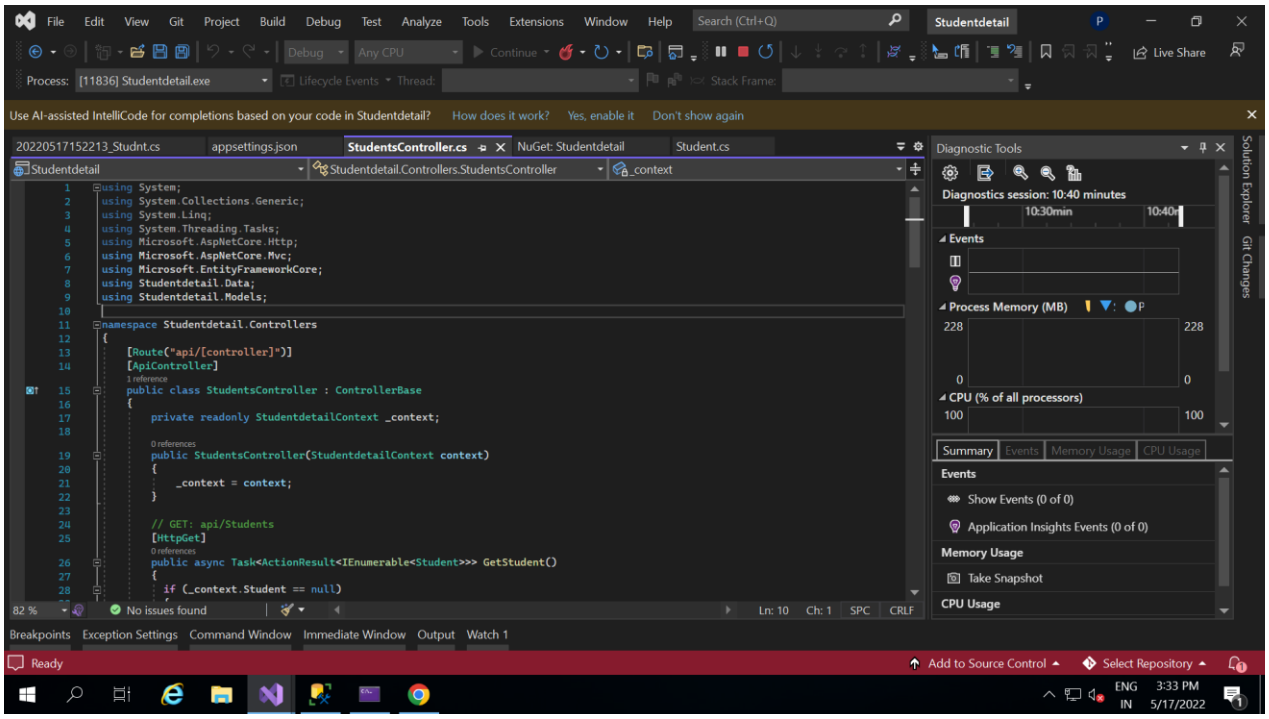
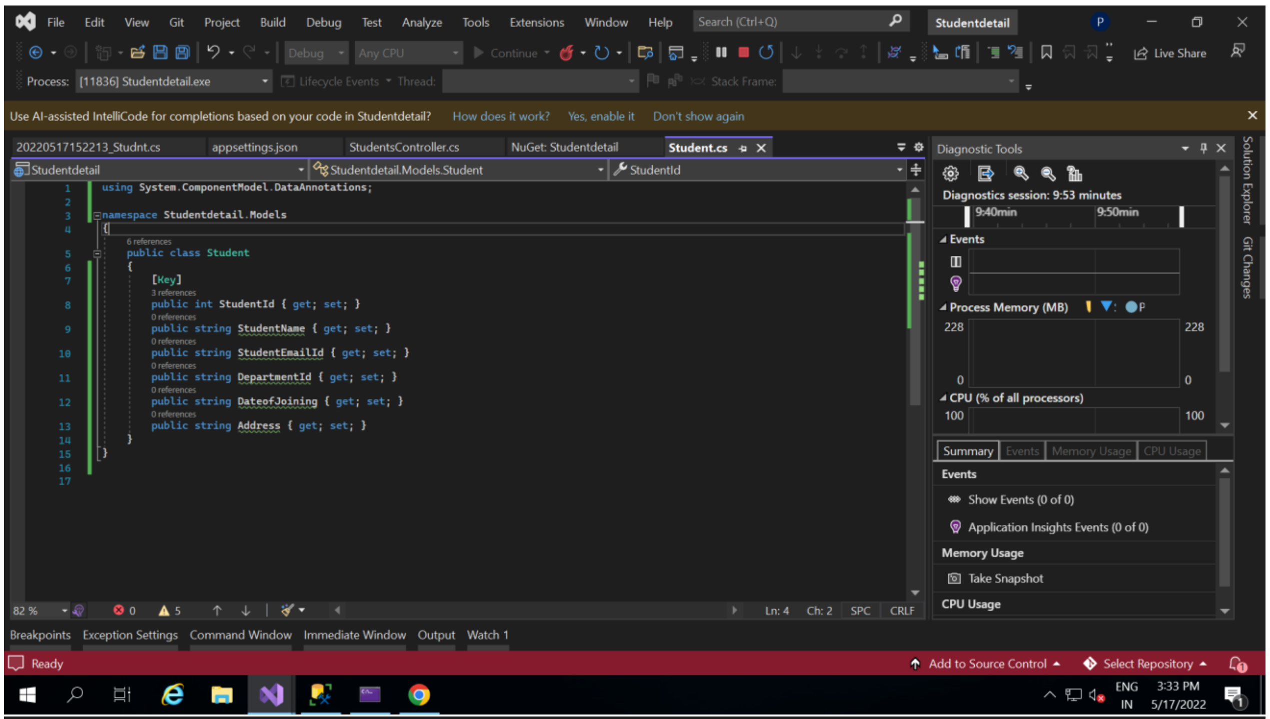
Milestone-REassessment-5

