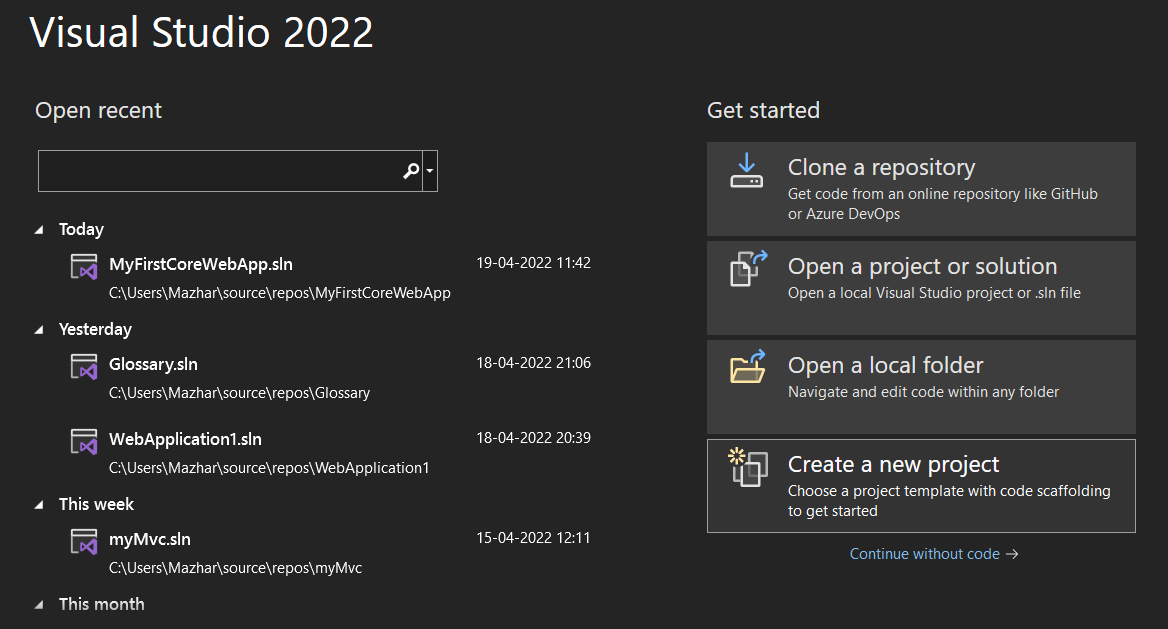
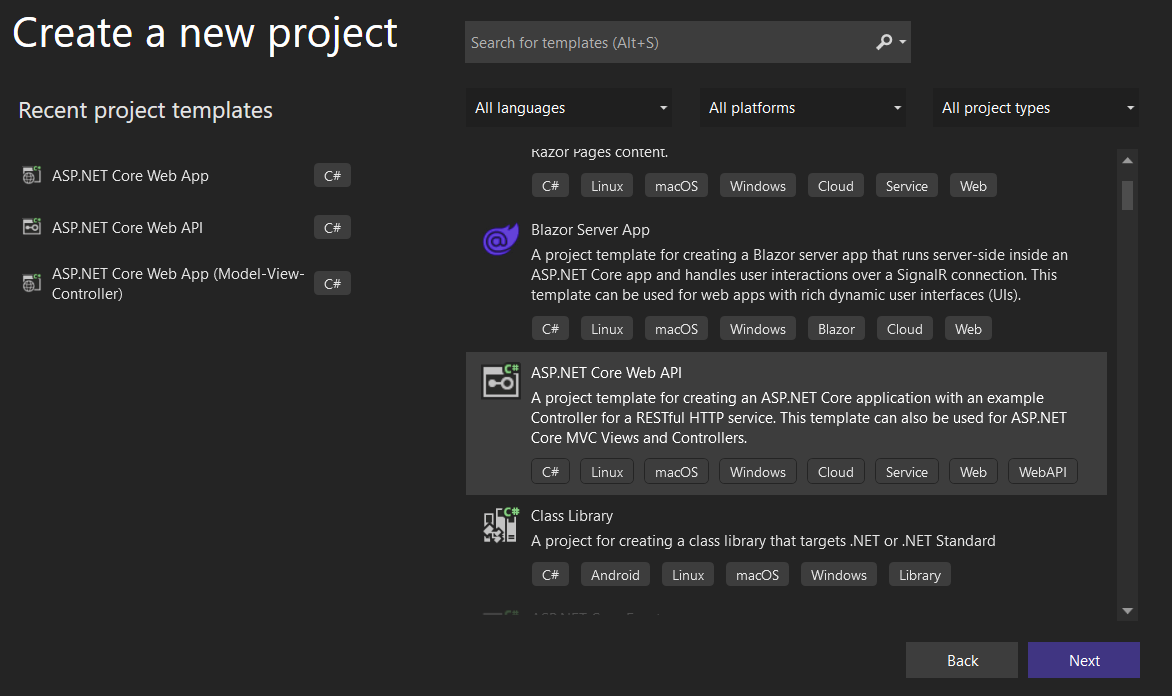
**Practical number 5**

**Creating ASP.NET Core Web API Project Using Visual Studio 2022**

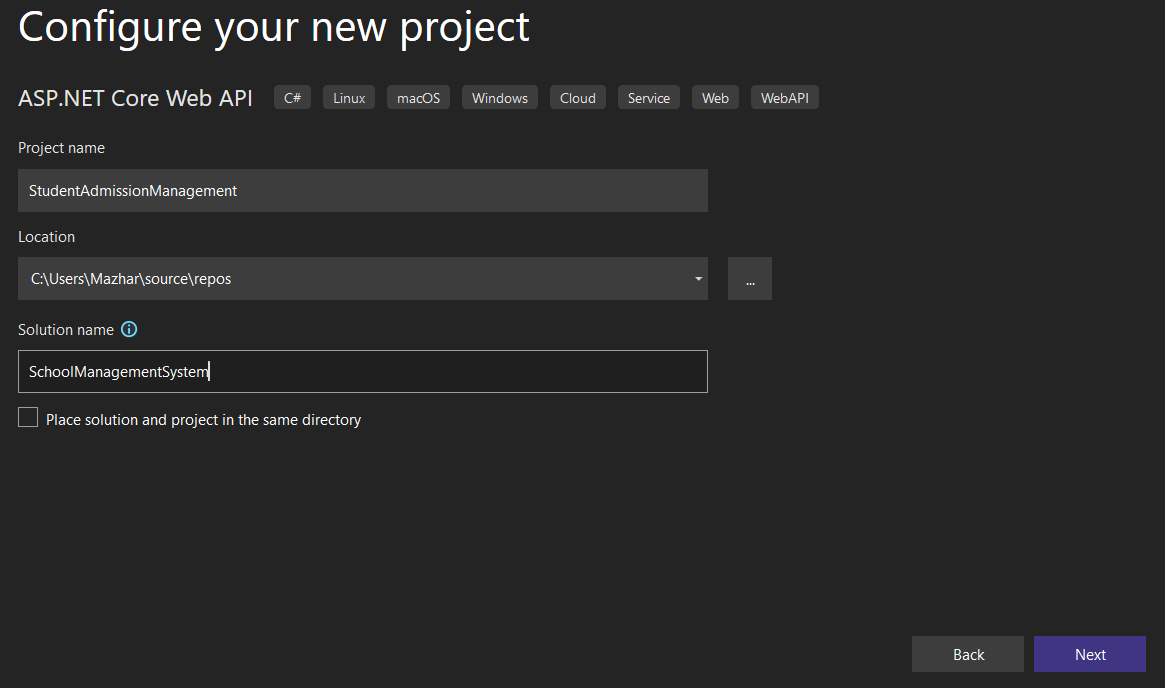
Now, we are going to create the ASP.NET Core Web API project using Visual Studio 2022. So, open Visual Studio 2022 and then click on the Create a new project option as shown in the below image.



