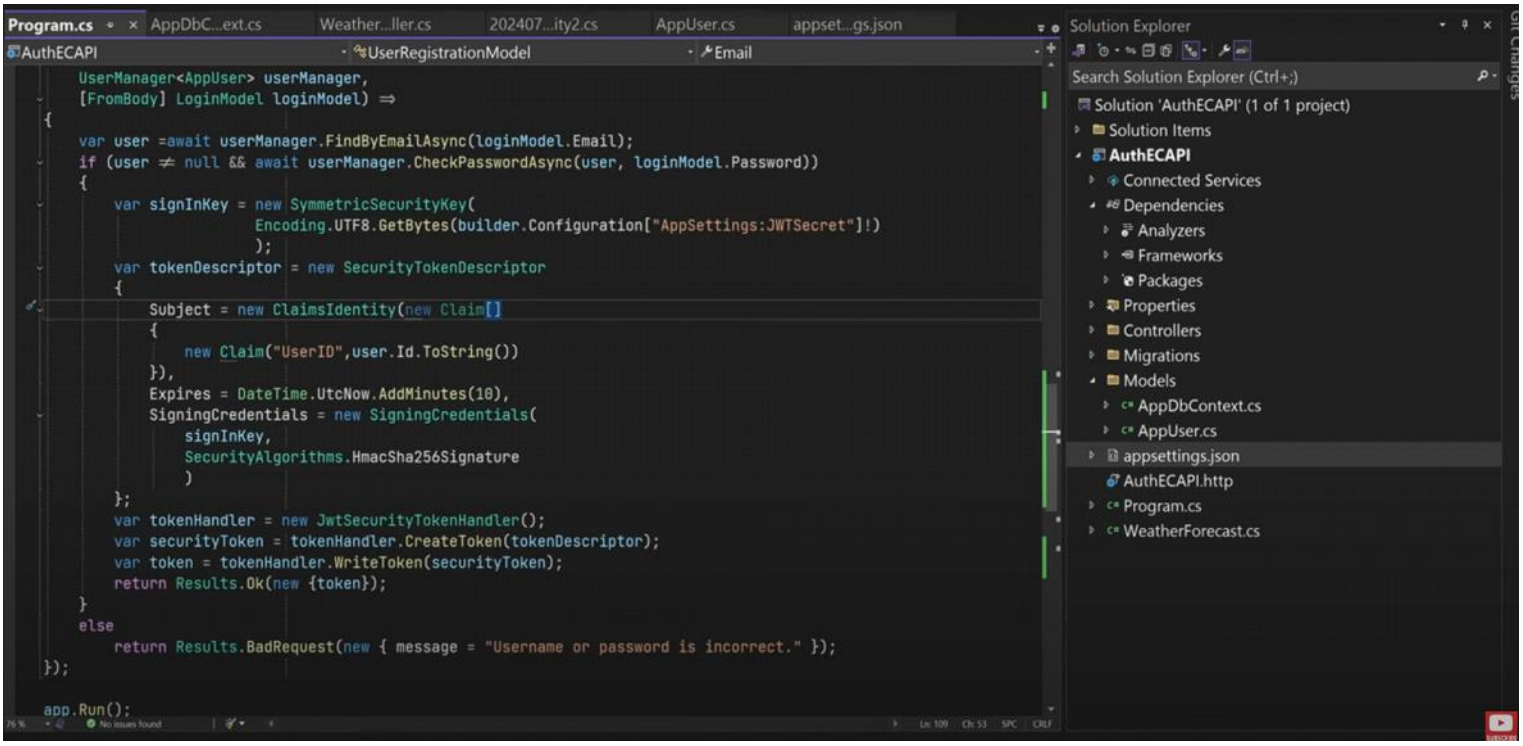
.

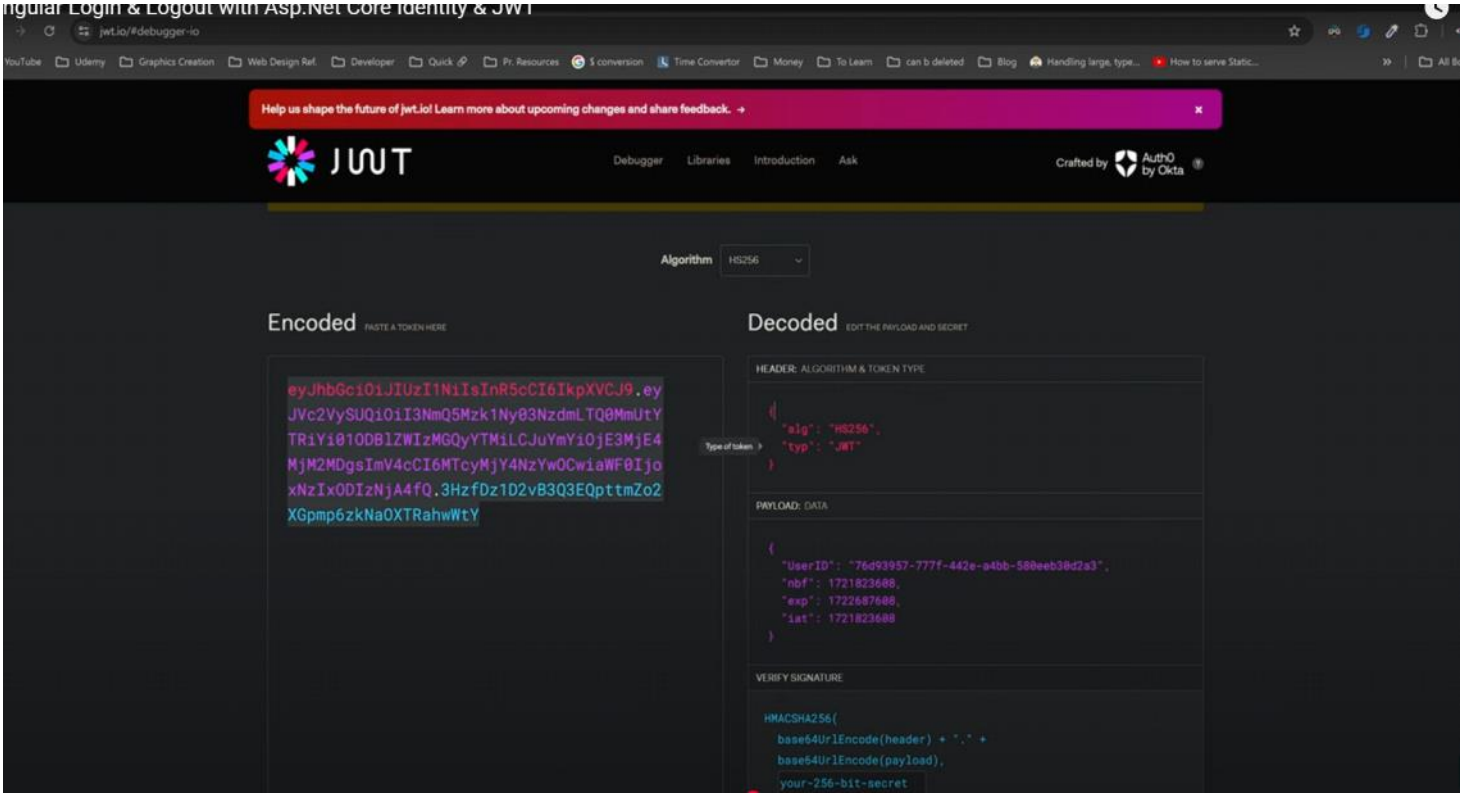


In programe.cs

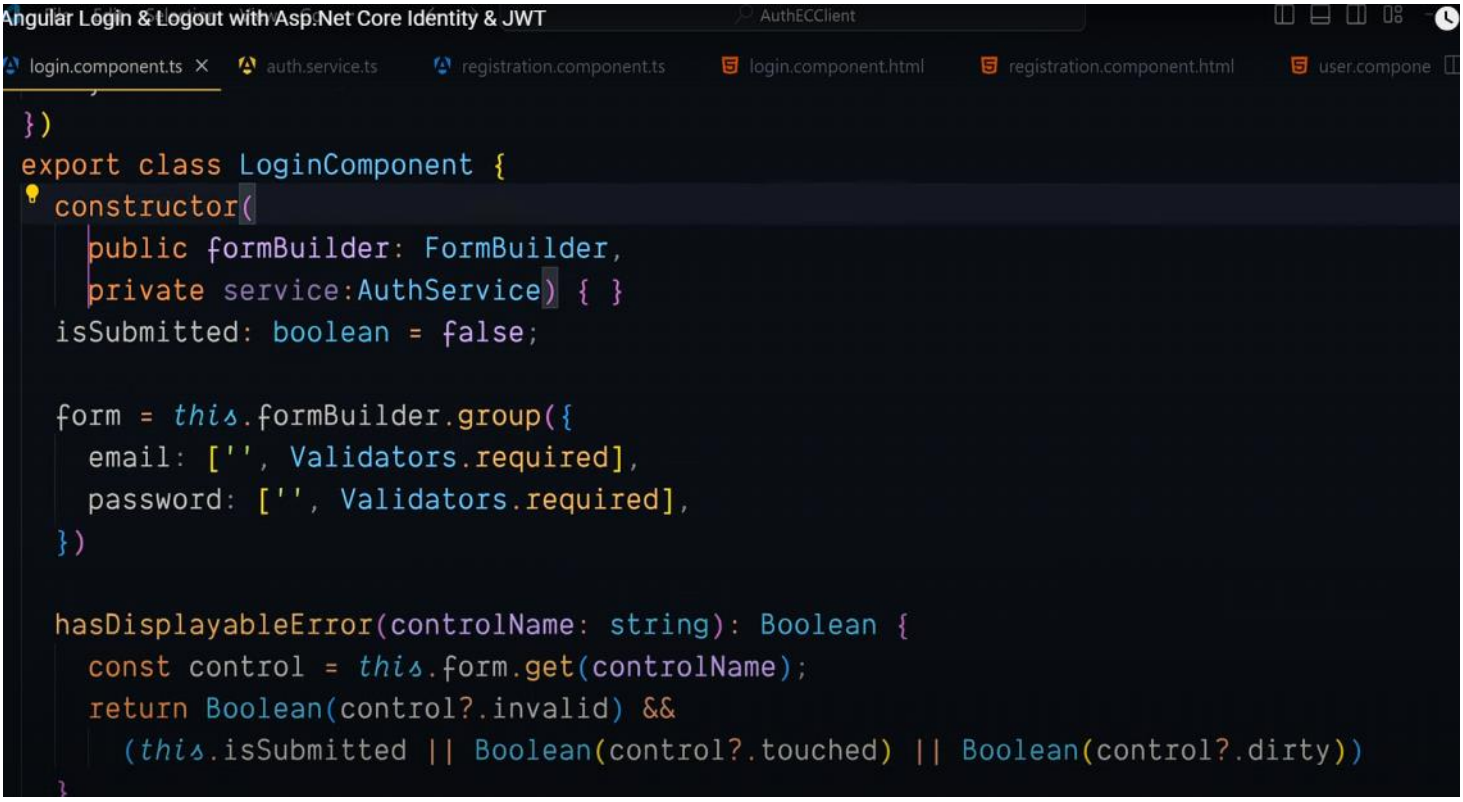
```
public class LoginModel
{
    public string Email { get; set; }
    public string Password { get; set; }
}
```

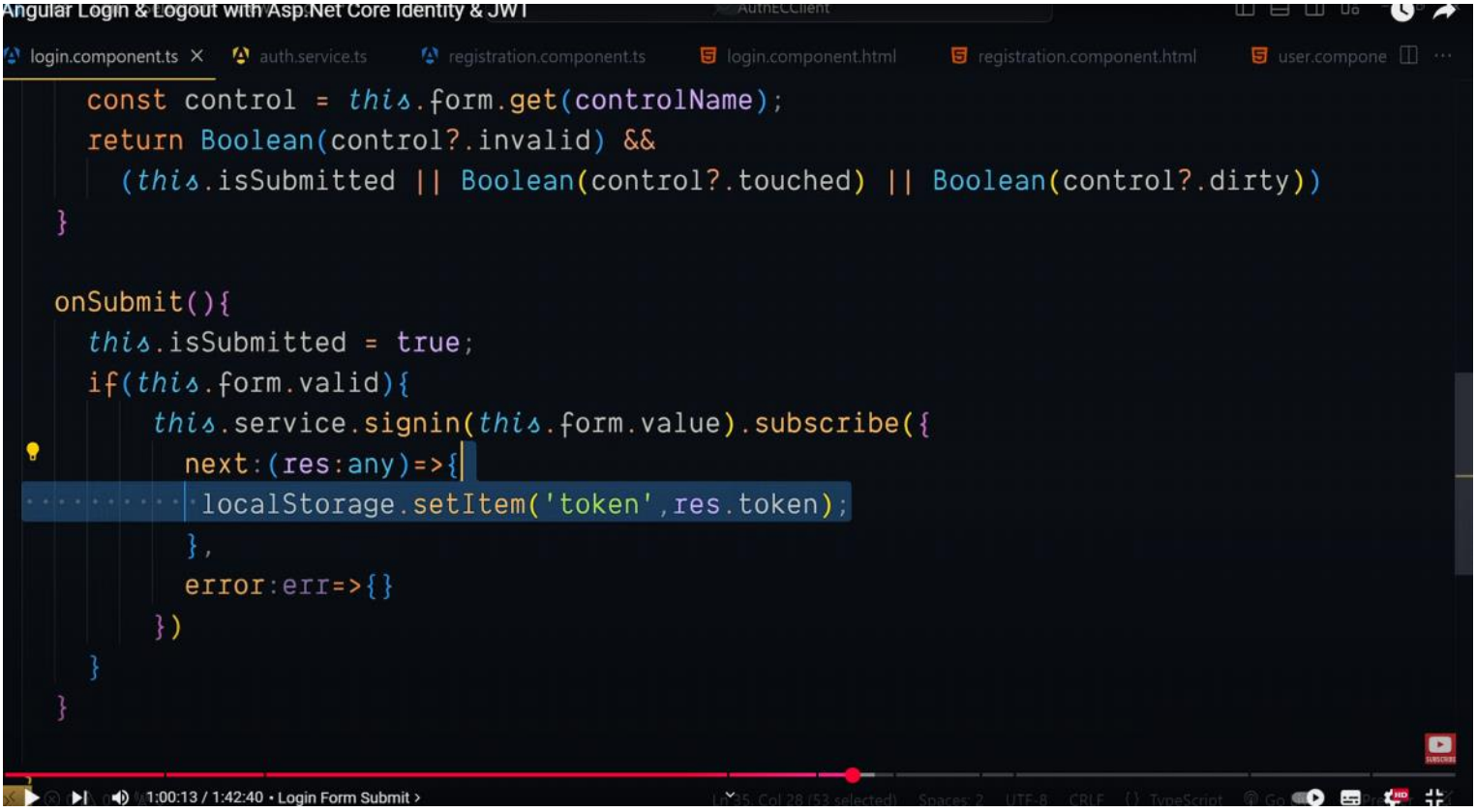


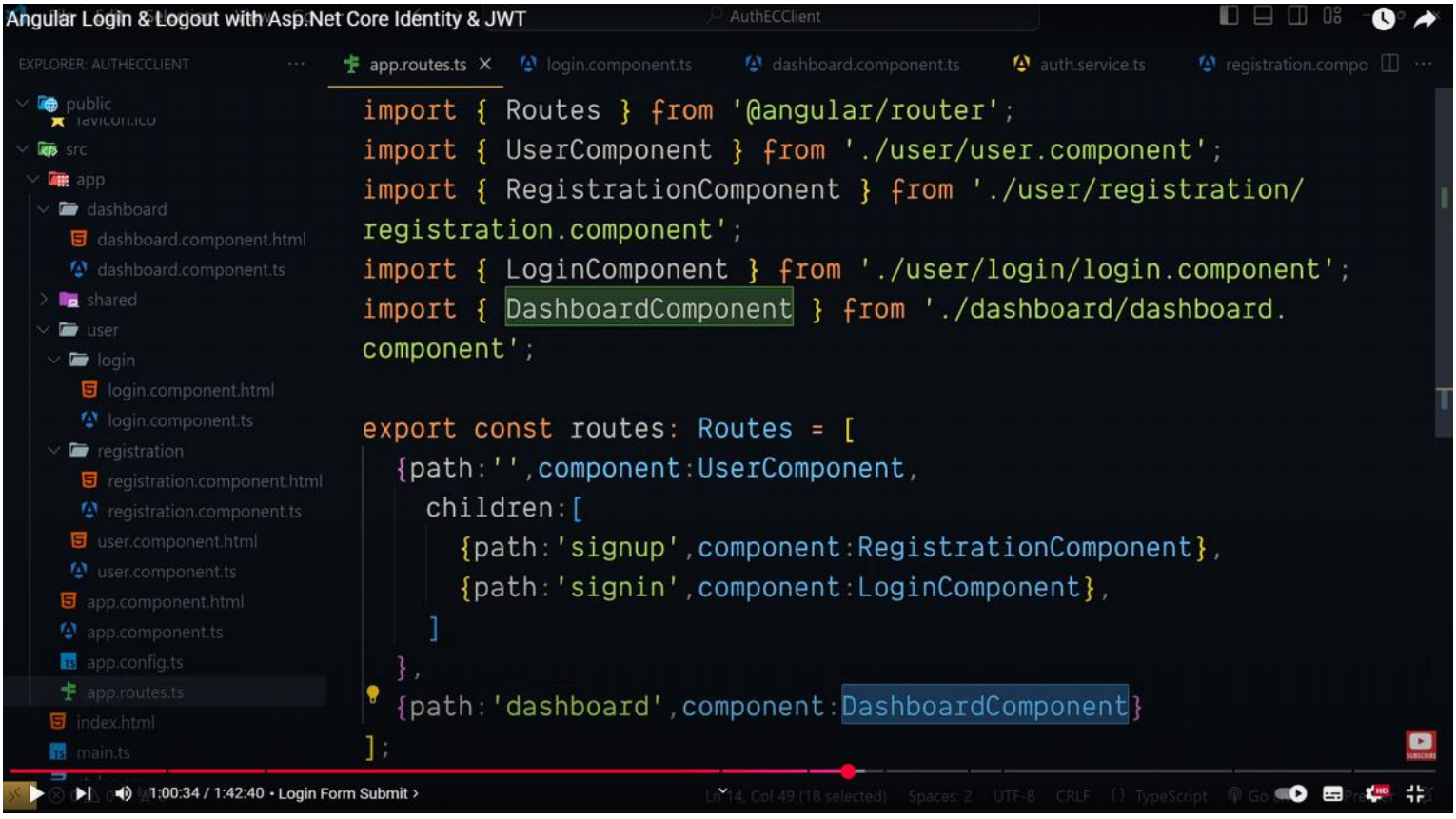

Program.cs
AppDbC...ext.cs
Weather...ller.cs
202407...ity2.cs
AppUser.cs
appset...gs.json
AuthECAPI
UserRegistrationModel
Email
UserManager<AppUser> userManager,
[FromBody] LoginModel loginModel) =>
{
var user =await userManager.FindByEmailAsync(loginModel.Email);
if (user != null && await userManager.CheckPasswordAsync(user, loginModel.Password))
{
var signInKey = new SymmetricSecurityKey(
Encoding.UTF8.GetBytes(builder.Configuration["AppSettings:JWTSecret"]!)
);
var tokenDescriptor = new SecurityTokenDescriptor
{
Subject = new ClaimsIdentity(new Claim[]
{
new Claim("UserID",user.Id.ToString())
}),
Expires = DateTime.UtcNow.AddMinutes(10),
SigningCredentials = new SigningCredentials(
signInKey,
SecurityAlgorithms.HmacSha256Signature
)
};
var tokenHandler = new JwtSecurityTokenHandler();
var securityToken = tokenHandler.CreateToken(tokenDescriptor);
var token = tokenHandler.WriteToken(securityToken);
return Results.Ok(new {token});
}
else
return Results.BadRequest(new { message = "Username or password is incorrect." });
});
app.Run();
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Analyzers
Frameworks
Packages
Properties
Controllers
Migrations
Models
AppDbContext.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

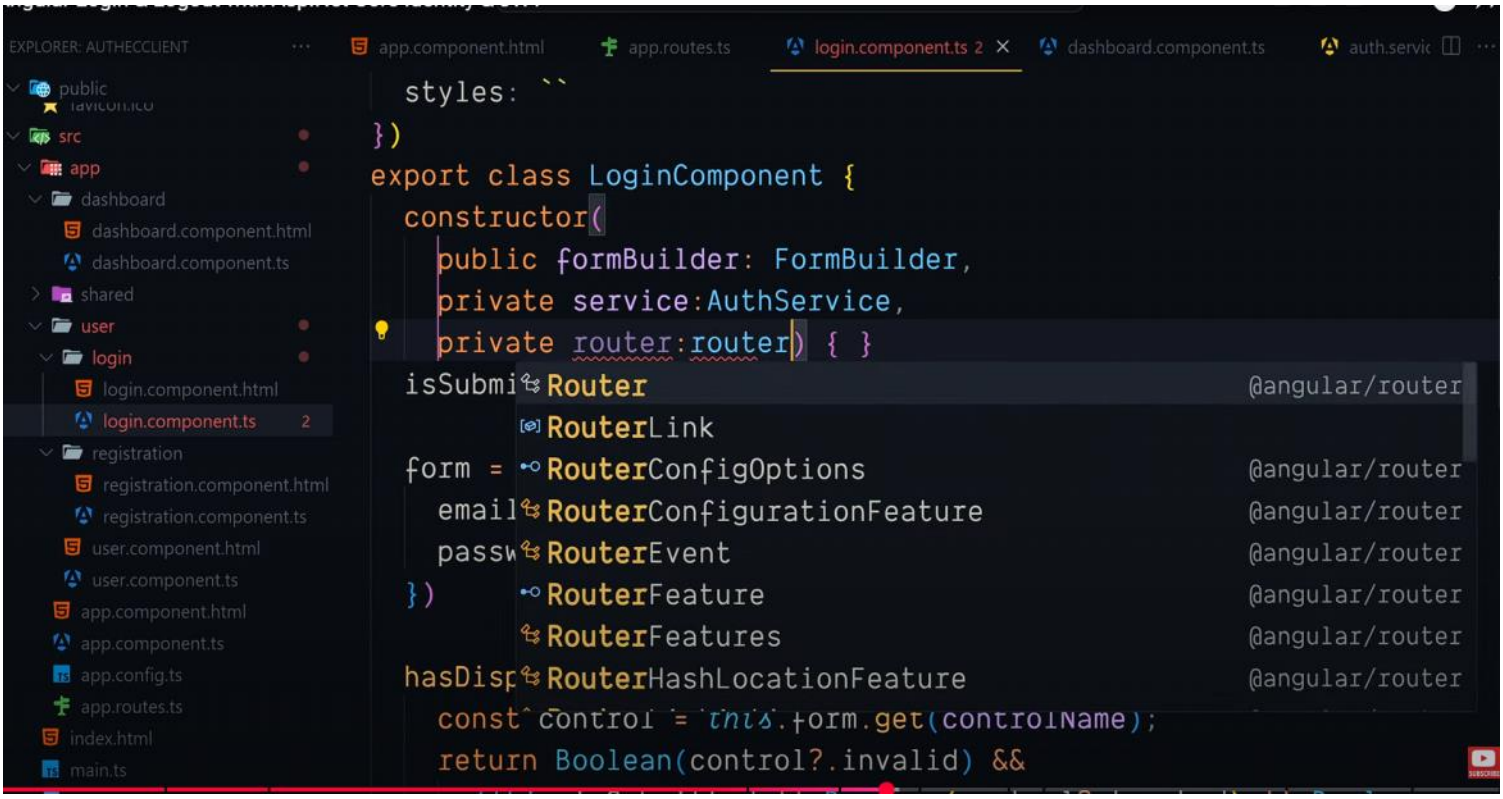

ngular Login & Logout with Asp.Net Core Identity & JWT
jwt.io/#debugger-io
Help us shape the future of jwt.io! Learn more about upcoming changes and share feedback.
JWT
Debugger
Libraries
Introduction
Ask
Crafted by Auth0 by Okta
Algorithm HS256
Encoded PASTE A TOKEN HERE
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJVc2VySUQiOiI3NmQ5Mzk1Ny03NzdmLTQ0MmUtYTRiYi01ODBlZWIzMGQyYTMiLCJuYmYiOjE3MjE4MjM2MDgsImV4cCI6MTcyMjY4NzYwOCwiaWF0IjoxNzIxODIzNjA4fQ.3HzfDz1D2vB3Q3EQpttmZo2XGpmp6zkNaOXTRahwWtY
Decoded EDIT THE PAYLOAD AND SECRET
HEADER: ALGORITHM & TOKEN TYPE
{
"alg": "HS256",
"typ": "JWT"
}
Type of token
PAYLOAD: DATA
{
"UserID": "76d93957-777f-442e-a4bb-580eeb30d2a3",
"nbf": 1721823608,
"exp": 1722687608,
"iat": 1721823608
}
VERIFY SIGNATURE
HMACSHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
your-256-bit-secret

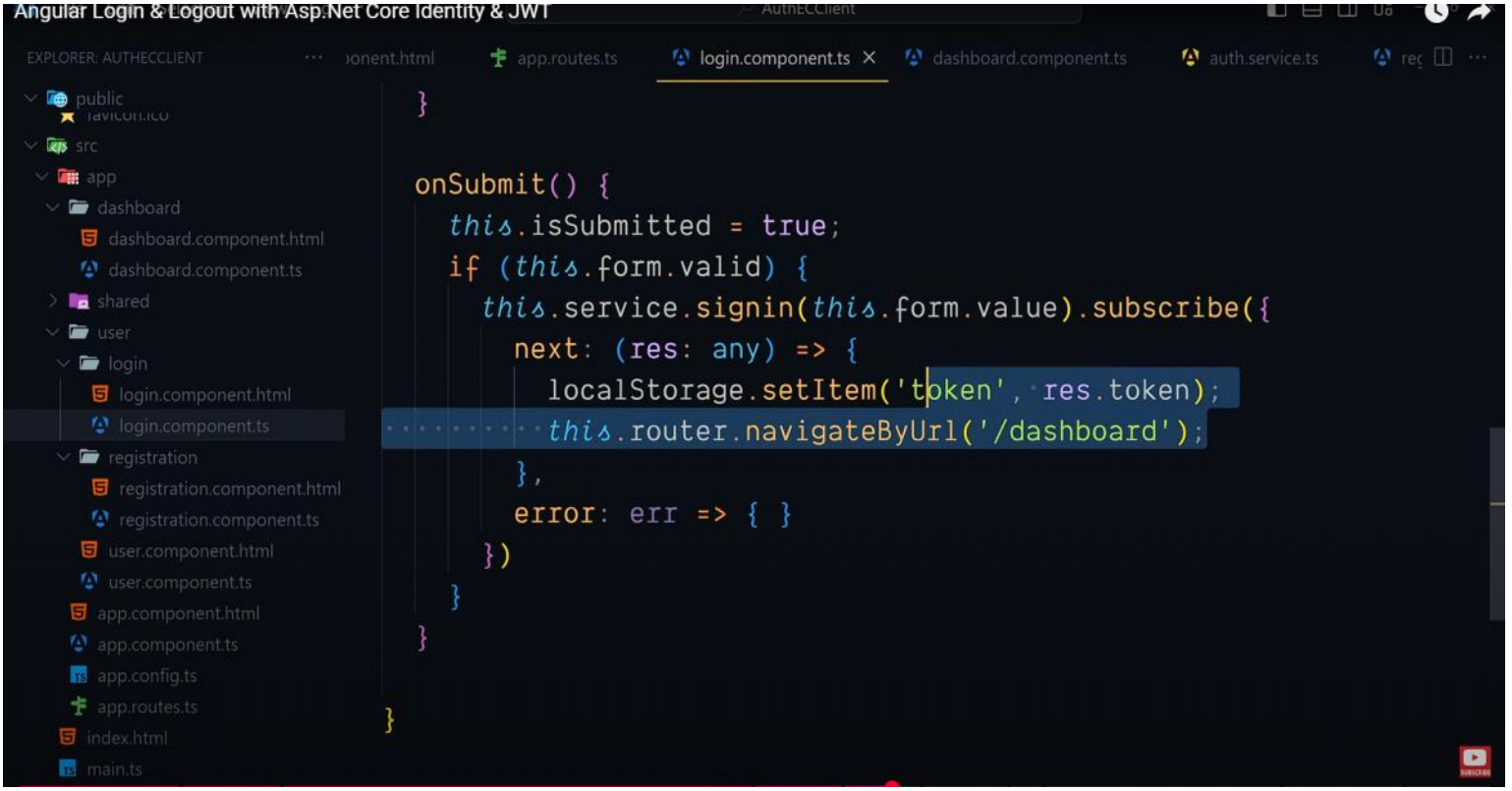
Service class

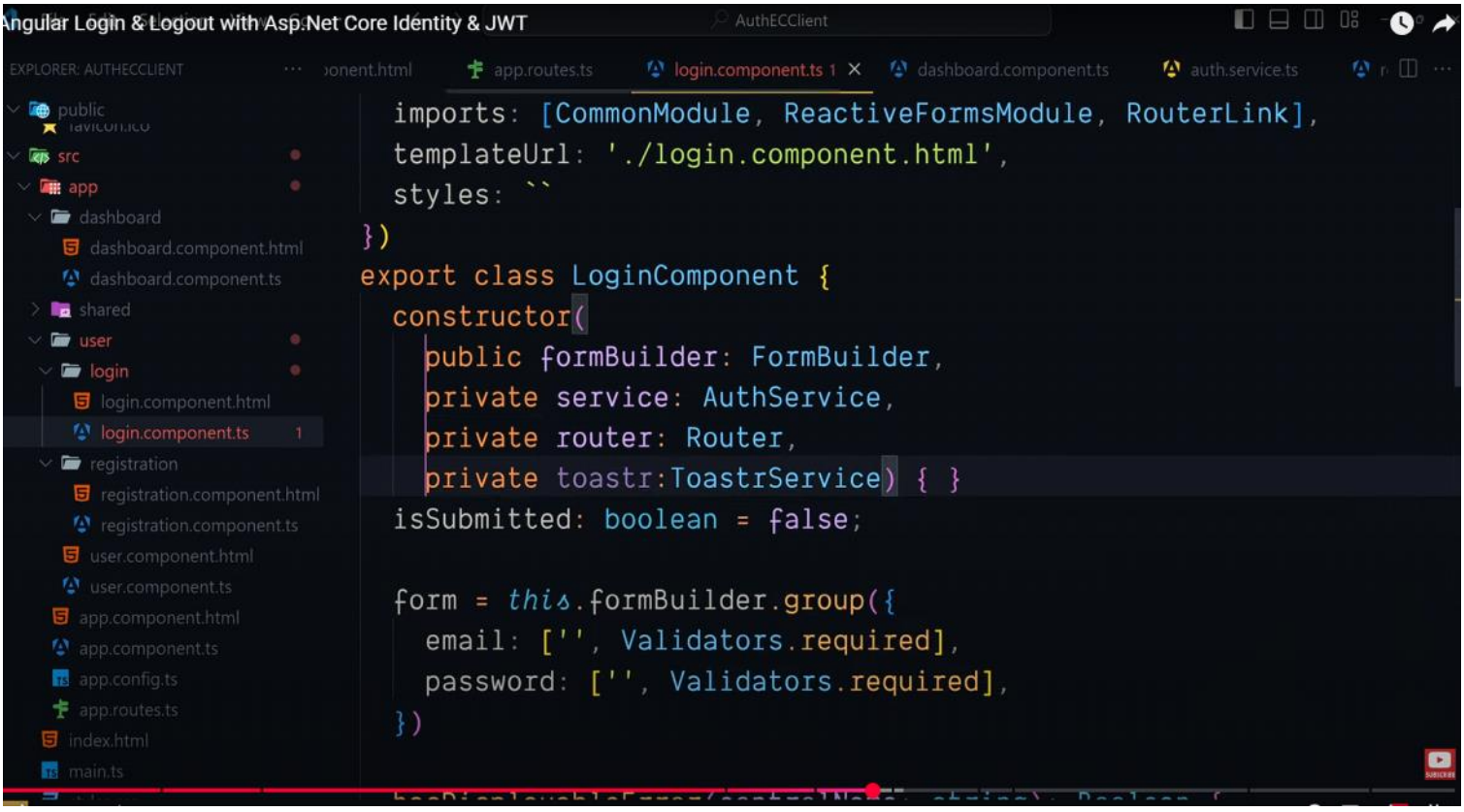


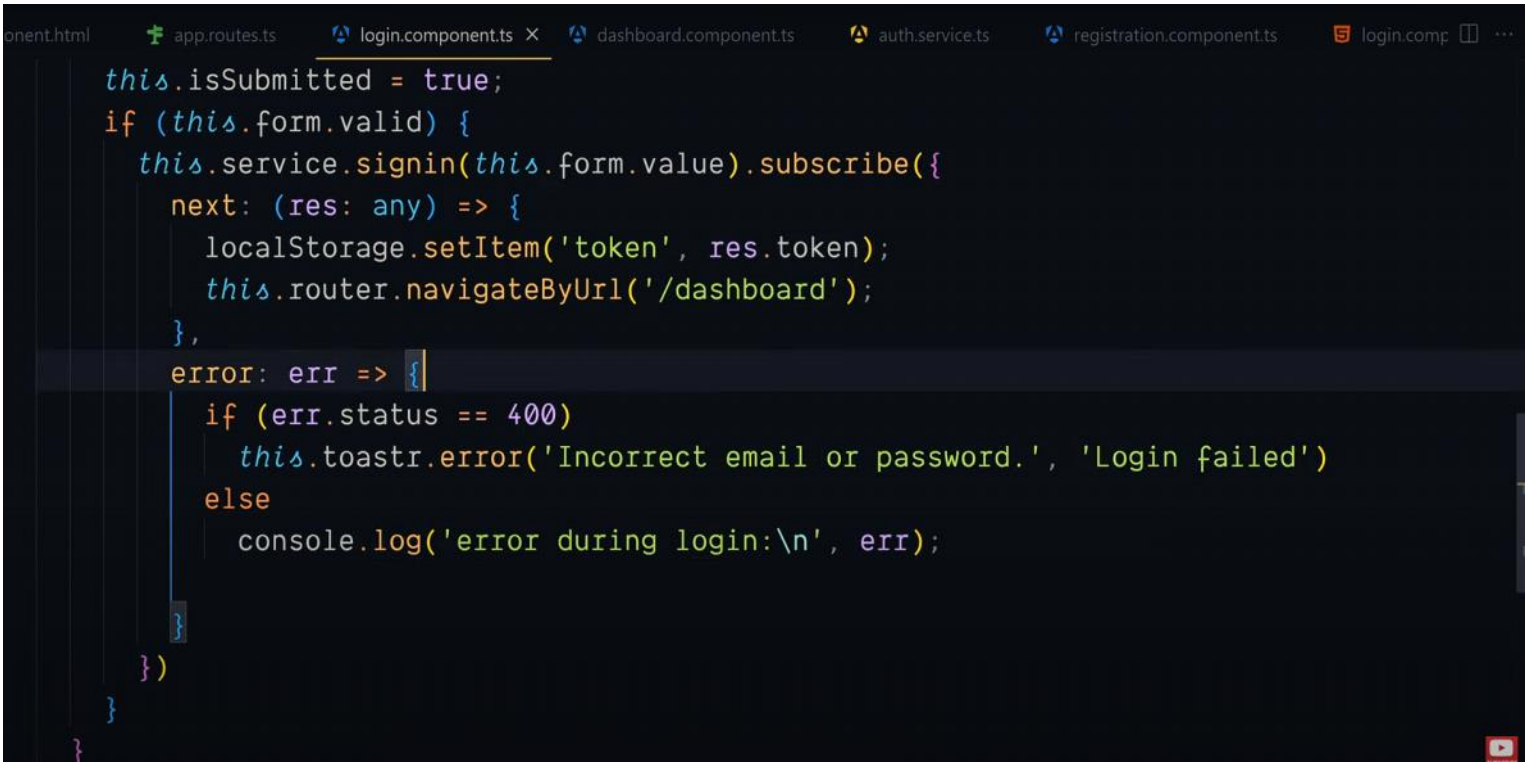

Angular Login & Logout with Asp.Net Core Identity & JWT
login.component.ts
auth.service.ts
registration.component.ts
login.component.html
registration.component.html
const control = this.form.get(controlName);
return Boolean(control?.invalid) &&
(this.isSubmitted || Boolean(control?.touched) || Boolean(control?.dirty))
}
onSubmit(){
this.isSubmitted = true;
if(this.form.valid){
this.service.signin(this.form.value).subscribe({
next:(res:any)=>{
localStorage.setItem('token',res.token);
},
error:err=>{}
})
}
}
1:00:13 / 1:42:40 • Login Form Submit ›