Name : Priyansu Jena

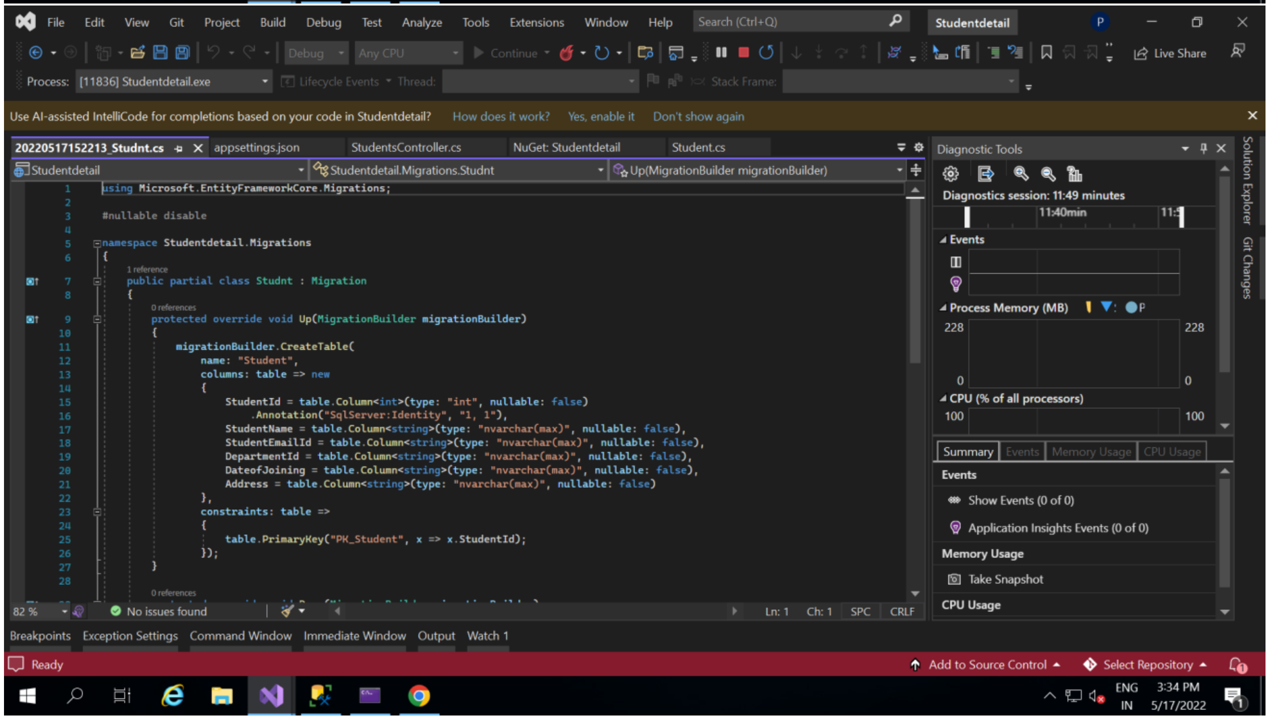
MID :M1082626



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Studentdetail.Data;
using Studentdetail.Models;

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

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

        // GET: api/Students
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Student>>> GetStudent()
        {
          if (_context.Student == null)
```



```
using System.ComponentModel.DataAnnotations;

namespace Studentdetail.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; }
    }
}
```