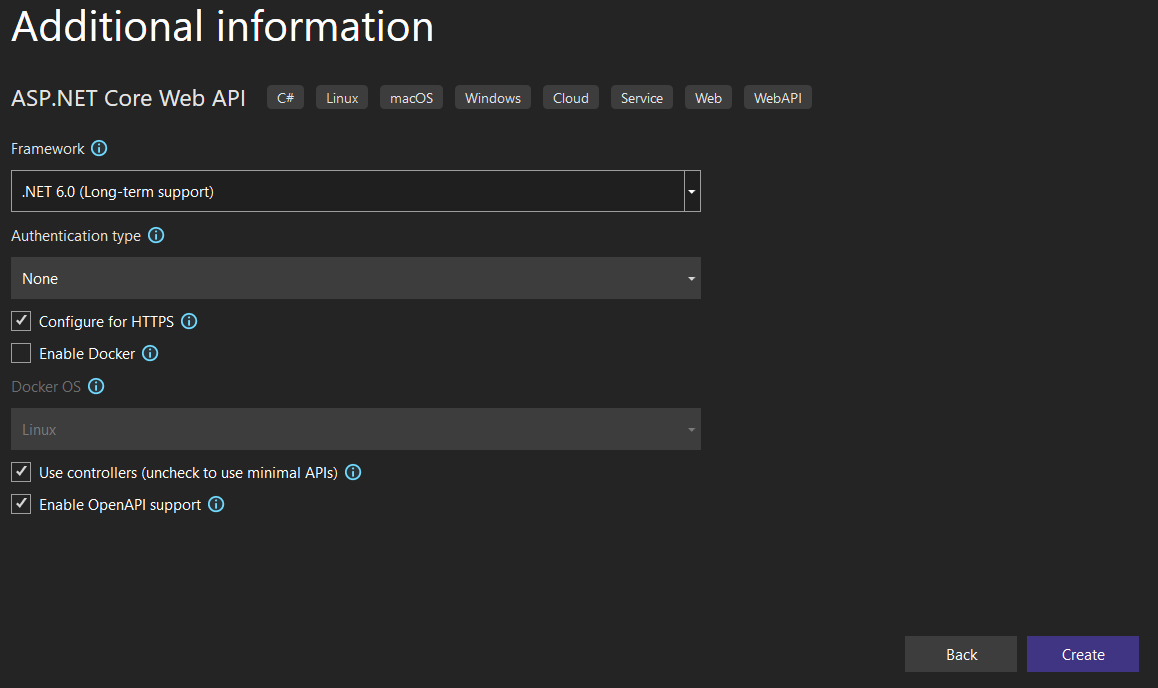
Once you click on the Create a new project option, the following Create a new project window will open. Here, you can find two projects template for creating the **ASP.NET Core Web API**project. One is using C# language and the other one is using F# language. I am going to use C# as the programming language, so I select the project template which uses C# Language as shown in the below image.



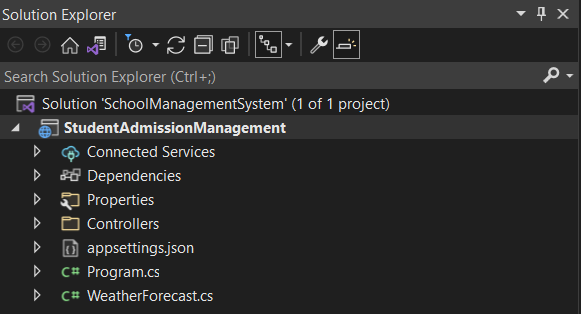
Once you click on the **Next** button, then the configure your new project window will open. Here, you need to specify the Project name Solution name, and the location where you want to create the project. I am providing the Project name as StudentAdmissionManagement, solution name as SchoolManagementSystem, and finally, click on the **Next**button as shown in the below image.



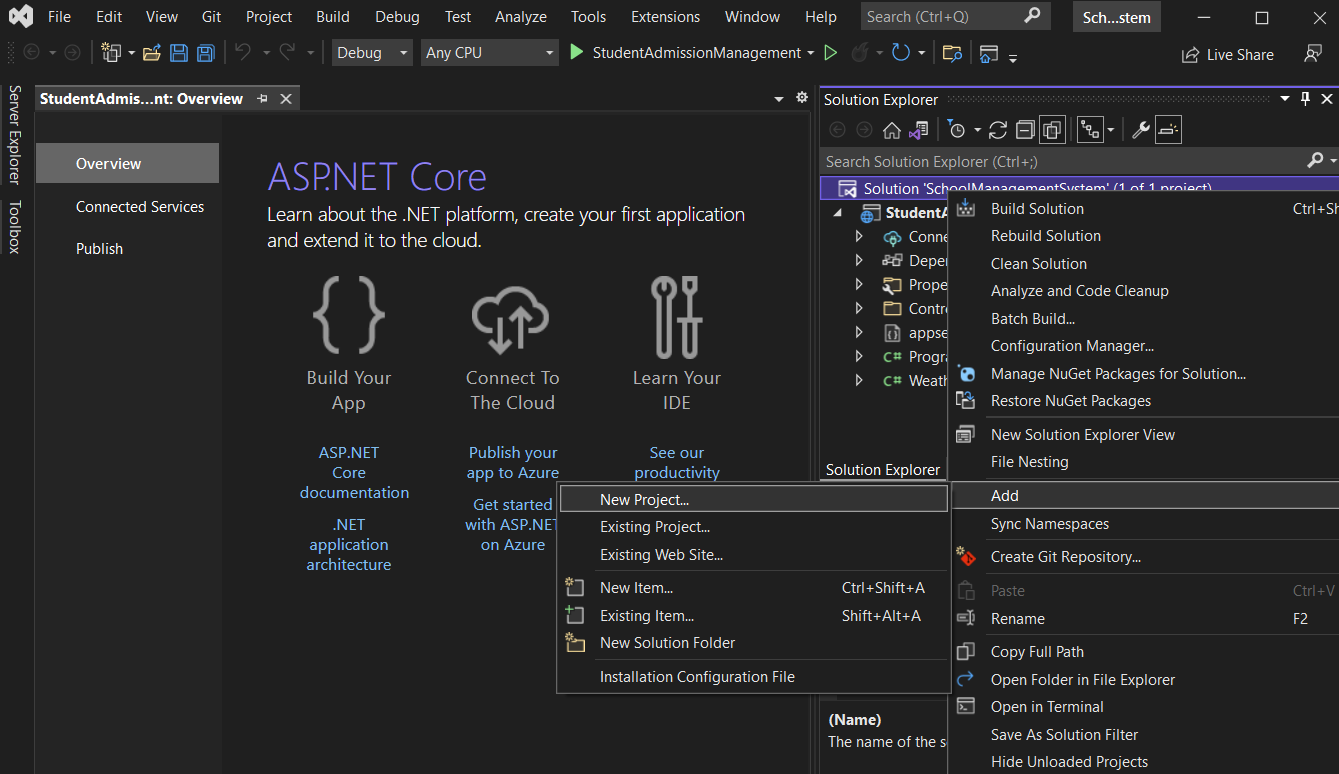
Once you click on the Next button, it will open the Additional Information window. Here, you need to select the Target .NET Framework version. The authentication Types. Whether you want to configure HTTPS and enable Docker. Select .NET Core 3.1 for now because it is having long Term support from Microsoft, select authentication type as None, check the Configure for HTTPS and uncheck the Enable Docker checkboxes and then click on the Create button as shown in the below image.



