**Week 5: Microservices Architecture using ASP.NET Core Web API**

Submitted by- Sweta Singh

Superset id: 6363535

**Hands-On Exercises: Authentication and Authorization in ASP.NET Core Web API Microservices**

**Question 1: Implement JWT Authentication in ASP.NET Core Web API**

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Enter JWT token with Bearer prefix like: Bearer eyJhbGciOi...",

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

options.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] { }

}

});

});

builder.Services.AddAuthentication("Bearer")

.AddJwtBearer("Bearer", options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = builder.Configuration["Jwt:Issuer"],

ValidAudience = builder.Configuration["Jwt:Audience"],

IssuerSigningKey = new SymmetricSecurityKey(

Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))

};

options.Events = new JwtBearerEvents

{

OnAuthenticationFailed = context =>

{

if (context.Exception is SecurityTokenExpiredException)

{

context.Response.Headers.Add("Token-Expired", "true");

context.Response.StatusCode = StatusCodes.Status401Unauthorized;

context.Response.ContentType = "application/json";

var result = System.Text.Json.JsonSerializer.Serialize(new

{

message = "Token has expired. Please login again."

});

return context.Response.WriteAsync(result);

}

return Task.CompletedTask;

},

OnChallenge = context =>

{

if (!context.Response.HasStarted)

{

context.Response.StatusCode = StatusCodes.Status401Unauthorized;

context.Response.ContentType = "application/json";

var result = System.Text.Json.JsonSerializer.Serialize(new

{

message = "You are not authorized. Token is missing or invalid."

});

return context.Response.WriteAsync(result);

}

return Task.CompletedTask;

}

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Models:**

**LoginModel.cs**

namespace JwtAuthDemo.Models

{

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

public string Role { get; set; }

}

}

**Controllers:**

[**AdminController.cs**](http://admincontroller.cs)

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class AdminController : ControllerBase

{

[HttpGet("dashboard")]

[Authorize(Roles = "Admin")]

public IActionResult GetAdminDashboard()

{

return Ok("Welcome to the admin dashboard.");

}

}

}

**AuthController.cs**

using JwtAuthDemo.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using JwtAuthDemo.Models;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly IConfiguration \_configuration;

public AuthController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (IsValidUser(model.Username, model.Password, model.Role))

{

var token = GenerateJwtToken(model.Username, model.Role);

return Ok(new { Token = token });

}

return Unauthorized("Invalid credentials");

}

private bool IsValidUser(string username, string password, string role)

{

var users = new List<LoginModel>

{

new LoginModel { Username = "admin", Password = "admin123", Role = "Admin" },

new LoginModel { Username = "student", Password = "student123", Role = "Student" }

};

return users.Any(u =>

u.Username == username &&

u.Password == password &&

u.Role == role);

}

private string GenerateJwtToken(string username, string role)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username),

new Claim(ClaimTypes.Role, role)

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_configuration["Jwt:Key"]));

var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

var token = new JwtSecurityToken(

issuer: \_configuration["Jwt:Issuer"],

audience: \_configuration["Jwt:Audience"],

claims: claims,

expires: DateTime.Now.AddSeconds(30),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

**SecureController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class SecureController : ControllerBase

{

[HttpGet("data")]

[Authorize]

public IActionResult GetSecureData()

{

return Ok("This is protected data.");

}

}

}

**AppSettings.json**

{

"Jwt": {

"Key": "ThisIsASecretKeyForJwtToken123456",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 1