```
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Studentdetail.Migrations
{
    public partial class Studnt : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Student",
                columns: table => new
                {
                    StudentId = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    StudentName = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    StudentEmailId = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    DepartmentId = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    DateofJoining = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    Address = table.Column<string>(type: "nvarchar(max)", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Student", x => x.StudentId);
                });
        }
```

Code:
Student.cs:

```
using System.ComponentModel.DataAnnotations;

namespace Studentdetail.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; }
    }
}
```

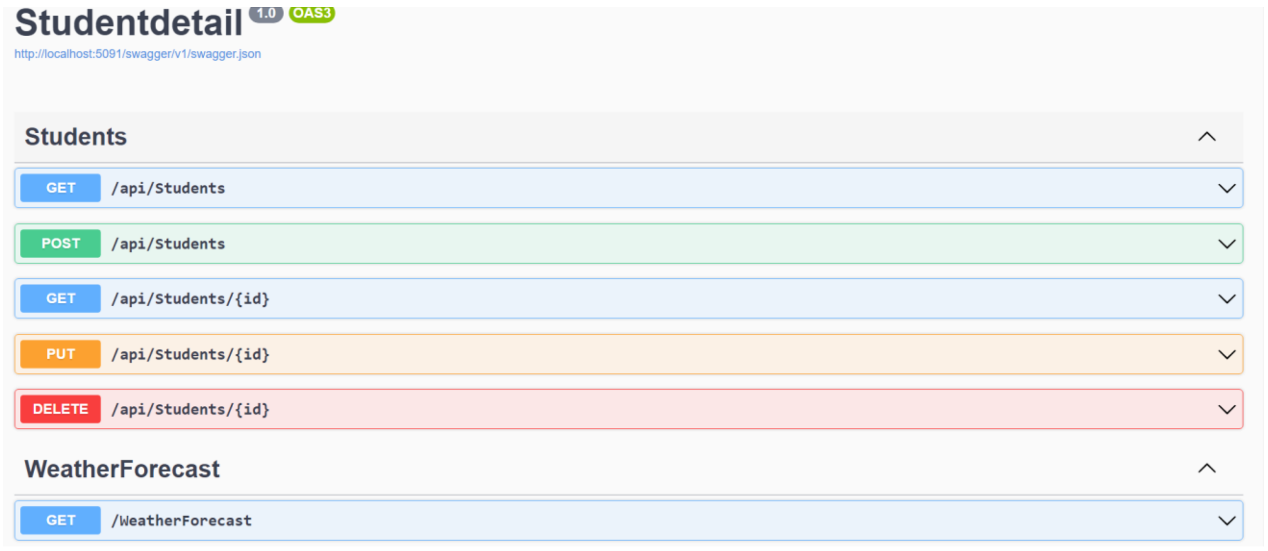
Controller

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Studentdetail.Data;
using Studentdetail.Models;

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

        public StudentsController(StudentdetailContext 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<IActionResult> 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 'StudentdetailContext.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<IActionResult> 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();
        }
    }
}
```

Migration:

```
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Studentdetail.Migrations
{
    public partial class Studnt : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Student",
                columns: table => new
                {
                    StudentId = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    StudentName = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    StudentEmailId = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    DepartmentId = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    DateofJoining = table.Column<string>(type: "nvarchar(max)", nullable: false),
                    Address = table.Column<string>(type: "nvarchar(max)", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Student", x => x.StudentId);
                });
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Student");
        }
    }
}
```



