Lab Manual for Cloud Computing

Lab No. 5

Creating ASP.Net Core Web API

LAB 05: CREATING ASP.NET CORE WEB API

1. INTRODUCTION:

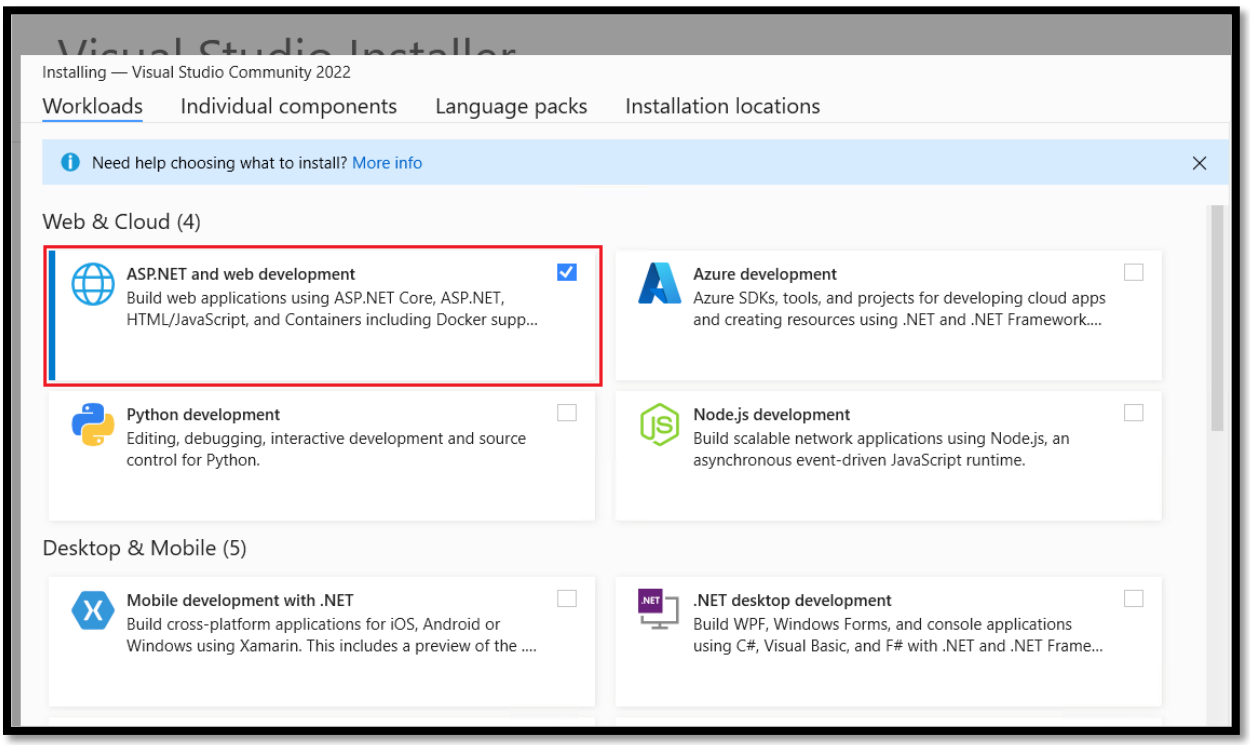
An API (Application Programming Interface) acts as an intermediary, enabling different software applications to communicate with each other by defining a set of rules and protocols. Specifically, a Web API extends this functionality to the web, allowing web-based applications to interact with each other or with backend services over the internet. ASP.NET Core, a robust and cross-platform framework, offers developers the tools to create efficient and scalable Web APIs seamlessly integrated with web applications. Leveraging ASP.NET Core's features, developers can design endpoints that handle HTTP requests and responses, enabling data exchange and interaction between client and server components. This concise introduction encapsulates the essence of APIs, Web APIs, and their synergy with ASP.NET Core web applications, facilitating seamless communication and data exchange over the internet.

ASP.NET Core Web APIs leverage the power of HTTP methods like GET, POST, PUT, and DELETE to enable CRUD (Create, Read, Update, Delete) operations on resources. With ASP.NET Core's middleware pipeline, developers can easily integrate authentication, authorization, and other cross-cutting concerns into their Web API endpoints. This combination of ASP.NET Core's versatility and Web API functionality empowers developers to build robust, secure, and scalable web applications that can efficiently handle diverse client requests while ensuring seamless data exchange and interoperability.

Prerequisites:

1. **Visual studio 2022** with the **ASP.NET and web development** workload

If they aren't installed, run the visual studio installer, and go to the option "modify". In the workload, select it and then click option modify.



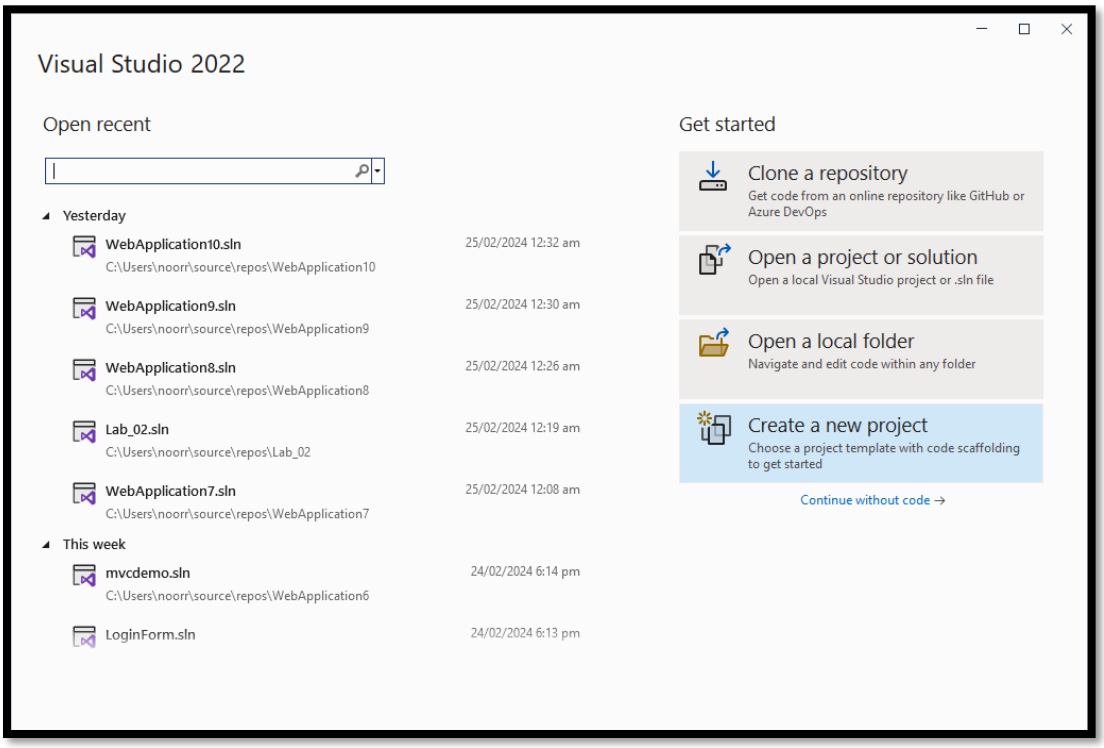
Creating a Project

In this .NET Core Web API project, we aim to develop a todo APIs, which can be done in following steps:

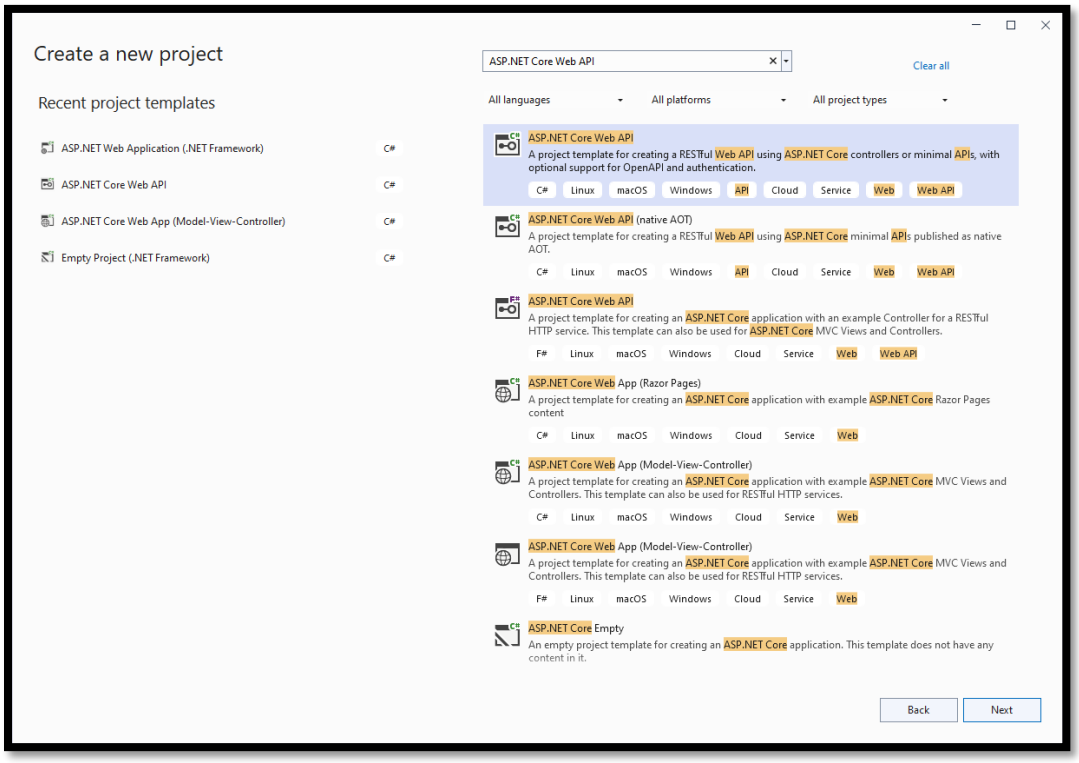
Step 1: Create a new project

- Open Visual Studio 2022.

