**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.

**a)Program.cs:**

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 = "FirstWebApi", Version = "v1" });

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthorization();

app.MapControllers();

app.Run();

**b) ValuesController.cs:**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace FirstWebApi.Controllers

{

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

[HttpGet("{id}")]

public string Get(int id)

{

return $"value {id}";

}

[HttpPost]

public void Post([FromBody] string value)

{

}

[HttpPut("{id}")]

public void Put(int id, [FromBody] string value)

{

}

[HttpDelete("{id}")]

public void Delete(int id)

{

}

}

}

**c) appsettings.json:**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**d) appsettings.Development.json:**

{

"Logging": {

"LogLevel": {

"Default": "Information",

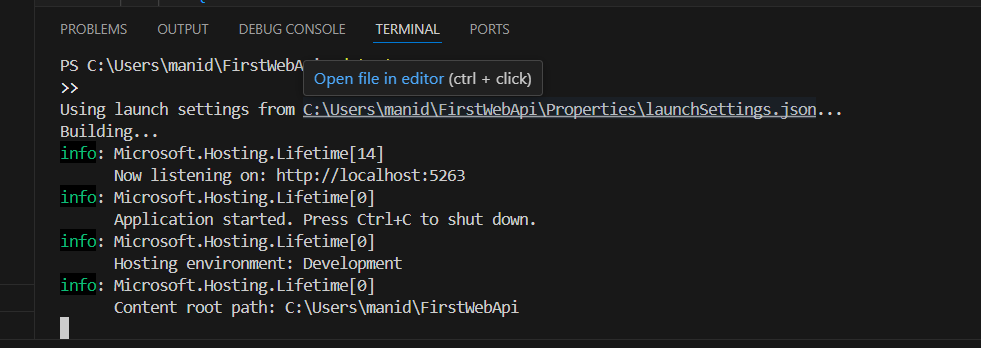
"Microsoft.AspNetCore": "Warning"

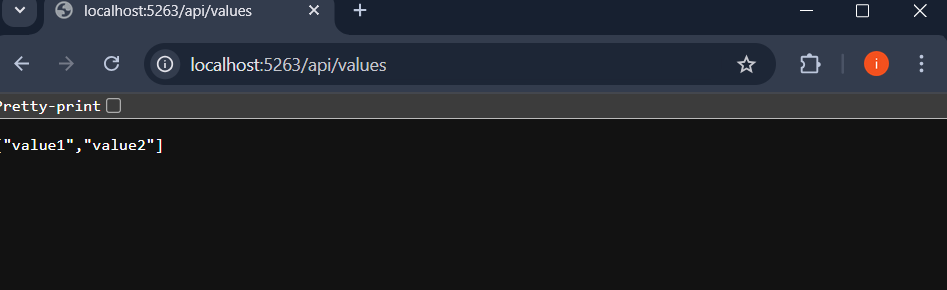
}

}

}

**Output:**





**2.Web Api using .Net core with Swagger**

**a)Program.cs:**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

Url = new Uri("https://www.example.com")

},

License = new OpenApiLicense

{

Name = "License Terms",

Url = new Uri("https://www.example.com")

}

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

});

}

app.UseAuthorization();

app.MapControllers();

app.Run();

**b) EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace FirstWebApi.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] { "John", "Jane", "David" };