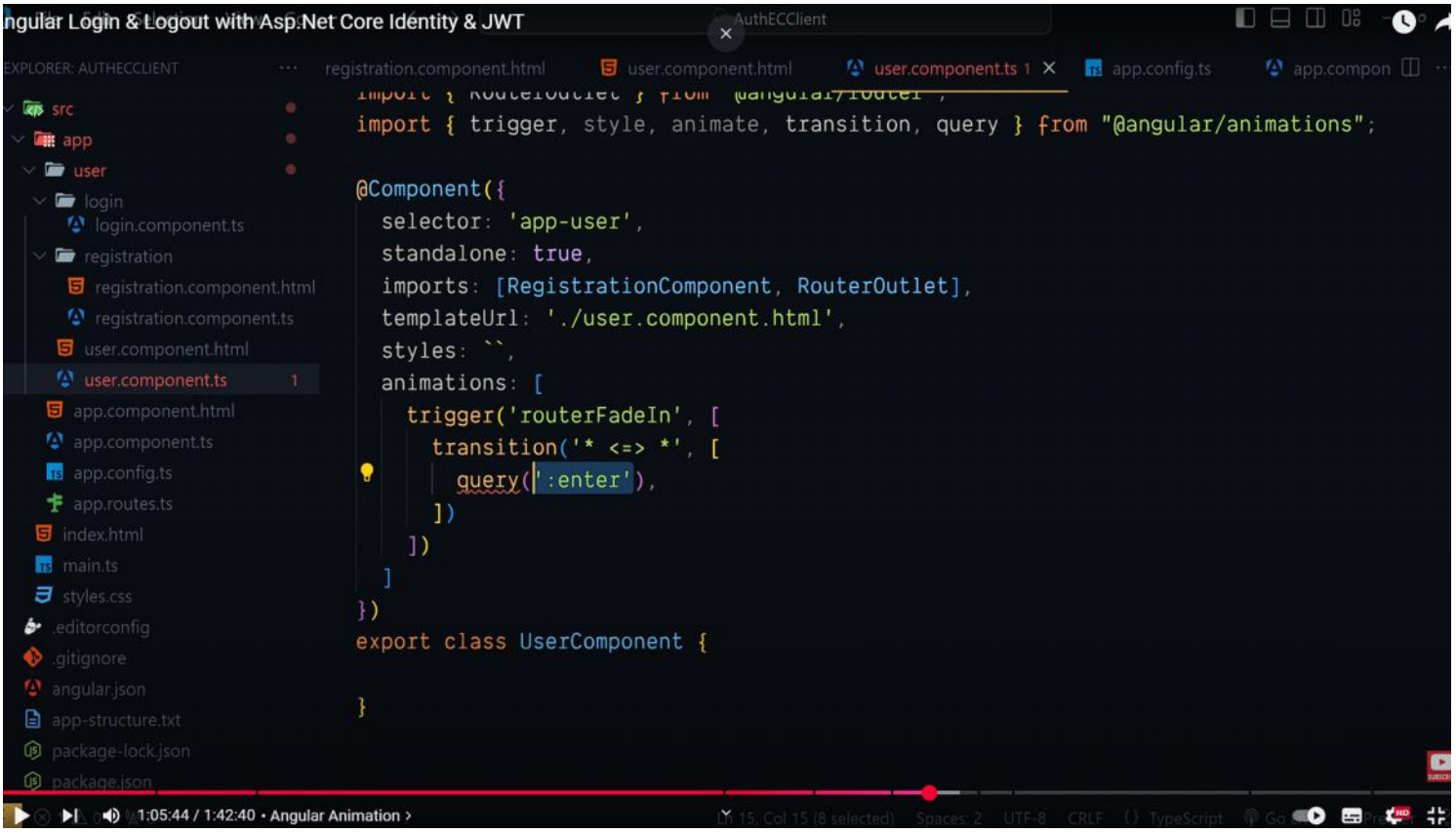

Angular Login & Logout with Asp.Net Core Identity & JWT
EXPLORER: AUTHECCLIENT
app.routes.ts
login.component.ts
dashboard.component.ts
auth.service.ts
public
src
app
dashboard
dashboard.component.html
dashboard.component.ts
shared
user
login
login.component.html
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
import { Routes } from '@angular/router';
import { UserComponent } from './user/user.component';
import { RegistrationComponent } from './user/registration/
registration.component';
import { LoginComponent } from './user/login/login.component';
import { DashboardComponent } from './dashboard/dashboard.
component';
export const routes: Routes = [
{path:'',component:UserComponent,
children:[
{path:'signup',component:RegistrationComponent},
{path:'signin',component:LoginComponent},
]
},
{path:'dashboard',component:DashboardComponent}
];
1:00:34 / 1:42:40 • Login Form Submit ›

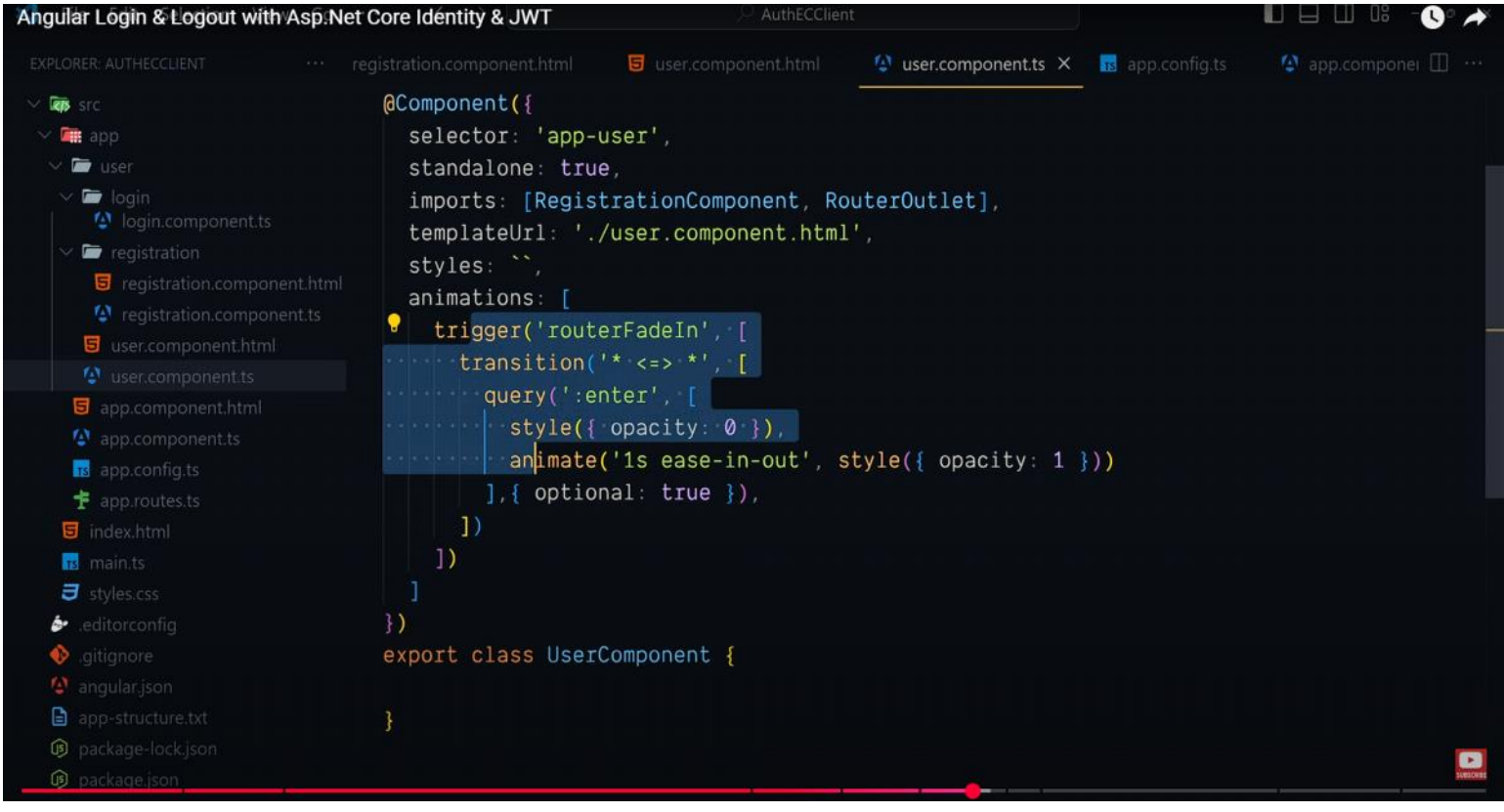

EXPLORER: AUTHECCLIENT
app.component.html
app.routes.ts
login.component.ts 2
dashboard.component.ts
auth.servic
public
src
app
dashboard
dashboard.component.html
dashboard.component.ts
shared
user
login
login.component.html
login.component.ts 2
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
styles: ``
})
export class LoginComponent {
constructor(
public formBuilder: FormBuilder,
private service:AuthService,
private router:router) { }
isSubmi
form =
email
passw
})
hasDisp
const control = this.form.get(controlName);
return Boolean(control?.invalid) &&
Router @angular/router
RouterLink
RouterConfigOptions @angular/router
RouterConfigurationFeature @angular/router
RouterEvent @angular/router
RouterFeature @angular/router
RouterFeatures @angular/router
RouterHashLocationFeature @angular/router

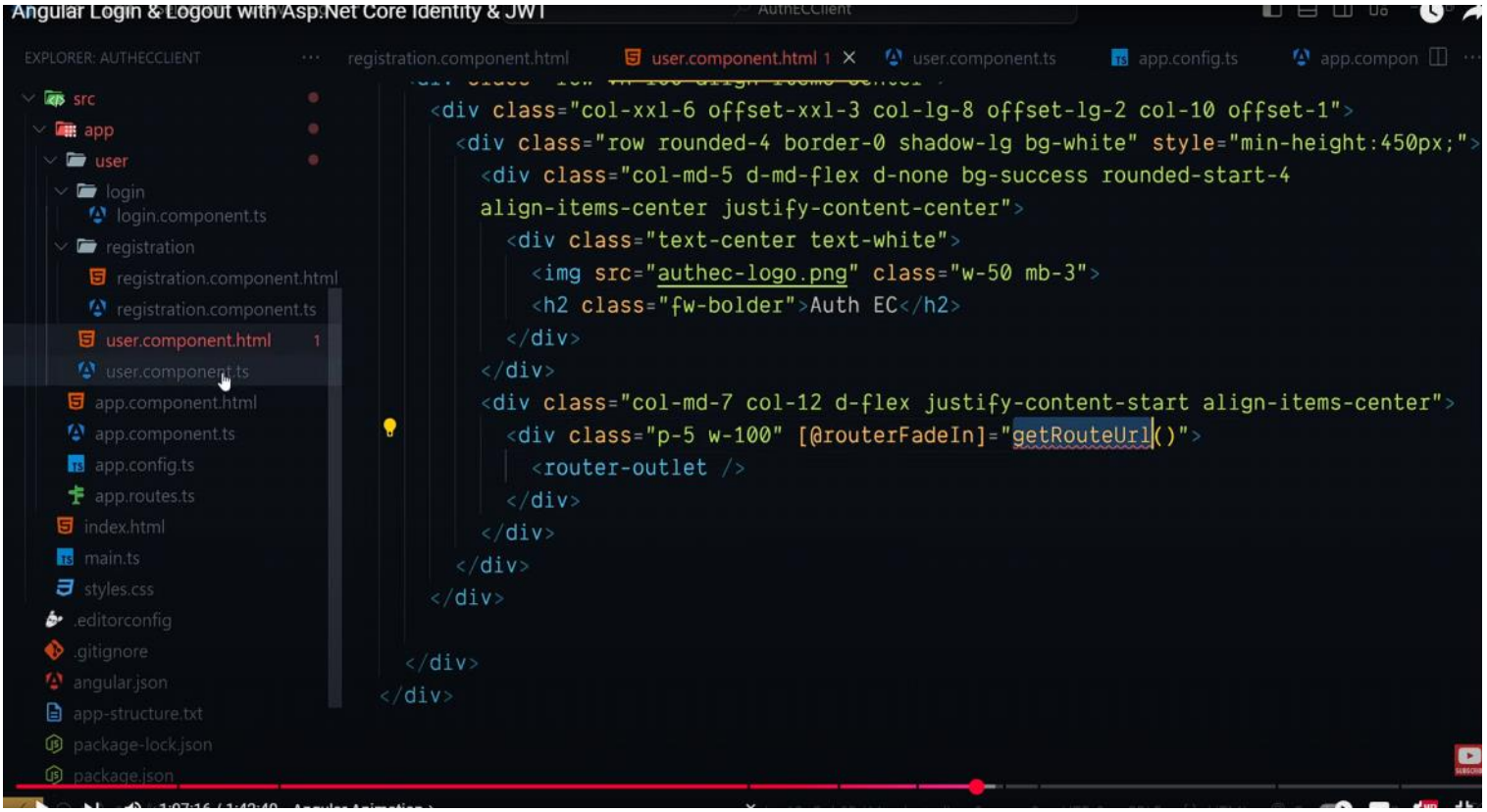

Angular Login & Logout with Asp.Net Core Identity & JWT
EXPLORER: AUTHECCLIENT
app.routes.ts
login.component.ts
dashboard.component.ts
auth.service.ts
public
src
app
dashboard
dashboard.component.html
dashboard.component.ts
shared
user
login
login.component.html
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
}
onSubmit() {
this.isSubmitted = true;
if (this.form.valid) {
this.service.signin(this.form.value).subscribe({
next: (res: any) => {
localStorage.setItem('token', res.token);
this.router.navigateByUrl('/dashboard');
},
error: err => { }
})
}
}
}


Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECClient
EXPLORER: AUTHECCLIENT
app.routes.ts
login.component.ts 1
dashboard.component.ts
auth.service.ts
public
src
app
dashboard
dashboard.component.html
dashboard.component.ts
shared
user
login
login.component.html
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
imports: [CommonModule, ReactiveFormsModule, RouterLink],
templateUrl: './login.component.html',
styles: ``
})
export class LoginComponent {
constructor(
public formBuilder: FormBuilder,
private service: AuthService,
private router: Router,
private toastr:ToastrService) { }
isSubmitted: boolean = false;
form = this.formBuilder.group({
email: ['', Validators.required],
password: ['', Validators.required],
})


app.routes.ts
login.component.ts
dashboard.component.ts
auth.service.ts
registration.component.ts
this.isSubmitted = true;
if (this.form.valid) {
this.service.signin(this.form.value).subscribe({
next: (res: any) => {
localStorage.setItem('token', res.token);
this.router.navigateByUrl('/dashboard');
},
error: err => {
if (err.status == 400)
this.toastr.error('Incorrect email or password.', 'Login failed')
else
console.log('error during login:\n', err);
}
})
}
}


Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECClient
EXPLORER: AUTHECCLIENT
registration.component.html
user.component.html
user.component.ts 1
app.config.ts
app.compon
src
app
user
login
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts 1
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
app-structure.txt
package-lock.json
package.json
import { trigger, style, animate, transition, query } from "@angular/animations";
@Component({
selector: 'app-user',
standalone: true,
imports: [RegistrationComponent, RouterOutlet],
templateUrl: './user.component.html',
styles: ``,
animations: [
trigger('routerFadeIn', [
transition('* <=> *', [
query(':enter'),
])
])
]
})
export class UserComponent {
}
1:05:44 / 1:42:40 • Angular Animation ›
Ln 15, Col 15 (8 selected)
Spaces: 2
UTF-8
CRLF
TypeScript

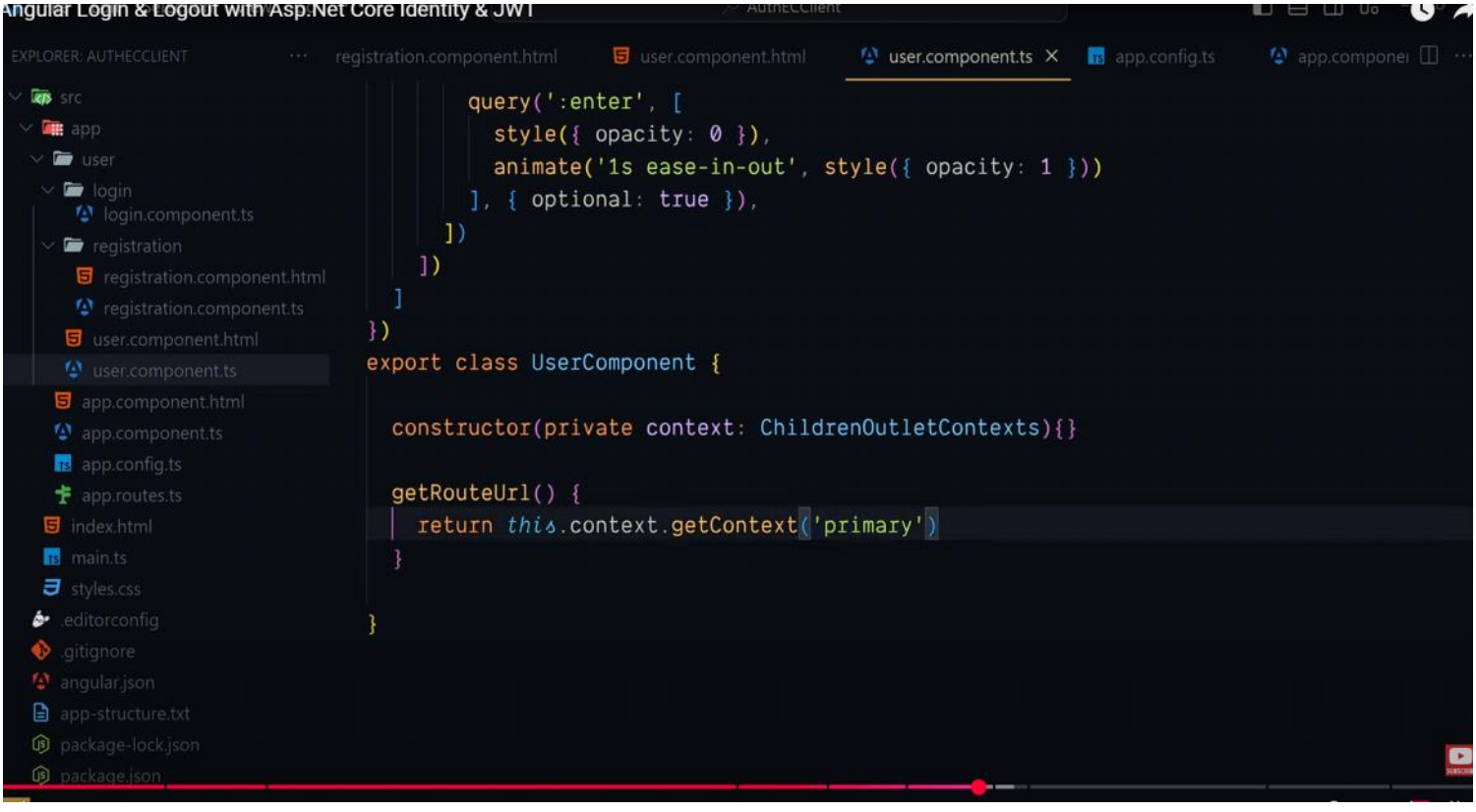

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECClient
EXPLORER: AUTHECCLIENT
registration.component.html
user.component.html
user.component.ts
app.config.ts
app.componei
src
app
user
login
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
app-structure.txt
package-lock.json
package.json
@Component({
selector: 'app-user',
standalone: true,
imports: [RegistrationComponent, RouterOutlet],
templateUrl: './user.component.html',
styles: ``,
animations: [
trigger('routerFadeIn', [
transition('* <=> *', [
query(':enter', [
style({ opacity: 0 }),
animate('1s ease-in-out', style({ opacity: 1 }))
],{ optional: true }),
])
])
]
})
export class UserComponent {
}

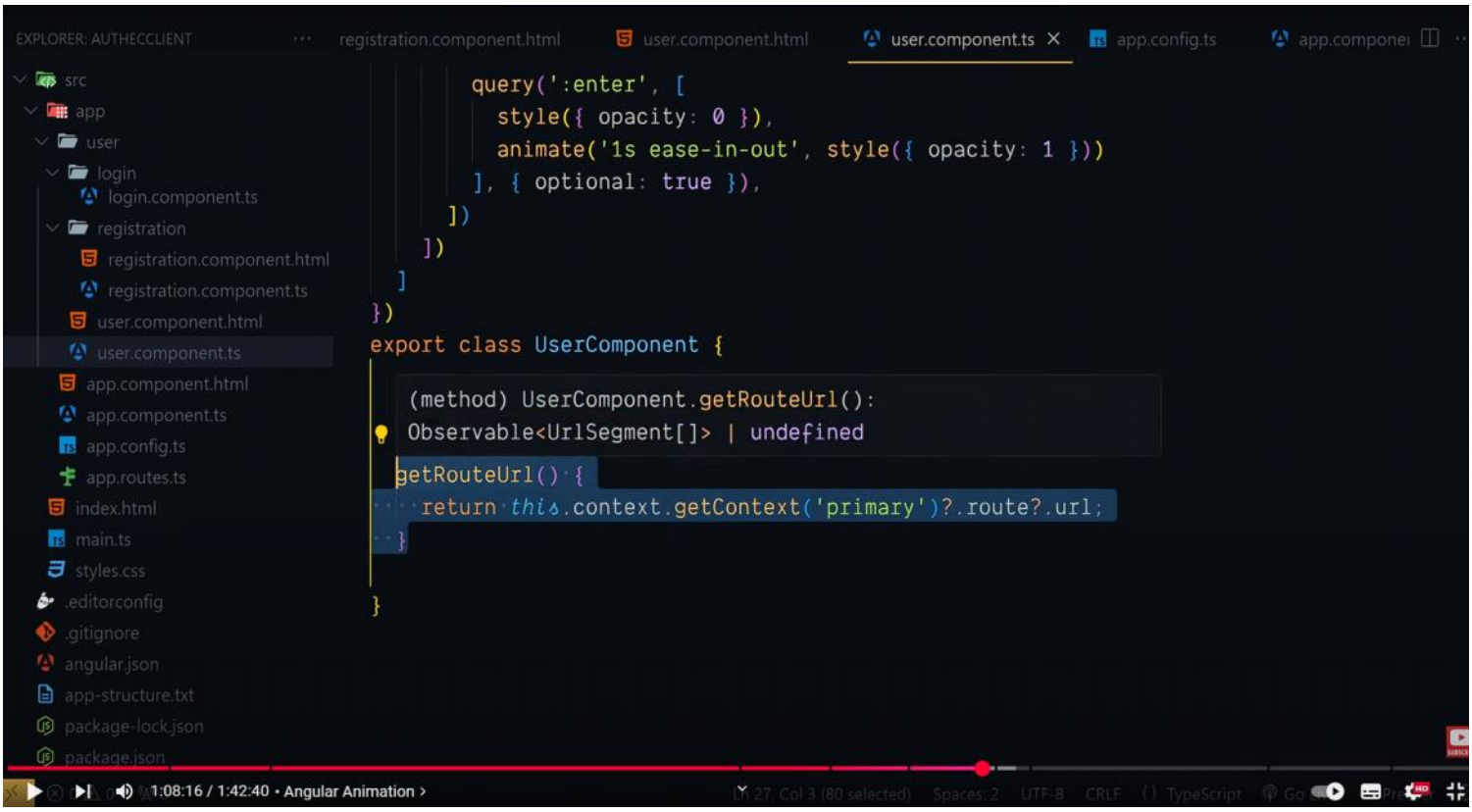

Angular Login & Logout with Asp.Net Core Identity & JWT
EXPLORER: AUTHECCLIENT
registration.component.html
user.component.html 1
user.component.ts
app.config.ts
src
app
user
login
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
app-structure.txt
package-lock.json
package.json

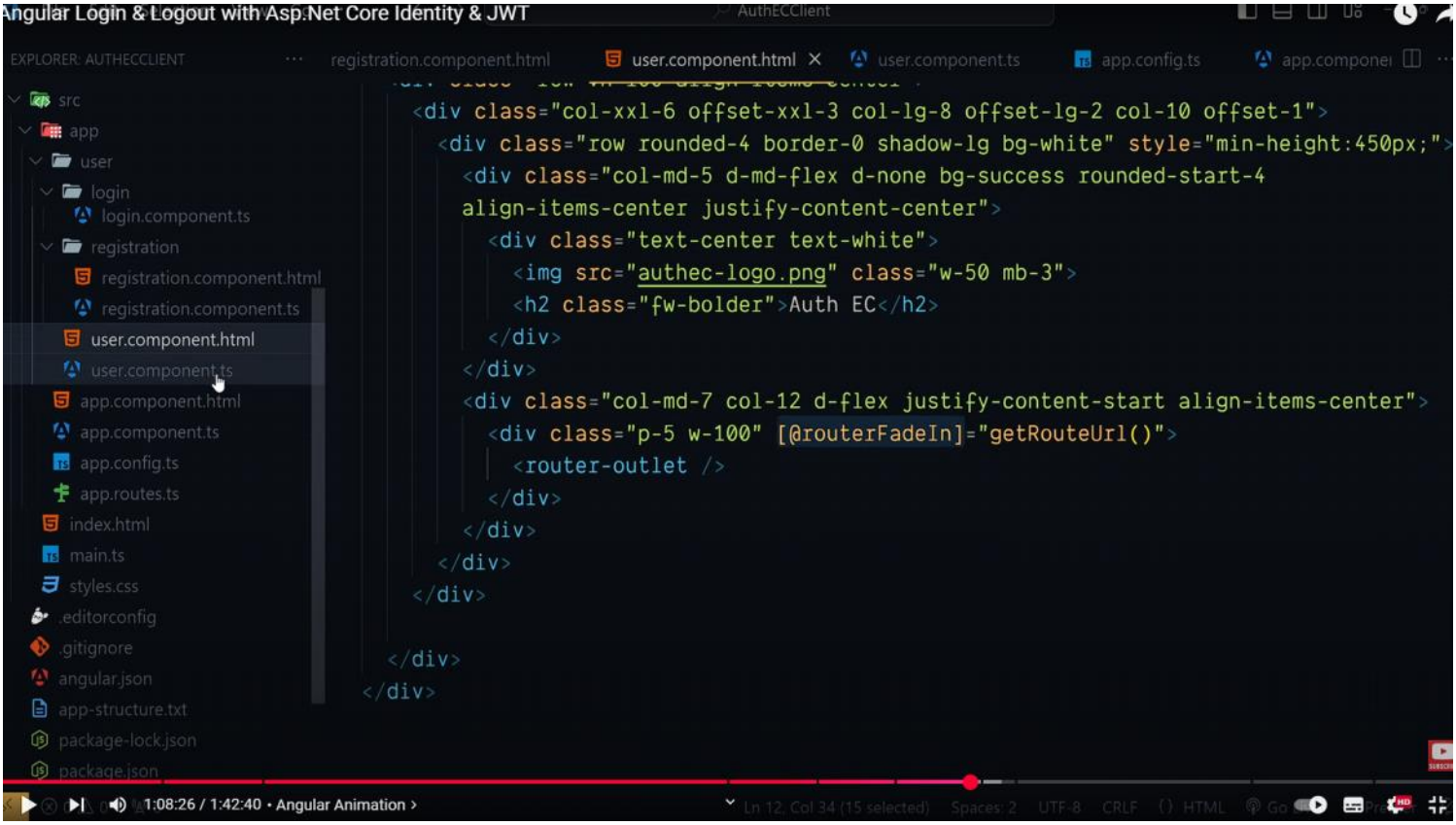
<div class="col-md-5 d-md-flex d-none bg-success rounded-start-4
align-items-center justify-content-center">

<img src="authec-logo.png" class="w-50 mb-3">
<h2 class="fw-bolder">Auth EC</h2>

<div class="p-5 w-100" [@routerFadeIn]="getRouteUrl()">
<router-outlet />


Angular Login & Logout with Asp.Net Core Identity & JWT
EXPLORER: AUTHECCLIENT
registration.component.html
user.component.html
user.component.ts
app.config.ts
src
app
user
login
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
app-structure.txt
package-lock.json
package.json
query(':enter', [
style({ opacity: 0 }),
animate('1s ease-in-out', style({ opacity: 1 }))
], { optional: true }),
])
])
]
})
export class UserComponent {
constructor(private context: ChildrenOutletContexts){}
getRouteUrl() {
return this.context.getContext('primary')
}
}


EXPLORER: AUTHECCLIENT
registration.component.html
user.component.html
user.component.ts
app.config.ts
src
app
user
login
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
app-structure.txt
package-lock.json
package.json
query(':enter', [
style({ opacity: 0 }),
animate('1s ease-in-out', style({ opacity: 1 }))
], { optional: true }),
])
])
]
})
export class UserComponent {
(method) UserComponent.getRouteUrl():
Observable<UrlSegment[]> | undefined
getRouteUrl() {
return this.context.getContext('primary')?.route?.url;
}
}
1:08:16 / 1:42:40 • Angular Animation ›

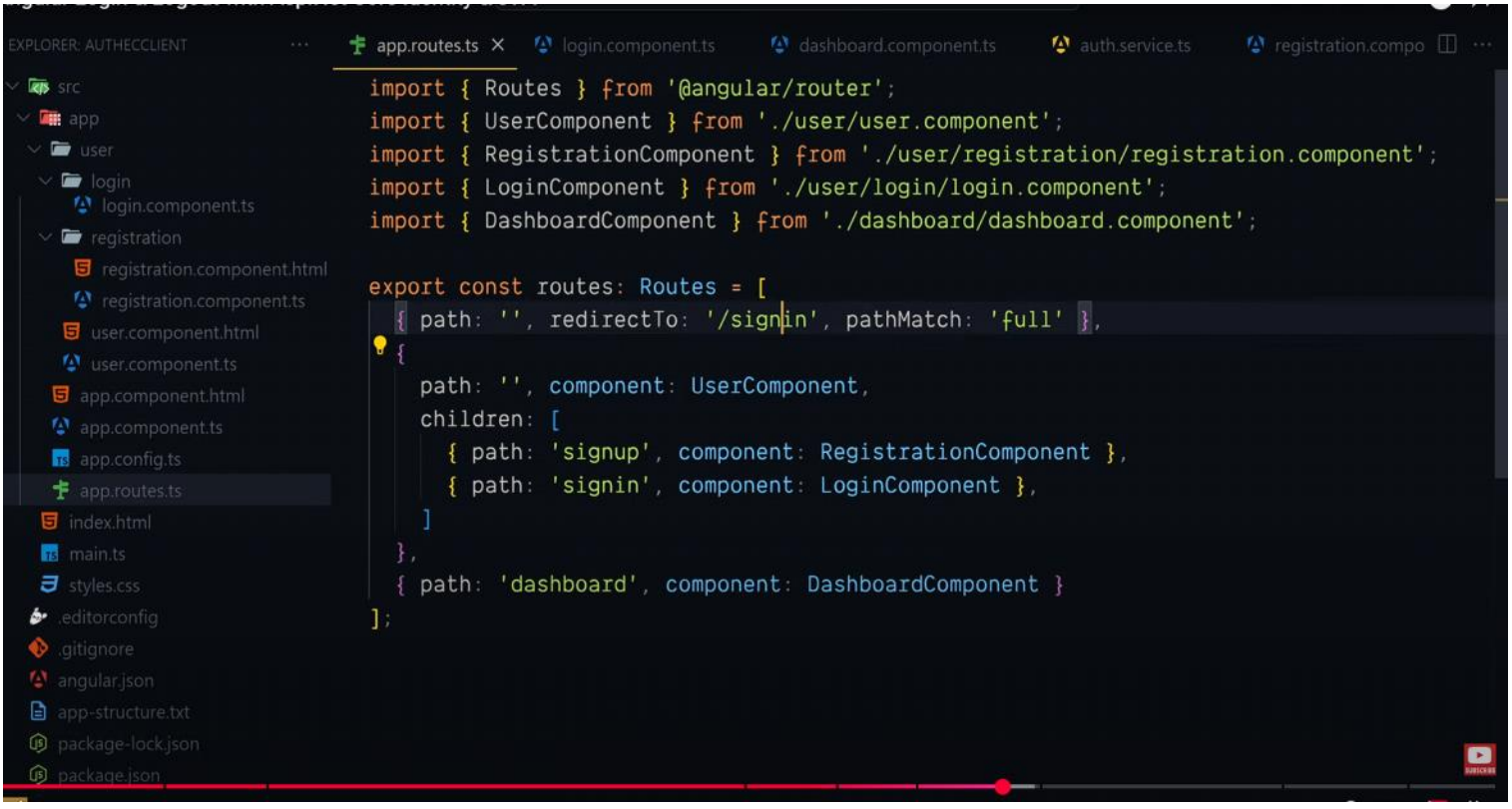

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECClient
EXPLORER: AUTHECCLIENT
registration.component.html
user.component.html
user.component.ts
app.config.ts
src
app
user
login
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
app-structure.txt
package-lock.json
package.json

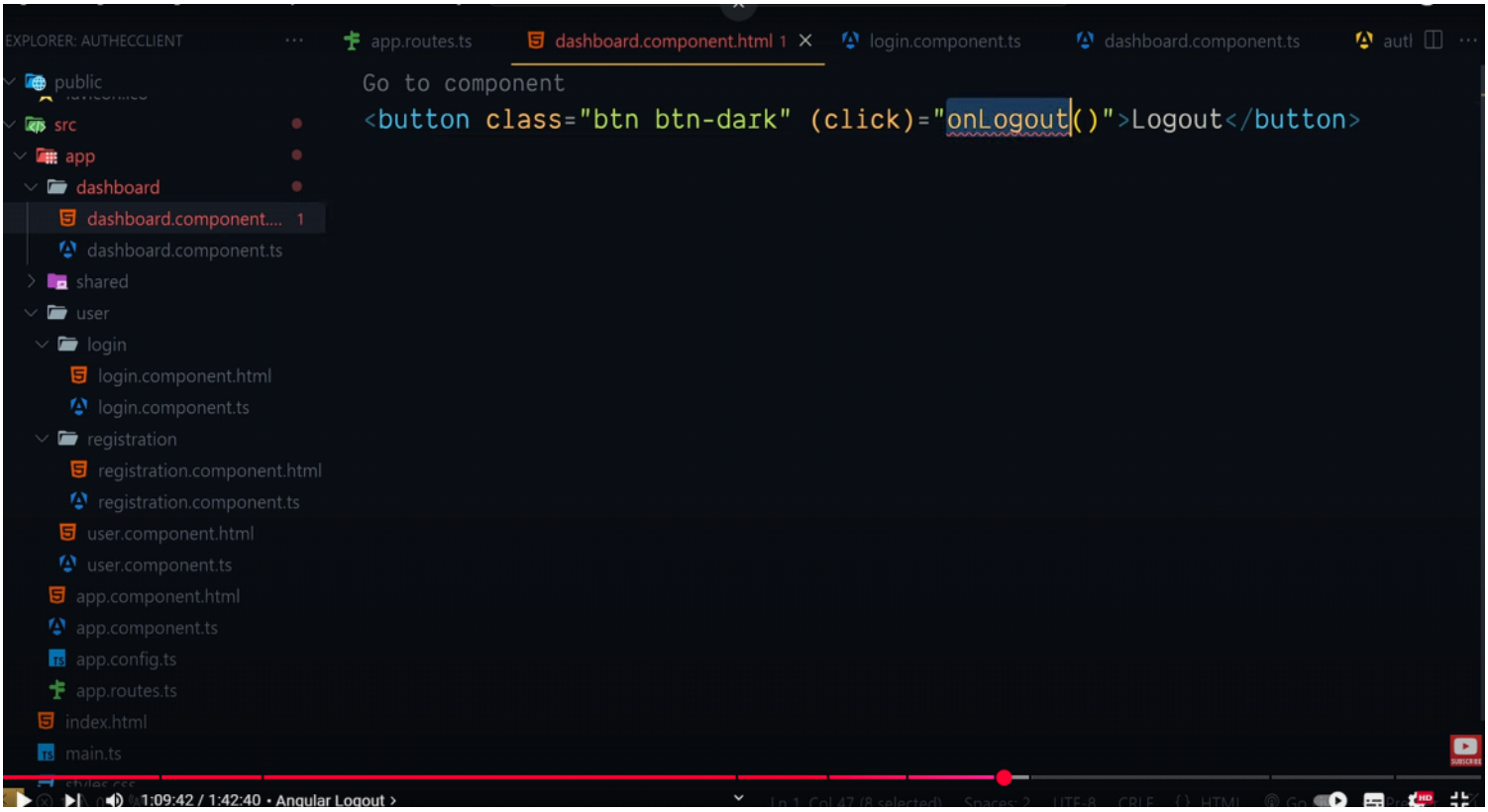
<div class="col-md-5 d-md-flex d-none bg-success rounded-start-4
align-items-center justify-content-center">

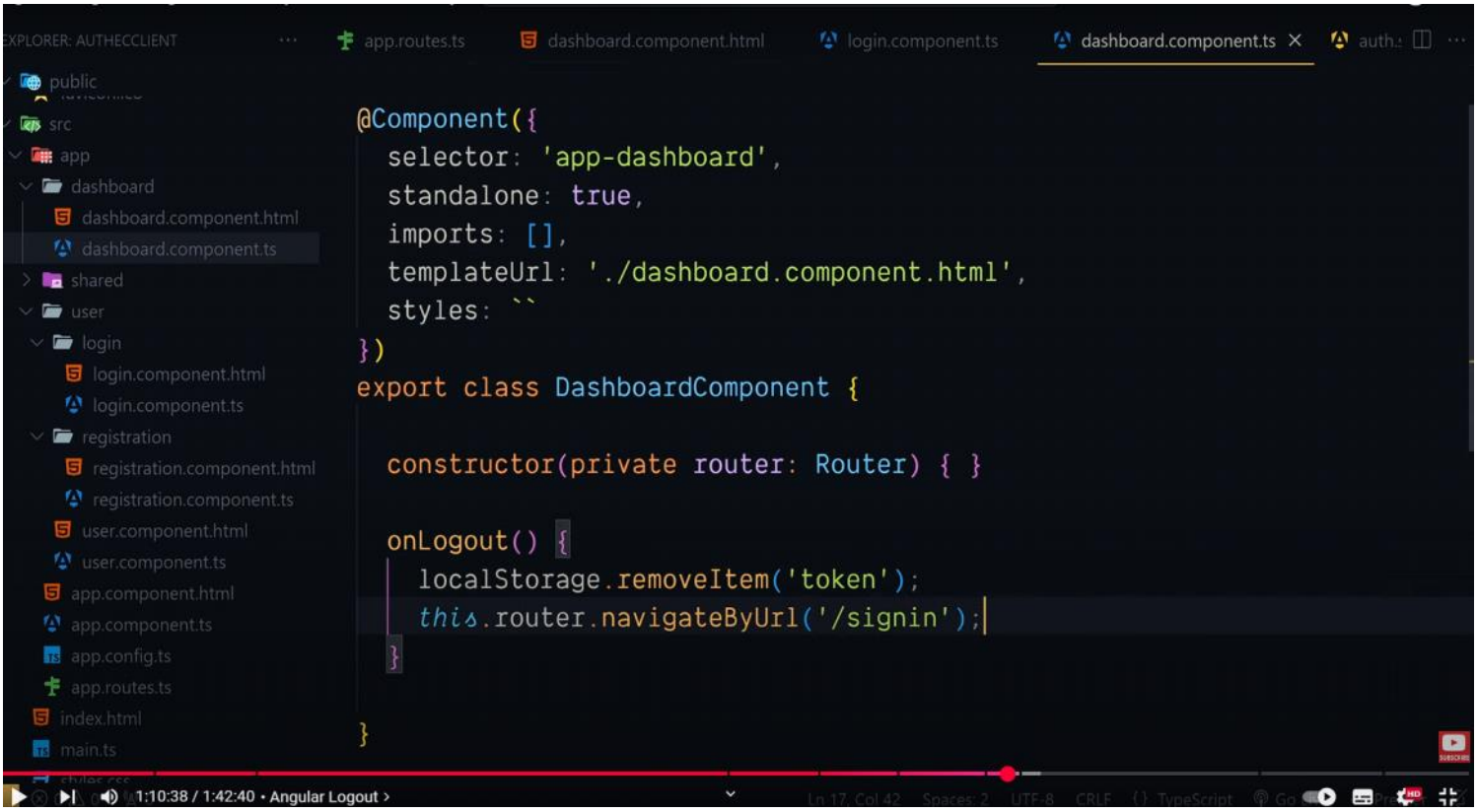
<img src="authec-logo.png" class="w-50 mb-3">
<h2 class="fw-bolder">Auth EC</h2>

