

## **Milestone-Assessment-6**

**Name:B Karthik**

**MID:M1082972**

**SourceCode:**

<https://github.com/BKarthik1/BKarthik1.git>

### **Question**

#### **Case study1**

#### **Milestone Assessment -6 Attempt-1**

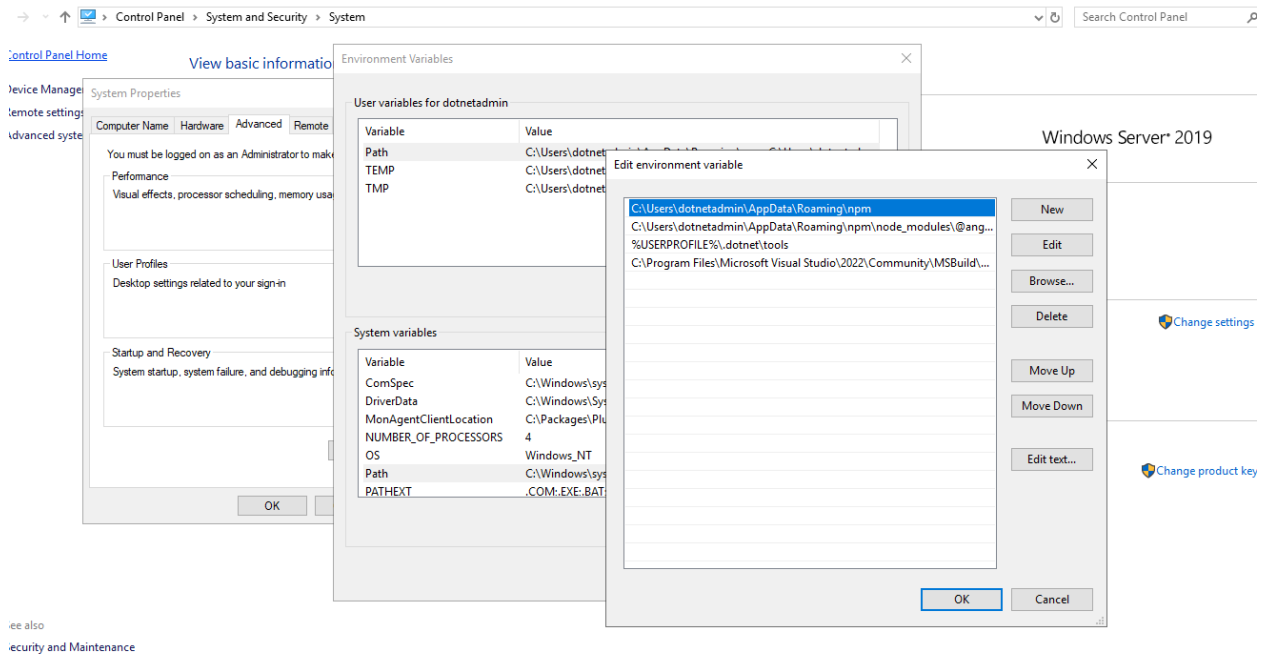
##### **Case Study 1 [1 Hour]**

**The following points should be considered while performing the case study**

1. Create a basic project to implement the given described system with appropriate schema structure and OOP concepts.
2. Follow best practices to be followed while coding
3. Use MSBuild to create the solution
4. Provide application path using settings.
5. Customize the necessary targets and use build conditions as applicable
6. Build and execute project using cmd.