}

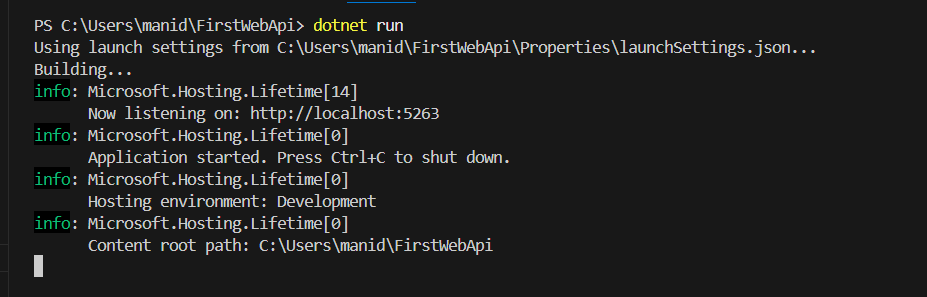
}

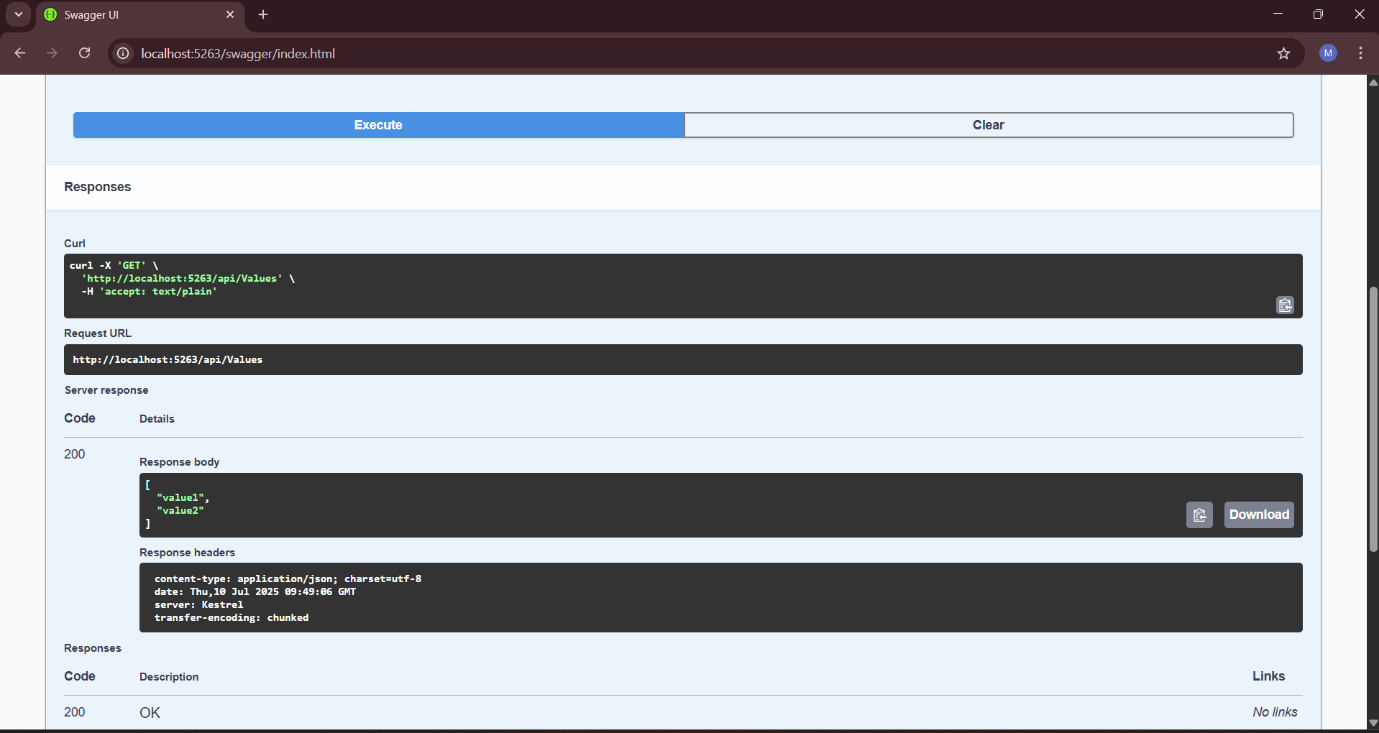
}

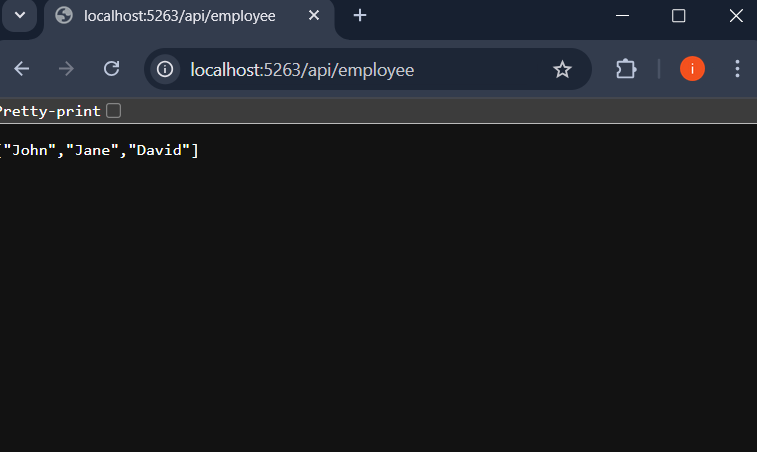
**URLs**

* **Swagger**: http://localhost:5263/swagger
* **API (Postman or Browser)**: http://localhost:5263/api/employee

Output:







**3a)Web Api using custom model class**

**b) Create a Custom action filter for Authorization**

**c) Custom Exception filter**

**Program.cs**

using FirstWebApi.Filters;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Custom Model, Authorization, Exception Filter"

});

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Description = "Enter 'Bearer' followed by your token.\nExample: Bearer abc123",

