**Week\_5 Handson**

**Name: Rajesh A**

**Superset Id: 6384182**

# JWT Authentication Microservice – ASP.NET Core

## Overview

A secure Web API built with **ASP.NET Core** that:

* Authenticates users via login
* Issues **JSON Web Tokens (JWT)**
* Validates tokens in requests
* Enables role-based access
* Integrates with Postman for testing

## JWT Configuration (appsettings.json)

Use a base64-encoded 256-bit key

"Jwt": {

"Key": "3fZk+rBliQOz+9rKJZsWpd8BYYUNJjYULyWKgMGnRAQ=",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

}

## Program Setup (Program.cs)

using System.Text;

using System.Security.Cryptography;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

var builder = WebApplication.CreateBuilder(args);

var jwtSettings = builder.Configuration.GetSection("Jwt");

var keyBytes = Convert.FromBase64String(jwtSettings["Key"]);

builder.Services.AddControllers();

builder.Services.AddOpenApi();

builder.Services.AddAuthentication("Bearer")

.AddJwtBearer("Bearer", options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = jwtSettings["Issuer"],

ValidAudience = jwtSettings["Audience"],

IssuerSigningKey = new SymmetricSecurityKey(keyBytes)

};

options.Events = new JwtBearerEvents

{

OnAuthenticationFailed = context =>

{

if (context.Exception is SecurityTokenExpiredException)

{

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

}

return Task.CompletedTask;

}

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

if (app.Environment.IsDevelopment()) app.MapOpenApi();

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

## Login Model (LoginModel.cs)

namespace JwtAuthMicroservice.Models

{

public class LoginModel

{

public required string Username { get; set; }

public required string Password { get; set; }

}

}

## Auth Controller (AuthController.cs)

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using JwtAuthMicroservice.Models;

namespace JwtAuthMicroservice.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 (model.Username == "admin" && model.Password == "password")

{

var token = GenerateJwtToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized();

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username),

new Claim(ClaimTypes.Role, "Admin")

};

var key = Convert.FromBase64String(\_config["Jwt:Key"]);

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

var token = new JwtSecurityToken(

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

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

claims: claims,

expires: DateTime.Now.AddMinutes(Convert.ToDouble(\_config["Jwt:DurationInMinutes"])),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

## Secure Endpoint (SecureController.cs)

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthMicroservice.Controllers

{

[ApiController]

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

public class SecureController : ControllerBase

{

[HttpGet("data")]

[Authorize]

public IActionResult GetSecureData()

{

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

}

}

}

## Admin Endpoint (AdminController.cs)

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthMicroservice.Controllers

{

[ApiController]

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

public class AdminController : ControllerBase

{

[HttpGet("dashboard")]

[Authorize(Roles = "Admin")]

public IActionResult GetDashboard()