<div class="p-5 w-100" [@routerFadeIn]="getRouteUrl()">
<router-outlet />

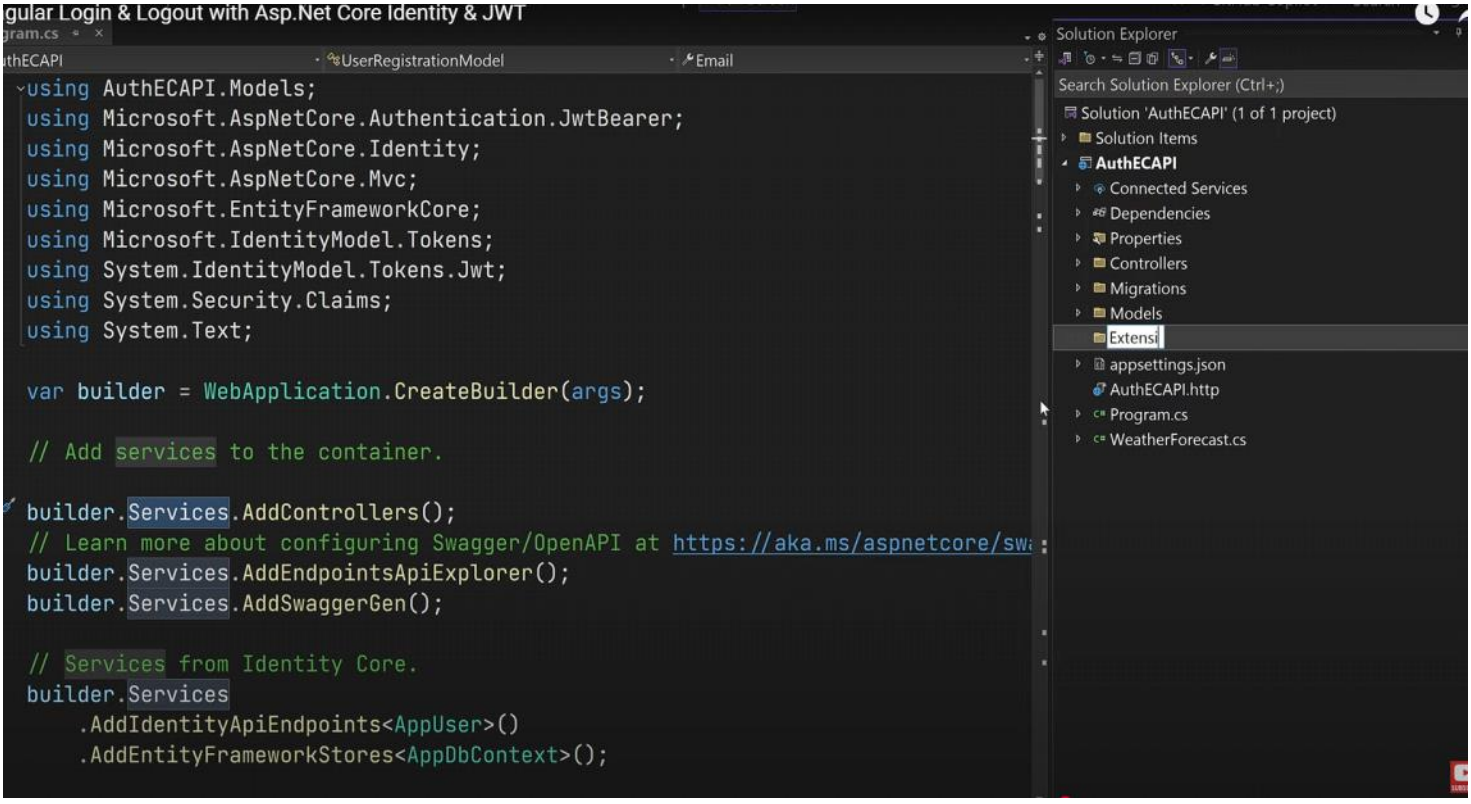
1:08:26 / 1:42:40 • Angular Animation ›

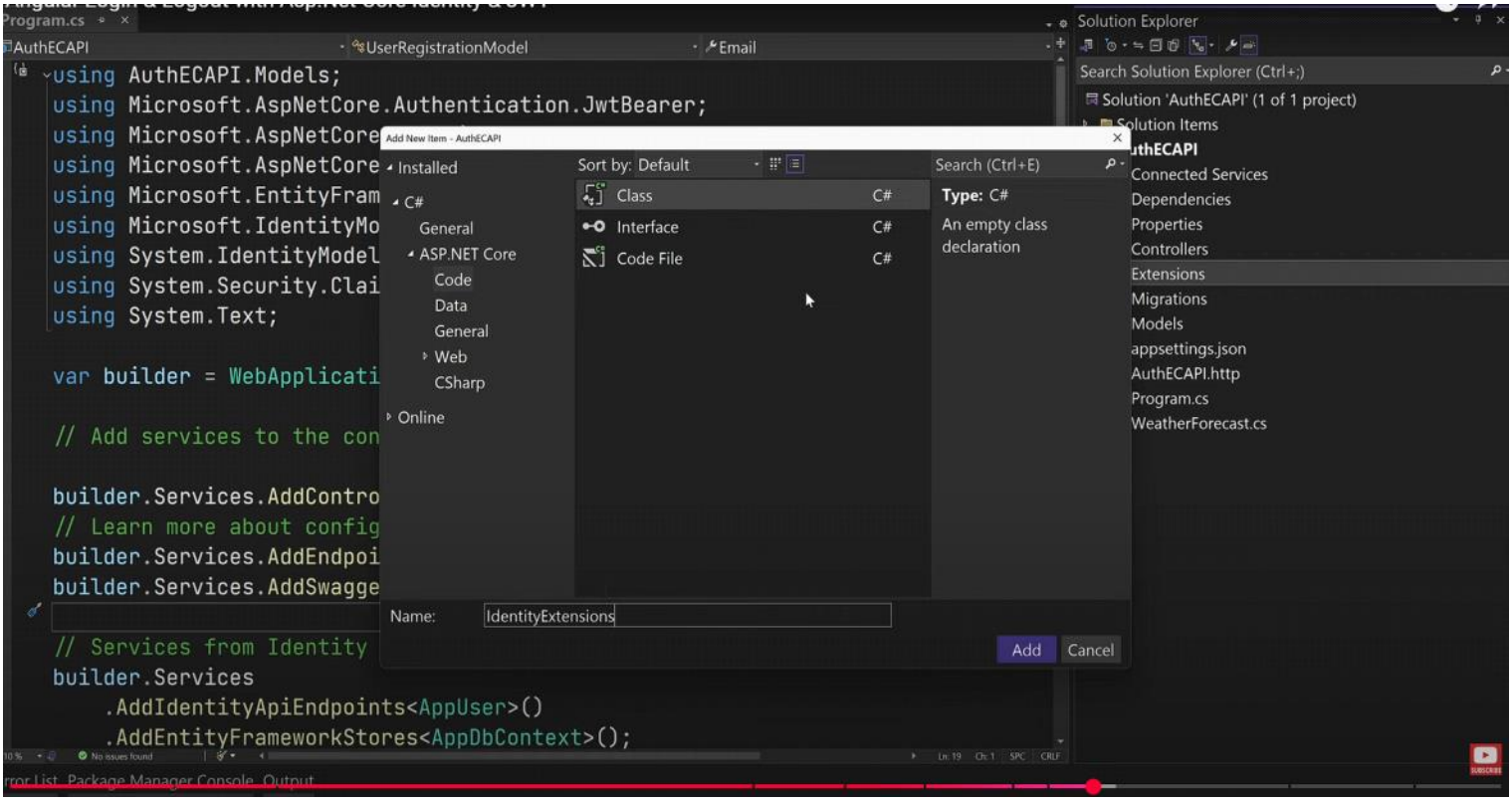

EXPLORER: AUTHECCLIENT
app.routes.ts
login.component.ts
dashboard.component.ts
auth.service.ts
registration.compo
src
app
user
login
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
app-structure.txt
package-lock.json
package.json
import { Routes } from '@angular/router';
import { UserComponent } from './user/user.component';
import { RegistrationComponent } from './user/registration/registration.component';
import { LoginComponent } from './user/login/login.component';
import { DashboardComponent } from './dashboard/dashboard.component';
export const routes: Routes = [
{ path: '', redirectTo: '/signin', pathMatch: 'full' },
{
path: '', component: UserComponent,
children: [
{ path: 'signup', component: RegistrationComponent },
{ path: 'signin', component: LoginComponent },
]
},
{ path: 'dashboard', component: DashboardComponent }
];

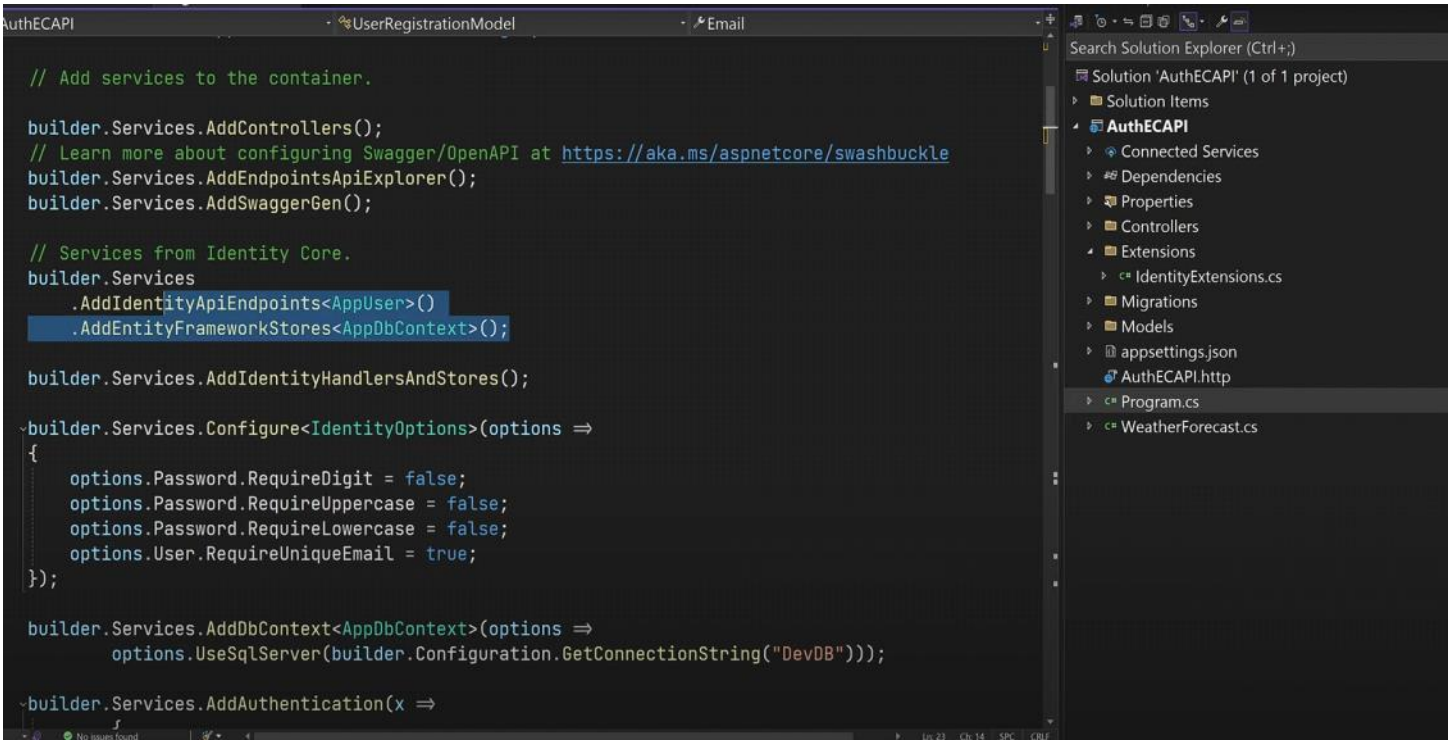

EXPLORER: AUTHECCLIENT
app.routes.ts
dashboard.component.html 1
login.component.ts
dashboard.component.ts
public
src
app
dashboard
dashboard.component.... 1
dashboard.component.ts
shared
user
login
login.component.html
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
Go to component
<button class="btn btn-dark" (click)="onLogout()">Logout</button>
1:09:42 / 1:42:40 • Angular Logout ›


EXPLORER: AUTHECCLIENT
app.routes.ts
dashboard.component.html
login.component.ts
dashboard.component.ts
public
src
app
dashboard
dashboard.component.html
dashboard.component.ts
shared
user
login
login.component.html
login.component.ts
registration
registration.component.html
registration.component.ts
user.component.html
user.component.ts
app.component.html
app.component.ts
app.config.ts
app.routes.ts
index.html
main.ts
@Component({
selector: 'app-dashboard',
standalone: true,
imports: [],
templateUrl: './dashboard.component.html',
styles: ``
})
export class DashboardComponent {
constructor(private router: Router) { }
onLogout() {
localStorage.removeItem('token');
this.router.navigateByUrl('/signin');
}
}
1:10:38 / 1:42:40 • Angular Logout ›

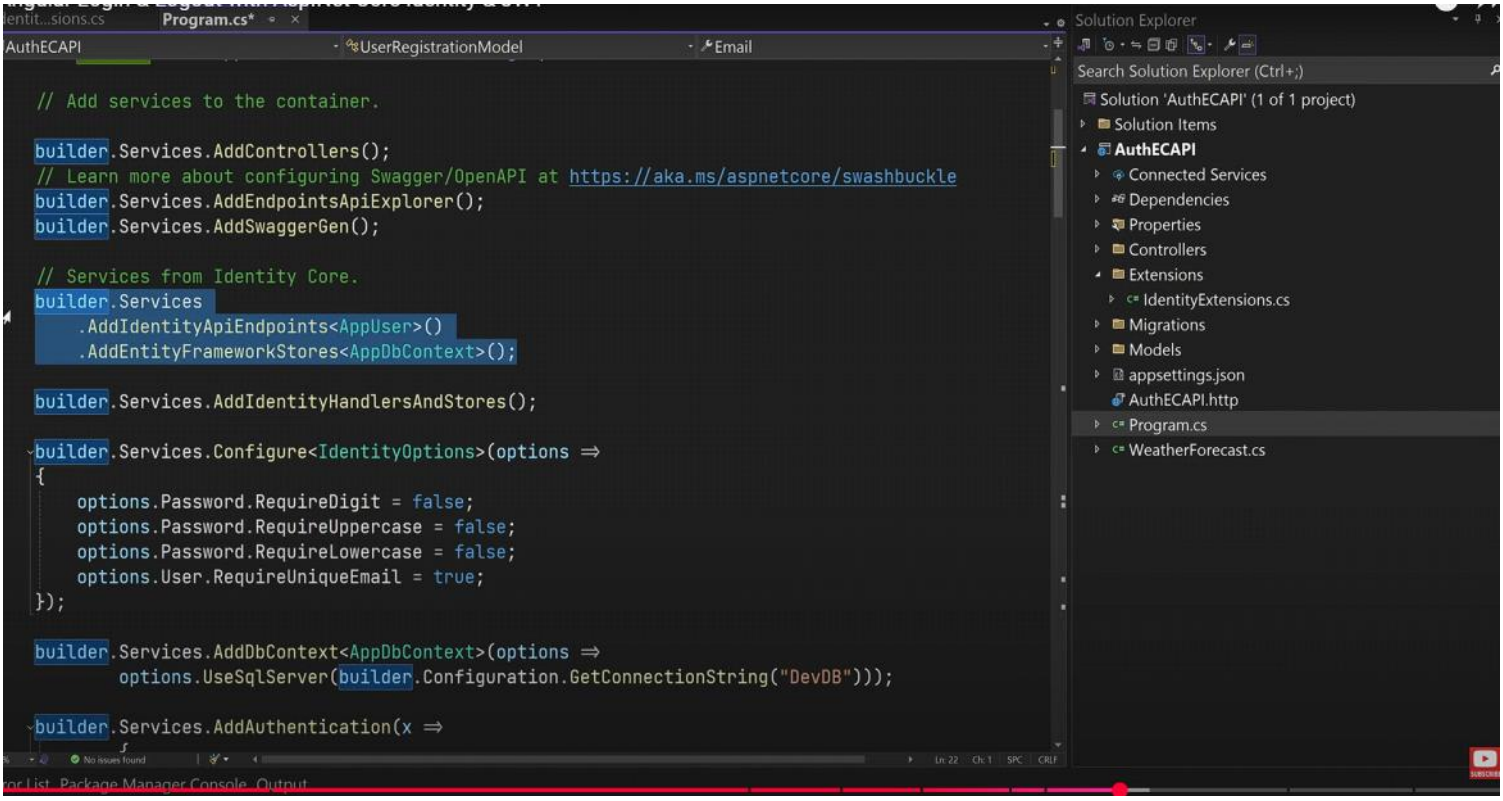

C# Extension Methods
Framework ABC
namespace ThirdParty
{
public interface/class Original
{
// Owned Methods & Properties.
}
}
A Seprate File
namespace Extensions
{
public static class MeaningfulName
{
public static void ExtraMethod1(this Original org)
{...}
public static void ExtraMethod2(this Original org)
{...}
}
}
Program.cs
...
//Say org1 is of the type Original
org1.ExtraMethod1();
org1.ExtraMethod2();

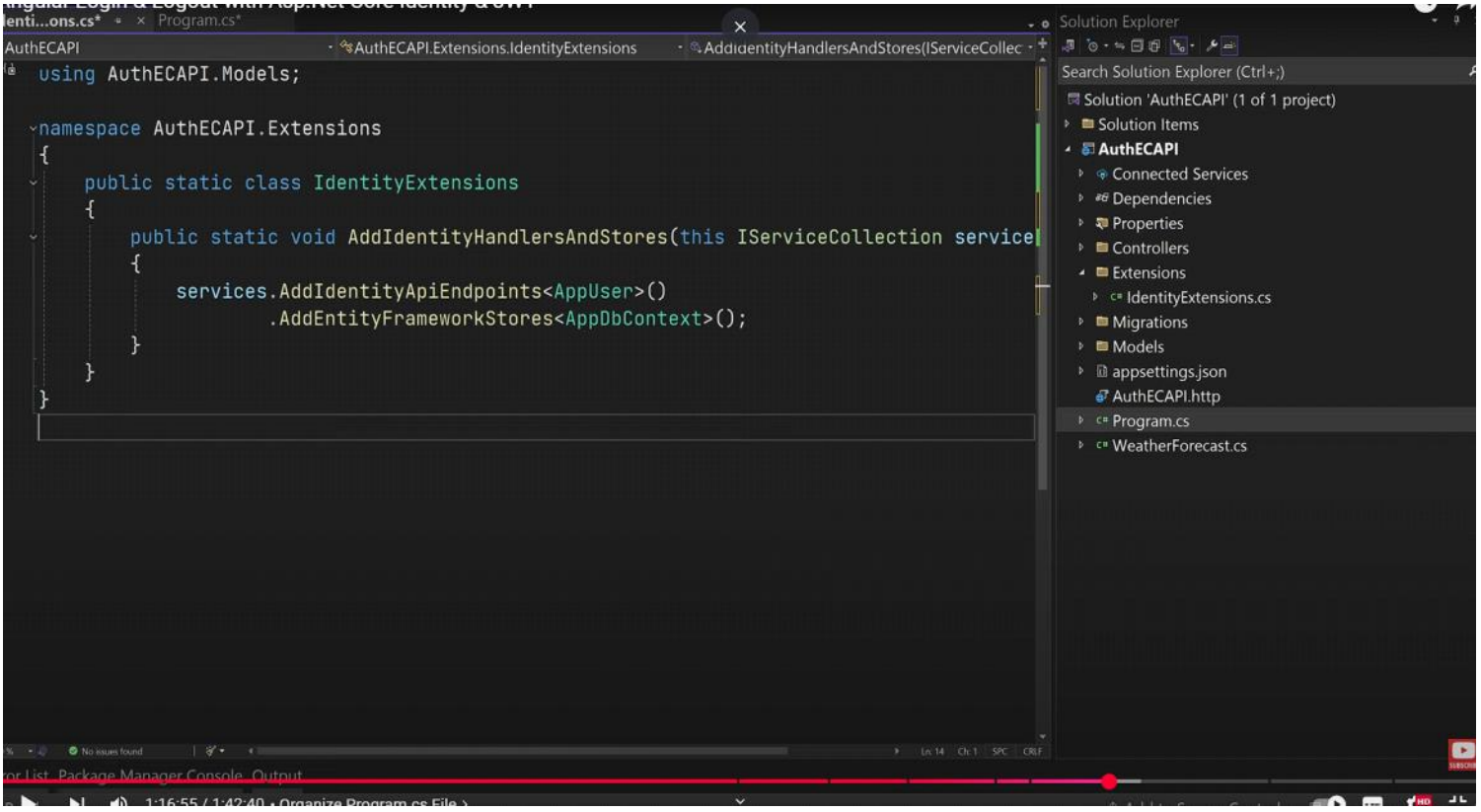

gular Login & Logout with Asp.Net Core Identity & JWT
gram.cs
thECAPI
UserRegistrationModel
Email
using AuthECAPI.Models;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swa
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
// Services from Identity Core.
builder.Services
.AddIdentityApiEndpoints<AppUser>()
.AddEntityFrameworkStores<AppDbContext>();
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Migrations
Models
Extensi
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs


Program.cs
AuthECAPI
UserRegistrationModel
Email
using AuthECAPI.Models;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore
using Microsoft.AspNetCore
using Microsoft.EntityFram
using Microsoft.IdentityMo
using System.IdentityModel
using System.Security.Clai
using System.Text;
var builder = WebApplicati
// Add services to the con
builder.Services.AddContro
// Learn more about config
builder.Services.AddEndpoi
builder.Services.AddSwagge
// Services from Identity
builder.Services
.AddIdentityApiEndpoints<AppUser>()
.AddEntityFrameworkStores<AppDbContext>();
Add New Item - AuthECAPI
Installed
C#
General
ASP.NET Core
Code
Data
General
Web
CSharp
Online
Sort by: Default
Class C#
Interface C#
Code File C#
Search (Ctrl+E)
Type: C#
An empty class declaration
Name: IdentityExtensions
Add
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
Error List Package Manager Console Output

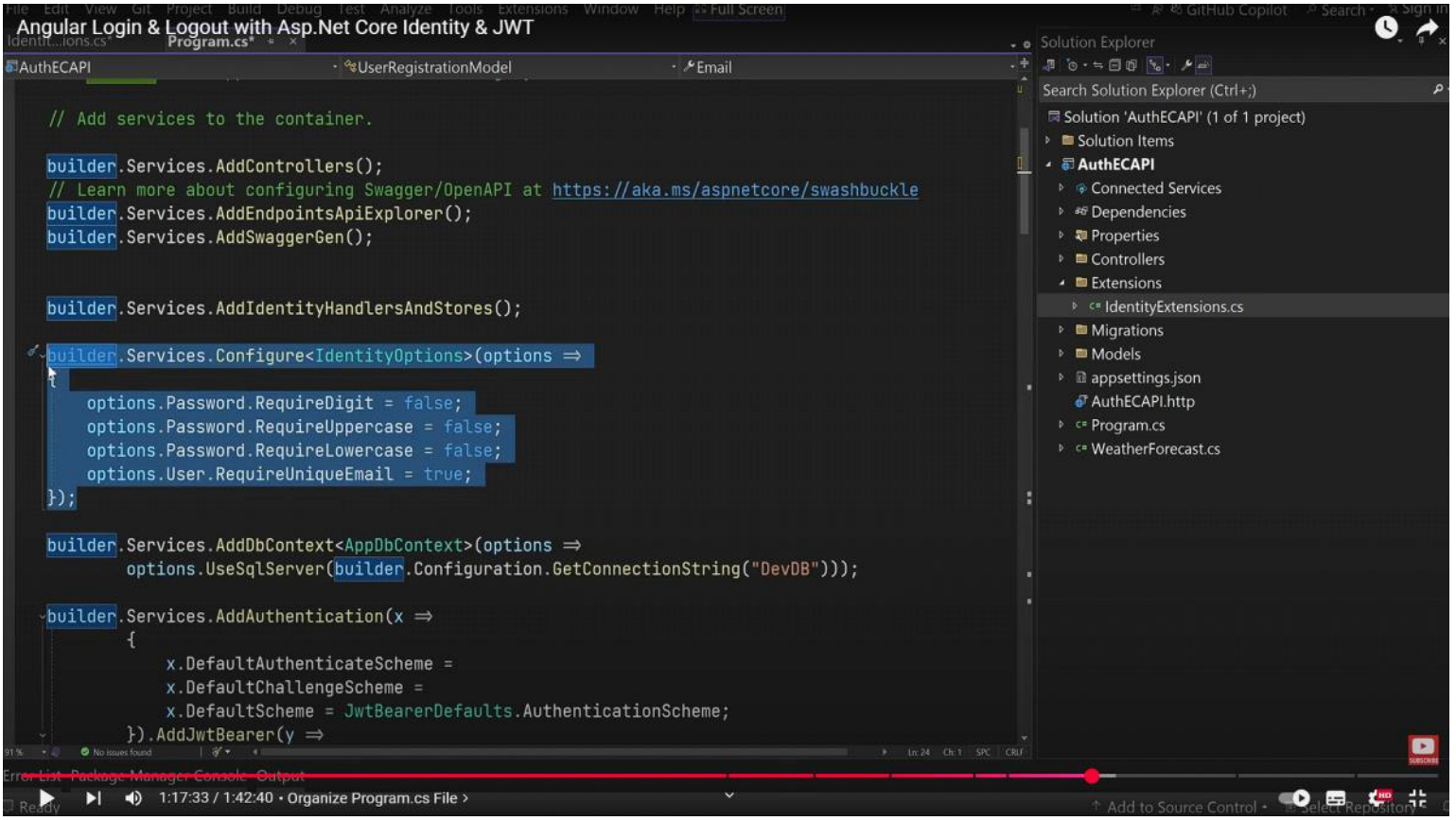

AuthECAPI
UserRegistrationModel
Email
// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
// Services from Identity Core.
builder.Services
.AddIdentityApiEndpoints<AppUser>()
.AddEntityFrameworkStores<AppDbContext>();
builder.Services.AddIdentityHandlersAndStores();
builder.Services.Configure<IdentityOptions>(options =>
{
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DevDB")));
builder.Services.AddAuthentication(x =>
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

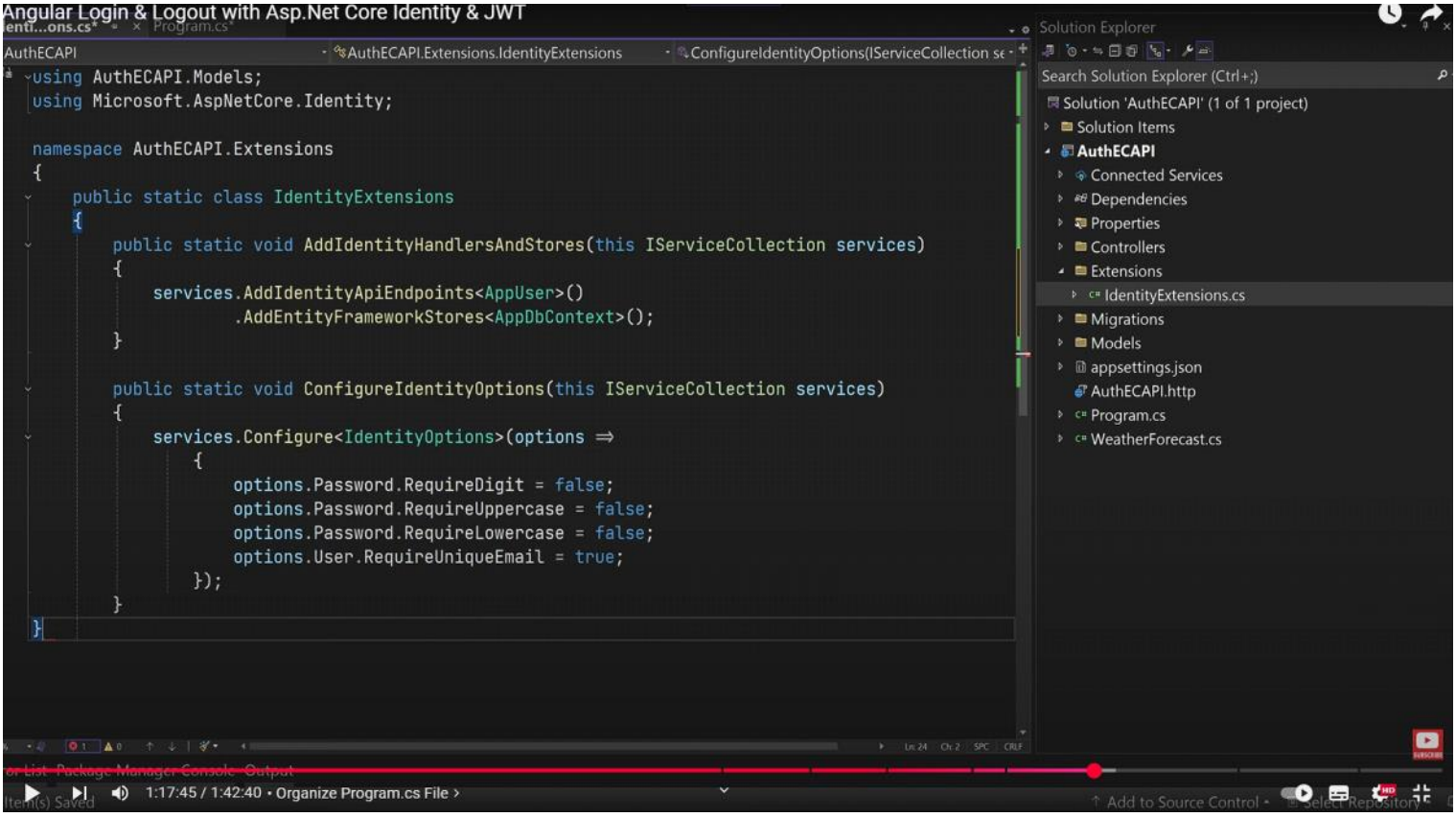
Cut that

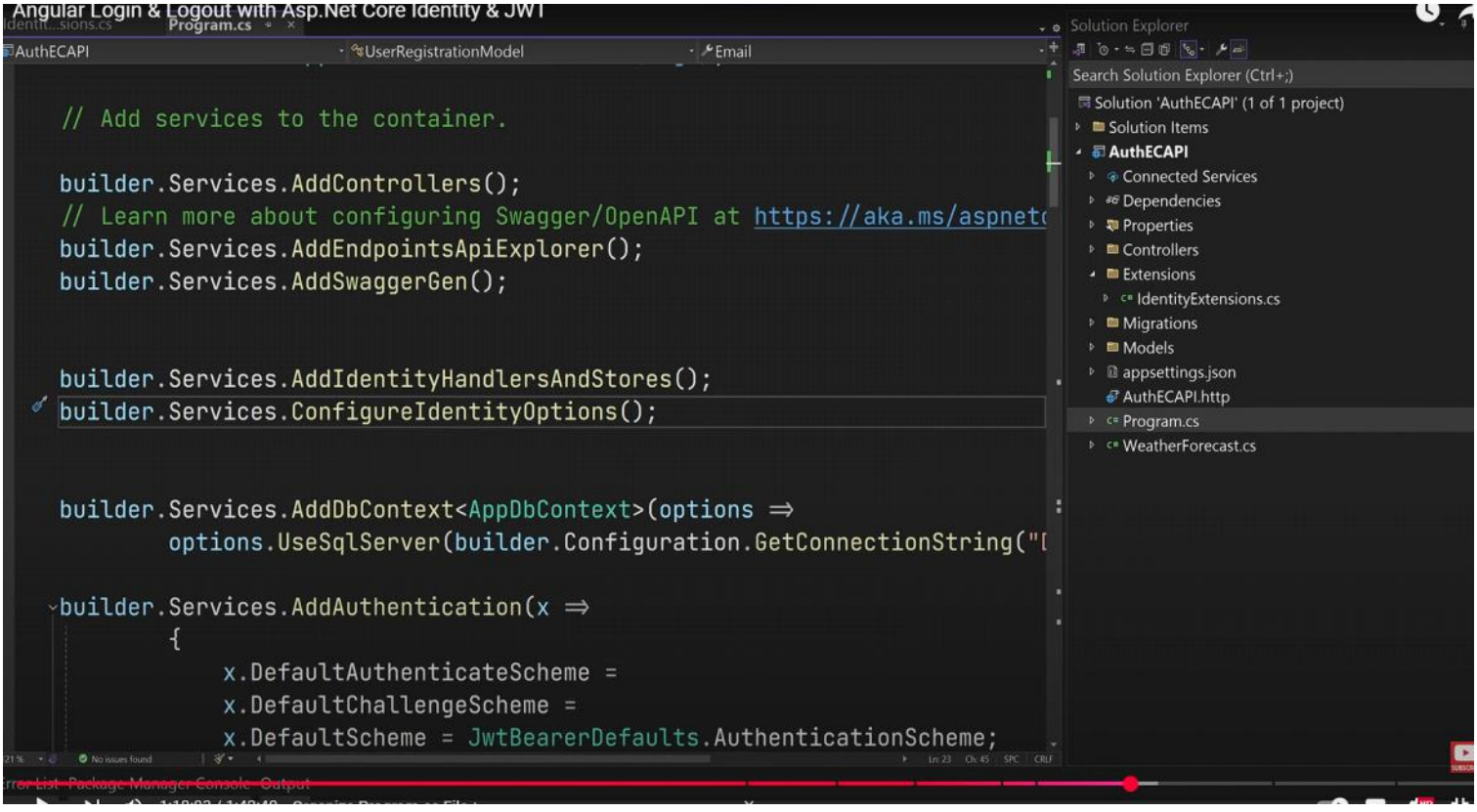

Program.cs*
AuthECAPI
UserRegistrationModel
Email
// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
// Services from Identity Core.
builder.Services
.AddIdentityApiEndpoints<AppUser>()
.AddEntityFrameworkStores<AppDbContext>();
builder.Services.AddIdentityHandlersAndStores();
builder.Services.Configure<IdentityOptions>(options =>
{
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DevDB")));
builder.Services.AddAuthentication(x =>
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

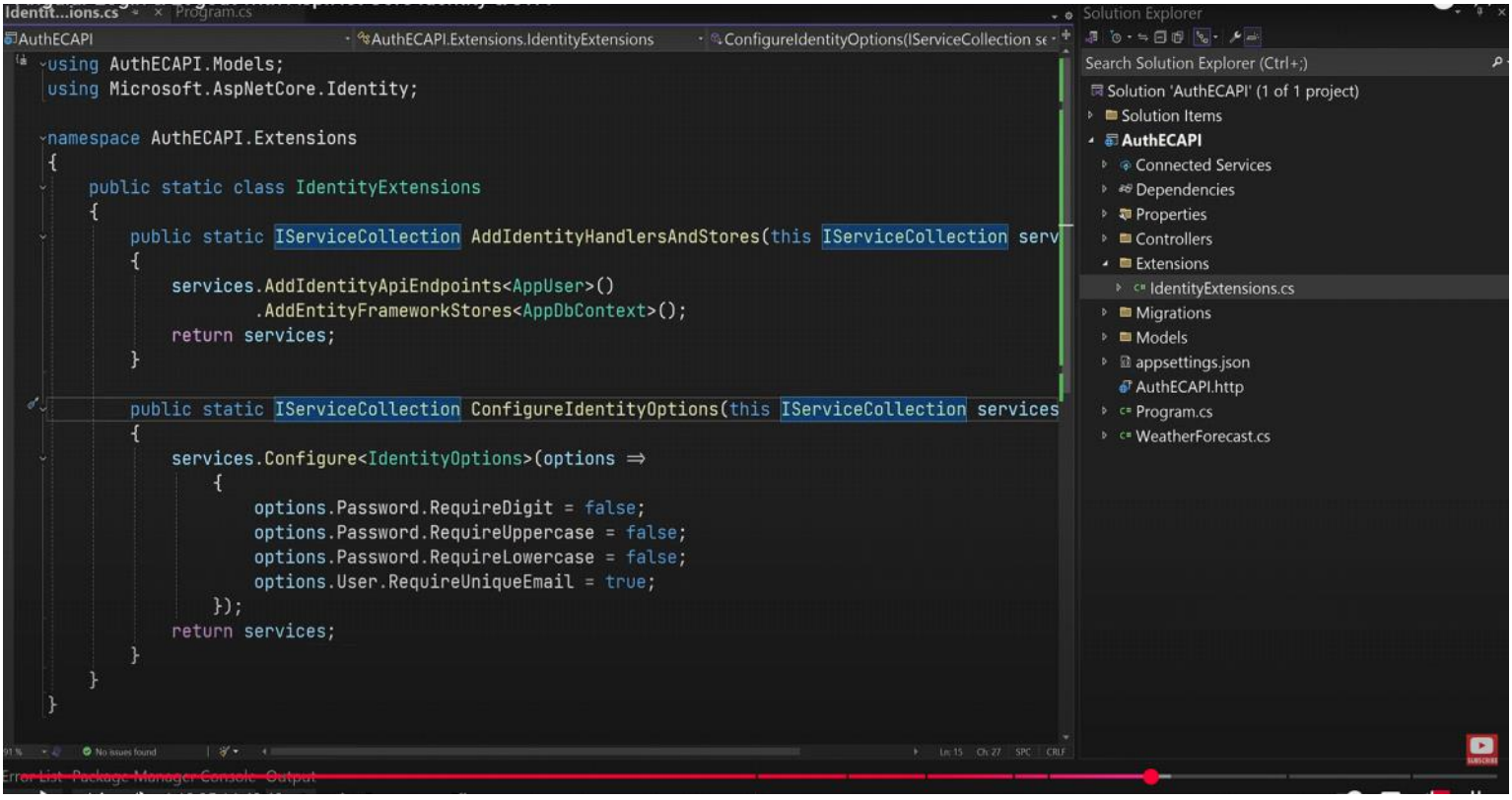

Program.cs*
AuthECAPI
AuthECAPI.Extensions.IdentityExtensions
using AuthECAPI.Models;
namespace AuthECAPI.Extensions
{
public static class IdentityExtensions
{
public static void AddIdentityHandlersAndStores(this IServiceCollection service
{
services.AddIdentityApiEndpoints<AppUser>()
.AddEntityFrameworkStores<AppDbContext>();
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