Name = "Authorization",

In = ParameterLocation.Header,

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Models/Department.cs**

namespace FirstWebApi.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Models/Skill.cs**

namespace FirstWebApi.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Models/Employee.cs:**

using System;

using System.Collections.Generic;

namespace FirstWebApi.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; }

}

}

**Controllers/EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

using FirstWebApi.Models;

using FirstWebApi.Filters;

using Microsoft.AspNetCore.Http;

using System.Collections.Generic;

using System.Linq;

using System;

namespace FirstWebApi.Controllers

{

[ApiController]

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

[CustomAuthFilter]

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees;

static EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

[HttpGet("standard")]

public ActionResult<Employee> GetStandard()

{

return Ok(\_employees.FirstOrDefault());

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

\_employees.Add(emp);

return CreatedAtAction(nameof(Get), new { id = emp.Id }, emp);

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] Employee emp)

{

var existing = \_employees.FirstOrDefault(e => e.Id == id);

if (existing == null) return NotFound();

existing.Name = emp.Name;

existing.Salary = emp.Salary;

existing.Permanent = emp.Permanent;

existing.Department = emp.Department;

existing.Skills = emp.Skills;

existing.DateOfBirth = emp.DateOfBirth;

return NoContent();

}

private static List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 60000,

Permanent = true,

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1990, 5, 20)

}

};

}

}

}

**Filters/CustomAuthFilter.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace FirstWebApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

if (!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token))

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

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

{

context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

}

}

}

}

**Filters/CustomExceptionFilter.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

using System.IO;

namespace FirstWebApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

File.AppendAllText("logs.txt", $"{DateTime.Now} - {exception.Message}\n");

context.Result = new ObjectResult("Internal Server Error")

{

StatusCode = 500

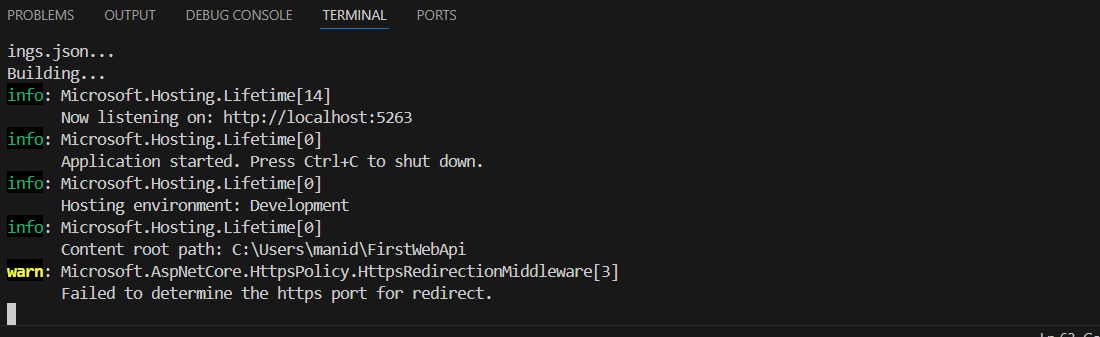
};