},

"Logging": {

"LogLevel": {

"Default": "Information",

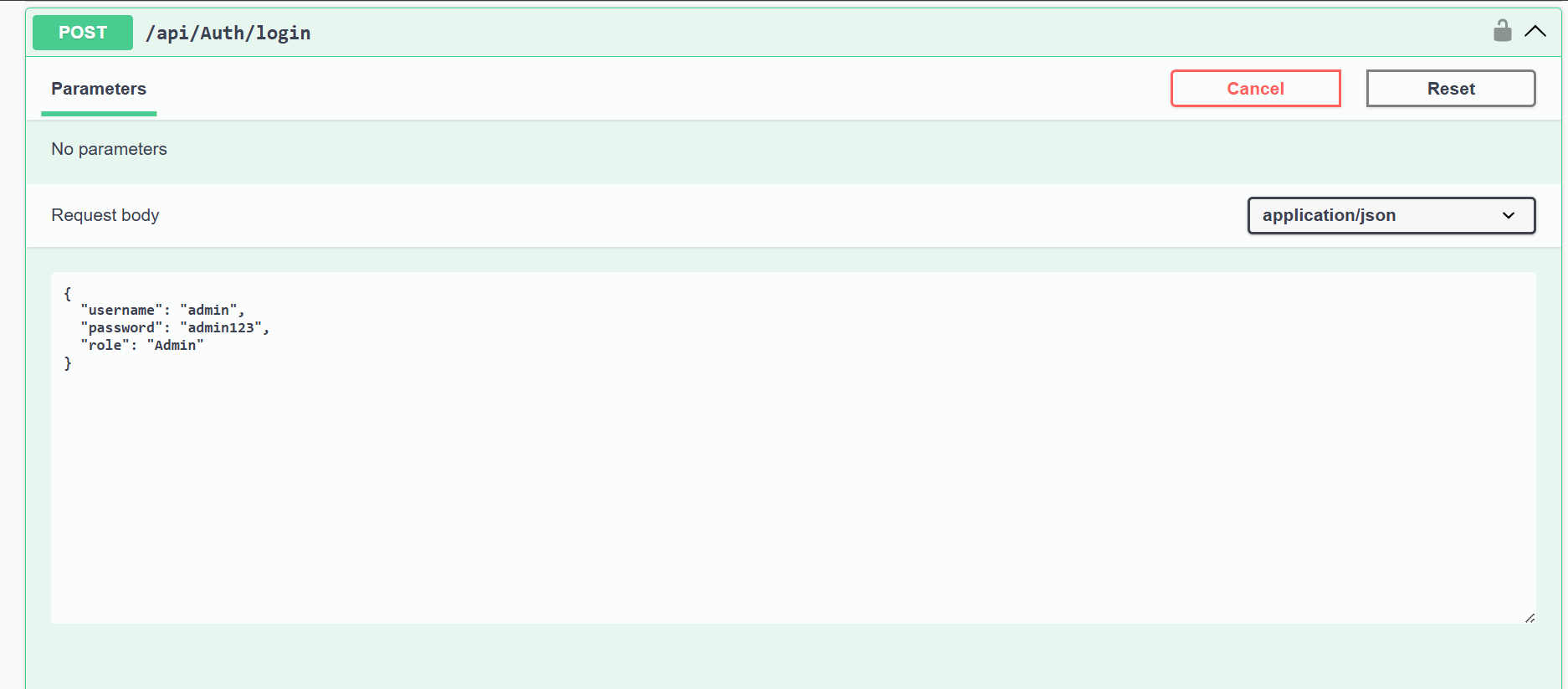
"Microsoft.AspNetCore": "Warning"

