**WEEK – 5 Assignments**

**Microservices Architecture using ASP. NET Core WebAPI**

**Mandatory hands-on :-**

**1. Microservices - JWT :**

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

**Scenario:**

You are building a microservice that requires secure login. You need to implement JWT based authentication.

Steps:

1. Create a new ASP.NET Core Web API project.

2. Add a `User` model and a login endpoint.

3. Generate a JWT token upon successful login.

4. Secure an endpoint using `[Authorize]`.

**Code :**

**Appsettings.json :**

| **{ "Jwt": { "Key": "ThisIsASecretKeyForJwtToken", "Issuer": "MyAuthServer", "Audience": "MyApiUsers", "DurationInMinutes": 60 } }** |
| --- |

[**Program.cs**](http://employeecontroller.cs) **:**

| **using Microsoft.AspNetCore.Authentication.JwtBearer; using Microsoft.IdentityModel.Tokens; using System.Text;  var builder = WebApplication.CreateBuilder(args);  var jwtKey = builder.Configuration["Jwt:Key"]  ?? throw new InvalidOperationException("JWT Key is missing in configuration");  var jwtIssuer = builder.Configuration["Jwt:Issuer"]; var jwtAudience = builder.Configuration["Jwt:Audience"];  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 = jwtIssuer,  ValidAudience = jwtAudience,  IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey))  }; });  builder.Services.AddAuthorization(); builder.Services.AddControllers();  var app = builder.Build();  app.UseHttpsRedirection(); app.UseAuthentication(); app.UseAuthorization();  app.MapControllers();  app.Run();** |
| --- |

[**LoginModel.cs**](http://loginmodel.cs) **:**

| **namespace JwtAuthDemo.Models {  public class LoginModel  {  public string? Username { get; set; }  public string? Password { get; set; }  } }** |
| --- |

[**AuthController.cs**](http://authcontroller.cs) **:**

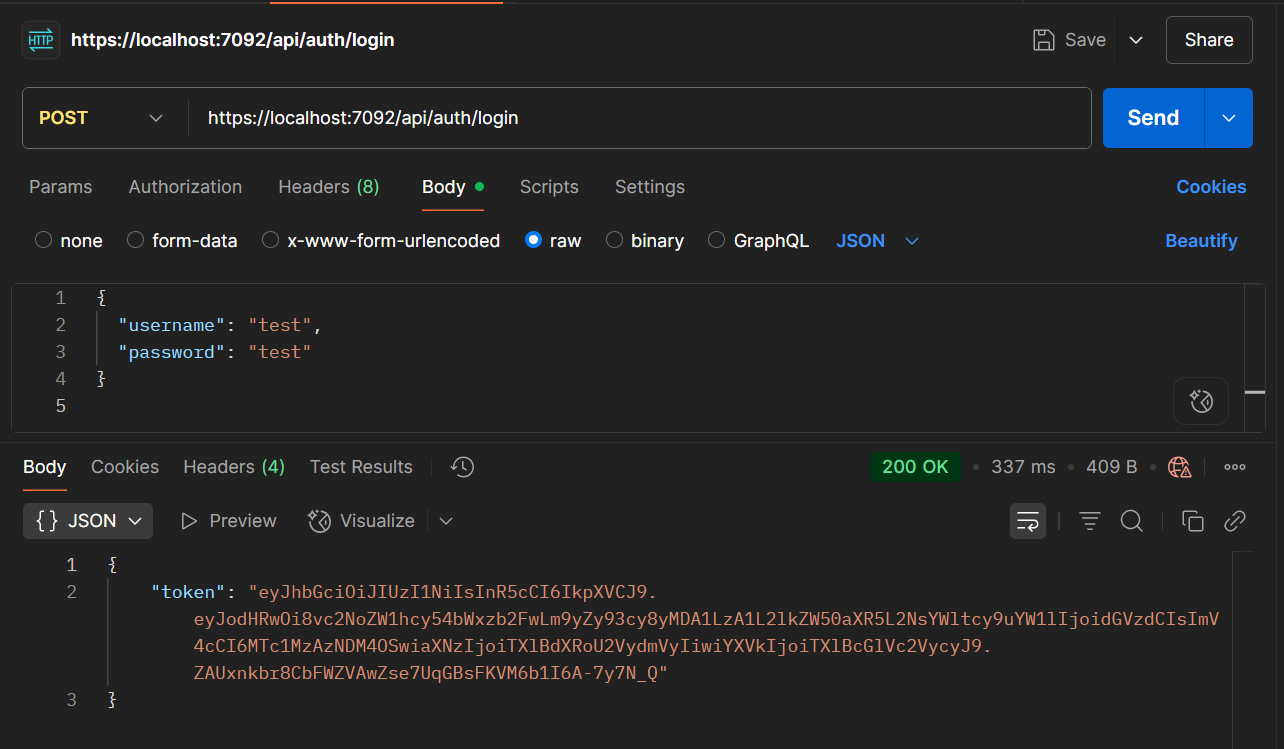
| **using JwtAuthDemo.Models; using Microsoft.AspNetCore.Mvc; using Microsoft.IdentityModel.Tokens; using System.IdentityModel.Tokens.Jwt; using System.Security.Claims; using System.Text;  namespace JwtAuthDemo.Controllers {  [ApiController]  [Route("api/[controller]")]  public class AuthController : ControllerBase  {  private readonly IConfiguration \_config;   public AuthController(IConfiguration config)  {  \_config = config;  }   [HttpPost("login")]  public IActionResult Login([FromBody] LoginModel model)  {  if (IsValidUser(model))  {  var token = GenerateJwtToken(model.Username!);  return Ok(new { Token = token });  }   return Unauthorized("Invalid credentials.");  }   private bool IsValidUser(LoginModel model)  {  return model is not null &&  !string.IsNullOrWhiteSpace(model.Username) &&  model.Username == model.Password;   }   private string GenerateJwtToken(string username)  {  var claims = new[]  {  new Claim(ClaimTypes.Name, username)  };   var key = new SymmetricSecurityKey(  Encoding.UTF8.GetBytes(\_config["Jwt:Key"]!));   var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);   var token = new JwtSecurityToken(  issuer: \_config["Jwt:Issuer"],  audience: \_config["Jwt:Audience"],  claims: claims,  expires: DateTime.Now.AddMinutes(60),  signingCredentials: creds  );   return new JwtSecurityTokenHandler().WriteToken(token);  }  } }** |
| --- |