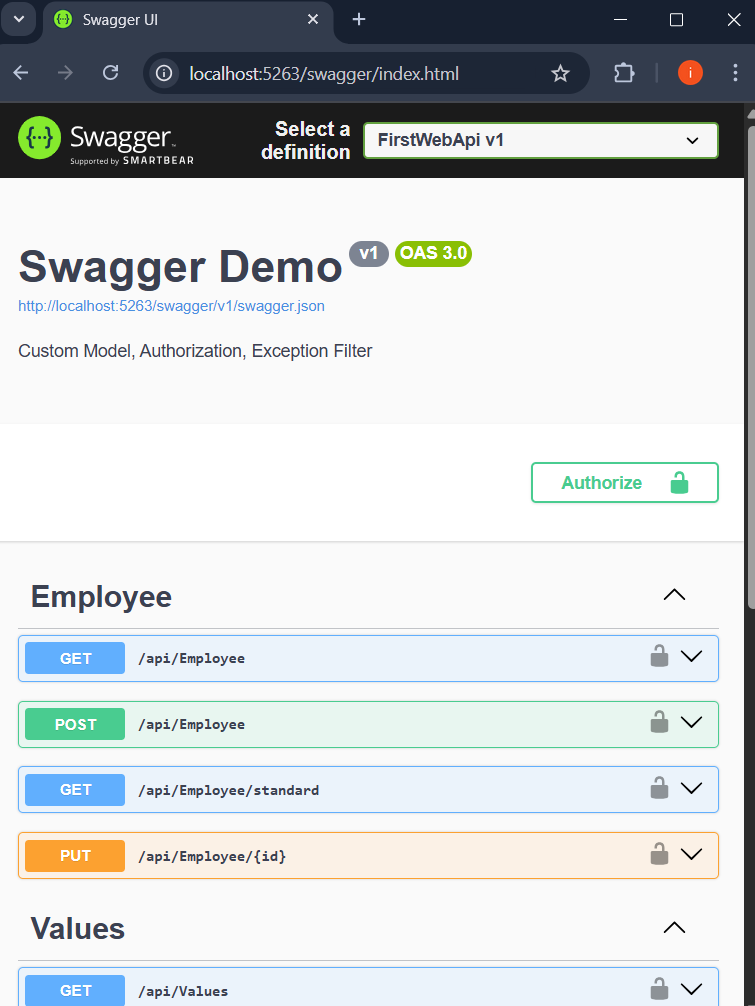
}

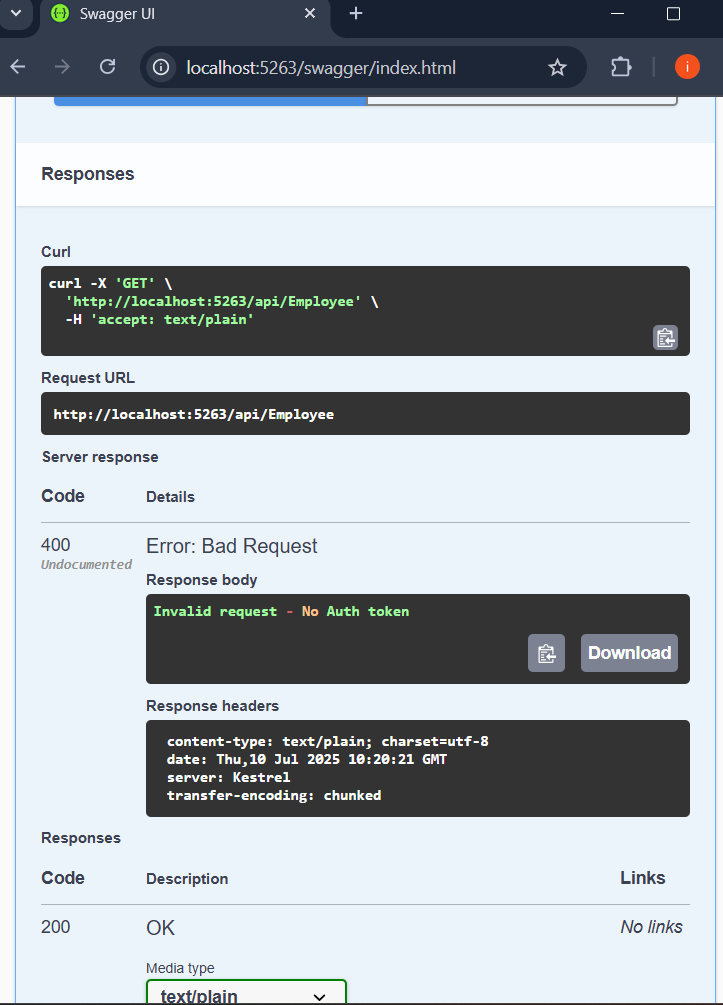
}

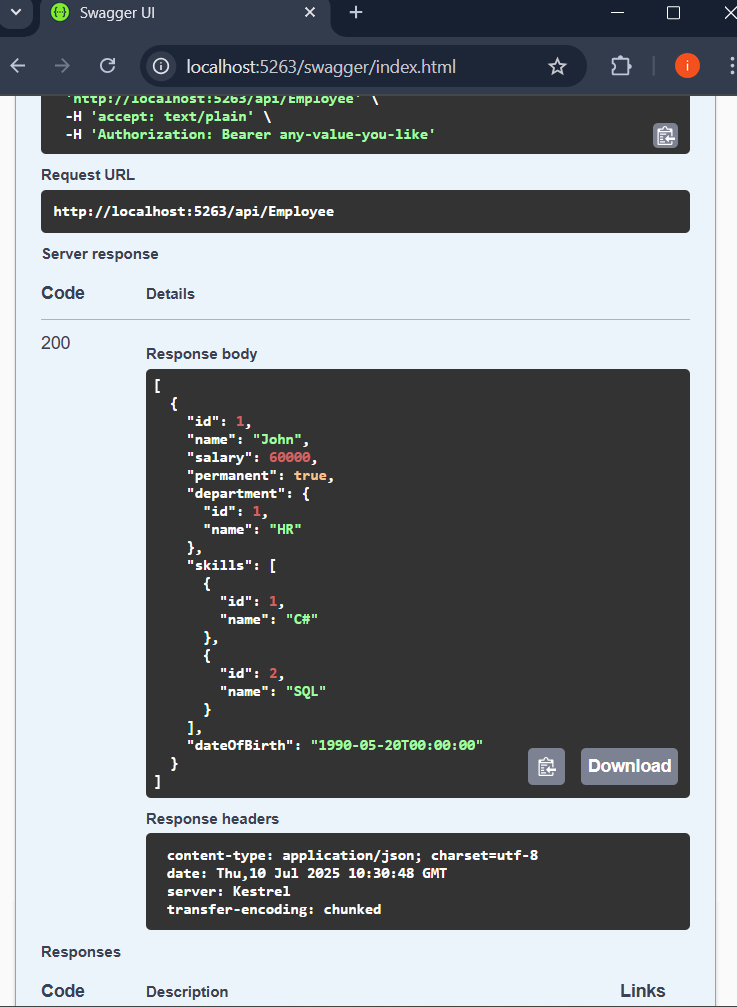
}

**Output:**



****

****

****

**4.Web Api CRUD operation**

Update Employee data as per the input thru Web API PUT action method call

Employee information has to be updated based on the user input. Use Swagger tool to invoke the action method mapped with Http PUT action verb to update an employee data.

Modify the action method to return Employee data thru ActionResult.

Check if the id value is lesser than or equal to 0. If true, throw BadRequest action result with the message ‘Invalid employee id’

If the value is greater than 0 but not available in the list of employee ids that is there in the hardcoded list of employees, throw BadRequest action result with the same message as stated above.

If the id value is valid, use the JSON data from the input body and update the hardcoded list. Filter the employee list data for the input id and return that as the output.

**Models/Employee.cs:**

namespace FirstWebApi.Models

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public string Department { get; set; }

public double Salary { get; set; }

}

}

;

**Controllers/EmployeesController.cs:**

using Microsoft.AspNetCore.Mvc;

using FirstWebApi.Models;

using System.Collections.Generic;

using System.Linq;

namespace FirstWebApi.Controllers

{

[ApiController]

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

public class EmployeesController : ControllerBase

{