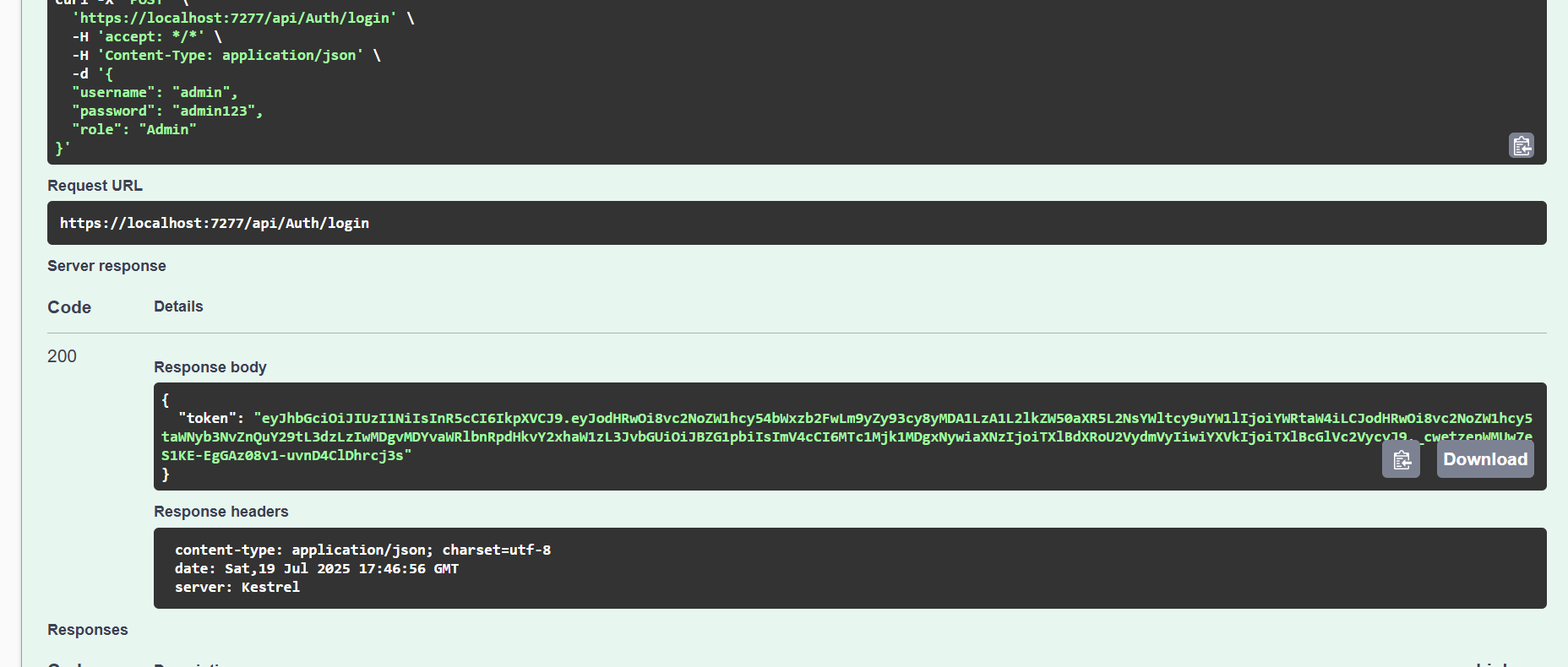
}

},

"AllowedHosts": "\*"

}





**Question 2: Secure an API Endpoint Using JWT**

**SecureController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class SecureController : ControllerBase

{

[HttpGet("data")]

[Authorize]

public IActionResult GetSecureData()