Studentdetail 1.0 OAS3

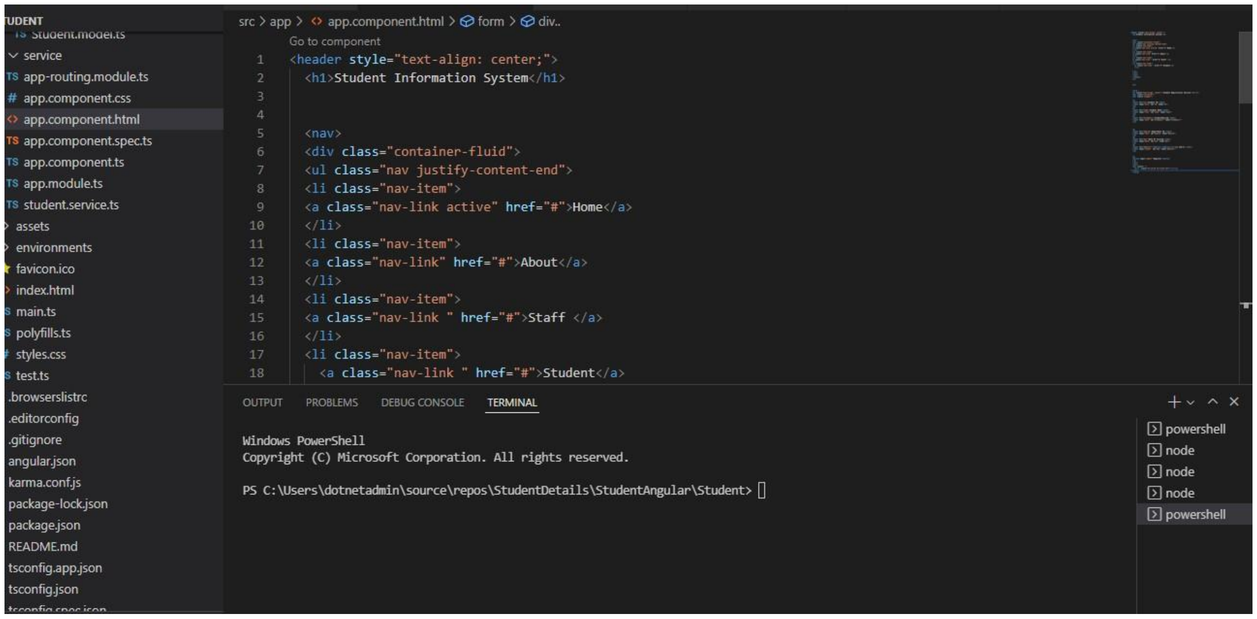
http://localhost:5091/swagger/v1/swagger.json

Students

GET	/api/Students
POST	/api/Students
GET	/api/Students/{id}
PUT	/api/Students/{id}
DELETE	/api/Students/{id}

WeatherForecast

GET	/WeatherForecast



```
<header style="text-align: center;">
  <h1>Student Information System</h1>


  <nav>
  <div class="container-fluid">
  <ul class="nav justify-content-end">
  <li class="nav-item">
  <a class="nav-link active" href="#">Home</a>
  </li>
  <li class="nav-item">
  <a class="nav-link" href="#">About</a>
  </li>
  <li class="nav-item">
  <a class="nav-link " href="#">Staff </a>
  </li>
  <li class="nav-item">
    <a class="nav-link " href="#">Student</a>
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\dotnetadmin\source\repos\StudentDetails\StudentAngular\Student>
```

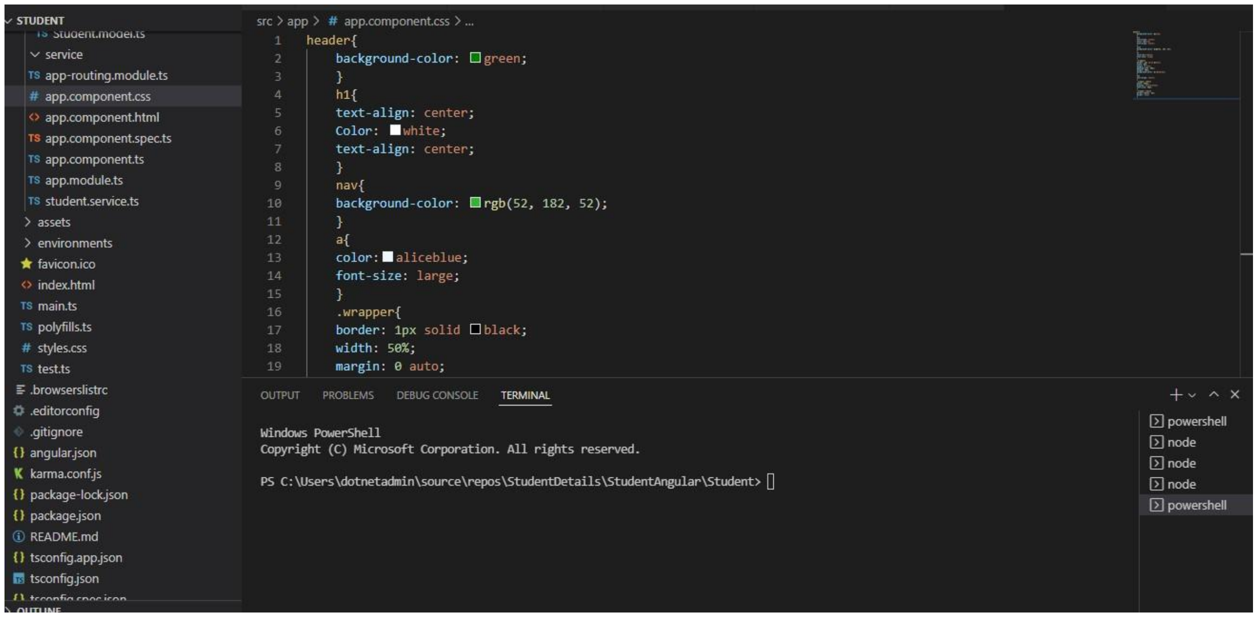
Html Code

```
<header style="text-align: center;">
  <h1>Student Information System</h1>
```

```html
<nav>
<div class="container-fluid">
<ul class="nav justify-content-end">
<li class="nav-item">
<a class="nav-link active" href="#">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">About</a>
</li>
<li class="nav-item">
<a class="nav-link " href="#">Staff </a>
</li>
<li class="nav-item">
  <a class="nav-link " href="#">Student</a>
  </li>

</ul>
</div>
</nav>
</header>
<br>
```

```html
<br>


<form >
<h3 style="text-align: center;">Student Registration Section</h3><br>
<div class="container">
<div class="wrapper">

<p>
<label for="id">Student ID</label>
<input type="text" id="id" name="id">
</p>
<p>
<label for="name">Student Name</label>
<input type="text" id="name" name="name">
</p>
<p>
<label for="StudEmail">StudentEmailId</label>
<input type="text" id="StudEmail" name="StudEmail">
</p>



<p>
<label for="Deptid">Department ID</label>
<input type="text" id="Deptid" name="Deptid">
</p>
<p>
<label for="doj">Date Of Joining</label>
<input type="date" id="doj" name="doj">
</p>
<p>
<label for="Address"><Address></Address><br>(in years)</label>
<input type="number" id="Add" name="Address">
</p>


<p>
<button type="submit">Register</button>
</p>
</div>
</div>
<div class=" ">
  <a><i  class="fa-solid fa-trash-can"></i></a>
</div>
</form>
```