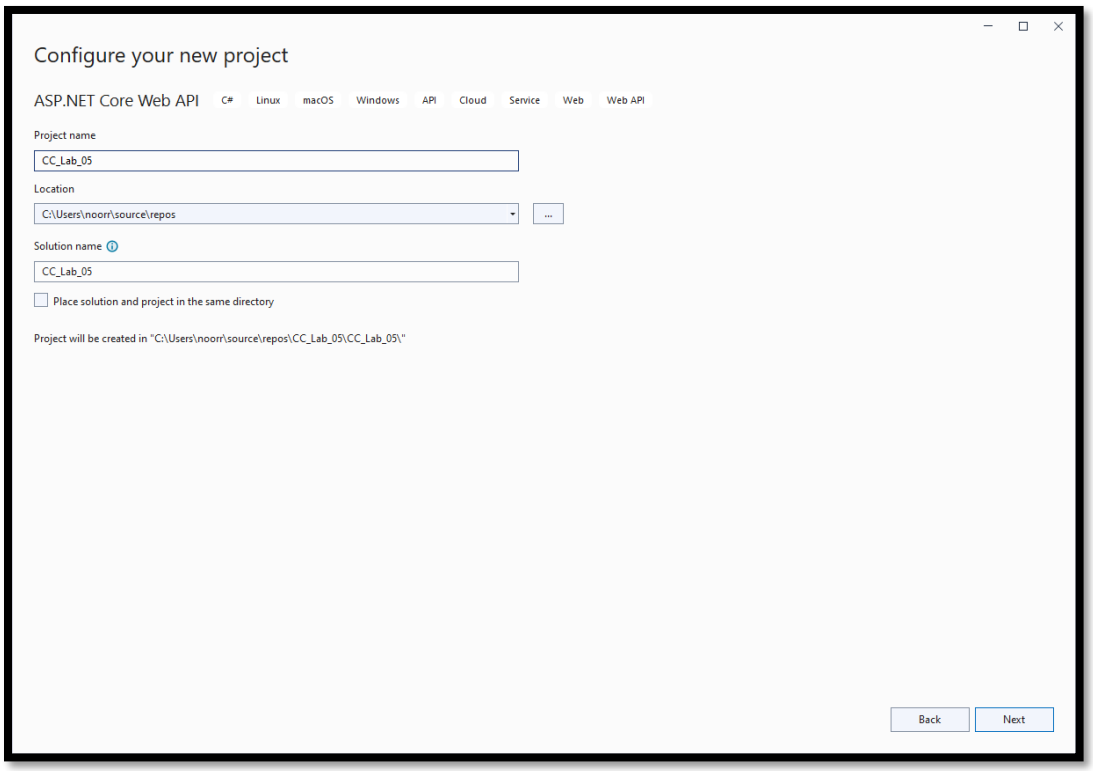
- Click on "Create a new project" in the start window.



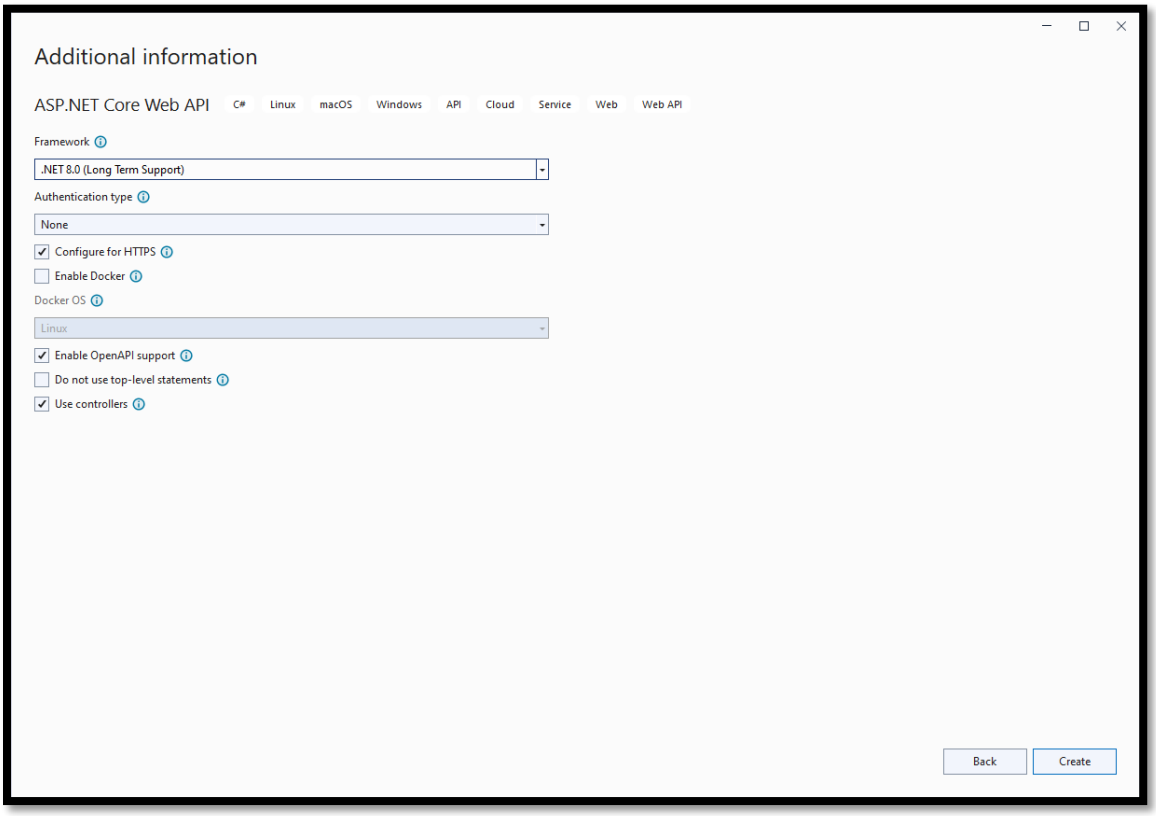
- In the "Create a new project" window, search for "ASP.NET Core Web API" in the search bar, and then click NEXT.

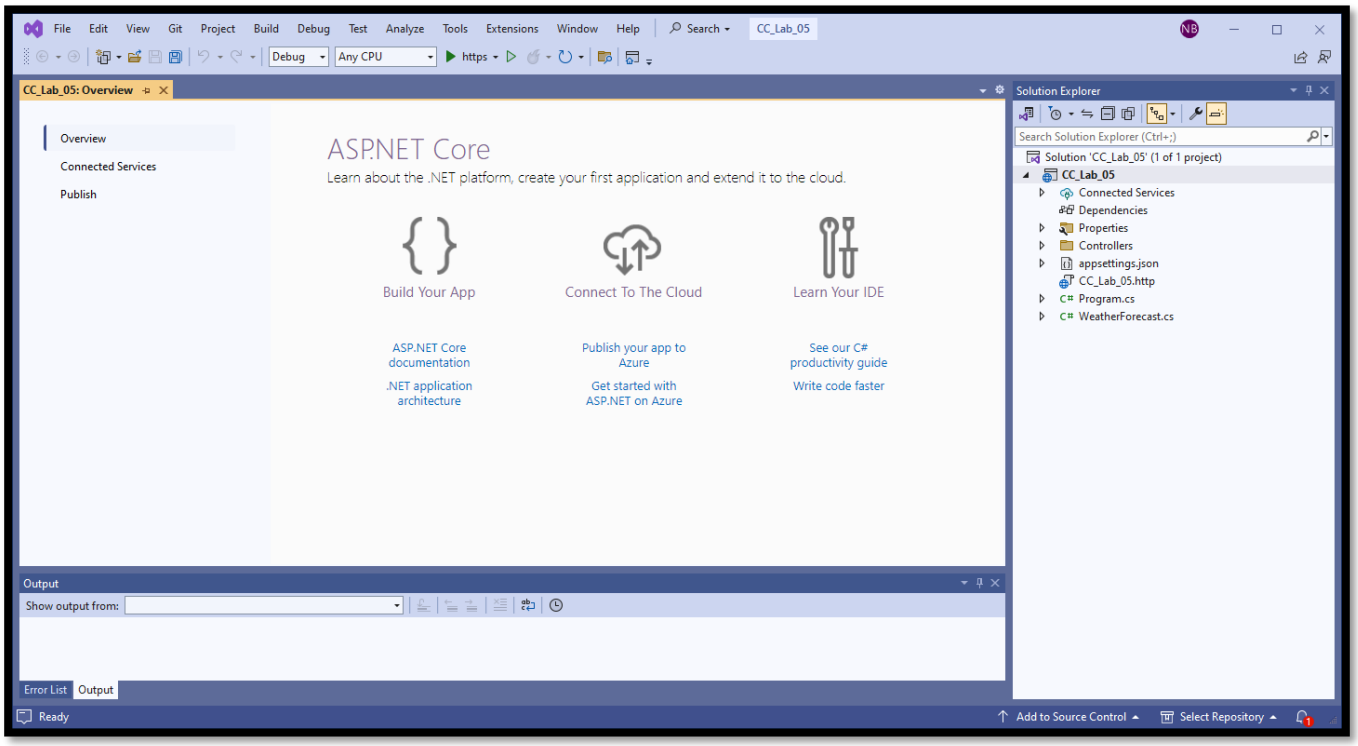


- Provide a name and location for your project and then Click NEXT.



