-: WEEK - 4 :-

Web API Hands-On Report - Task 1

1. Objectives :-

- Explain the concept of RESTful web service, Web API & Microservice
- Demonstrate features of REST architecture
- Understand HttpRequest & HttpResponse
- Use Http Verbs: GET, POST, PUT, DELETE in Web API
- Apply HttpStatusCodes like OK, NotFound, etc.
- Create a simple WebAPI with Read and Write actions
- Structure the WebAPI using Controllers and Action Methods
- Describe configuration files like Startup.cs, appSettings.json, launchSettings.json

2. Concepts Explained :-

2.1 RESTful Web Service and Web API :-

A RESTful Web Service is an architectural style that uses standard HTTP methods to perform CRUD operations. It is stateless and resource-oriented.

A Web API is a framework for building HTTP services that can be consumed by clients like browsers, mobile apps, or other servers.

2.2 Microservice :-

Microservices architecture structures an application as a collection of loosely coupled services. Each service implements a specific business capability and is independently deployable.

2.3 HTTP Request and Response :-

I . HttpRequest represents the incoming request from a client to the server.

II . HttpResponse represents the server's response, typically including a status code and data (JSON/XML).

2.4 Action Verbs and Status Codes :-

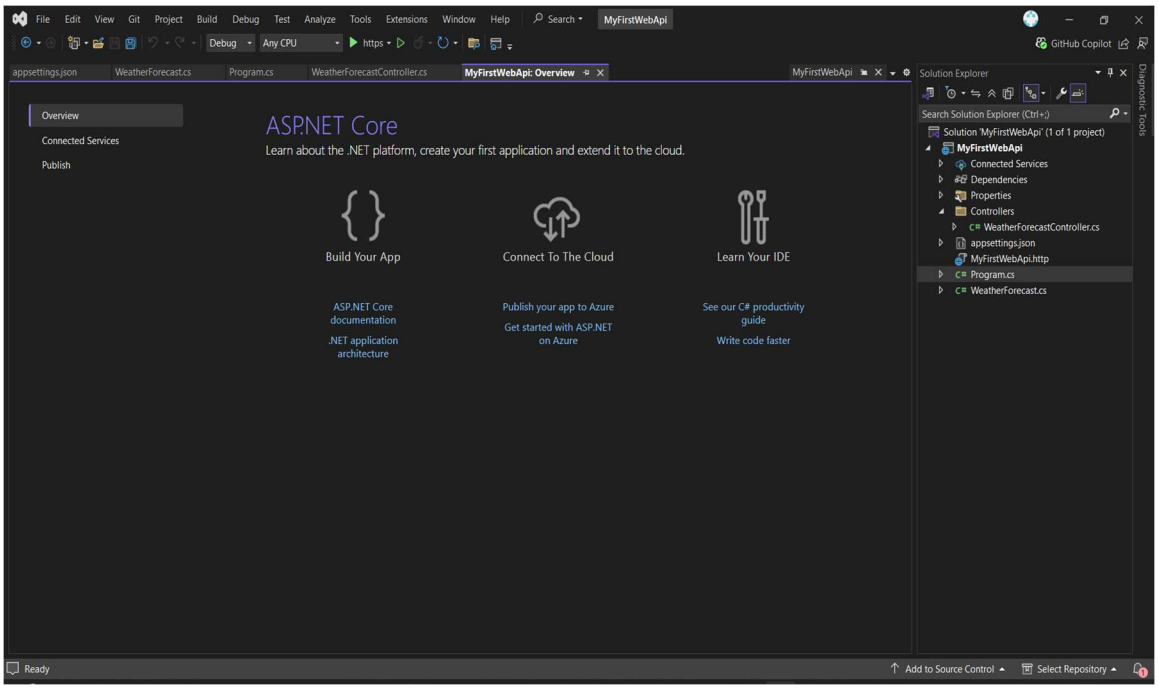
- **HttpGet**: Retrieves data.
- **HttpPost**: Creates a new resource.
- **HttpPut**: Updates an existing resource.
- **HttpDelete**: Deletes a resource.

Common HTTP status codes:-

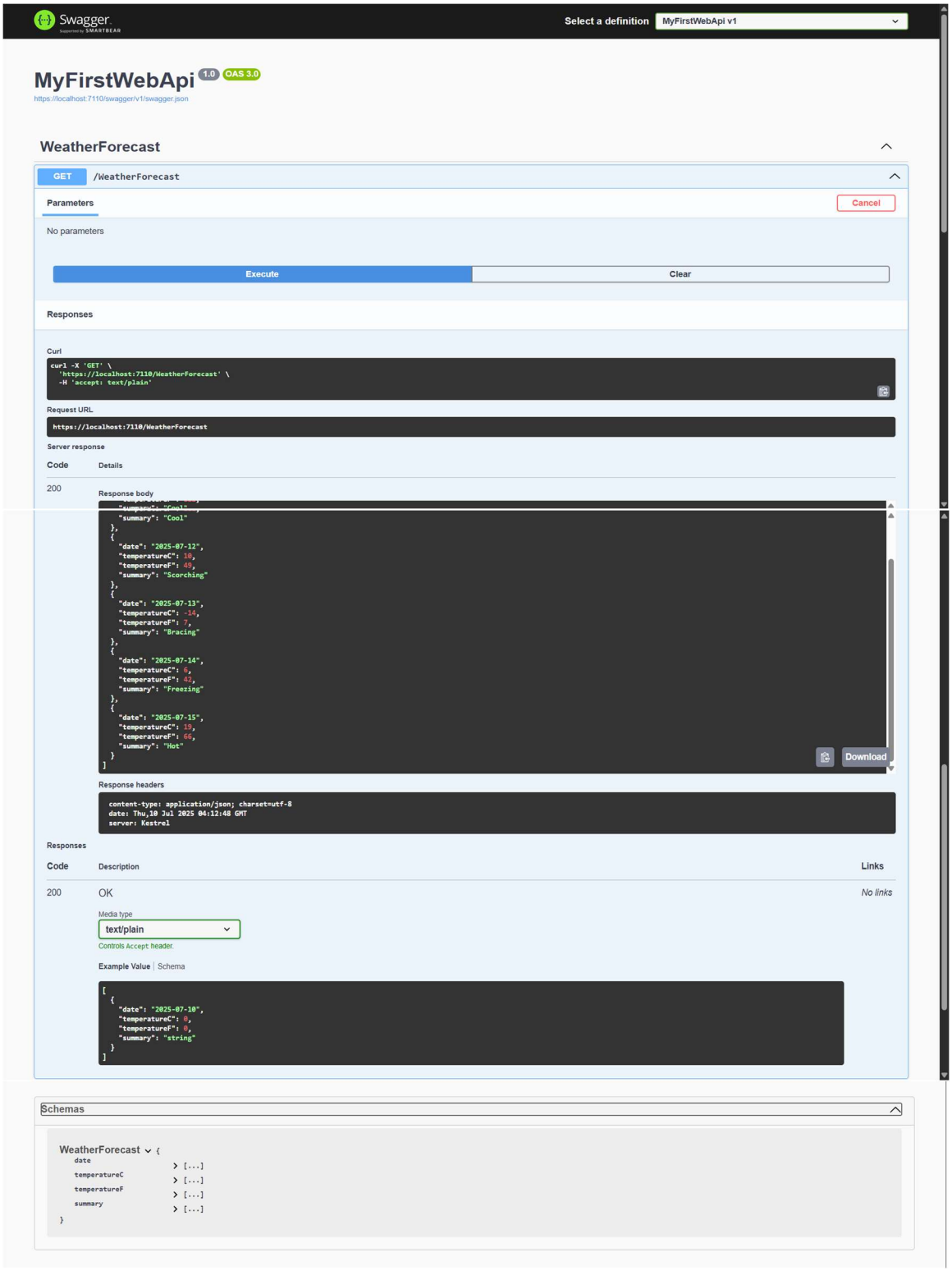
- **200 OK**: Request succeeded.
- **404 Not Found**: Resource not found.
- **500 Internal Server Erro**r: Unexpected server error.
- **400 Bad Request**: Malformed request.
- **401 Unauthorized**: Access denied.

3. Implementation Steps :-

3.1 Project Setup :-



3.2 Default Controller Test :-



3.3 Create Custom Controller :-

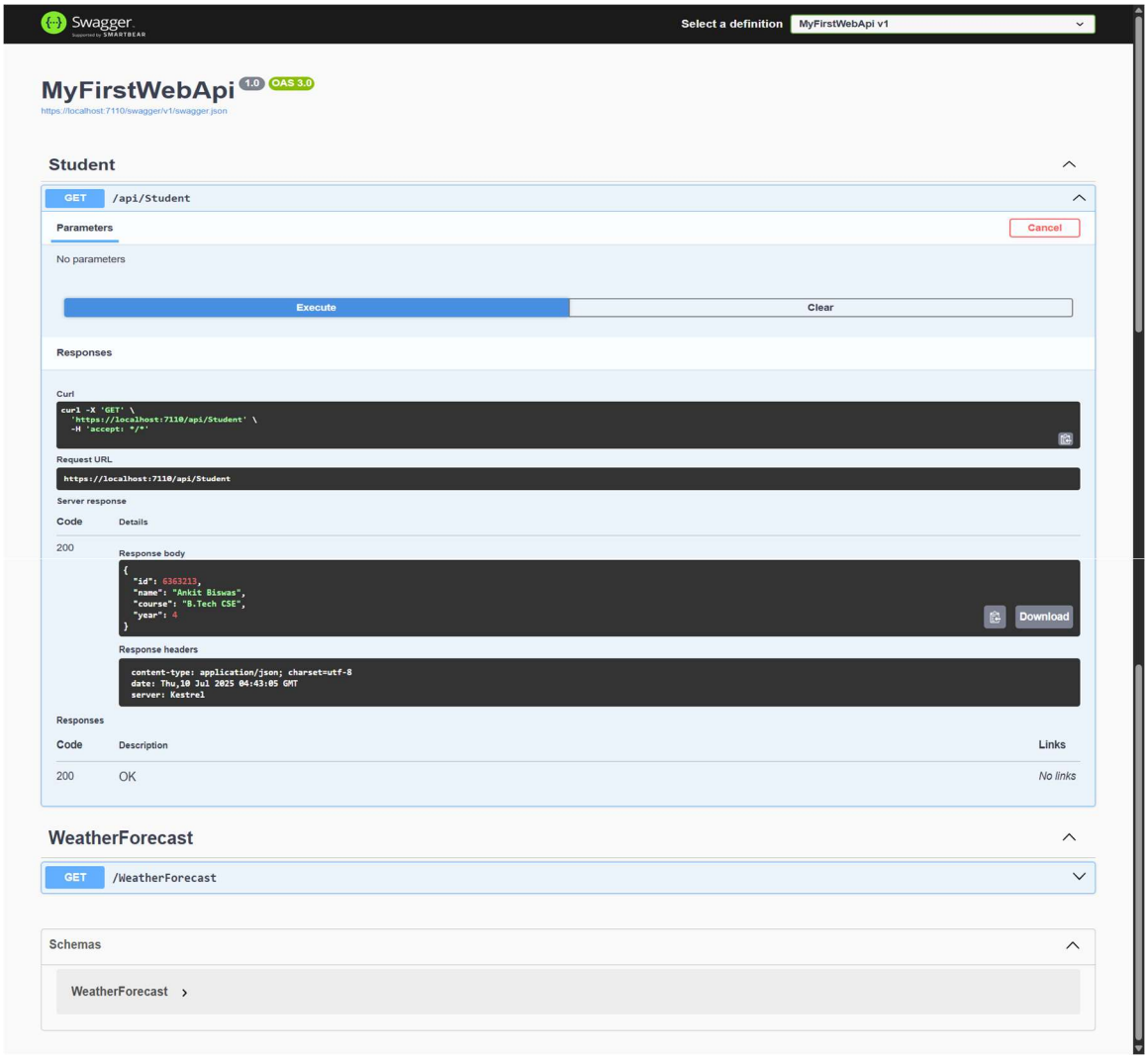
```
using Microsoft.AspNetCore.Mvc;


namespace MyFirstWebApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class StudentController : ControllerBase
    {
        private static Student currentStudent = new Student
        {
            Id = 6363213,
            Name = "Ankit Biswas",
            Course = "B.Tech CSE",
            Year = 4
        };
        [HttpGet]
        public IActionResult GetStudent()
        {
            return Ok(currentStudent);
        }
        [HttpPost]
        public IActionResult AddStudent([FromBody] Student newStudent)
        {
            currentStudent = newStudent;
            return Ok(new
            {
                message = "Student added successfully!",
```

```
                data = currentStudent
            });
        }
        [HttpPut]
        public IActionResult UpdateStudent([FromBody] Student updatedStudent)
        {
            currentStudent = updatedStudent;
            return Ok(new
            {
                message = "Student updated successfully!",
                data = currentStudent
            });
        }
        [HttpDelete("{id}")]
        public IActionResult DeleteStudent(int id)
        {
            if (currentStudent.Id == id)
            {
                currentStudent = null;
                return Ok(new { message = $"Student with ID {id} deleted."
});
            }
            else
            {
                return NotFound(new { message = "Student not found." });
            }
        }
    }
    public class Student
```