{

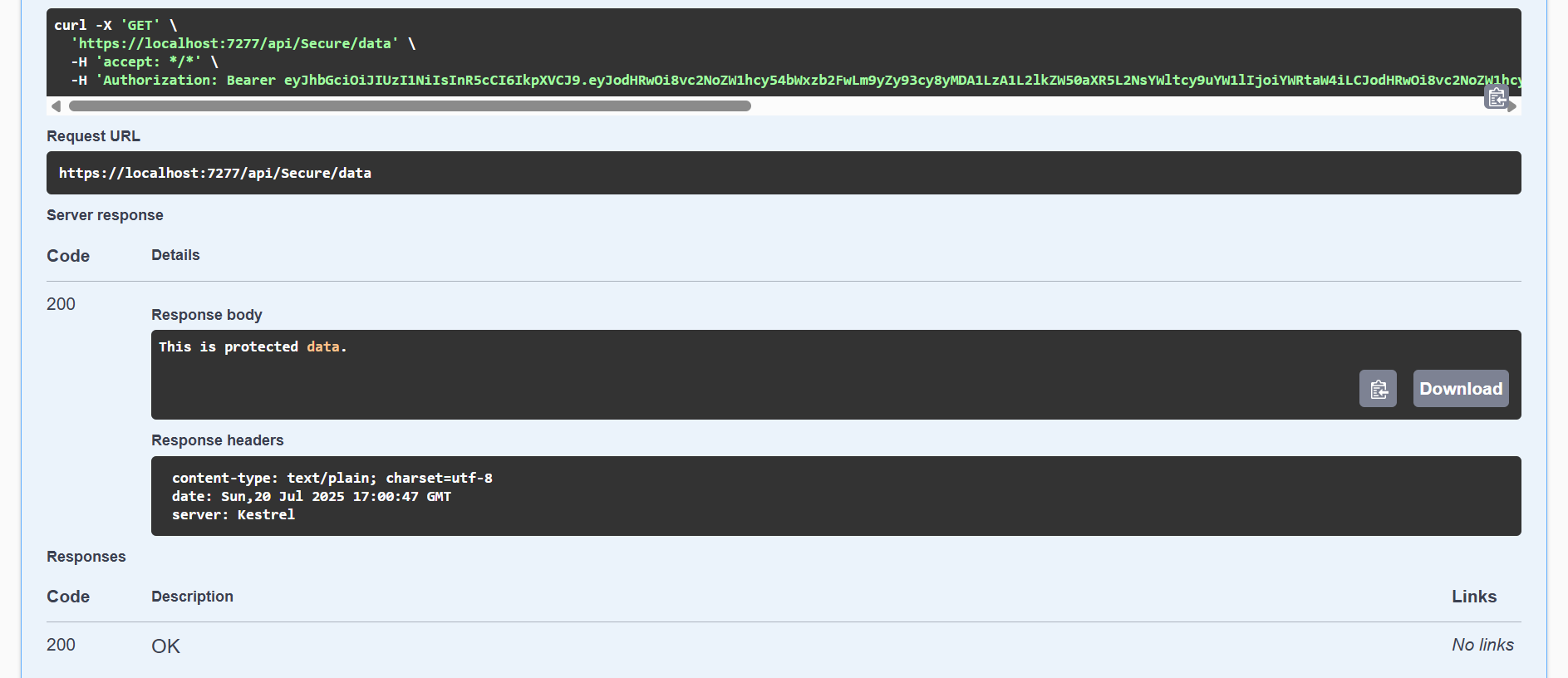
return Ok("This is protected data.");