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

{

new Employee { Id = 1, Name = "Alice", Department = "HR", Salary = 50000 },

new Employee { Id = 2, Name = "Bob", Department = "IT", Salary = 60000 },

new Employee { Id = 3, Name = "Charlie", Department = "Finance", Salary = 55000 }

};

[HttpPost]

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

{

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

employees.Add(newEmployee);

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

}

[HttpGet("{id}")]

public ActionResult<Employee> GetEmployee(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 updatedEmployee)

{

if (id <= 0) return BadRequest("Invalid employee id");

var existing = employees.FirstOrDefault(e => e.Id == id);

if (existing == null) return BadRequest("Invalid employee id");

existing.Name = updatedEmployee.Name;

existing.Department = updatedEmployee.Department;

existing.Salary = updatedEmployee.Salary;

return Ok(existing);

}

[HttpDelete("{id}")]

public ActionResult DeleteEmployee(int id)

{

var emp = employees.FirstOrDefault(e => e.Id == id);

if (emp == null) return NotFound("Employee not found");

employees.Remove(emp);

return Ok("Employee deleted successfully");

}

}

}

**Program.cs:**

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 = "Employee API",

Version = "v1",

Description = "Simple Web API for CRUD operations on employees"

});

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseAuthorization();

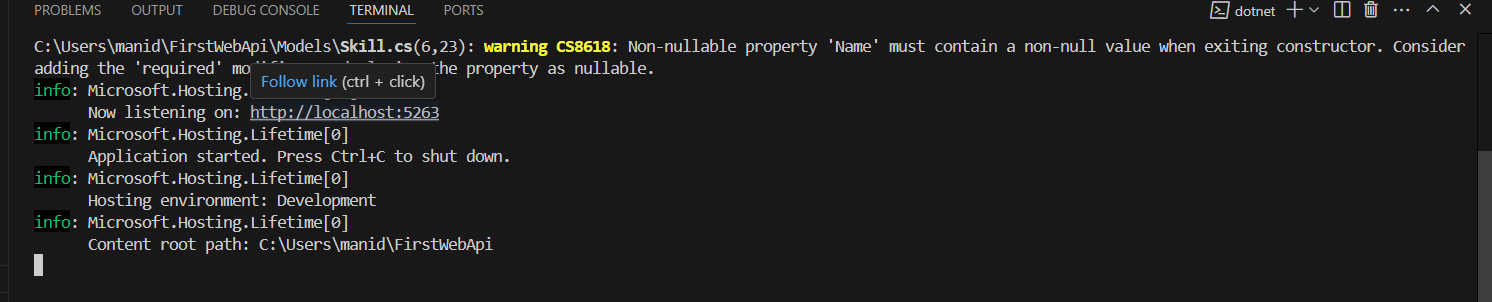
app.MapControllers();