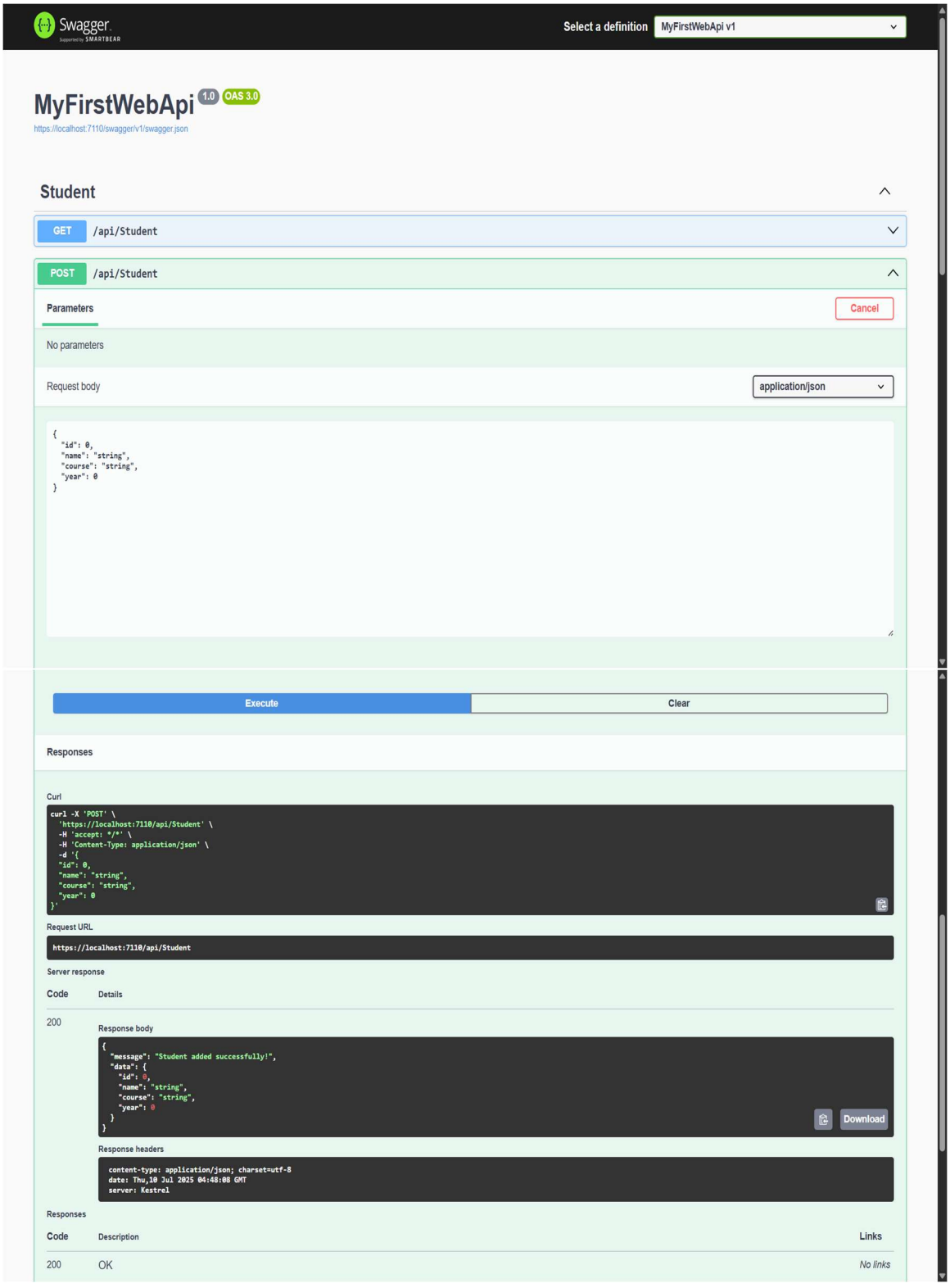
```
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Course { get; set; }
        public int Year { get; set; }
    }
}
```

3.4 Test in Swagger :-

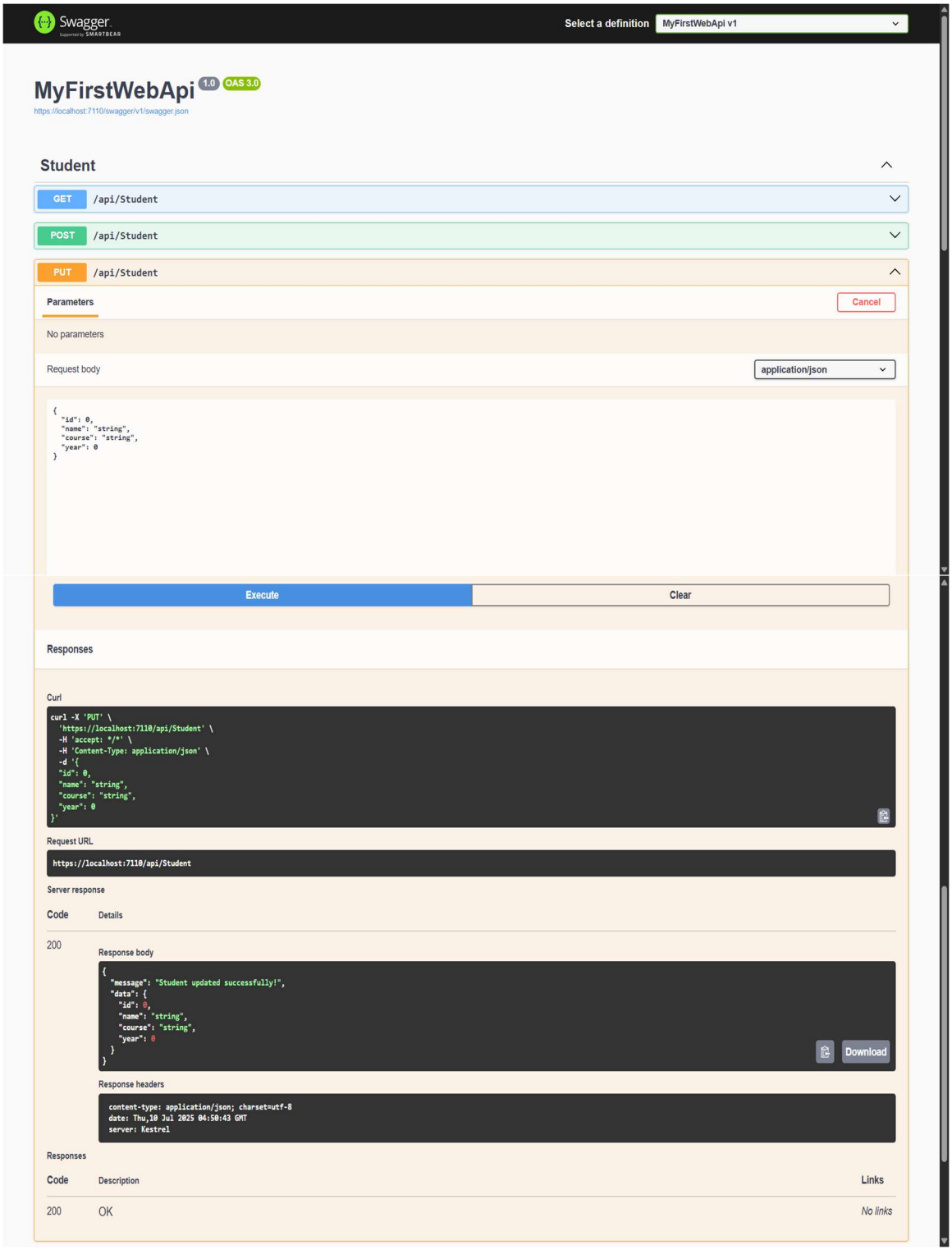
GET :-


Swagger
Select a definition
MyFirstWebApi v1
MyFirstWebApi
1.0
OAS 3.0
https://localhost:7110/swagger/v1/swagger.json
Student
GET
/api/Student
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl
curl -X 'GET' \
'https://localhost:7110/api/Student' \
-H 'accept: */*'
Request URL
https://localhost:7110/api/Student
Server response
Code
Details
200
Response body
{
"id": 6363213,
"name": "Ankit Biswas",
"course": "B.Tech CSE",
"year": 4
}
Download
Response headers
content-type: application/json; charset=utf-8
date: Thu,10 Jul 2025 04:43:05 GMT
server: Kestrel
Responses
Code
Description
Links
200
OK
No links
WeatherForecast
GET
/WeatherForecast
Schemas
WeatherForecast

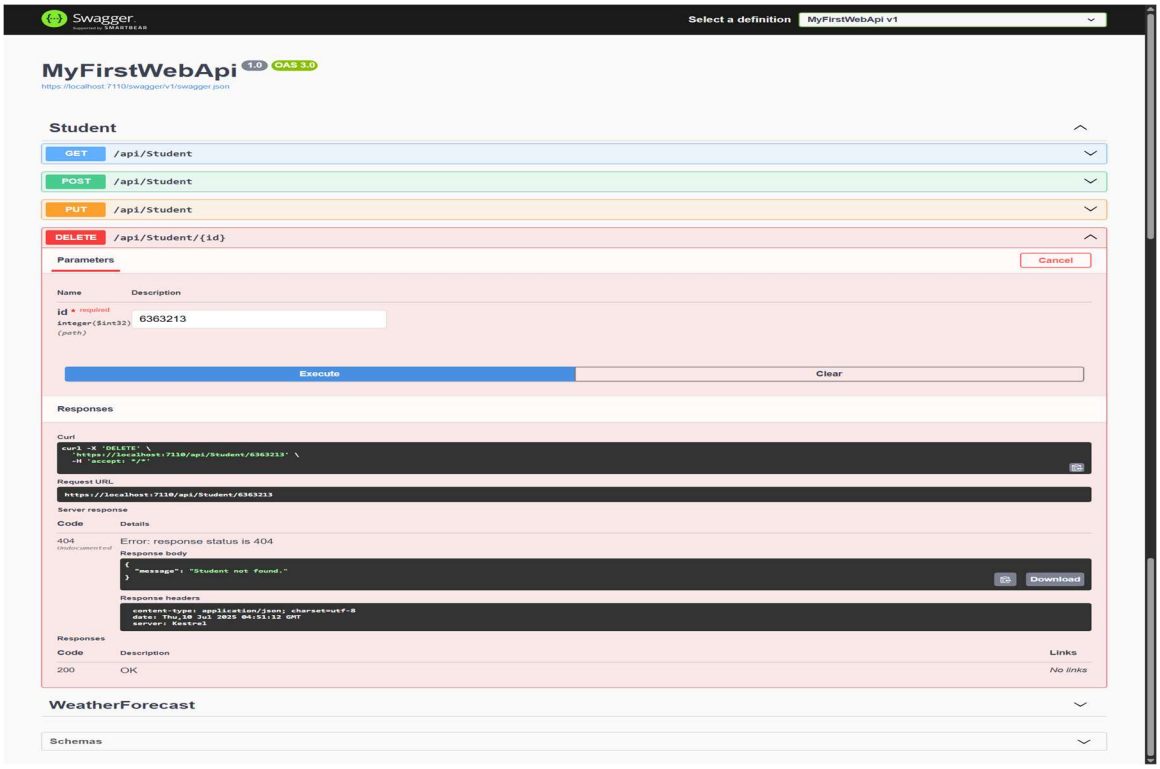
POST:-


Swagger
Supported by SMARTBEAR
Select a definition
MyFirstWebApi v1
MyFirstWebApi
1.0
OAS 3.0
https://localhost:7110/swagger/v1/swagger.json
Student
GET
/api/Student
POST
/api/Student
Parameters
Cancel
No parameters
Request body
application/json
{
"id": 0,
"name": "string",
"course": "string",
"year": 0
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'https://localhost:7110/api/Student' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"id": 0,
"name": "string",
"course": "string",
"year": 0
}'
Request URL
https://localhost:7110/api/Student
Server response
Code
Details
200
Response body
{
"message": "Student added successfully!",
"data": {
"id": 0,
"name": "string",
"course": "string",
"year": 0
}
}
Download
Response headers
content-type: application/json; charset=utf-8
date: Thu,10 Jul 2025 04:48:08 GMT
server: Kestrel
Responses
Code
Description
Links
200
OK
No links

