ASP.NET Core 8.0 Web API

1.WebApi\_Handson

RESTful Web Service, Web API & Microservice

REST (Representational State Transfer):

Stateless communication over HTTP

Resources identified by URIs

Uses standard HTTP methods (GET, POST, PUT, DELETE)

Responses not limited to XML; JSON preferred

Web API:

Lightweight framework to build RESTful HTTP services

Returns data (usually JSON) to clients like browsers or mobile apps

Microservice:

Independently deployable, small services

Communicate via APIs (often REST)

Each service handles a specific business function

WebService vs WebAPI

Feature WebService (SOAP) WebAPI (REST)

Protocol SOAP HTTP/HTTPS

Data Format XML only JSON, XML, etc.

Lightweight No Yes

Use Case Enterprise apps Web/mobile apps

HttpRequest & HttpResponse

HttpRequest: Sent by the client to request data or perform actions.

HttpResponse: Sent by the server to return data, status, or error.

Example:

POST /api/items

Request Body: { "name": "Notebook" }

Response: 201 Created

Response Body: { "id": 1, "name": "Notebook" }

Action Verbs (HTTP Methods)

Verb Description Attribute in Web API

GET Fetch data [HttpGet]

POST Add new data [HttpPost]

PUT Update data [HttpPut]

DELETE Delete data [HttpDelete]

HTTP Status Codes

Code Meaning Method Usage

200 OK return Ok(data);

201 Created return CreatedAtAction(...)

400 Bad Request return BadRequest();

401 Unauthorized return Unauthorized();

500 Internal Server Error return StatusCode(500);

Simple Web API: Read & Write

Structure

Controller: Inherits from ApiController

Actions: Defined using action verbs

Controllers/ItemsController.cs

[ApiController]

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

public class ItemsController : ControllerBase

{

public ItemsController()

{

Console.WriteLine("\nItemsController loaded\n");

}

private static List<Item> items = new()

{

new Item { Id = 1, Name = "Pen" },

new Item { Id = 2, Name = "Book" }

};

[HttpGet]

public IActionResult Get() => Ok(items);

[HttpPost]

public IActionResult Post(Item item)

{

item.Id = items.Count + 1;

items.Add(item);

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

}

}

Test API with Postman

GET

Method: GET

URL: http://localhost:<post>/api/items

POST

Method: POST

URL: http://localhost:<post>/api/items

Body: raw -> JSON

{

"name": "Notebook"

}

Configuration Files in Web API

File Description

Startup.cs or Program.cs Configures services, middleware (dependency injection)

appsettings.json App-level config: connection strings, secrets

launchSettings.json Debug profiles for Kestrel, IIS Express

Web.config / RouteConfig.cs (ASP.NET 4.5) For older .NET Framework apps, define routing and settings

