**WEEK 4: ASP .NET CORE 8.0 WEB API**

**EXERCISE 1: FIRST WEB API USING .NET CORE**

Create a .Net core web application with API template. Use the option to create controller with Read Write permissions. Notice the ValuesController creation with Action methods corresponding to the Action verbs.

On creation of the Web API, execute the application and check if the GET action method result is returned as expected.

**Program.cs**

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

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();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

StudentApiProj.http

@StudentApiProj\_HostAddress = http://localhost:5081

GET {{StudentApiProj\_HostAddress}}/weatherforecast/

Accept: application/json

###

Weatherforecast.cs

namespace StudentApiProj

{

public class WeatherForecast

{

public DateOnly Date { get; set; }

public int TemperatureC { get; set; }

public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);

public string? Summary { get; set; }

}

}

**StudentController.cs**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace StudentApiProj.Controllers

{

[ApiController]

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

public class StudentController : ControllerBase

{

private static List<string> students = new List<string> { "Gayathri", "Ram", "Arun" };

[HttpGet]

public IActionResult Get()

{

return Ok(students);

}

[HttpPost]

public IActionResult Post([FromBody] string name)

{

students.Add(name);

return Ok($"Added student: {name}");

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] string name)

{

if (id < 0 || id >= students.Count)

return BadRequest("Invalid student ID");

students[id] = name;

return Ok($"Updated student at index {id} to {name}");

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

if (id < 0 || id >= students.Count)

return BadRequest("Invalid student ID");

students.RemoveAt(id);

return Ok($"Deleted student at index {id}");

}

}

}

**WeatherForecastController.cs**

using Microsoft.AspNetCore.Mvc;

namespace StudentApiProj.Controllers

{

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet(Name = "GetWeatherForecast")]

public IEnumerable<WeatherForecast> Get()

{

return Enumerable.Range(1, 5).Select(index => new WeatherForecast

{

Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = Summaries[Random.Shared.Next(Summaries.Length)]

})

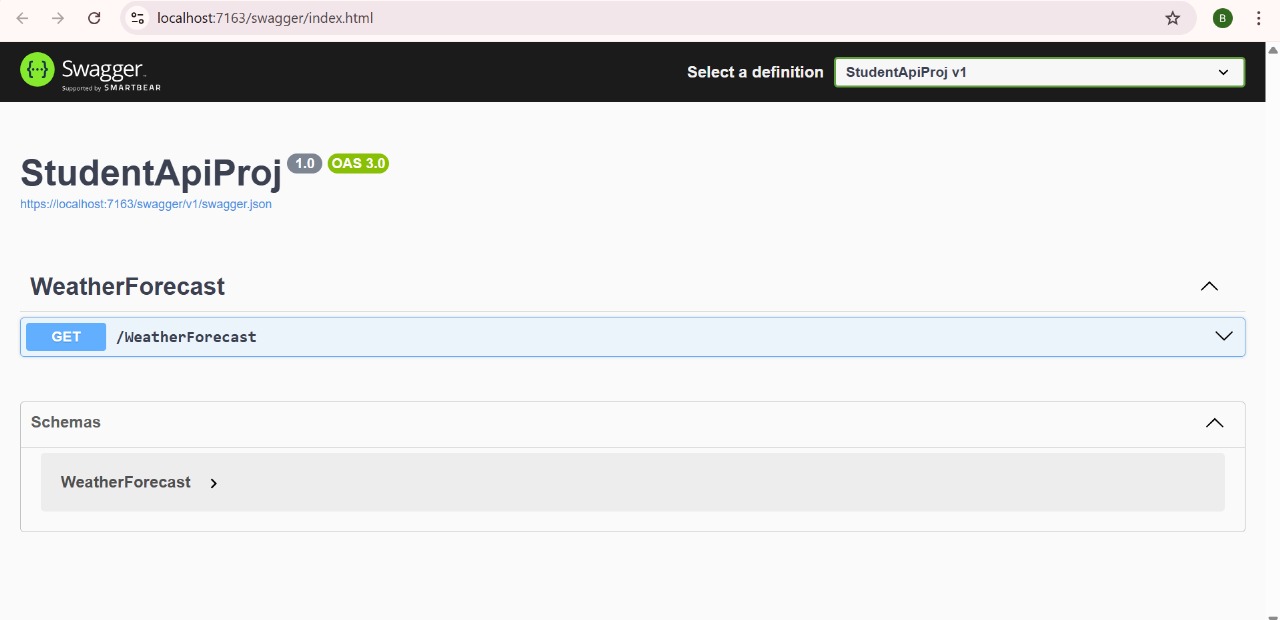
.ToArray();

}

}

}

OUTPUT:



**EXERCISE 2: WEB API USING .NET CORE WITH SWAGGER**

**Program.cs**

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

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();

StudentController.cs

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace StudentApiProj.Controllers

{

[ApiController]

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

public class StudentController : ControllerBase

{

private static List<string> students = new List<string> { "Gayathri", "Ram", "Arun" };

[HttpGet]

public IActionResult Get()

{

return Ok(students);

}

[HttpPost]

public IActionResult Post([FromBody] string name)

{

if (string.IsNullOrWhiteSpace(name))

return BadRequest("Name cannot be empty");

students.Add(name);

return Ok($"Added student: {name}");

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] string name)

{

if (id < 0 || id >= students.Count)

return BadRequest("Invalid student ID");

students[id] = name;

return Ok($"Updated student at index {id} to {name}");

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

if (id < 0 || id >= students.Count)

return BadRequest("Invalid student ID");

var removed = students[id];

students.RemoveAt(id);

return Ok($"Deleted student: {removed}");

}

}

}

**launchSettings.json**

"profiles": {

"StudentApiProj": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"launchUrl": "swagger",

"applicationUrl": "https://localhost:7163;http://localhost:5163",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