PUT:-

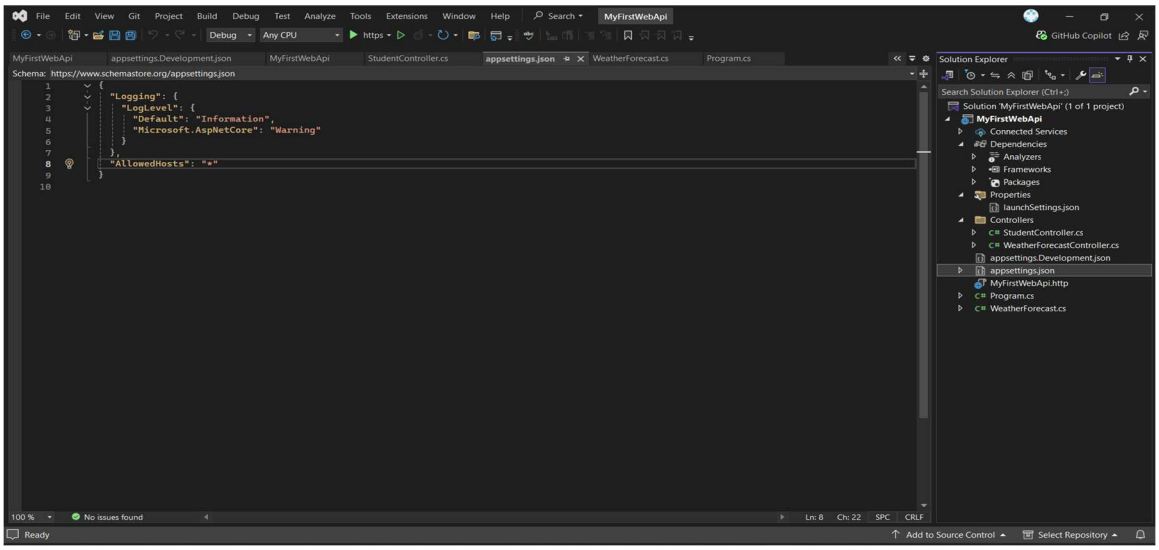

Swagger
Select a definition
MyFirstWebApi v1
MyFirstWebApi
1.0
OAS 3.0
https://localhost:7110/swagger/v1/swagger.json
Student
GET /api/Student
POST /api/Student
PUT /api/Student
Parameters
Cancel
No parameters
Request body
application/json
{
"id": 0,
"name": "string",
"course": "string",
"year": 0
}
Execute
Clear
Responses
Curl
curl -X 'PUT' \
'https://localhost:7110/api/Student' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
"id": 0,
"name": "string",
"course": "string",
"year": 0
}'
Request URL
https://localhost:7110/api/Student
Server response
Code
Details
200
Response body
{
"message": "Student updated successfully!",
"data": {
"id": 0,
"name": "string",
"course": "string",
"year": 0
}
}
Download
Response headers
content-type: application/json; charset=utf-8
date: Thu,10 Jul 2025 04:50:43 GMT
server: Kestrel
Responses
Code
Description
Links
200
OK
No links

DELETE:-

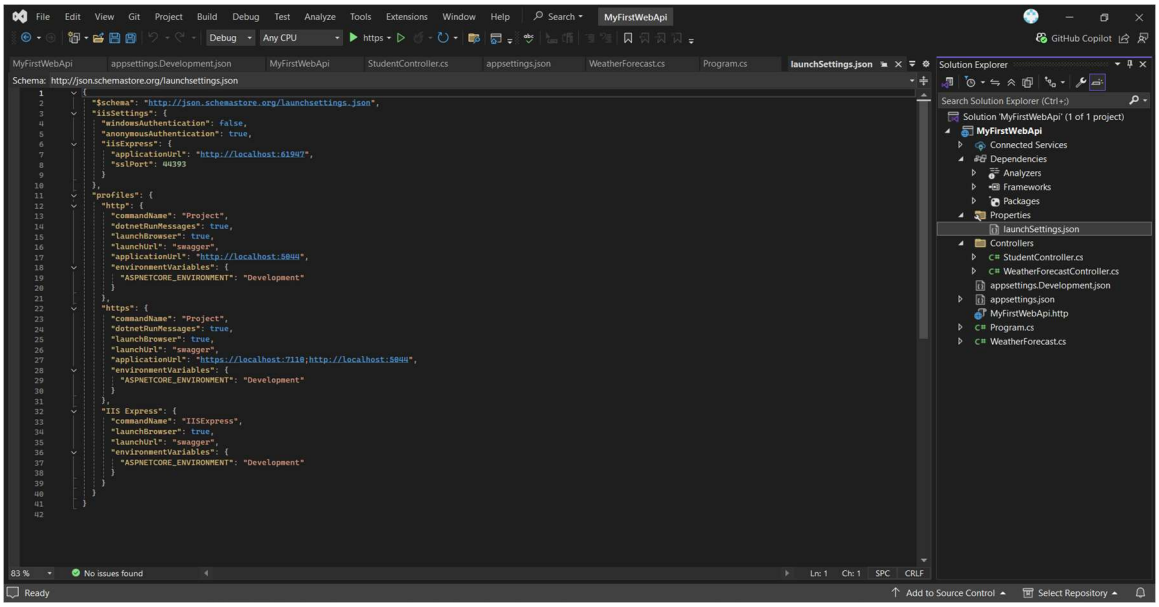


3.5 Configuration Files :-

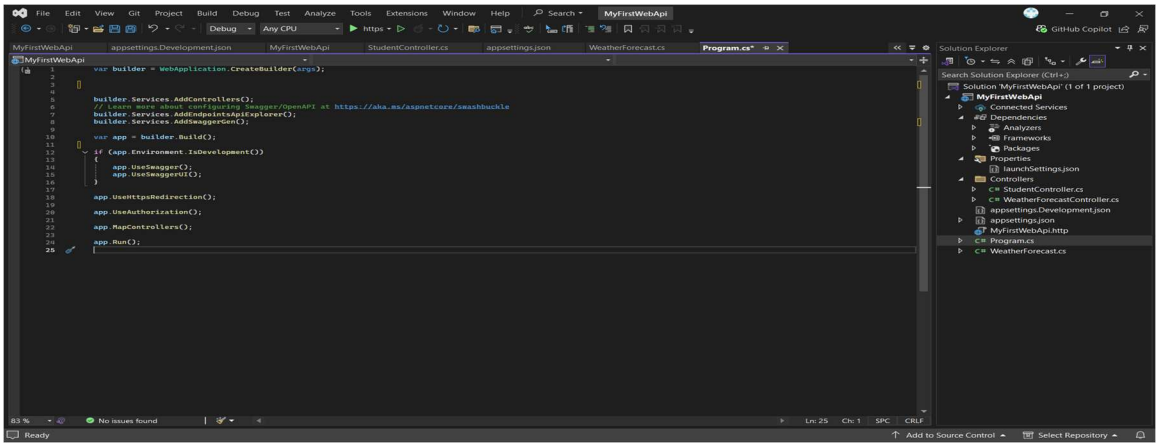
appsettings.json: -



launchSettings.json :-


launchSettings.json
Schema: http://json.schemastore.org/launchsettings.json
{
"$schema": "http://json.schemastore.org/launchsettings.json",
"iisSettings": {
"windowsAuthentication": false,
"anonymousAuthentication": true,
"iisExpress": {
"applicationUrl": "http://localhost:61947",
"sslPort": 44393
}
},
"profiles": {
"http": {
"commandName": "Project",
"dotnetRunMessages": true,
"launchBrowser": true,
"launchUrl": "swagger",
"applicationUrl": "http://localhost:5044",
"environmentVariables": {
"ASPNETCORE_ENVIRONMENT": "Development"
}
},
"https": {
"commandName": "Project",
"dotnetRunMessages": true,
"launchBrowser": true,
"launchUrl": "swagger",
"applicationUrl": "https://localhost:7118;http://localhost:5044",
"environmentVariables": {
"ASPNETCORE_ENVIRONMENT": "Development"
}
},
"IIS Express": {
"commandName": "IISExpress",
"launchBrowser": true,
"launchUrl": "swagger",
"environmentVariables": {
"ASPNETCORE_ENVIRONMENT": "Development"
}
}
}
}
Solution Explorer
Solution 'MyFirstWebApi' (1 of 1 project)
MyFirstWebApi
Connected Services
Dependencies
Analyzers
Frameworks
Packages
Properties
launchSettings.json
Controllers
StudentController.cs
WeatherForecastController.cs
appsettings.Development.json
appsettings.json
MyFirstWebApi.http
Program.cs
WeatherForecast.cs
No issues found
Ready
Add to Source Control
Select Repository

Program.cs:-


var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
var app = builder.Build();
if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}
app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.Run();
Program.cs
Solution 'MyFirstWebApi' (1 of 1 project)
No issues found
Ready
Add to Source Control
Select Repository

Web API Hands-On Lab 2 Report

Objective :-

1. Demonstrate Swagger Installation to WebAPI and WebAPI Listing on Browser:-

To install Swagger in a .NET Core Web API project, the NuGet package 'Swashbuckle.AspNetCore' is required. This package enables automatic generation of OpenAPI documentation.

After installing the package, the following service needs to be added in the Program.cs file to configure Swagger:

```
builder.Services.AddSwaggerGen();
```

In the same file, Swagger middleware is enabled with the following code:

```
app.UseSwagger();
```

```
app.UseSwaggerUI();
```

Swagger UI can be accessed via the browser at https://localhost:<port>/swagger. It displays all available controllers and actions in the API.

The ProducesResponseType attribute is used to explicitly define the expected HTTP status codes returned by an action method. For example: [ProducesResponseType(200)] indicates a successful response with status code 200.

2. Demonstrate the Usage of Postman Tool to Hit WebAPI Methods:-

Postman is a widely-used tool for testing RESTful APIs. It provides the ability to organize requests, send headers and body, choose request types, and inspect responses.

The structure in Postman includes:

• Request collections to organize related API requests.

• Request builder with method selector (GET, POST, etc.).

• Headers tab for setting custom headers (e.g., Authorization).

• Body tab for sending JSON payloads in POST requests.

• Tabs in the center pane showing request configuration and response details.

To test the Web API, a POST request can be made to the 'AddEmployee' endpoint with a raw JSON body. Similarly, a GET request can be used to retrieve a list of employees. Both requests return the expected results as tested in Postman.

3. Demonstrate the Usage of Route and Explain Name Attribute in HTTP Requests :-

The Route attribute defines the base path used for an API controller. For example, [Route("Emp")] will make the base route '/Emp'. This provides clarity and user-friendly URLs when accessing endpoints.

The ActionName attribute allows a developer to define multiple action methods with the same HTTP verb, but distinct names. This is especially useful in REST APIs to distinguish between similar actions like 'GetAll' and 'GetById'.

Example usage:-

```
[HttpGet("GetAllEmployees")]

[ActionName("GetAll")]
```

4. Demonstrate Creation of a Simple WebAPI - With Read and Write Actions :-

A simple Web API was developed with two core actions: one for reading data (GET) and another for writing data (POST).

The GET action retrieves a list of predefined employees, while the POST action allows adding new employees via JSON input.