WebApiConfig.cs (.NET 4.5) Register HTTP routes for APIs

CLI Steps to Create API

dotnet new webapi -n <Api\_Name>

cd <Api\_Name>

Add Model

Models/Item.cs

public class Item

{

public int Id { get; set; }

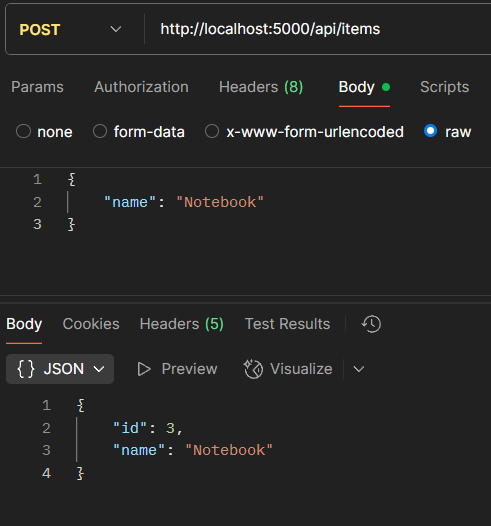
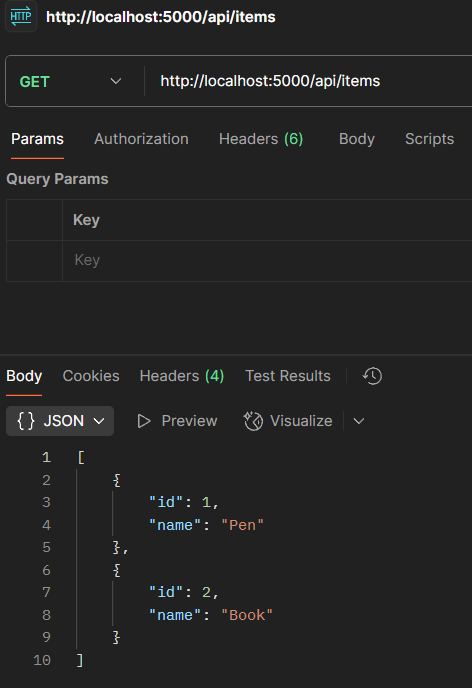
public string Name { get; set; }

}

Add Controller

Show Above in README section Structure sub-sectionControllers/ItemsController.cs

OUTPUT:



2. WebApi\_Handson

Create Project

dotnet new webapi -n exercise2

cd exercise2

Install Swagger Package

dotnet add package Swashbuckle.AspNetCore

Startup.cs Configuration

ConfigureServices Method

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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