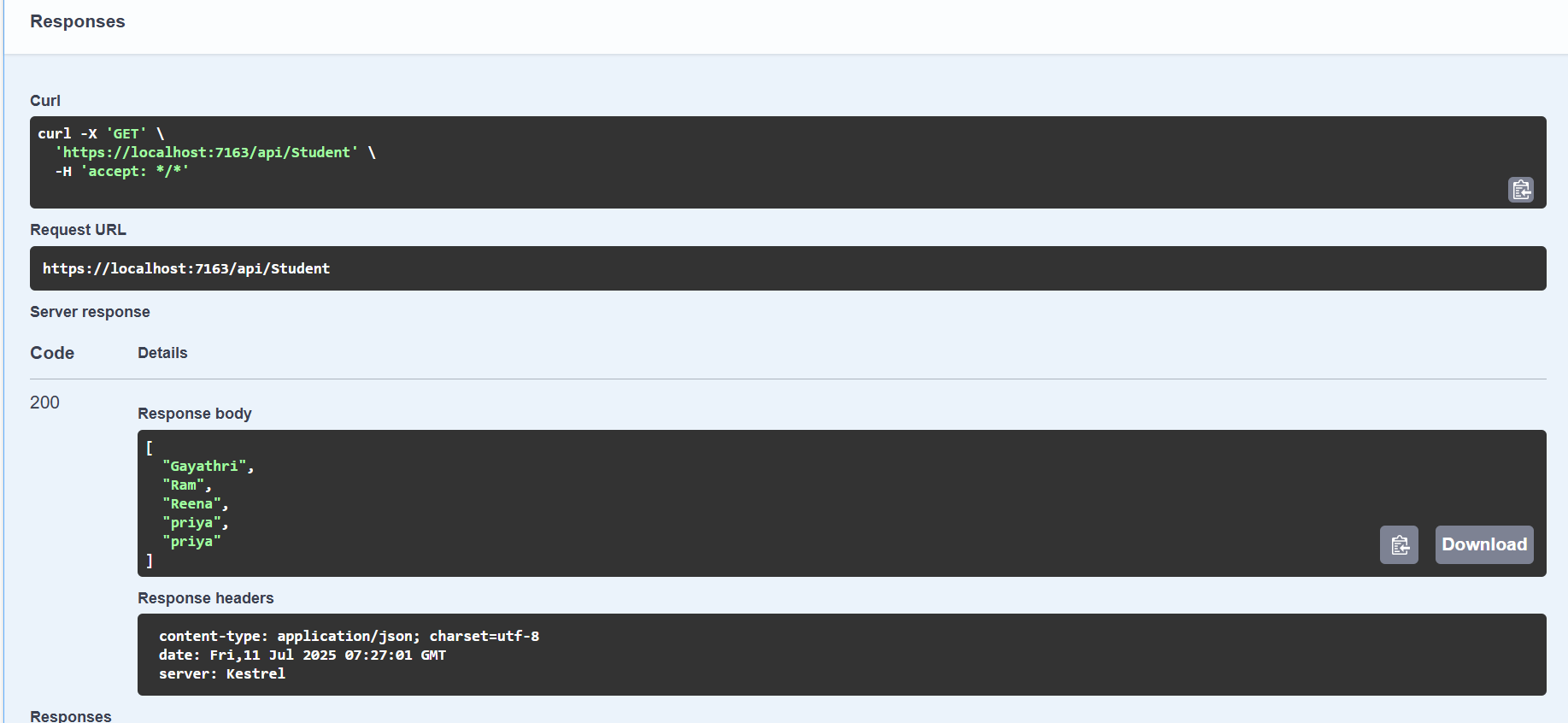
}

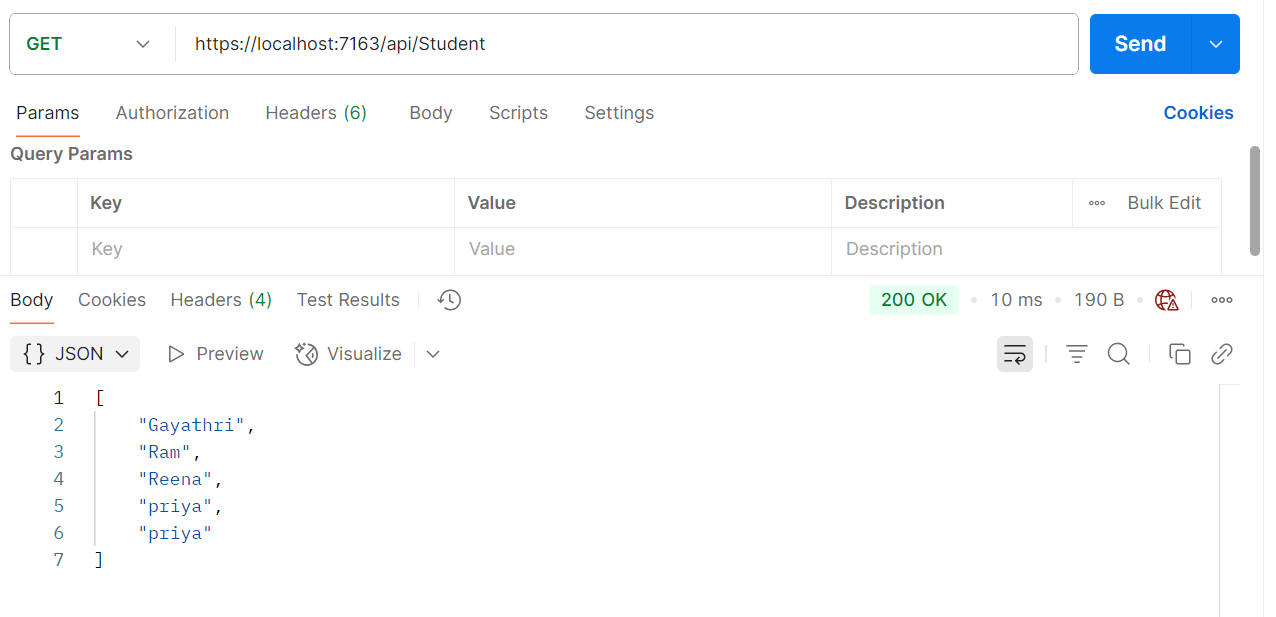
}

}

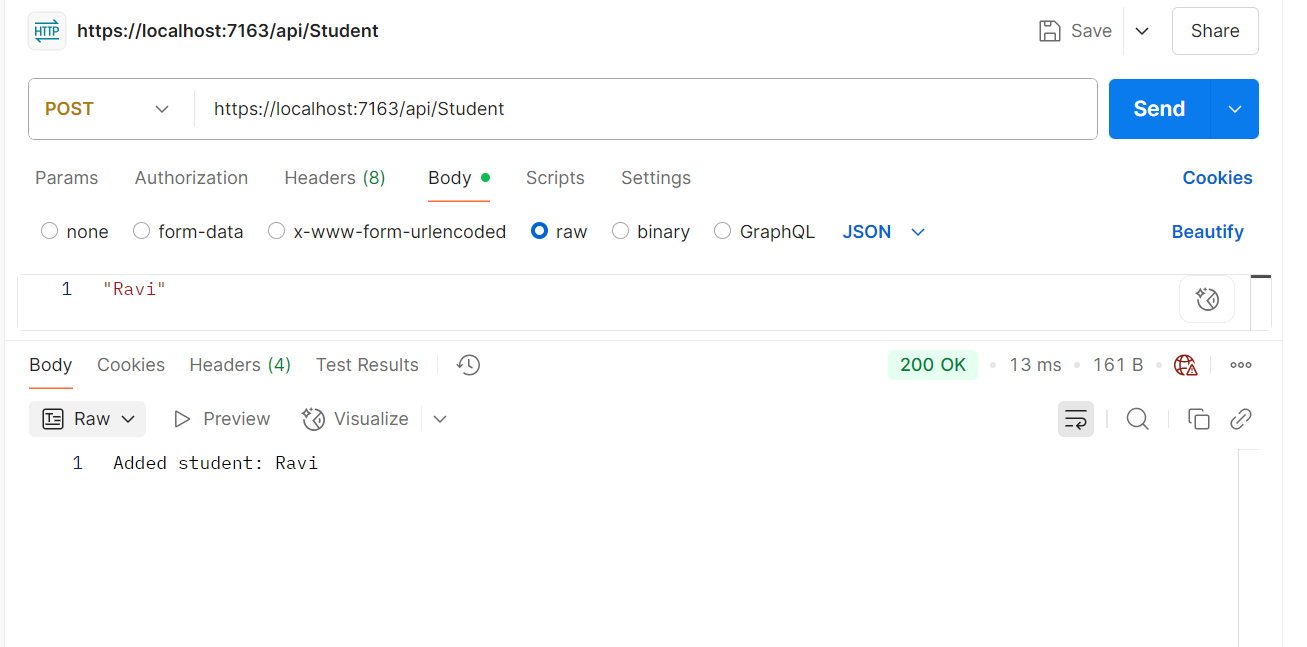
OUTPUT:

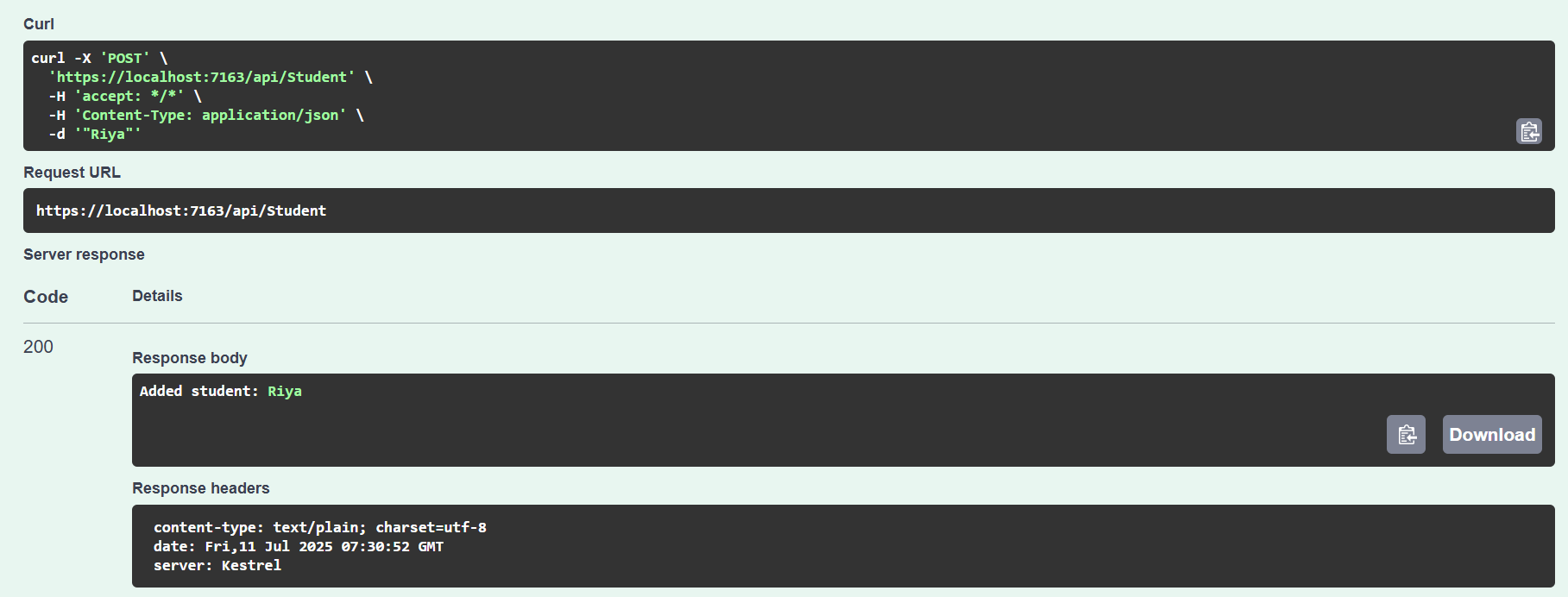
GET





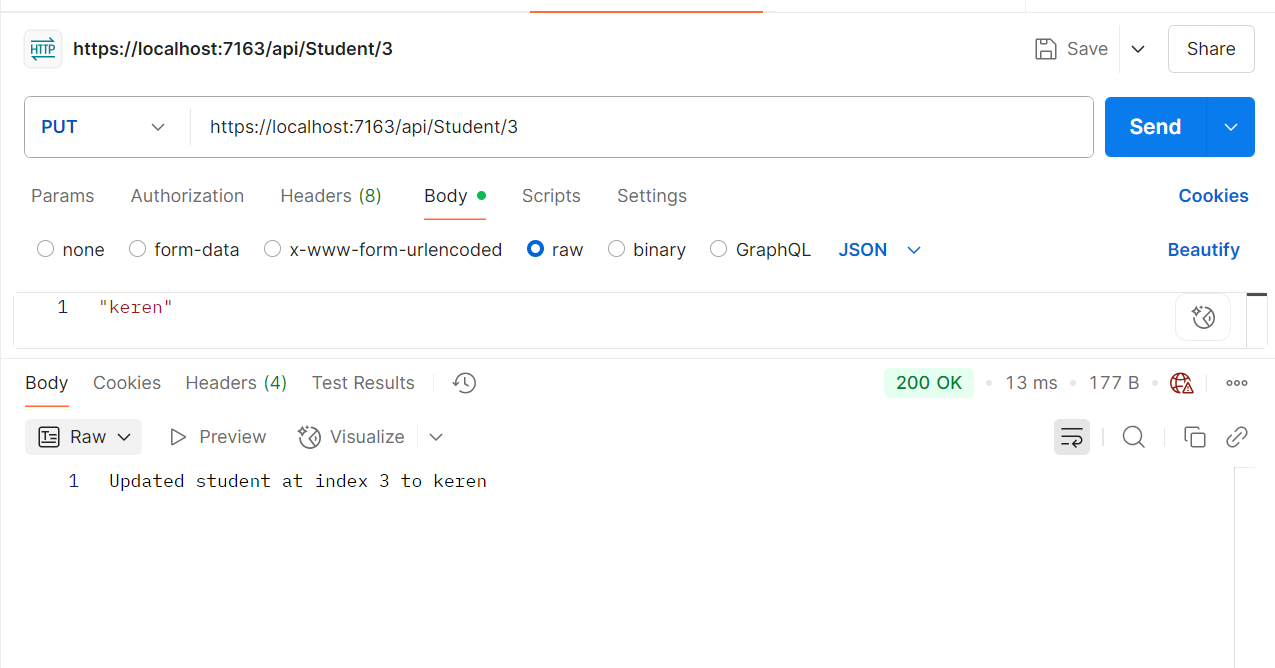
POST





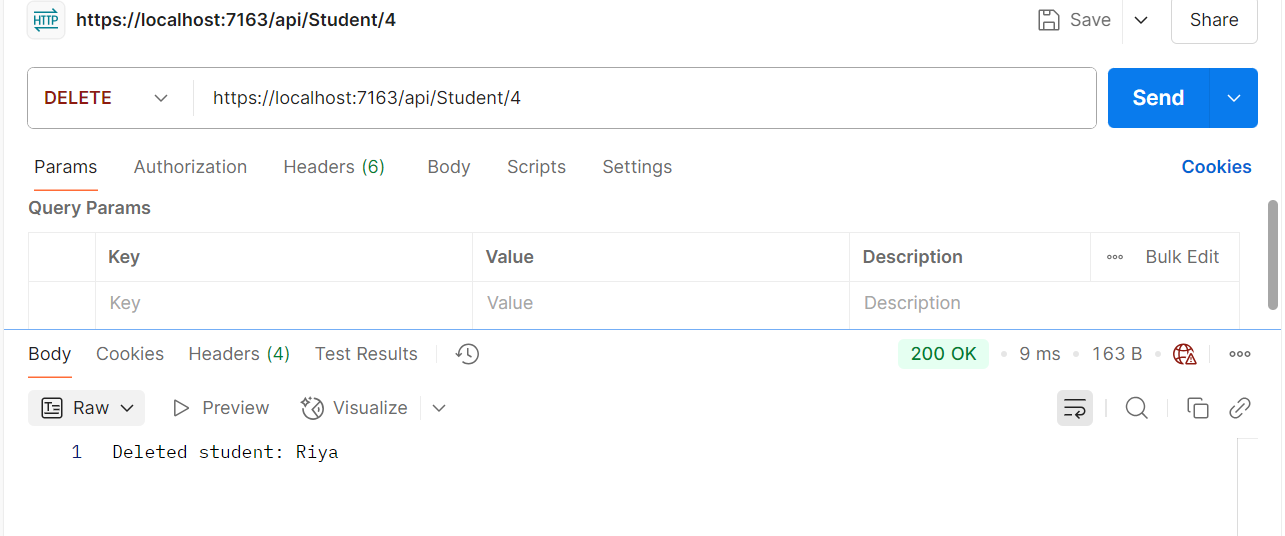
PUT

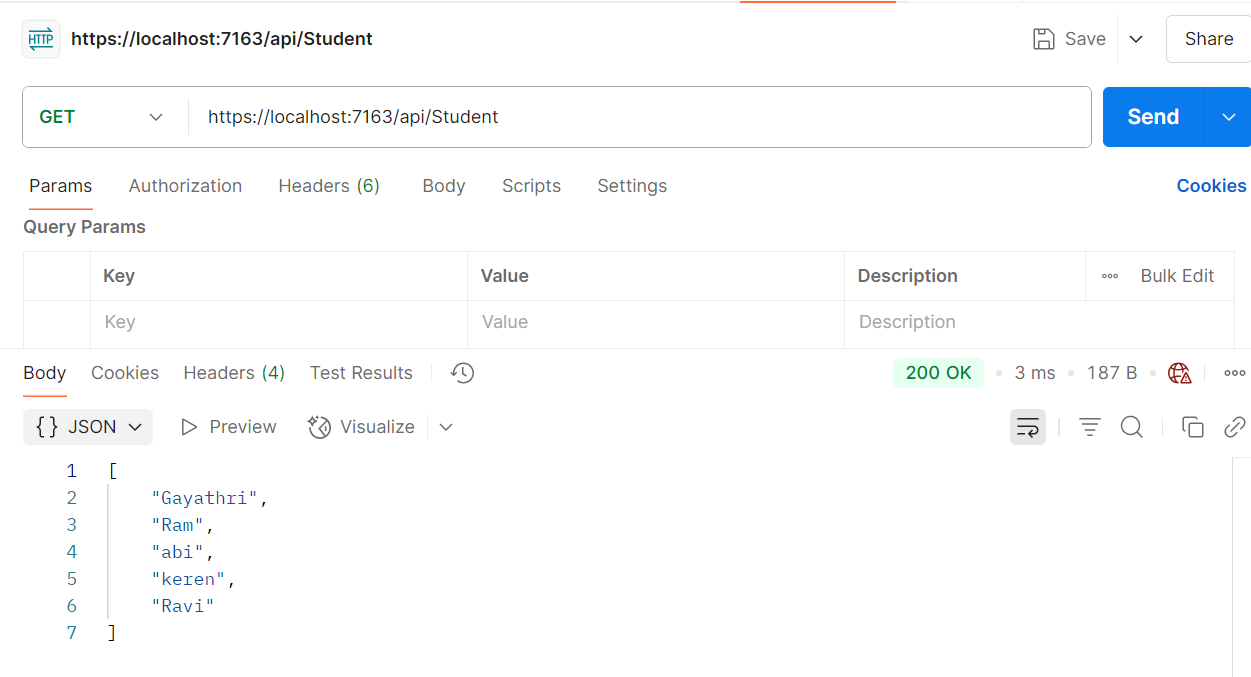


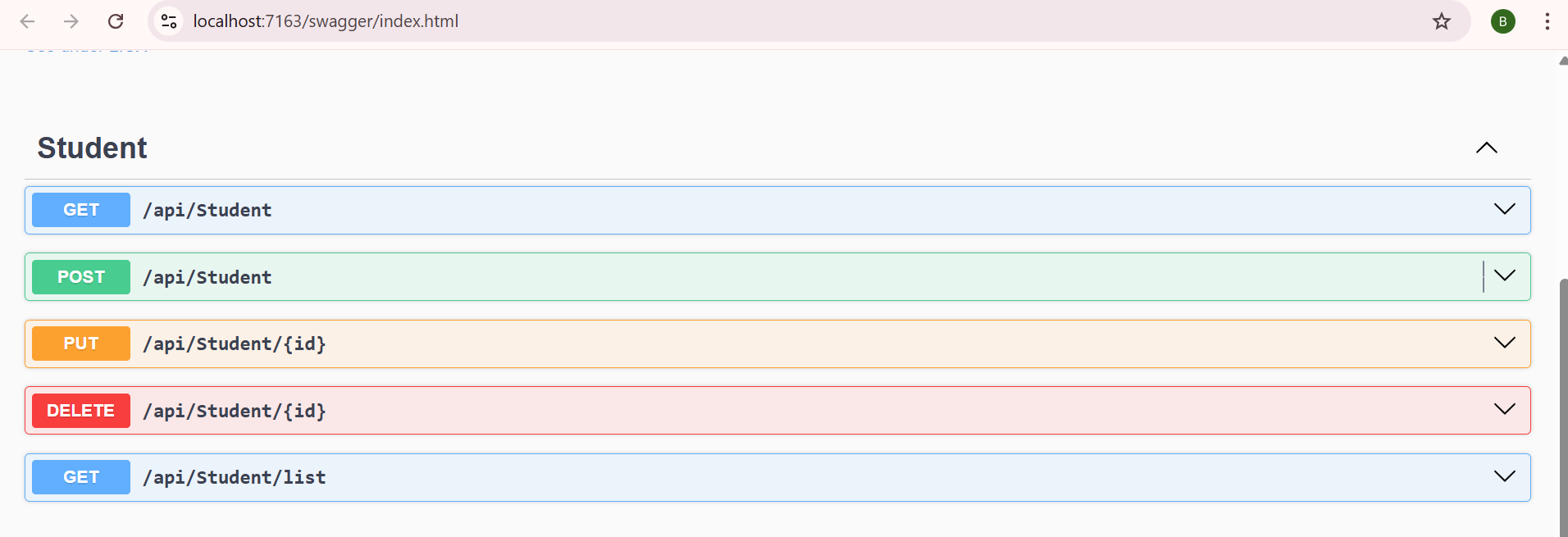


DELETE









**EXERCISE 3: WEB API USING CUSTOM MODEL CLASS**

**CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeApiDemo.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

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

