WEEK\_4-Handson

Rajesh A

6384182

# Lab 5 - JWT Authentication & Authorization

## Objectives

• Explain and enable CORS for external access.  
• Implement JWT-based authentication using tokens.  
• Use [Authorize], [AllowAnonymous], and Roles with JWT claims.  
• Secure Web API endpoints with token validation.  
• Validate JWT using Postman and Swagger.

## Install Required NuGet Package

Install the following NuGet package in your project:

* Microsoft.AspNetCore.Authentication.JwtBearer

## 2. Configure JWT in Program.cs

var builder = WebApplication.CreateBuilder(args);  
  
var key = "mysuperdupersecretjwtkeywith32chars!"; // at least 32 chars  
var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(key));  
  
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)  
 .AddJwtBearer(options =>  
 {  
 options.TokenValidationParameters = new TokenValidationParameters  
 {  
 ValidateIssuer = true,  
 ValidateAudience = true,  
 ValidateLifetime = true,  
 ValidateIssuerSigningKey = true,  
 ValidIssuer = "mySystem",  
 ValidAudience = "myUsers",  
 IssuerSigningKey = symmetricSecurityKey  
 };  
 });  
  
builder.Services.AddAuthorization();  
  
var app = builder.Build();  
app.UseAuthentication();  
app.UseAuthorization();

## 3. Create AuthController.cs

[ApiController]  
[Route("api/[controller]")]  
[AllowAnonymous]  
public class AuthController : ControllerBase  
{  
 [HttpGet("token")]  
 public IActionResult GetToken()  
 {  
 var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretjwtkeywith32chars!"));  
 var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);  
  
 var claims = new List<Claim>  
 {  
 new Claim(ClaimTypes.Role, "Admin"),  
 new Claim("UserId", "1")  
 };  
  
 var token = new JwtSecurityToken(  
 issuer: "mySystem",  
 audience: "myUsers",  
 claims: claims,  
 expires: DateTime.Now.AddMinutes(2),  
 signingCredentials: credentials);  
  
 return Ok(new JwtSecurityTokenHandler().WriteToken(token));  
 }  
}

## Secure EmployeeController.cs

Update the controller to require authorization using roles:

[Authorize(Roles = "Admin,POC")]  
[Route("api/[controller]")]  
[ApiController]  
public class EmployeeController : ControllerBase  
{  
 // Existing GET/PUT code  
}

## Testing Steps

Use Swagger and Postman to test token generation and API access.

In Swagger:

• Call /api/Auth/token

• Copy the token

• Click Authorize and paste Bearer token

In Postman:

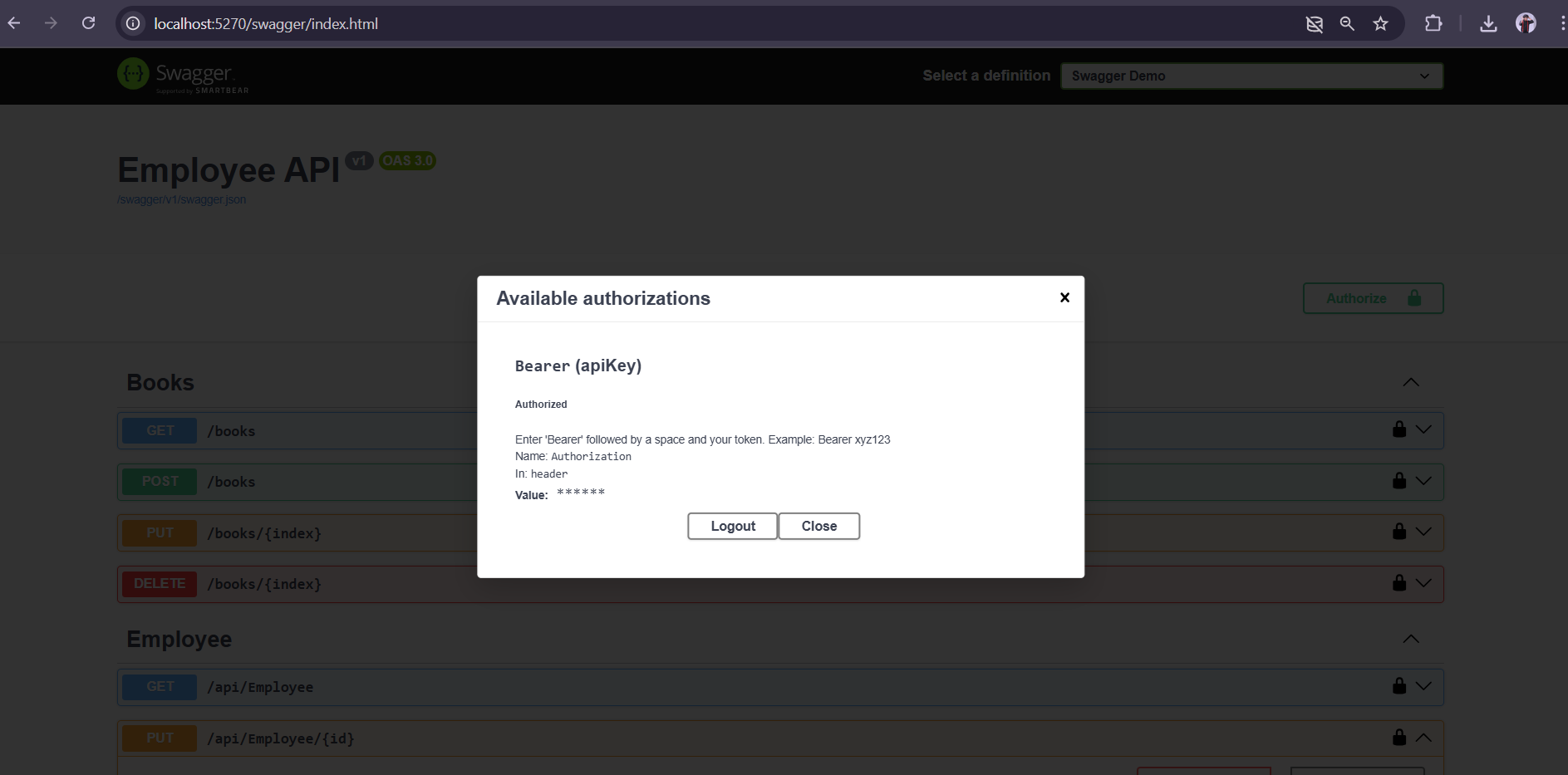
• Use GET/PUT with Authorization header: Bearer <token>