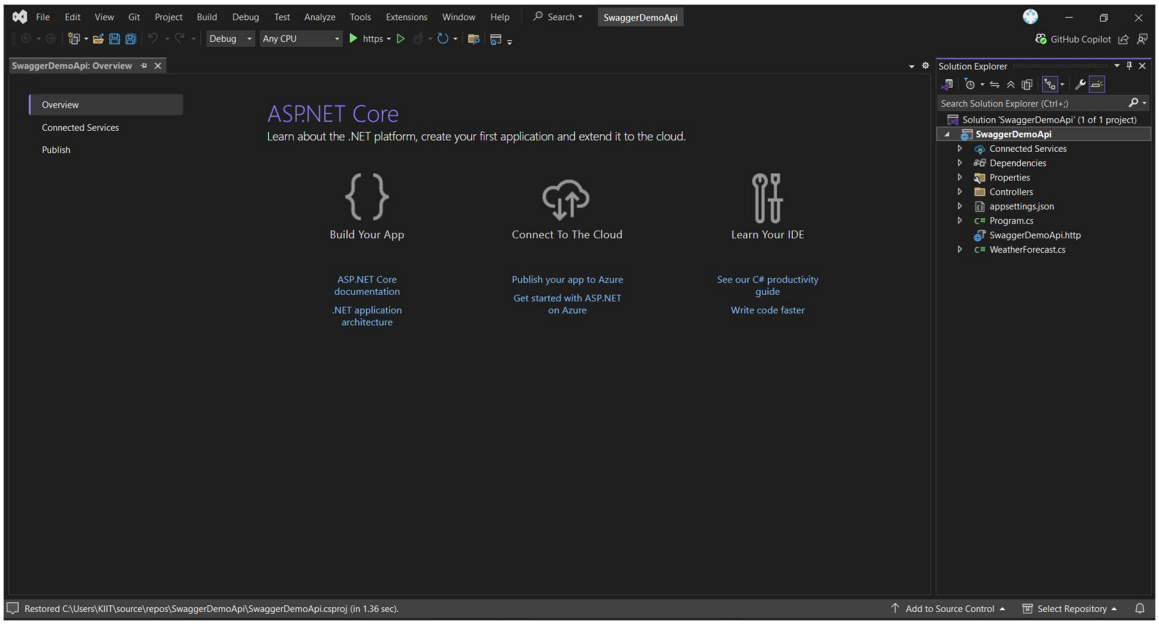
GET Endpoint: /Emp/GetAllEmployees

POST Endpoint: /Emp/AddEmployee

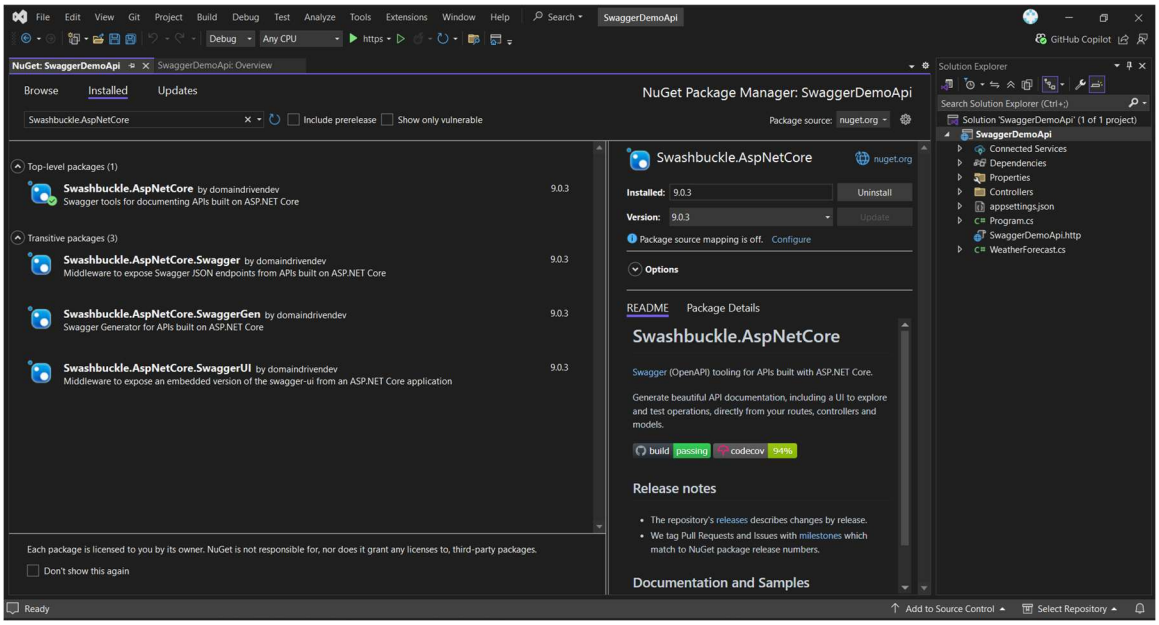
Both actions are successfully tested in Postman. The API is functional, well-documented using Swagger, and follows proper RESTful practices.

Steps and Implementation :-

Step 1: Create a .Net core web application with API template. :-



Step 2 : Install Swashbuckle.AspNetCore Nuget package. :-



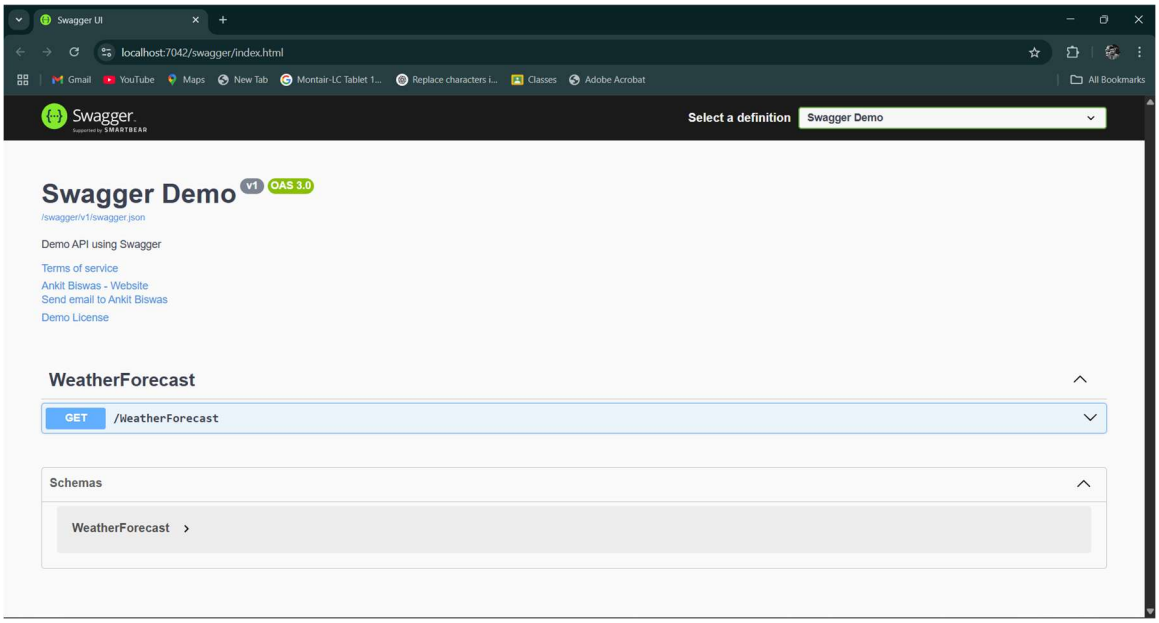
Step 3: Configure Swagger in Program.cs :-

```
using Microsoft.AspNetCore.Builder;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.OpenApi.Models;
```

```
var builder = WebApplication.CreateBuilder(args);


builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new OpenApiInfo
    {
        Title = "Swagger Demo",
        Version = "v1",
        Description = "Demo API using Swagger",
        TermsOfService = new Uri("https://example.com/terms"),
        Contact = new OpenApiContact
        {
            Name = "Ankit Biswas",
            Email = "ankit@example.com",
            Url = new Uri("https://example.com")
        },
        License = new OpenApiLicense
        {
            Name = "Demo License",
            Url = new Uri("https://example.com/license")
        }
    });
});


var app = builder.Build();
```

```
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI(c =>
    {
        c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");
    });
}

app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

OUTPUT:-



Step 4: Create Employee.cs :-

```
namespace SwaggerDemoApi.Models
{
    public class Employee
```

```
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Role { get; set; }
    }
}
```

Step 5: Create EmployeeController.cs:-

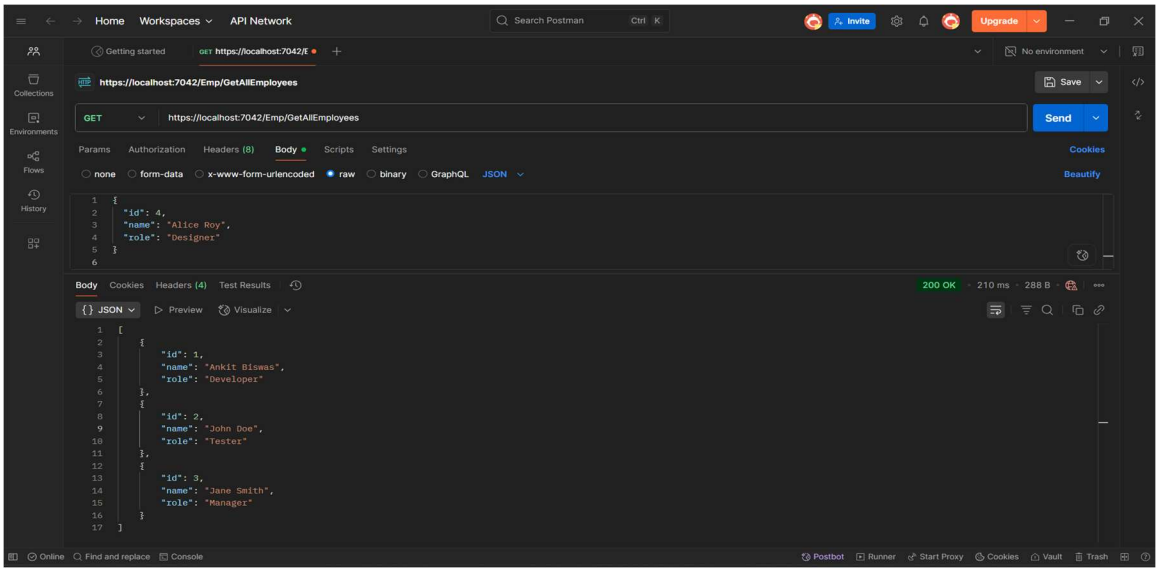
```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using SwaggerDemoApi.Models;

namespace SwaggerDemoApi.Controllers
{
    [ApiController]
    [Route("Emp")]
    public class EmployeeController : ControllerBase
    {
        private static List<Employee> employees = new List<Employee>
        {
            new Employee { Id = 1, Name = "Ankit Biswas", Role = "Developer"
},
            new Employee { Id = 2, Name = "John Doe", Role = "Tester" },
            new Employee { Id = 3, Name = "Jane Smith", Role = "Manager" }
        };

        [HttpGet("GetAllEmployees")]
        [ActionName("GetAll")]
        [ProducesResponseType(200)]
```

```
        public IActionResult GetAll()
        {
            return Ok(employees);
        }

        [HttpPost("AddEmployee")]
        [ActionName("Add")]
        [ProducesResponseType(201)]
        [ProducesResponseType(400)]
        public IActionResult Add([FromBody] Employee newEmployee)
        {
            if (newEmployee == null ||
string.IsNullOrWhiteSpace(newEmployee.Name))
                return BadRequest("Invalid employee data.");

            employees.Add(newEmployee);
            return Created("", newEmployee);
        }
    }
}
```

Step 6: Testing in Postman:-

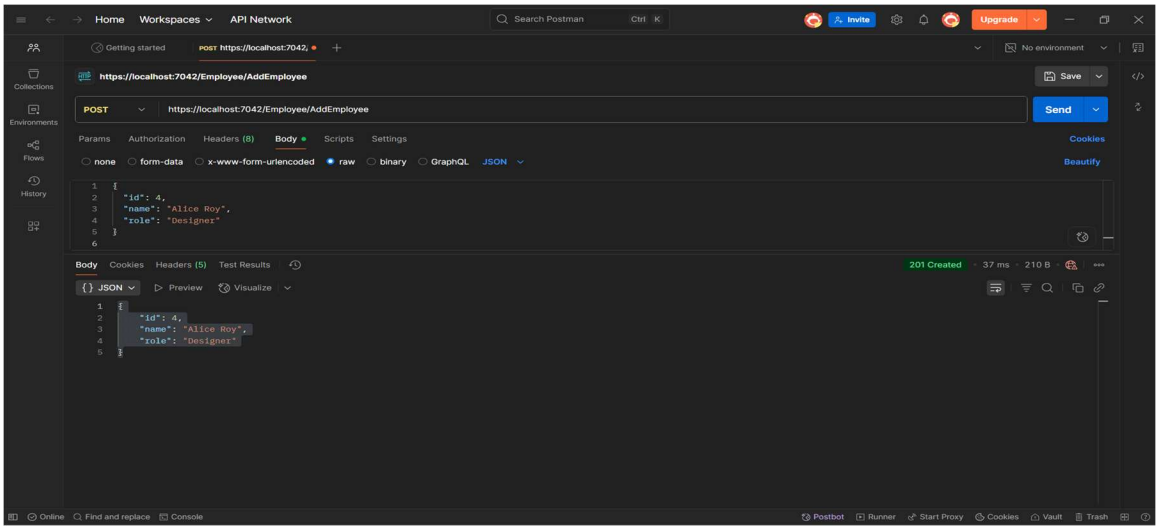
GET: https://localhost:7042/Emp/GetAllEmployees

OUTPUT:-



POST: https://localhost:7042/Emp/AddEmployee

OUTPUT:-



The POST body was sent in raw JSON format:-