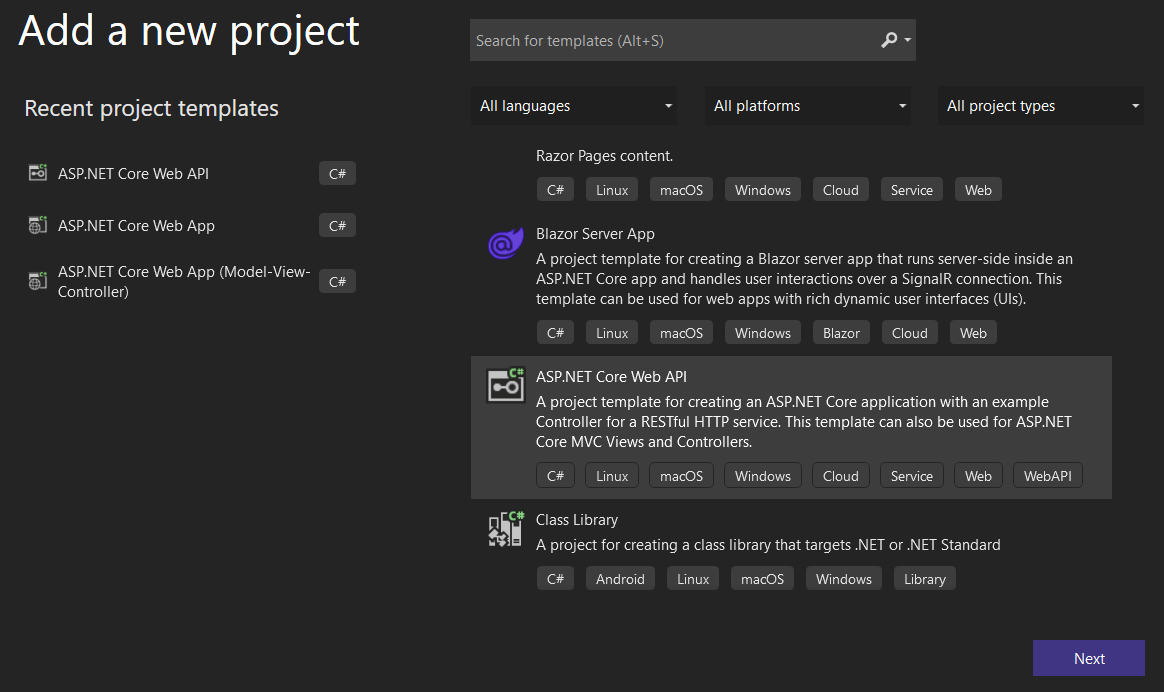
Once you click on the Create button, it will create the ASP.NET Core Web API project with the following file and folder structure. Initially, I have created only one separate project for student admission purposes which can be considered as a single microservice that works for student admission.



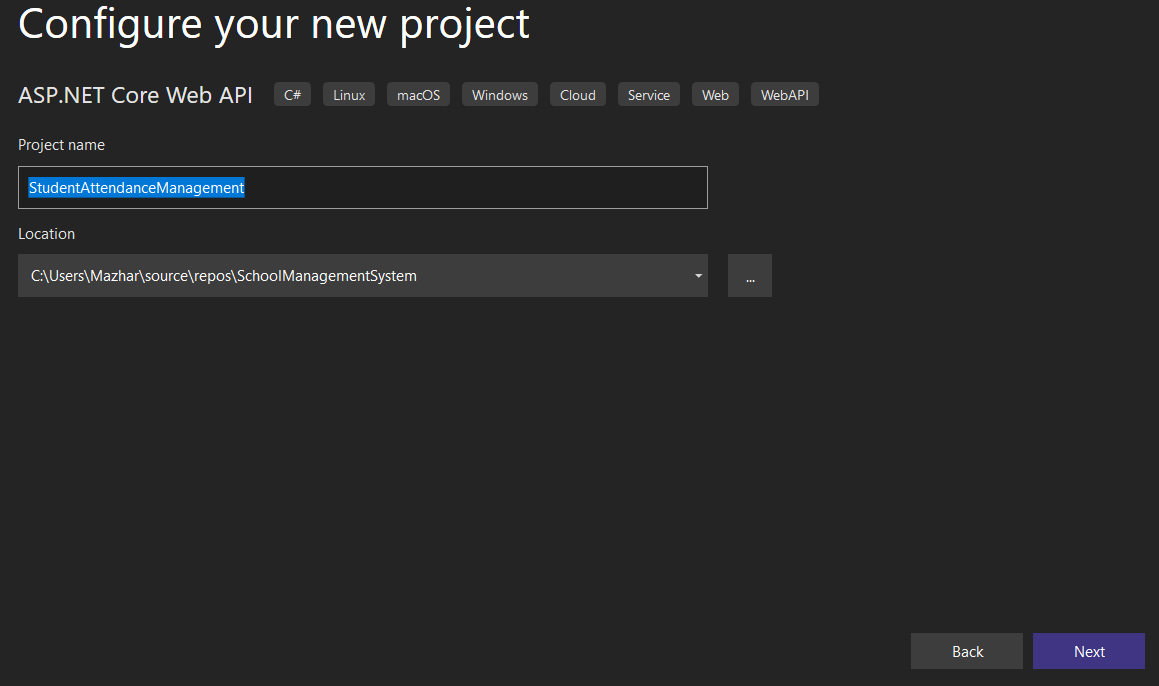
Let us create one more project under the solution which is another microservice for purpose of student attendance. To do so, right-click on the solution and then select Add => New Project option from the context menu as shown in the below image.