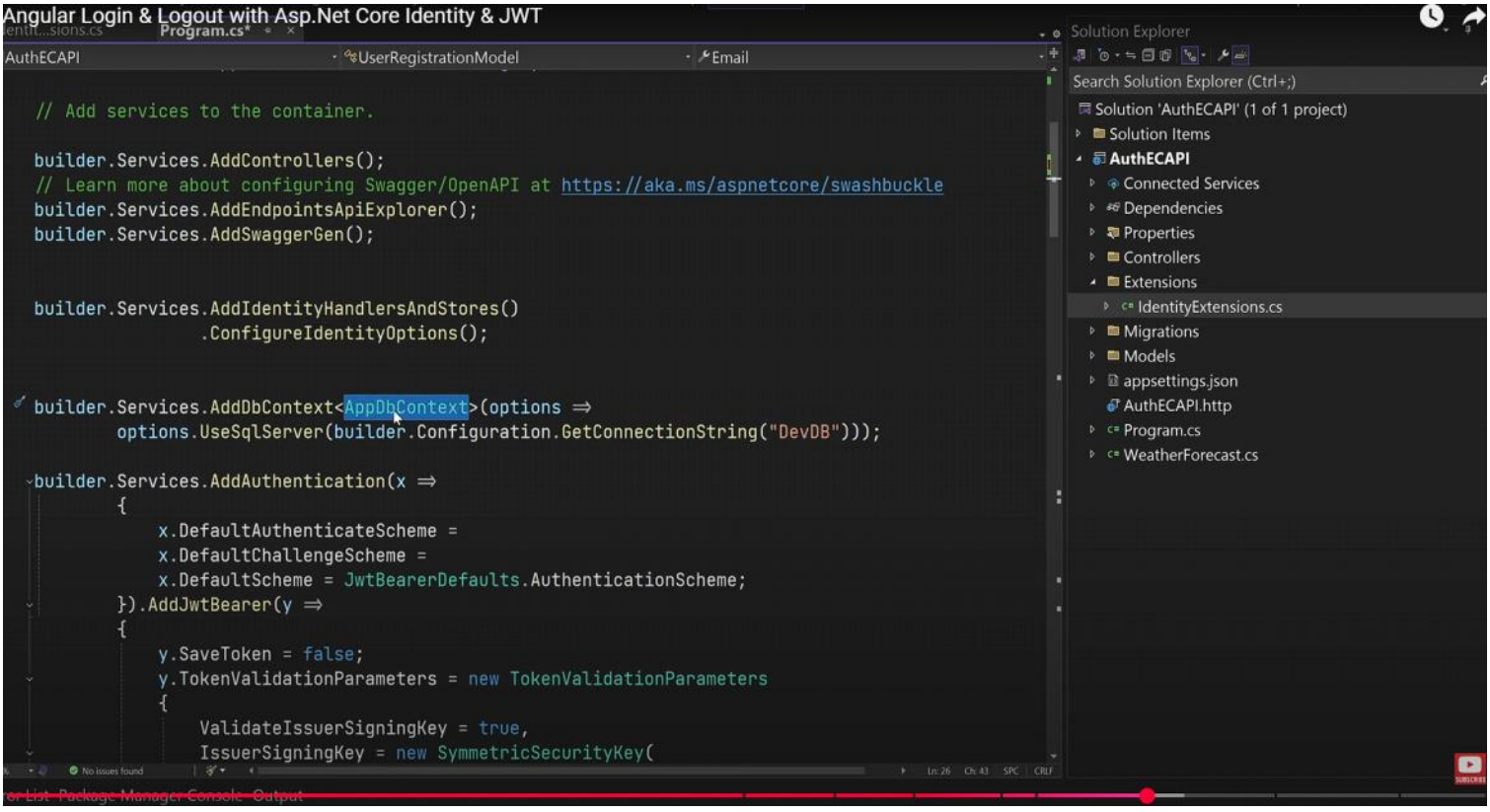
Cut this
Into

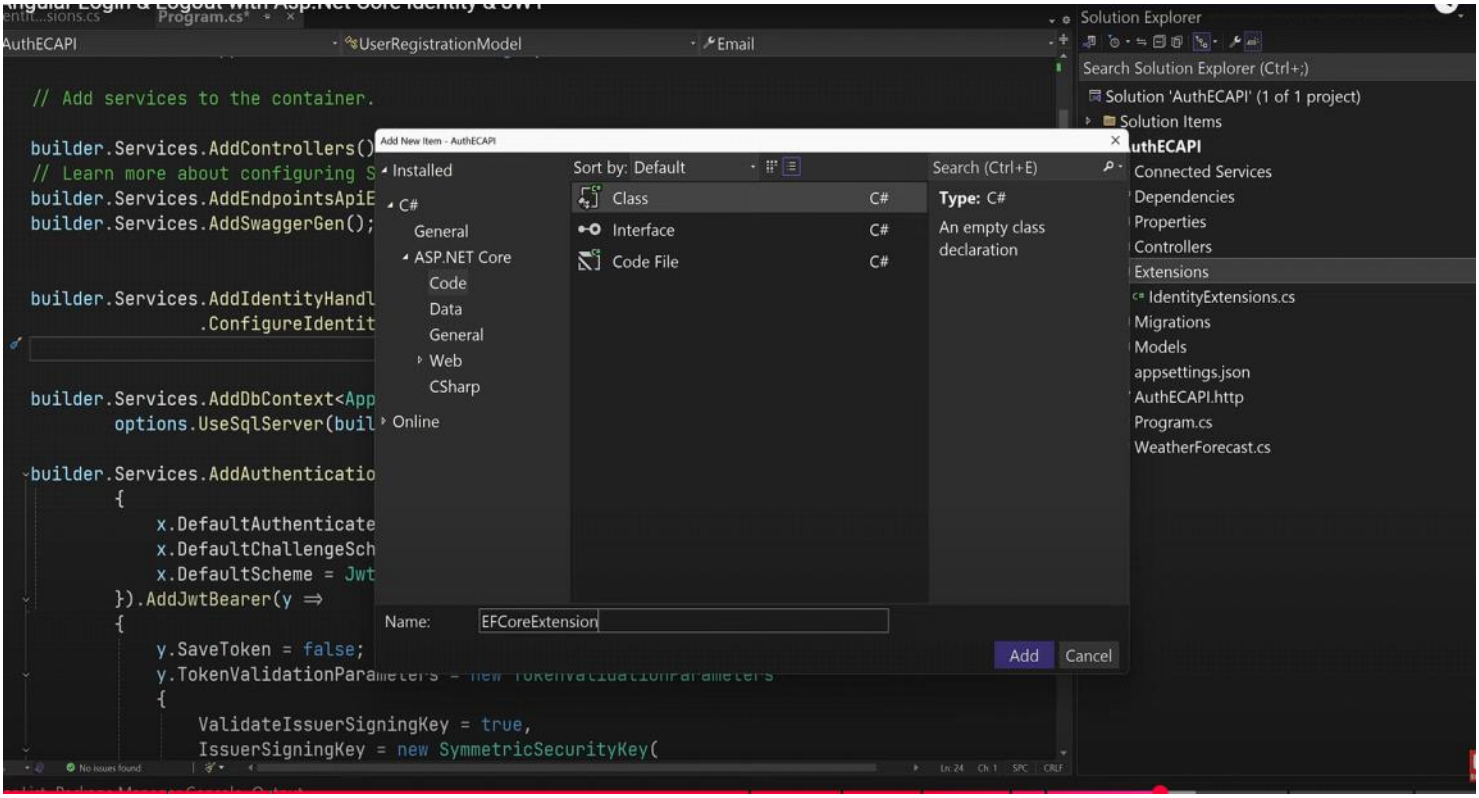


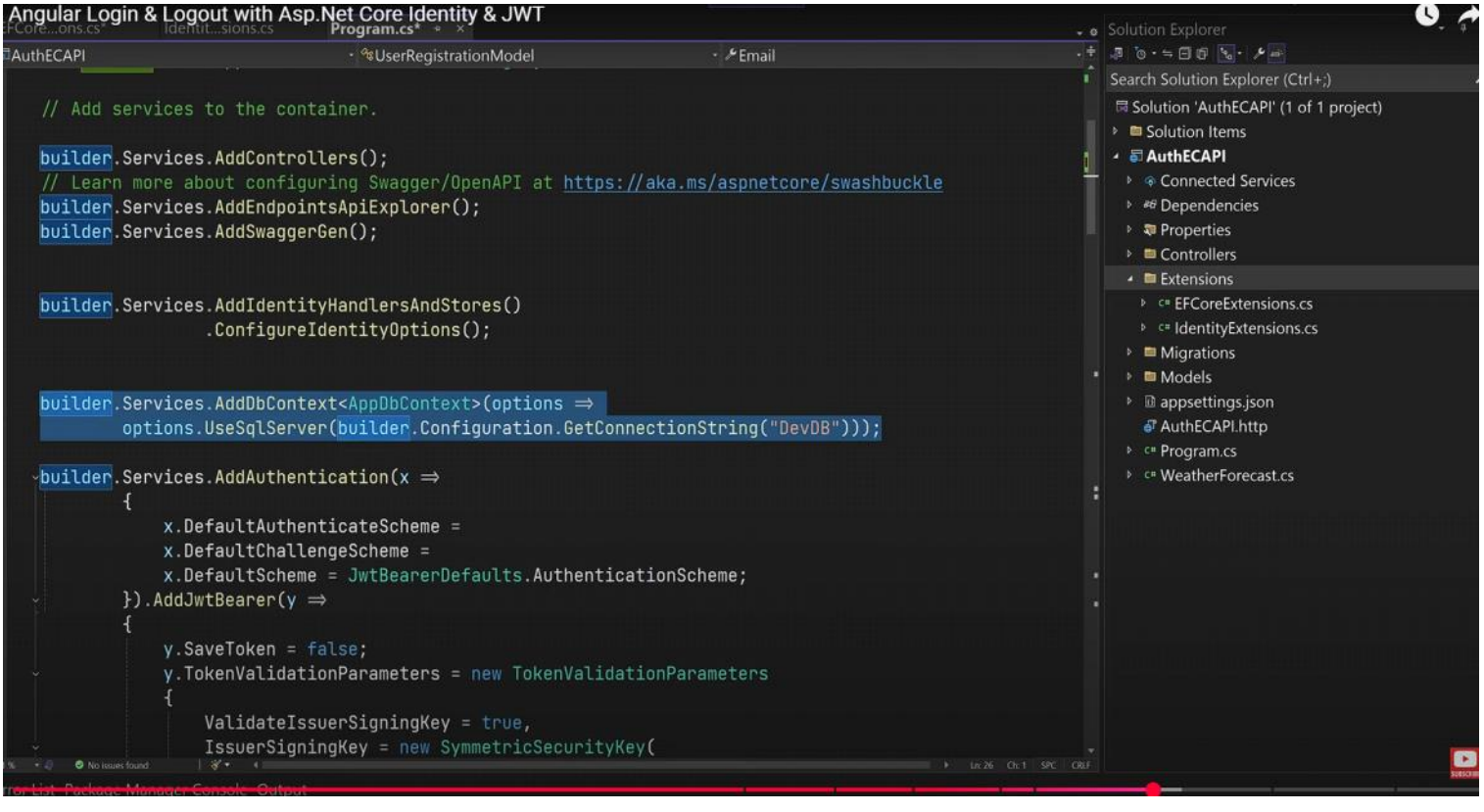

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECAPI
AuthECAPI.Extensions.IdentityExtensions
ConfigureIdentityOptions(IServiceCollection se
using AuthECAPI.Models;
using Microsoft.AspNetCore.Identity;
namespace AuthECAPI.Extensions
{
public static class IdentityExtensions
{
public static void AddIdentityHandlersAndStores(this IServiceCollection services)
{
services.AddIdentityApiEndpoints<AppUser>()
.AddEntityFrameworkStores<AppDbContext>();
}
public static void ConfigureIdentityOptions(this IServiceCollection services)
{
services.Configure<IdentityOptions>(options =>
{
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:17:45 / 1:42:40 • Organize Program.cs File

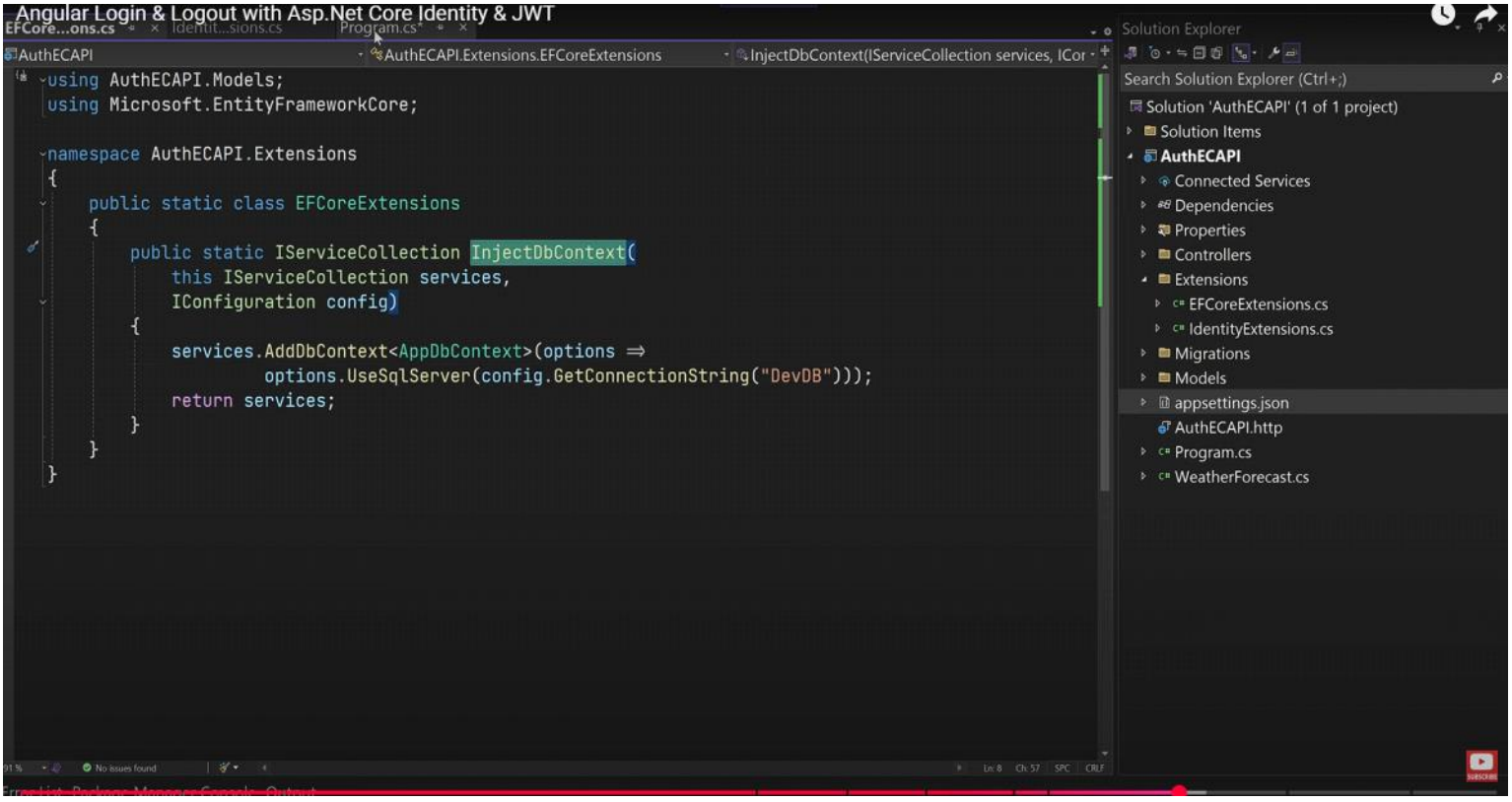

Angular Login & Logout with Asp.Net Core Identity & JWT
Program.cs
AuthECAPI
UserRegistrationModel
Email
// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetc
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddIdentityHandlersAndStores();
builder.Services.ConfigureIdentityOptions();
builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("D
builder.Services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

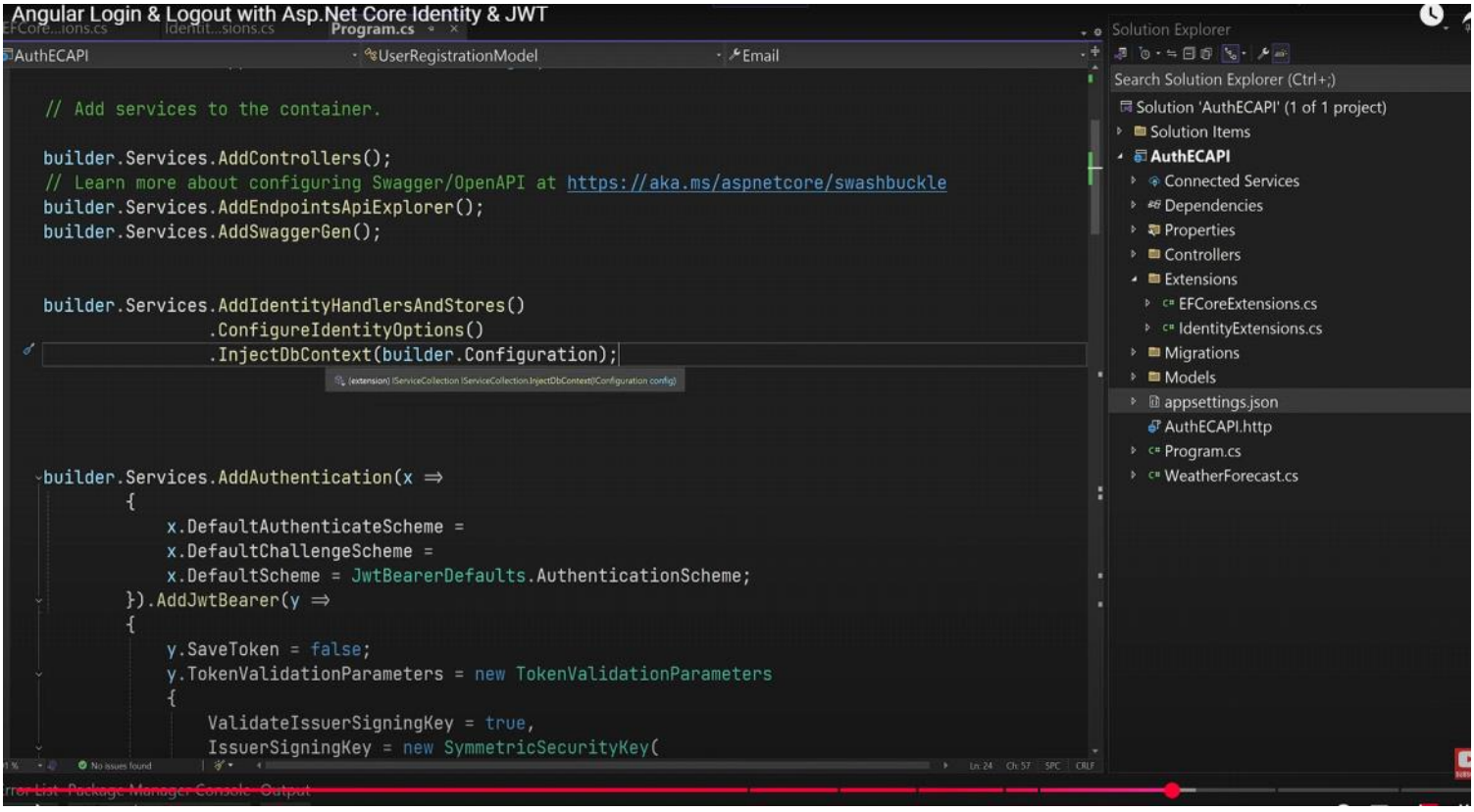

AuthECAPI
AuthECAPI.Extensions.IdentityExtensions
ConfigureIdentityOptions(IServiceCollection se
using AuthECAPI.Models;
using Microsoft.AspNetCore.Identity;
namespace AuthECAPI.Extensions
{
public static class IdentityExtensions
{
public static IServiceCollection AddIdentityHandlersAndStores(this IServiceCollection serv
{
services.AddIdentityApiEndpoints<AppUser>()
.AddEntityFrameworkStores<AppDbContext>();
return services;
}
public static IServiceCollection ConfigureIdentityOptions(this IServiceCollection services
{
services.Configure<IdentityOptions>(options =>
{
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
return services;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

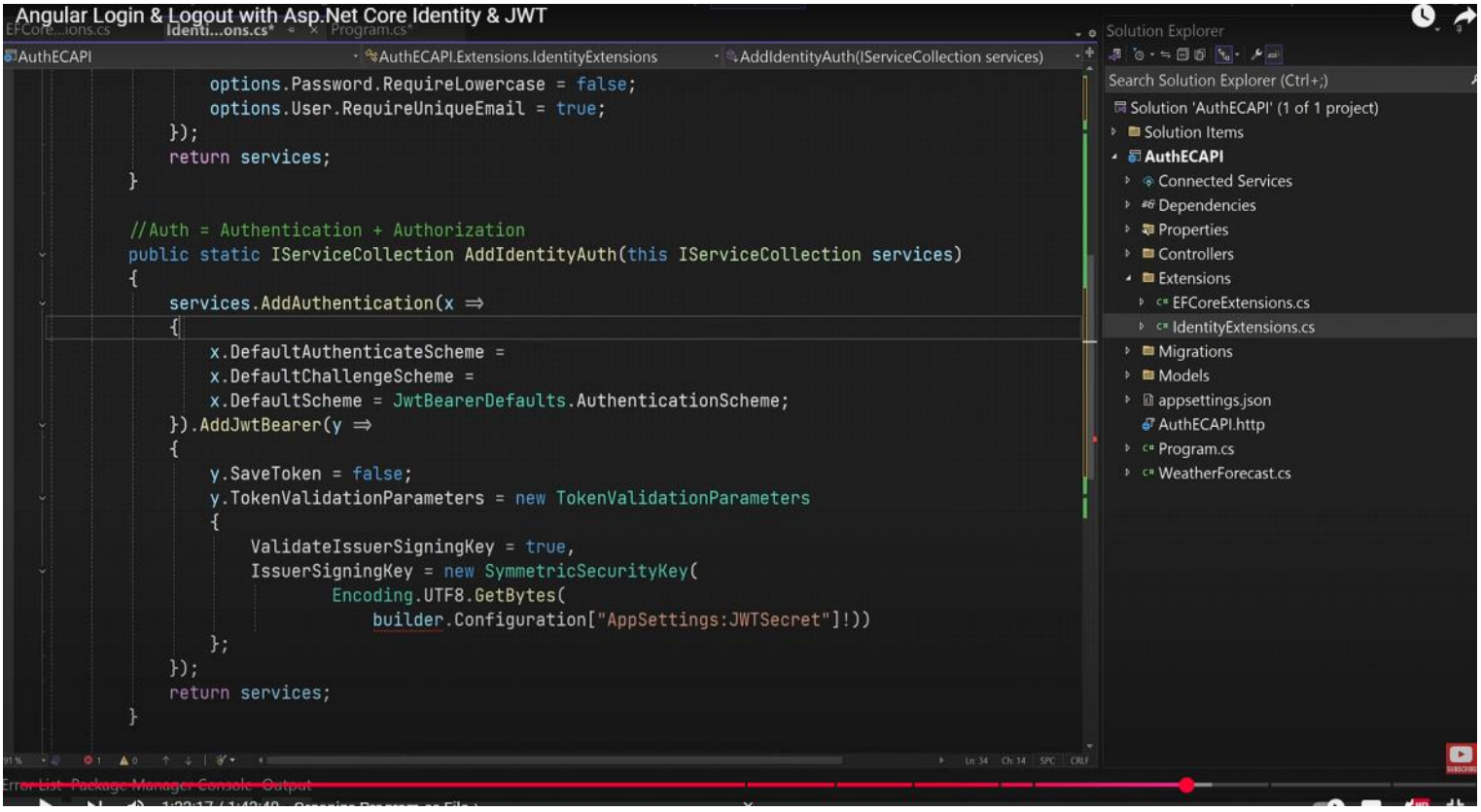

Angular Login & Logout with Asp.Net Core Identity & JWT
Program.cs*
AuthECAPI
UserRegistrationModel
Email
// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions();
builder.Services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DevDB")));
builder.Services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

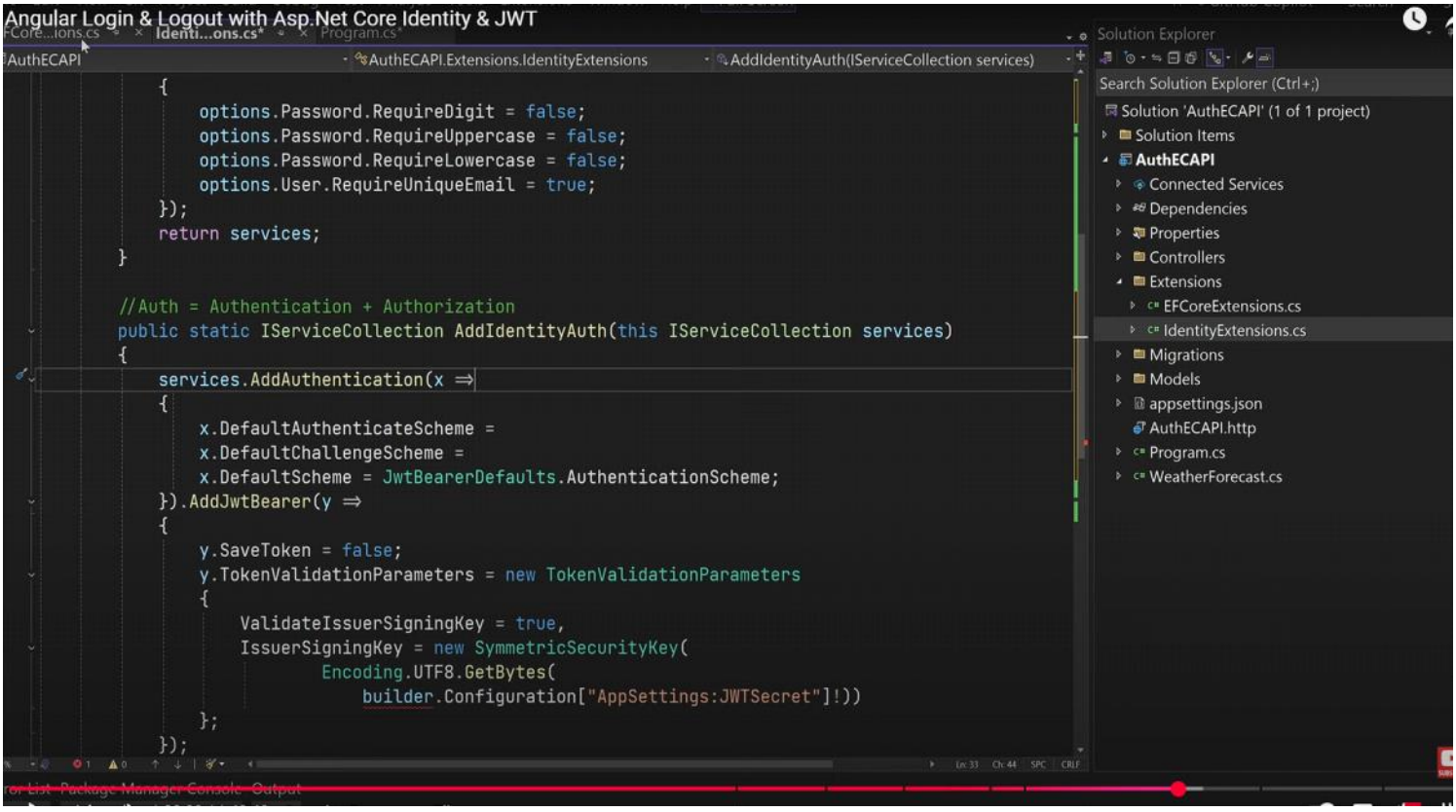

Add New Item - AuthECAPI
Installed
C#
General
ASP.NET Core
Code
Data
General
Web
CSharp
Online
Sort by: Default
Class C#
Interface C#
Code File C#
Search (Ctrl+E)
Type: C#
An empty class declaration
Name: EFCoreExtension
Add
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

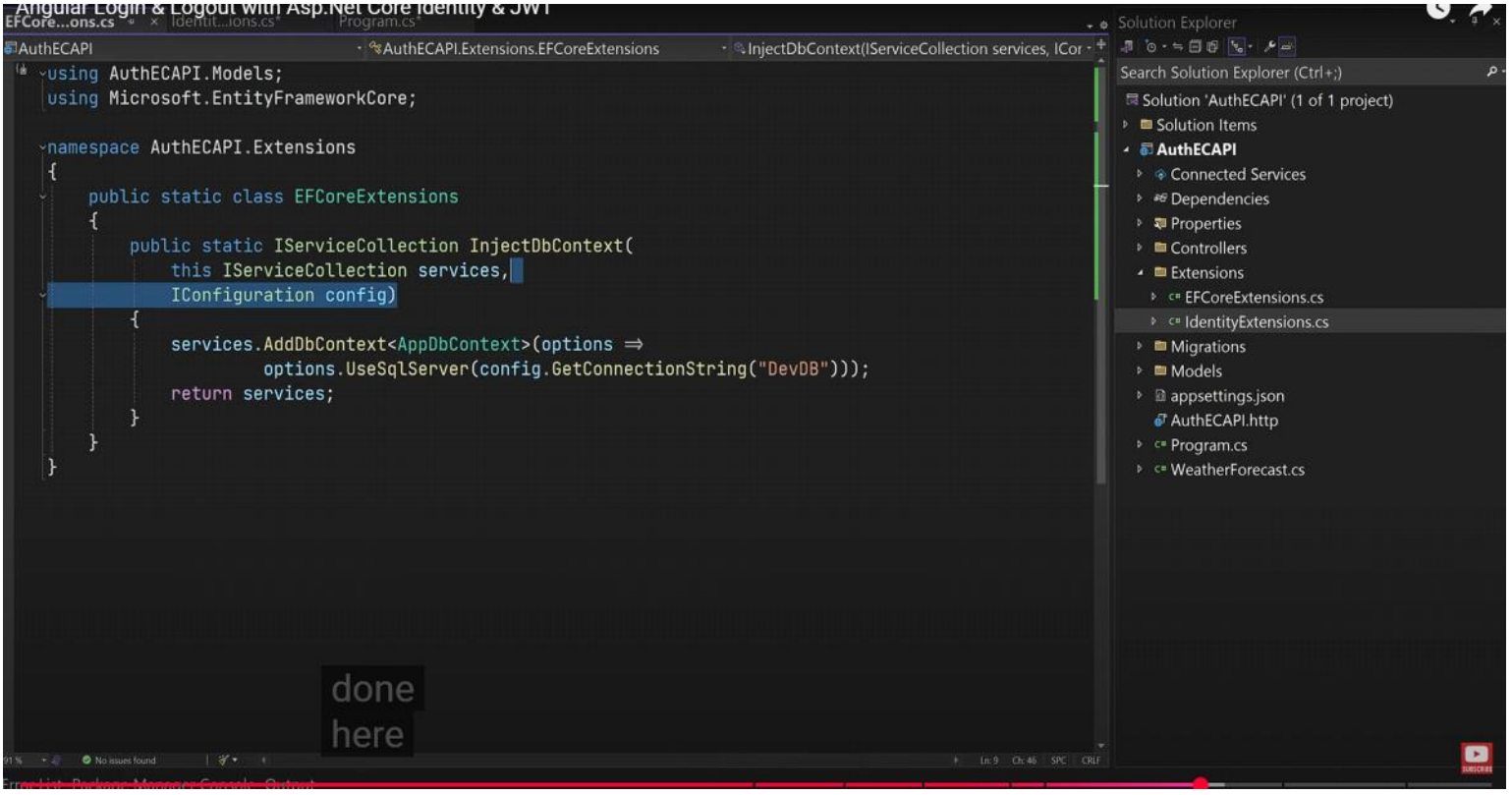

Angular Login & Logout with Asp.Net Core Identity & JWT
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

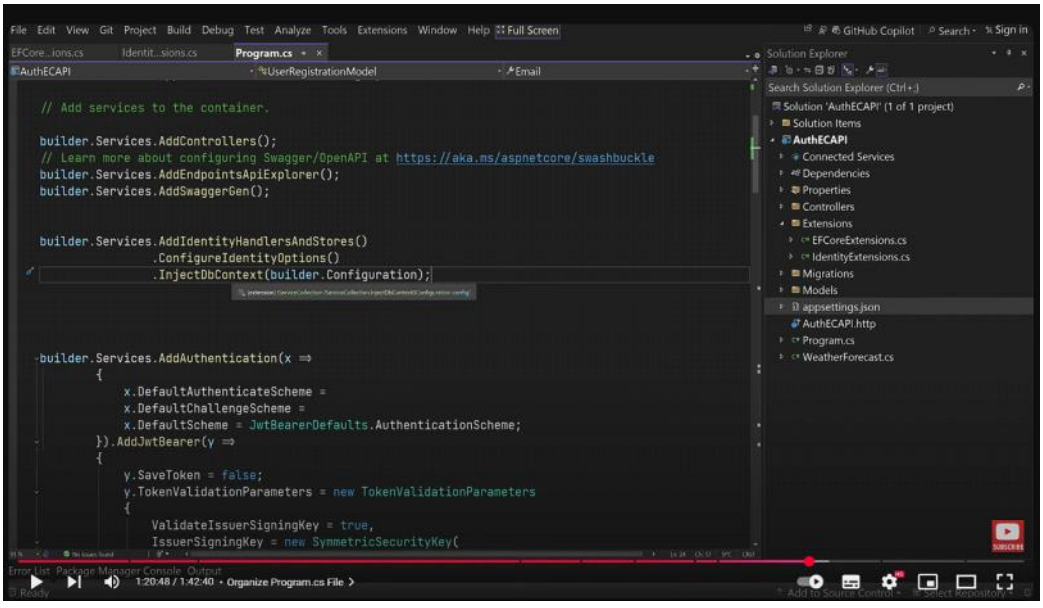

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECAPI
AuthECAPI.Extensions.EFCoreExtensions
InjectDbContext(IServiceCollection services, ICor
using AuthECAPI.Models;
using Microsoft.EntityFrameworkCore;
namespace AuthECAPI.Extensions
{
public static class EFCoreExtensions
{
public static IServiceCollection InjectDbContext(
this IServiceCollection services,
IConfiguration config)
{
services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(config.GetConnectionString("DevDB")));
return services;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

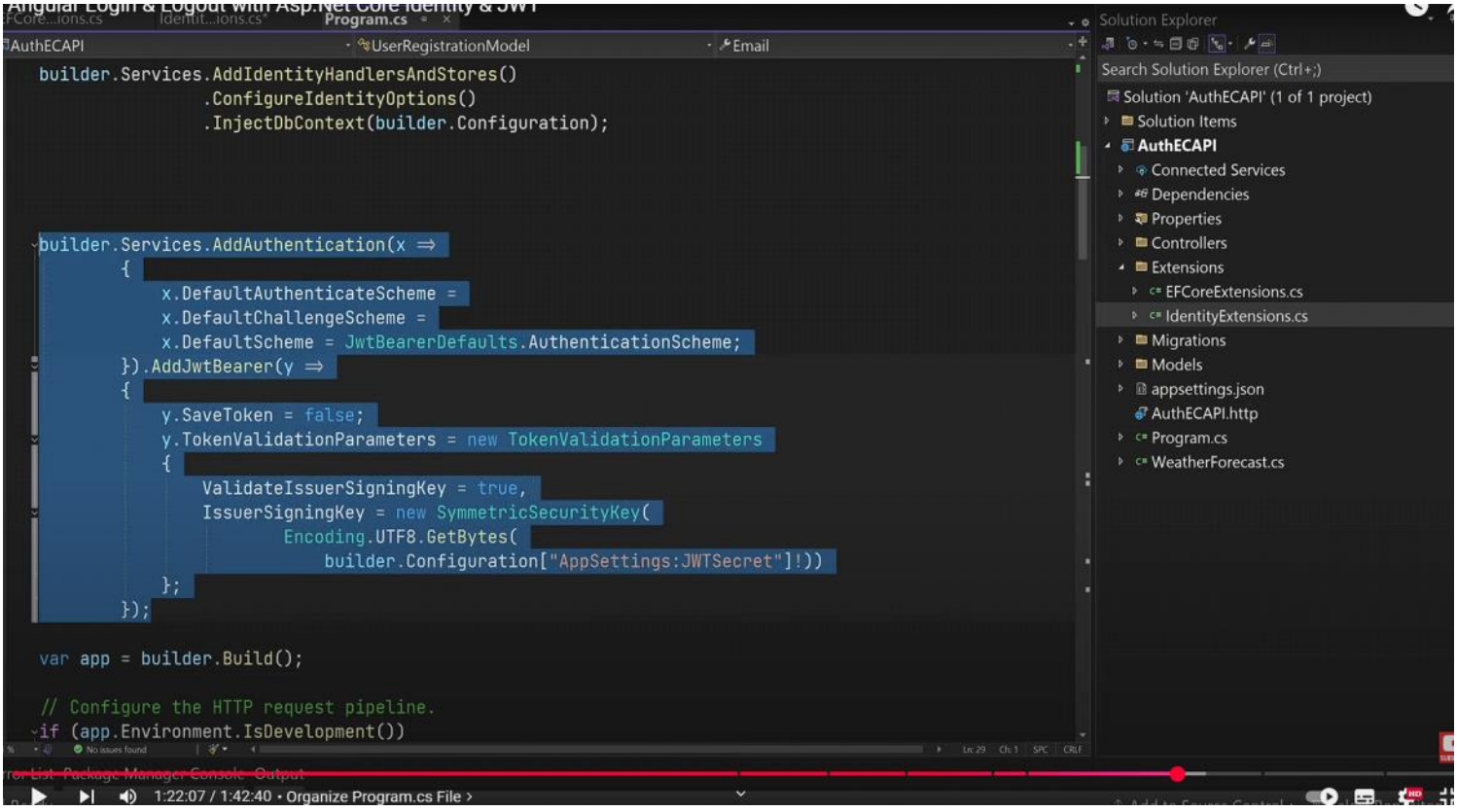

Angular Login & Logout with Asp.Net Core Identity & JWT
Program.cs
AuthECAPI
UserRegistrationModel
Email
// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions()
.InjectDbContext(builder.Configuration);
builder.Services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

