DotNet-FSE Mandatory Hands-On

WEEK-5 **NAME:** Sri Ranjani Priya P

EXERCISE 1:

Implement a secure Web API in ASP.NET Core that supports JWT-based authentication and role-based authorization. The API should support login functionality and allow access to specific endpoints based on the user's role (e.g., Admin or User).

CODE:(C#)

File: Models/LoginModel.cs

```
namespace JwtAuthWithRoles.Models
{
   public class LoginModel
   {
      public string Username { get; set; }
      public string Password { get; set; }
   }
}
```

File: Program.cs

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

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 = builder.Configuration["Jwt:Issuer"],
      ValidAudience = builder.Configuration["Jwt:Audience"],
      IssuerSigningKey = new SymmetricSecurityKey(
```

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

builder.Services.AddAuthorization();
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

app.UseHttpsRedirection();
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

FILE: appsettings.json:

```
{
  "Jwt": {
    "Key": "ThisIsASecretKeyForJwt",
    "Issuer": "https://localhost:7006",
    "Audience": "https://localhost:7006"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

FILE: Controllers/AuthController.cs:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using JwtAuthWithRoles.Models;
```

```
namespace JwtAuthWithRoles.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 user)
      {
         if (user.Username == "admin" && user.Password == "password")
         {
            var token = GenerateJwtToken(user.Username, "Admin");
            return Ok(new { token });
         }
         else if (user.Username == "user" && user.Password == "password")
         {
            var token = GenerateJwtToken(user.Username, "User");
            return Ok(new { token });
         }

         return Unauthorized();
      }

      private string GenerateJwtToken(string username, string role)
      {
         var securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]));
         var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

         var claims = new[]
         {
            new Claim(ClaimTypes.Name, username),
            new Claim(ClaimTypes.Role, role),
            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
         };

         var token = new JwtSecurityToken(
```

```
            issuer: _configuration["Jwt:Issuer"],
            audience: _configuration["Jwt:Audience"],
            claims: claims,
            expires: DateTime.Now.AddMinutes(60),
            signingCredentials: credentials
        );

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
  }
}
```

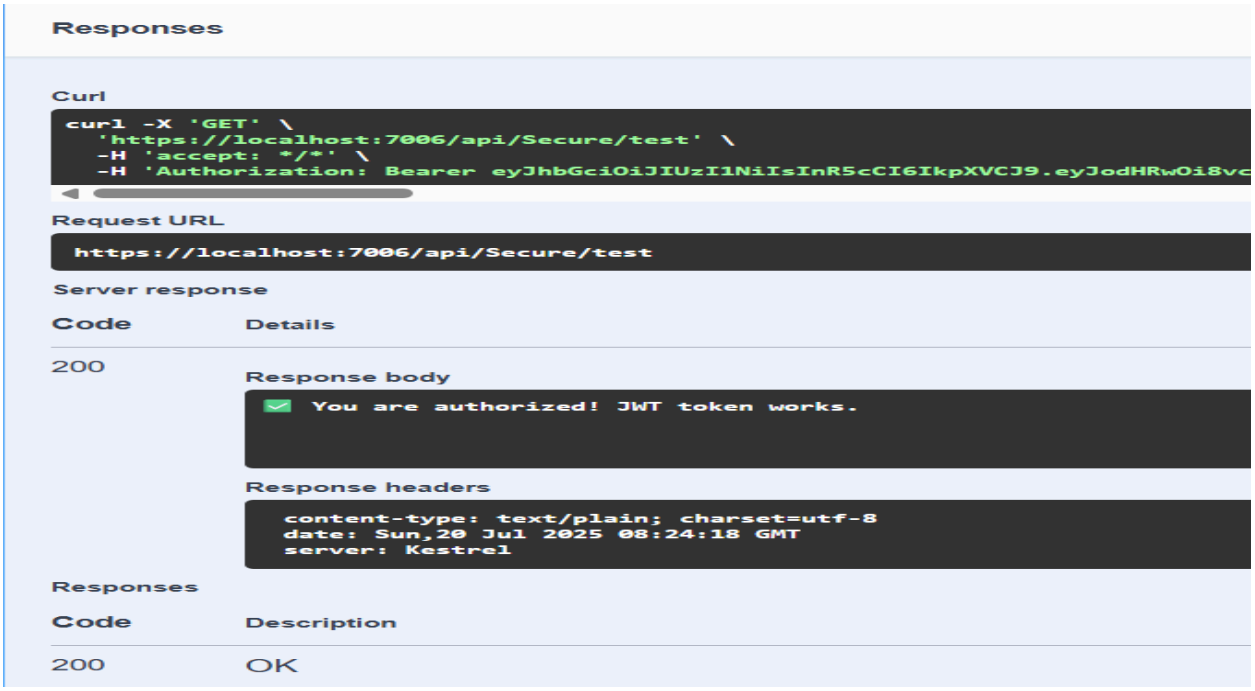
FILE: Controllers/SecureController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace JwtAuthWithRoles.Controllers
{
  [ApiController]
  [Route("api/[controller]")]
  public class SecureController : ControllerBase
  {
    [HttpGet("test")]
    [Authorize]
    public IActionResult Test()
    {
      return Ok("✅ You are authorized! JWT token works.");
    }

    [HttpGet("admin")]
    [Authorize(Roles = "Admin")]
    public IActionResult AdminOnly()
    {
      return Ok("✅ Welcome, Admin!");
    }

    [HttpGet("user")]
    [Authorize(Roles = "User")]
    public IActionResult UserOnly()
    {
      return Ok("✅ Welcome, User!");
    }
 }}
```