#### **Answer for 1<sup>st</sup> Case Study**

##### **Providing path using settings**

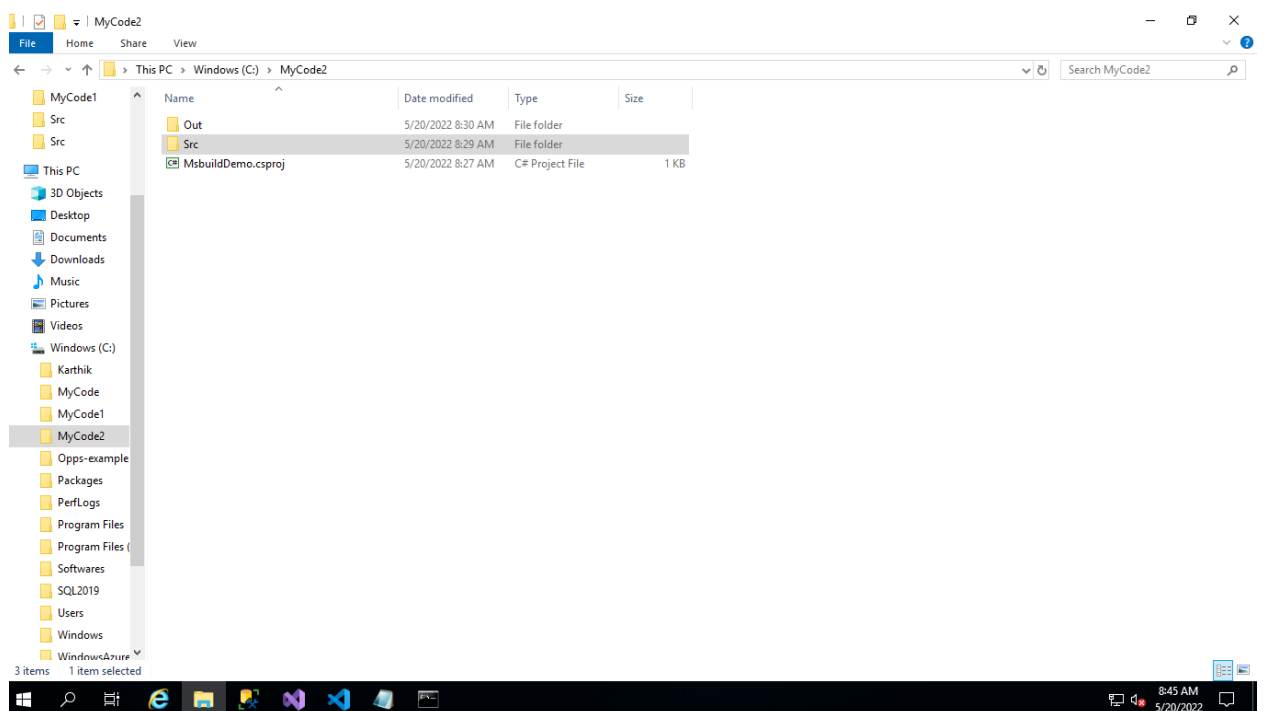


## Next Creating Folders

My folder name is MyCode2 in that I created

Src and Out

And creating two documents files



## The Code For Msbuild Path is

### Document name is MsbuildDemo.csproj

```
<Project InitialTargets="MsbuildDemo">
  <PropertyGroup>
    <Basepath>C:\MyCode2</Basepath>
    <Srcpath>$(Basepath)\Src</Srcpath>
    <Destpath>$(Basepath)\Out</Destpath>
  </PropertyGroup>
  <ItemGroup>
    <Srcfiles Include="C:\MyCode2\Src\Oops.cs"> </Srcfiles>

  </ItemGroup>
  <Target Name="EnvSetup">
    <MakeDir Directories="$(Srcpath)"> </MakeDir>
    <MakeDir Directories="$(Destpath)"> </MakeDir>
  </Target>
  <Target Name="MsbuildDemo" DependsOnTargets="EnvSetup">
    <Copy SourceFiles="@(\srcfiles)" DestinationFolder="$(Srcpath)"> </Copy>
    <Csc Sources="$(Srcpath)\*.cs" OutputAssembly="$(Destpath)\MsbuildDemo.exe"> </Csc>
  </Target>
</Project>
```