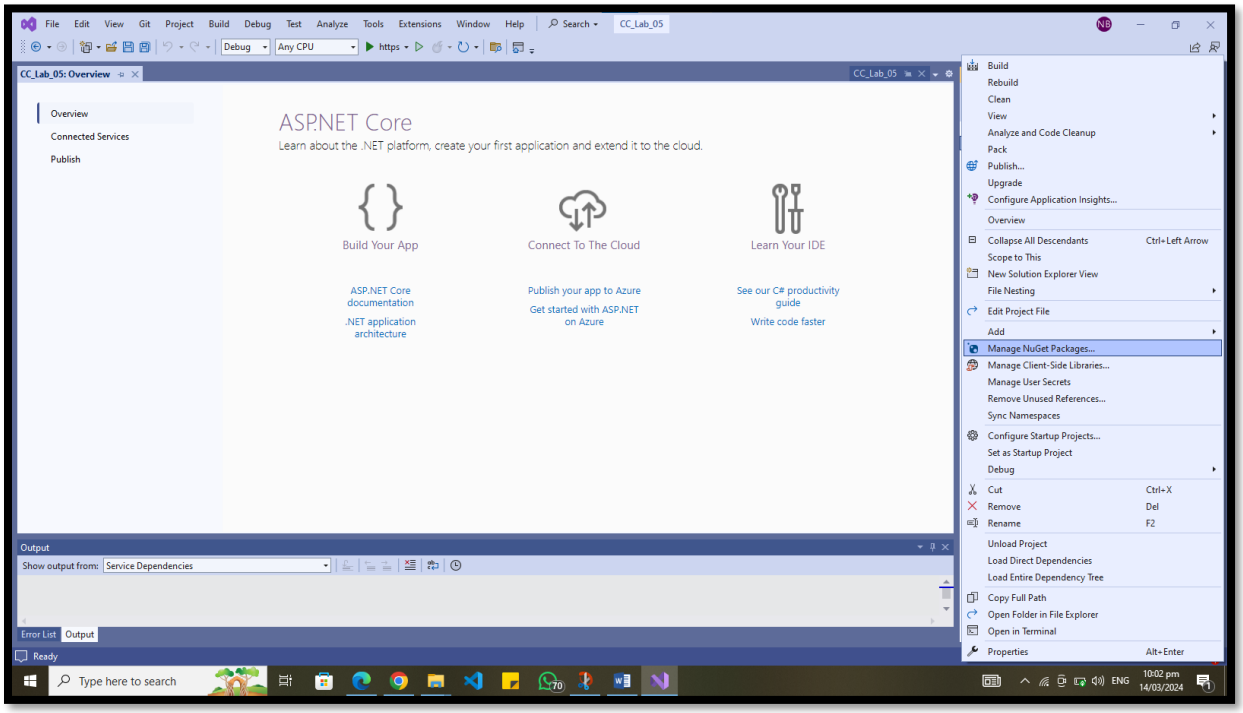
- Select the desired framework version (.NET Core) and authentication type, and Click CREATE.



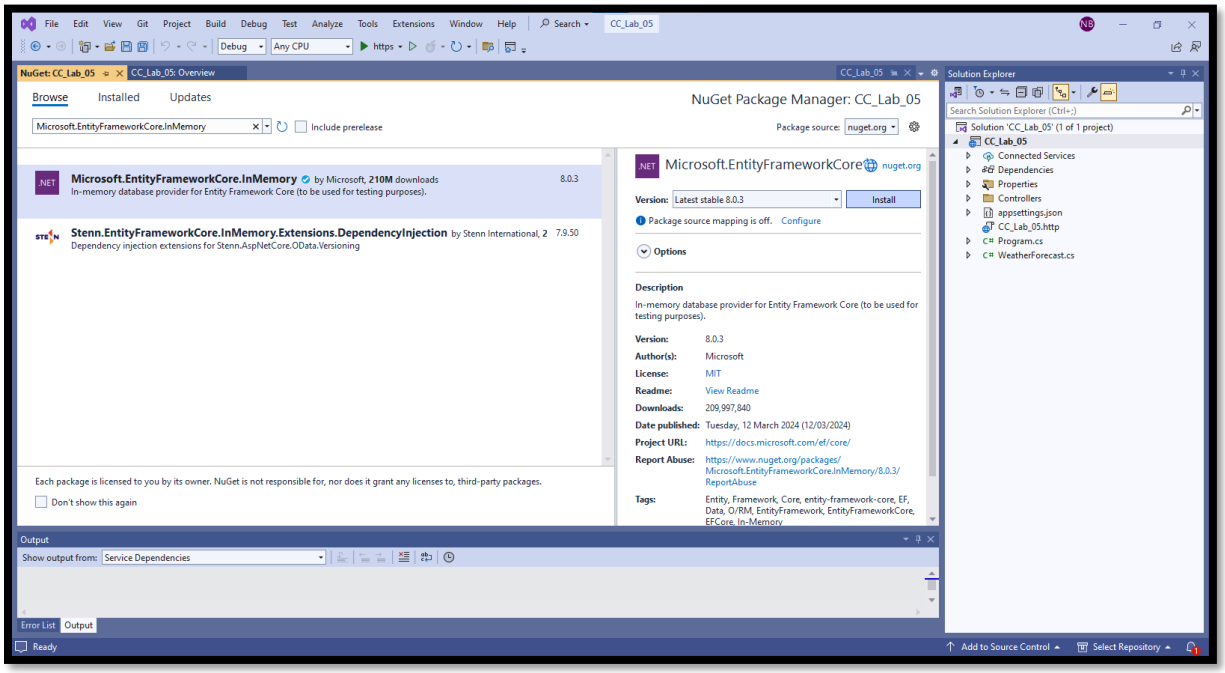


Step 2: Adding a NuGet package

- Right-click on your project in Solution Explorer, and then Choose "Manage Nugget Packages".

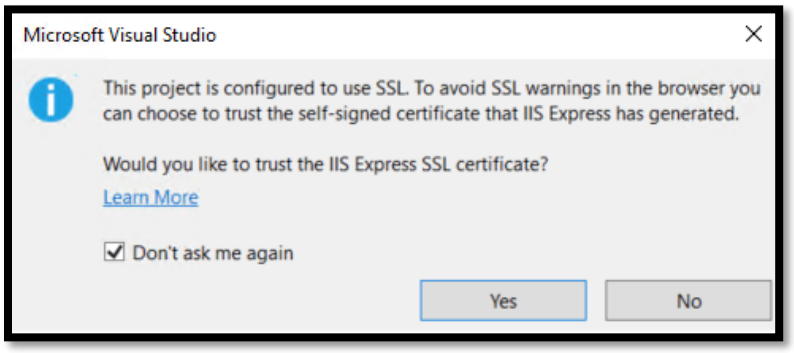


- Select the **Browse** tab. Enter **Microsoft.EntityFrameworkCore.InMemory** in the search box, and then select Microsoft.EntityFrameworkCore.InMemory, then click install.