{

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

}

}

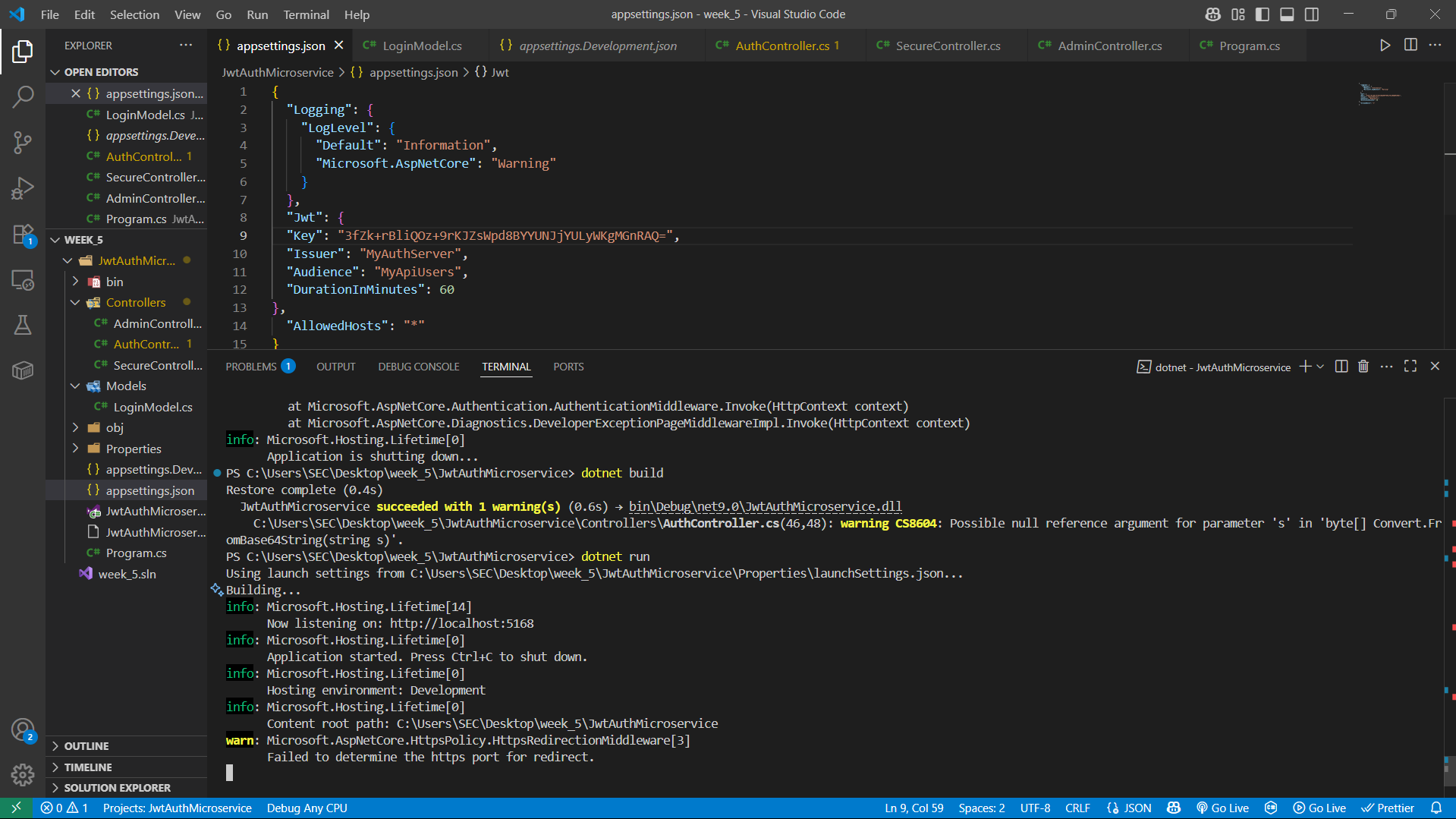
}

## Build and Run

dotnet build

dotnet run

**Output:**



## Testing with Postman

### 1. Login

**POST** http://localhost:5168/api/auth/login  
Header:

Content-Type: application/json

Body:

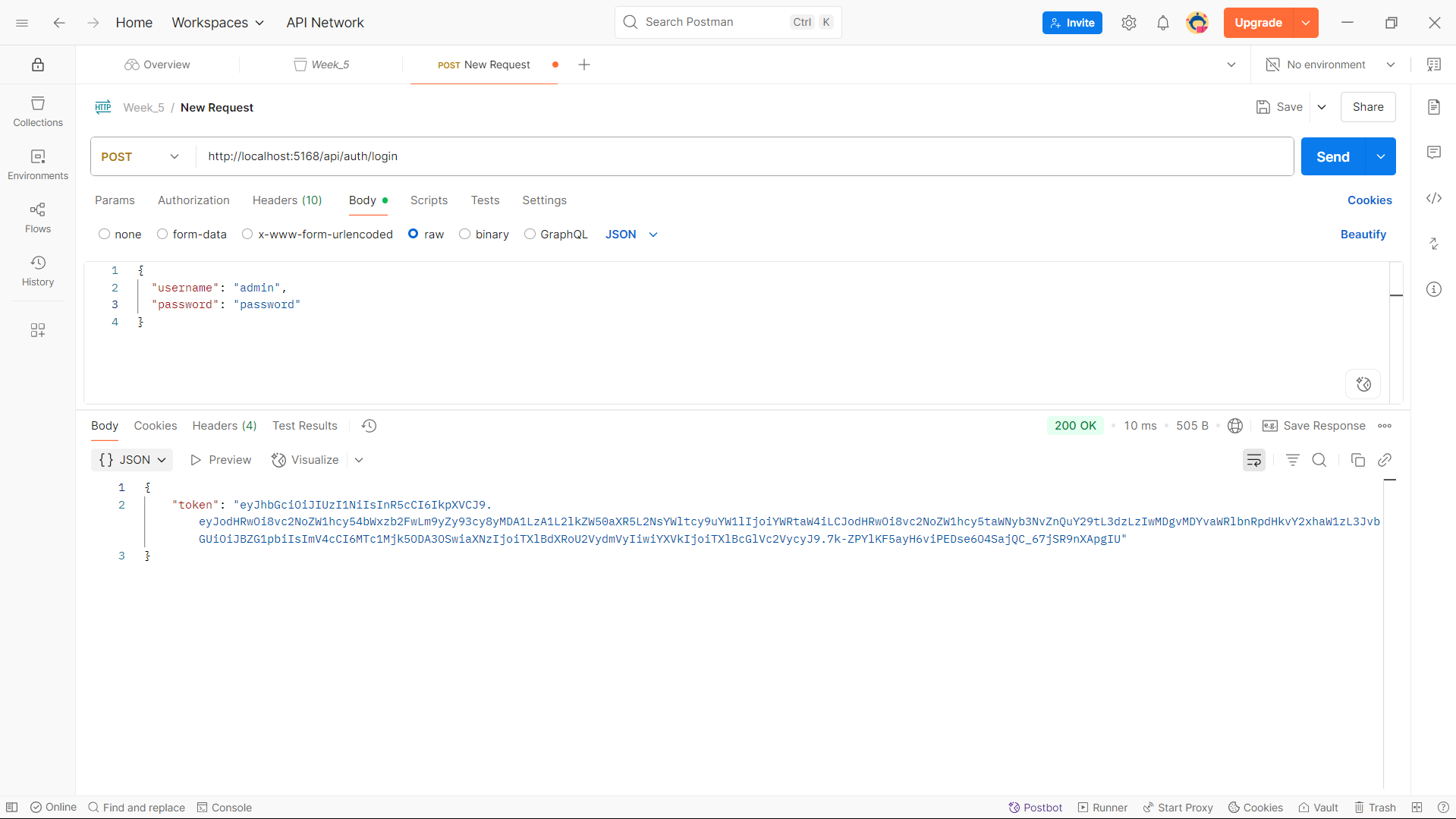
{

"username": "admin",

"password": "password"