Next in Src folder under we have Create a Documentfile

Documentfile Name is Oops.cs

Write code on that

Code in c#

The Code is About Opps Concep in that Code I Showed That

1)Inheritance

2)Abstraction

3)Polymarphism

4)Encapsulation

So it Contains Opps Concepts

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
class Program
```

```
{
```

```
void print(int i, int j)
```

```
{
```

```
Console.WriteLine("Printing int: {0}", (i + j));
```

```
}
```

```
void print(string a, string b)
```

```
{
```

```
Console.WriteLine("Printing String " + a + b);
```

```
}
```

```
static void Main(string[] args)
```

```
{
```

```
Program prog = new Program();
```

```
// Call print for sum of integers
```

```
prog.print(9, 11);
```

```
// Call to concatenate strings
```

```
prog.print("Hello", "World");
```

```
College hyn = new College();  
String desc = hyn.Describe();
```

```
Console.WriteLine(desc);
```

```
Console.ReadKey();
```

```
}
```

```
}
```

```
abstract class Student
```

```
{
```

```
public virtual string Describe()
```

```
{
```

```
return "Description of the Student";
```

```
}
```

```
}
```

```
class College : Student
```

```
{
```

```
public override string Describe()
```

```
{
```

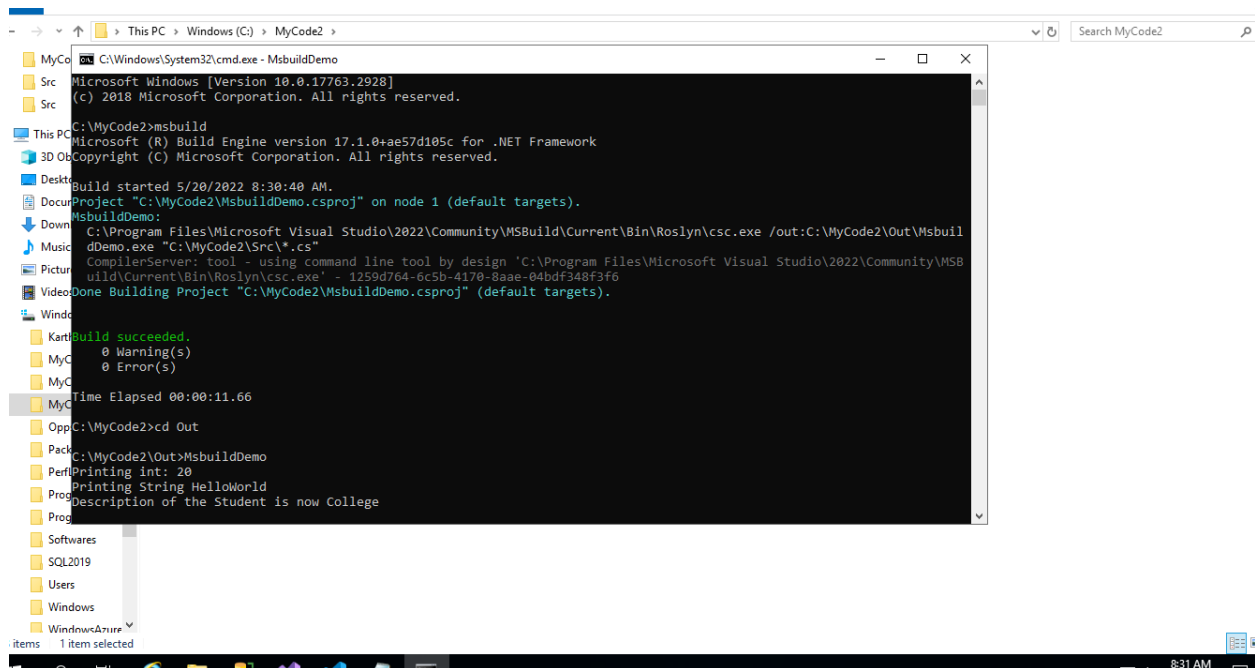
```
return "Description of the Student is now College";
```

```
}
```

```
}
```

## Output

I Run That Code in Command Prompt  
And its runned Successfully



```
C:\Windows\System32\cmd.exe - MsbuildDemo
Microsoft Windows [Version 10.0.17763.2928]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\MyCode2>msbuild
Microsoft (R) Build Engine version 17.1.0+ae57d105c for .NET Framework
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 5/20/2022 8:30:40 AM.
Project "C:\MyCode2\MsbuildDemo.csproj" on node 1 (default targets).
  MsbuildDemo:
    C:\Program Files\Microsoft Visual Studio\2022\Community\MSBuild\Current\Bin\Roslyn\csc.exe /out:C:\MyCode2\Out\MsbuildDemo.exe "C:\MyCode2\Src\*.cs"
    CompilerServer: tool - using command line tool by design 'C:\Program Files\Microsoft Visual Studio\2022\Community\MSBuild\Current\Bin\Roslyn\csc.exe' - 1259d764-6c5b-4170-8aae-04bdf348f3f6
  Done Building Project "C:\MyCode2\MsbuildDemo.csproj" (default targets).