OUTPUT:

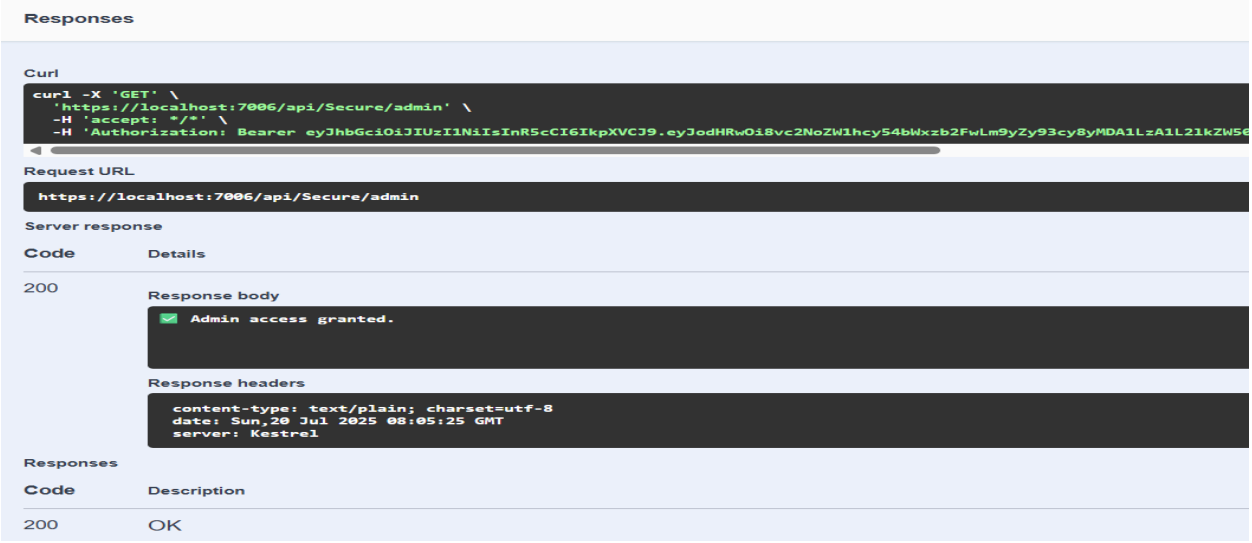
Login as User:


```
{
 "username": "User_name",
 "password": "password"
}
```

Login as Admin:


```
{
 "username": "admin",
 "password": "password"
}
```

JWT Secured Endpoint – 403 Forbidden Error(for unauthorized role access):