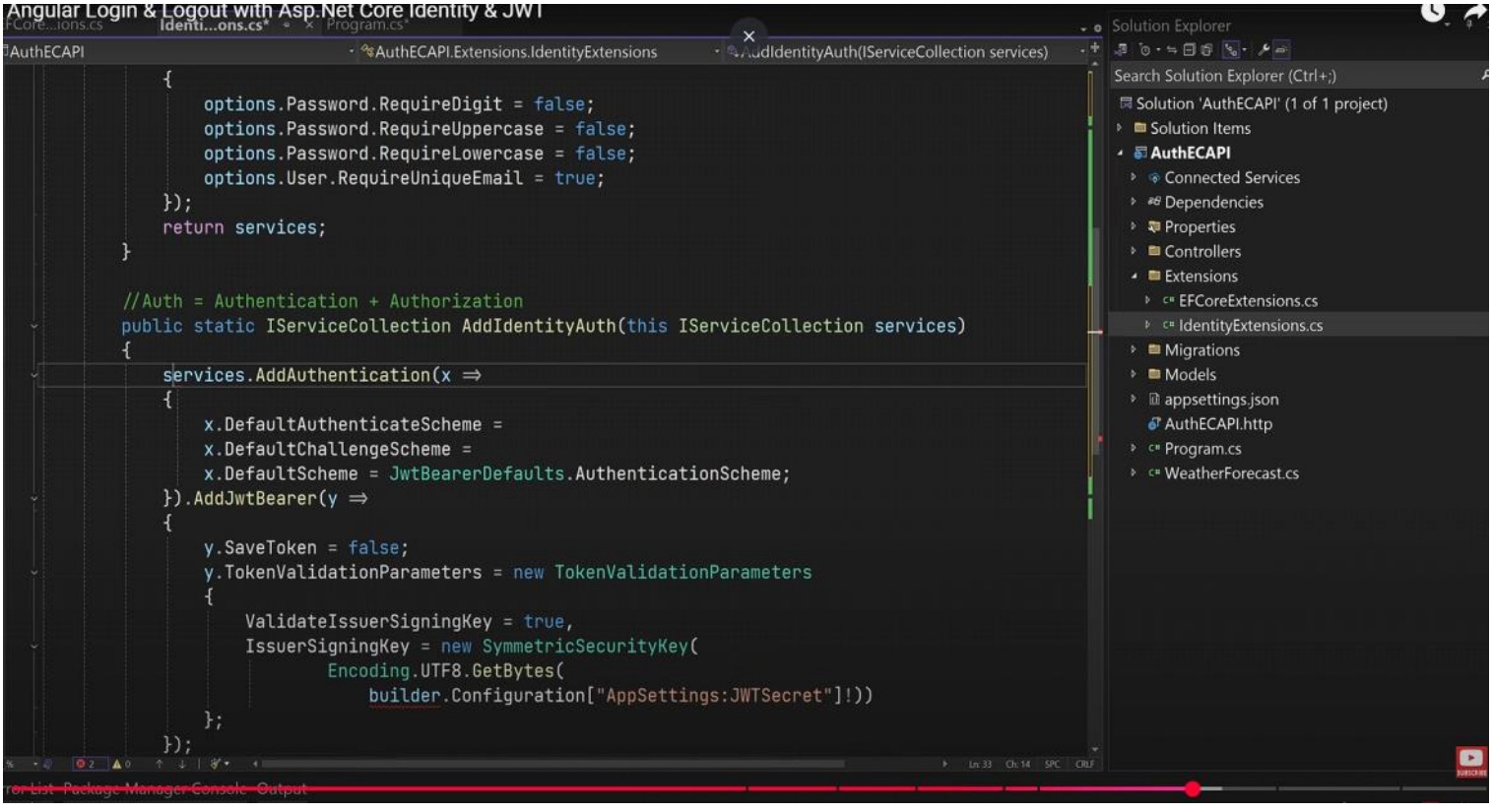

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECAPI
AuthECAPI.Extensions.IdentityExtensions
AddIdentityAuth(IServiceCollection services)
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
return services;
}
//Auth = Authentication + Authorization
public static IServiceCollection AddIdentityAuth(this IServiceCollection services)
{
services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Encoding.UTF8.GetBytes(
builder.Configuration["AppSettings:JWTSecret"]!))
};
});
return services;
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

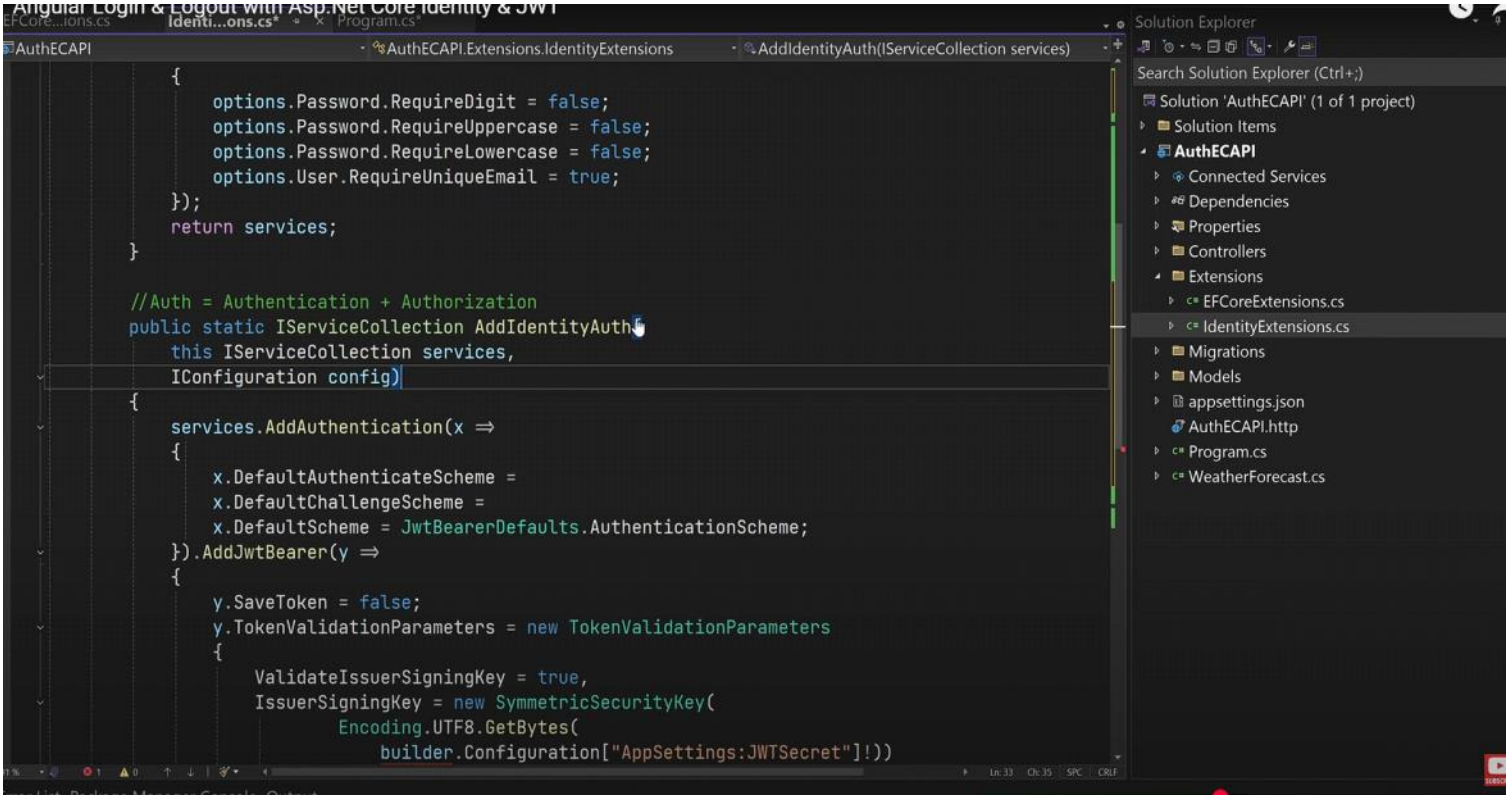

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECAPI
AuthECAPI.Extensions.IdentityExtensions
AddIdentityAuth(IServiceCollection services)
{
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
return services;
}
//Auth = Authentication + Authorization
public static IServiceCollection AddIdentityAuth(this IServiceCollection services)
{
services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Encoding.UTF8.GetBytes(
builder.Configuration["AppSettings:JWTSecret"]!))
};
});
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

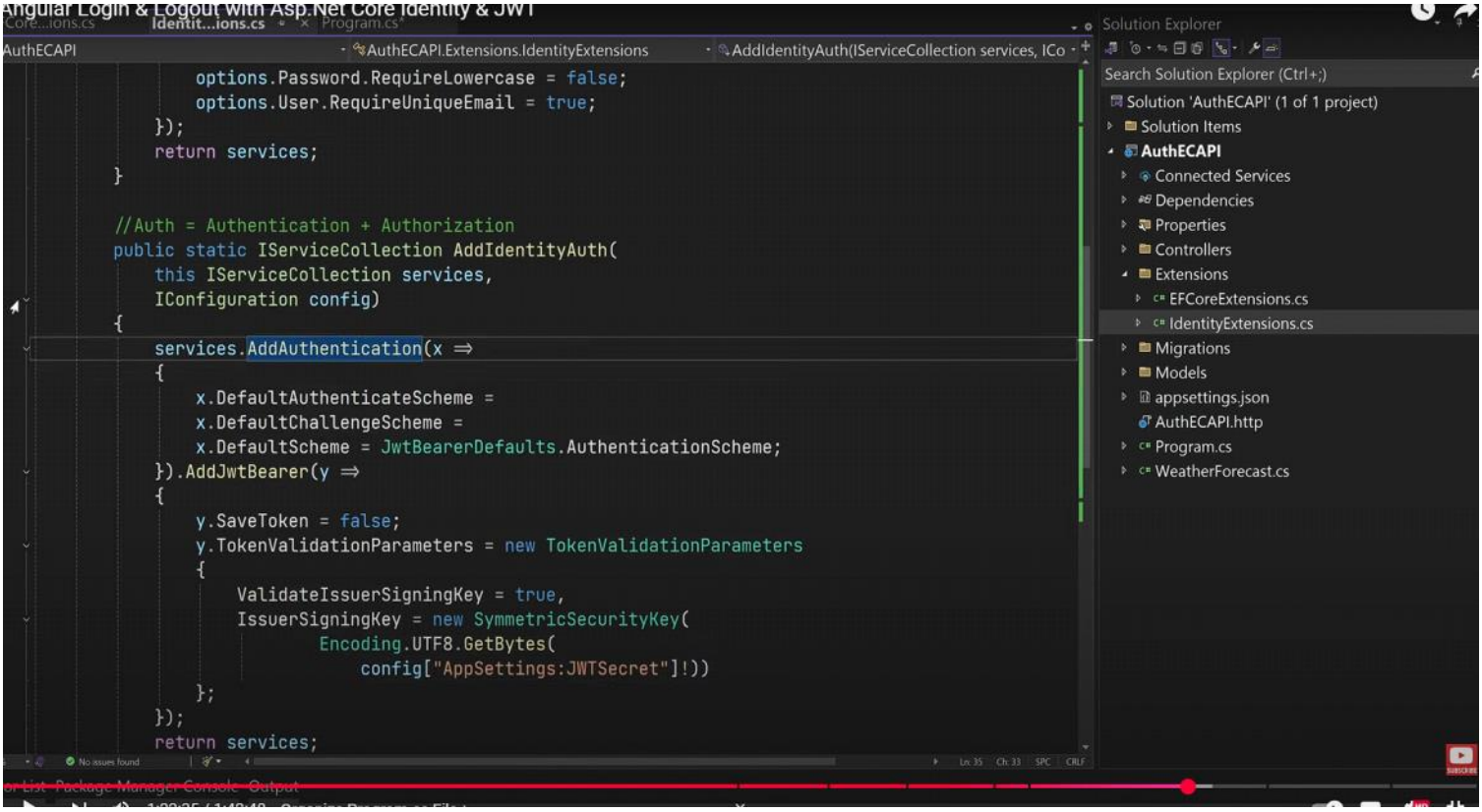

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECAPI
AuthECAPI.Extensions.EFCoreExtensions
InjectDbContext(IServiceCollection services, ICor
using AuthECAPI.Models;
using Microsoft.EntityFrameworkCore;
namespace AuthECAPI.Extensions
{
public static class EFCoreExtensions
{
public static IServiceCollection InjectDbContext(
this IServiceCollection services,
IConfiguration config)
{
services.AddDbContext<AppDbContext>(options =>
options.UseSqlServer(config.GetConnectionString("DevDB")));
return services;
}
}
}
done
here
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

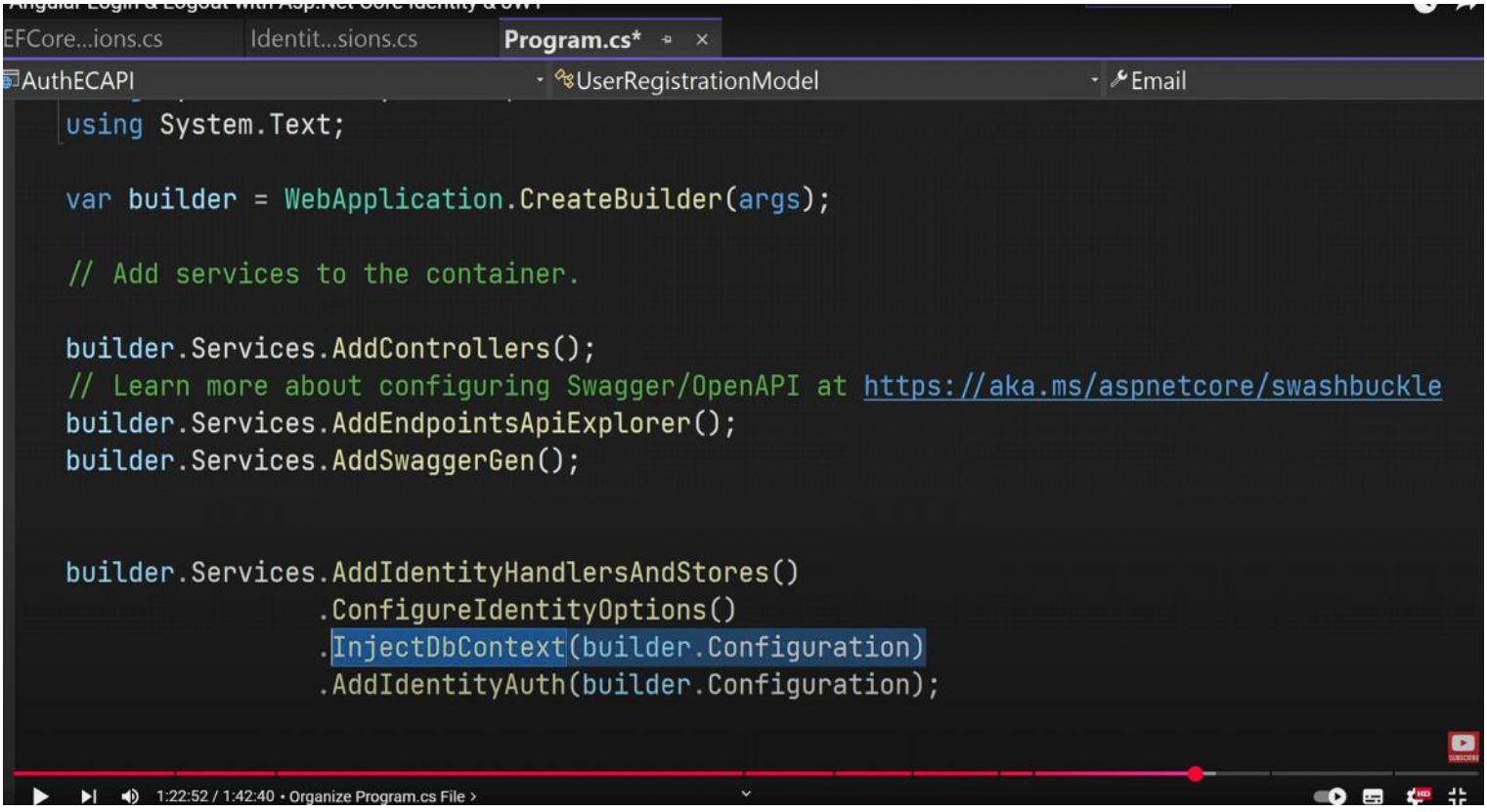

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Full Screen
GitHub Copilot
Search
Sign in
Program.cs
AuthECAPI
UserRegistrationModel
Email
// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions()
.InjectDbContext(builder.Configuration);
builder.Services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:20:48 / 1:42:40 • Organize Program.cs File

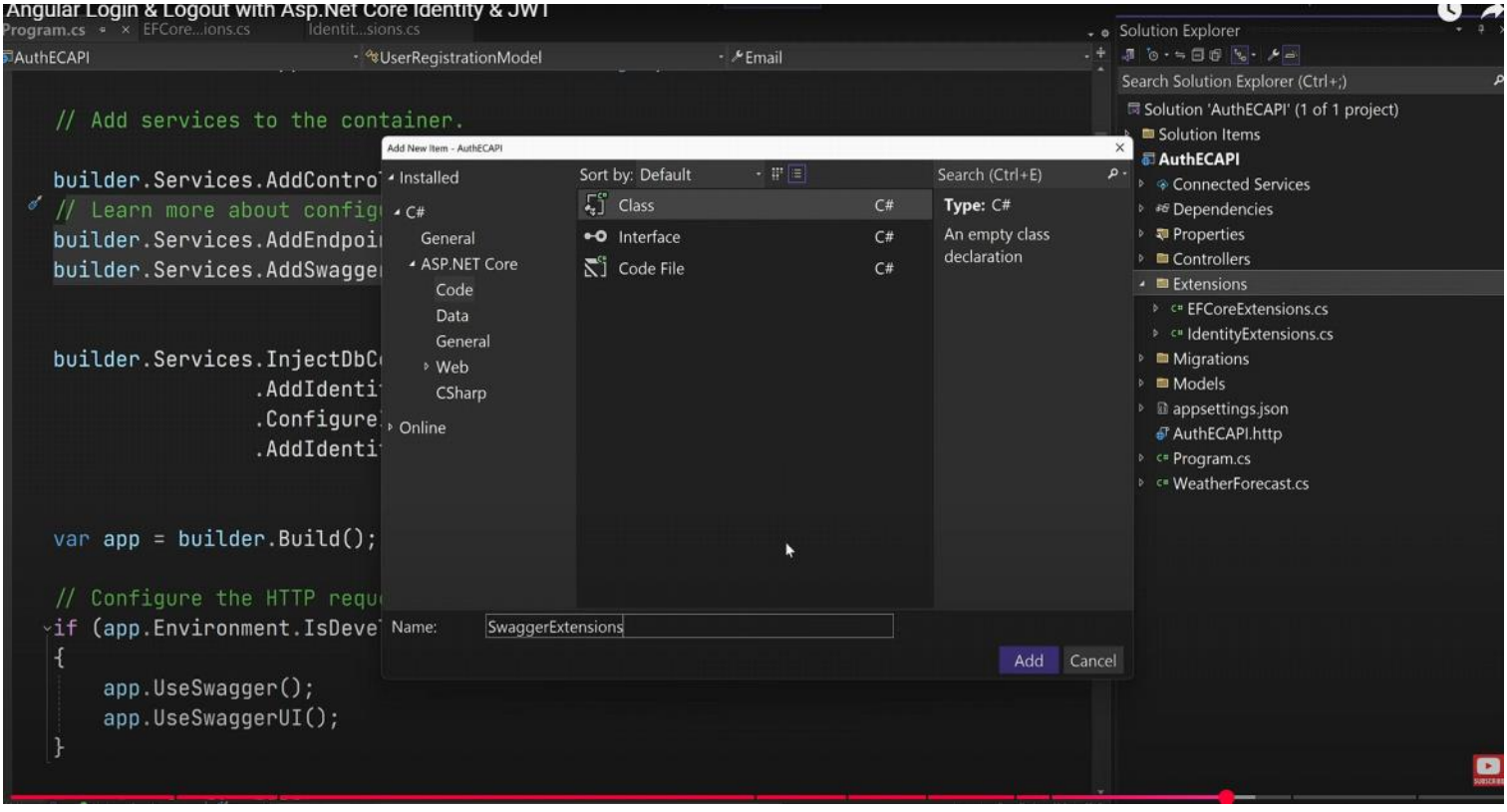

Angular Login & Logout with Asp.Net Core Identity & JWT
Program.cs
AuthECAPI
UserRegistrationModel
Email
builder.Services.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions()
.InjectDbContext(builder.Configuration);
builder.Services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Encoding.UTF8.GetBytes(
builder.Configuration["AppSettings:JWTSecret"]!))
};
});
var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:22:07 / 1:42:40 • Organize Program.cs File ›

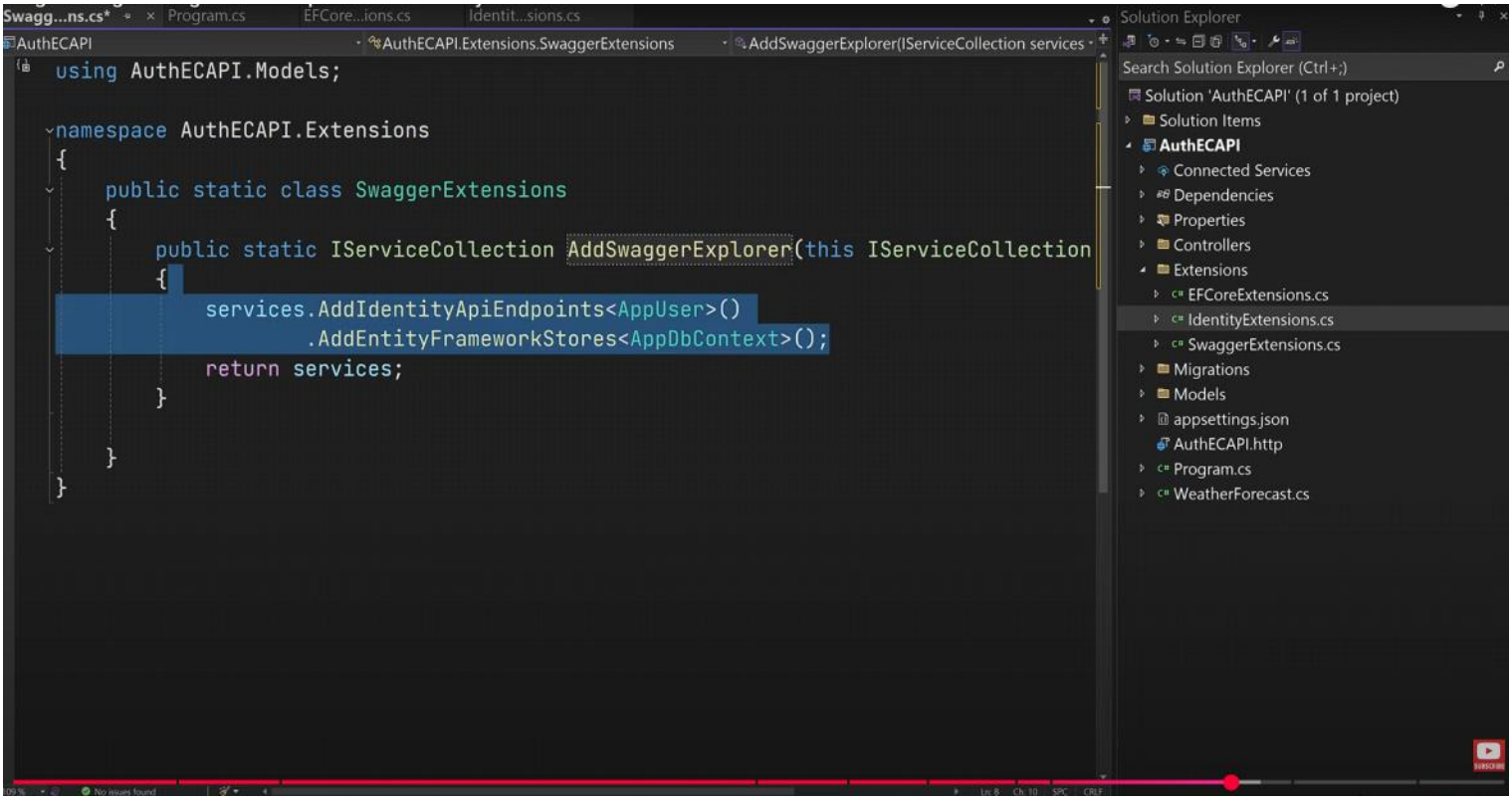

Angular Login & Logout with Asp.Net Core Identity & JWT
Identi...ons.cs*
Program.cs
AuthECAPI
AuthECAPI.Extensions.IdentityExtensions
{
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
return services;
}
//Auth = Authentication + Authorization
public static IServiceCollection AddIdentityAuth(this IServiceCollection services)
{
services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Encoding.UTF8.GetBytes(
builder.Configuration["AppSettings:JWTSecret"]!))
};
});
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

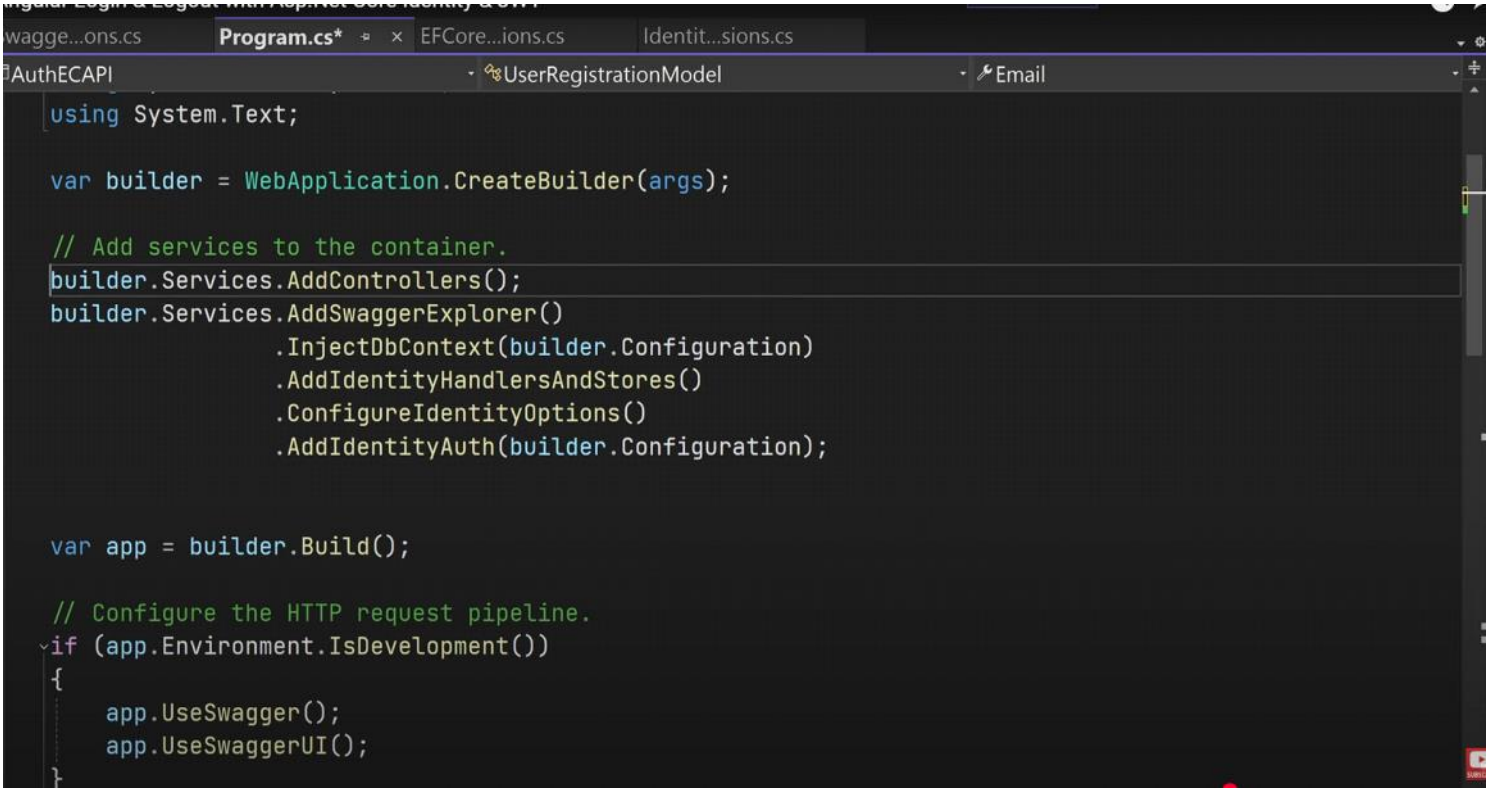

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECAPI
AuthECAPI.Extensions.IdentityExtensions
AddIdentityAuth(IServiceCollection services)
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
return services;
}
//Auth = Authentication + Authorization
public static IServiceCollection AddIdentityAuth(
this IServiceCollection services,
IConfiguration config)
{
services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Encoding.UTF8.GetBytes(
builder.Configuration["AppSettings:JWTSecret"]!))
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

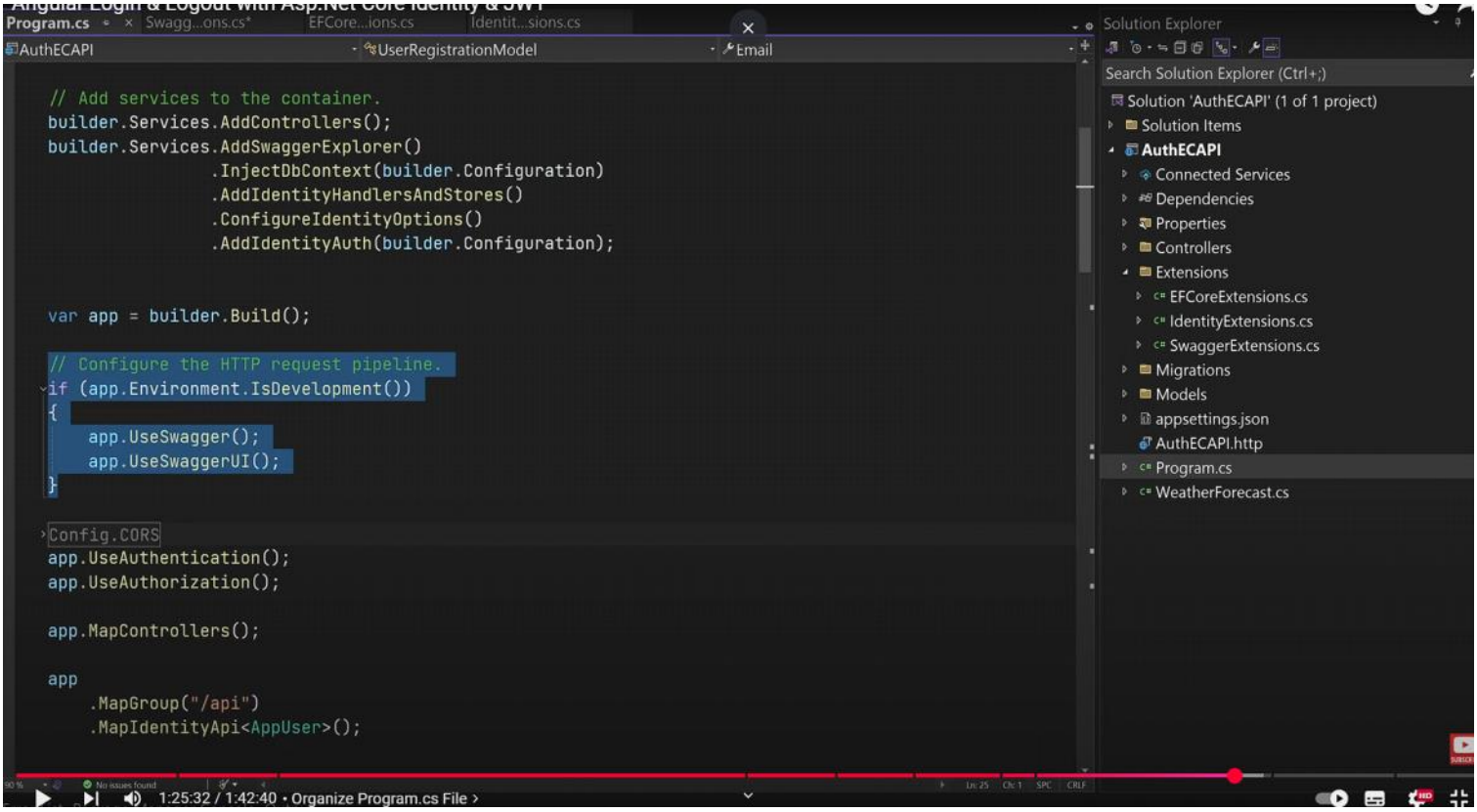

Angular Login & Logout with Asp.Net Core Identity & JWT
AuthECAPI
AuthECAPI.Extensions.IdentityExtensions
AddIdentityAuth(IServiceCollection services, ICo
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
return services;
}
//Auth = Authentication + Authorization
public static IServiceCollection AddIdentityAuth(
this IServiceCollection services,
IConfiguration config)
{
services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme =
x.DefaultChallengeScheme =
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(y =>
{
y.SaveToken = false;
y.TokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(
Encoding.UTF8.GetBytes(
config["AppSettings:JWTSecret"]!))
};
});
return services;
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

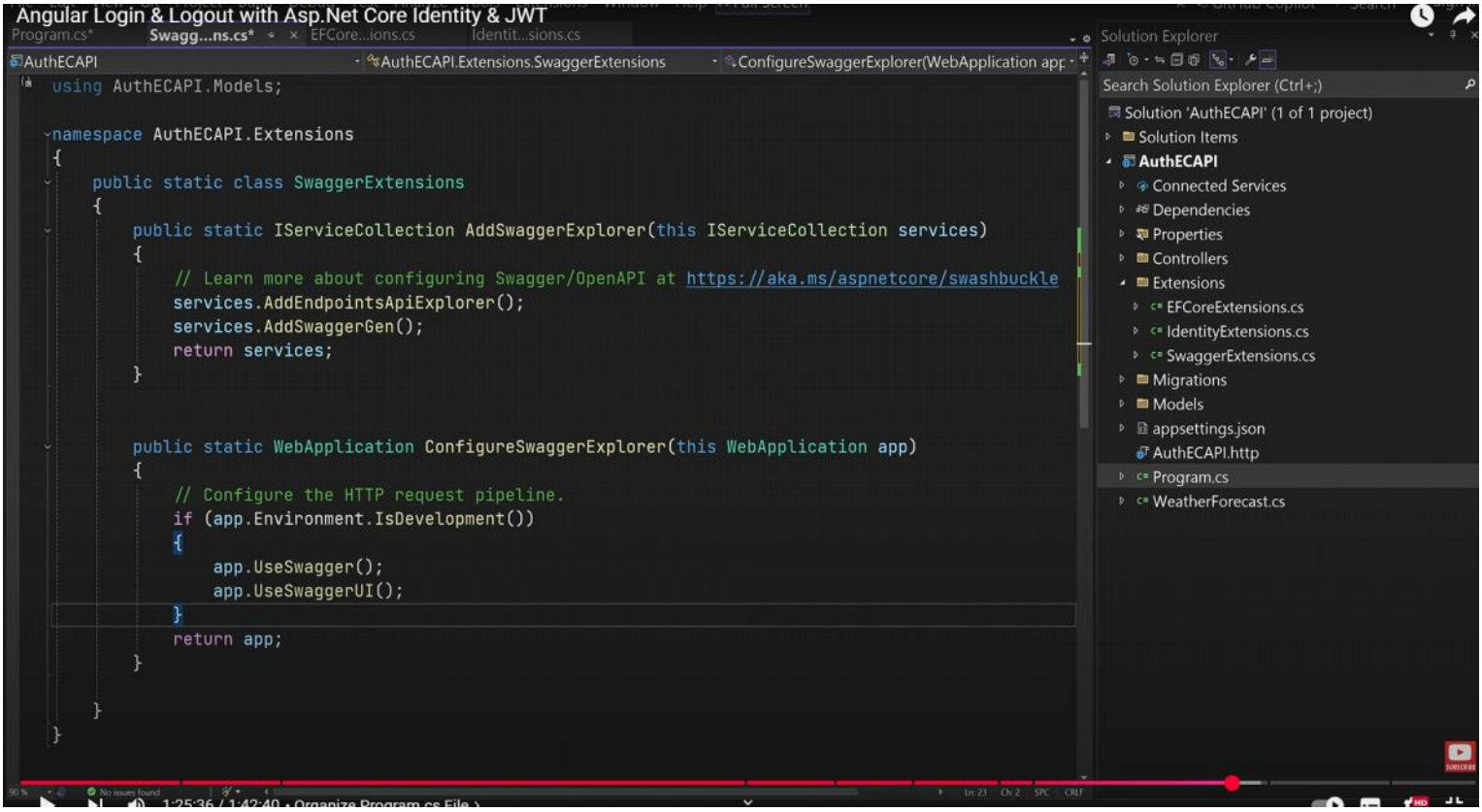

EFCore...ions.cs
Identit...sions.cs
Program.cs*
AuthECAPI
UserRegistrationModel
Email
using System.Text;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions()
.InjectDbContext(builder.Configuration)
.AddIdentityAuth(builder.Configuration);
1:22:52 / 1:42:40 • Organize Program.cs File >