Build succeeded.
    0 Warning(s)
    0 Error(s)
Time Elapsed 00:00:11.66
C:\MyCode2>cd Out
C:\MyCode2\Out>MsbuildDemo
Printing int: 20
Printing String HelloWorld
Description of the Student is now College
```

For CaseStudy2

Application should manage the bugs, assigned in the project, with attributes: BugId, Title, Description, module name, Owner and Severity. User should able to add new bug to the collection and update a bug in the collection based on BugId.

#### Case Study 2 [30 Mins]

Attach appropriate screenshots wherever needed, as the screenshots will be used for evaluation if the application is unable to open.

Design and develop application for online movie ticket booking system. The following are the requirements shared by the CLIENT. Identify the major requirements and divide them appropriately.

#### In Manage Screen Module:

a. Headers, Top Menu, page title and information present on the page should be available appropriately.

Downloaded by KPF from online Internet files

## UserStories

A) EPIC:

User Registration to a page for booking a ticket(or)Screen

USER STORIES/TASKS:Assigned to Bhavana

<As a new User>

<I want to register myself with my details including username and password >

<So that the system can easily store my information for next time so i can login easily >

SUBTASKS:

1. user should give details along with password and username
2. Add the Register button
3. Add an option 'AlreadyanUser' so the user can directly login into page as the details were already stored.

B) EPIC:

User Login to the page

USER STORIES/TASKS:Assigned to Karthik

<As a User>

<I want to login to the page with my username and password only>

<So that i dont have to give all my details>

SUBTASKS:

1. Add Username box
2. Add password box
3. Add Login Button so that user can login to page by this two details

C) EPIC:

Searching for the movie in the search box based on Location and Screeing name

USER STORIES/TASKS:Assigned to Chandu

<As a User>

<I want to locate to search bar and have option like city and screen name to be select as per choice>

<so that i can easily select my prefered city and screenname instead of searching all of the Menu page>

SUBTASKS:

1. Add options City and ScreenName to the search bar
2. Add city names in it
3. Add Screen Names in it so user can easily select



D) EPIC:  
BOOKING TICKET

USER STORIES/TASKS:Assigned to Shiva  
<As a User>  
<I want to select my prefer movie with my preferable date and time>  
<So that i am available at the time or date>

SUBTASKS:  
1. enable the user to select his preferred movie by giving select option rightside of the screen name  
2. Add Calender and TimeLine so that they can book there wanted date

E) EPIC:  
CANCELLING THE TICKET

USER STORIES/TASKS:Assigned to Sravan  
<As a user>  
< I Want to have a option like cancelling the movie ticket before the booked date>  
<So that i can cancel my show if i am not available>

SUBTASKS:  
1.Add Cancel Booking in My showList  
2.Make the date to be visible one day prior to the booked date to get cancelled

change assigned names

this is for only ticket module

For Admin

1. Log-in

Similar to users, admins also require to log-in. So, provide your admin panel a log-in access to control events.

2. Dashboard

The dashboard is a crucial feature for your online ticket booking app. It provides insights into your business, app performance, and more. Thus, you can make informed decisions.

### 3. Management

As an admin, you are entitled with a number of responsibilities, including the management of shows/events, users, bookings, customers, and earnings. So, you need to define separate sections for managing each aspect.

### 4. Notifications

This adds a touch of personalization to the user experience you offer. Admins must be given the rights to send notifications to users – upcoming shows/events, app update, etc.

### Reviews and Feedback

Businesses can capture user feedback and reviews, and use them to make the needed improvements to provide better service to customers.

## Case Study-3

### **Case Study 3 [1 Hour]**

**Attach appropriate screenshots wherever needed, as the screenshots will be used for evaluation if the application is unable to open.**

Create a Web API in .net core project with EF for College Management System. There is only one module in this App.

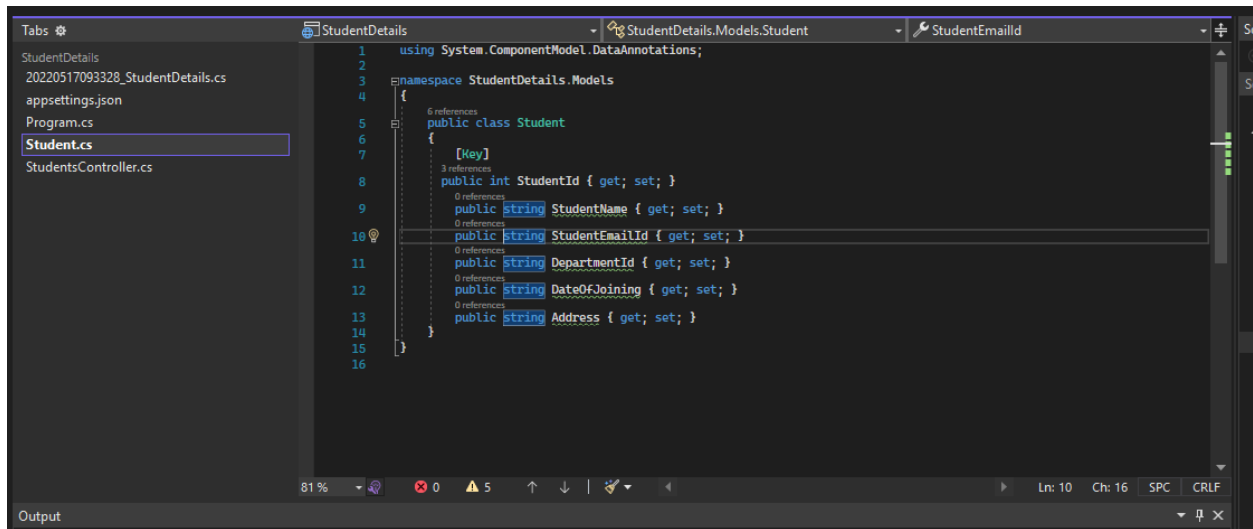
**Student Management Module:** It contains two functionalities: Display all Students, Add new Student.

Using MS Test framework, write all the test cases for the application and use MOQ wherever required and show the reports.

Use Git command to push, pull, resolve merge conflict (if any).

Show the usage of create a new branch, rename, move commands for the above app.

**Here I created Web Api Student Management details**



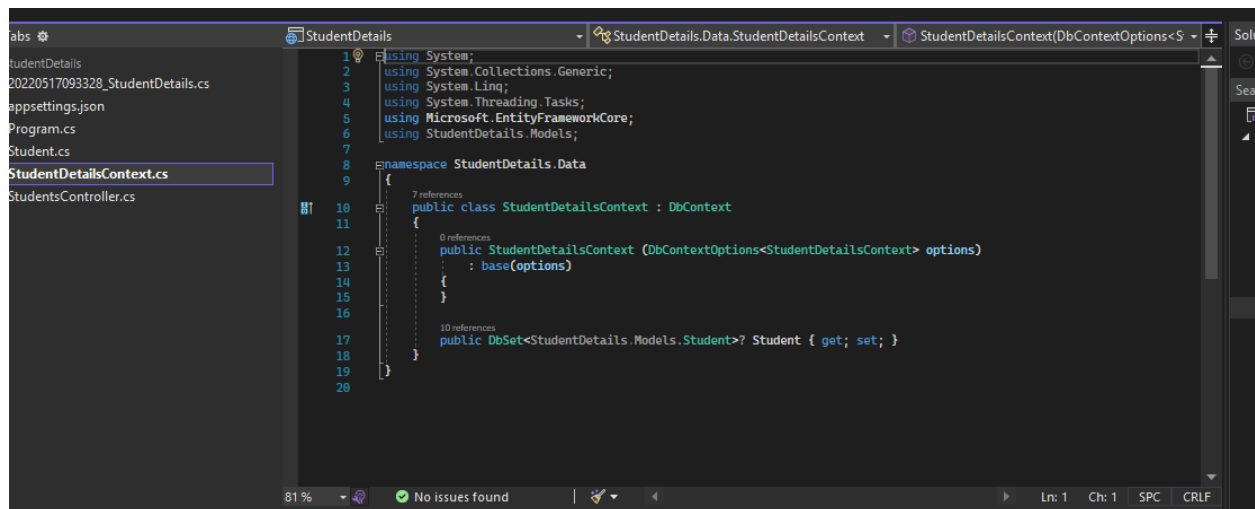
## Model Part code