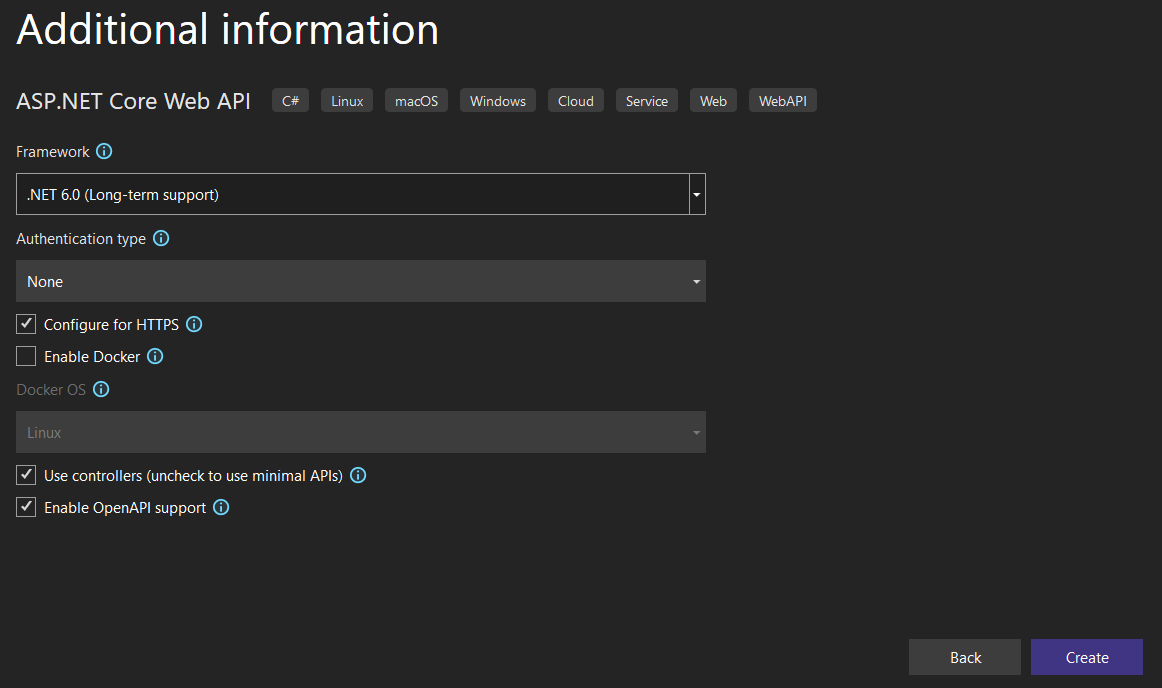
Once you click on the Add => New Project option, it will open the Add New Project window. From this window, select ASP.NET Core Web API (which uses C# language) and click on the Next button as shown in the below image.



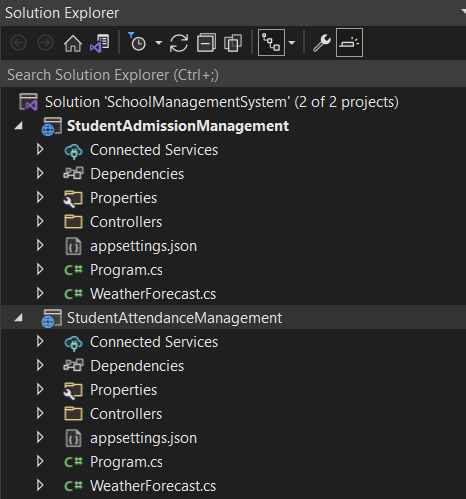
Once you click on the Next button, it will open Configure your new project window. From this window, provide the project name as StudentAttendanceManagement and click on the Next button as shown in the below image.



Once you click on the Next button, it will open the Additional Information window. Here, Select .NET Core 3.1 as Target Framework, select authentication type as None, check the Configure for HTTPS and uncheck the Enable Docker checkboxes and then click on the Create button as shown in the below image.



Once you click on the Create button, then it will add the new project to the existing solution. Now, our solution containing two projects with the following file and folder structure.



Now we have two microservices defined one for Student Admission purposes and the other for Student Attendance purposes.

**Creating Models:**

Now let’s create a model class to hold details of Admission and Attendance in both the projects respectively and create a CRUD operation Controllers in each of project respectively under the Controllers folder. So, right-click on the StudentAdmissionManagement project and add a class file with the name StudentAdmissionDetailsModel.cs and then copy and paste the following code in it.

namespace StudentAdmissionManagement

{

public class StudentAdmissionDetailsModel

{

public int StudentID { get; set; }

public string StudentName { get; set; }

public string StudentClass { get; set; }

public DateTime DateofJoining { get; set; }

}

}

Now, right-click on the StudentAttendanceManagement project and add a class file with the name StudentAttendanceDetailsModel.cs and then copy and paste the following code in it.

namespace StudentAttendanceManagement

{

public class StudentAttendanceDetailsModel

{

public int StudentID { get; set; }

public string StudentName { get; set; }

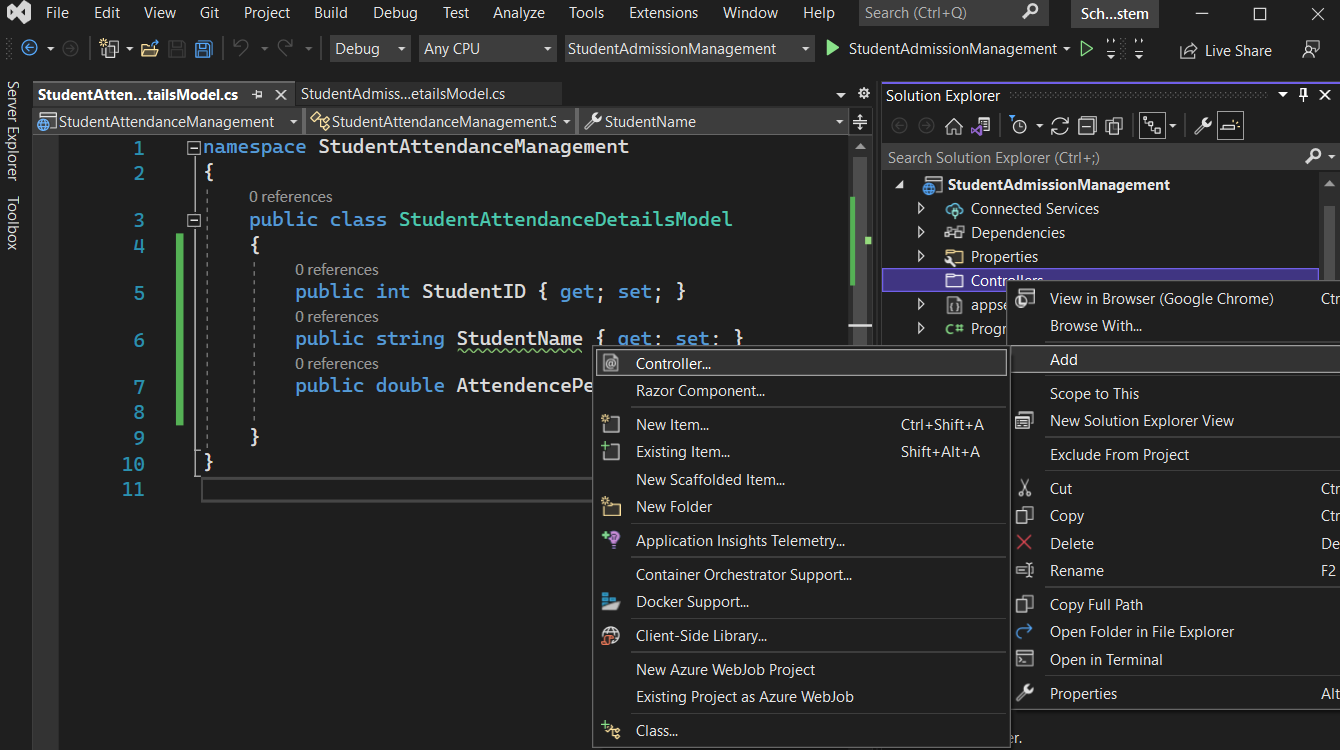
public double AttendencePercentage { get; set; }