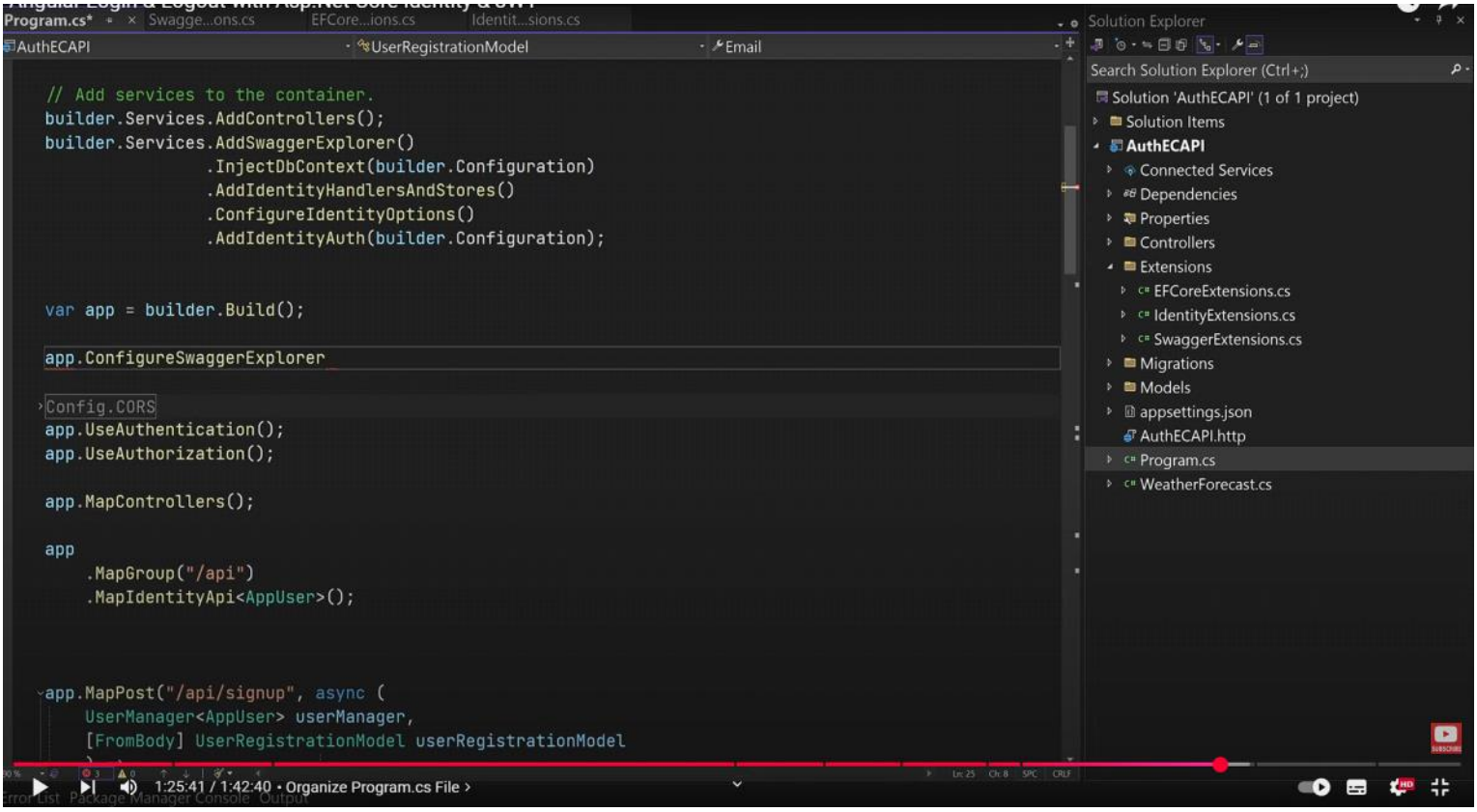

Angular Login & Logout with Asp.Net Core Identity & JWT
Program.cs
EFCore...ions.cs
Identit...sions.cs
AuthECAPI
UserRegistrationModel
Email
// Add services to the container.
builder.Services.AddContro
// Learn more about config
builder.Services.AddEndpoi
builder.Services.AddSwagge
builder.Services.InjectDbC
.AddIdenti
.Configure
.AddIdenti
var app = builder.Build();
// Configure the HTTP requ
if (app.Environment.IsDeve
{
app.UseSwagger();
app.UseSwaggerUI();
}
Add New Item - AuthECAPI
Installed
C#
General
ASP.NET Core
Code
Data
General
Web
CSharp
Online
Sort by: Default
Class C#
Interface C#
Code File C#
Search (Ctrl+E)
Type: C#
An empty class declaration
Name: SwaggerExtensions
Add
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

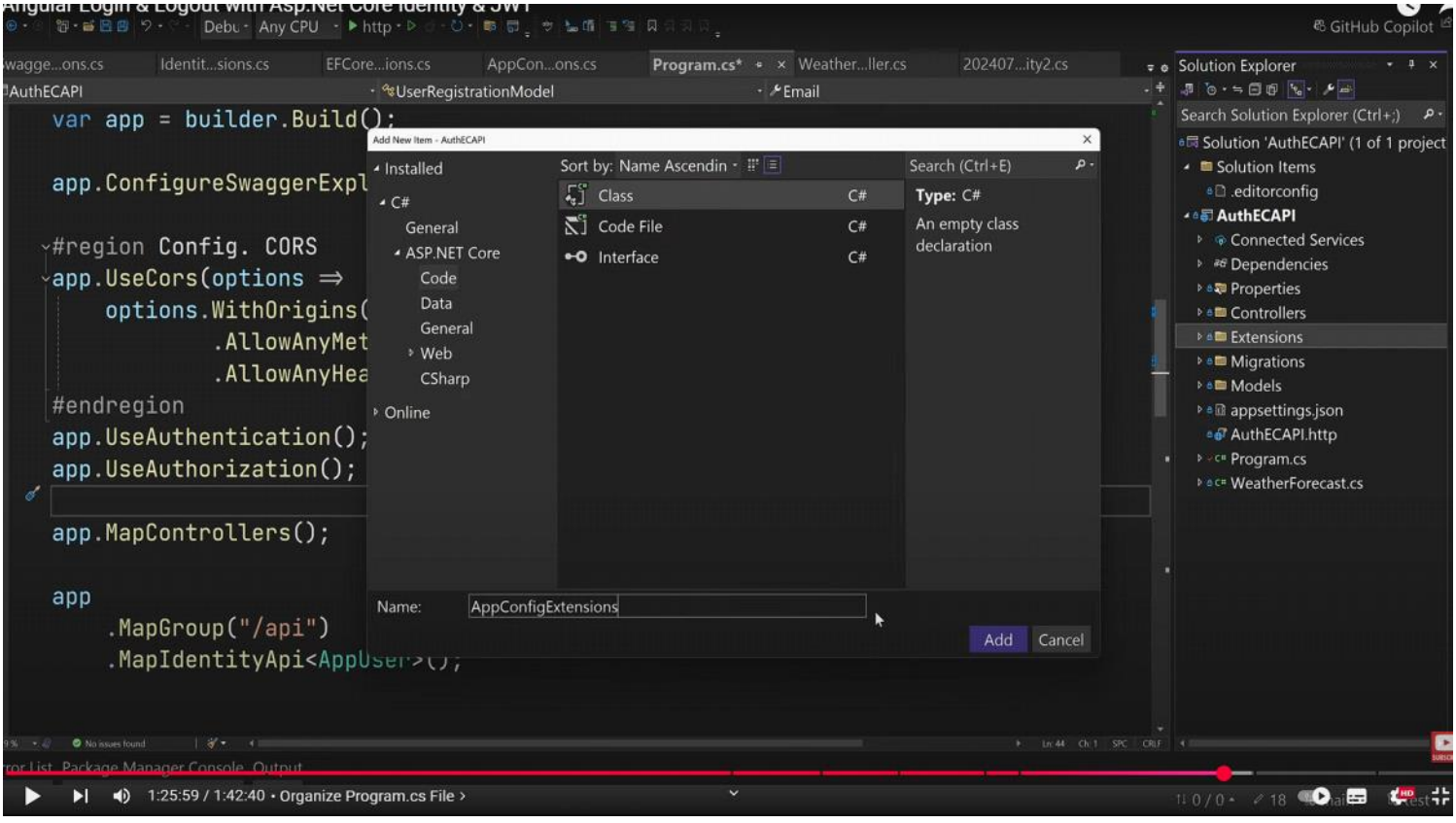

Swagg...ns.cs*
Program.cs
EFCore...ions.cs
Identit...sions.cs
AuthECAPI
AuthECAPI.Extensions.SwaggerExtensions
AddSwaggerExplorer(IServiceCollection services
using AuthECAPI.Models;
namespace AuthECAPI.Extensions
{
public static class SwaggerExtensions
{
public static IServiceCollection AddSwaggerExplorer(this IServiceCollection
{
services.AddIdentityApiEndpoints<AppUser>()
.AddEntityFrameworkStores<AppDbContext>();
return services;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

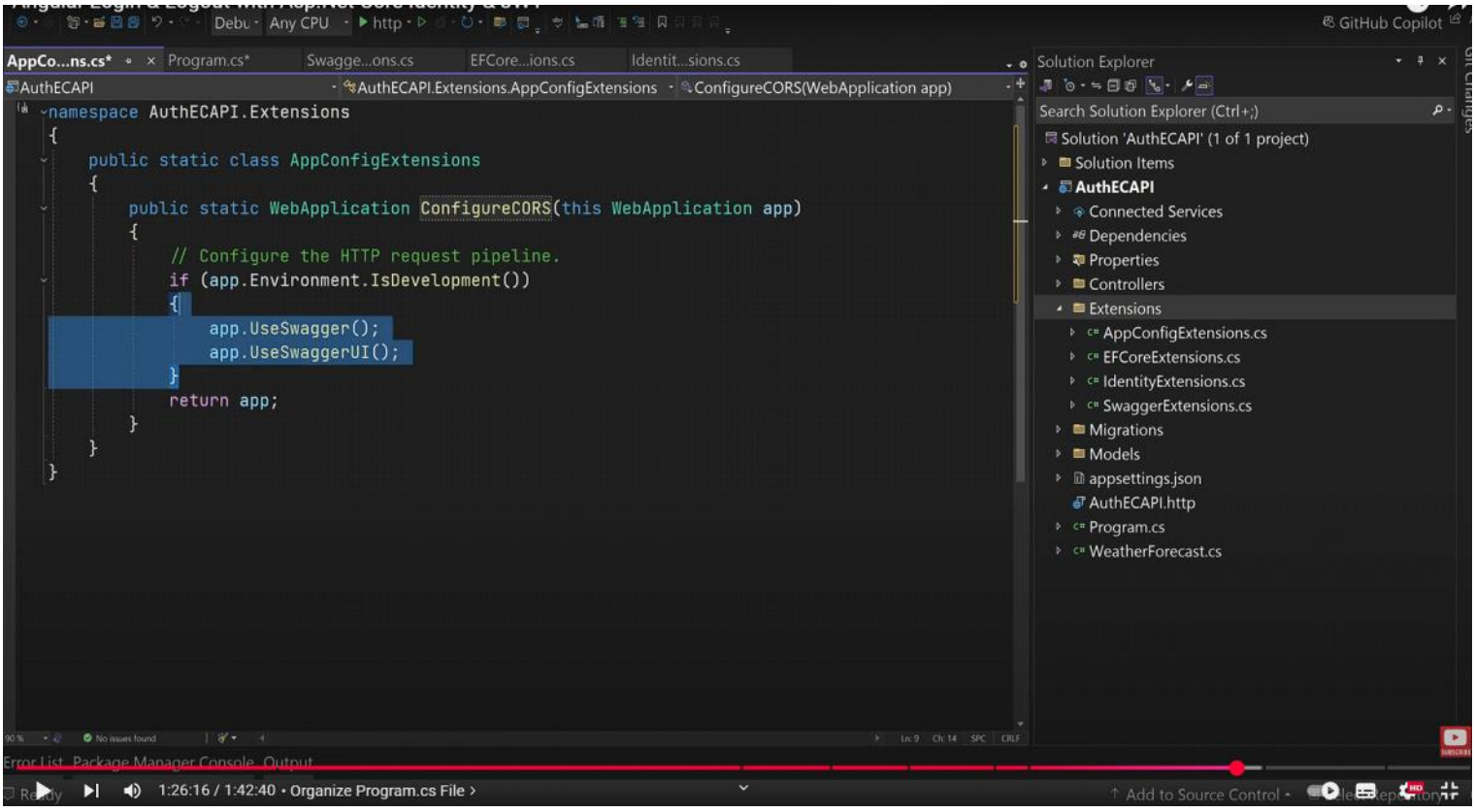

wagge...ons.cs
Program.cs*
EFCore...ions.cs
Identit...sions.cs
AuthECAPI
UserRegistrationModel
Email
using System.Text;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddSwaggerExplorer()
.InjectDbContext(builder.Configuration)
.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions()
.AddIdentityAuth(builder.Configuration);
var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}

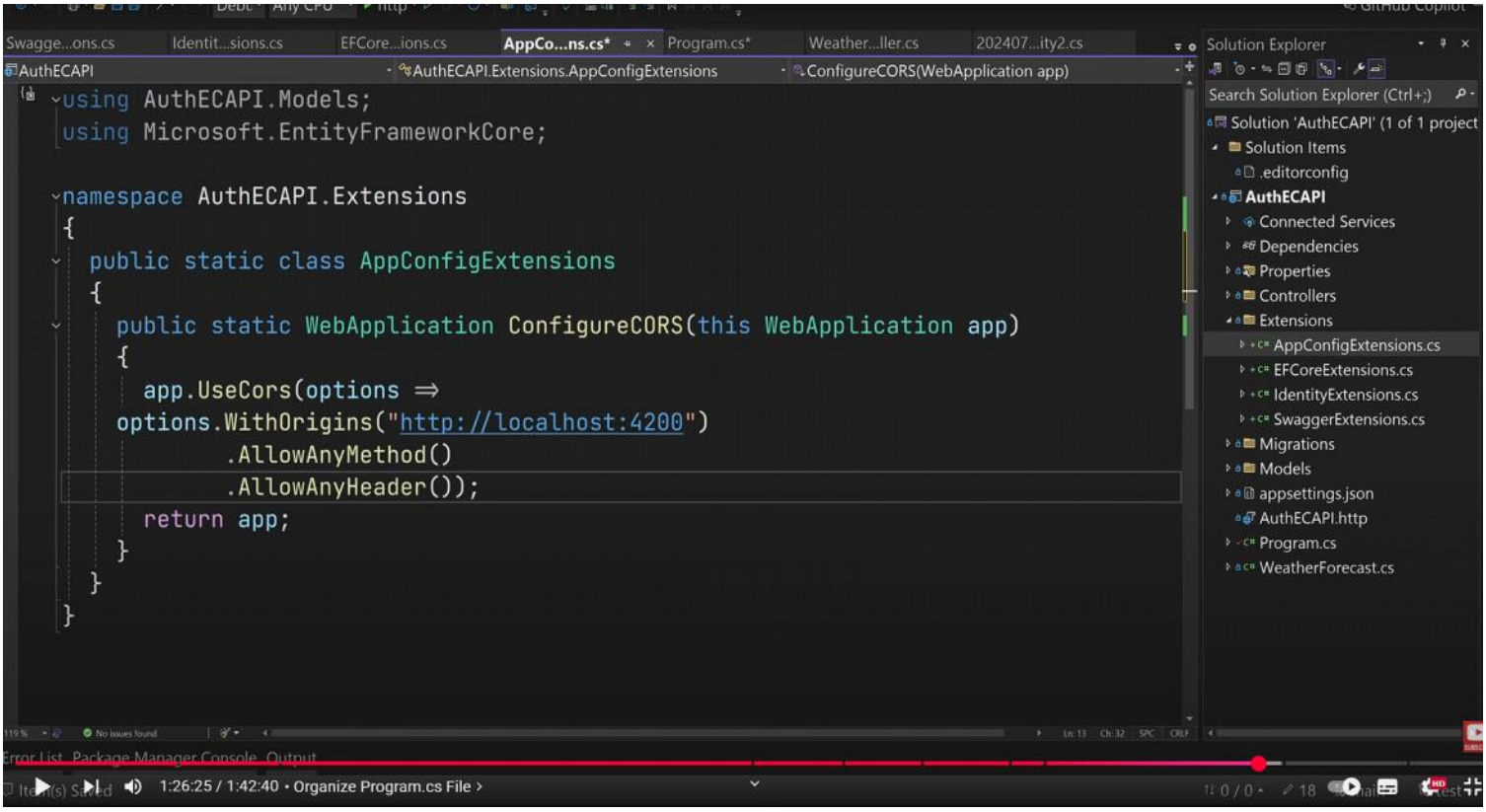

Program.cs
AuthECAPI
UserRegistrationModel
Email
// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddSwaggerExplorer()
.InjectDbContext(builder.Configuration)
.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions()
.AddIdentityAuth(builder.Configuration);
var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}
Config.CORS
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();
app
.MapGroup("/api")
.MapIdentityApi<AppUser>();
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:25:32 / 1:42:40 • Organize Program.cs File >


Angular Login & Logout with Asp.Net Core Identity & JWT
Program.cs
Swagg...ns.cs*
AuthECAPI
AuthECAPI.Extensions.SwaggerExtensions
ConfigureSwaggerExplorer(WebApplication app
using AuthECAPI.Models;
namespace AuthECAPI.Extensions
{
public static class SwaggerExtensions
{
public static IServiceCollection AddSwaggerExplorer(this IServiceCollection services)
{
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
services.AddEndpointsApiExplorer();
services.AddSwaggerGen();
return services;
}
public static WebApplication ConfigureSwaggerExplorer(this WebApplication app)
{
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}
return app;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

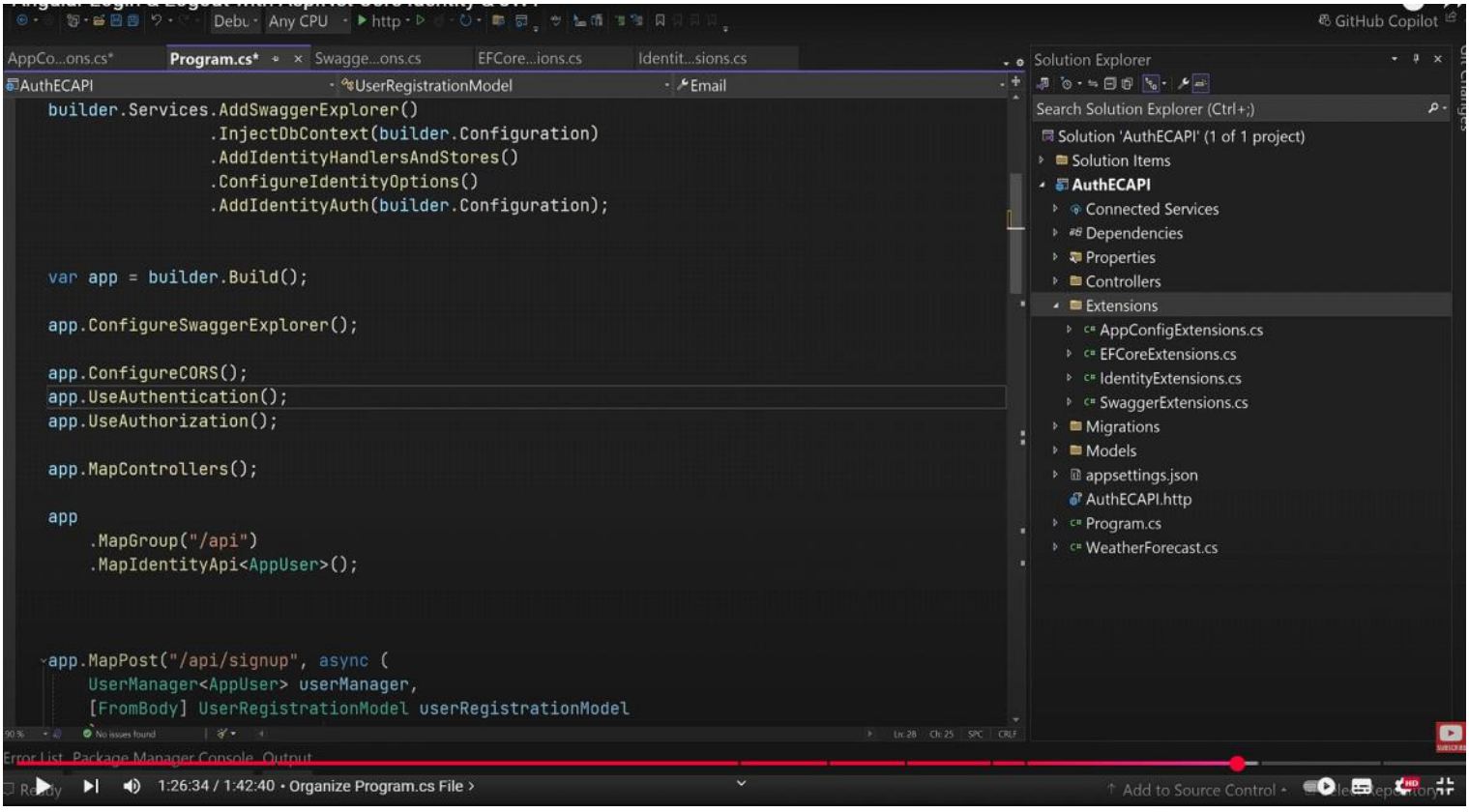

Program.cs*
AuthECAPI
UserRegistrationModel
Email
// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddSwaggerExplorer()
.InjectDbContext(builder.Configuration)
.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions()
.AddIdentityAuth(builder.Configuration);
var app = builder.Build();
app.ConfigureSwaggerExplorer
Config.CORS
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();
app
.MapGroup("/api")
.MapIdentityApi<AppUser>();
app.MapPost("/api/signup", async (
UserManager<AppUser> userManager,
[FromBody] UserRegistrationModel userRegistrationModel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:25:41 / 1:42:40 • Organize Program.cs File >

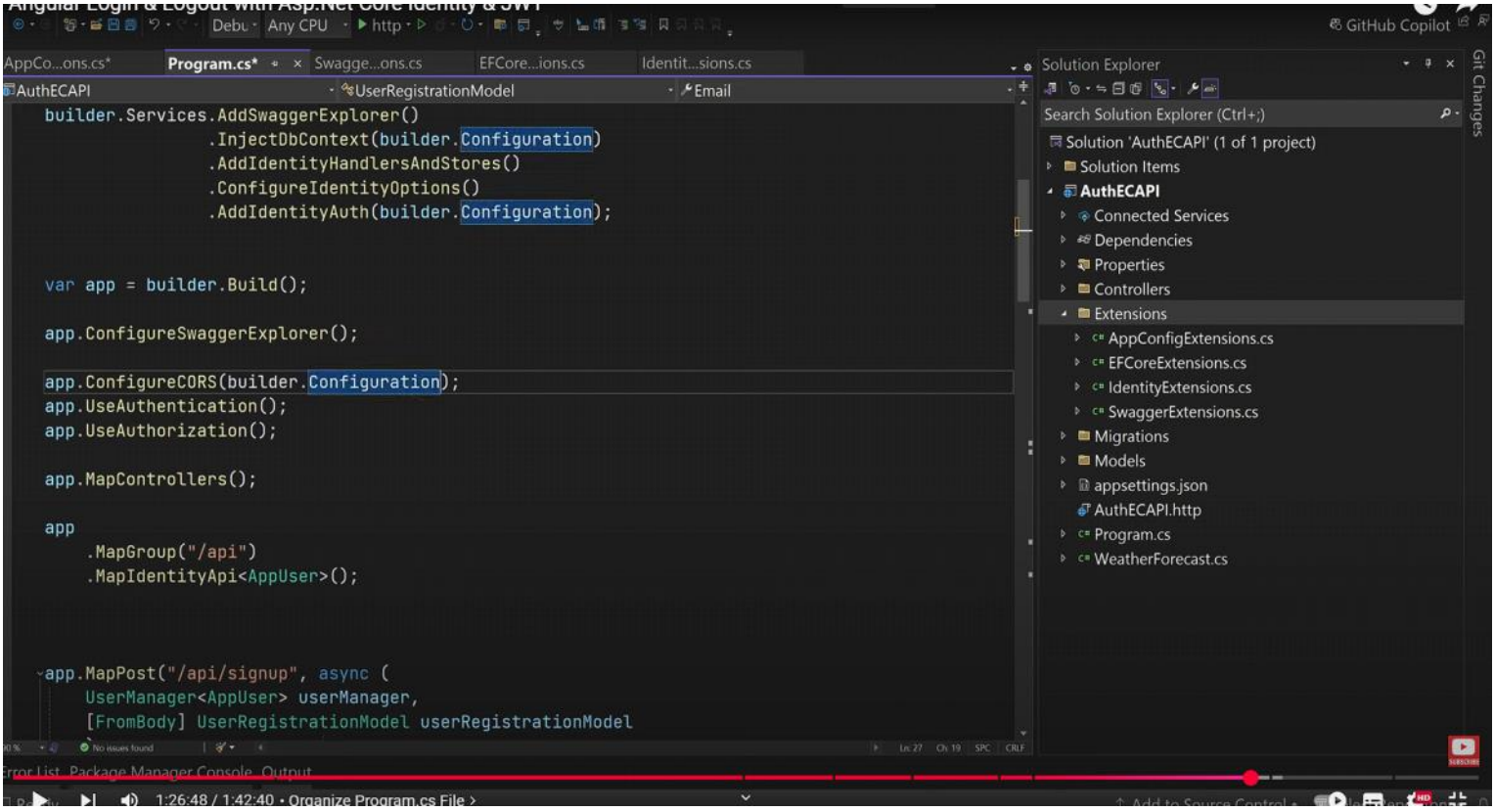

GitHub Copilot
Program.cs*
AuthECAPI
UserRegistrationModel
Email
var app = builder.Build();
app.ConfigureSwaggerExpl
#region Config. CORS
app.UseCors(options ⇒
options.WithOrigins(
.AllowAnyMet
.AllowAnyHea
#endregion
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();
app
.MapGroup("/api")
.MapIdentityApi<AppUser>();
Add New Item - AuthECAPI
Installed
C#
General
ASP.NET Core
Code
Data
General
Web
CSharp
Online
Sort by: Name Ascendin
Class C#
Code File C#
Interface C#
Search (Ctrl+E)
Type: C#
An empty class declaration
Name: AppConfigExtensions
Add
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project
Solution Items
.editorconfig
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:25:59 / 1:42:40 • Organize Program.cs File >

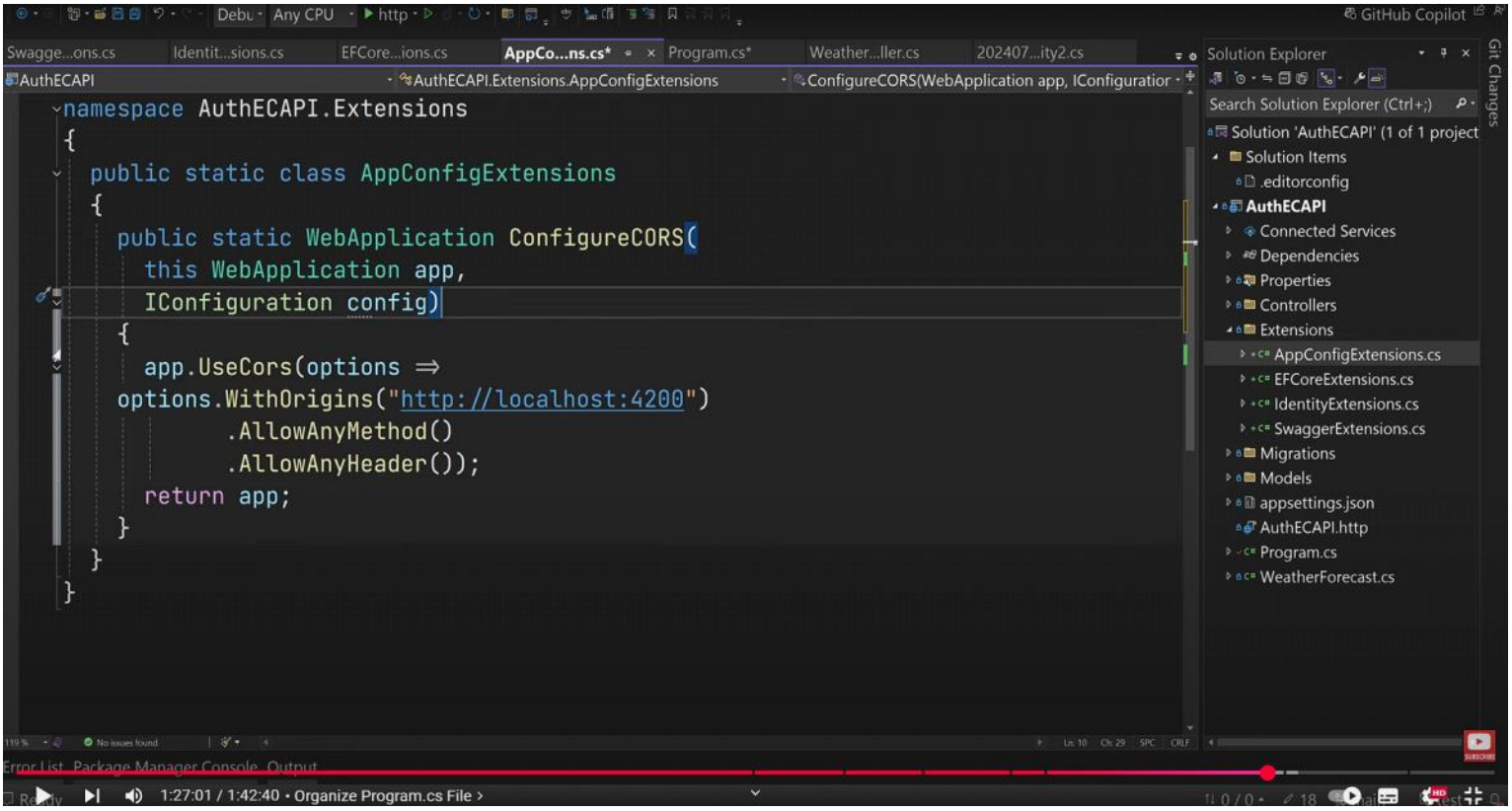

AppCo...ns.cs*
Program.cs*
Swagge...ons.cs
EFCore...ions.cs
Identit...sions.cs
AuthECAPI
AuthECAPI.Extensions.AppConfigExtensions
ConfigureCORS(WebApplication app)
namespace AuthECAPI.Extensions
{
public static class AppConfigExtensions
{
public static WebApplication ConfigureCORS(this WebApplication app)
{
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}
return app;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:26:16 / 1:42:40 • Organize Program.cs File

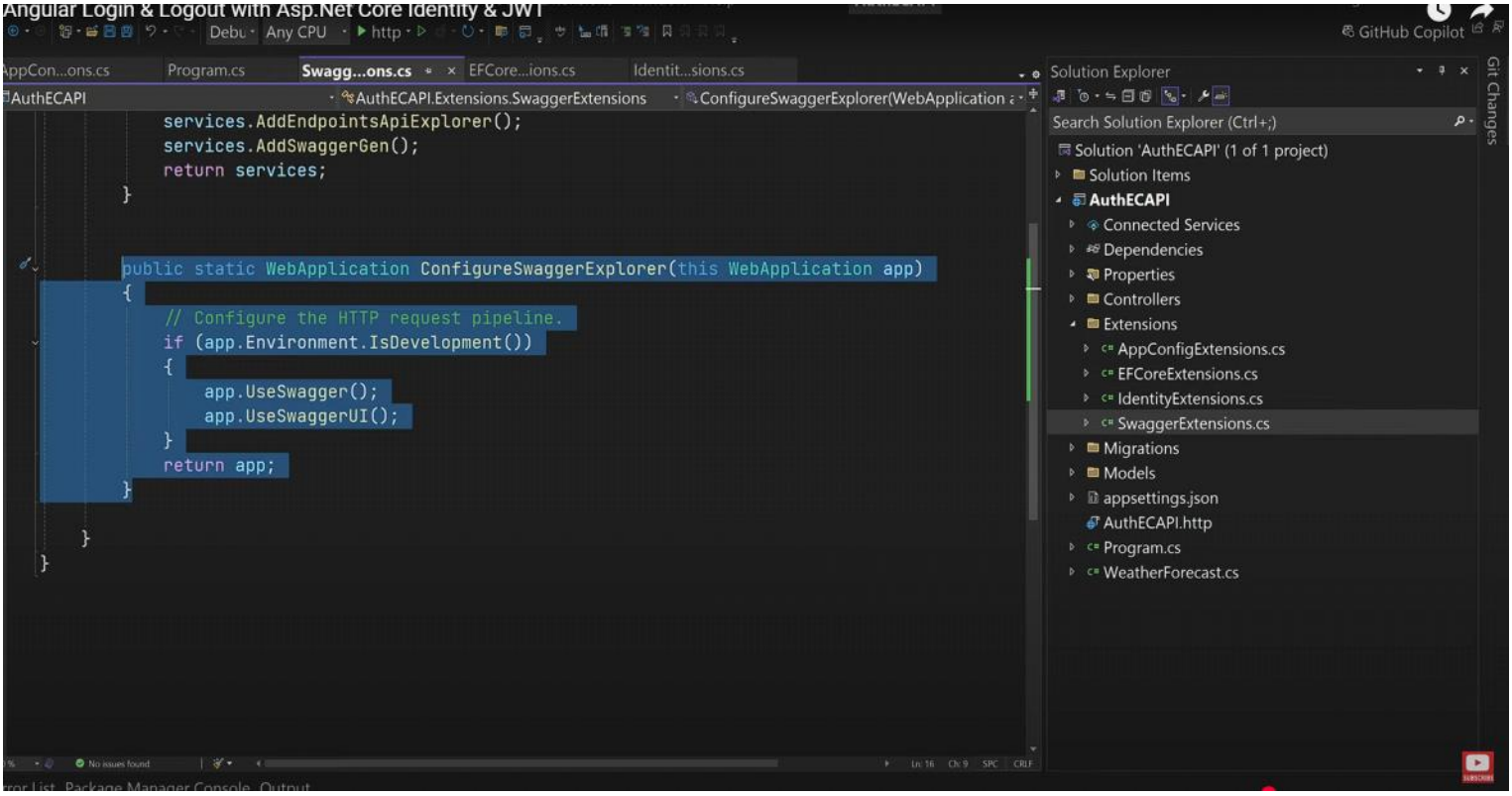

Swagge...ons.cs
Identit...sions.cs
EFCore...ions.cs
AppCo...ns.cs*
Program.cs*
Weather...ller.cs
202407...ity2.cs
AuthECAPI
AuthECAPI.Extensions.AppConfigExtensions
ConfigureCORS(WebApplication app)
using AuthECAPI.Models;
using Microsoft.EntityFrameworkCore;
namespace AuthECAPI.Extensions
{
public static class AppConfigExtensions
{
public static WebApplication ConfigureCORS(this WebApplication app)
{
app.UseCors(options ⇒
options.WithOrigins("http://localhost:4200")
.AllowAnyMethod()
.AllowAnyHeader());
return app;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project
Solution Items
.editorconfig
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:26:25 / 1:42:40 • Organize Program.cs File