```
using System.ComponentModel.DataAnnotations;
```

```
namespace StudentDetails.Models
```

```
{
    public class Student
    {
        [Key]
        public int StudentId { get; set; }
        public string StudentName { get; set; }
        public string StudentEmailId { get; set; }
        public string DepartmentId { get; set; }
        public string DateOfJoining { get; set; }
        public string Address { get; set; }
    }
}
```

## Next Create Data folder



## Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using StudentDetails.Models;

namespace StudentDetails.Data
{
    public class StudentDetailsContext : DbContext
    {
        public StudentDetailsContext (DbContextOptions<StudentDetailsContext> options)
            : base(options)
        {
        }

        public DbSet<StudentDetails.Models.Student>? Student { get; set; }
    }
}
```

Next we have to create Controller

## Code

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StudentDetails.Data;
using StudentDetails.Models;
```

```

namespace StudentDetails.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class StudentsController : ControllerBase
    {
        private readonly StudentDetailsContext _context;

        public StudentsController(StudentDetailsContext context)
        {
            _context = context;
        }

        // GET: api/Students
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Student>>> GetStudent()
        {
            if (_context.Student == null)
            {
                return NotFound();
            }
            return await _context.Student.ToListAsync();
        }

        // GET: api/Students/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Student>> GetStudent(int id)
        {
            if (_context.Student == null)
            {
                return NotFound();
            }
            var student = await _context.Student.FindAsync(id);

            if (student == null)
            {
                return NotFound();
            }

            return student;
        }

        // PUT: api/Students/5
        // To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPut("{id}")]
        public async Task<ActionResult> PutStudent(int id, Student student)
        {
            if (id != student.StudentId)
            {
                return BadRequest();
            }

            _context.Entry(student).State = EntityState.Modified;

```

```

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!StudentExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return NoContent();
}

// POST: api/Students
// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
[HttpPost]
public async Task<ActionResult<Student>> PostStudent(Student student)
{
    if (_context.Student == null)
    {
        return Problem("Entity set 'StudentDetailsContext.Student' is null.");
    }
    _context.Student.Add(student);
    await _context.SaveChangesAsync();

    return CreatedAtAction("GetStudent", new { id = student.StudentId }, student);
}

// DELETE: api/Students/5
[HttpDelete("{id}")]
public async Task<ActionResult> DeleteStudent(int id)
{
    if (_context.Student == null)
    {
        return NotFound();
    }
    var student = await _context.Student.FindAsync(id);
    if (student == null)
    {
        return NotFound();
    }

    _context.Student.Remove(student);
    await _context.SaveChangesAsync();

    return NoContent();
}

private bool StudentExists(int id)

