**1. JsonWebToken**

**1.Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

using YourNamespace.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddScoped<CustomAuthFilter>();

string securityKey = "mysuperdupersecretmysuperdupersecret";

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(options =>

{

options.Filters.Add<CustomExceptionFilter>(); // your global exception filter

});

builder.Services.AddEndpointsApiExplorer();

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")

}

});

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Description = "Enter 'Bearer' [space] and then 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"

}

},

Array.Empty<string>()

}

});

});

var app = builder.Build();

// Enable Swagger UI

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

2. **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 YourNamespace.Controllers

{

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin"); // You can change the role here

return Ok(token);

}

private string GenerateJSONWebToken(int userId, string userRole)

{

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

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

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, "POC"),

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);

}

}

}

3. **EmployeeController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using YourNamespace.Filters;

using YourNamespace.Models;

namespace YourNamespace.Controllers

{

[ApiController]

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

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

public class EmployeeController : ControllerBase

{

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

{

new Employee

{

Id = 1,

Name = "Sumiran",

Salary = 50000,

Permanent = true,

Department = new Department { DeptId = 1, DeptName = "HR" },

Skills = new List<Skill>

{

new Skill { SkillId = 1, SkillName = "C#" },

new Skill { SkillId = 2, SkillName = "SQL" }

},

DateOfBirth = new DateTime(1998, 8, 15)

},

new Employee

{

Id = 2,

Name = "Anita",

Salary = 60000,

Permanent = true,

Department = new Department { DeptId = 2, DeptName = "Finance" },

Skills = new List<Skill>

{

new Skill { SkillId = 3, SkillName = "Excel" }

},

DateOfBirth = new DateTime(1999, 5, 10)

}

};

// GET: api/Employee

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employees);

}

// POST: api/Employee

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

public ActionResult<Employee> Post([FromBody] Employee employee)

{

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

employees.Add(employee);

return CreatedAtAction(nameof(GetStandard), new { id = employee.Id }, employee);

}

// PUT: api/Employee

[HttpPut]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

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

{

if (updatedEmployee.Id <= 0)

{

return BadRequest("Invalid employee id");

}

var existingEmployee = employees.FirstOrDefault(e => e.Id == updatedEmployee.Id);

if (existingEmployee == null)

{

return BadRequest("Invalid employee id");

}

existingEmployee.Name = updatedEmployee.Name;

existingEmployee.Salary = updatedEmployee.Salary;

existingEmployee.Permanent = updatedEmployee.Permanent;

existingEmployee.Department = updatedEmployee.Department;

existingEmployee.Skills = updatedEmployee.Skills;

existingEmployee.DateOfBirth = updatedEmployee.DateOfBirth;

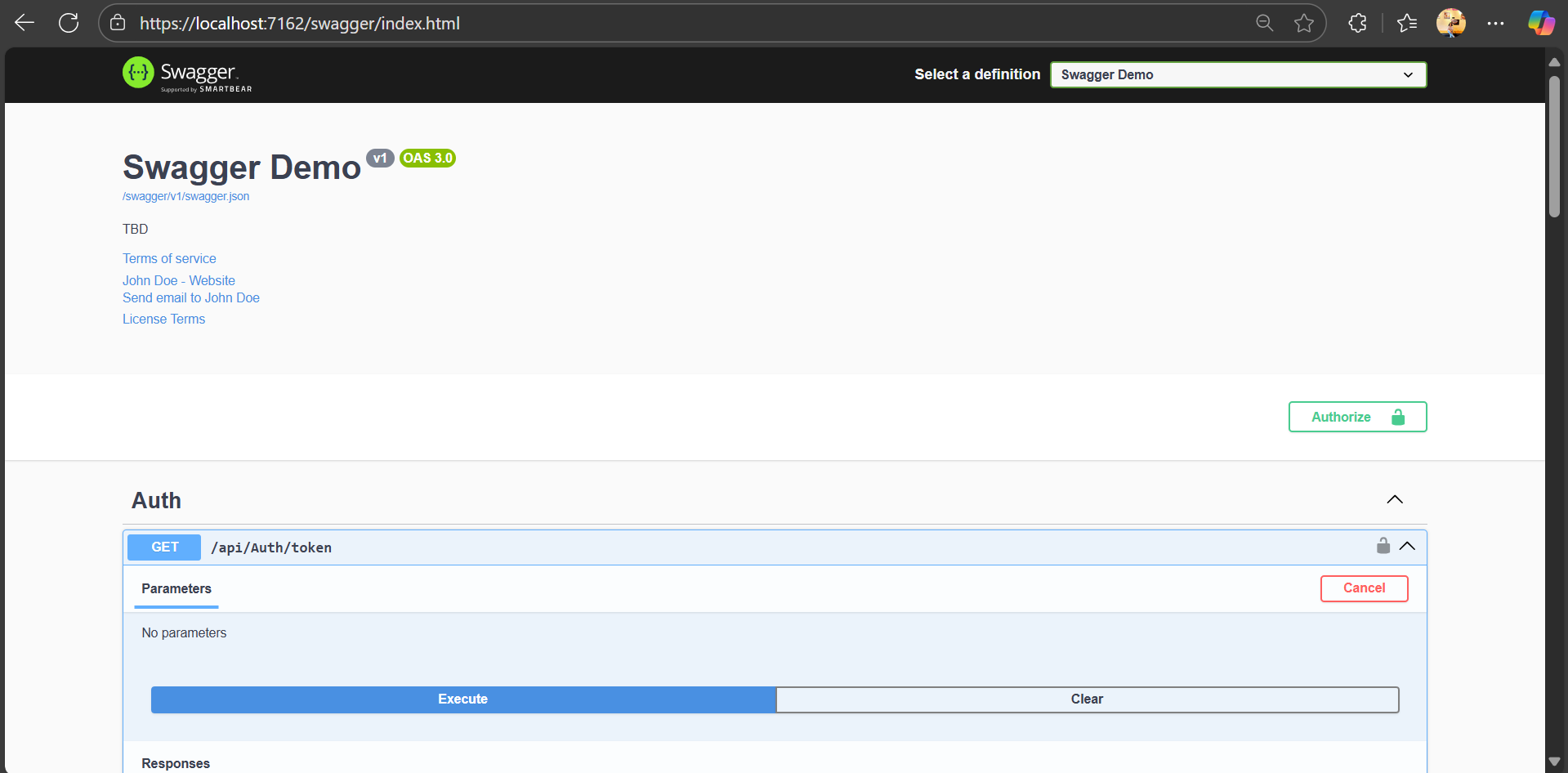
return Ok(existingEmployee);