}

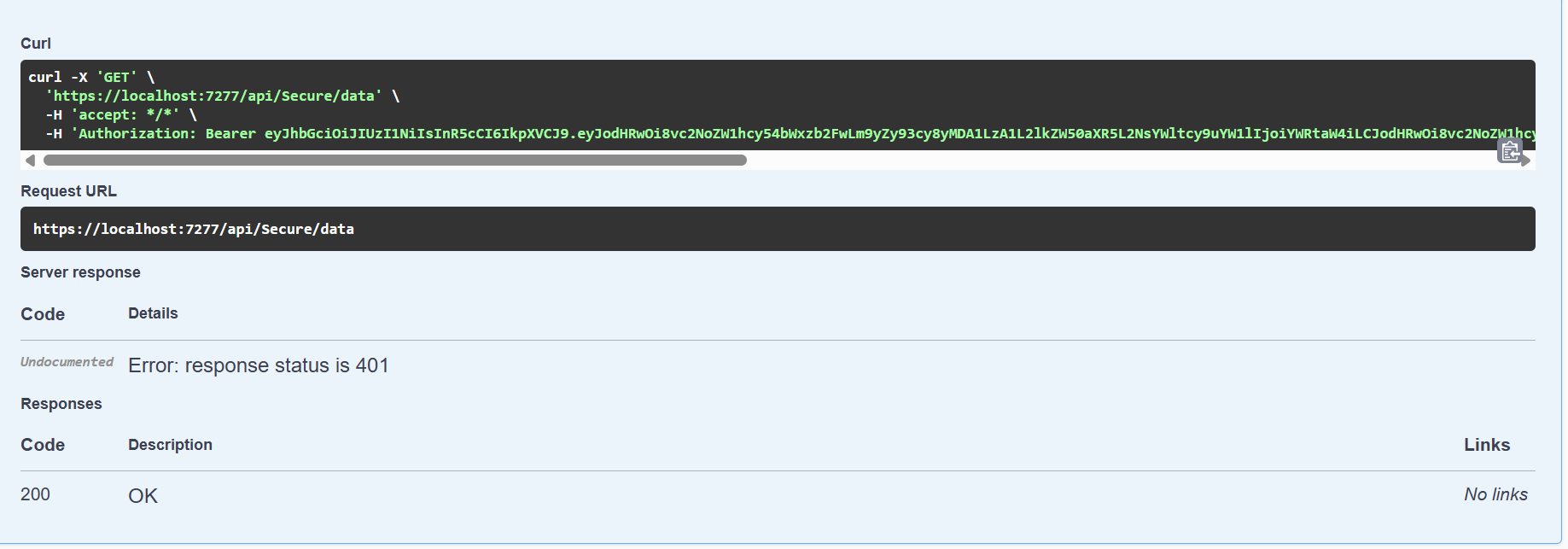
}

}

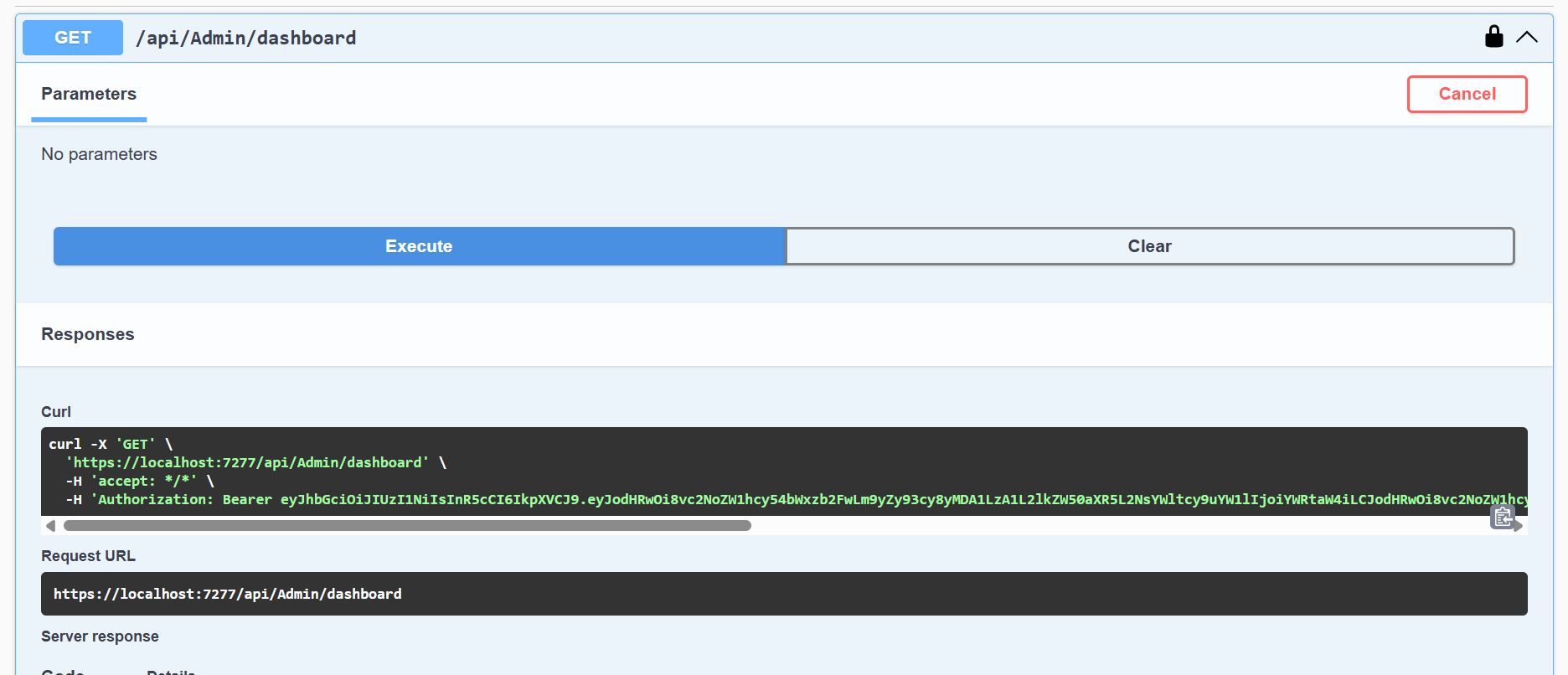
With Valid token