```
{
  "id": 4,
  "name": "Alice Roy",
  "role": "Designer"
}
```

Web API Hands-On Lab Report – Task 3

1. Demonstrate creation of an Action method to return list of custom class entity:-

This is demonstrated by the GetEmployee() and GetStandard() action methods in EmployeeController.cs. These methods return objects of the custom class Employee, which includes nested classes like Department and a list of Skill.

Code:-

```
using Microsoft.AspNetCore.Mvc;

[HttpGet("GetEmployee")]

public IActionResult GetEmployee()

{

    var employee = GetStandardEmployee();

    return Ok(employee);

}
```

2. Model class creation, Use AllowAnonymous attribute, Use HttpGet action method:-

- **Model Classes** like **Employee**, **Department**, and **Skill** were created with properties and proper data types.
- **AllowAnonymous** attribute allows unauthenticated access to specific endpoints, but in our task we used a custom filter instead for authentication.
- **HttpGet** is used in action methods like:

Code:-

```
using Microsoft.AspNetCore.Mvc;

[HttpGet("GetStandard")]

public IActionResult GetStandard()

{ ... }
```

3. Explain the usage of FromBody attribute:-

The **[FromBody]** attribute is used to bind complex types sent in the body of an HTTP request to method parameters in POST actions.

Code:-

```
using Microsoft.AspNetCore.Mvc;
```

```
using WebApiLab3.Models;

[HttpPost("AddEmployee")]

public IActionResult AddEmployee([FromBody] Employee emp)
```

This allows the API to receive an entire Employee object from a JSON-formatted body sent via tools like Postman.

4. Read the model object from request, other than the query string parameter:-

By using [FromBody] in the POST method, we read complex objects from the request body rather than query strings or route parameters. This is especially helpful for large objects with nested properties like lists or other classes.

5. Demonstrate Custom filter

We created a custom filter using ActionFilterAttribute to implement a basic authorization mechanism that checks for the presence of a specific header (e.g., "Authorization").

Code:-

```
using Microsoft.AspNetCore.Mvc.Filters;


public class CustomAuthFilter : ActionFilterAttribute

{

    public override void OnActionExecuting(ActionExecutingContext context)

    {

        ...

    }

}
```

6. Usage of ActionFilterAttribute, OnActionExecuting method to intercept the request:-

The custom class CustomAuthFilter inherits from ActionFilterAttribute and overrides OnActionExecuting() to inspect the HTTP request before the action method executes.

Code:-

```
using Microsoft.AspNetCore.Mvc.Filters;


public override void OnActionExecuting(ActionExecutingContext context)

{
```

```
    // Intercept and validate Authorization header
```

7. Create filter for Custom exception – Requires `Microsoft.AspNetCore.Mvc.WebApiCompatShim` package. :-

To handle exceptions globally, a custom exception filter can be created that implements `IExceptionFilter`. The `WebApiCompatShim` package provides backward compatibility for Web API-style exception filters. However, for this lab, we focused on authentication filters.

Complete Code Implementation:-

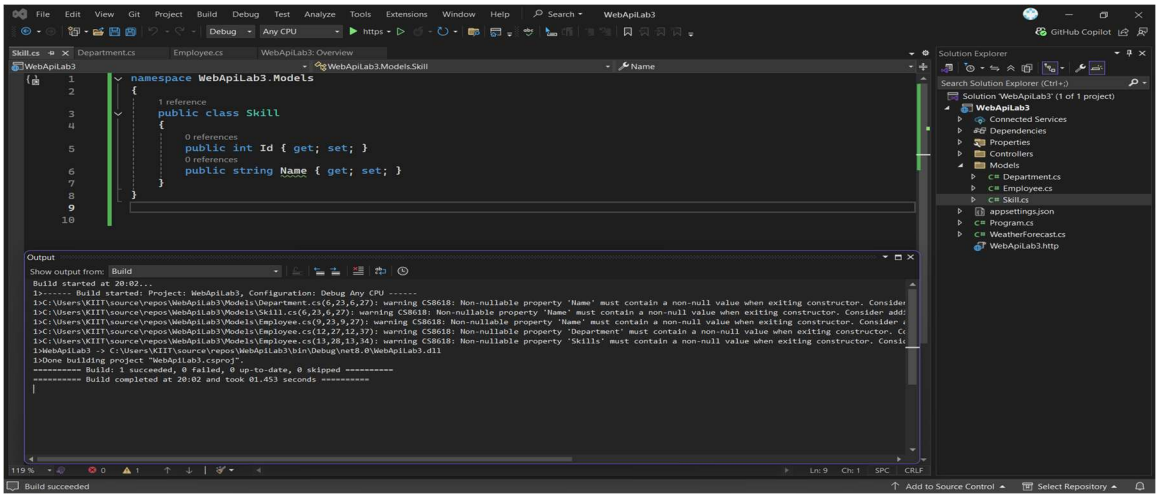
1. Model Classes:-

Skill.cs:-

Code:-

```
namespace WebApiLab3.Models

{

    public class Skill

    {

        public int Id { get; set; }

        public string Name { get; set; }

    }

}
```

Output:-

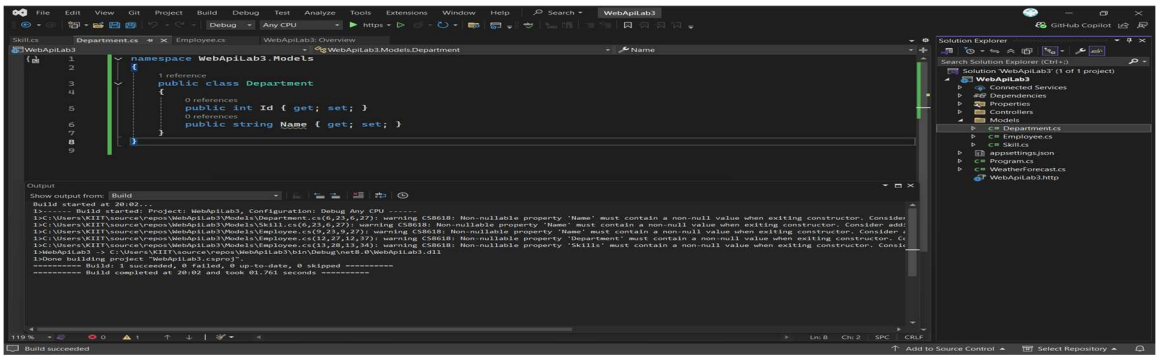


Department.cs:-

Code:-

```
namespace WebApiLab3.Models
{
    public class Department
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
}
```

Output:-



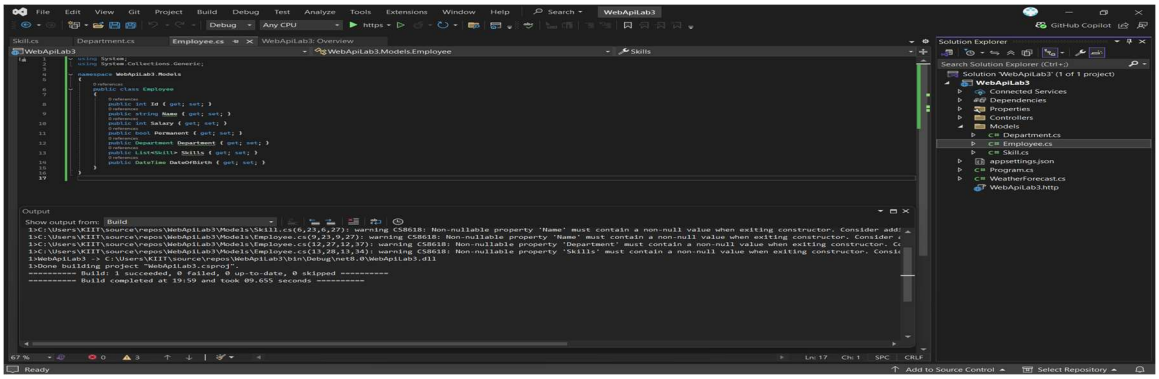
Employee.cs:-

Code:-

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

namespace WebApiLab3.Models
{
    public class Employee
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Salary { get; set; }
```

```
        public bool Permanent { get; set; }
        public Department Department { get; set; }
        public List<Skill> Skills { get; set; }
        public DateTime DateOfBirth { get; set; }
    }
}
```

Output:-



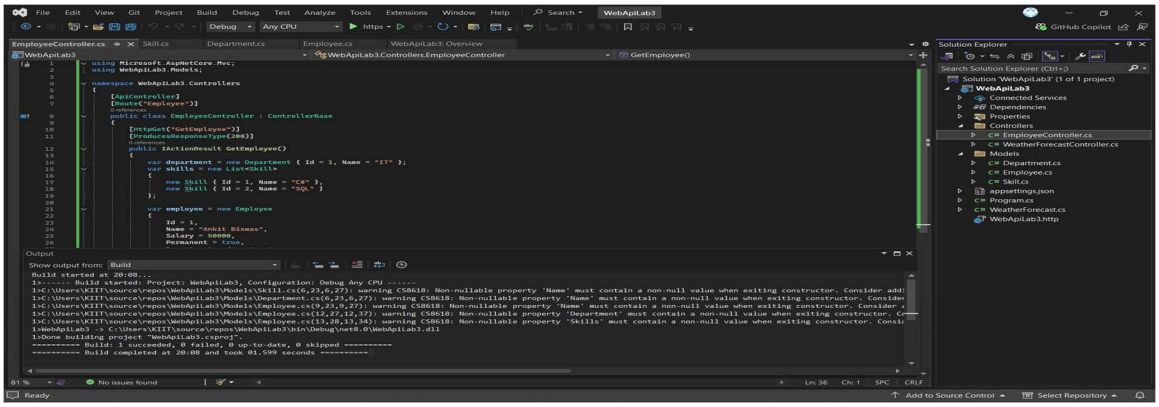
Step 2: Create Controller – EmployeeController.cs:-

Code:-

```
using Microsoft.AspNetCore.Mvc;
using WebApiLab3.Models;

namespace WebApiLab3.Controllers
{
    [ApiController]
    [Route("Employee")]
    public class EmployeeController : ControllerBase
    {
        [HttpGet("GetEmployee")]
        [ProducesResponseType(200)]
        public IActionResult GetEmployee()
        {
```

```
            var department = new Department { Id = 1, Name = "IT" };
            var skills = new List<Skill>
            {
                new Skill { Id = 1, Name = "C#" },
                new Skill { Id = 2, Name = "SQL" }
            };

            var employee = new Employee
            {
                Id = 1,
                Name = "Ankit Biswas",
                Salary = 50000,
                Permanent = true,
                Department = department,
                Skills = skills,
                DateOfBirth = new DateTime(2000, 1, 1)
            };

            return Ok(employee);
        }
    }
}
```