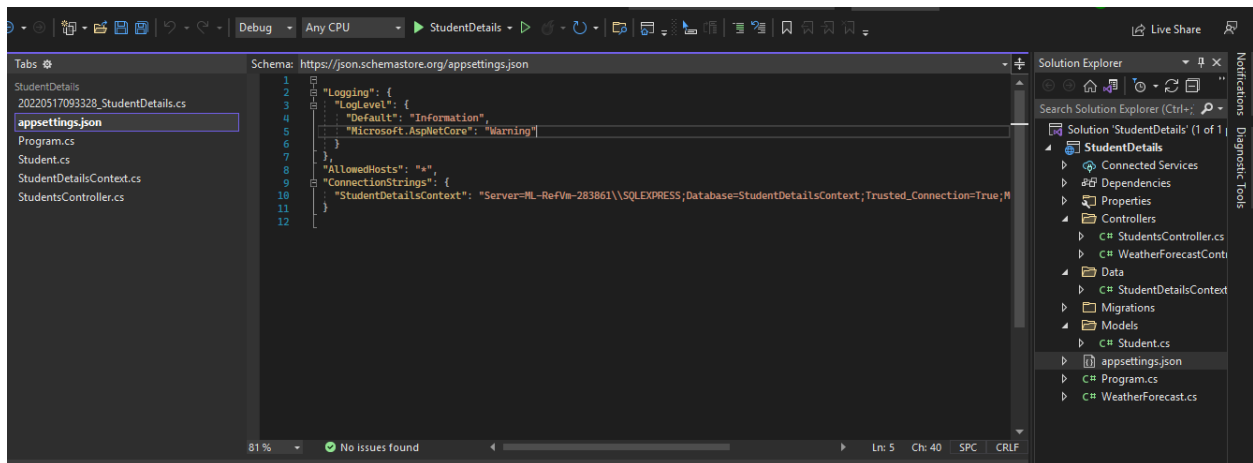
```

```

    {
        return _context.Student?.Any(e => e.StudentId == id).GetValueOrDefault();
    }
}
}

```

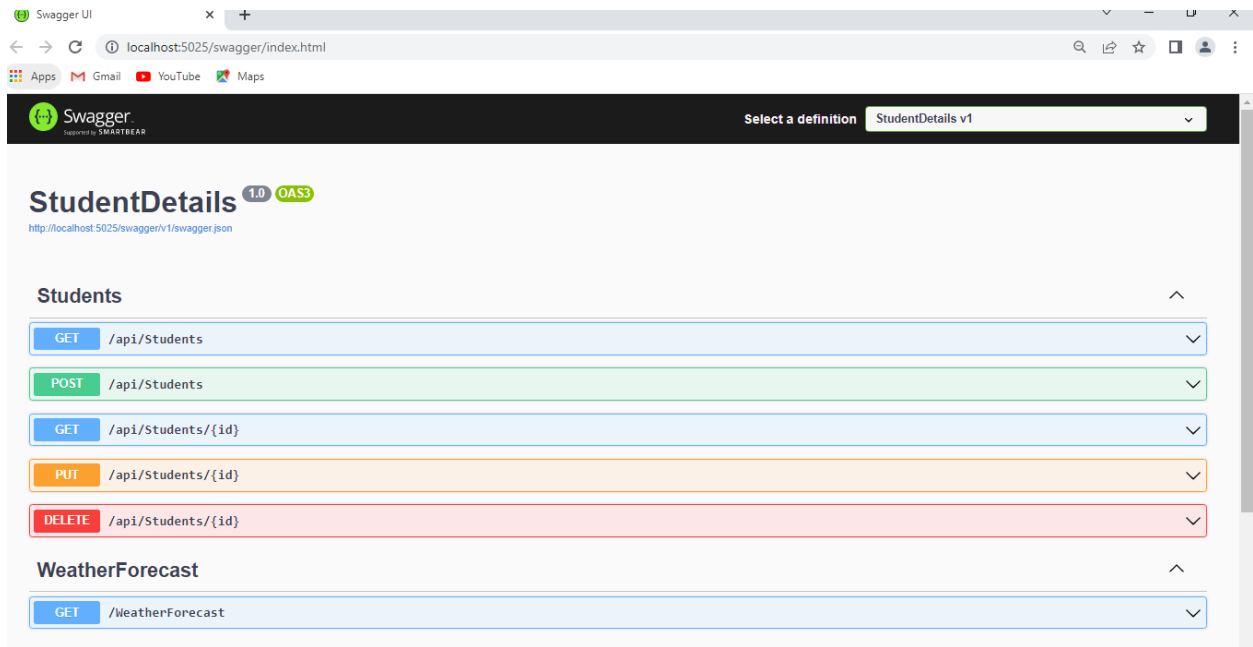
## Next Connection to database



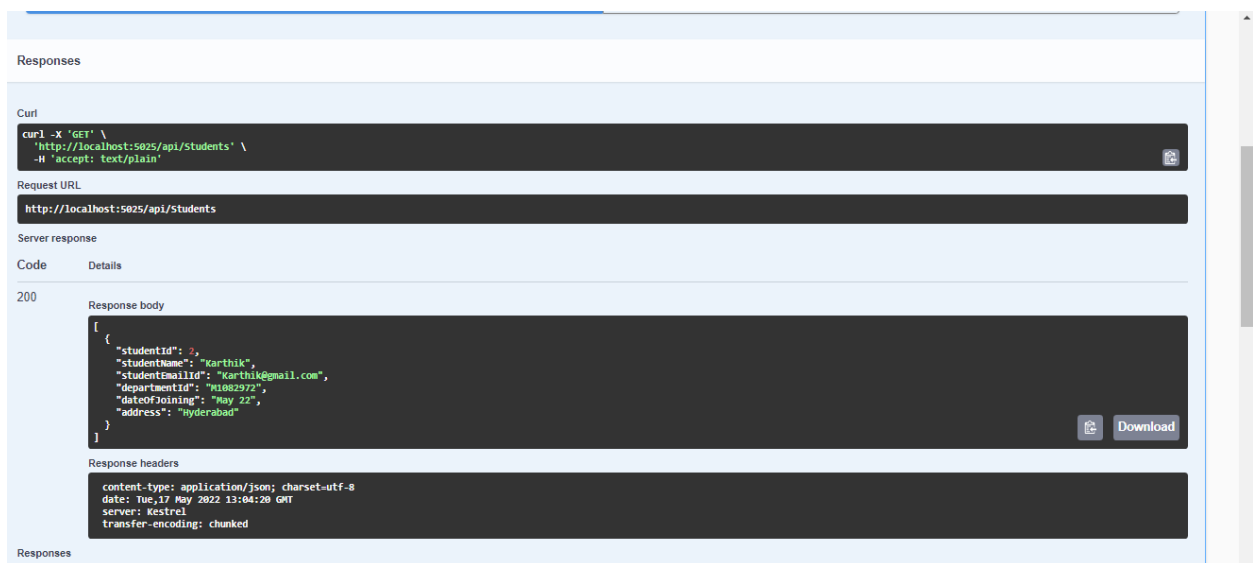
Next add-migration

Update Migration

After doing this we get



It running



All get, post , get , put , Delete all are running

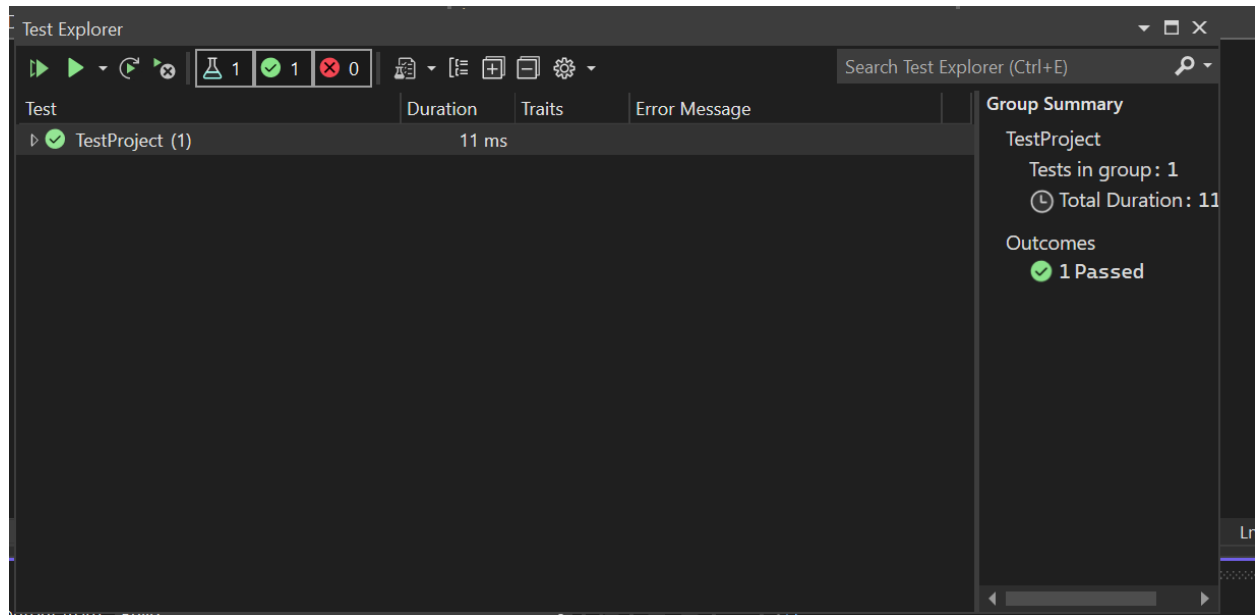
After Testing We get

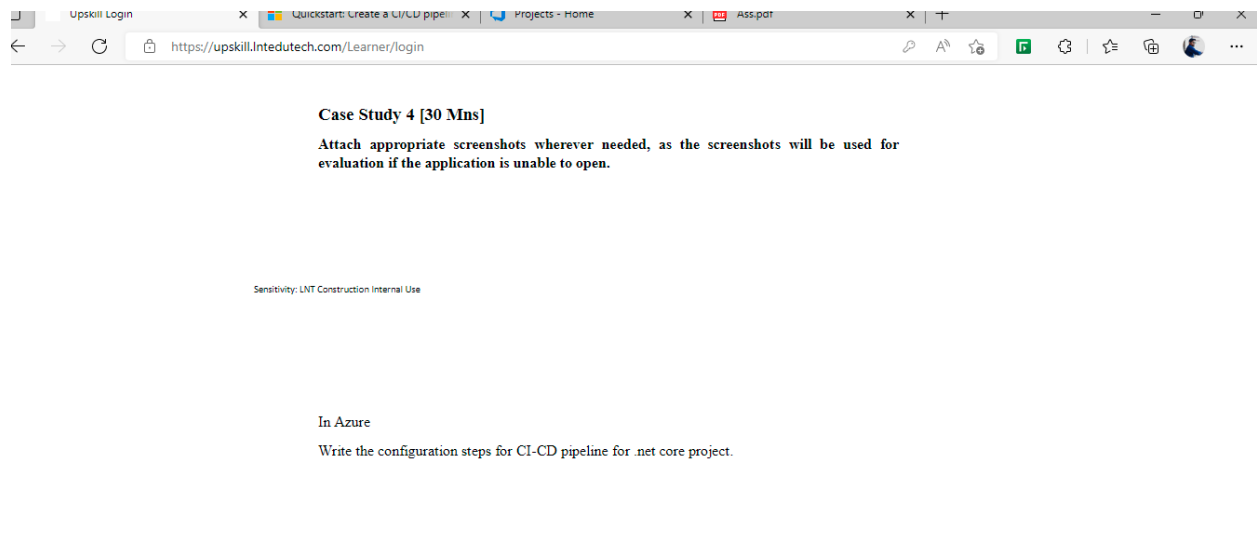
Succeed

Let Test



It Passed





#### CaseStudy-4

Write the configuration steps for CI-CD pipeline for .net core project.

##### Step:-1

The first step is to navigate to <https://dev.azure.com> and sign-in to Azure DevOps. You need to create at least an organization to store your projects, your repositories.

##### Step:-2

In the search box, type DevOps Starter, and then select. Click on Add to create a new one.

##### Step:-3

Select the .NET sample application. The .NET samples include a choice of either the open-source ASP.NET framework or the cross-platform .NET Core framework.

##### Step:-4

This sample is an ASP.NET Core MVC application. Select the .NET Core application framework, then select Next.

#### Step:-5

Select Windows Web App as a deployment target, then select Next. Optionally, you can choose other Azure services for your deployment. The application framework, which you chose previously, dictates the type of Azure service deployment target's available here.

#### Step:-6

Let Enter a Project name.

#### Steps:-7

Let Create a new free Azure DevOps Organization or choose an existing organization from the dropdown.

#### Step:-8

Select your Azure Subscription, enter a name for your Web app or take the default, then select Done. After a few minutes, the DevOps Starter Deployment Overview is displayed in the Azure portal.

#### Step:-9

Select Go to resource to view the DevOps Starter dashboard. In the upper right corner, pin the Project to your dashboard for quick access. A sample app is set up in a repo in your Azure DevOps Organization. A build is executed, and your app is deployed to Azure.