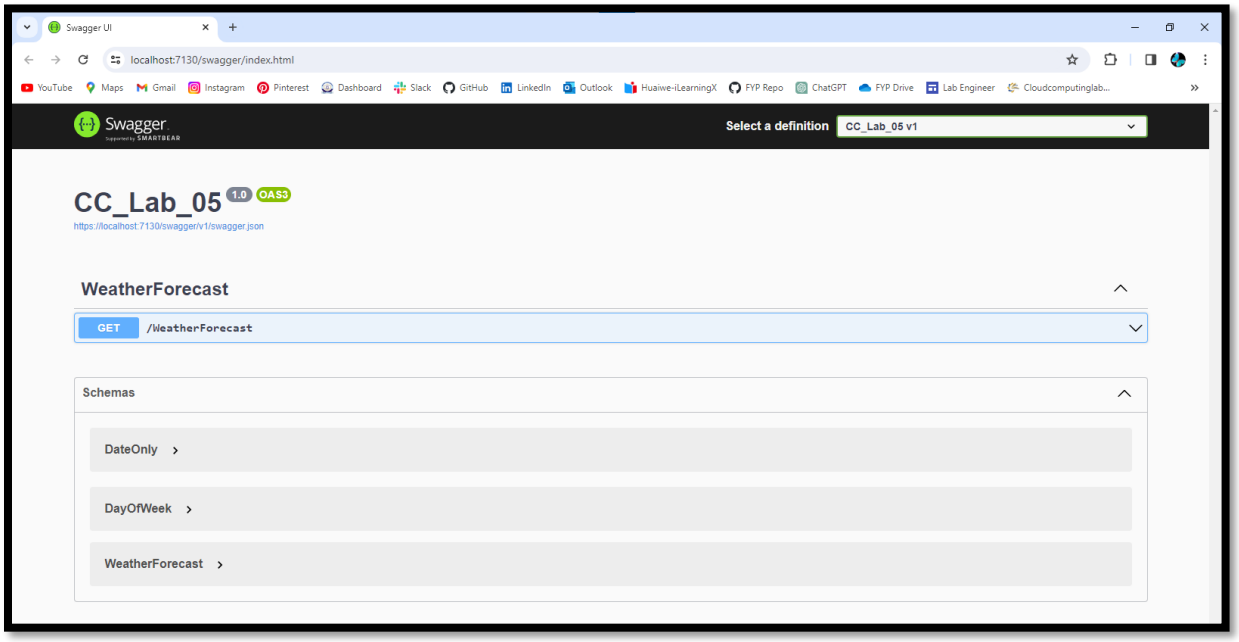
Step 3: Test the project

- The project template creates a WeatherForecast API with support for Swagger. Press Ctrl+F5 to run without the debugger.Visual Studio displays the following dialog when a project is not yet configured to use SSL.



- Select **Yes** if you trust the IIS Express SSL certificate. The following dialog is displayed, Select **Yes** if you agree to trust the development certificate.

- Visual Studio launches the default browser and navigates to https://localhost:<port>/swagger/index.html, where <port> is a randomly chosen port number set at the project creation.

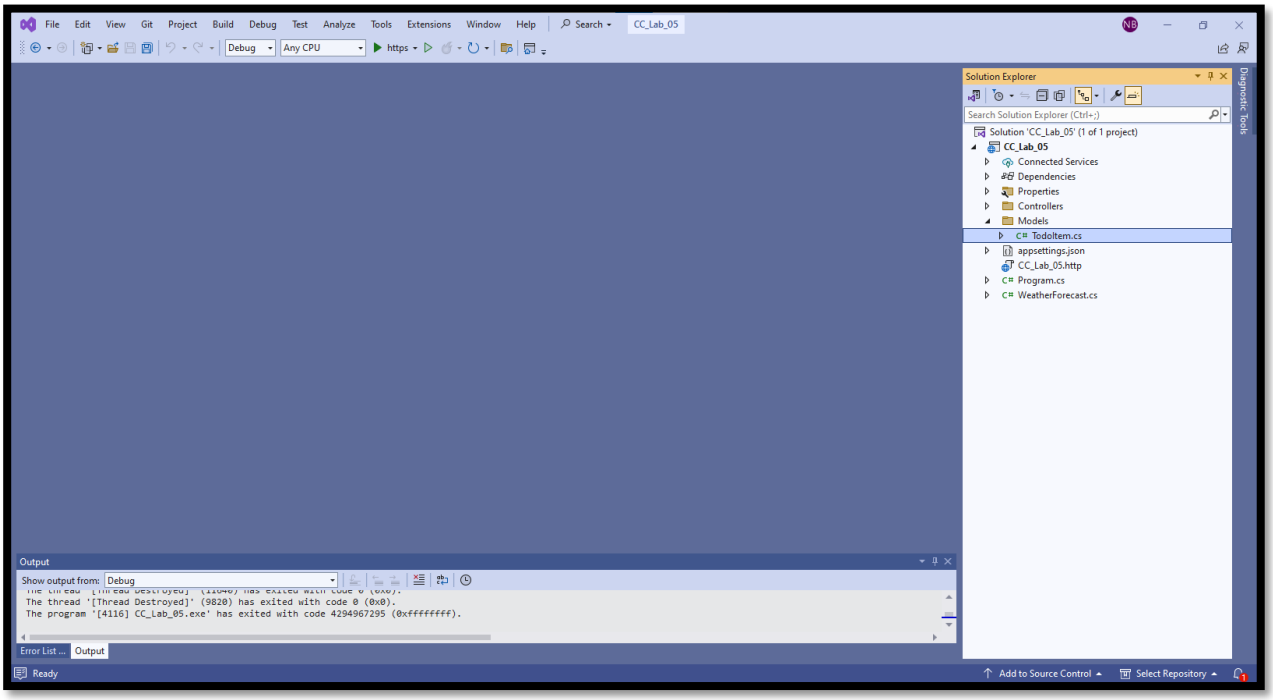


Now come back to the visual studio, and start creating the to do Item API by using following steps.

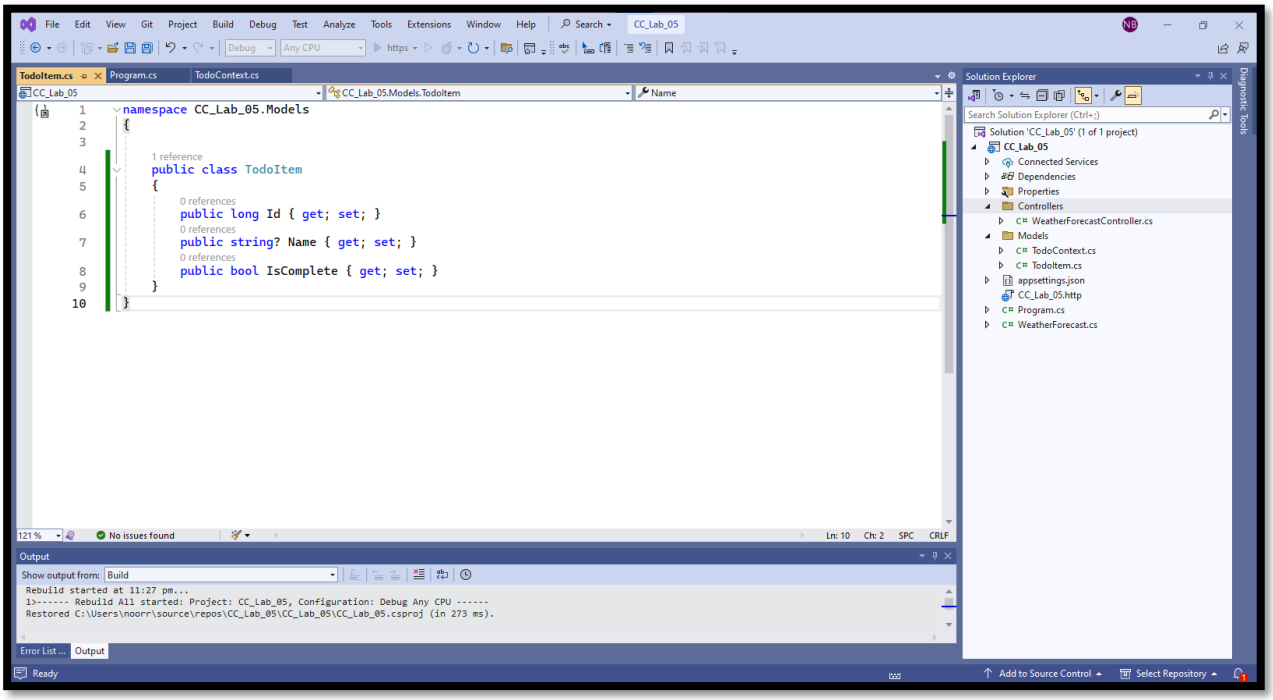
Step 4: Add a model class

A *model* is a set of classes that represent the data that the app manages. The model for this app is the TodoItem class.

- In **Solution Explorer,** right-click the project. Select **Add** > **New Folder.** Name the folder Models.Right-click the Models folder and select **Add** > **Class.** Name the class *TodoItem* and select **Add.**