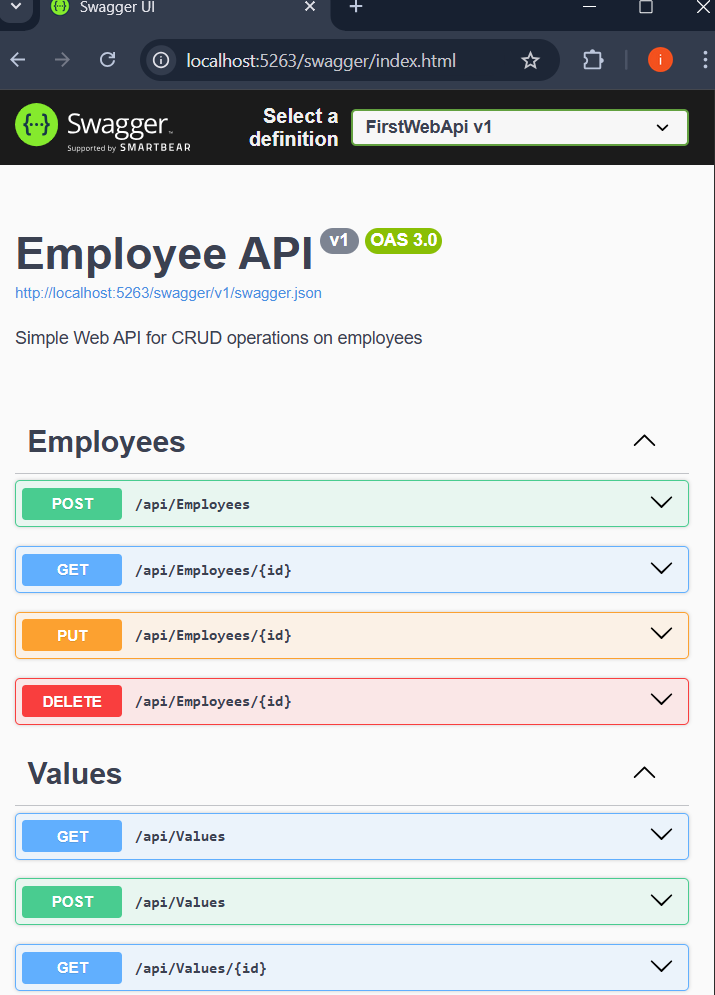
app.Run();