}

}

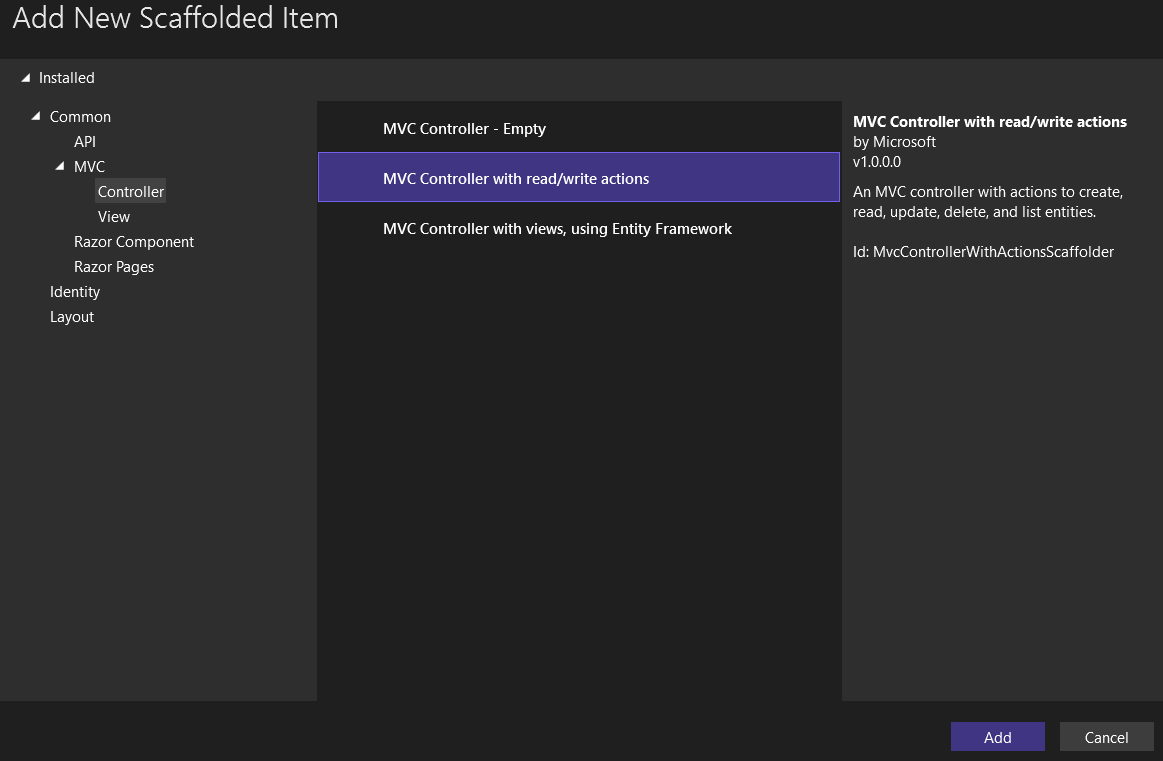
**Creating Controllers:**

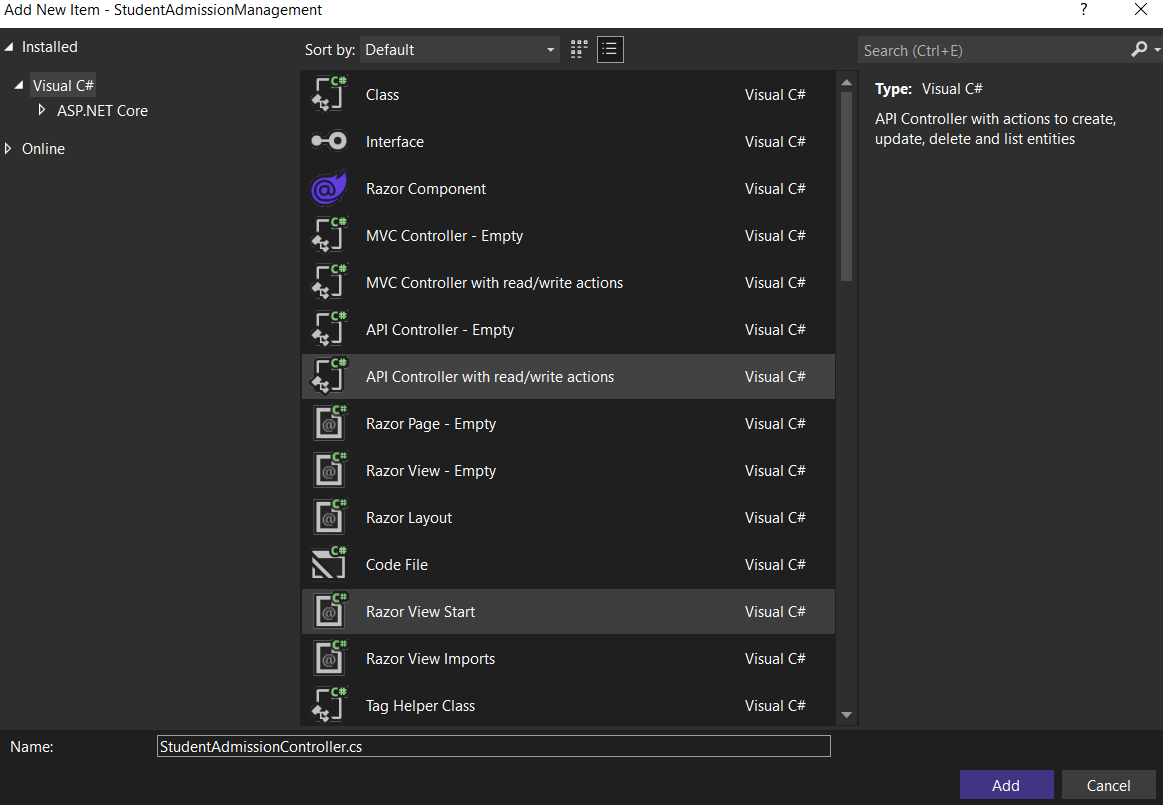
Now Let us create controllers which are endpoints for microservices for defining CRUD operations. Right-click on the Controllers folder of StudentAdmissionManagement and then select Add => Controller option from the context menu as shown in the below image.



Once you select the Add => Controller option, then it will open the Add New Scaffolded Item window. Here, first, select the API template and then select the Add Controller with read/write action template and click on the Add button as shown in the below image.

Once you click on the Add button, from the next window provide the name for your controller. Here, I am providing the name as StudentAdmissionController and click on the Add button as shown in the below image.





Once you click on the Add button, then it will add the StudentAdmissionController within the Controllers folder of your StudentAdmissionManagement project.