css



```
header{
    background-color: green;
    }
    h1{
    text-align: center;
    Color: white;
    text-align: center;
    }
    nav{
    background-color: rgb(52, 182, 52);
    }
    a{
    color: aliceblue;
    font-size: large;
    }
    .wrapper{
    border: 1px solid black;
    width: 50%;
    margin: 0 auto;
```

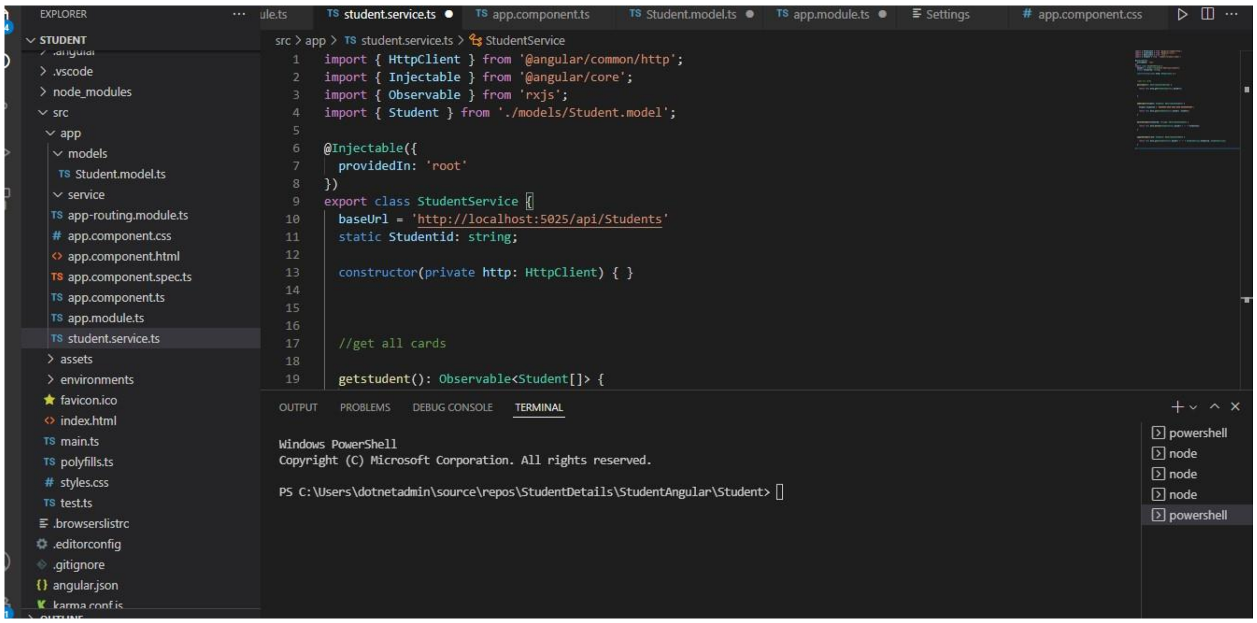
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\dotnetadmin\source\repos\StudentDetails\StudentAngular\Student>
```

Css code