Contact = new Microsoft.OpenApi.Models.OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new Microsoft.OpenApi.Models.OpenApiLicense

{

Name = "License Terms",

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

}

});

});

Configure Method

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

EmployeeController

Controllers/EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

namespace exercise2.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static readonly List<string> employees = new()

{

"Alice", "Bob", "Charlie"

};

[HttpGet]

[ProducesResponseType(200)]

public IActionResult Get()

{

return Ok(employees);

}

[HttpPost]

[ProducesResponseType(201)]

public IActionResult Post([FromBody] string name)

{

employees.Add(name);

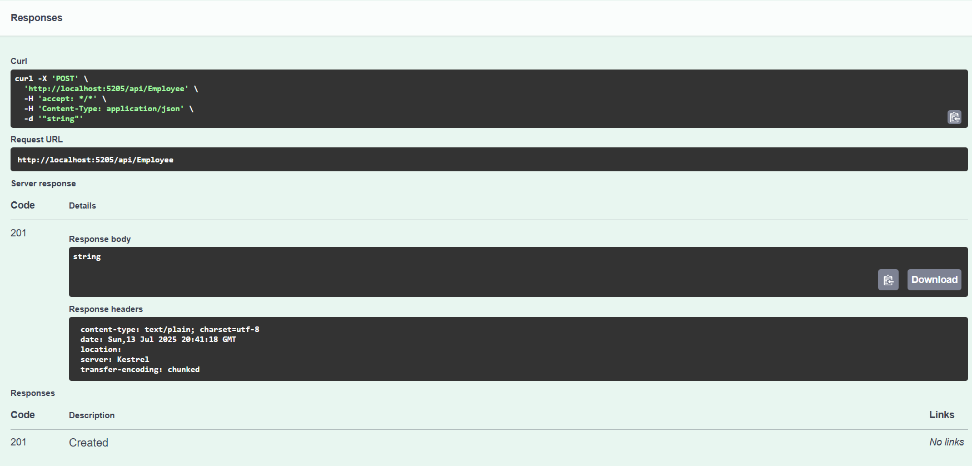
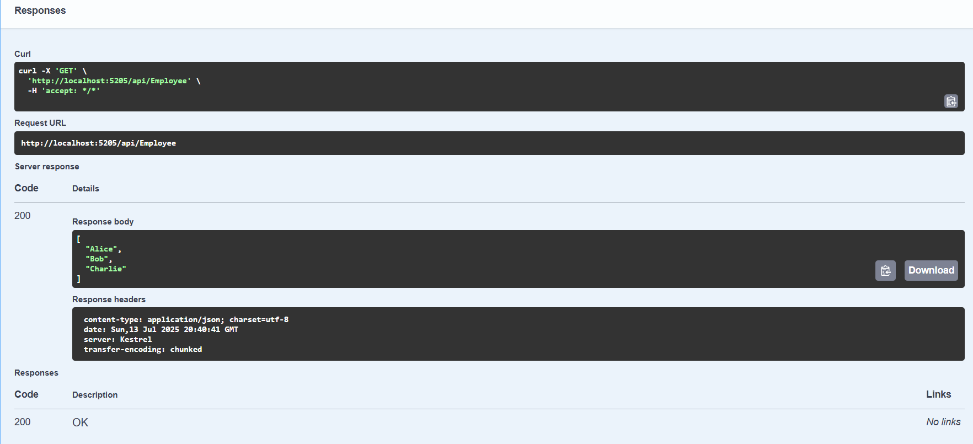
return Created("", name);