if (!hasHeader)

{

context.Result = new JsonResult(new { message = "Invalid request - No Auth token" })

{

StatusCode = 400

};

return;

}

if (!token.ToString().Contains("Bearer"))

{

context.Result = new JsonResult(new { message = "Invalid request - Token present but Bearer unavailable" })

{

StatusCode = 400

};

return;

}

base.OnActionExecuting(context);

}

}

}

**CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

using System.IO;

namespace EmployeeApiDemo.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string logMessage = $"[{DateTime.Now}] Exception: {context.Exception.Message}{Environment.NewLine}";

// Write to logs.txt in output directory

var logPath = @"D:\Logs\logs.txt";

Directory.CreateDirectory(Path.GetDirectoryName(logPath));

File.AppendAllText(logPath, logMessage);

context.Result = new ObjectResult(new { message = "An unexpected error occurred." })

{

StatusCode = 500

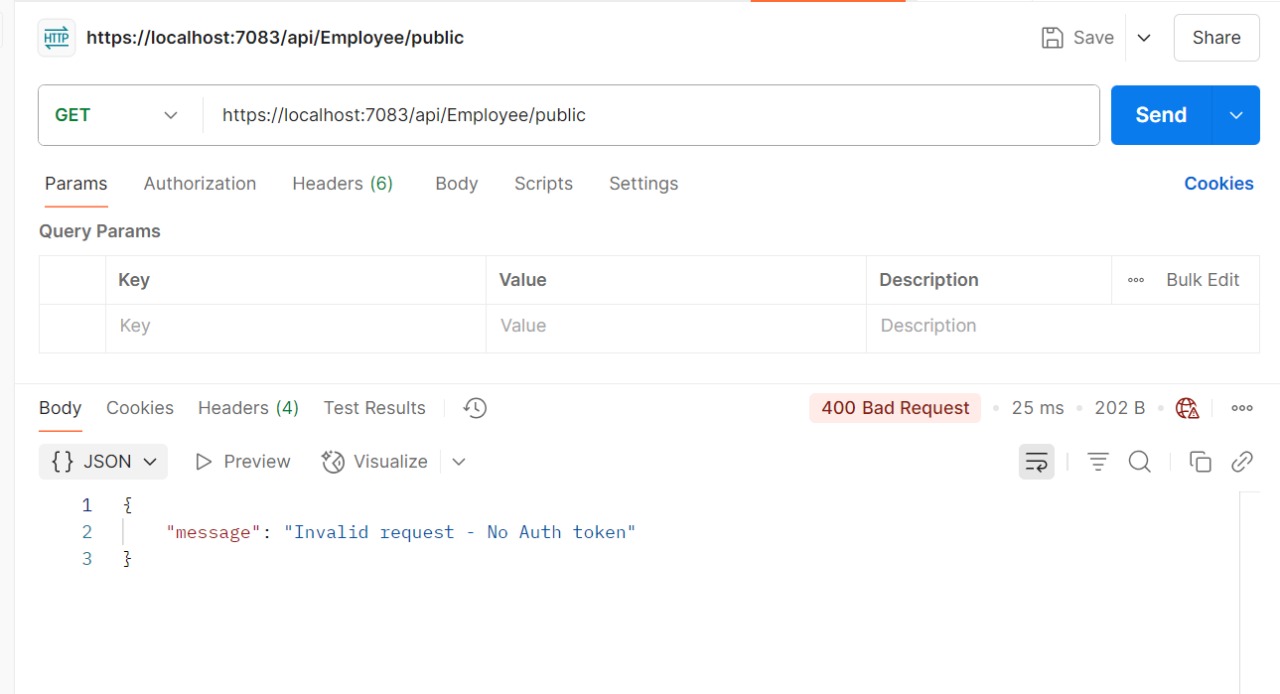
};