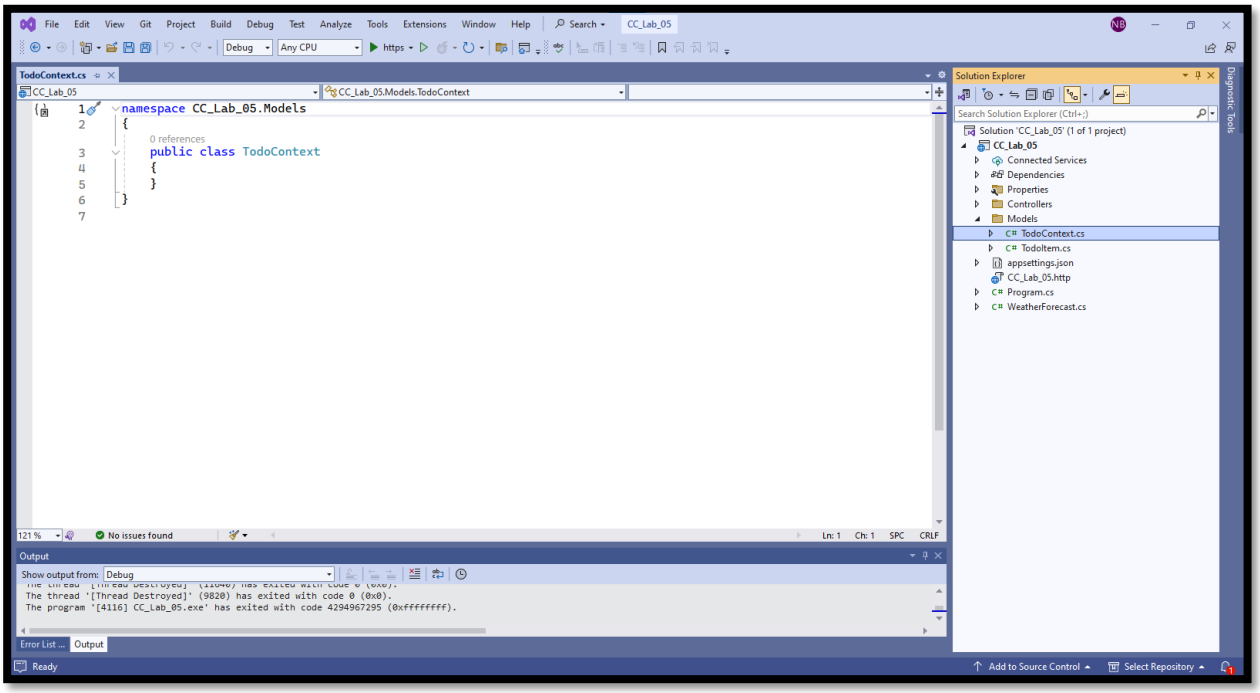
- Replace the template code with the following:



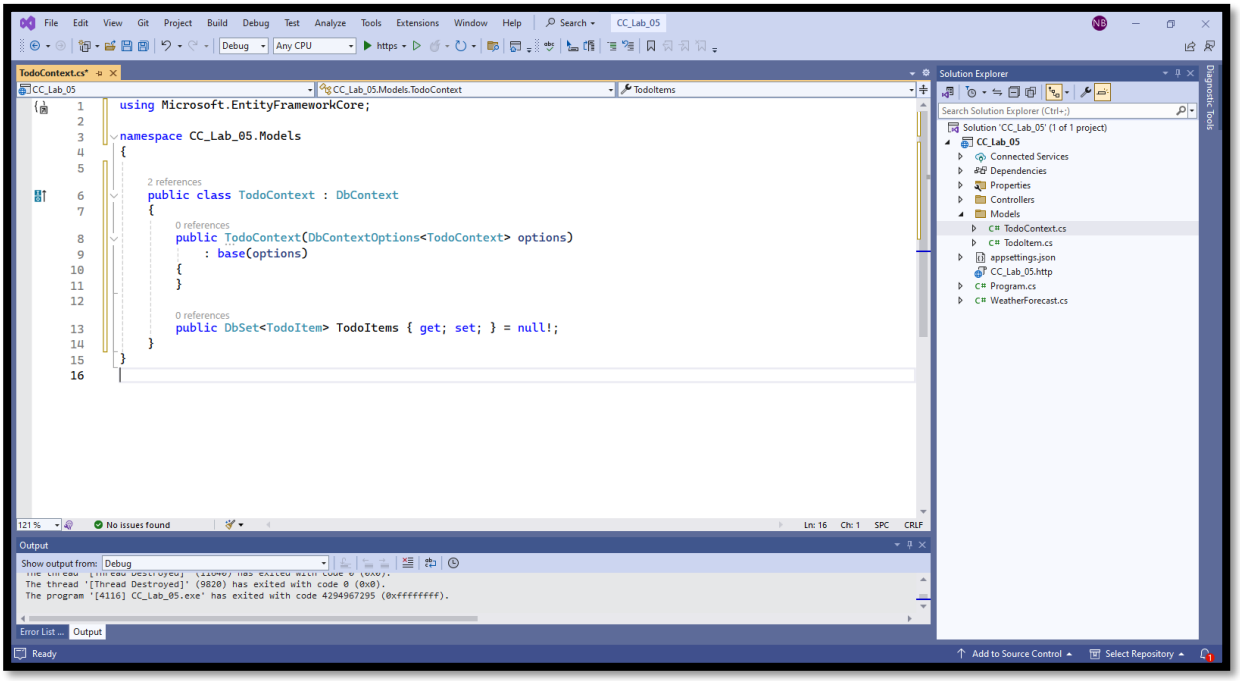
Step 5: Add a database context

The *database context* is the main class that coordinates Entity Framework functionality for a data model. This class is created by deriving from the Microsoft.EntityFrameworkCore.DbContext class.

- Right-click the Models folder and select **Add** > **Class.** Name the class *TodoContext* and click **Add.**



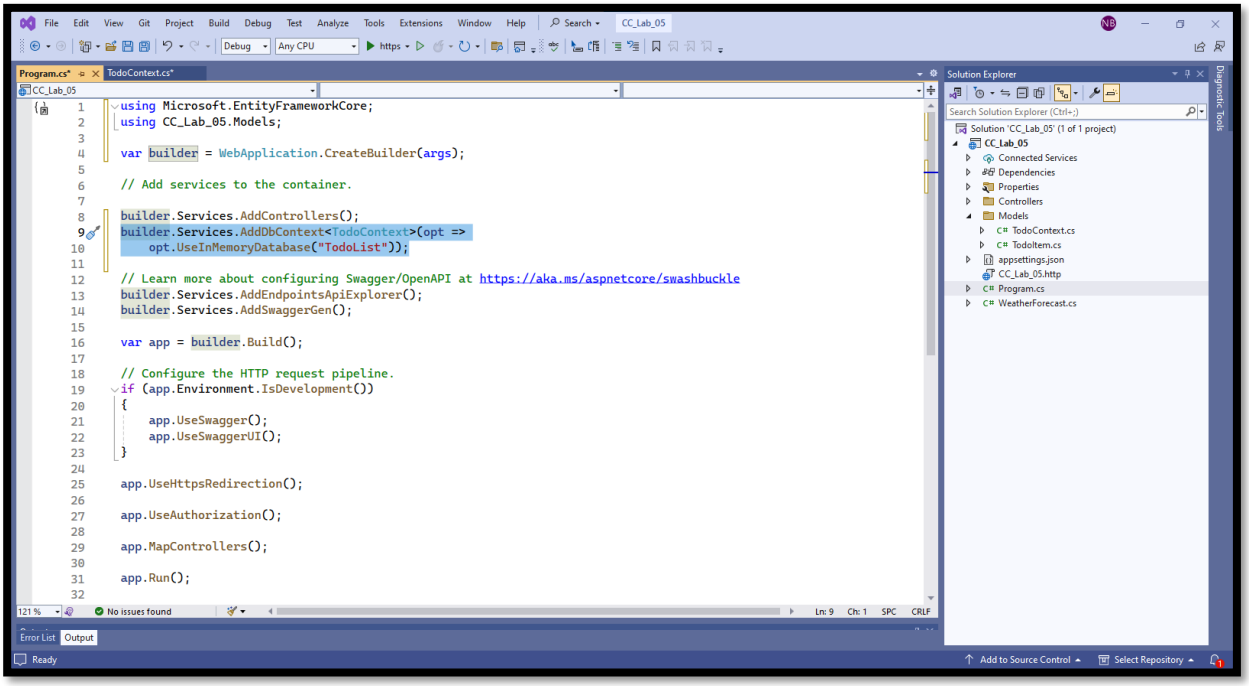
- Enter the following code:



Step 6: Register the database context

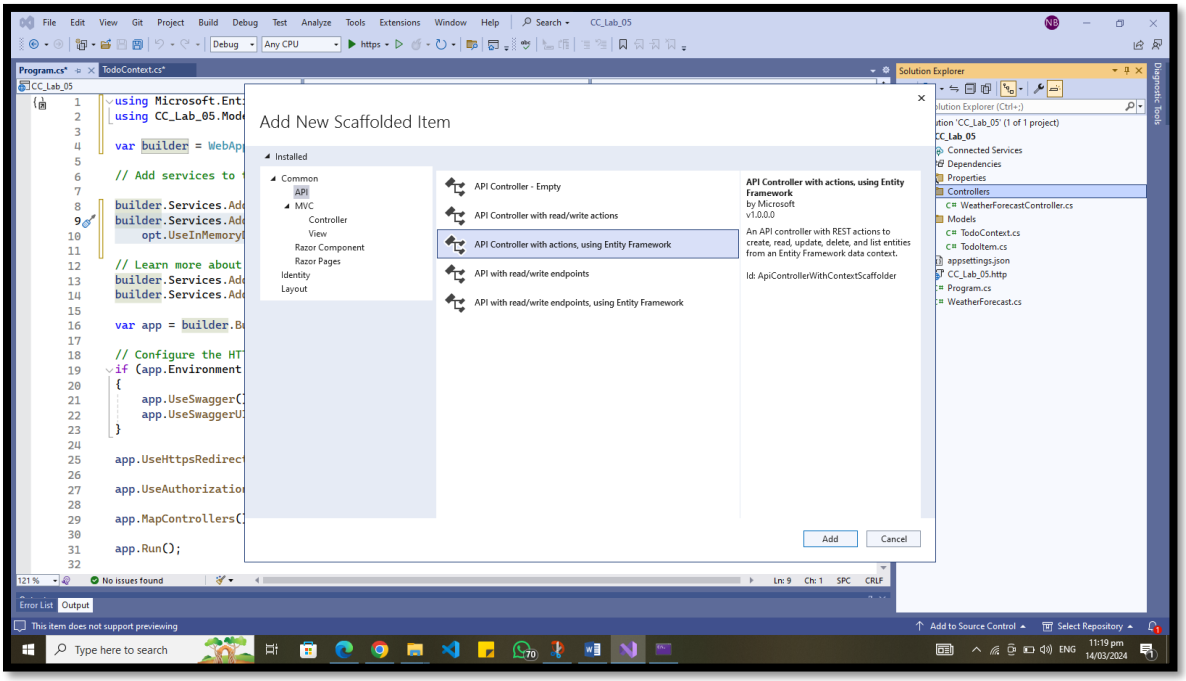
In ASP.NET Core, services such as the DB context must be registered with the dependency injection (DI) container. The container provides the service to controllers.