}

}

}

OUTPUT:



3. WebApi\_Handson

Install Dependencies

dotnet add package Microsoft.AspNetCore.Mvc.NewtonsoftJson

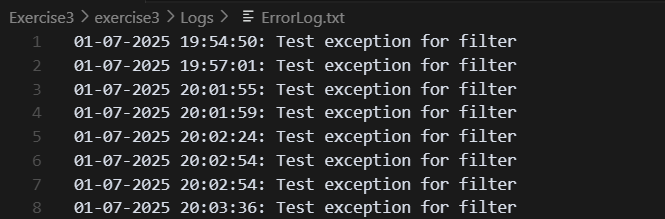
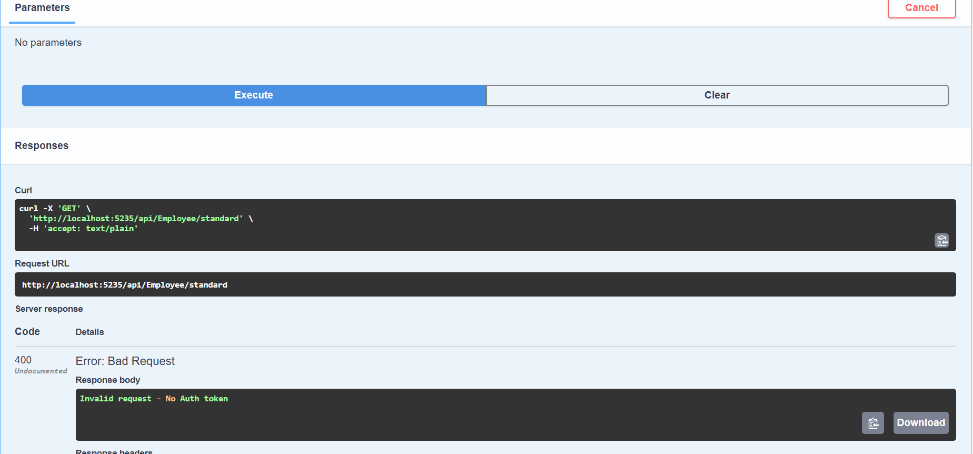
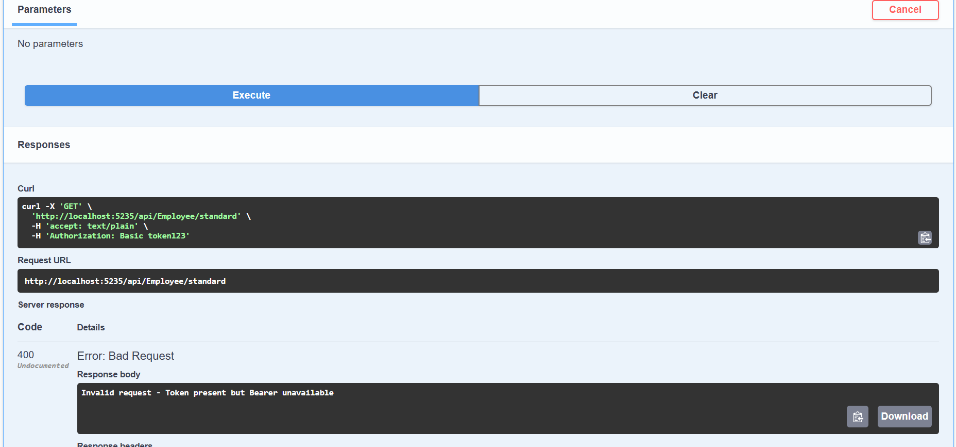
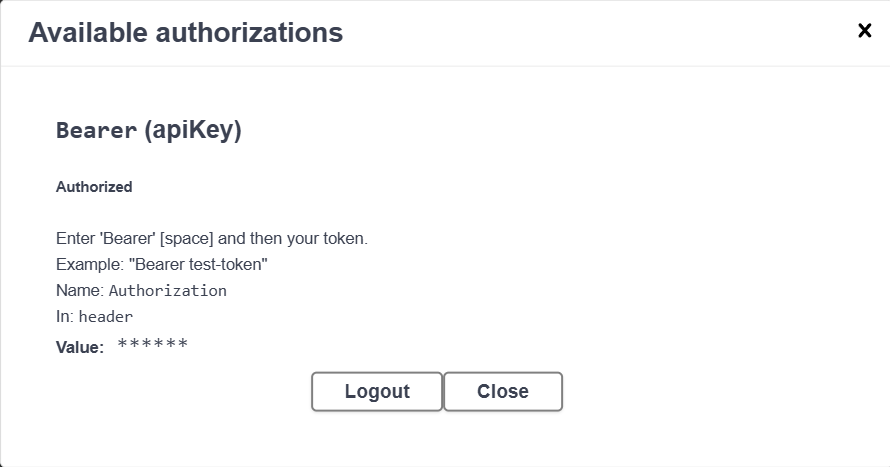
dotnet add package Swashbuckle.AspNetCore