}

**Output:**

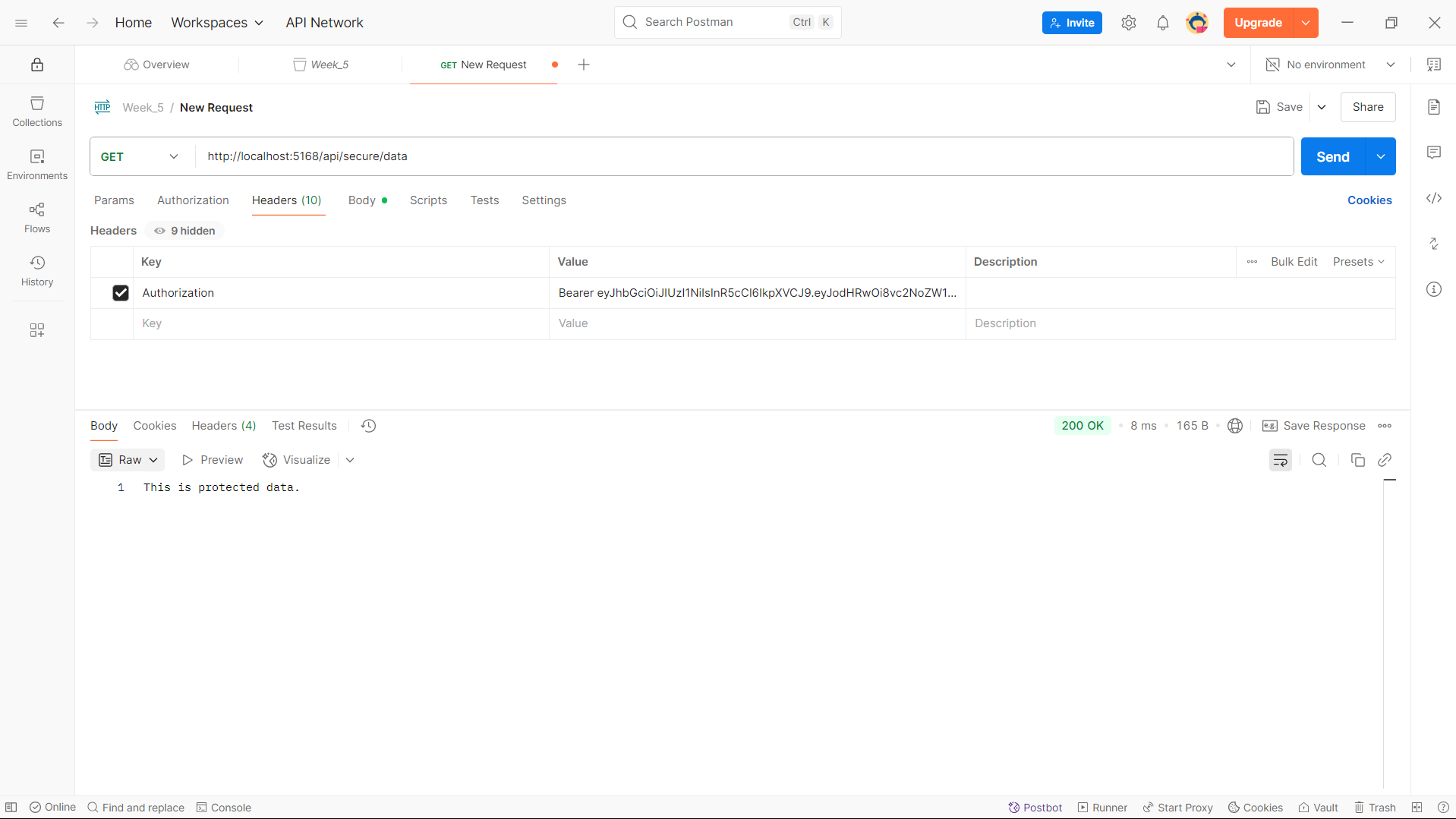


### **2. Access Secure Endpoint**

**GET** http://localhost:5168/api/secure/data  
Header:

Authorization: Bearer <your\_token>

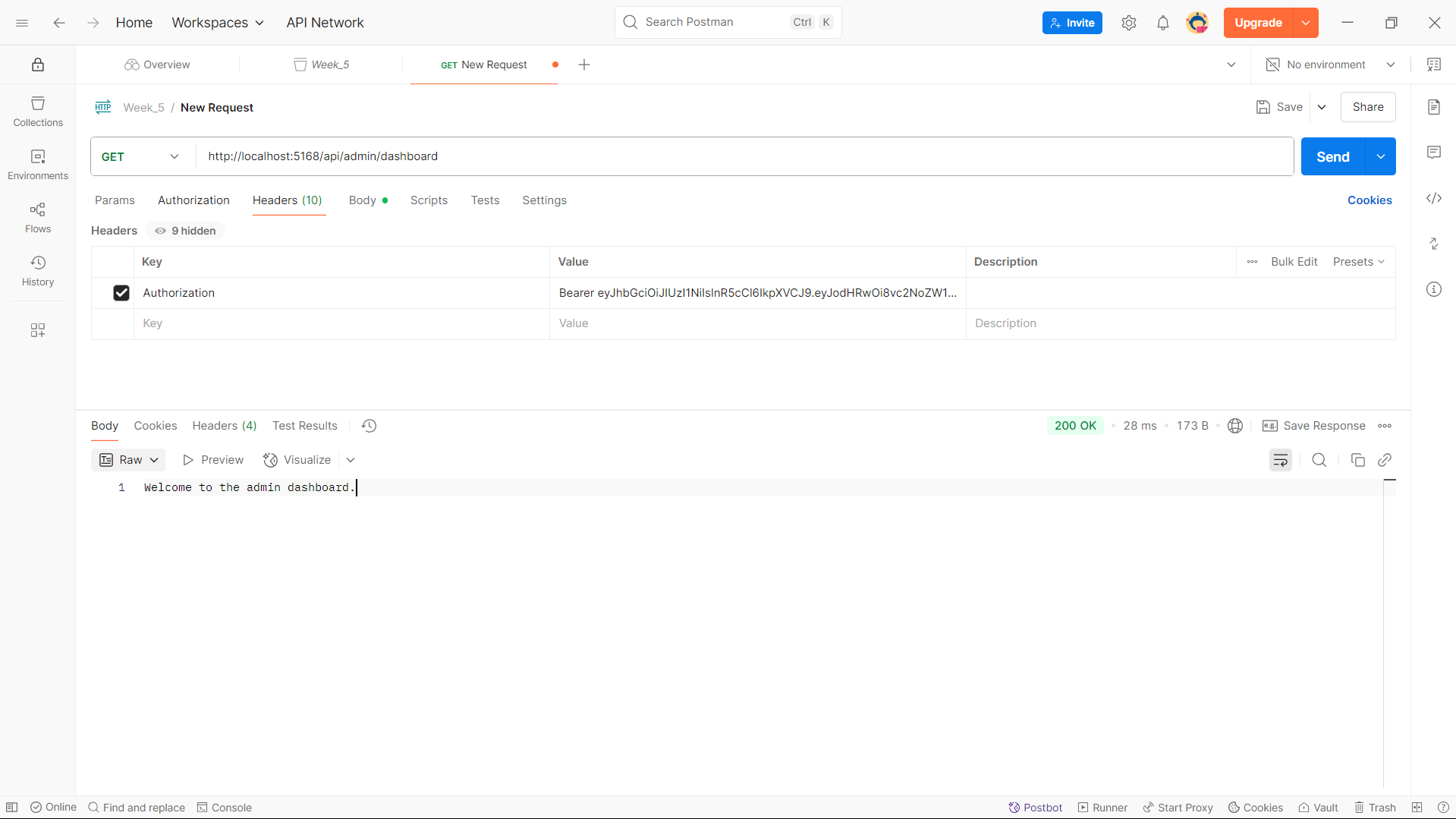
**Output:**



### 3. Access Admin Dashboard

**GET** <http://localhost:5168/api/admin/dashboard>

Same header



**Thank you**