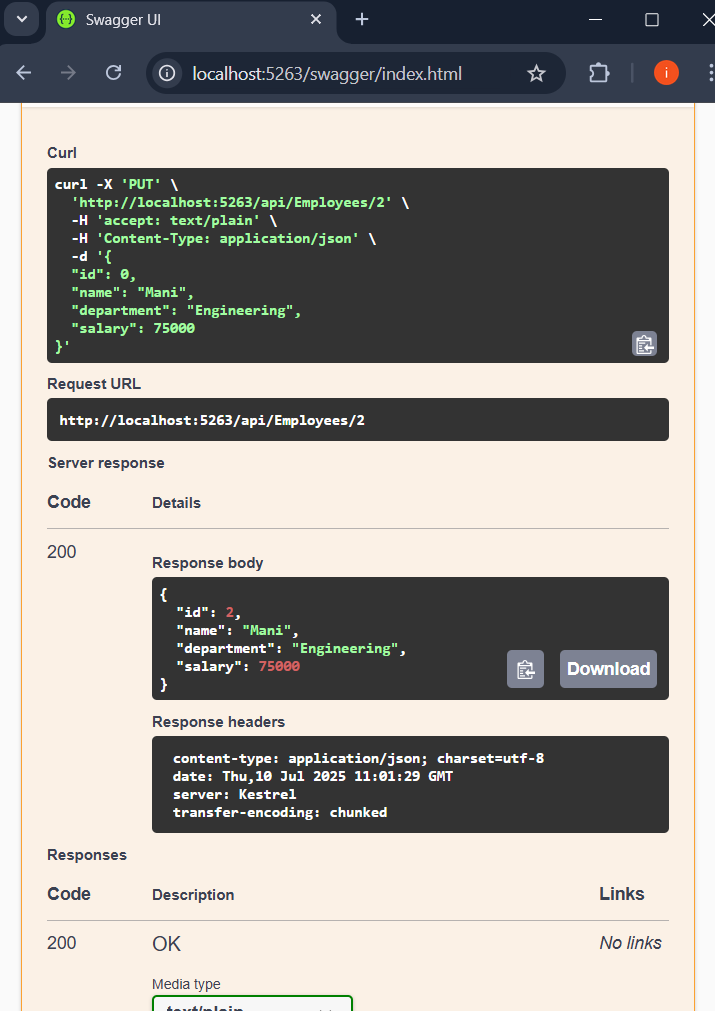
**URLs:**

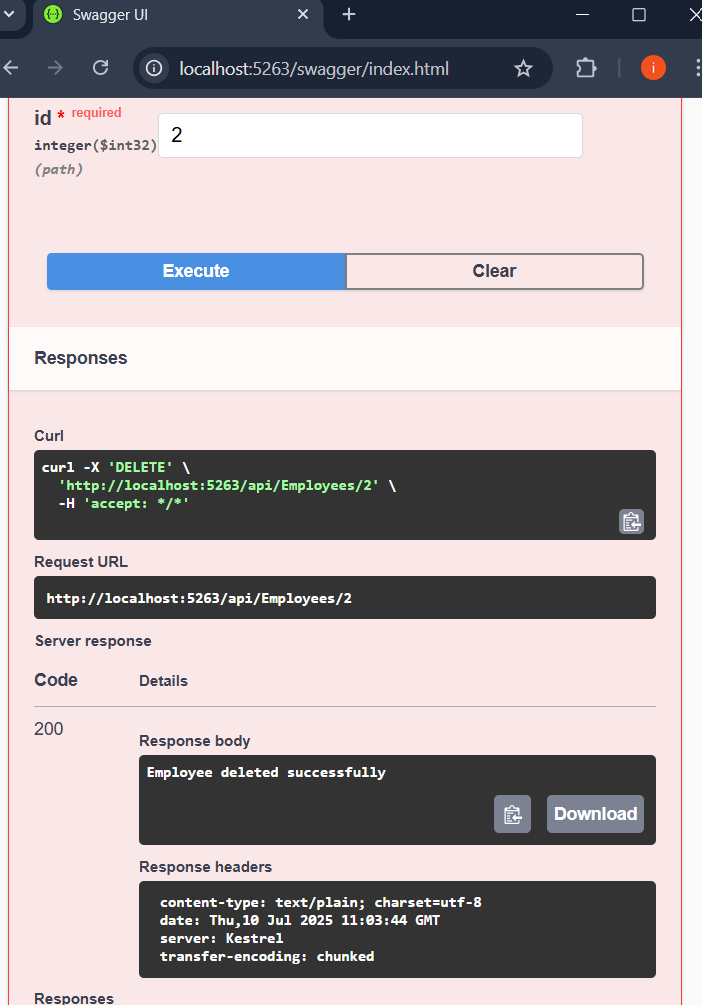
* **Swagger UI**: http://localhost:5000/swagger
* **POST** http://localhost:5000/api/employees
* **GET** http://localhost:5000/api/employees/{id}
* **PUT** http://localhost:5000/api/employees/{id}
* **DELETE** http://localhost:5000/api/employees/{id}

**Output:**

****

****

****

****

**4a)JsonWebToken**

**b) Use the JWT generated thru the AuthController to be used in POSTMAN request**

**c) Check for JWT expiration**

**d) 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;

[AllowAnonymous]

[ApiController]

[Route("[controller]")]

public class AuthController : ControllerBase

{

[HttpGet("token")]

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("mysuperdupersecretkey123456789012"));

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(10),

signingCredentials: credentials

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

**EmployeeController.cs:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

[Authorize(Roles = "Admin,POC")]

[ApiController]

[Route("[controller]")]

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

return Ok("Employee data accessed.");

}

}

**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("AllowAllOrigins", policy =>

{

policy.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod();

});

});

string securityKey = "mysuperdupersecretkey123456789012";

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

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

builder.Services.AddControllers();

var app = builder.Build();

app.UseRouting();

app.UseCors("AllowAllOrigins");

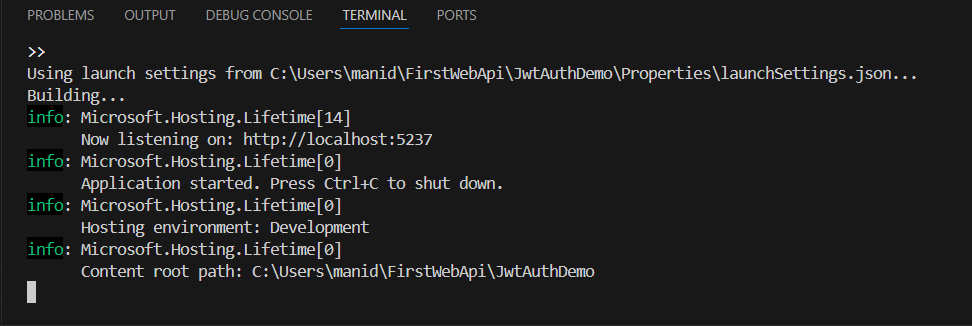
app.UseAuthentication();