OUTPUT:-

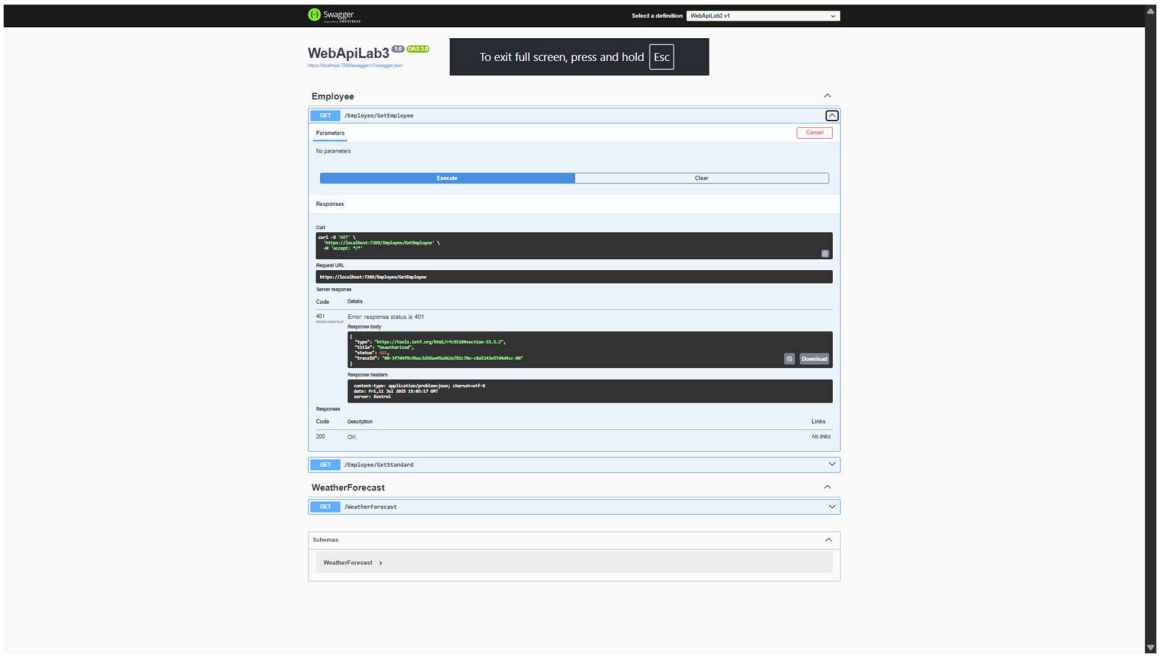


Step 3: Custom Authorization Filter :-

Code:-

```
using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;


namespace WebApiLab3.Filters

{

    public class CustomAuthorizeAttribute : Attribute, IAuthorizationFilter

    {

        public void OnAuthorization(AuthorizationFilterContext context)

        {

            var hasHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization",
out var headerValue);


            if (!hasHeader || headerValue != "Admin")

            {

                context.Result = new UnauthorizedResult();

            }

        }

    }

}
```

OUTPUT:-



Step 4: Custom Exception filter:-

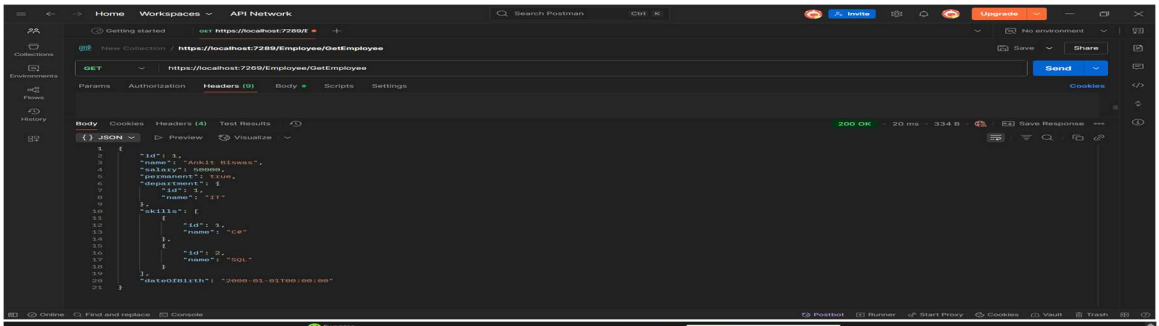
Code:-

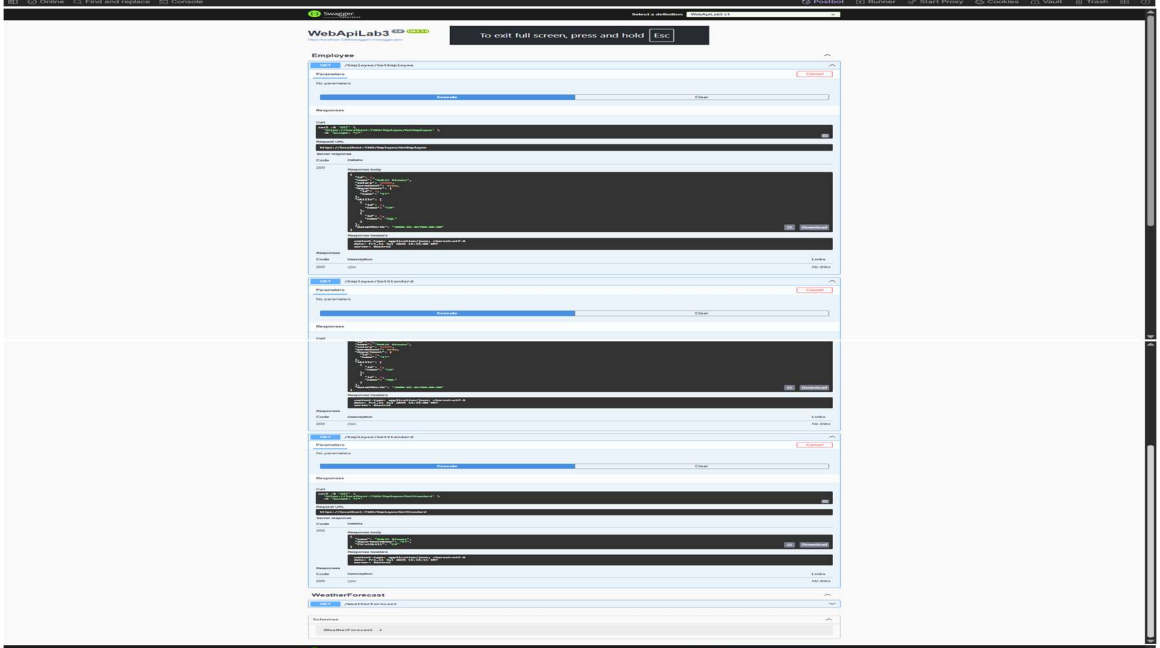
```
using Microsoft.AspNetCore.Mvc;

using WebApiLab3.Models;

namespace WebApiLab3.Controllers

{

    [ApiController]

    [Route("Employee")]

    public class EmployeeController : ControllerBase

    {

        [HttpGet("GetEmployee")]

        [ProducesResponseType(200)]

        public IActionResult GetEmployee()

        {

            var department = new Department { Id = 1, Name = "IT" };

            var skills = new List<Skill>

            {

                new Skill { Id = 1, Name = "C#" },
```

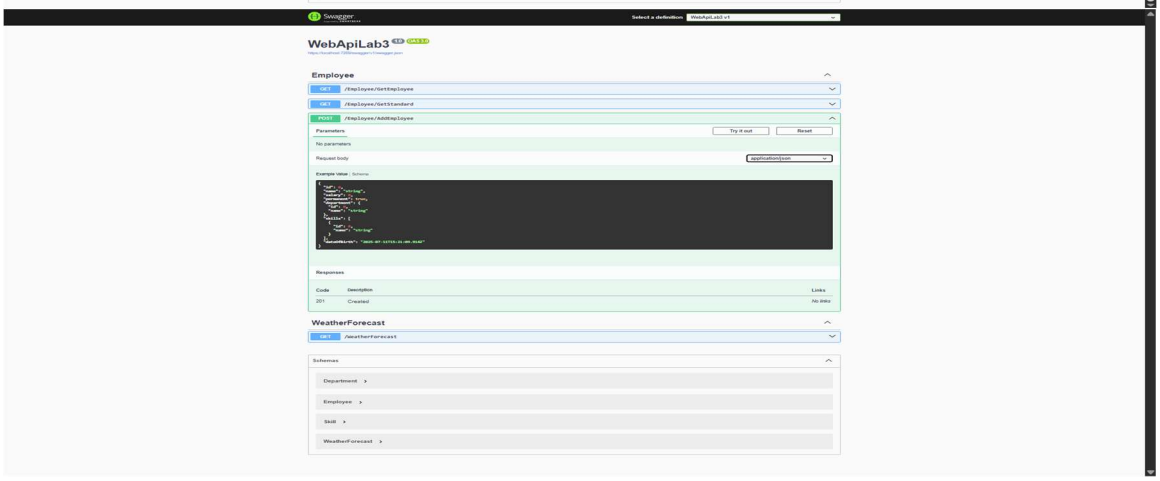
```
            new Skill { Id = 2, Name = "SQL" }
        };
        var employee = new Employee
        {
            Id = 1,
            Name = "Ankit Biswas",
            Salary = 50000,
            Permanent = true,
            Department = department,
            Skills = skills,
            DateOfBirth = new DateTime(2000, 1, 1)
        };
        return Ok(employee);
    }
    [HttpGet("GetStandard")]
    [ProducesResponseType(200)]
    public IActionResult GetStandard()
    {
        var department = new Department { Id = 1, Name = "IT" };
        var skills = new List<Skill>
        {
            new Skill { Id = 1, Name = "C#" },
            new Skill { Id = 2, Name = "SQL" }
        };

        var employee = new Employee
        {
            Id = 1,
            Name = "Ankit Biswas",
            Salary = 50000,
            Permanent = true,
```

```
                Department = department,
                Skills = skills,
                DateOfBirth = new DateTime(2000, 1, 1)
            };
            var result = new
            {
                employee.Name,
                DepartmentName = employee.Department.Name,
                FirstSkill = employee.Skills.FirstOrDefault()?.Name
            };

            return Ok(result);
        }
        [HttpPost("AddEmployee")]
        [ProducesResponseType(201)]
        public IActionResult AddEmployee([FromBody] Employee employee)
        {
                    return CreatedAtAction(nameof(GetEmployee), new { id = employee.Id },
employee);
        }
    }
}
```

OUTPUT:-


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
GET https://localhost:7269/E
New Collection / https://localhost:7269/Employee/GetEmployee
Save
Share
GET
https://localhost:7269/Employee/GetEmployee
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
Body
Cookies
Headers (4)
Test Results
200 OK
20 ms
334 B
Save Response
JSON
Preview
Visualize
{
"id": 1,
"name": "Ankit Biswas",
"salary": 50000,
"permanent": true,
"department": {
"id": 1,
"name": "IT"
},
"skills": [
{
"id": 1,
"name": "C#"
},
{
"id": 2,
"name": "SQL"
}
],
"dateOfBirth": "2000-01-01T00:00:00"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash


Swagger
Select a definition WebApiLab3 v1
WebApiLab3 OAS 3.0
To exit full screen, press and hold Esc
Employee
GET /Employee/GetEmployee
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl
Request URL
Server response
Code
Details
200
Response body
Response headers
Response content type
Download
Responses
Code
Description
Links
200
OK
No links
GET /Employee/GetStandard
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl
Request URL
Server response
Code
Details
200
Response body
Response headers
Response content type
Download
Responses
Code
Description
Links
200
OK
No links
GET /Employee/GetStandard
Parameters
Cancel
No parameters
Execute
Clear
Responses
Curl
Request URL
Server response
Code
Details
200
Response body
Response headers
Response content type
Download
Responses
Code
Description
Links
200
OK
No links
WeatherForecast
GET /WeatherForecast
Schemas
WeatherForecast


Swagger
Select a definition WebApiLab3 v1
WebApiLab3 OAS 3.0
Employee
GET /Employee/GetEmployee
GET /Employee/GetStandard
POST /Employee/AddEmployee
Parameters
Try it out
Reset
No parameters
Request body
application/json
Example Value | Schema
{
"id": 0,
"name": "string",
"salary": 0,
"permanent": true,
"department": {
"id": 0,
"name": "string"
},
"skills": [
{
"id": 0,
"name": "string"
}
],
"dateOfBirth": "2025-07-12T15:31:05.914Z"
}
Responses
Code
Description
Links
201
Created
No links
WeatherForecast
GET /WeatherForecast
Schemas
Department
Employee
Skill
WeatherForecast

Web API Hands-On Lab Report – Task 4:- Update Employee using PUT

Objectives:-

- Demonstrate creation of an Action method to perform data create, update & delete operation
    - Use FromBody attribute, extract data to custom model class using FromBody attribute, use hardcoded data to update & delete data, Use Swagger and POSTMAN to test