```
header{     background-
color: green;
    }
h1{
    text-align: center;
Color: white;     text-align:
center;
    }
nav{
    background-color: rgb(52, 182, 52);
    }     a{
color:aliceblue;
font-size: large;
    }
    .wrapper{     border:
1px solid black;     width:
50%;     margin: 0 auto;
```

```
    padding-left: 20px;
height: 100%;      background-
color: lightgreen;
    }     h2{      text-
align: center;
    }
    .wrapper label{
width: 200px;      display:
inline-block;      font-
size: 20px;
    }
    .wrapper input{
border-radius: 5px;
border: none;
    }
```

Service.ts



```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Student } from './models/Student.model';

@Injectable({
  providedIn: 'root'
})
export class StudentService {
  baseUrl = 'http://localhost:5025/api/Students'
  static Studentid: string;

  constructor(private http: HttpClient) { }



  //get all cards

  getstudent(): Observable<Student[]> {
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\dotnetadmin\source\repos\StudentDetails\StudentAngular\Student>
```

Appcomponent.ts

```
import { Component, OnInit } from '@angular/core';
import { Student } from './models/Student.model'; import
{ StudentService } from './student.service';

@Component({
    selector: 'app-
root',
      templateUrl:
'./app.component.html',
     styleUrls:
['./app.component.css']

  })       export class
AppComponent {
      title =
'Student';

   Student: Student[] = [];
      student: Student
= {

    StudentId: '',
    StudentName: ' ',
    StudentEmailId: ' ',
    DepartmentId: ' ',
    DateOfJoining: ' ',
    Address: ' '


   }        constructor(private StudentService:
StudentService) {
   }
```

```
      ngOnInit():
void {

this.getstudent();

   }
getstudent(){

this.StudentService.getstudent()
  .subscribe(
     response
=> {
      this.Student =
response;

   }
  );

  }
onSubmit () {
      if (this.student.StudentId ===
'') {

this.StudentService.addStudent(this.student)

  .subscribe(

   response => {

   this.getstudent();



   }
  );

  }else {

this.updateCard(this.student);
```

```
  }
  }       deleteCard(id:
string) {

this.StudentService.deleteStudent(id)

  .subscribe(


response => {


this.getstudent();

  }

   );

  }       populateForm(Student:
Student) {
    this.student =
Student;

  }
  updateCard(Student: Student) {

  this.StudentService.updateStudent(Student)
   .subscribe(

  response => {

this.getstudent();