}

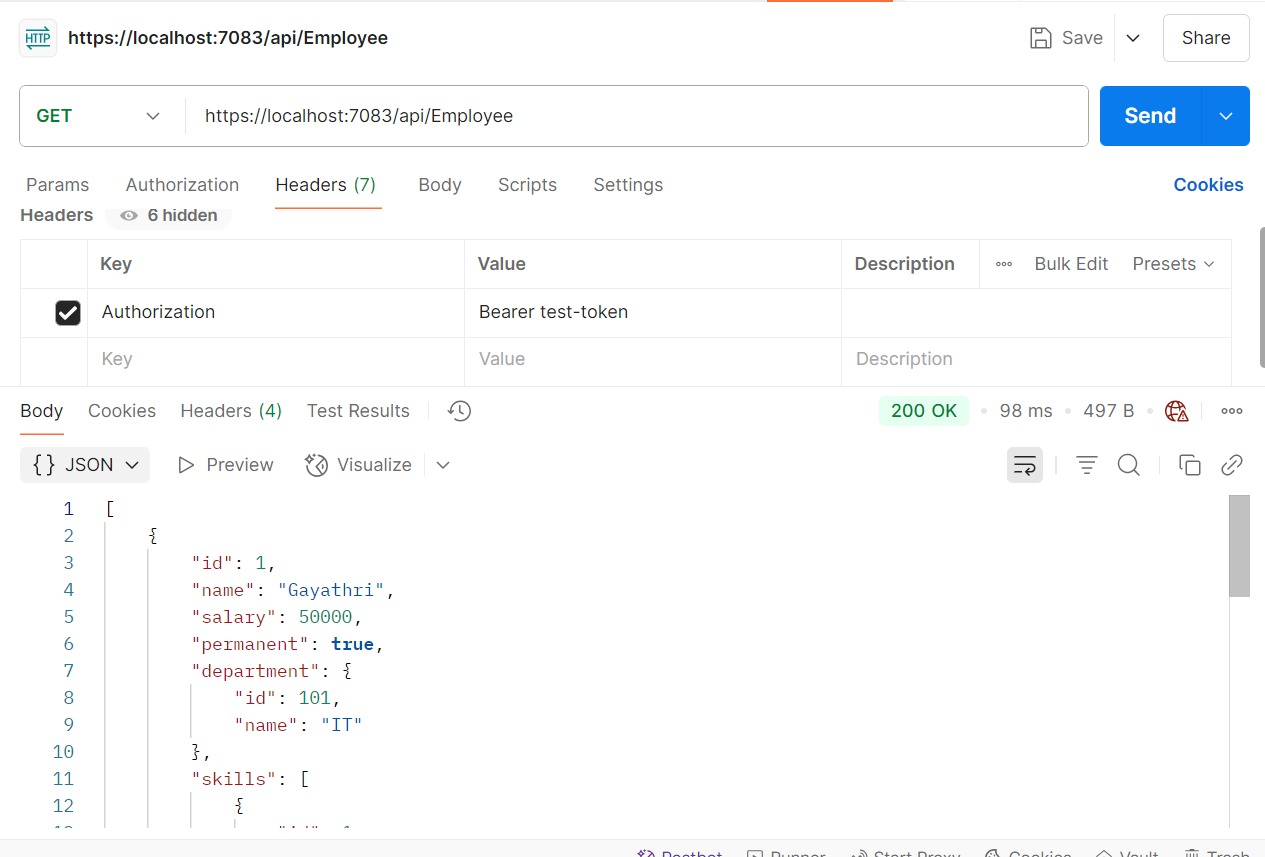
}

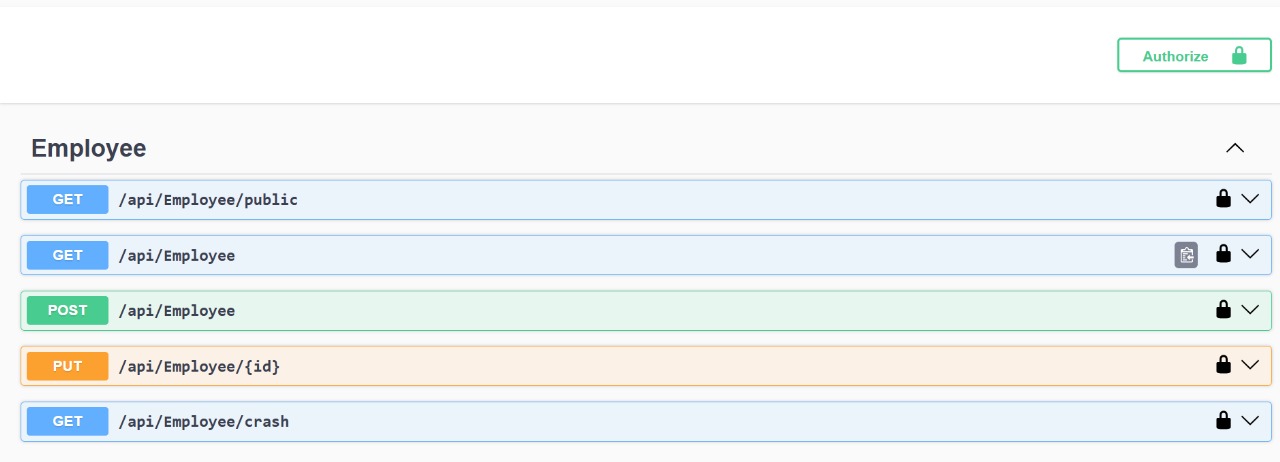
}

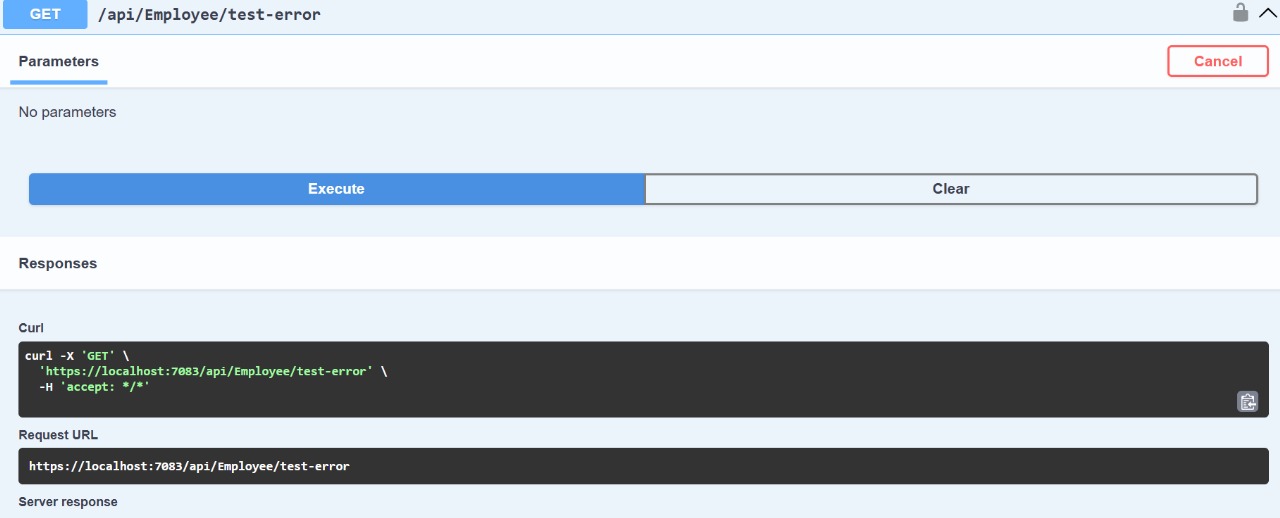
OUTPUT:











**EXERCISE 4: WEB API CRUD OPERATION**

**EmployeeController.cs**

using EmployeeApiDemo.Models;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

namespace EmployeeApiDemo.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> employeeList = new List<Employee>

{

new Employee

{

Id = 1,

Name = "Gayathri",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 101, Name = "IT" },

Skills = new List<Skill>

{

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

new Skill { Id = 2, Name = "SQL" }

},

DateOfBirth = new DateTime(2000, 5, 15)

},

new Employee

{

Id = 2,

Name = "Arun",

Salary = 45000,

Permanent = false,

Department = new Department { Id = 102, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "Excel" }

},

DateOfBirth = new DateTime(1998, 10, 25)

}

};

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> GetAll()

{

return Ok(employeeList);

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

public ActionResult<Employee> Create([FromBody] Employee newEmployee)

{

newEmployee.Id = employeeList.Max(e => e.Id) + 1;

employeeList.Add(newEmployee);

return CreatedAtAction(nameof(GetAll), new { id = newEmployee.Id }, newEmployee);

}

[HttpPut("{id}")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

{

if (id <= 0)

return BadRequest("Invalid employee id");

var employee = employeeList.FirstOrDefault(e => e.Id == id);

if (employee == null)

return BadRequest("Invalid employee id");

employee.Name = updatedEmployee.Name;

employee.Salary = updatedEmployee.Salary;

employee.Permanent = updatedEmployee.Permanent;

employee.Department = updatedEmployee.Department;

employee.Skills = updatedEmployee.Skills;