#### Step:-10

The dashboard provides visibility into your code repo, your CI/CD pipeline, and your app in Azure. At the right under Azure resources, select Browse to view your running app.

#### Step:-11

On the left of the DevOps Starter dashboard, select the link for your main branch. This link opens a view to the newly created Git repository.

#### Step:-12

In the next few steps, you can use the web browser to make and commit code changes directly to the main branch. You can also clone your Git repository in your favorite IDE by selecting Clone from the top right of the repository page.

#### Step:-13

On the left, navigate the application file structure to Application/aspnet-core-dotnet-core/Pages/Index.cshtml.

#### Step:-14

Select Edit, and then make a change to the h2 heading. For example, type Get started right away with the Azure DevOps Starter or make some other change.

#### Step:-15

Select Commit, leave a comment and select Commit again.

#### Step:-16

In your browser, go to the Azure DevOps Starter dashboard. You should now see a build is in progress. The changes you made are automatically built and deployed via a CI/CD pipeline.

#### Step:-17

At the top of the DevOps Starter dashboard, select Build Pipelines. This link opens a browser tab and the Azure DevOps build pipeline for your new project.

#### Step:-18

Select the ellipsis (...). This action opens a menu where you can start several activities such as queuing a new build, pausing a build, and editing the build pipeline.

#### Step:-19

Select Edit.

#### Step:-20

In this pane, you can examine the various tasks for your build pipeline. The build performs various tasks, such as fetching sources from the Git repository, restoring dependencies, and publishing outputs used that are used for deployments.

#### Step:-21

At the top of the build pipeline, select the build pipeline name.

#### Step:-22

Change the name of your build pipeline to something more descriptive, select Save & queue, and then select Save.