  }

  );
  }

  }
```

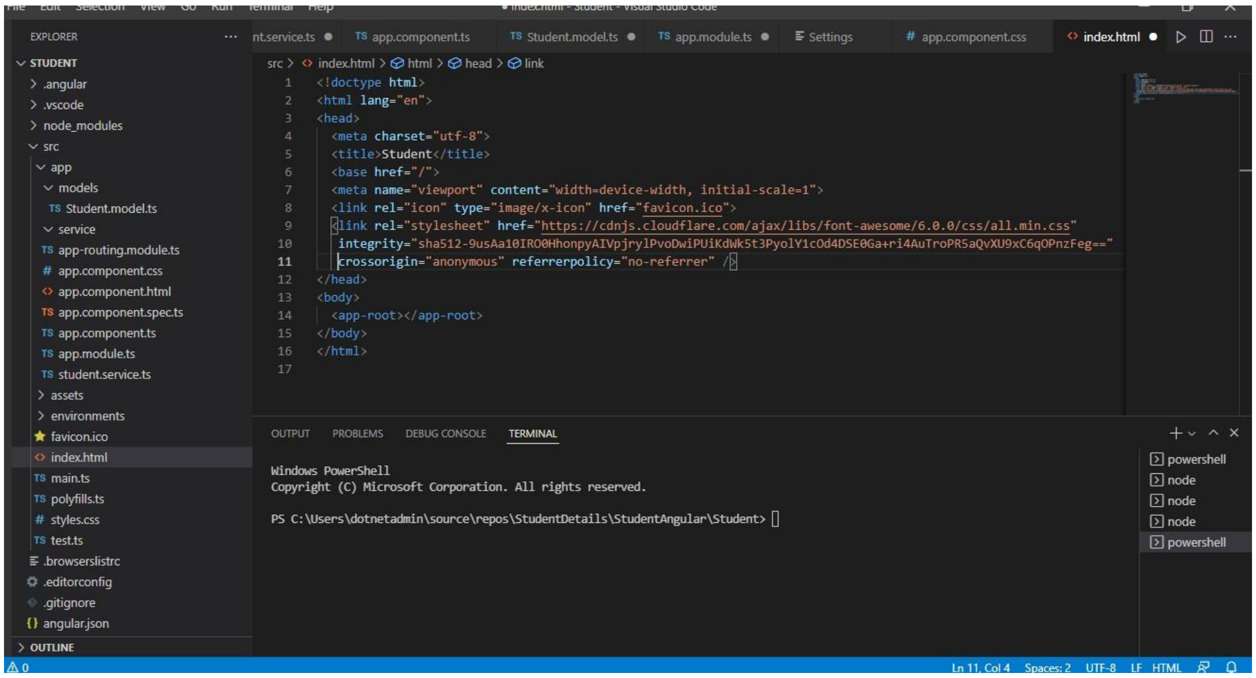
Appmodule.ts

```
import { NgModule } from '@angular/core'; import {
BrowserModule } from '@angular/platform-browser'; import
{HttpClientModule} from '@angular/common/http'; import {
AppRoutingModule } from './app-routing.module'; import {
AppComponent } from './app.component'; import {
FormsModule } from '@angular/forms';

@NgModule({
declarations: [
    AppComponent
  ],
  imports: [
BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule
  ],   providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
```

Student.model.ts

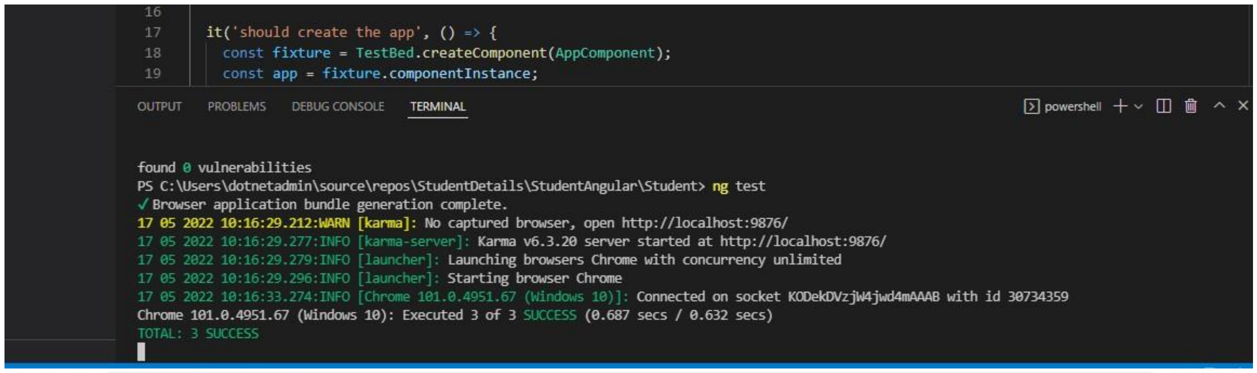
```
export interface Student{      StudentId: String;
    StudentName: String;
    StudentEmailId: String;
    DepartmentId: String;
    DateOfJoining: String;
    Address: String;
 }
```

Index.Html



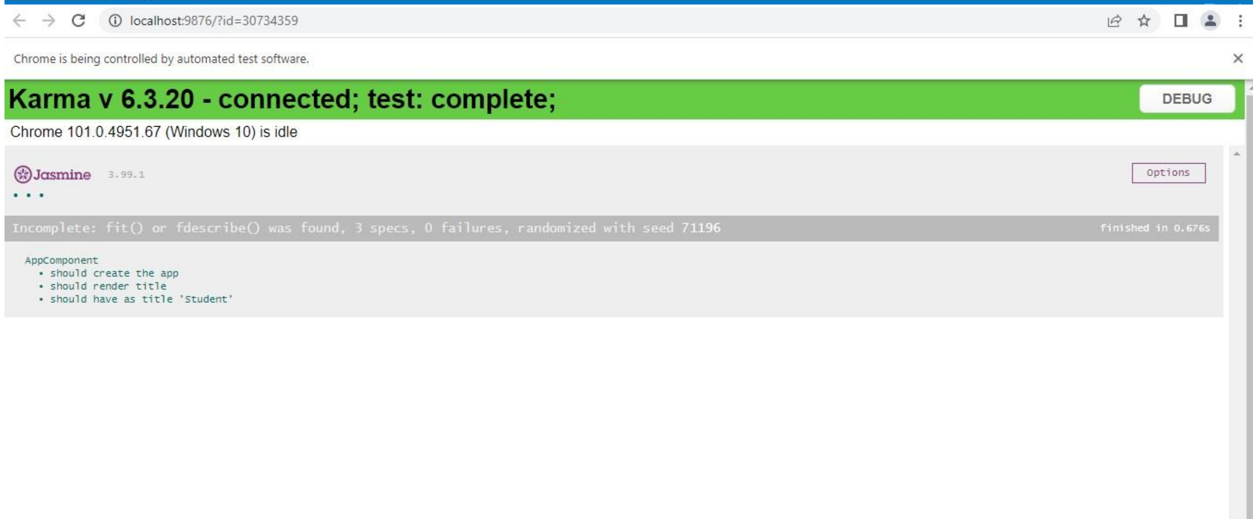
```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Student</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css"
   integrity="sha512-9usAa10IRO0HhonpyAIVpjrylPvoDwiPUiKdWk5t3PyolY1cOd4DSE0Ga+ri4AuTroPR5aQvXU9xC6qOPnzFeg=="
   crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\dotnetadmin\source\repos\StudentDetails\StudentAngular\Student>
```