[**SecureDataController.cs**](http://securedatacontroller.cs) **:**

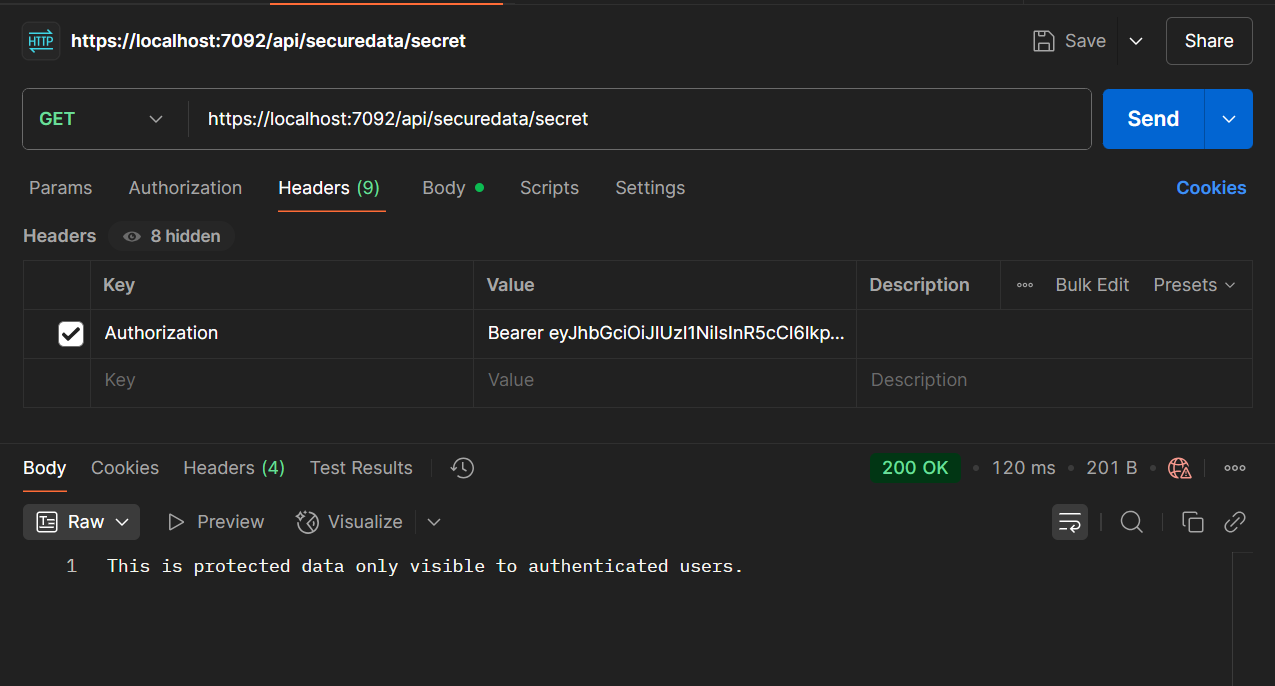
| **using Microsoft.AspNetCore.Authorization; using Microsoft.AspNetCore.Mvc;  namespace JwtAuthDemo.Controllers {  [Authorize]  [ApiController]  [Route("api/[controller]")]  public class SecureDataController : ControllerBase  {  [HttpGet("secret")]  public IActionResult GetSecret()  {  return Ok("This is protected data only visible to authenticated users.");  }  } }** |
| --- |

**OUTPUT :**

**POST : https://localhost:7092/api/auth/login :**

****

**GET : https://localhost:7092/api/securedata/secret :**

****