dotnet add package Microsoft.AspNetCore.Mvc.WebApiCompatShim

Running the Project

dotnet build

dotnet run

OUTPUT:  


4. WebApi\_Handson

Implementation

DTO: EmployeeDto

public class EmployeeDto

{

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public int Salary { get; set; }

}

Service: InMemoryEmployeeService

A service layer that holds hardcoded List<EmployeeDto> and supports:

GetAll()

GetById(int id)

Update(EmployeeDto emp)

Endpoints in EmployeeController

GET /api/employee Returns a list of all employees.

PUT /api/employee Updates an existing employee using [FromBody].

{

"id": 2,

"name": "Joseph",

"salary": 75000

}

Validation:

ID <= 0 : 400 BadRequest: Invalid employee id

ID not found in hardcoded list : 400 BadRequest: Invalid employee id

Otherwise : 200 OK

Testing

Swagger

Run the app: dotnet run

Navigate to: http://localhost:<port>/swagger

Use the PUT endpoint to test update

Postman

Set method to PUT

URL: http://localhost:<port>/api/employee

Set Headers: Content-Type: application/json

JSON Body:

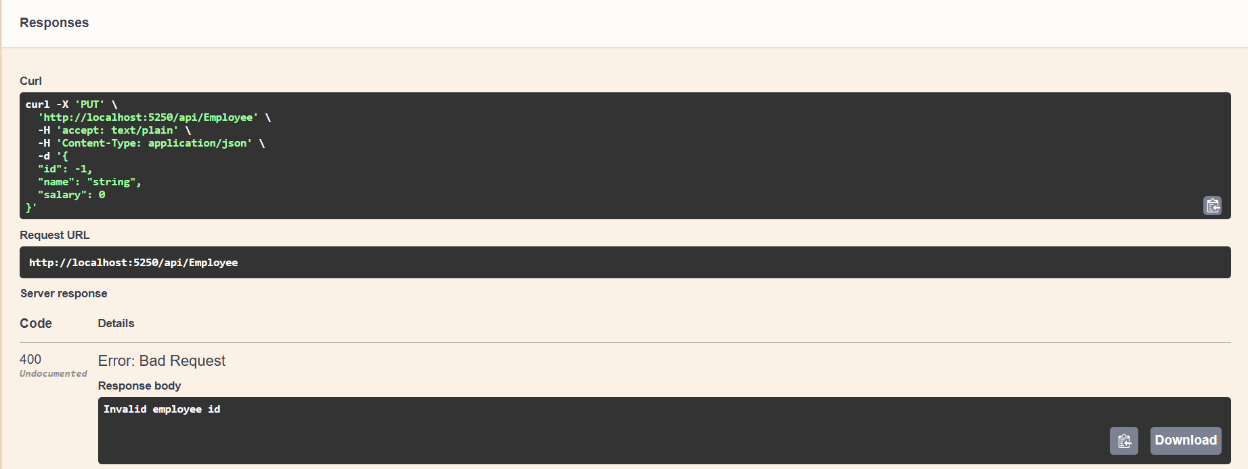
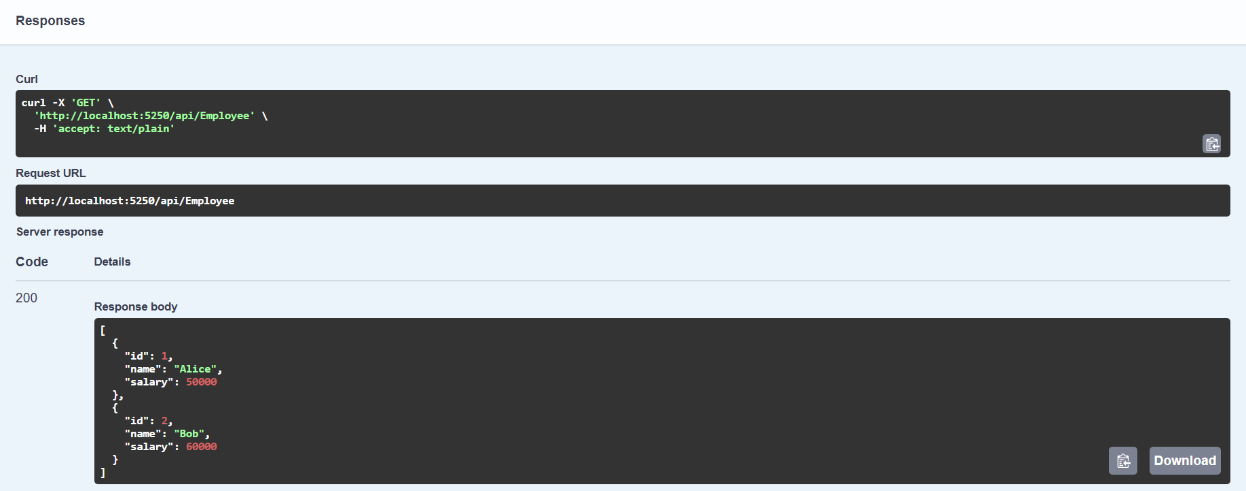
{