- Update Program.cs with the following highlighted code:

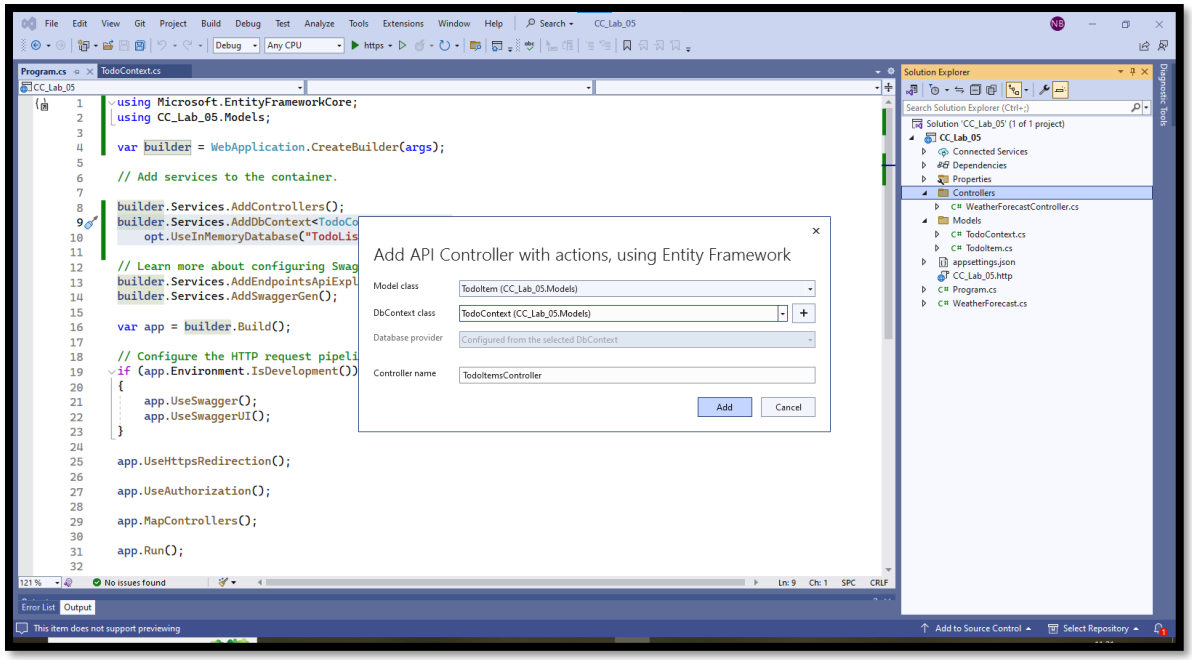


Step 7: Scaffold a controller

- Right-click the *Controllers* folder. Select **Add** > **Controller**. Select **API Controller with actions, using Entity Framework**, and then select **Add.**

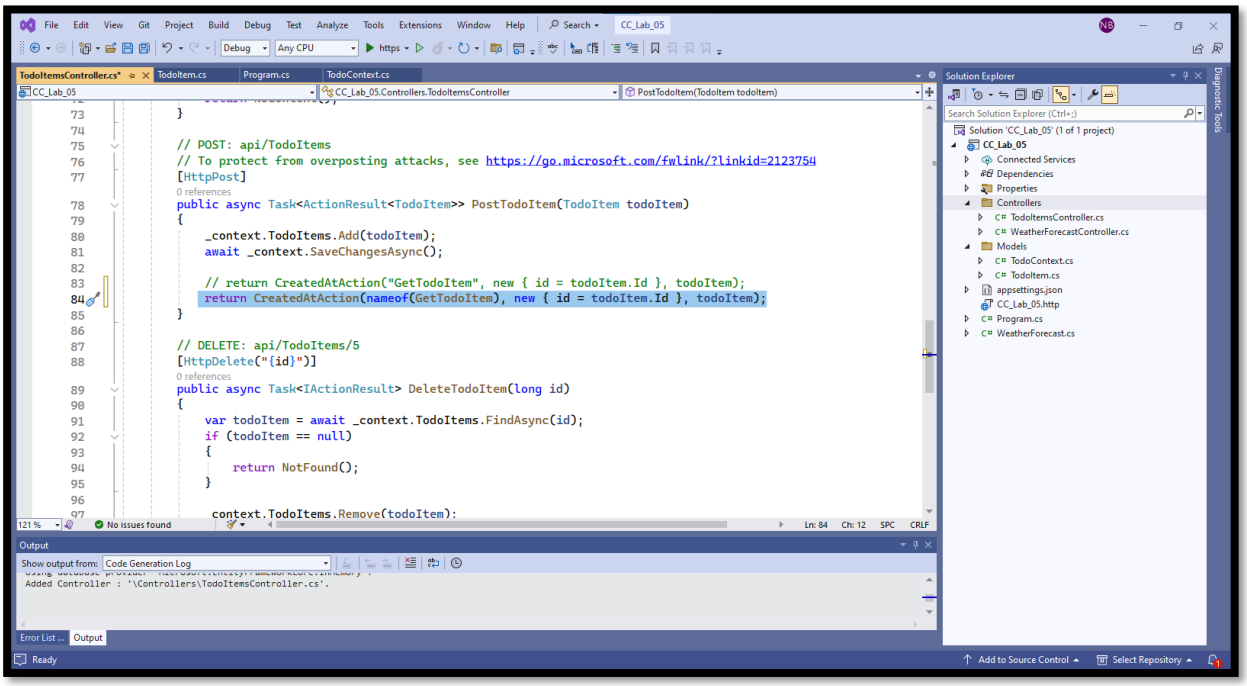


- In the **Add API Controller with actions, using Entity Framework** dialog. Select **TodoItem (TodoApi.Models)** in the **Model class**, **TodoContext (TodoApi.Models)** in the **Data context class.** Then select **Add.**