Updated Code for EmployeeController.cs :-

Code:-

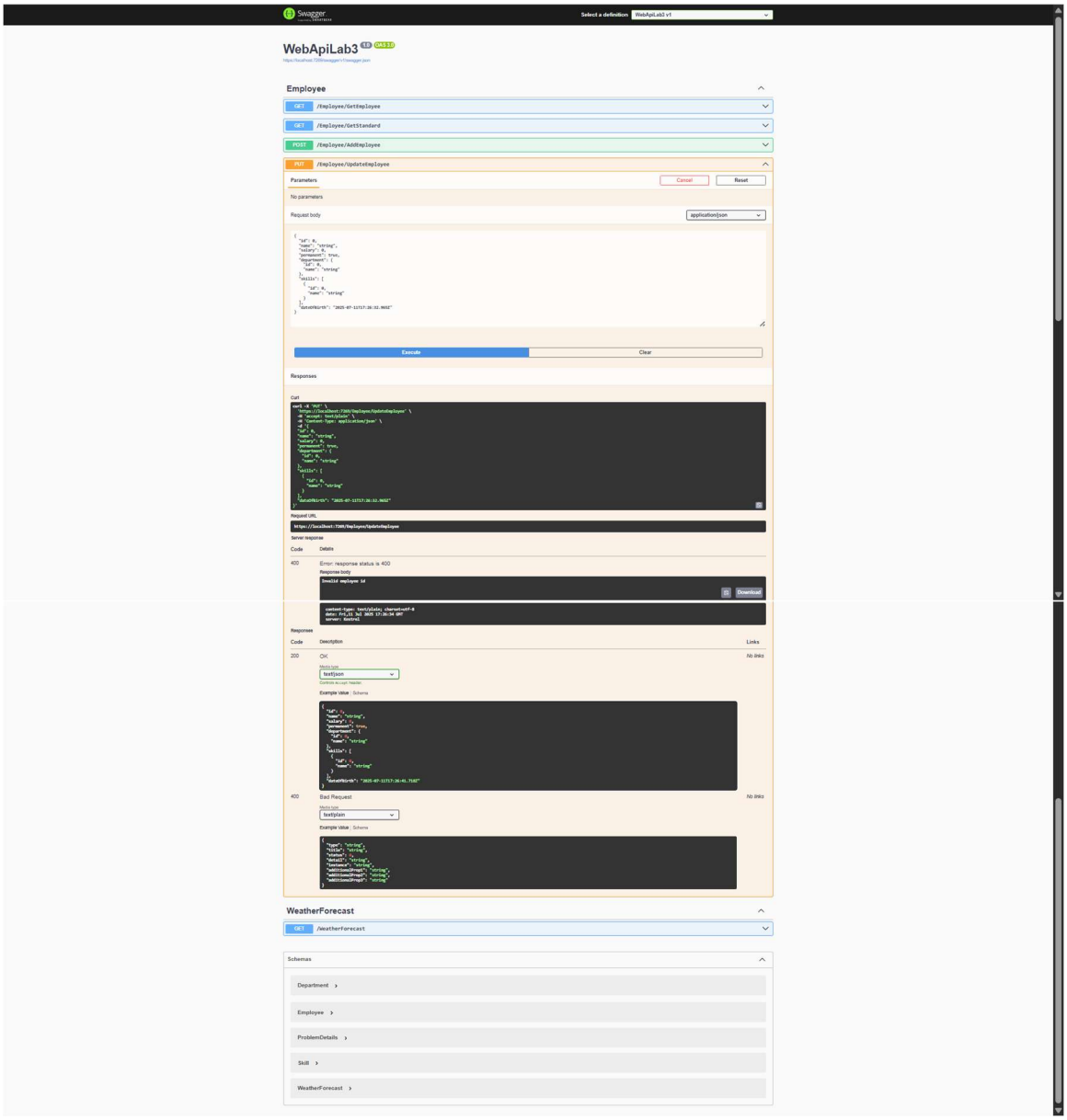
```
using Microsoft.AspNetCore.Mvc;
using WebApiLab3.Models;

namespace WebApiLab3.Controllers
{
    [ApiController]
    [Route("Employee")]
    public class EmployeeController : ControllerBase
    {
        private static List<Employee> employees = new List<Employee>
        {
            new Employee
            {
                Id = 1,
                Name = "Ankit Biswas",
                Salary = 50000,
                Permanent = true,
```

```
        Department = new Department { Id = 1, Name = "IT" },
        Skills = new List<Skill>
        {
            new Skill { Id = 1, Name = "C#" },
            new Skill { Id = 2, Name = "SQL" }
        },
        DateOfBirth = new DateTime(2000, 1, 1)
    }
};

[HttpGet("GetEmployee")]
[ProducesResponseType(200)]
public IActionResult GetEmployee()
{
    return Ok(employees.FirstOrDefault());
}

[HttpGet("GetStandard")]
[ProducesResponseType(200)]
public IActionResult GetStandard()
{
    var employee = employees.FirstOrDefault();

    var result = new
    {
        employee?.Name,
        DepartmentName = employee?.Department?.Name,
        FirstSkill = employee?.Skills?.FirstOrDefault()?.Name
    };

    return Ok(result);
```

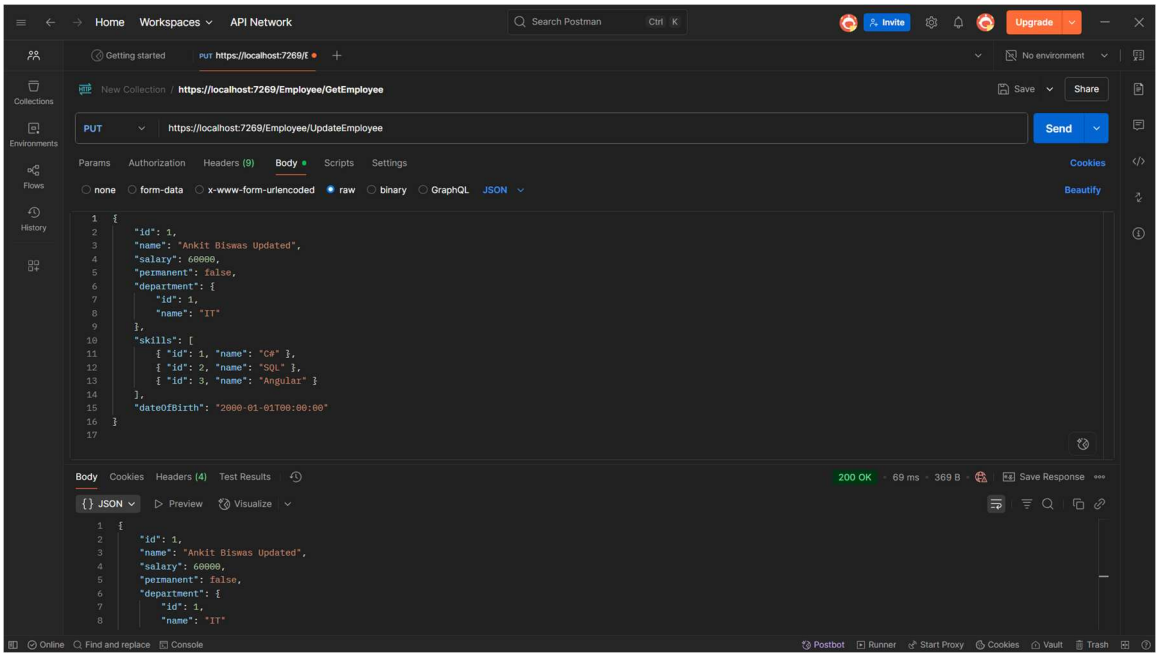
```
}

[HttpPost("AddEmployee")]
[ProducesResponseType(201)]
public IActionResult AddEmployee([FromBody] Employee employee)
{
    employees.Add(employee);
    return CreatedAtAction(nameof(GetEmployee), new { id = employee.Id }, employee);
}

[HttpPut("UpdateEmployee")]
[ProducesResponseType(200)]
[ProducesResponseType(400)]
public ActionResult<Employee> UpdateEmployee([FromBody] Employee updatedEmp)
{
    if (updatedEmp.Id <= 0)
        return BadRequest("Invalid employee id");

    var emp = employees.FirstOrDefault(e => e.Id == updatedEmp.Id);
    if (emp == null)
        return BadRequest("Employee not found");


    emp.Name = updatedEmp.Name;
    emp.Salary = updatedEmp.Salary;
    emp.Permanent = updatedEmp.Permanent;
    emp.Department = updatedEmp.Department;
    emp.Skills = updatedEmp.Skills;
    emp.DateOfBirth = updatedEmp.DateOfBirth;

    return Ok(emp);
```

```
        }
    }
}
```

TESTING WITH SWAGGER :-



After Checking in the Local Host now checking in the Postman and hence below are the results:-



-:Web API Hands-On Lab Report – Task 5:-

What is CORS?

Answer:-

CORS (Cross-Origin Resource Sharing) is a security feature in browsers that blocks requests from other origins (like frontend running on http://localhost:3000 accessing API on https://localhost:5001).
To **enable** access, you need to **enable CORS in your API.**

Updated Program.cs:-

```
using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;
```

```
var builder = WebApplication.CreateBuilder(args);


builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAll", policy =>
    {
        policy.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod();
    });
});


builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();


string securityKey = "mysuperdupersecretkeyforJWTtoken@123";
var symmetricSecurityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));



builder.Services.AddAuthentication(x =>
{
    x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
```

```
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = "mySystem",
        ValidAudience = "myUsers",
        IssuerSigningKey = symmetricSecurityKey
    };
});

var app = builder.Build();

app.UseSwagger();
app.UseSwaggerUI();
app.UseHttpsRedirection();
app.UseCors("AllowAll");
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

Create AuthController.cs in Controolers folder :-

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
```

```
using System.Security.Claims;
using System.Text;

namespace WorkingSwaggerDemo.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [AllowAnonymous]
    public class AuthController : ControllerBase
    {
        [HttpGet]
        public IActionResult GetToken()
        {
            var token = GenerateJSONWebToken(1, "Admin");
            return Ok(new { token });
        }

        private string GenerateJSONWebToken(int userId, string userRole)
        {
            var securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeyforJWTtoken@123"))
;
            var credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);

            var claims = new List<Claim>
            {
                new Claim(ClaimTypes.Role, userRole),
```

```
            new Claim("UserId", userId.ToString())
        };

        var token = new JwtSecurityToken(
            issuer: "mySystem",
            audience: "myUsers",
            claims: claims,
            expires: DateTime.Now.AddMinutes(2),
            signingCredentials: credentials);

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
  }
}
```

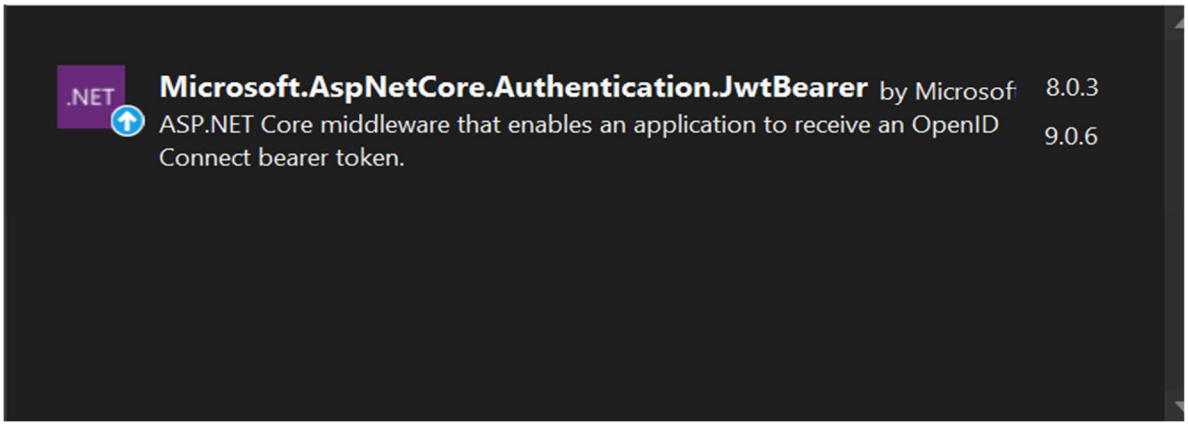
Updated Code for EmployeeController.cs in Controllers Folder for Authorization:-

