* Explain CORS enablement for Web API access for local application
  + What is CORS?, How to enable CORS thru Startup.cs, Install Cors nuget package to Web API application

CORS, or Cross-Origin Resource Sharing, is a security feature in browsers that keeps web applications from one origin (like http://localhost:4200) from accessing resources on another origin (like http://localhost:5065). This is a vital security measure that helps block unauthorized cross-site requests. However, when you're developing locally—like when a frontend app needs to call an ASP.NET Core Web API—you often need to allow these cross-origin requests explicitly. That’s where enabling CORS comes into play. To set up CORS in your ASP.NET Core Web API application, you start by defining a CORS policy in the Program.cs file (or Startup.cs for older projects). This policy outlines which origins, headers, and HTTP methods are permitted. For instance, if you want to allow requests from a frontend app running on http://localhost:4200, you can create a policy using builder.Services.AddCors() and configure it with WithOrigins("http://localhost:4200").AllowAnyHeader().AllowAnyMethod(). Once you’ve defined the policy, you need to apply it in the middleware pipeline with app.UseCors("PolicyName") before any authentication or authorization middleware kicks in. While CORS support is built into ASP.NET Core 3.1 and later, you can also install the Microsoft.AspNetCore.Cors NuGet package if necessary. This setup makes sure your local frontend app can securely communicate with your Web API without hitting any cross-origin errors in the browser console. So, enabling CORS is a key step when you're building distributed applications or working with separate frontend and backend projects during development.