For the demonstration purpose, we create a GET endpoint with details of two students which will return details of two students to client request over HTTP GET. So, modify the StudentAdmissionController as follows.

using Microsoft.AspNetCore.Mvc;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace StudentAdmissionManagement.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class StudentAdmissionController : ControllerBase

{

// GET: api/<StudentAdmissionController>

[HttpGet]

public IEnumerable<StudentAdmissionDetailsModel> Get()

{

StudentAdmissionDetailsModel admissionobj1 = new StudentAdmissionDetailsModel();

StudentAdmissionDetailsModel admissionobj2 = new StudentAdmissionDetailsModel();

admissionobj1.StudentID = 1;

admissionobj1.StudentName = "Adam";

admissionobj1.StudentClass = "X";

admissionobj1.DateofJoining = DateTime.Now;

admissionobj2.StudentID = 2;

admissionobj2.StudentName = "Brad";

admissionobj2.StudentClass = "IX";

admissionobj2.DateofJoining = DateTime.Now;

List<StudentAdmissionDetailsModel> listofobj = new List<StudentAdmissionDetailsModel>

{

admissionobj1,

admissionobj2

};

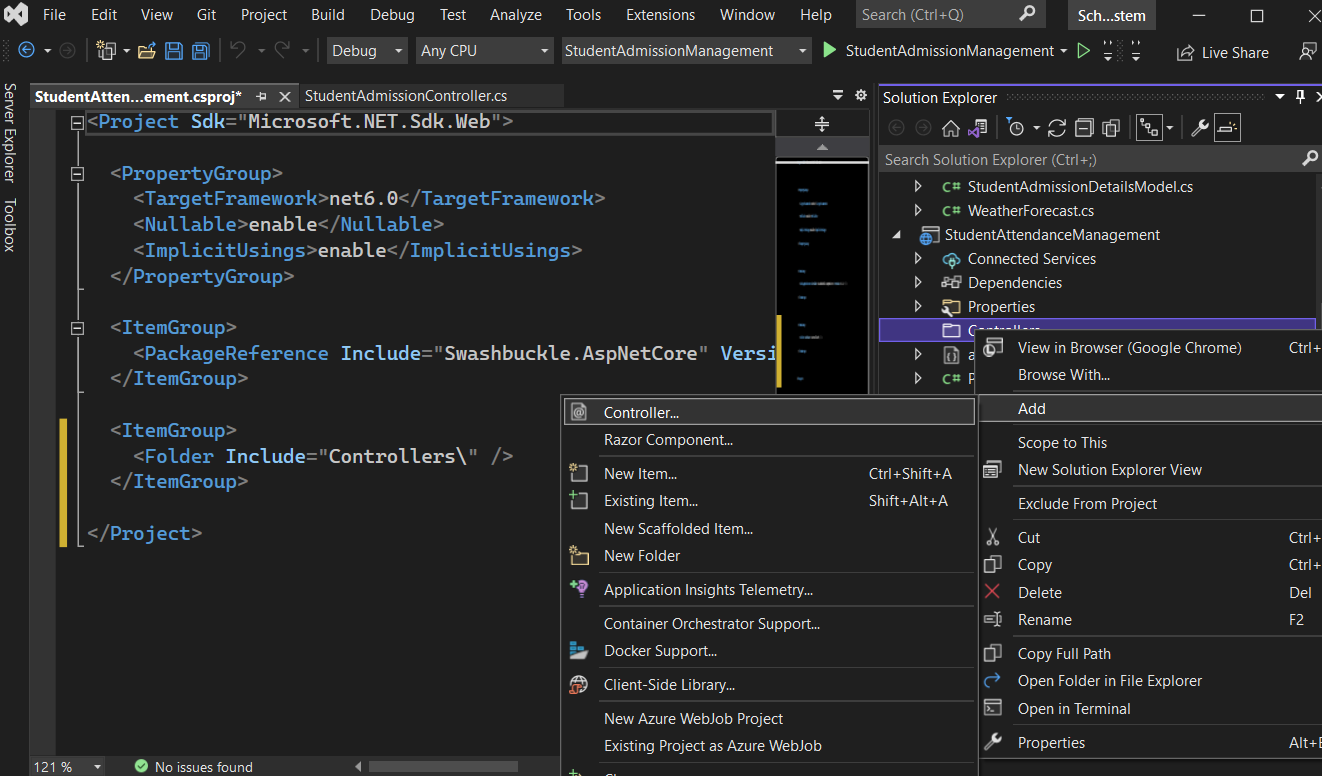
return listofobj;