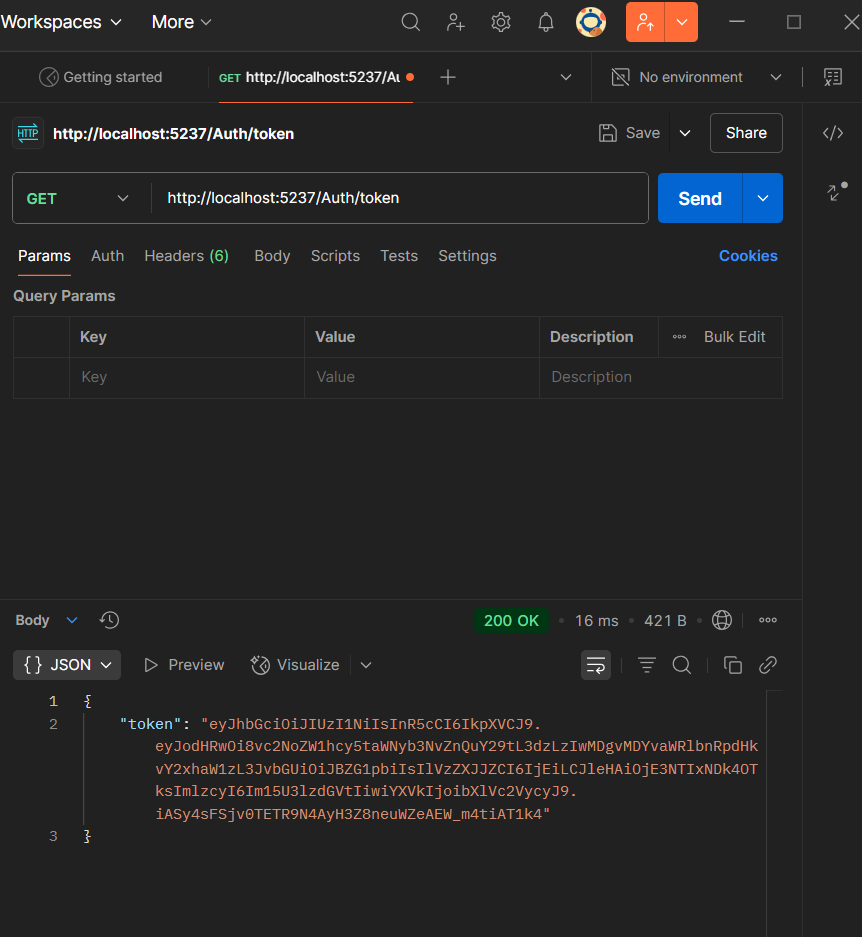
app.UseAuthorization();

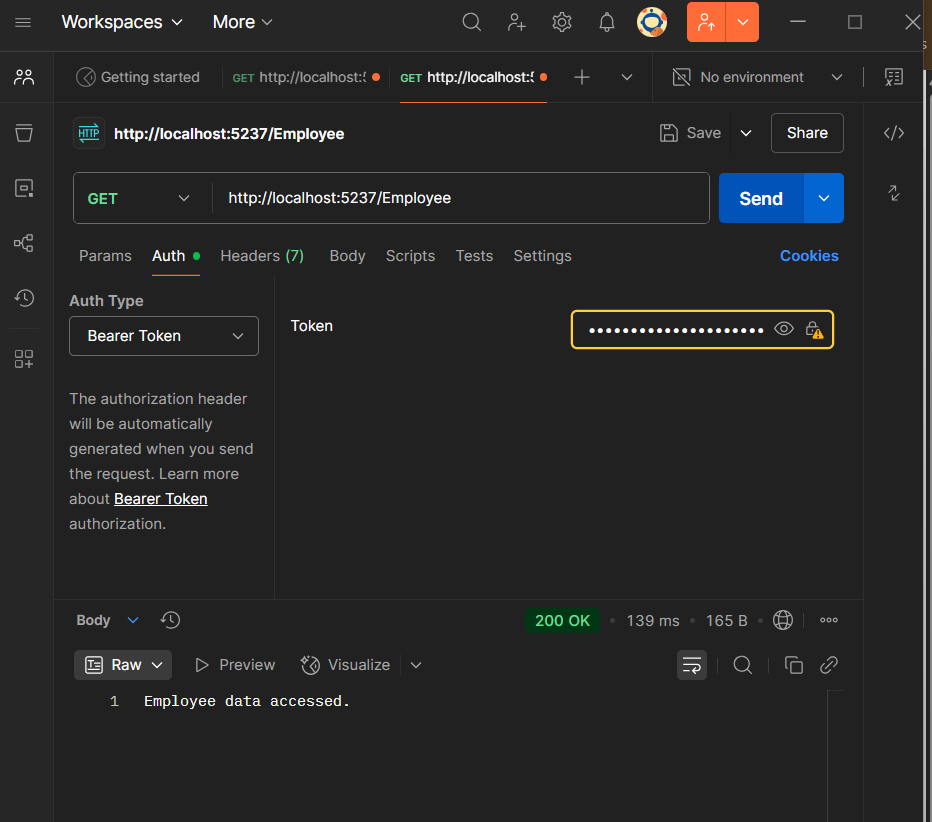
app.MapControllers();

app.Run();

**Output:**







Week-4:

Gopesh Sharma