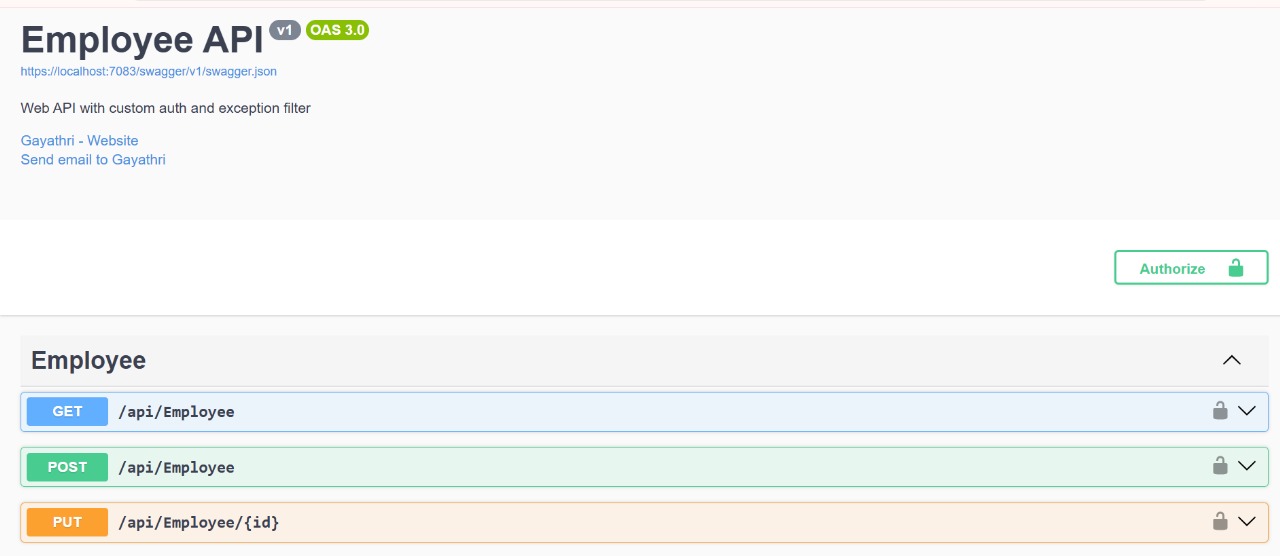
employee.DateOfBirth = updatedEmployee.DateOfBirth;

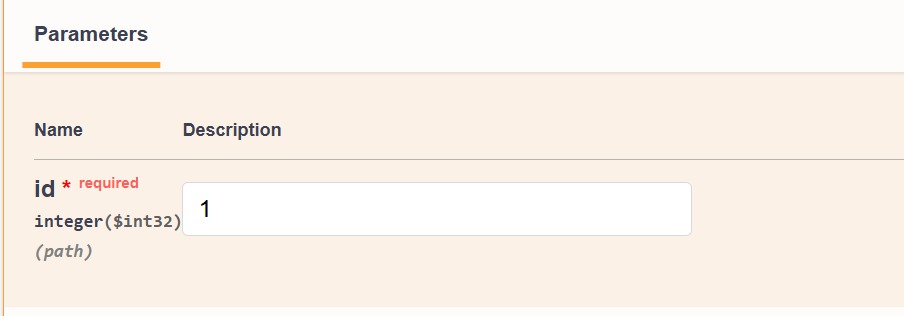
return Ok(employee);