}

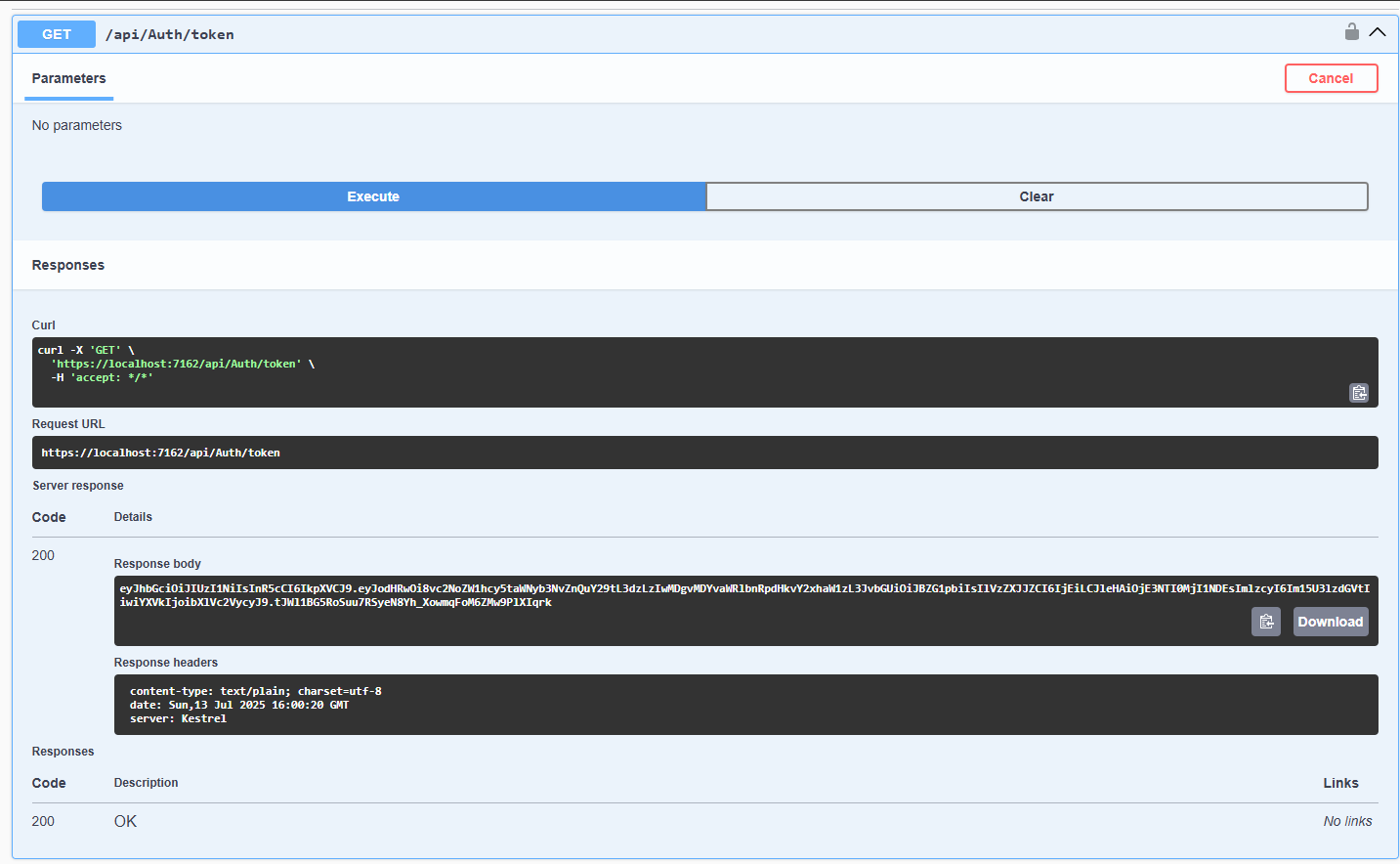
}

}

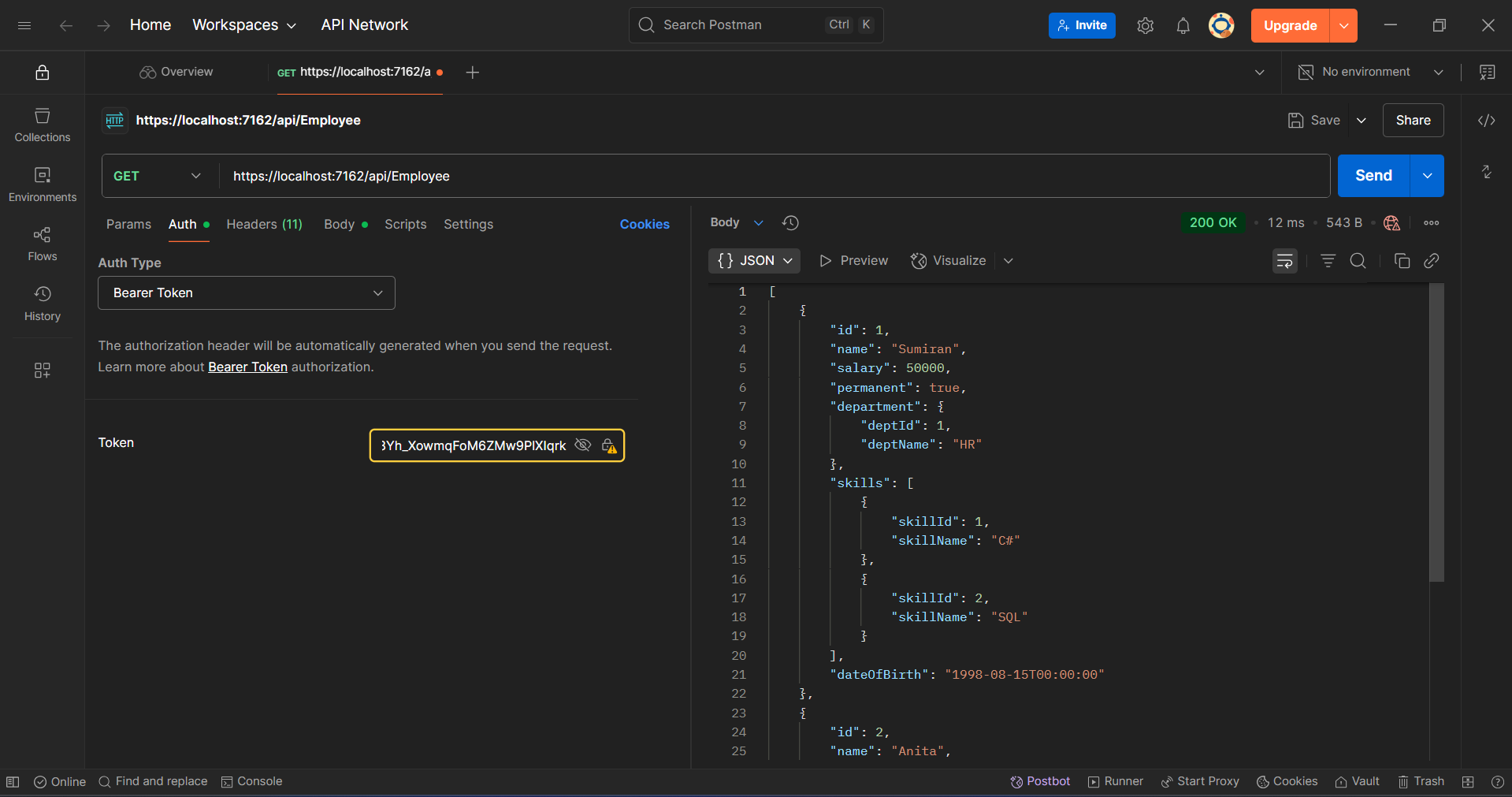
1. **Swagger UI**



1. **Swagger - Token Generation**



1. **Postman - GET /api/Employee with valid token**



**4.Postman - GET after token expiry (wait >2 mins)**