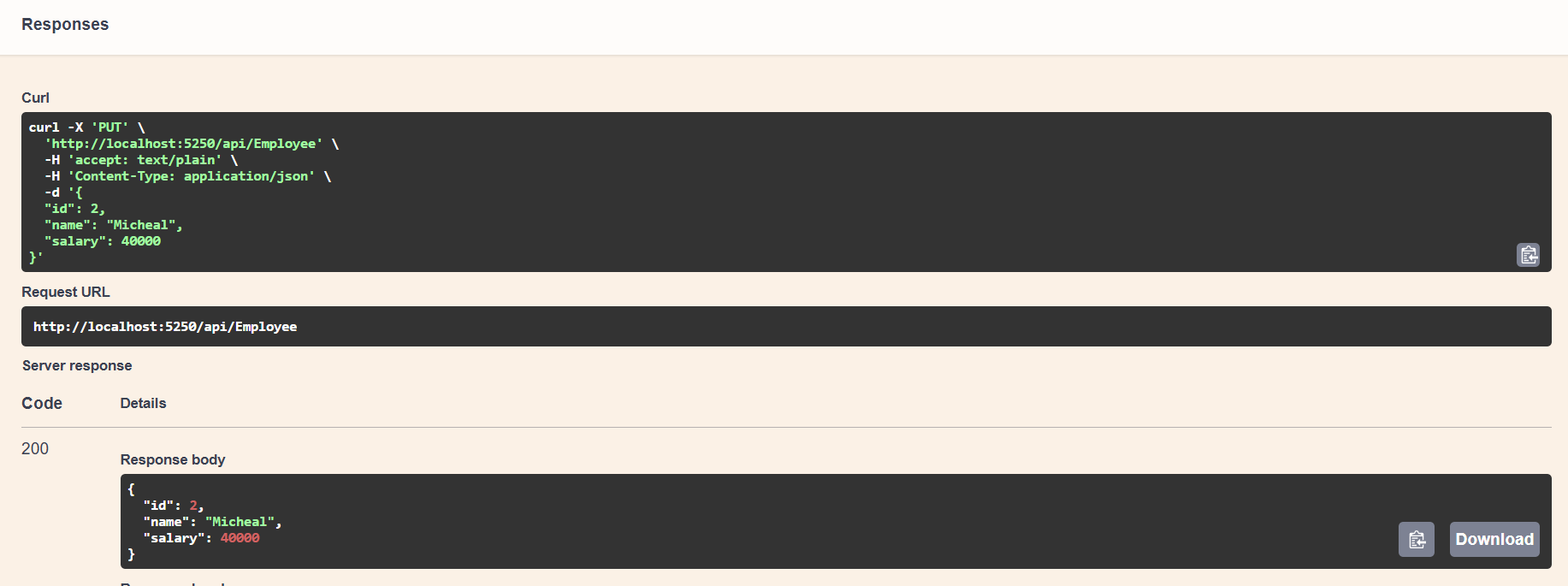
"id": 2,

"name": "Joseph",

"salary": 75000

}

OUTPUT:



5. WebApi\_Handson

Enable CORS

CORS (Cross-Origin Resource Sharing) allows APIs to be called from other domains Startup.cs

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAll", policy =>

{

policy.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader();

});

});

app.UseCors("AllowAll");

CORS enables your frontend apps (like React or Angular) to access the Web API.

JWT Authentication

Install required packages:

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

In startup.cs:

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("SECRET\_SECURITY\_KEY"));

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 = securityKey

};

});

In the middleware pipeline:

app.UseAuthentication();

app.UseAuthorization();

JWT Token Generation via AuthController

[AllowAnonymous]

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

[ApiController]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

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

return Ok(token);

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("SECRET\_SECURITY\_KEY"));

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);

}

}

Secure Endpoints with [Authorize]

[ApiController]

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

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

public class EmployeeController : ControllerBase

{

[HttpGet("data")]

public IActionResult GetEmployeeData()

{

return Ok("Secure Employee Data Returned.");

}

}

If token is missing or expired → Returns 401 Unauthorized

Testing in Postman

GET api/auth/token : Copy the JWT

Set Header in Postman:

Key: Authorization

Value: Bearer <token>

Hit api/employee : You should receive 200 OK if the token is valid.

Modify/Remove Token : You'll get 401 Unauthorized.

OUTPUT:  