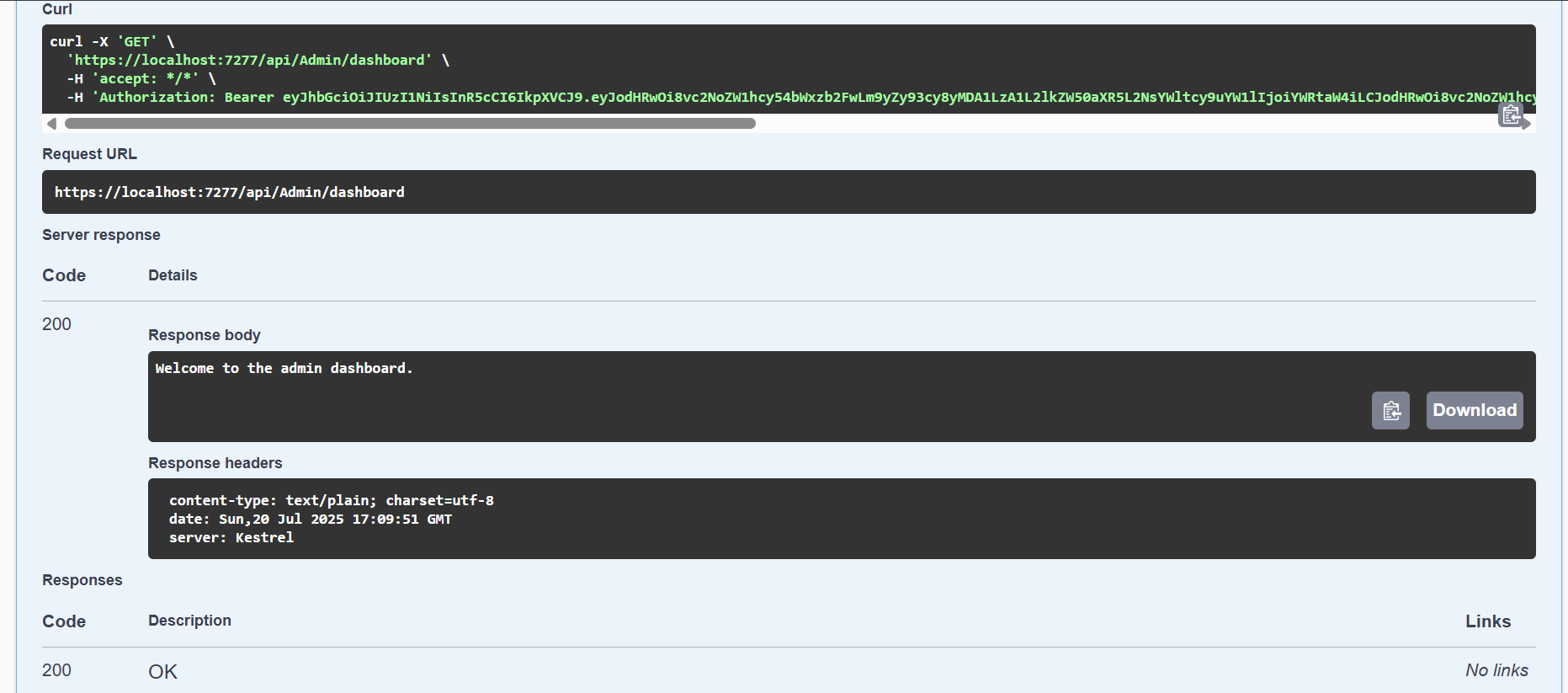


With Invalid token

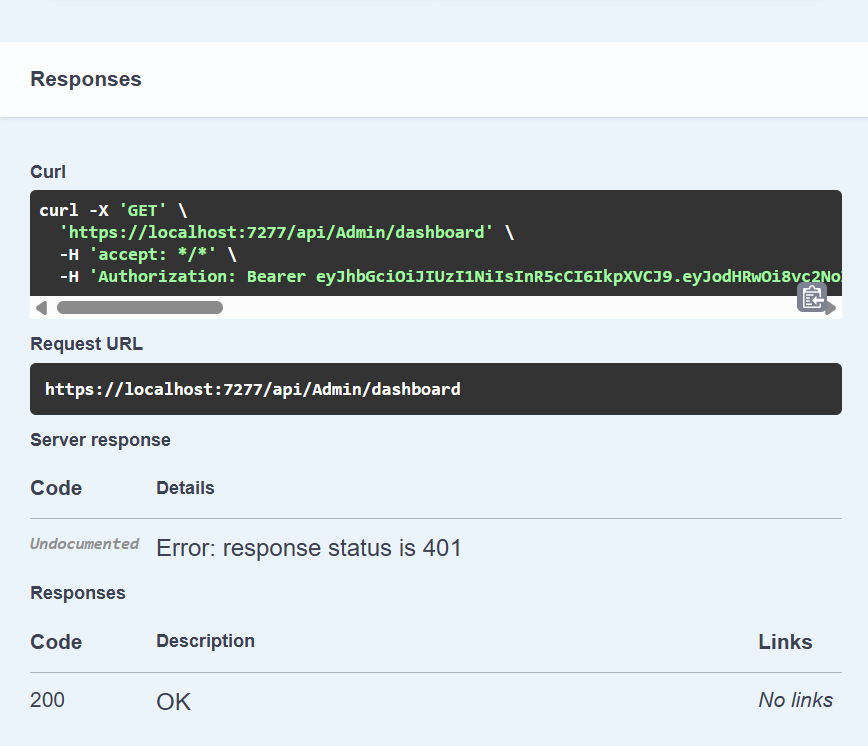