}

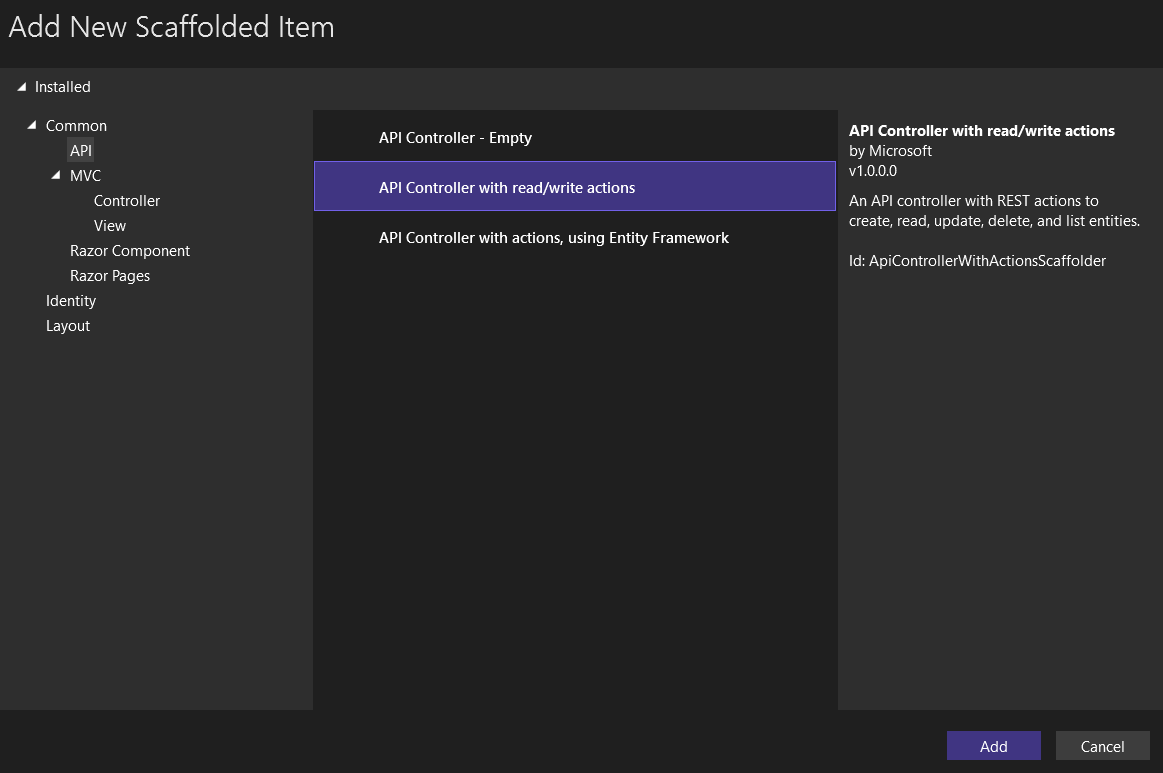
}

}

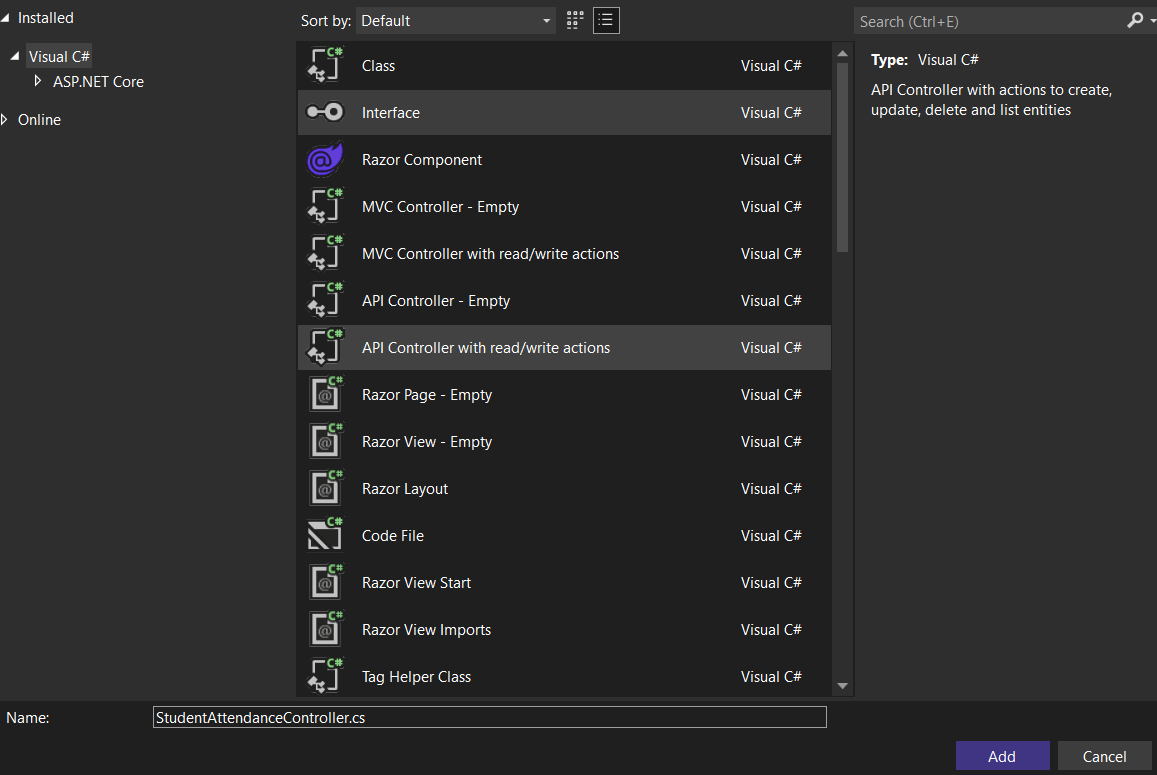
Now, Right-click on the Controllers folder of StudentAttendanceManagement and then select Add => Controller option from the context menu as shown in the below image.



Once you select the Add => Controller option, then it will open the Add New Scaffolded Item window. Here, first, select the API template and then select the Add Controller with read/write actions and click on the Add button as shown in the below image.



Once you click on the Add button, from the next window provide the name for your controller. Here, I am providing the name as StudentAttendanceController and click on the Add button as shown in the below image.



Once you click on the Add button, then it will add the StudentAttendanceController within the Controllers folder of your StudentAttendanceManagement project. For the demonstration purpose, we create a simple Get method for returning the two students’ attendance percentage on HTTP GET requests. So, modify the StudentAttendanceController as follows.

using Microsoft.AspNetCore.Mvc;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace StudentAttendanceManagement.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class StudentAttendanceController : ControllerBase

{

// GET: api/<StudentAttendanceController>

[HttpGet]

public IEnumerable<StudentAttendanceController> Get()

{

StudentAttendanceDetailsModel attendanceObj1 = new StudentAttendanceDetailsModel();

StudentAttendanceDetailsModel attendanceObj2 = new StudentAttendanceDetailsModel();

attendanceObj1.StudentID = 1;

attendanceObj1.StudentName = "Adam";

attendanceObj1.AttendencePercentage = 83.02;

attendanceObj2.StudentID = 2;

attendanceObj2.StudentName = "Brad";

attendanceObj2.AttendencePercentage = 71.02;

List<StudentAttendanceDetailsModel> listObj = new List<StudentAttendanceDetailsModel>

{

attendanceObj1,

attendanceObj2

};

return listObj;