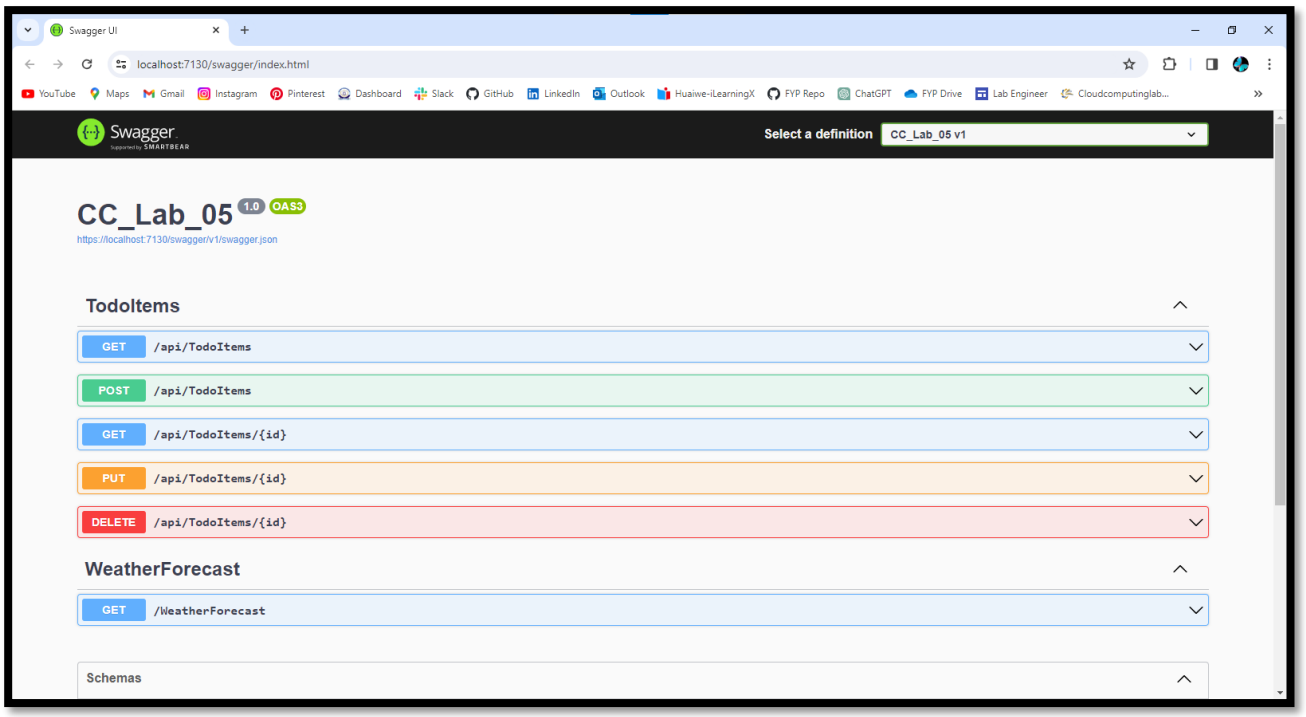


Step 8: Update the methods in TodoItemsController

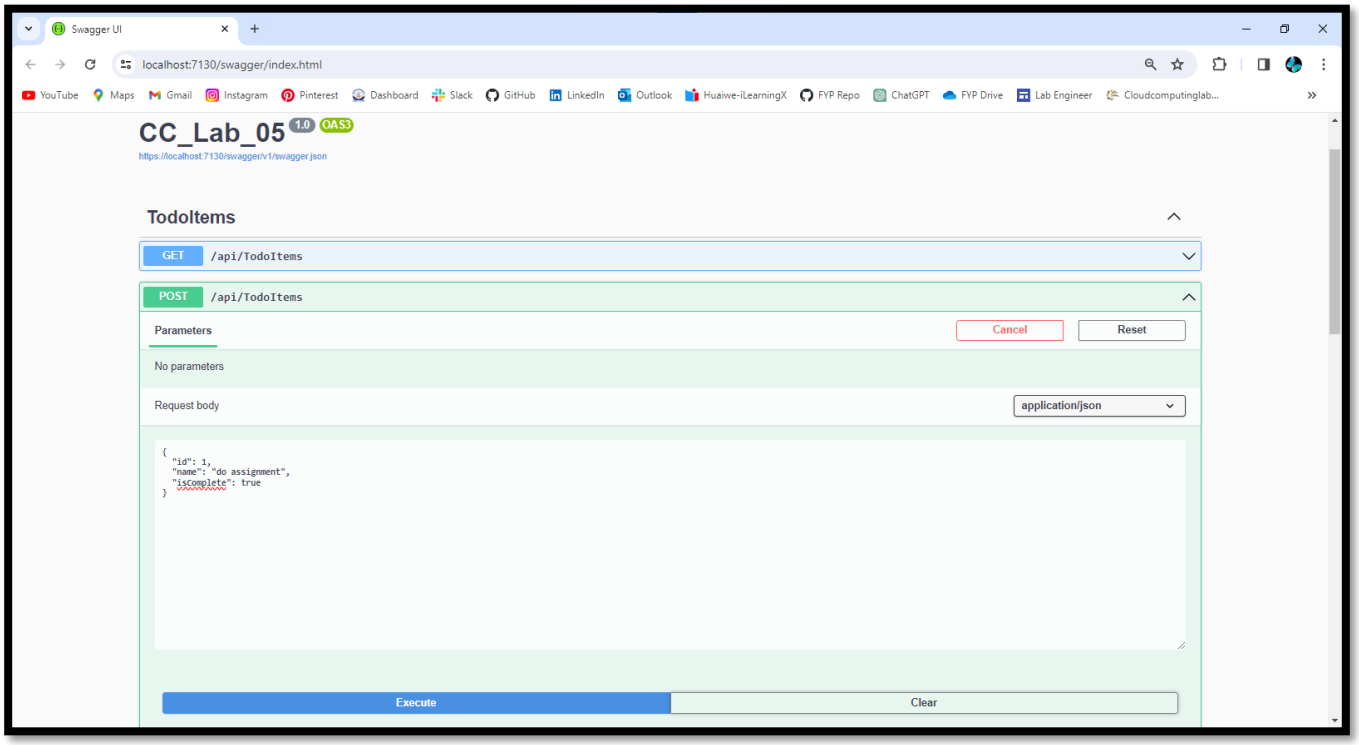
- Update the return statement in the `PostTodoItem` to use the nameof operator:

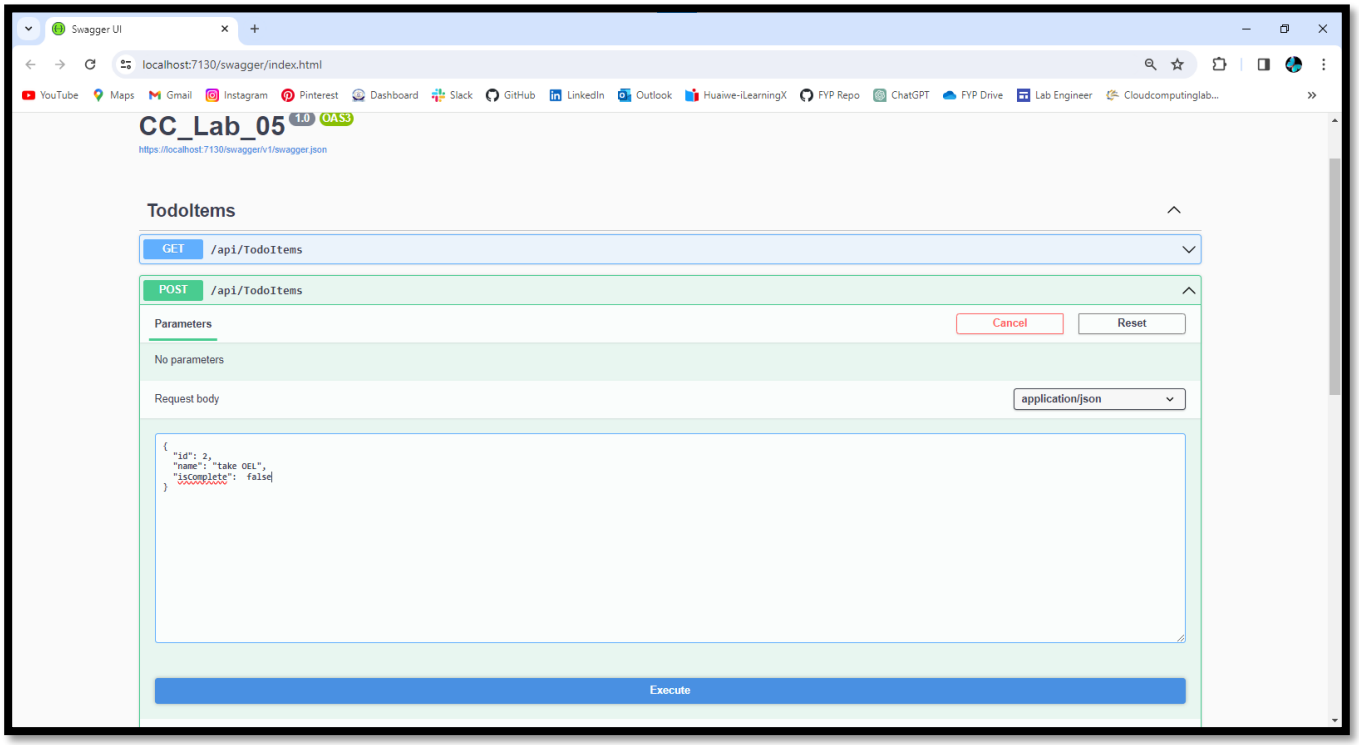


Step 9: Run the Project and then test all the methods API

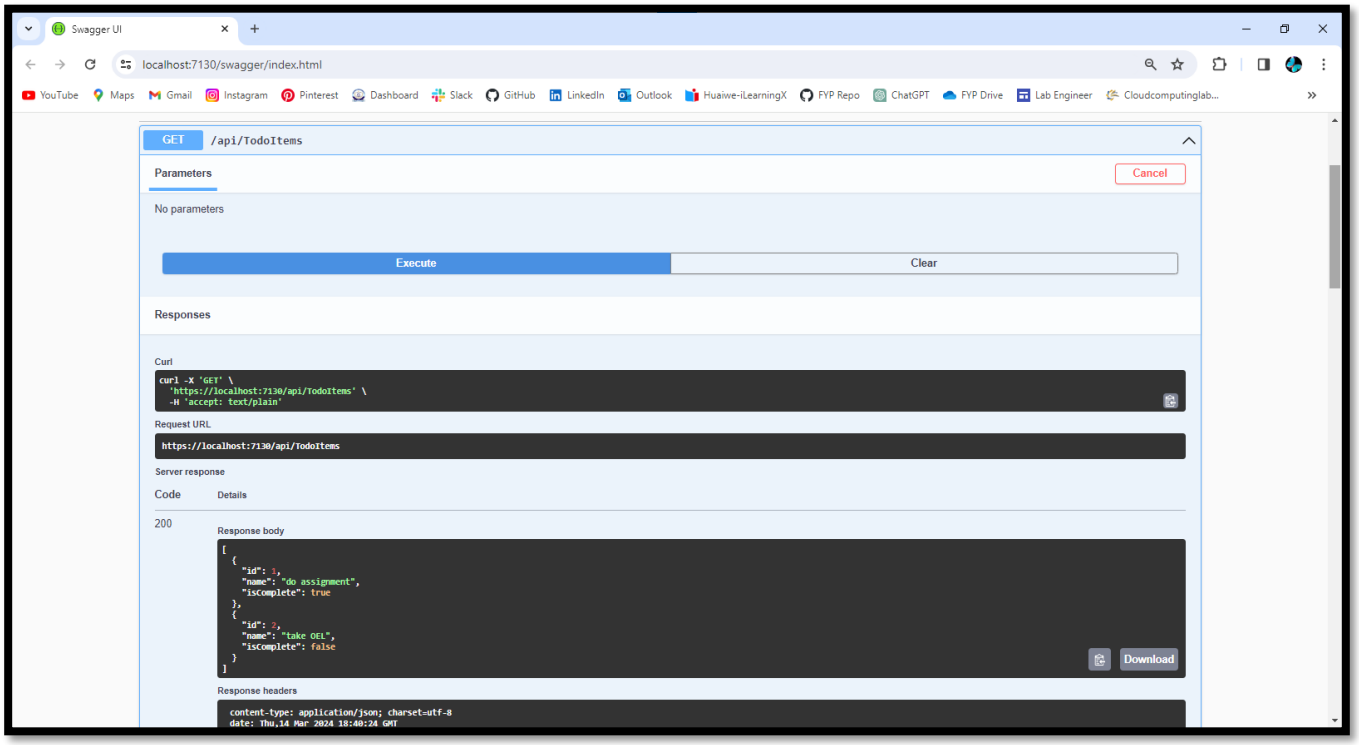


- Testing POST Method:

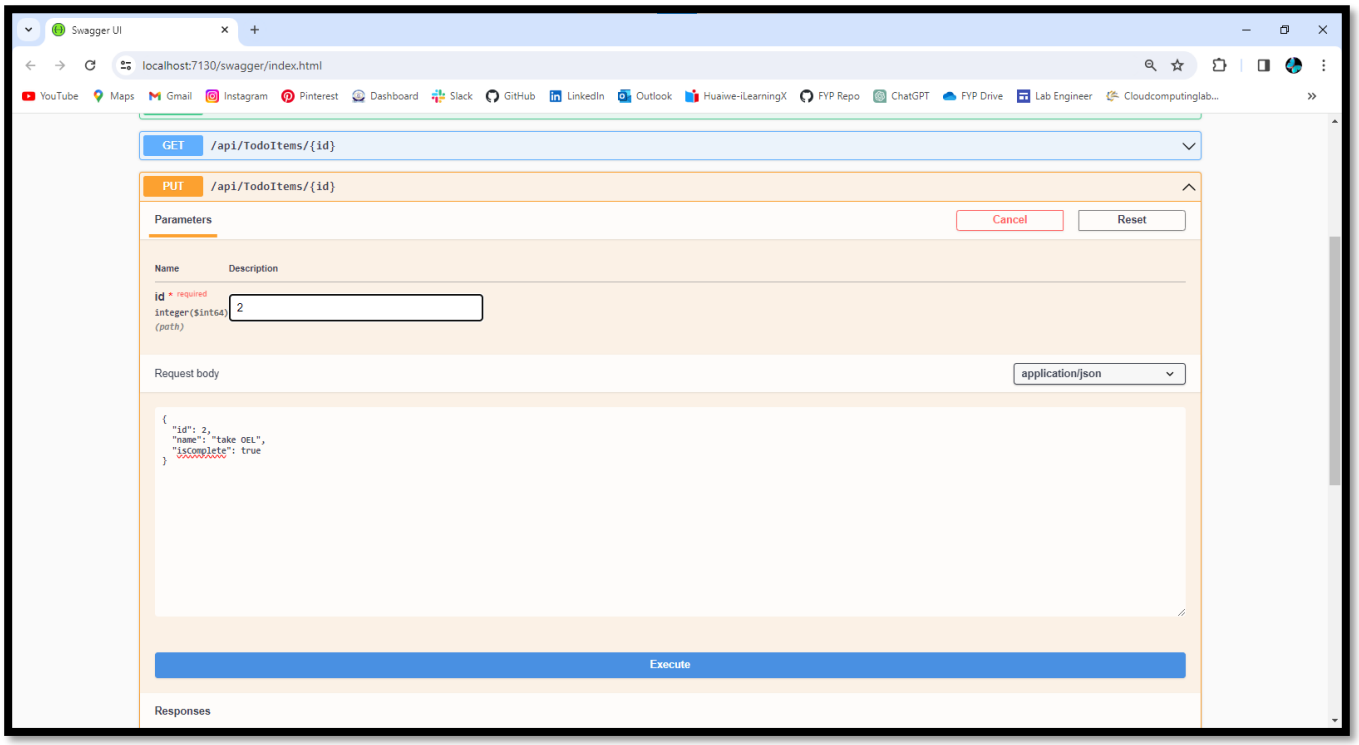




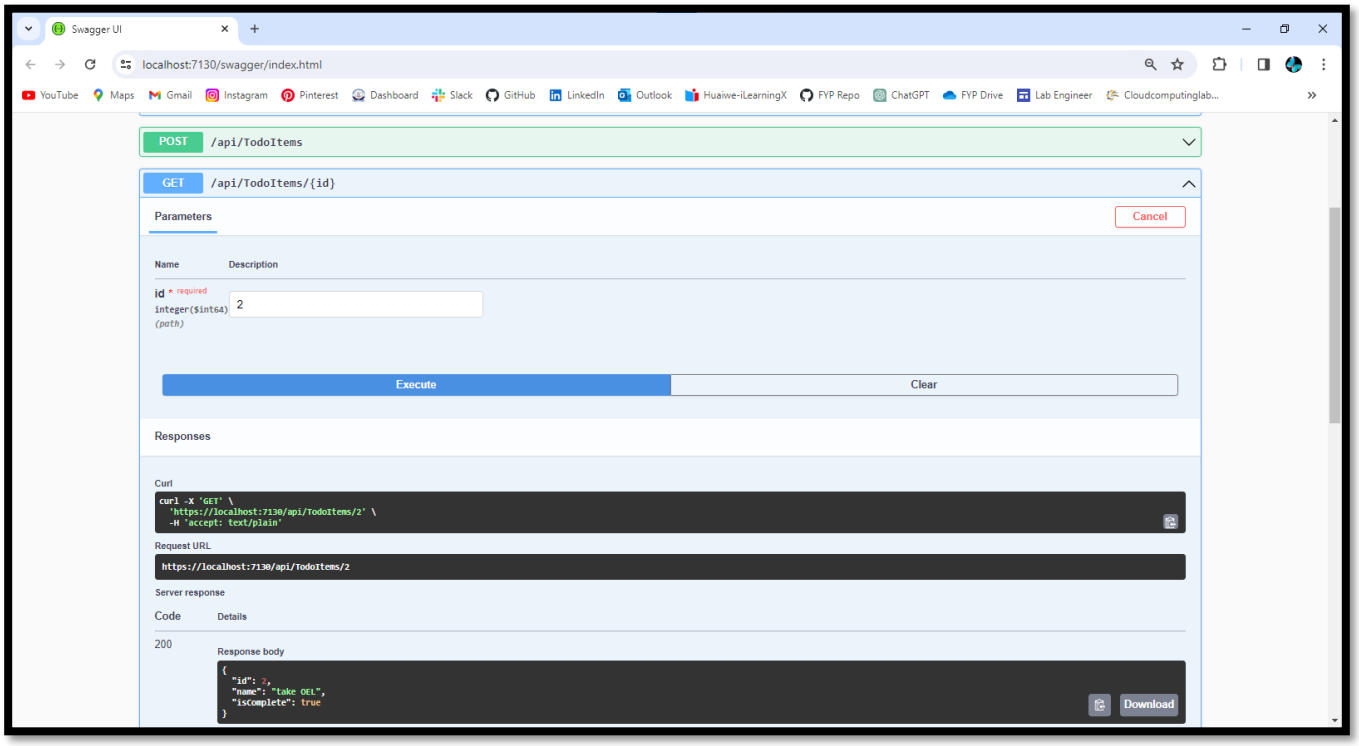
- Testing GET Method:



- Testing PUT Method:



- Testing GET (by id) Method:



2. Time Boxing

Activity Name	Activity Time	Total Time
Login Systems + Setting up Visual studio Environment	3 mints + 5 mints	8 mints
Walk through Theory & Tasks	60 mints	60 mints
Implement Tasks	80 mints	80 mints
Evaluation Time	30 mints	30 mints
	Total Duration	178 mints

3. Objectives

After completing this lab the student should be able to:

a. *Understand the concept of APIs and their role in facilitating communication between different software applications.*

b. *Create basic ASP.NET Core Web API endpoints to handle HTTP requests and responses.*

c. *Gain insights into the fundamental principles of ASP.NET Core Web API development, including routing, controllers, and data serialization.*

4. Lab Tasks/Practical Work

1. Design and implement an ASP.NET Core Web API with endpoints for GET (retrieve products), POST (add products), PUT (update products), and DELETE (remove products) methods to manage a product inventory system.

2. Create an ASP.NET Core Web API for managing student records, enabling CRUD operations (GET, POST, PUT, and DELETE) for adding, retrieving, updating, and deleting student information.