```
  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
```

```
found 0 vulnerabilities
PS C:\Users\dotnetadmin\source\repos\StudentDetails\StudentAngular\Student> ng test
✓ Browser application bundle generation complete.
17 05 2022 10:16:29.212:WARN [karma]: No captured browser, open http://localhost:9876/
17 05 2022 10:16:29.277:INFO [karma-server]: Karma v6.3.20 server started at http://localhost:9876/
17 05 2022 10:16:29.279:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
17 05 2022 10:16:29.296:INFO [launcher]: Starting browser Chrome
17 05 2022 10:16:33.274:INFO [Chrome 101.0.4951.67 (Windows 10)]: Connected on socket KODekDVzjW4jwd4mAAAB with id 30734359
Chrome 101.0.4951.67 (Windows 10): Executed 3 of 3 SUCCESS (0.687 secs / 0.632 secs)
TOTAL: 3 SUCCESS
```



localhost:9876/?id=30734359

Chrome is being controlled by automated test software.

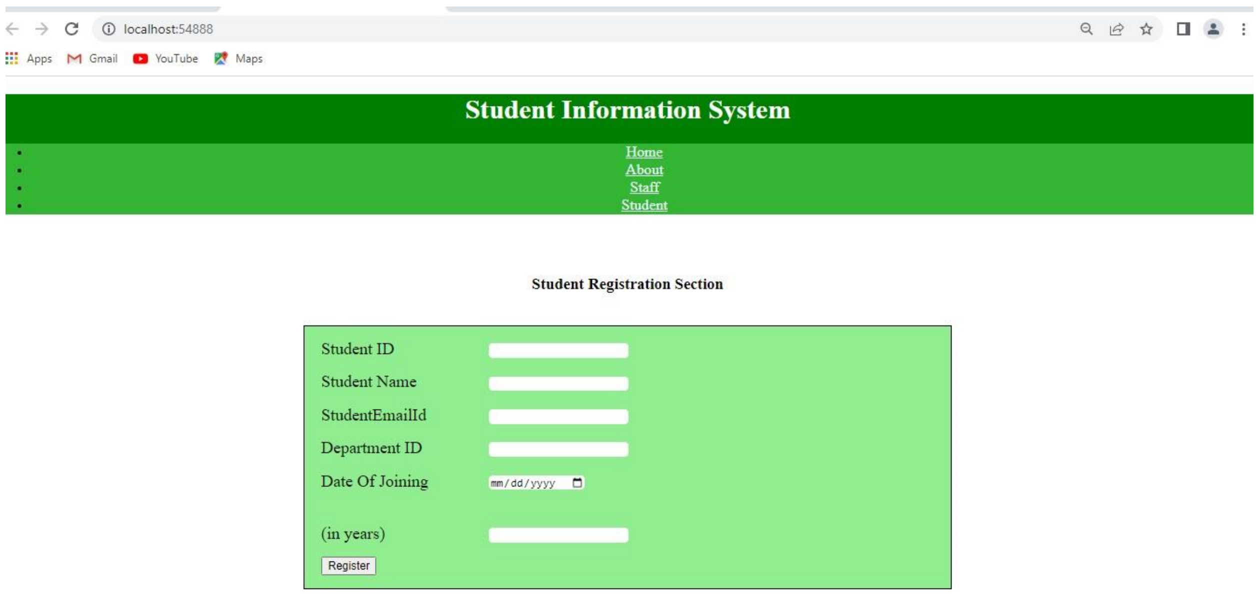
Karma v 6.3.20 - connected; test: complete;

Chrome 101.0.4951.67 (Windows 10) is idle

Jasmine 3.99.1

```
Incomplete: fit() or fdescribe() was found, 3 specs, 0 failures, randomized with seed 71196     finished in 0.676s

AppComponent
  • should create the app
  • should render title
  • should have as title 'Student'
```

Output



localhost:54888

Student Information System

- Home
- About
- Staff
- Student

Student Registration Section

Student ID	
Student Name	
StudentEmailId	
Department ID	
Date Of Joining	mm/dd/yyyy
(in years)	

Register