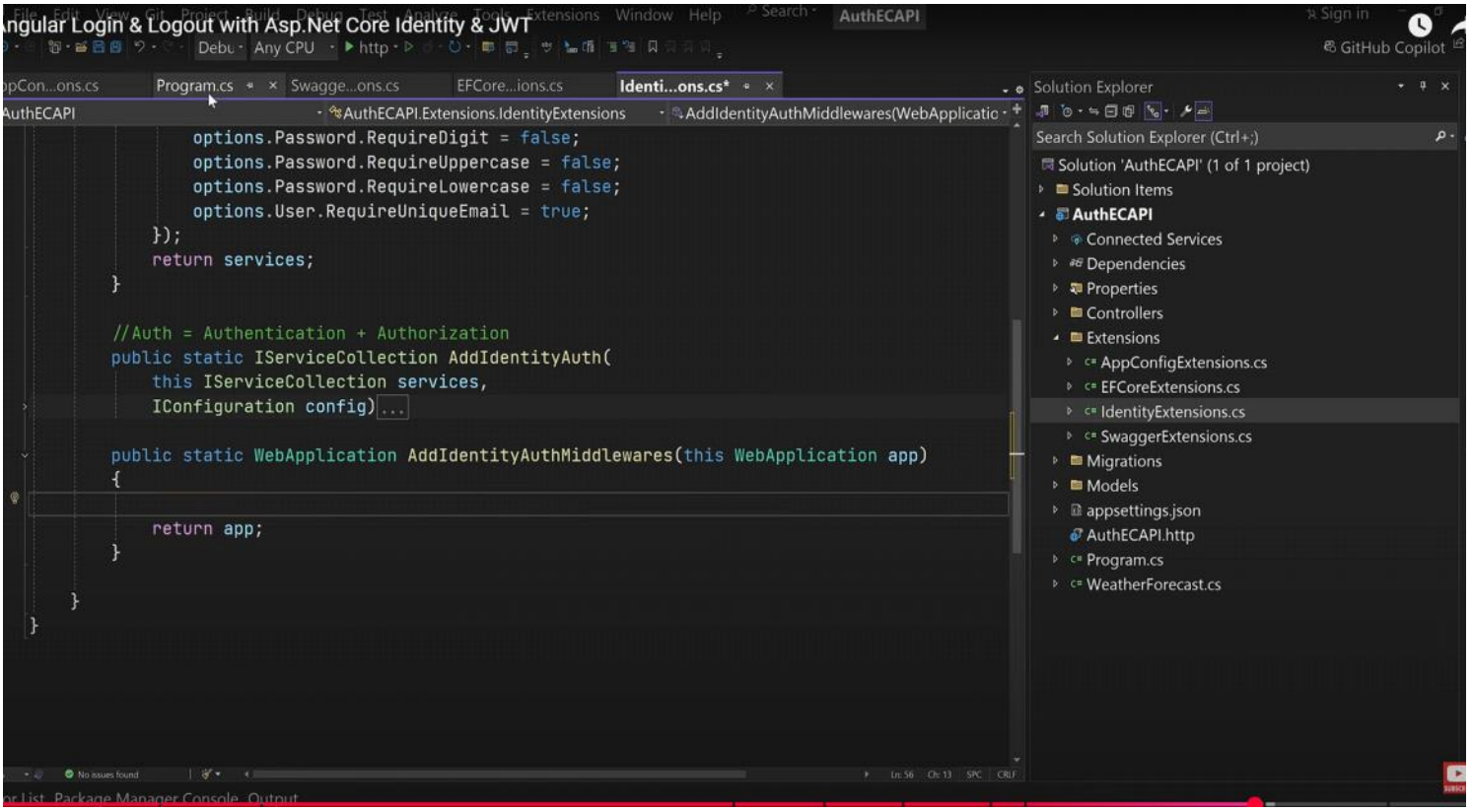
configureCORS

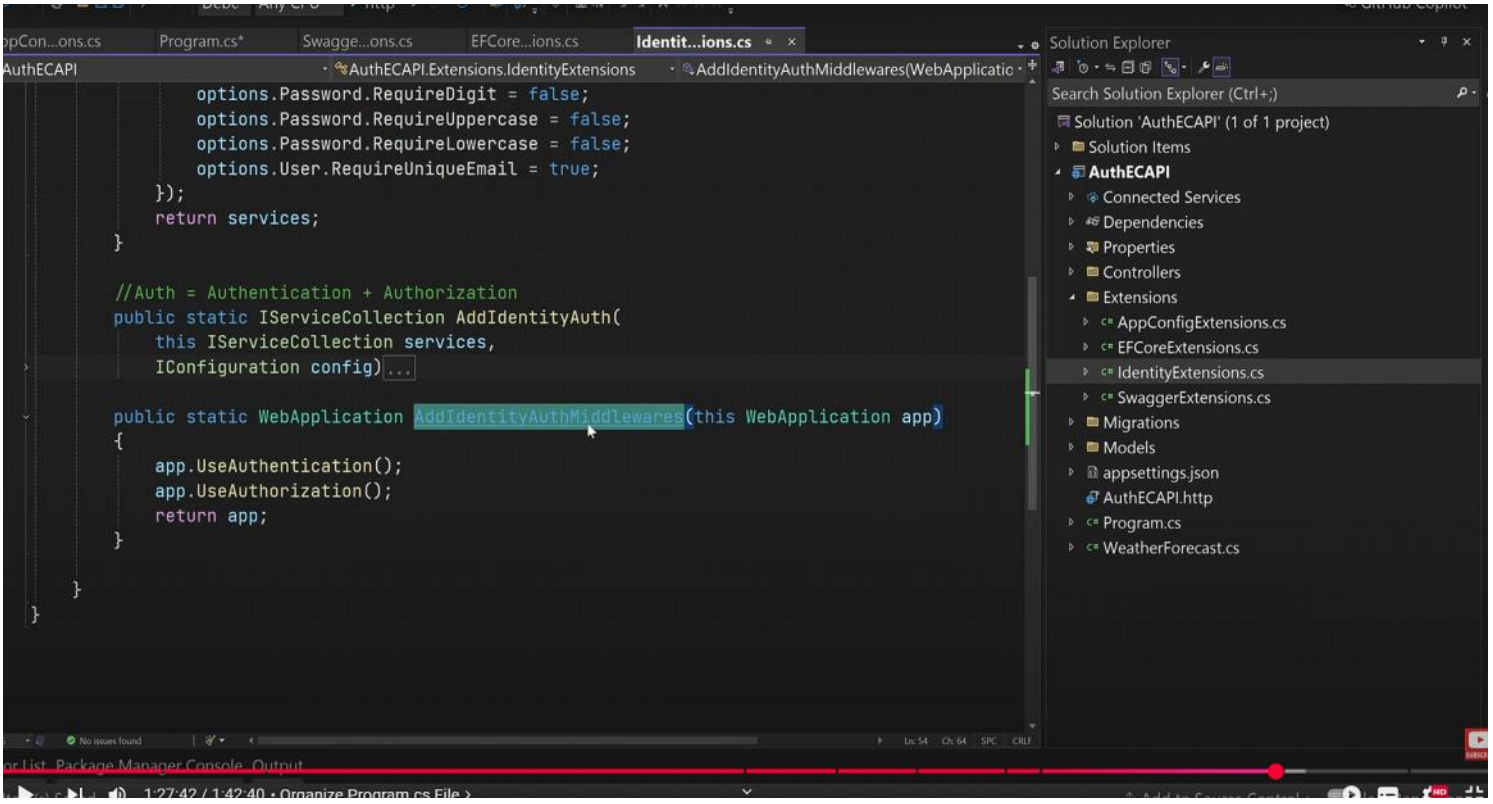


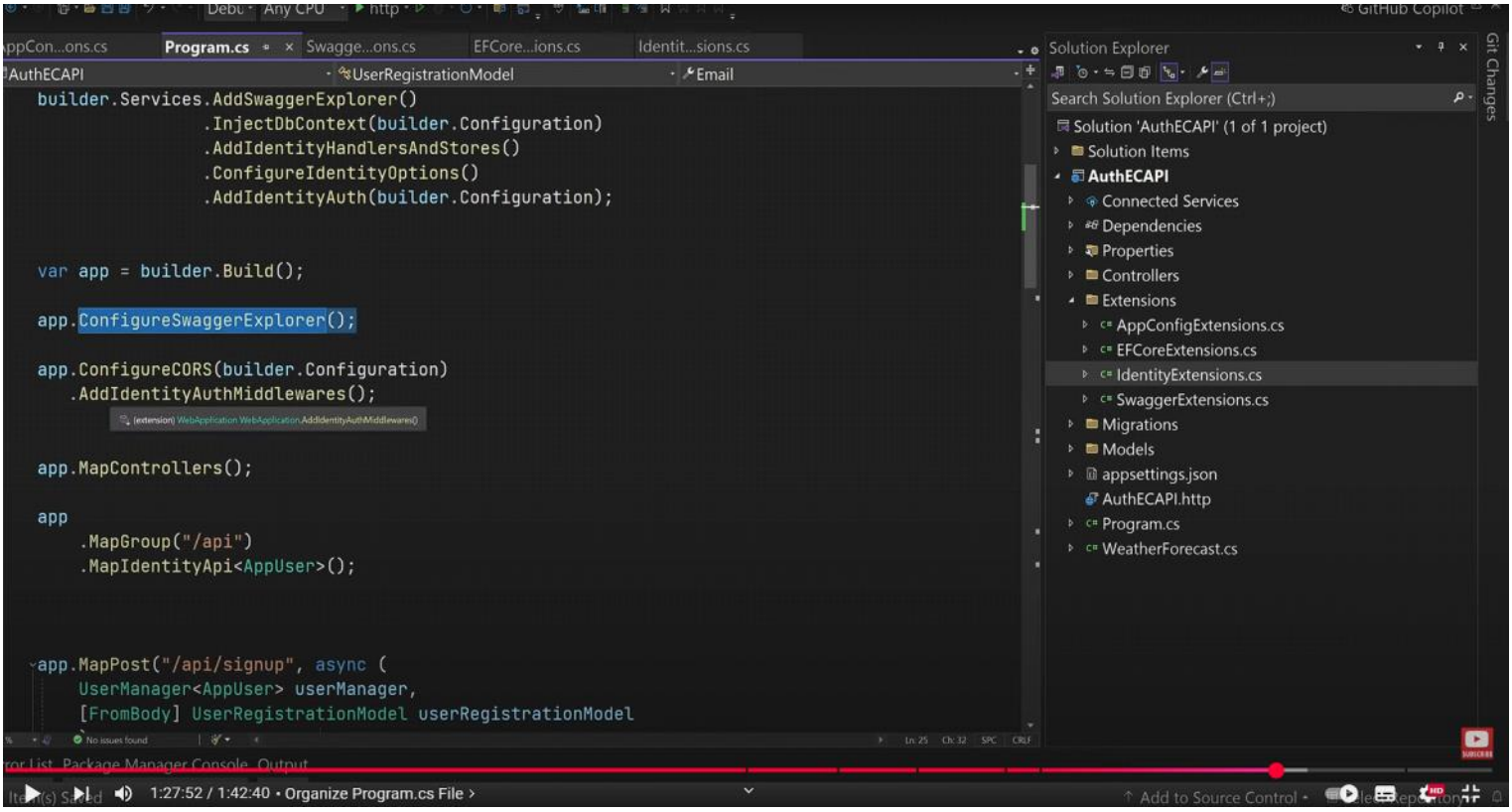


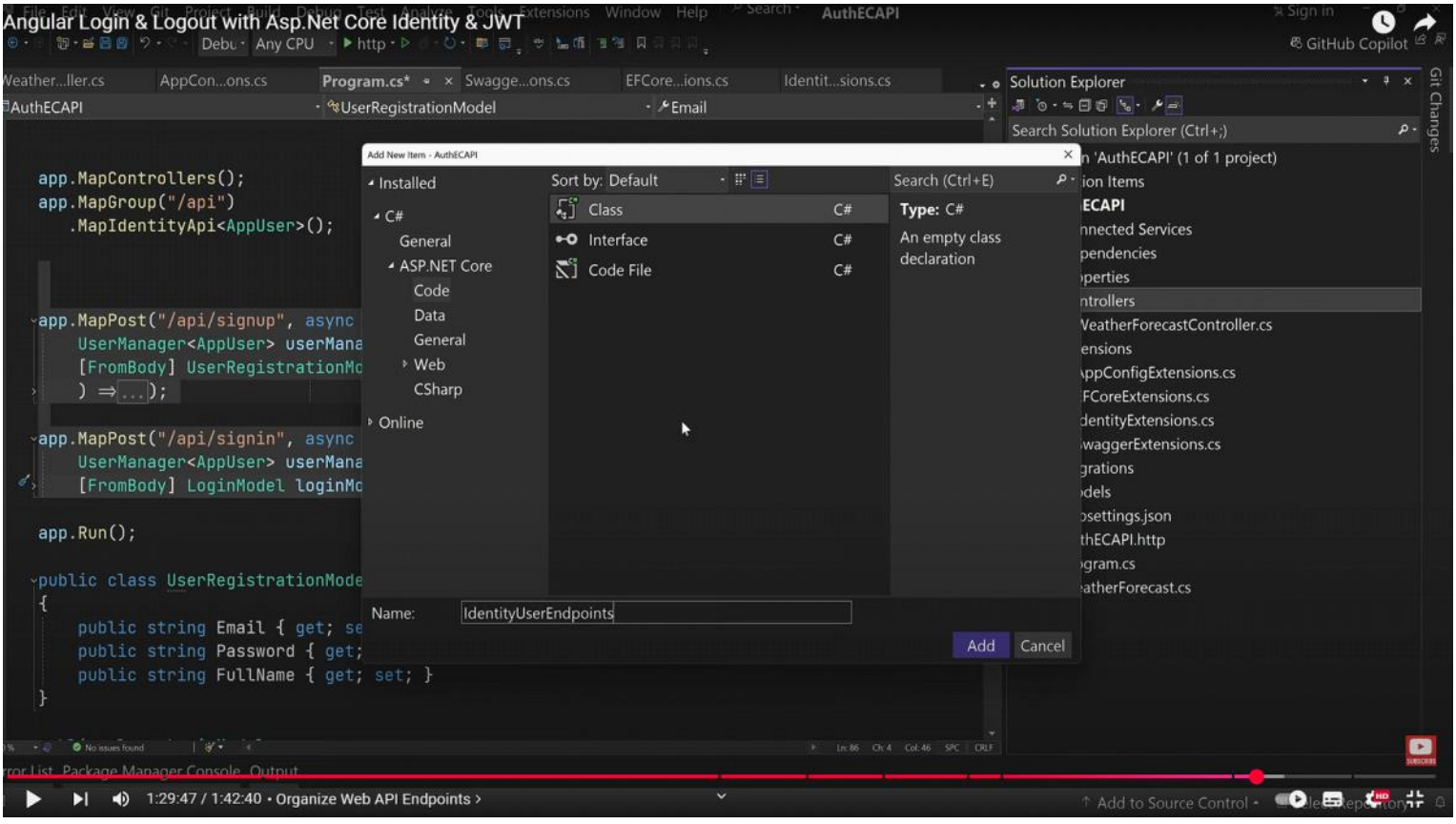

namespace AuthECAPI.Extensions
{
public static class AppConfigExtensions
{
public static WebApplication ConfigureCORS(
this WebApplication app,
IConfiguration config)
{
app.UseCors(options ⇒
options.WithOrigins("http://localhost:4200")
.AllowAnyMethod()
.AllowAnyHeader());
return app;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project
Solution Items
.editorconfig
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:27:01 / 1:42:40 • Organize Program.cs File ›


Angular Login & Logout with Asp.Net Core Identity & JWT
services.AddEndpointsApiExplorer();
services.AddSwaggerGen();
return services;
}
public static WebApplication ConfigureSwaggerExplorer(this WebApplication app)
{
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}
return app;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

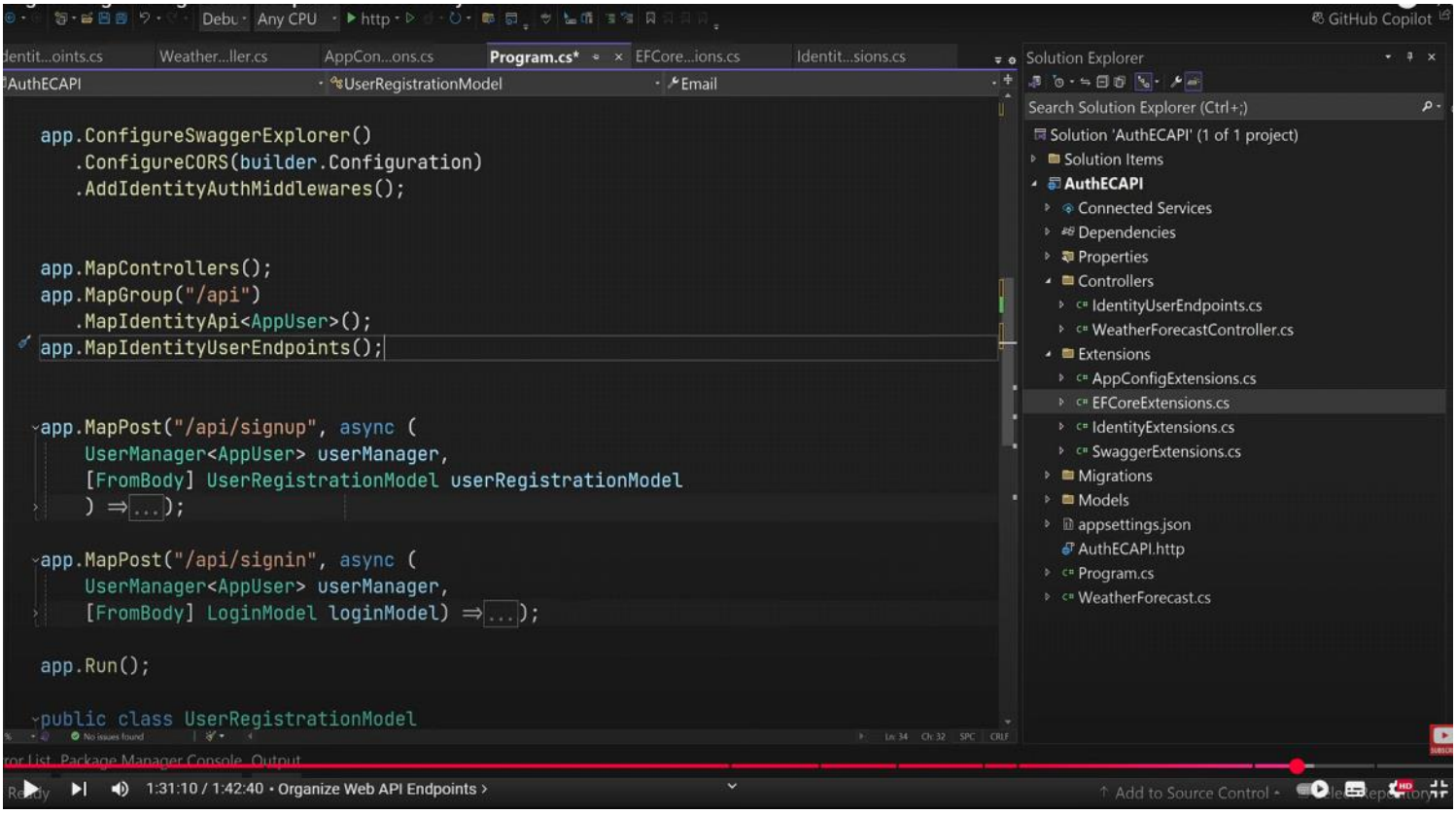

Angular Login & Logout with Asp.Net Core Identity & JWT
Program.cs
Swagge...ons.cs
EFCore...ions.cs
Identi...ons.cs*
AuthECAPI.Extensions.IdentityExtensions
AddIdentityAuthMiddlewares(WebApplicatio
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
return services;
}
//Auth = Authentication + Authorization
public static IServiceCollection AddIdentityAuth(
this IServiceCollection services,
IConfiguration config)...
public static WebApplication AddIdentityAuthMiddlewares(this WebApplication app)
{
return app;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

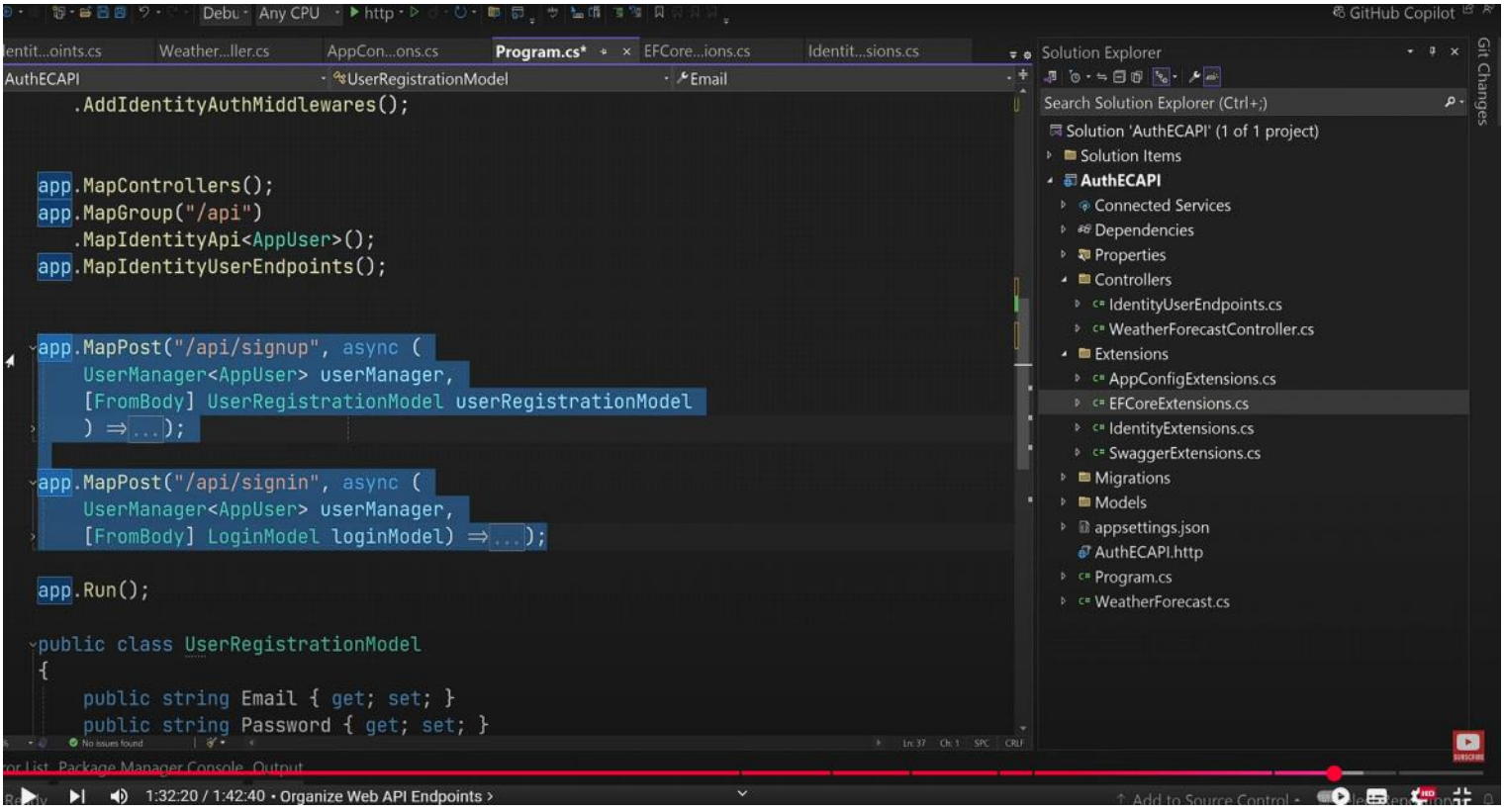

Program.cs*
Identit...ions.cs
AuthECAPI.Extensions.IdentityExtensions
AddIdentityAuthMiddlewares(WebApplicatio
options.Password.RequireDigit = false;
options.Password.RequireUppercase = false;
options.Password.RequireLowercase = false;
options.User.RequireUniqueEmail = true;
});
return services;
}
//Auth = Authentication + Authorization
public static IServiceCollection AddIdentityAuth(
this IServiceCollection services,
IConfiguration config)...
public static WebApplication AddIdentityAuthMiddlewares(this WebApplication app)
{
app.UseAuthentication();
app.UseAuthorization();
return app;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

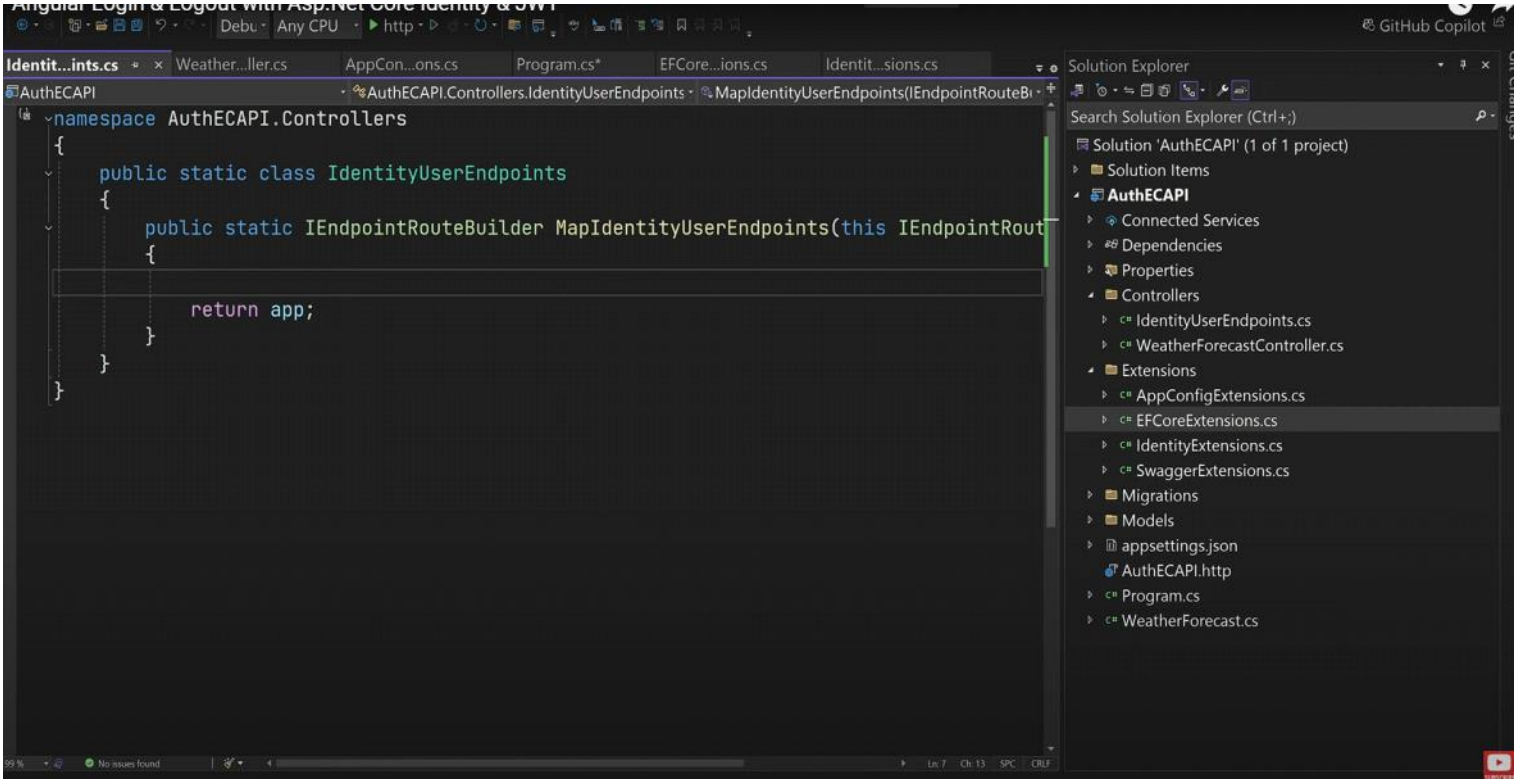


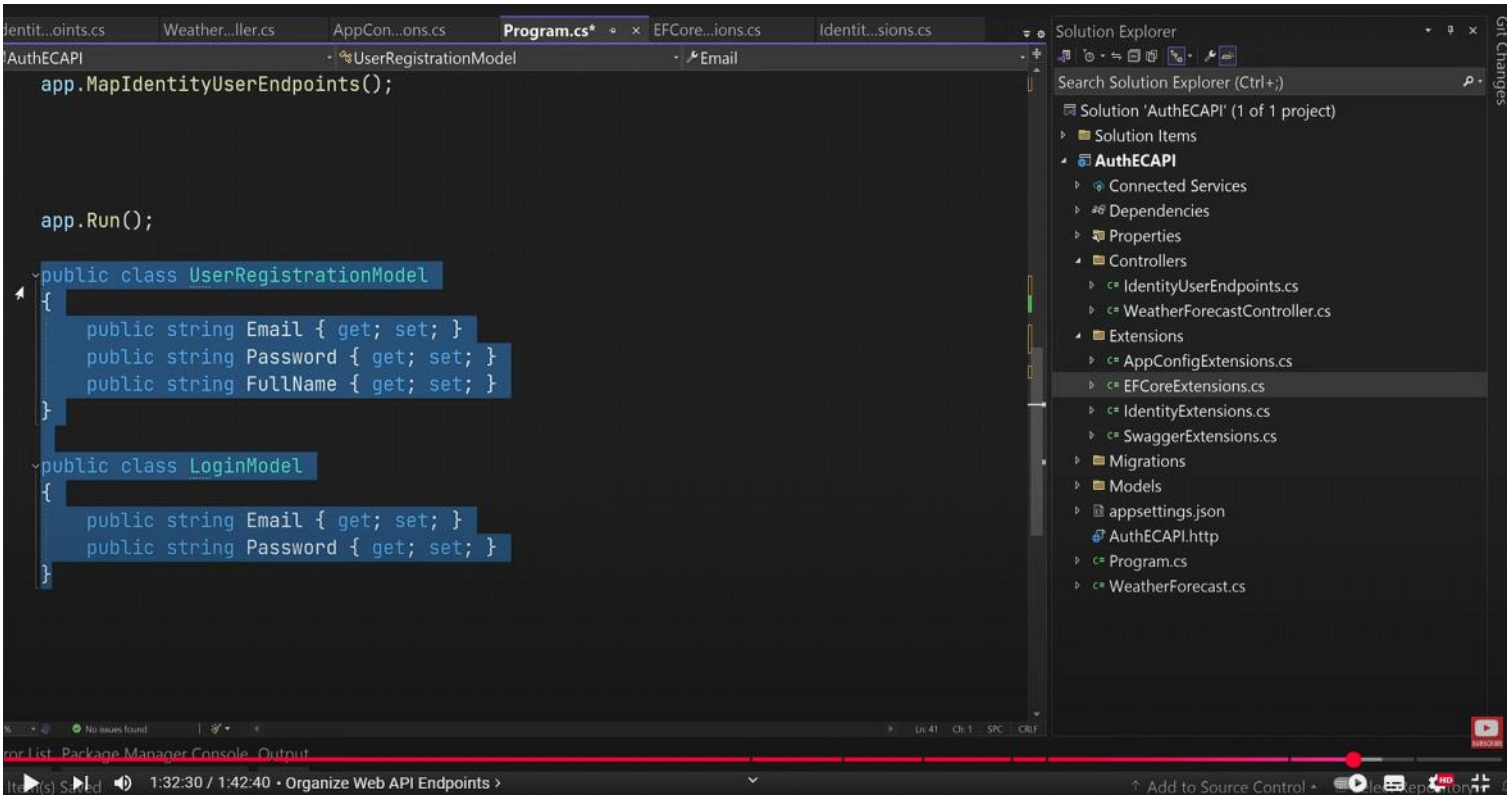


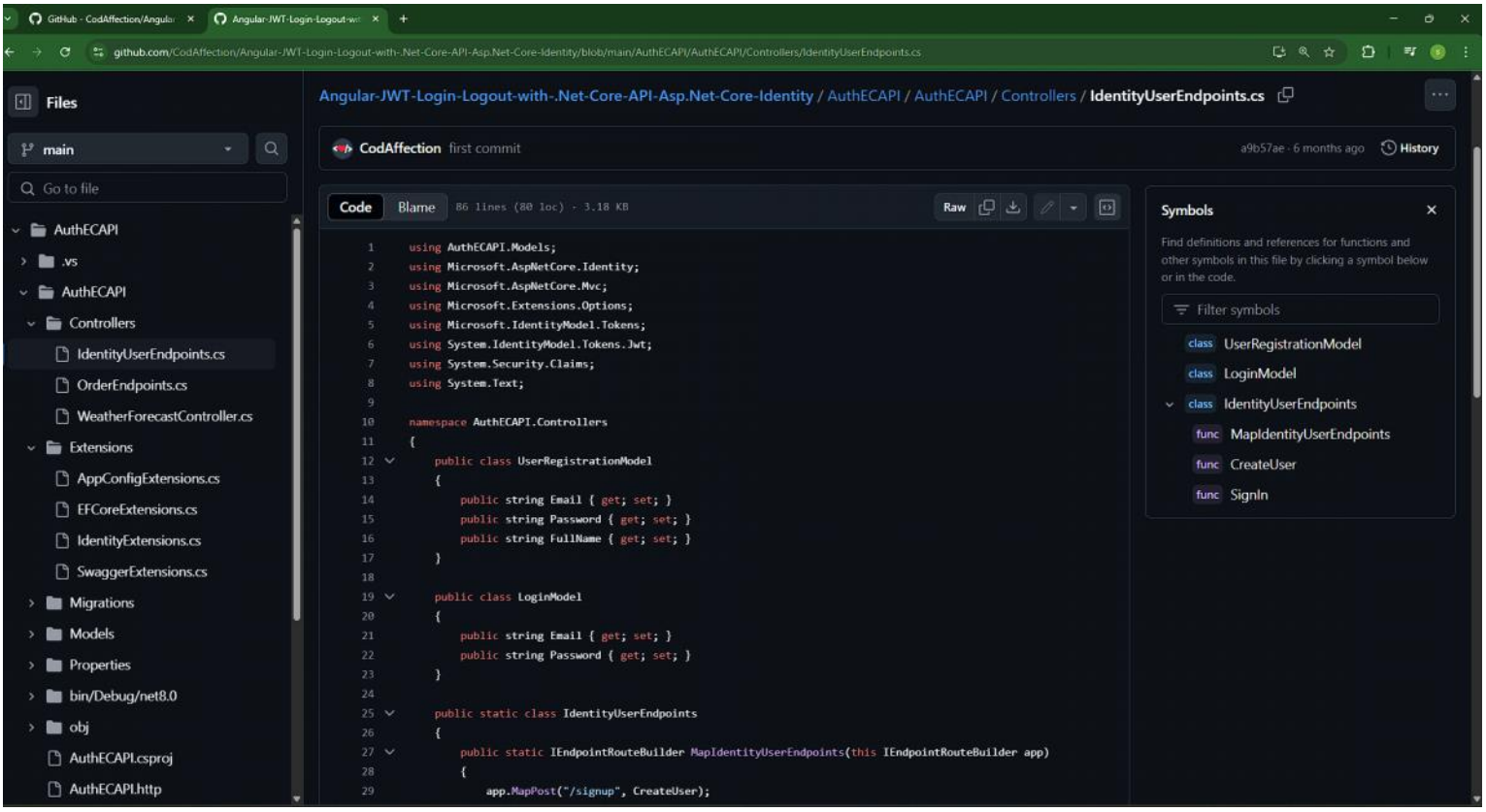
Mapidentityuserendpoing

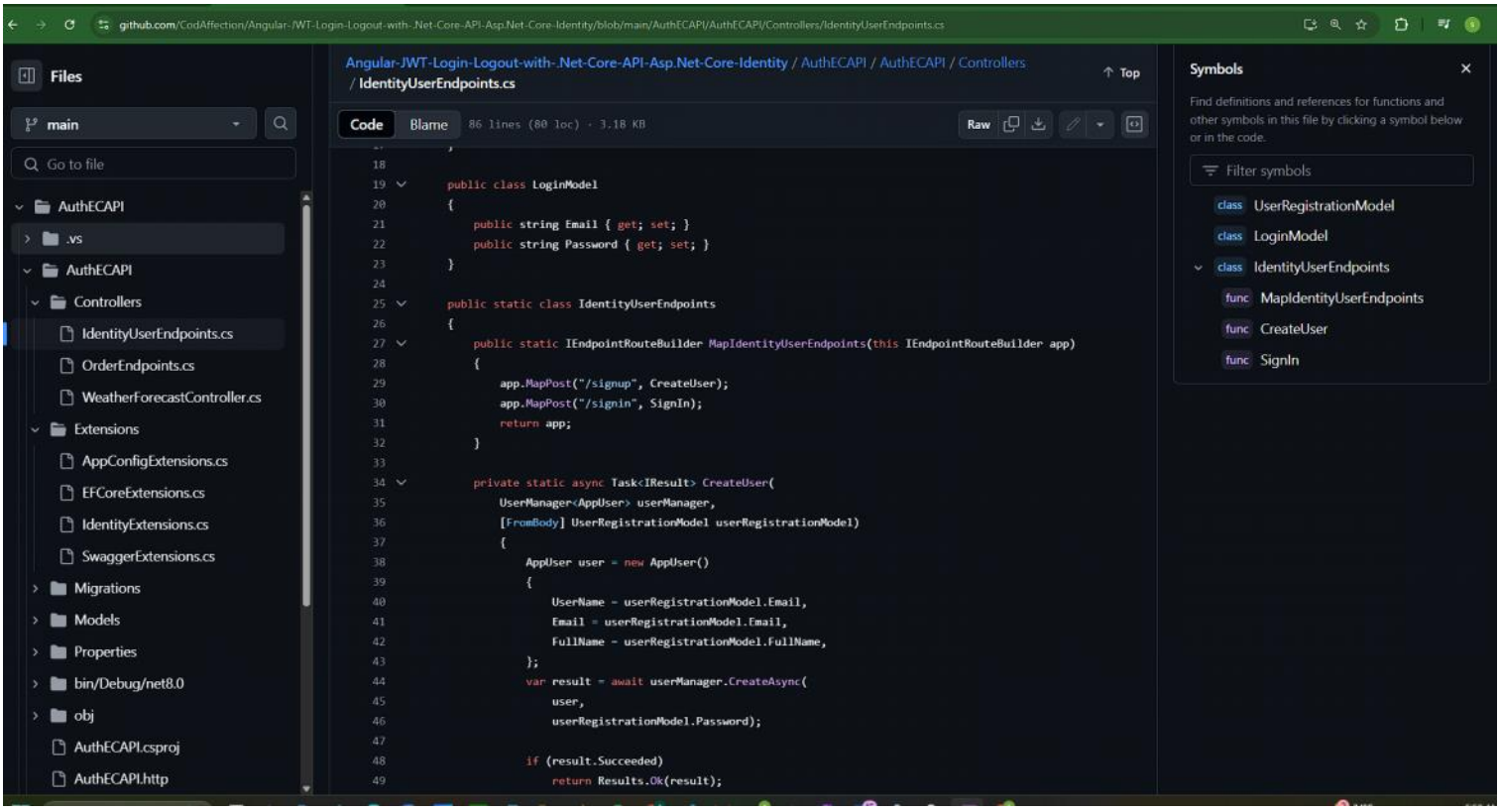

Program.cs*
AuthECAPI
UserRegistrationModel
Email
app.ConfigureSwaggerExplorer()
.ConfigureCORS(builder.Configuration)
.AddIdentityAuthMiddlewares();
app.MapControllers();
app.MapGroup("/api")
.MapIdentityApi<AppUser>();
app.MapIdentityUserEndpoints();
app.MapPost("/api/signup", async (
UserManager<AppUser> userManager,
[FromBody] UserRegistrationModel userRegistrationModel
) =>...);
app.MapPost("/api/signin", async (
UserManager<AppUser> userManager,
[FromBody] LoginModel loginModel) =>...);
app.Run();
public class UserRegistrationModel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
IdentityUserEndpoints.cs
WeatherForecastController.cs
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:31:10 / 1:42:40 • Organize Web API Endpoints