#### Step:-23

Under your build pipeline name, select History.

In the History pane, you see an audit trail of your recent changes for the build. Azure Pipelines keeps track of any changes that are made to the build pipeline, and it allows you to compare versions.

#### Step:-24

Select Triggers. DevOps Starter automatically created a CI trigger, and every commit to the repository starts a new build. You can optionally choose to include or exclude branches from the CI process.

#### Step:-25

Select Retention. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.

#### Step:-26

Select Build and Release, then select Releases.

DevOps Starter creates a release pipeline to manage deployments to Azure.

#### Step:-27

On the left, select the ellipsis (...) next to your release pipeline, and then select Edit. The release pipeline contains a pipeline, which defines the release process.

#### Step:-28

Under Artifacts, select Drop. The build pipeline you examined in the previous steps produces the output used for the artifact.

#### Step:-29

Next to the Drop icon, select the Continuous deployment trigger. This release pipeline has an enabled CD trigger, which runs a deployment every time there is a new build artifact available. Optionally, you can disable the trigger so that your deployments require manual execution.

#### Step:-30

On the left, select Tasks. The tasks are the activities that your deployment process performs. In this example, a task was created to deploy to Azure App Service.

#### Step:-31

On the right, select View releases. This view shows a history of releases.

#### Step:-32

Select the ellipsis (...) next to one of your releases, and then select Open. There are several menus to explore, such as a release summary, associated work items, and tests.

#### Step:-33

Select Commits. This view shows code commits that are associated with the specific deployment. Select Logs. The logs contain useful information about the deployment process. They can be viewed both during and after deployments.