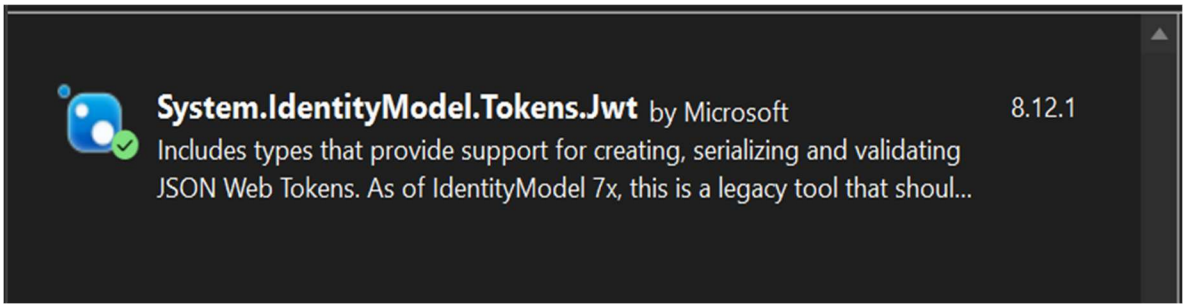
```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using WorkingSwaggerDemo.Models;

namespace WorkingSwaggerDemo.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  [Authorize(Roles = "Admin")]
  public class EmployeeController : ControllerBase
```

```
{
    private static List<Employee> employees = new List<Employee>
    {
        new Employee
        {
            Id = 1,
            Name = "Mickey",
            Salary = 60000,
            Permanent = true,
            DateOfBirth = new DateTime(1999, 10, 10),
            Department = new Department { Id = 101, Name = "Engineering" },
            Skills = new List<Skill>
            {
                new Skill { Id = 1, Name = "ASP.NET" },
                new Skill { Id = 2, Name = "Swagger" }
            }
        }
    };

    [HttpGet]
    public ActionResult<IEnumerable<Employee>> GetAll()
    {
        return Ok(employees);
    }

    [HttpGet("{id}")]
```

```
    public ActionResult<Employee> GetById(int id)
    {
        var emp = employees.FirstOrDefault(e => e.Id == id);
        if (emp == null) return NotFound("Employee not found");
        return Ok(emp);
    }

    [HttpPut("{id}")]
    public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updated)
    {
        if (id <= 0) return BadRequest("Invalid employee id");

        var emp = employees.FirstOrDefault(e => e.Id == id);
        if (emp == null) return BadRequest("Invalid employee id");

        emp.Name = updated.Name;
        emp.Salary = updated.Salary;
        emp.Permanent = updated.Permanent;
        emp.Department = updated.Department;
        emp.Skills = updated.Skills;
        emp.DateOfBirth = updated.DateOfBirth;

        return Ok(emp);
    }
  }
}
```

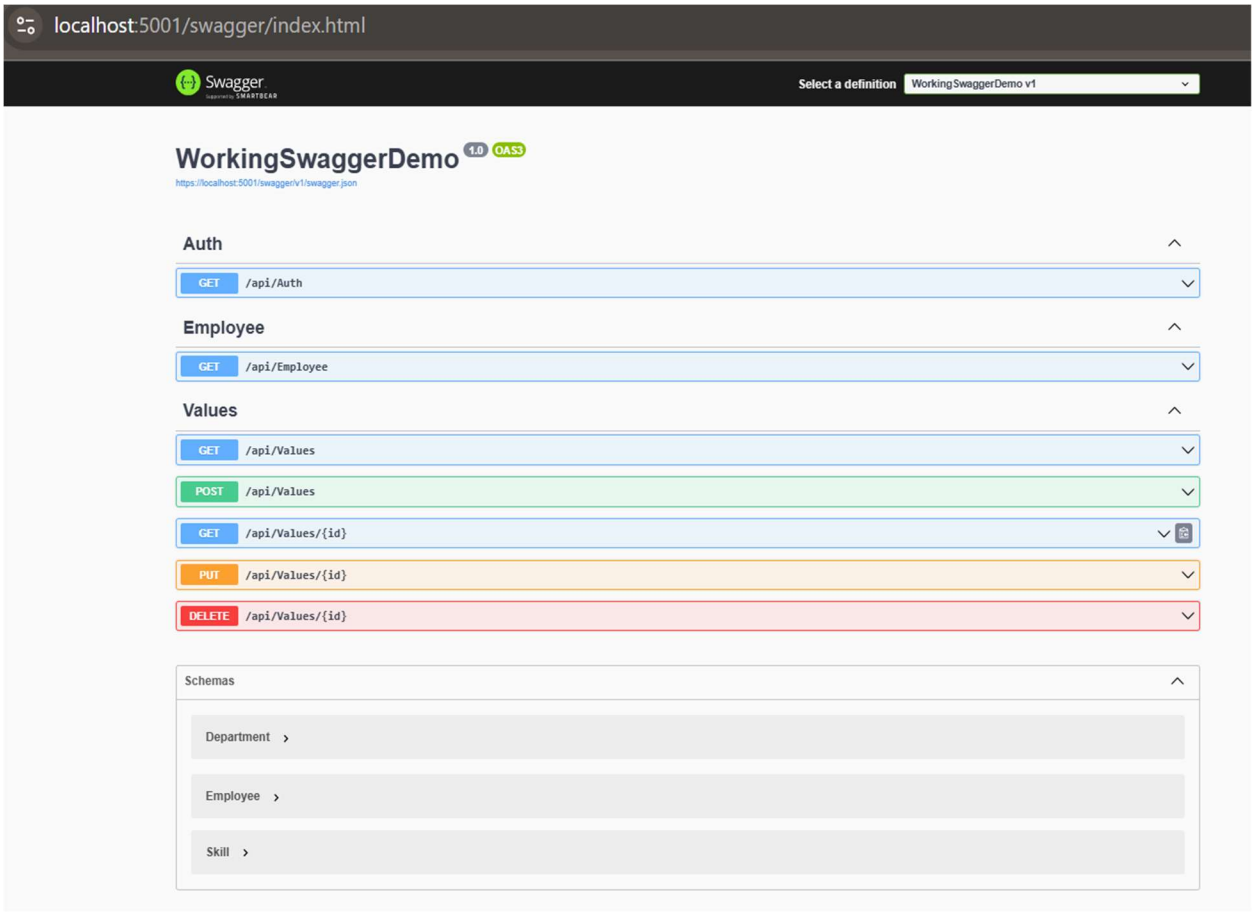
-: Added Microsoft.AspNetCore.Authentication.JwtBearer andSystem.IdentityModel.Tokens.Jwt :-





After doing this all the final view and Testing is Below:-

View Of Local Host:-

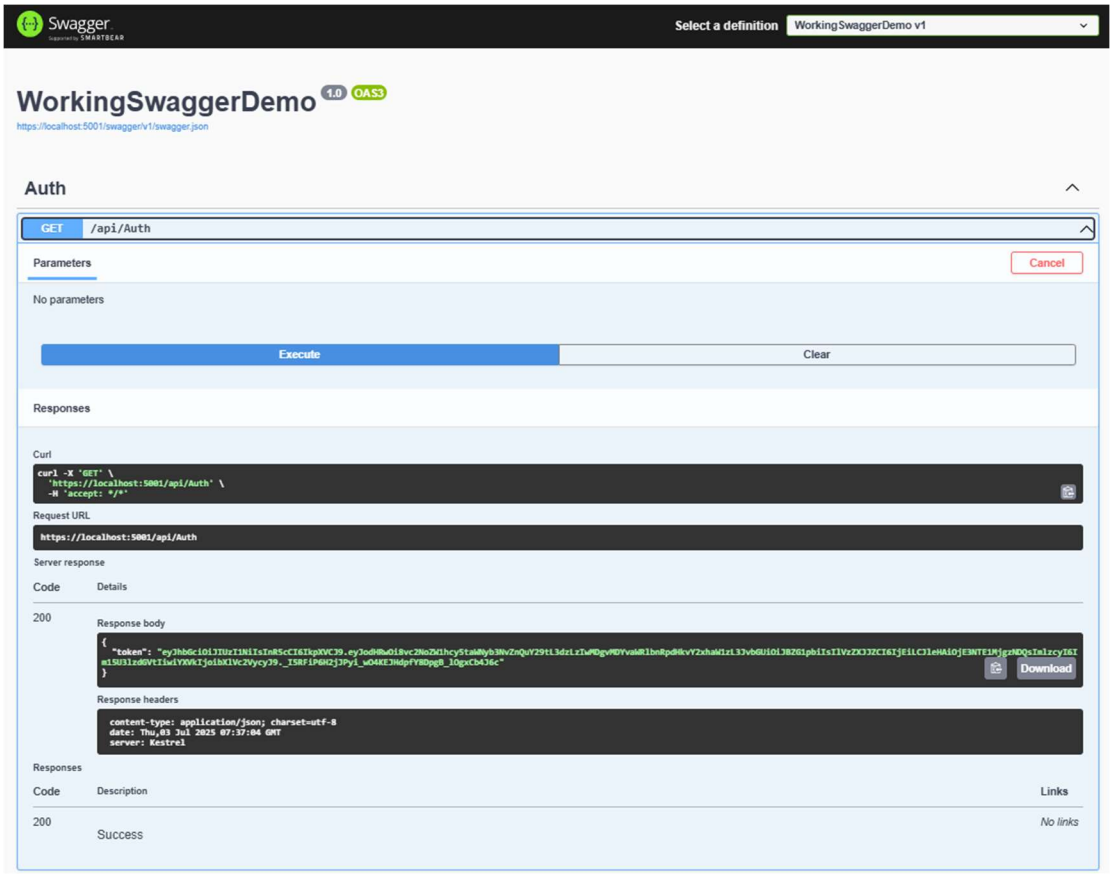


Use Postman to Test:-

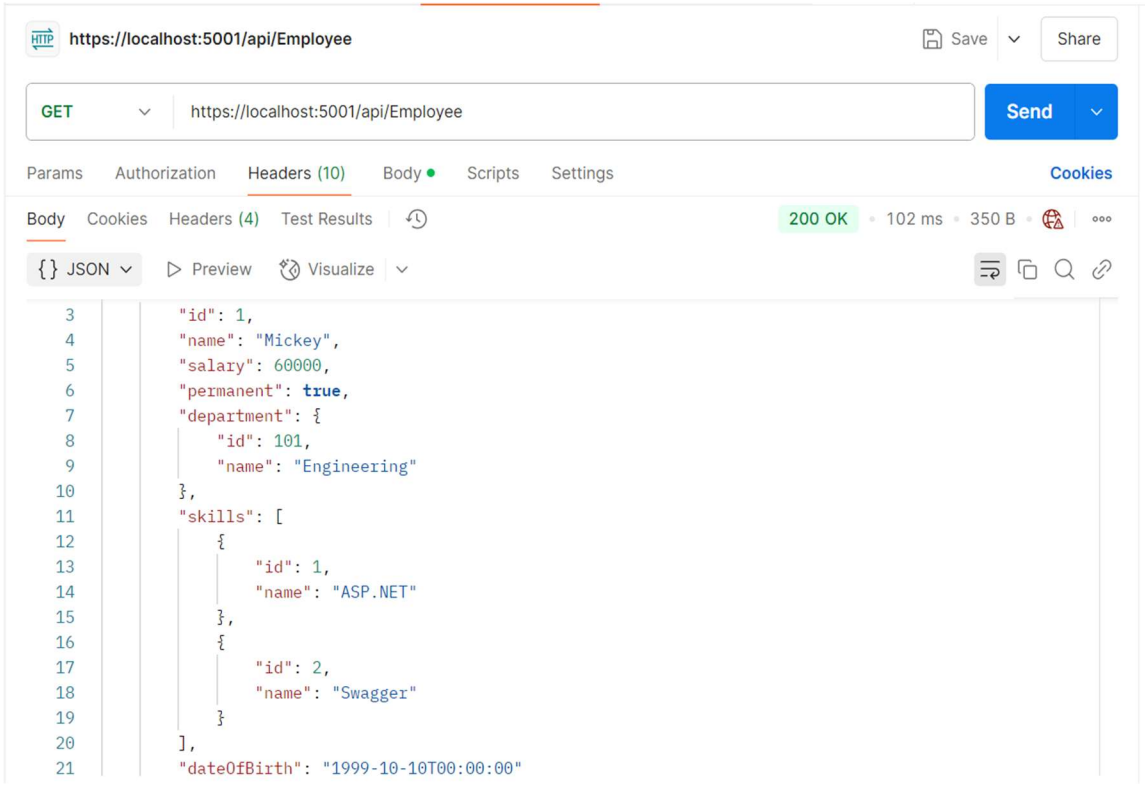
Generated Token for Test:-

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOiJBZG1pbiIsIlVzZXJJZCI6IjEiLCJleHAiOjE3NTE1MjgzNDQsImlzcyI6Im15U3lzdGVtIiwiYXVkIjoibXlVc2VycyJ9._I5RFiP6H2jJPyi_wO4KEJHdpfY8DpgB_lOgxCb4J6c

The View of local host:-



Now Final Checking in Postman and result is below:-


https://localhost:5001/api/Employee
Save
Share
GET
https://localhost:5001/api/Employee
Send
Params
Authorization
Headers (10)
Body
Scripts
Settings
Cookies
Body
Cookies
Headers (4)
Test Results
200 OK
102 ms
350 B
JSON
Preview
Visualize
"id": 1,
"name": "Mickey",
"salary": 60000,
"permanent": true,
"department": {
"id": 101,
"name": "Engineering"
},
"skills": [
{
"id": 1,
"name": "ASP.NET"
},
{
"id": 2,
"name": "Swagger"
}
],
"dateOfBirth": "1999-10-10T00:00:00"