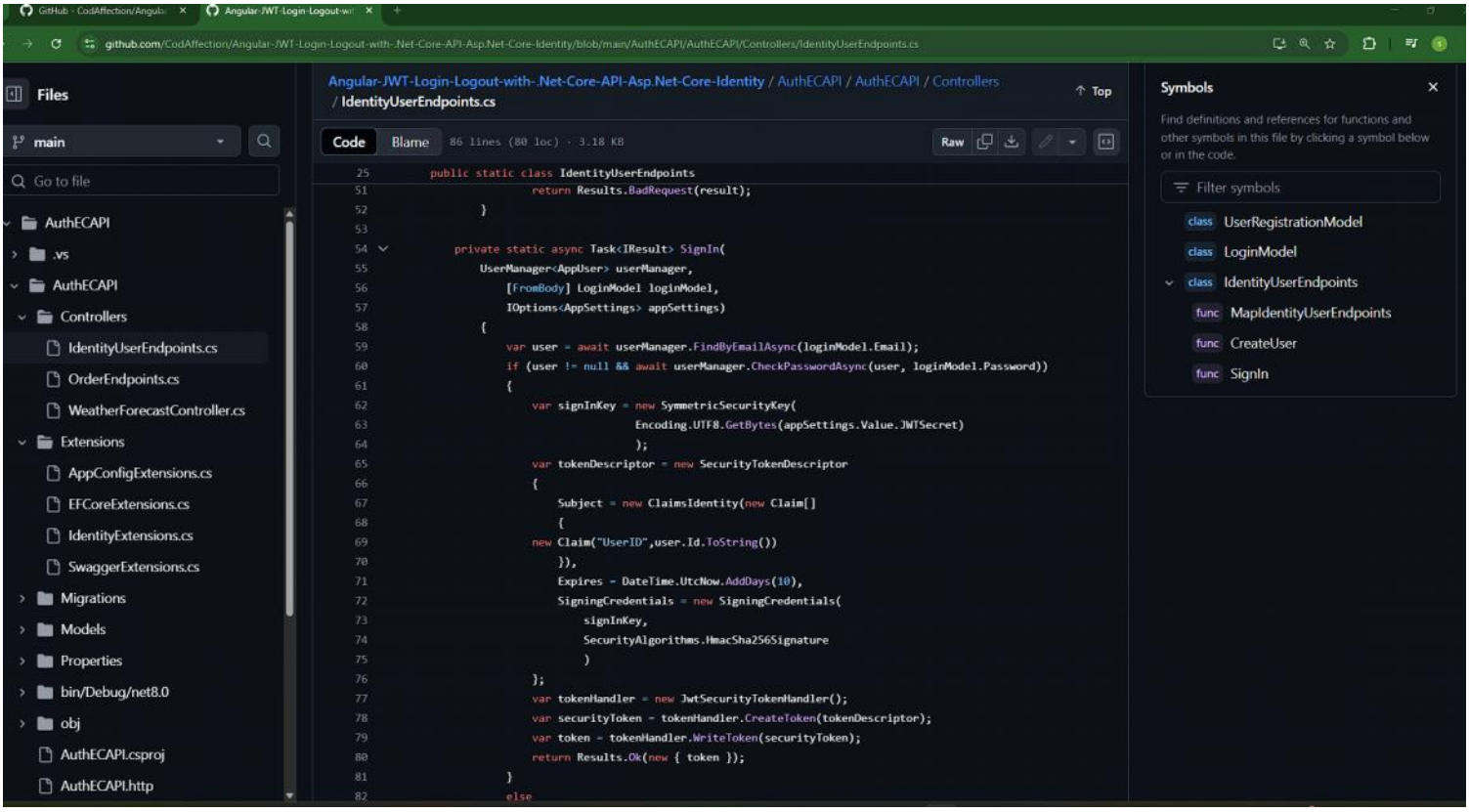

Program.cs*
AuthECAPI
UserRegistrationModel
Email
.AddIdentityAuthMiddlewares();
app.MapControllers();
app.MapGroup("/api")
.MapIdentityApi<AppUser>();
app.MapIdentityUserEndpoints();
app.MapPost("/api/signup", async (
UserManager<AppUser> userManager,
[FromBody] UserRegistrationModel userRegistrationModel
) =>...);
app.MapPost("/api/signin", async (
UserManager<AppUser> userManager,
[FromBody] LoginModel loginModel) =>...);
app.Run();
public class UserRegistrationModel
{
public string Email { get; set; }
public string Password { get; set; }
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
IdentityUserEndpoints.cs
WeatherForecastController.cs
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:32:20 / 1:42:40 • Organize Web API Endpoints


Identit...ints.cs
Weather...ller.cs
AppCon...ons.cs
Program.cs*
EFCore...ions.cs
Identit...sions.cs
AuthECAPI
AuthECAPI.Controllers.IdentityUserEndpoints
MapIdentityUserEndpoints(IEndpointRouteBu
namespace AuthECAPI.Controllers
{
public static class IdentityUserEndpoints
{
public static IEndpointRouteBuilder MapIdentityUserEndpoints(this IEndpointRout
{
return app;
}
}
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
IdentityUserEndpoints.cs
WeatherForecastController.cs
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs

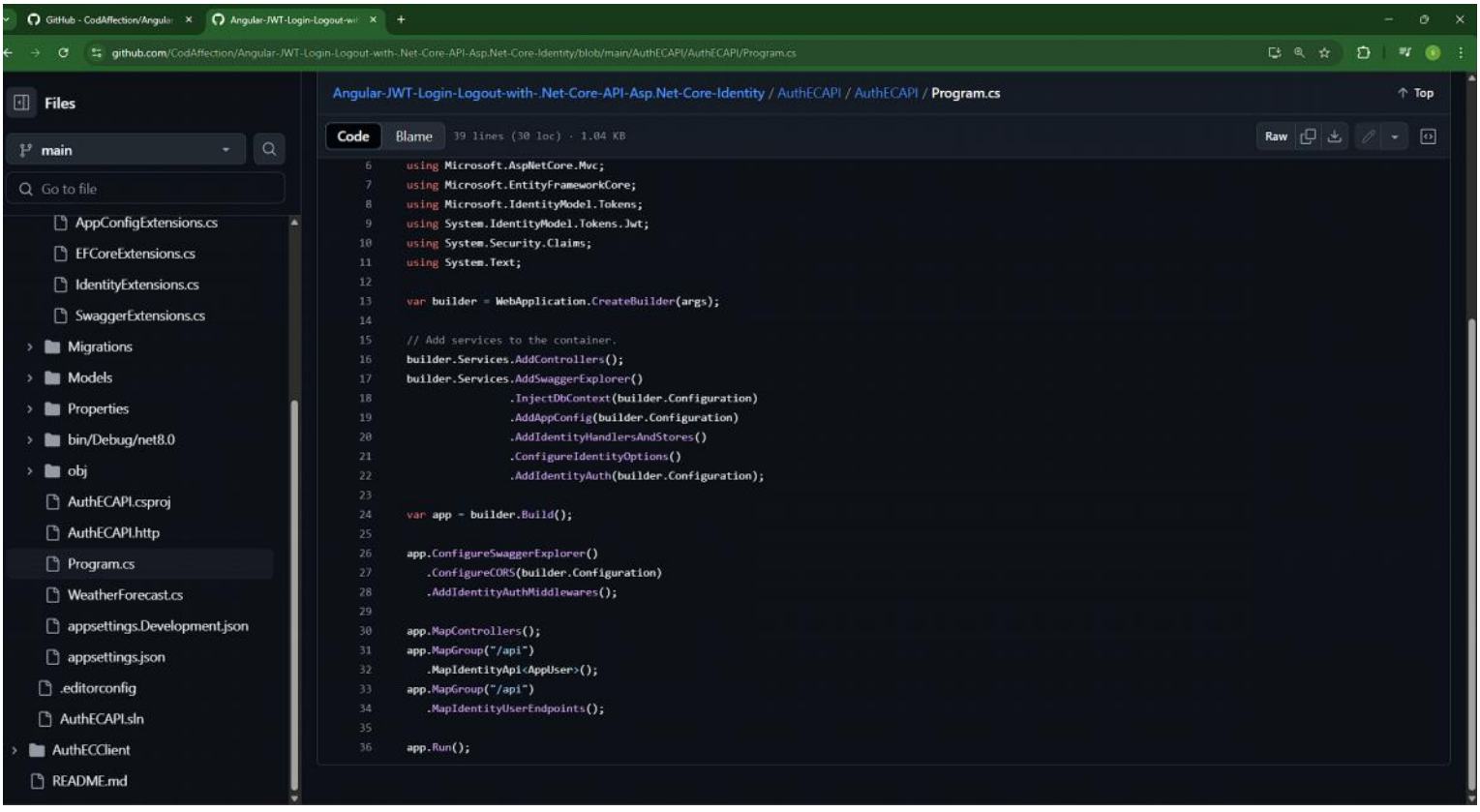

Program.cs*
UserRegistrationModel
Email
app.MapIdentityUserEndpoints();
app.Run();
public class UserRegistrationModel
{
public string Email { get; set; }
public string Password { get; set; }
public string FullName { get; set; }
}
public class LoginModel
{
public string Email { get; set; }
public string Password { get; set; }
}
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
IdentityUserEndpoints.cs
WeatherForecastController.cs
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
1:32:30 / 1:42:40 • Organize Web API Endpoints


github.com/CodAffection/Angular-JWT-Login-Logout-with-.Net-Core-API-Asp.Net-Core-Identity/blob/main/AuthECAPI/AuthECAPI/Controllers/IdentityUserEndpoints.cs
Files
main
Go to file
AuthECAPI
.vs
AuthECAPI
Controllers
IdentityUserEndpoints.cs
OrderEndpoints.cs
WeatherForecastController.cs
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
Properties
bin/Debug/net8.0
obj
AuthECAPI.csproj
AuthECAPI.http
Angular-JWT-Login-Logout-with-.Net-Core-API-Asp.Net-Core-Identity / AuthECAPI / AuthECAPI / Controllers / IdentityUserEndpoints.cs
CodAffection first commit
a9b57ae · 6 months ago
History
Code
Blame
86 lines (80 loc) · 3.18 KB
Raw
using AuthECAPI.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
namespace AuthECAPI.Controllers
{
public class UserRegistrationModel
{
public string Email { get; set; }
public string Password { get; set; }
public string FullName { get; set; }
}
public class LoginModel
{
public string Email { get; set; }
public string Password { get; set; }
}
public static class IdentityUserEndpoints
{
public static IEndpointRouteBuilder MapIdentityUserEndpoints(this IEndpointRouteBuilder app)
{
app.MapPost("/signup", CreateUser);
Symbols
Find definitions and references for functions and other symbols in this file by clicking a symbol below or in the code.
Filter symbols
class UserRegistrationModel
class LoginModel
class IdentityUserEndpoints
func MapIdentityUserEndpoints
func CreateUser
func SignIn

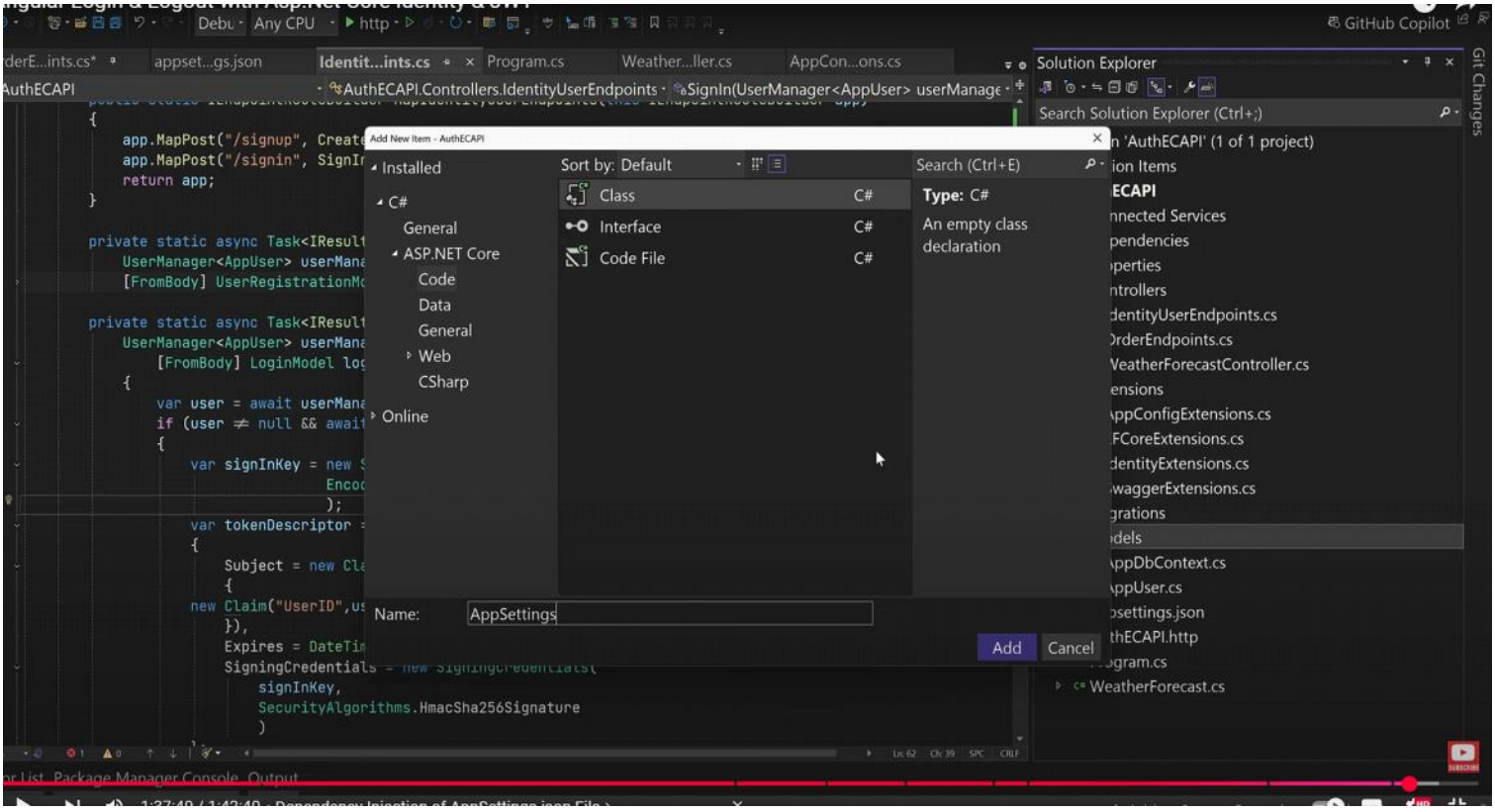

github.com/CodAffection/Angular-JWT-Login-Logout-with-.Net-Core-API-Asp.Net-Core-Identity/blob/main/AuthECAPI/AuthECAPI/Controllers/IdentityUserEndpoints.cs
Files
main
Go to file
AuthECAPI
.vs
AuthECAPI
Controllers
IdentityUserEndpoints.cs
OrderEndpoints.cs
WeatherForecastController.cs
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
Properties
bin/Debug/net8.0
obj
AuthECAPI.csproj
AuthECAPI.http
Angular-JWT-Login-Logout-with-.Net-Core-API-Asp.Net-Core-Identity / AuthECAPI / AuthECAPI / Controllers / IdentityUserEndpoints.cs
Top
Code
Blame
86 lines (80 loc) · 3.18 KB
Raw
public class LoginModel
{
public string Email { get; set; }
public string Password { get; set; }
}
public static class IdentityUserEndpoints
{
public static IEndpointRouteBuilder MapIdentityUserEndpoints(this IEndpointRouteBuilder app)
{
app.MapPost("/signup", CreateUser);
app.MapPost("/signin", SignIn);
return app;
}
private static async Task<IResult> CreateUser(
UserManager<AppUser> userManager,
[FromBody] UserRegistrationModel userRegistrationModel)
{
AppUser user = new AppUser()
{
UserName = userRegistrationModel.Email,
Email = userRegistrationModel.Email,
FullName = userRegistrationModel.FullName,
};
var result = await userManager.CreateAsync(
user,
userRegistrationModel.Password);
if (result.Succeeded)
return Results.Ok(result);
Symbols
Find definitions and references for functions and other symbols in this file by clicking a symbol below or in the code.
Filter symbols
class UserRegistrationModel
class LoginModel
class IdentityUserEndpoints
func MapIdentityUserEndpoints
func CreateUser
func SignIn

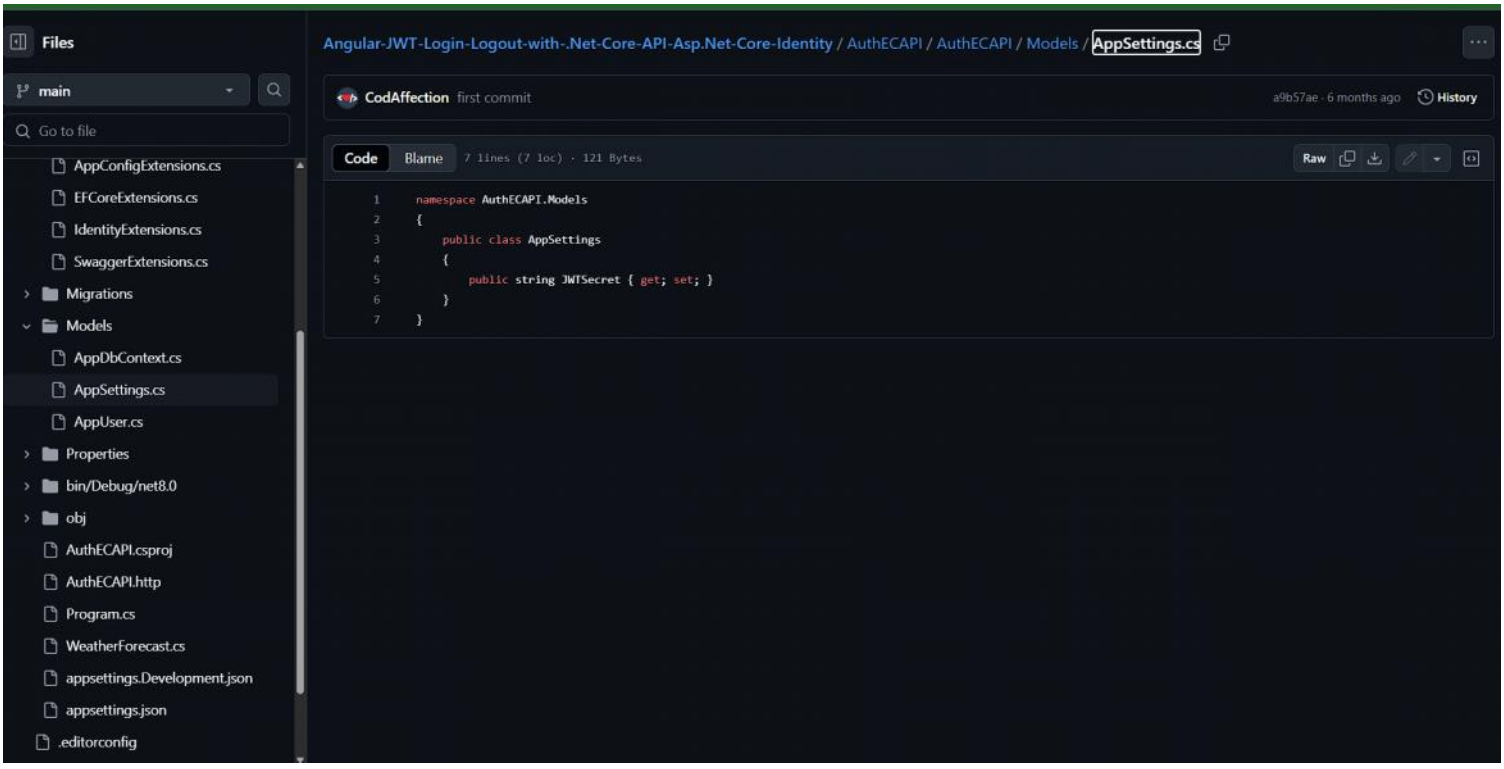


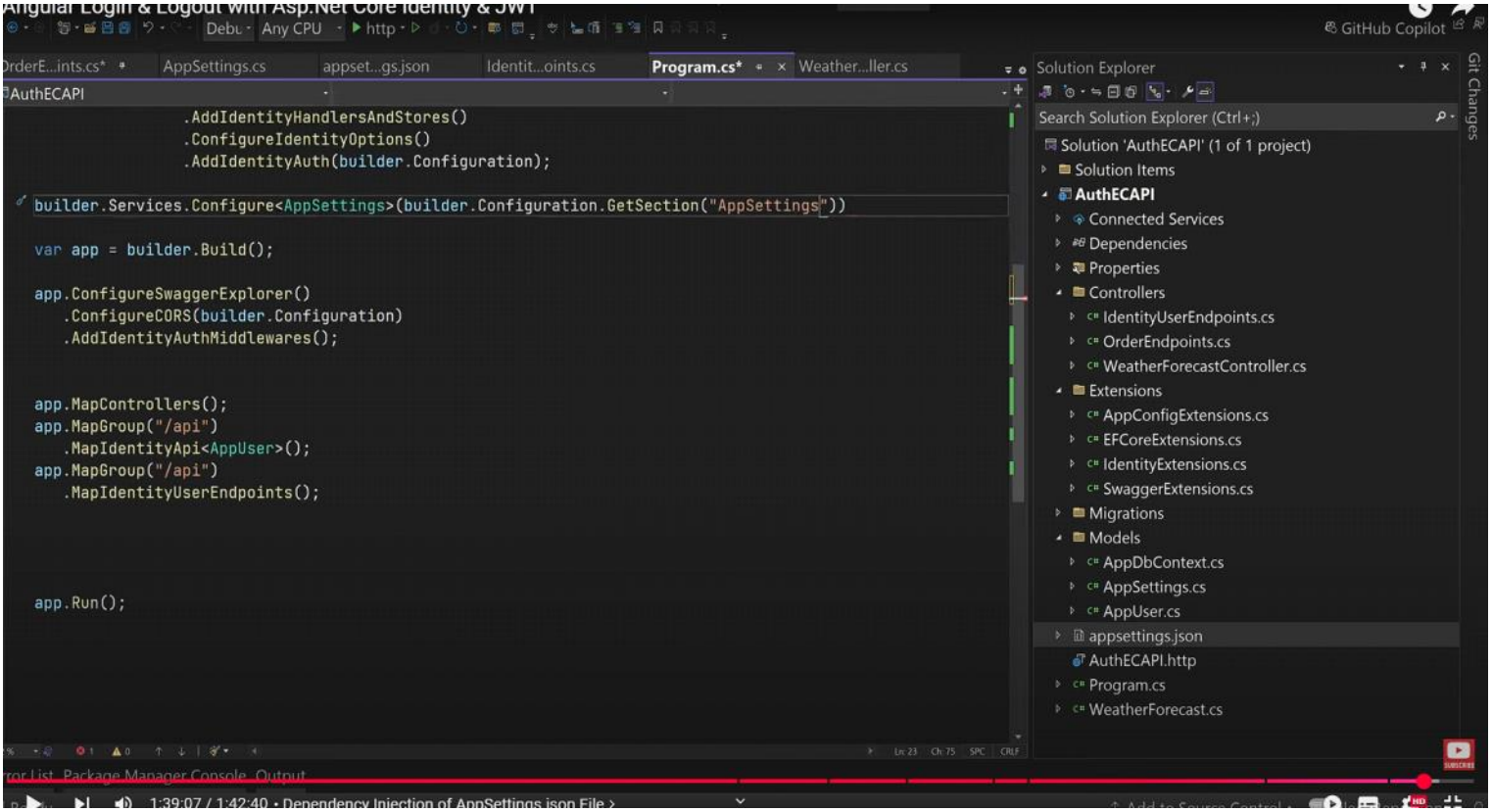

```
25      public static class IdentityUserEndpoints
51                  return Results.BadRequest(result);
52              }
53
54          private static async Task<IResult> SignIn(
55              UserManager<AppUser> userManager,
56                  [FromBody] LoginModel loginModel,
57                  IOptions<AppSettings> appSettings)
58              {
59                  var user = await userManager.FindByEmailAsync(loginModel.Email);
60                  if (user != null && await userManager.CheckPasswordAsync(user, loginModel.Password))
61                  {
62                      var signInKey = new SymmetricSecurityKey(
63                                      Encoding.UTF8.GetBytes(appSettings.Value.JWTSecret)
64                                      );
65                      var tokenDescriptor = new SecurityTokenDescriptor
66                      {
67                          Subject = new ClaimsIdentity(new Claim[]
68                          {
69                      new Claim("UserID",user.Id.ToString())
70                          }),
71                          Expires = DateTime.UtcNow.AddDays(10),
72                          SigningCredentials = new SigningCredentials(
73                              signInKey,
74                              SecurityAlgorithms.HmacSha256Signature
75                              )
76                      };
77                      var tokenHandler = new JwtSecurityTokenHandler();
78                      var securityToken = tokenHandler.CreateToken(tokenDescriptor);
79                      var token = tokenHandler.WriteToken(securityToken);
80                      return Results.Ok(new { token });
81                  }
82                  else
```

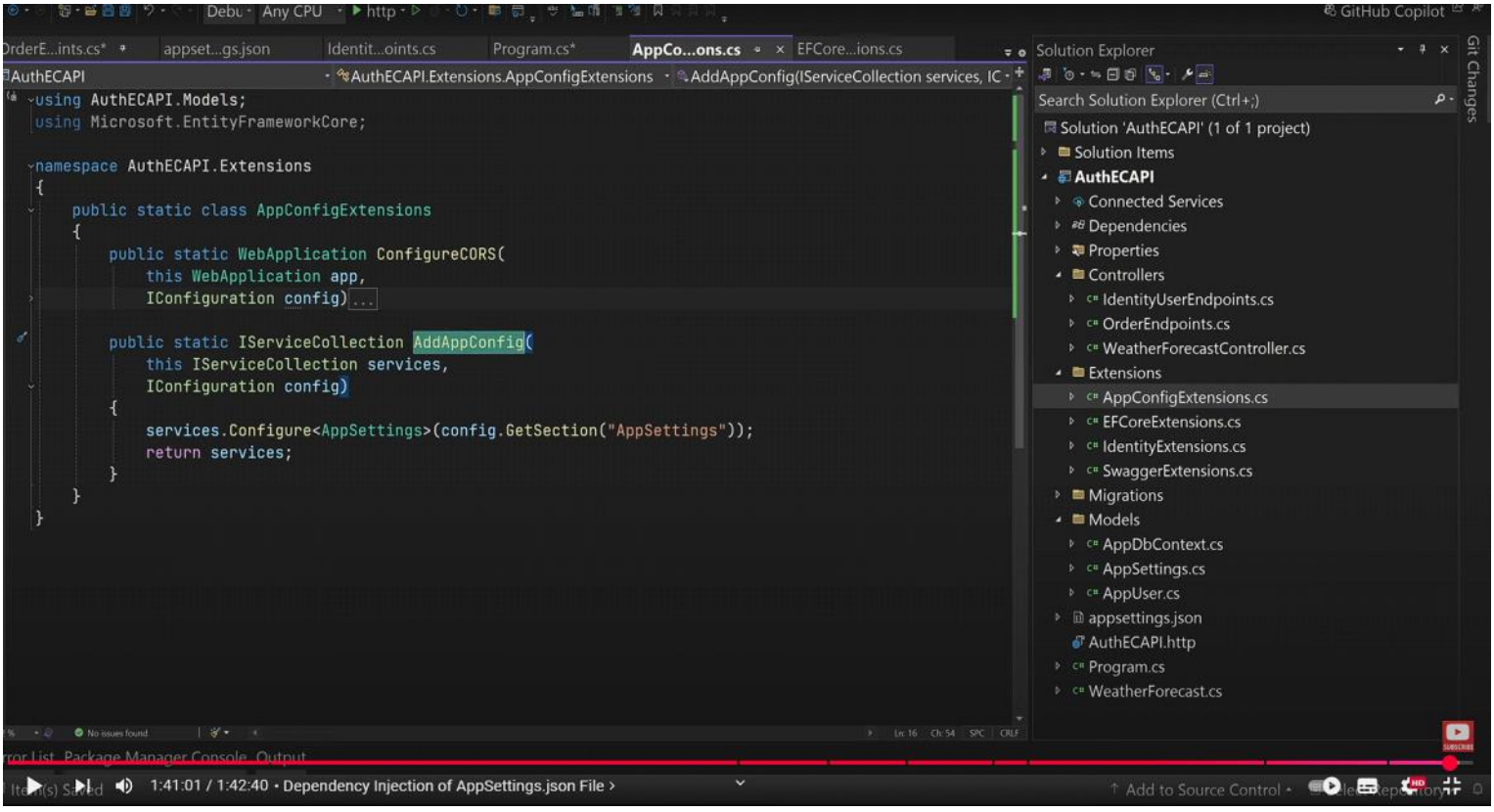


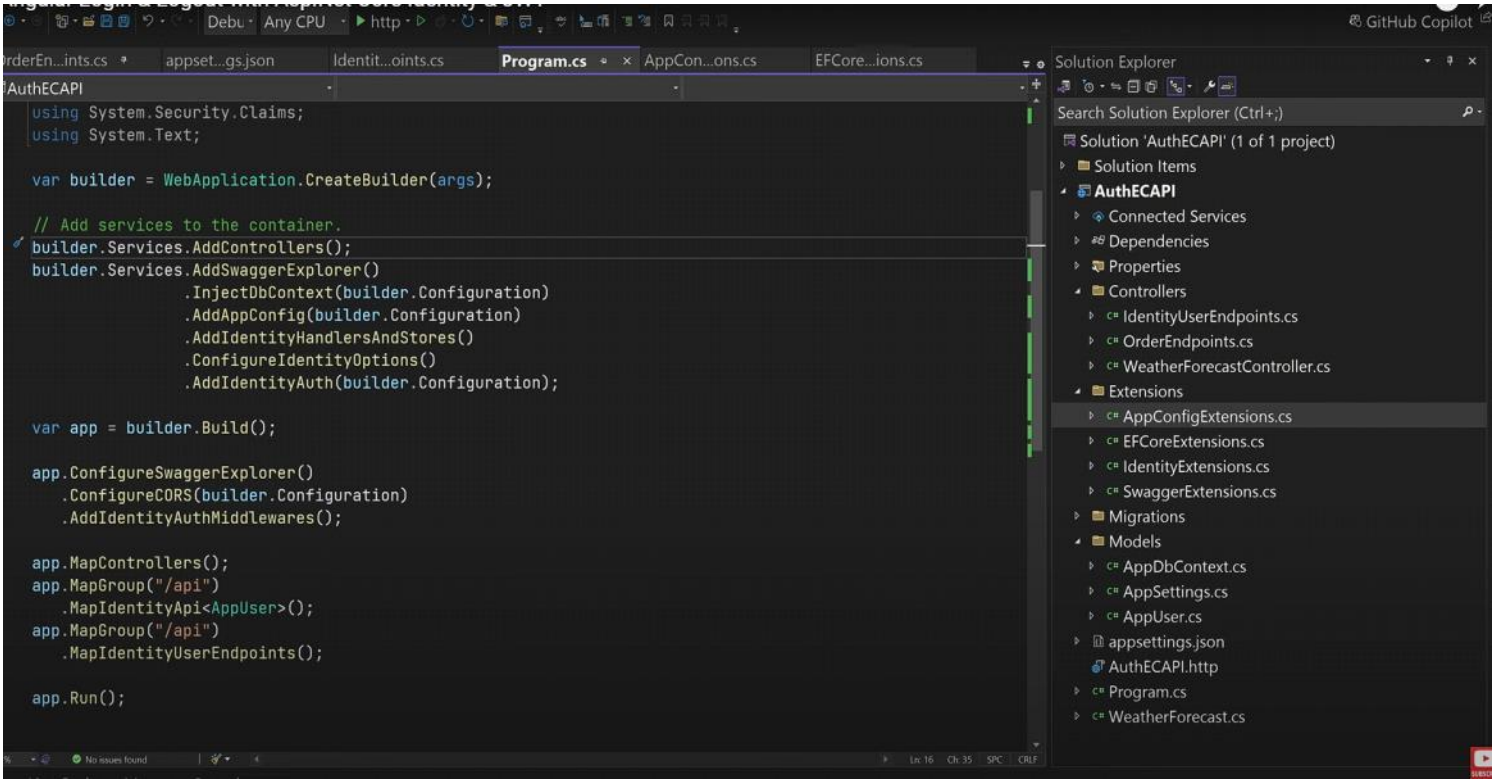

```
6   using Microsoft.AspNetCore.Mvc;
7   using Microsoft.EntityFrameworkCore;
8   using Microsoft.IdentityModel.Tokens;
9   using System.IdentityModel.Tokens.Jwt;
10  using System.Security.Claims;
11  using System.Text;
12
13  var builder = WebApplication.CreateBuilder(args);
14
15  // Add services to the container.
16  builder.Services.AddControllers();
17  builder.Services.AddSwaggerExplorer()
18                  .InjectDbContext(builder.Configuration)
19                  .AddAppConfig(builder.Configuration)
20                  .AddIdentityHandlersAndStores()
21                  .ConfigureIdentityOptions()
22                  .AddIdentityAuth(builder.Configuration);
23
24  var app = builder.Build();
25
26  app.ConfigureSwaggerExplorer()
27     .ConfigureCORS(builder.Configuration)
28     .AddIdentityAuthMiddlewares();
29
30  app.MapControllers();
31  app.MapGroup("/api")
32     .MapIdentityApi<AppUser>();
33  app.MapGroup("/api")
34     .MapIdentityUserEndpoints();
35
36  app.Run();
```


Add New Item - AuthECAPI
Installed
C#
General
ASP.NET Core
Code
Data
General
Web
CSharp
Online
Sort by: Default
Class C#
Interface C#
Code File C#
Search (Ctrl+E)
Type: C#
An empty class declaration
Name: AppSettings
Add
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
GitHub Copilot
Git Changes


Files
main
Go to file
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
AppDbContext.cs
AppSettings.cs
AppUser.cs
Properties
bin/Debug/net8.0
obj
AuthECAPI.csproj
AuthECAPI.http
Program.cs
WeatherForecast.cs
appsettings.Development.json
appsettings.json
.editorconfig
Angular-JWT-Login-Logout-with-.Net-Core-API-Asp.Net-Core-Identity / AuthECAPI / AuthECAPI / Models / AppSettings.cs
CodAffection first commit
a9b57ae · 6 months ago
History
Code
Blame
7 lines (7 loc) · 121 Bytes
Raw
namespace AuthECAPI.Models
{
public class AppSettings
{
public string JWTSecret { get; set; }
}
}


Angular Login & Logout with Asp.Net Core Identity & JWT
GitHub Copilot
Program.cs*
AppSettings.cs
.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions()
.AddIdentityAuth(builder.Configuration);
builder.Services.Configure<AppSettings>(builder.Configuration.GetSection("AppSettings"))
var app = builder.Build();
app.ConfigureSwaggerExplorer()
.ConfigureCORS(builder.Configuration)
.AddIdentityAuthMiddlewares();
app.MapControllers();
app.MapGroup("/api")
.MapIdentityApi<AppUser>();
app.MapGroup("/api")
.MapIdentityUserEndpoints();
app.Run();
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
IdentityUserEndpoints.cs
OrderEndpoints.cs
WeatherForecastController.cs
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
AppDbContext.cs
AppSettings.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs
Git Changes
1:39:07 / 1:42:40 • Dependency Injection of AppSettings.json File >


AppConfigExtensions.cs
using AuthECAPI.Models;
using Microsoft.EntityFrameworkCore;
namespace AuthECAPI.Extensions
{
public static class AppConfigExtensions
{
public static WebApplication ConfigureCORS(
this WebApplication app,
IConfiguration config)
public static IServiceCollection AddAppConfig(
this IServiceCollection services,
IConfiguration config)
{
services.Configure<AppSettings>(config.GetSection("AppSettings"));
return services;
}
}
}
Solution Explorer
Solution 'AuthECAPI' (1 of 1 project)
AuthECAPI
Controllers
Extensions
Models
1:41:01 / 1:42:40 • Dependency Injection of AppSettings.json File


Debu
Any CPU
http
GitHub Copilot
OrderEn...ints.cs
appset...gs.json
Identit...oints.cs
Program.cs
AppCon...ons.cs
EFCore...ions.cs
AuthECAPI
using System.Security.Claims;
using System.Text;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddSwaggerExplorer()
.InjectDbContext(builder.Configuration)
.AddAppConfig(builder.Configuration)
.AddIdentityHandlersAndStores()
.ConfigureIdentityOptions()
.AddIdentityAuth(builder.Configuration);
var app = builder.Build();
app.ConfigureSwaggerExplorer()
.ConfigureCORS(builder.Configuration)
.AddIdentityAuthMiddlewares();
app.MapControllers();
app.MapGroup("/api")
.MapIdentityApi<AppUser>();
app.MapGroup("/api")
.MapIdentityUserEndpoints();
app.Run();
No issues found
Ln: 16
Ch: 35
SPC
CRLF
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AuthECAPI' (1 of 1 project)
Solution Items
AuthECAPI
Connected Services
Dependencies
Properties
Controllers
IdentityUserEndpoints.cs
OrderEndpoints.cs
WeatherForecastController.cs
Extensions
AppConfigExtensions.cs
EFCoreExtensions.cs
IdentityExtensions.cs
SwaggerExtensions.cs
Migrations
Models
AppDbContext.cs
AppSettings.cs
AppUser.cs
appsettings.json
AuthECAPI.http
Program.cs
WeatherForecast.cs