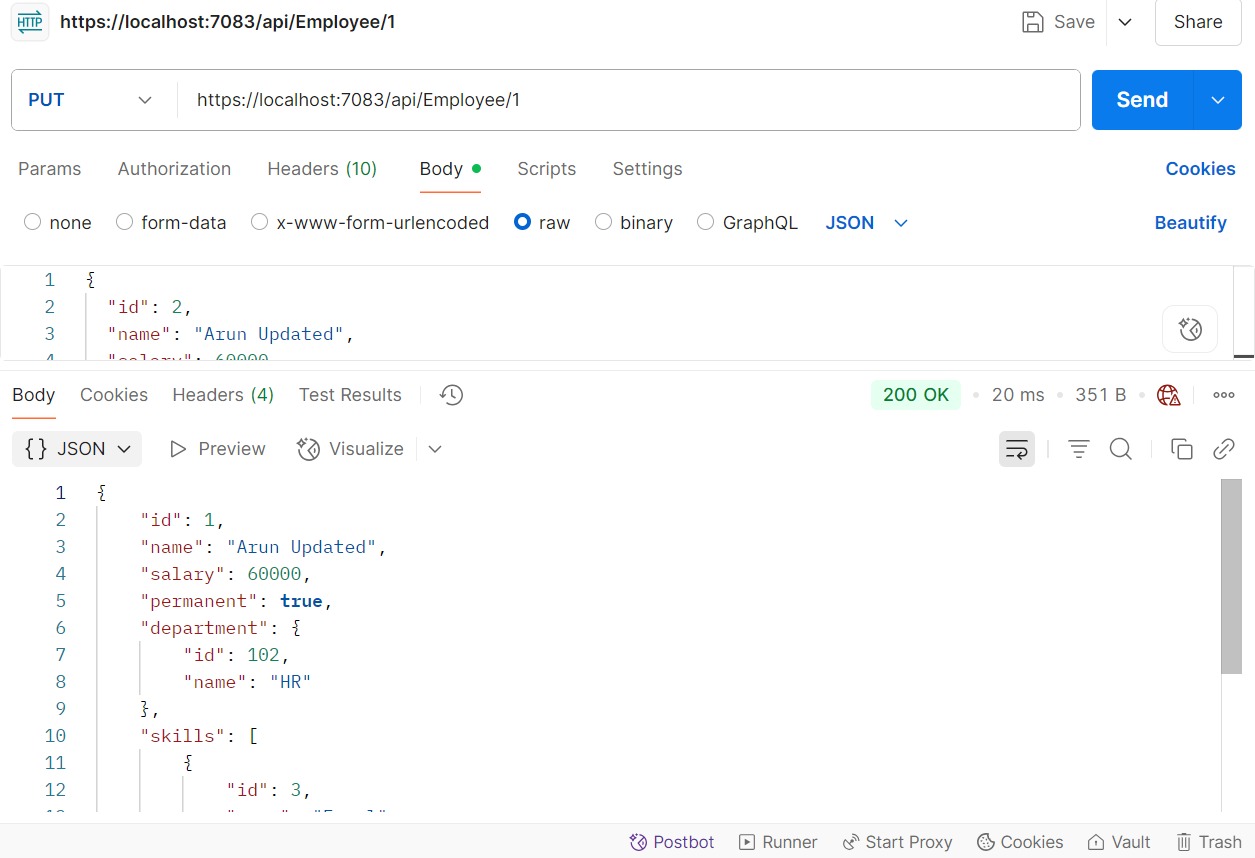
}

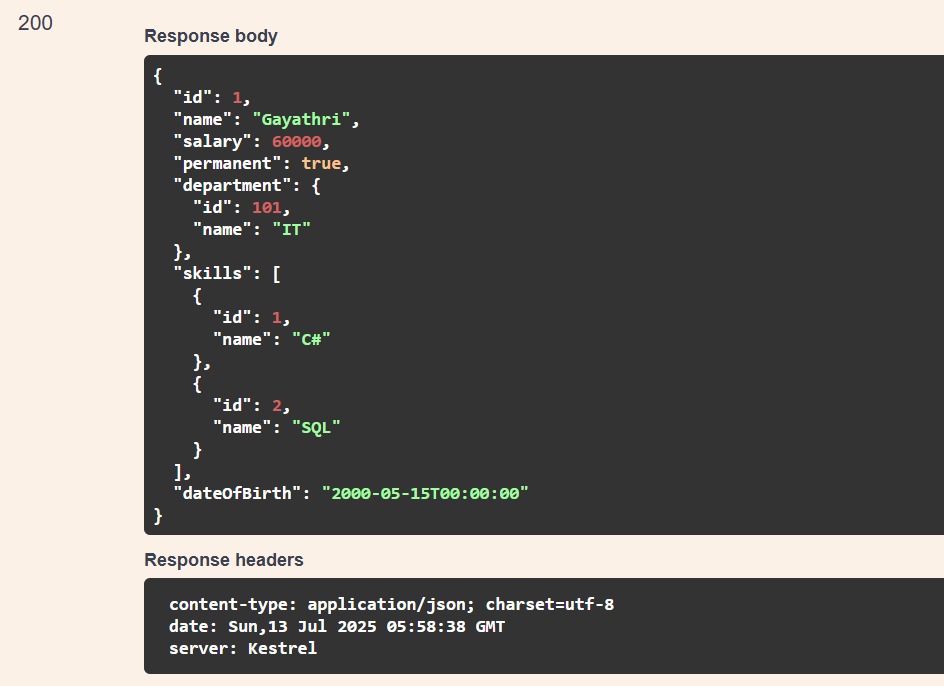
}

}









**EXERCISE 5:**

1. **JsonWebToken**
2. **Use the JWT generated thru the AuthController to be used in POSTMAN request.**
3. **Check for JWT expiration**
4. **Add the roles to be authorized in the Authorize attribute.**

**AuthController.cs**

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 EmployeeApiDemo.Controllers

{

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(101, "Admin");

//var token = GenerateJSONWebToken(101, "POC");

//var token = GenerateJSONWebToken(101, "Admin, POC");

return Ok(new { token });

}

private static string GenerateJSONWebToken(int userId, string role)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkey1234567890abcd"));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var claims = new[]

{

new Claim(ClaimTypes.Role, role),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(10),

// expires: DateTime.Now.AddMinutes(2),

signingCredentials: credentials

);

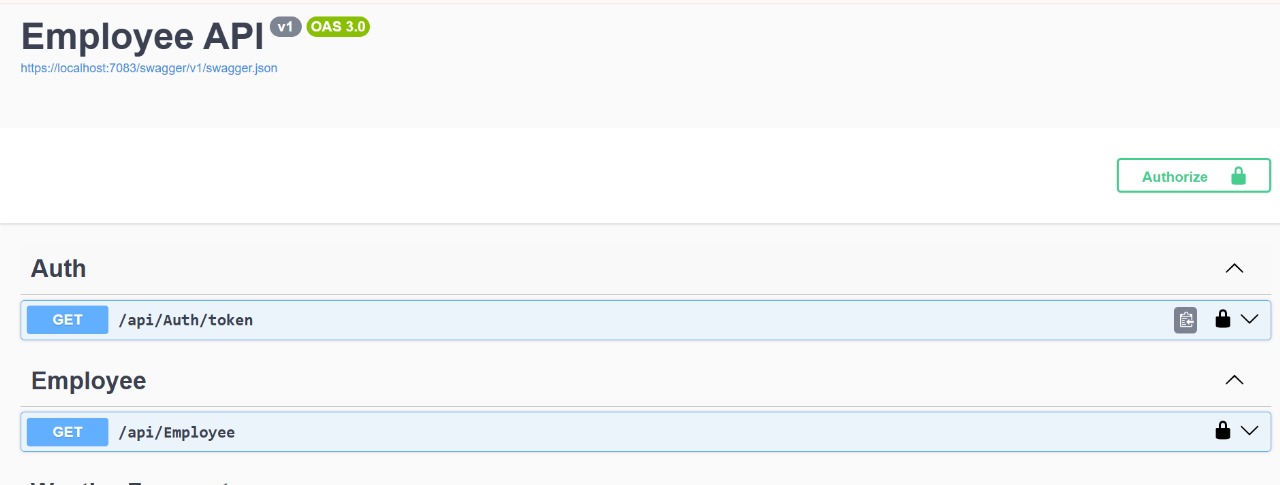
return new JwtSecurityTokenHandler().WriteToken(token);

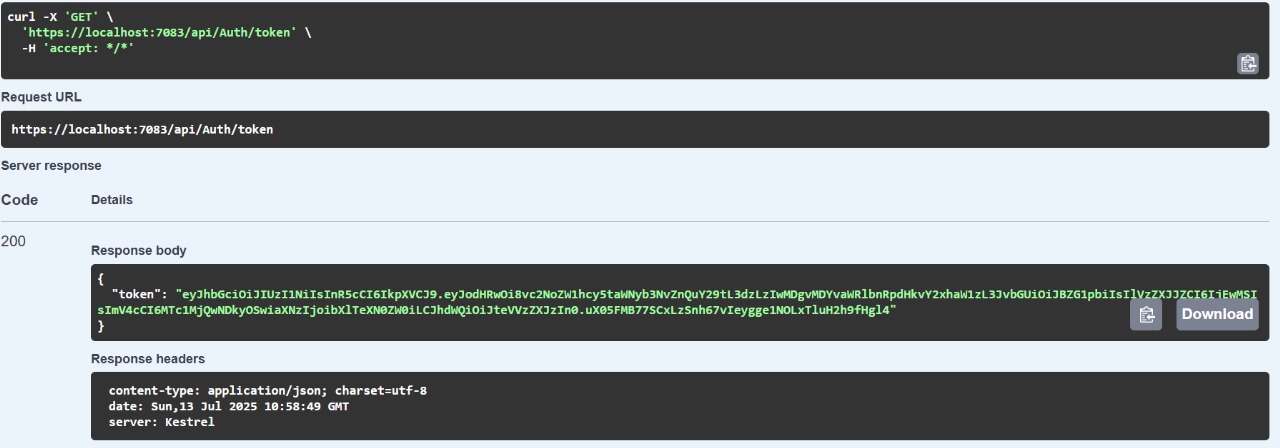
}

}

}

OUTPUT:

****

****

****

****