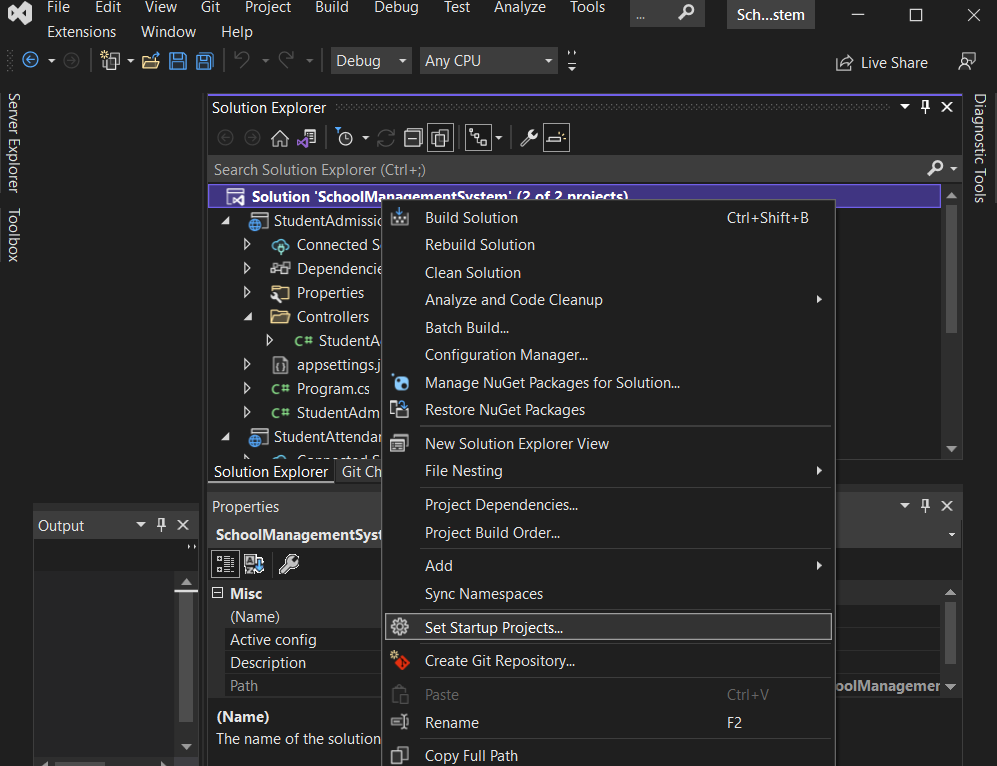
}

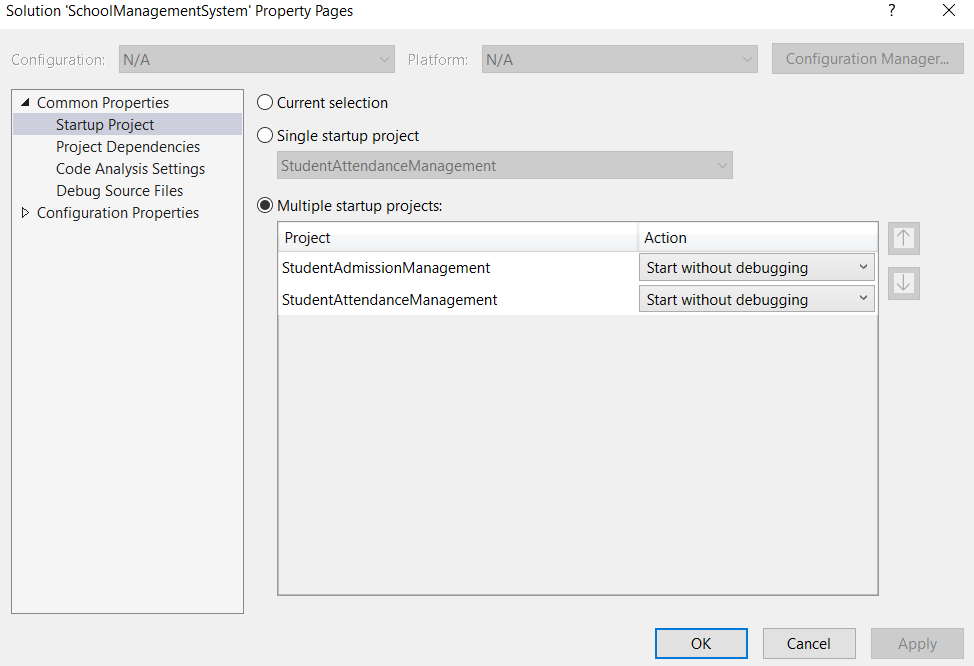
}

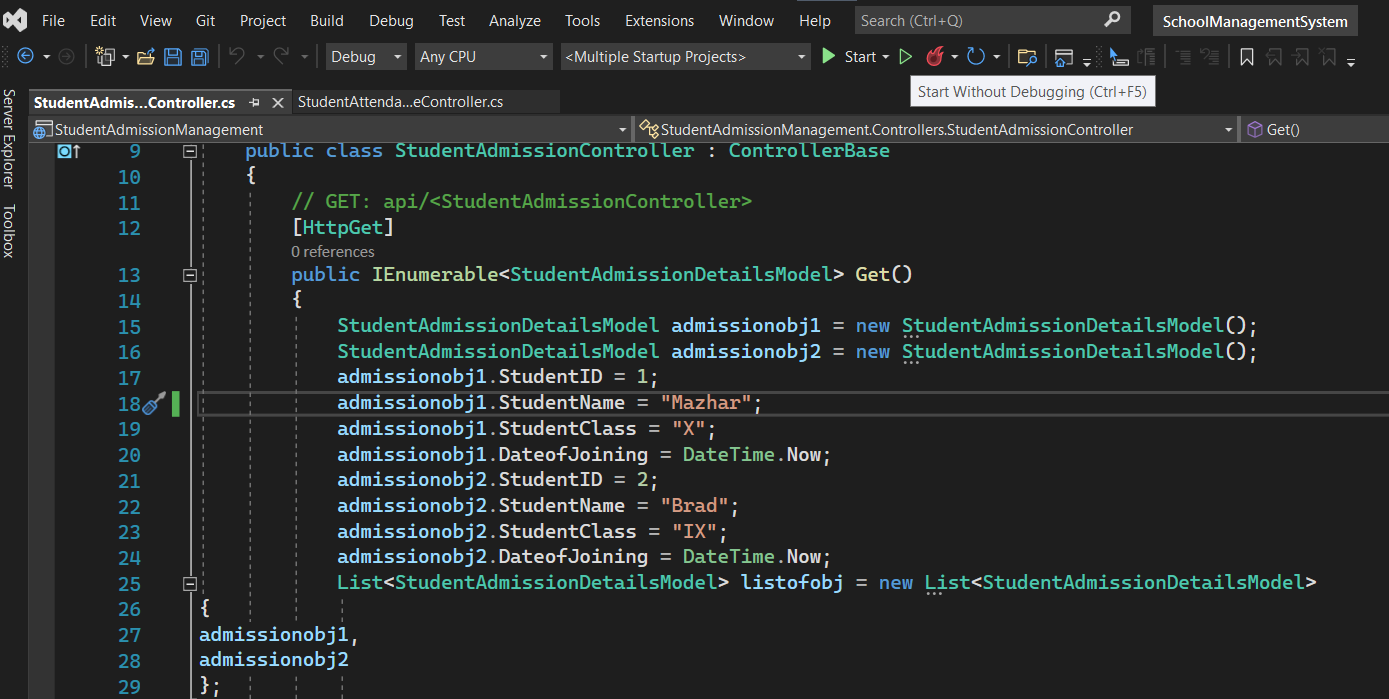
}

Let’s build the solution and launch both the Web API services (Microservices) simultaneously

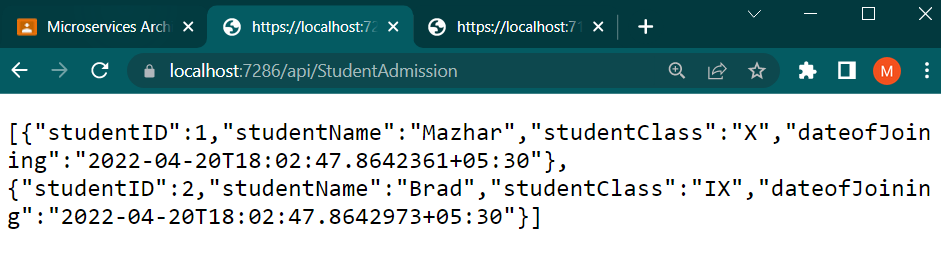
**Running Applications and Showing Outputs**

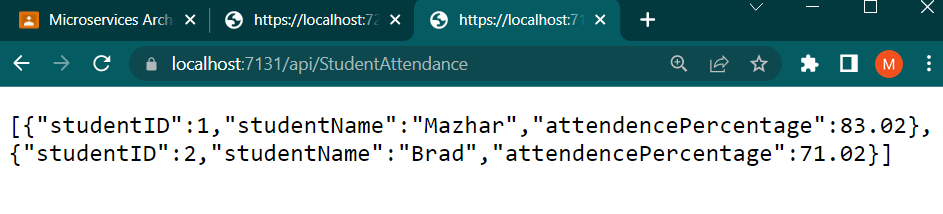
****

****



Click on the Start button and then we get two browsers opened and point to the controllers from browsers like below. Browser by default sends GET HTTP Verb.





They returned the details as expected. So, this is how microservices are developed and our two microservices are available to serve the user requests over HTTP request and response. To test Microservices with Postman application please refer to the Testing section provided at end of this learning path.

Till now we have observed the implementation of independent APIs (Microservices) which hold separate concerns of business functionalities, one is to get admission details of all the students and the other is to get the attendance percentage of all the students in examples. Which are directly exposed to the Client.