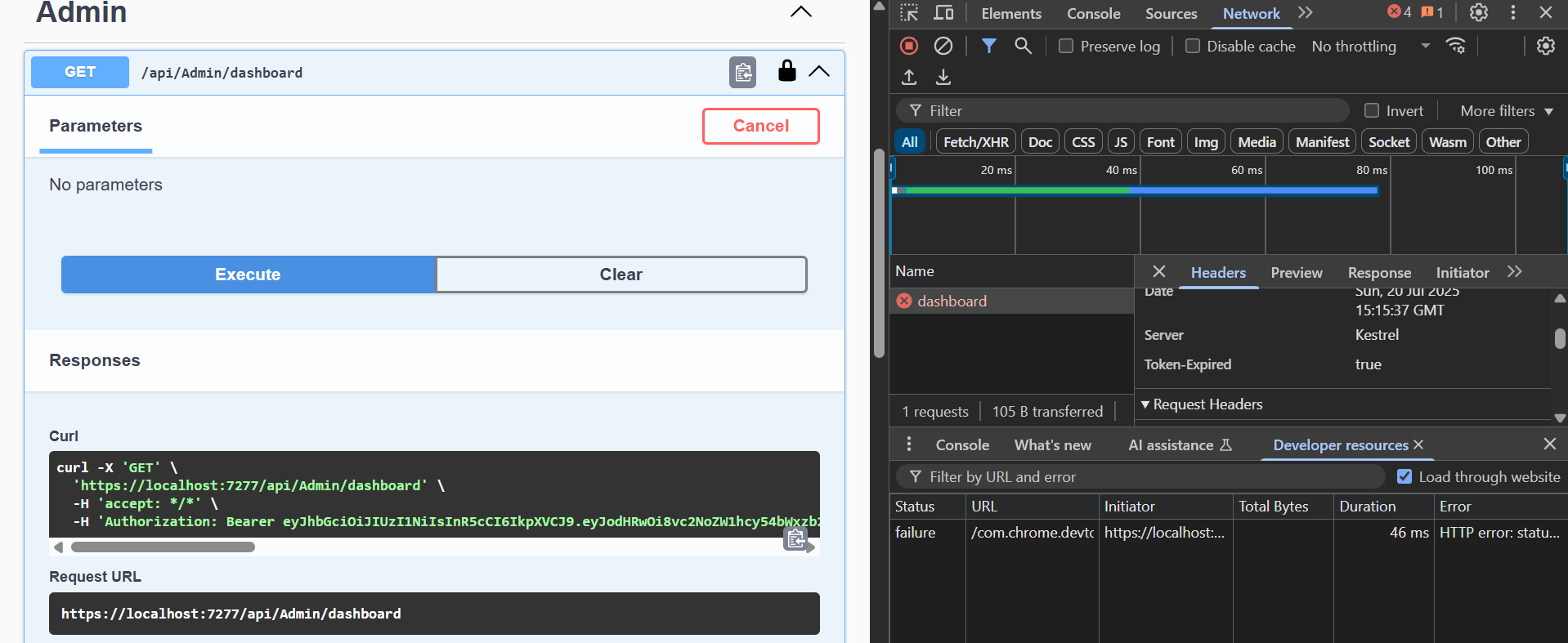
**Question 3: Add Role-Based Authorization**

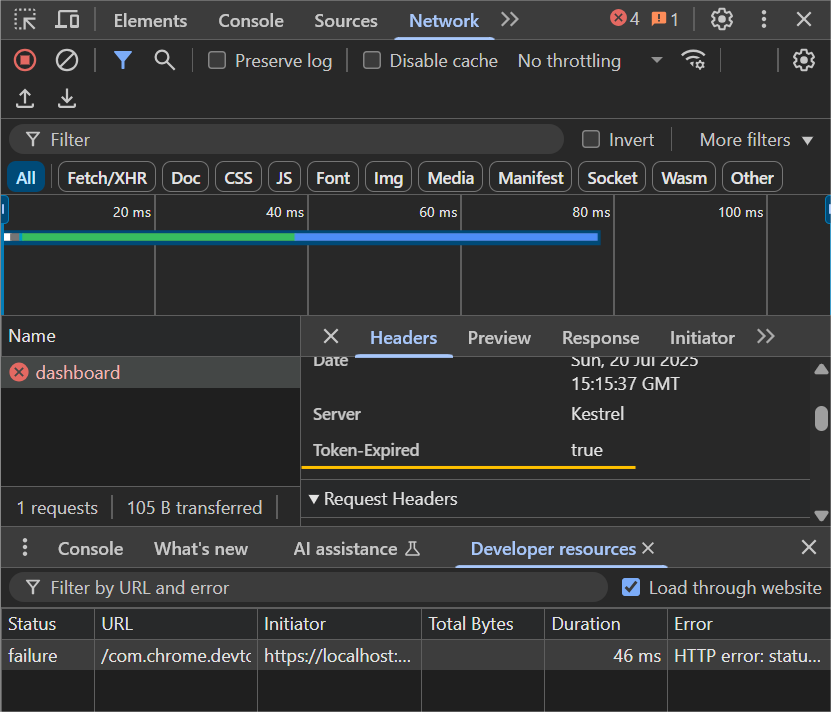
****

****

**Question 4: Validate JWT Token Expiry and Handle Unauthorized Access**

****

****

****