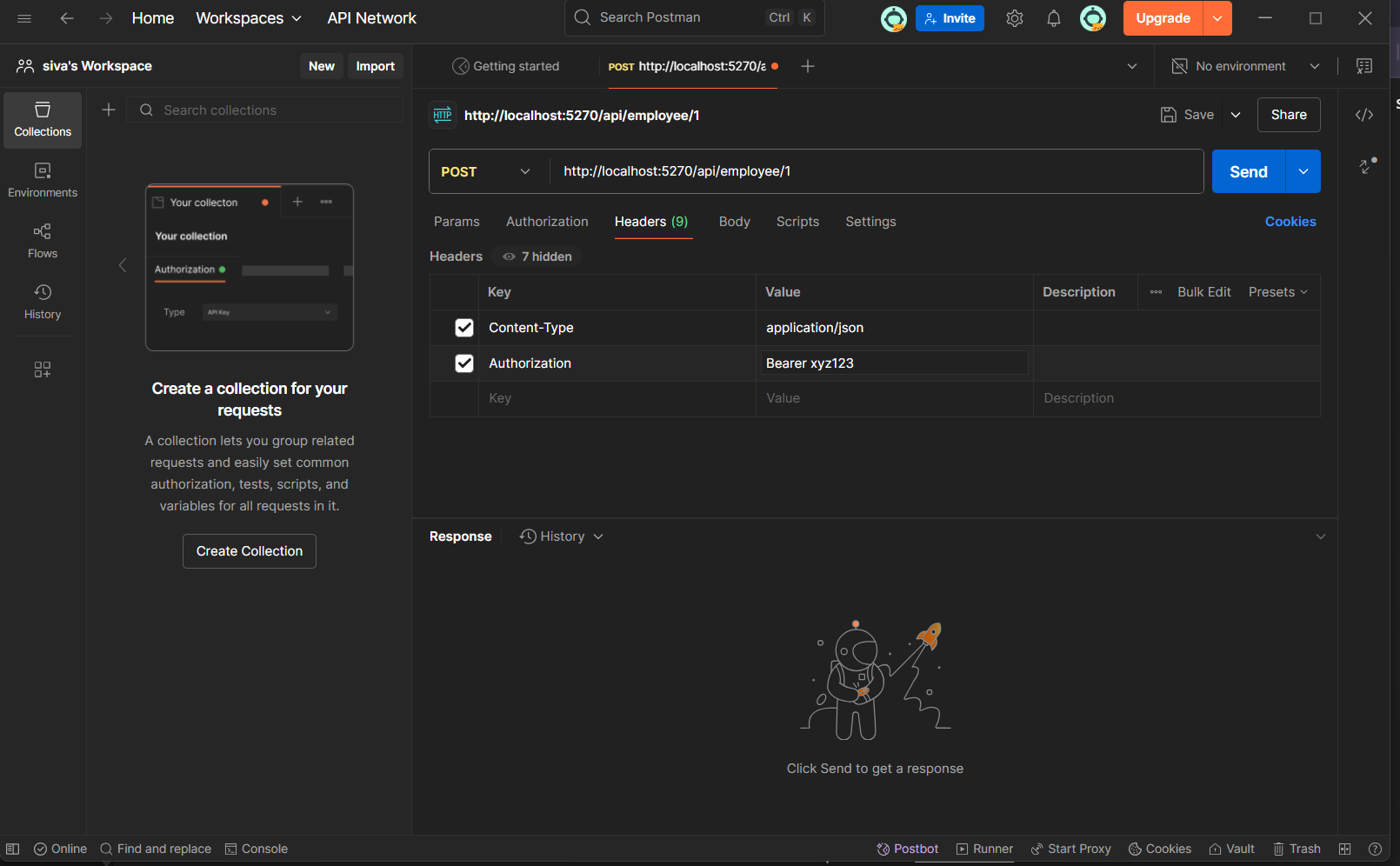
• Test for 200 OK with valid token

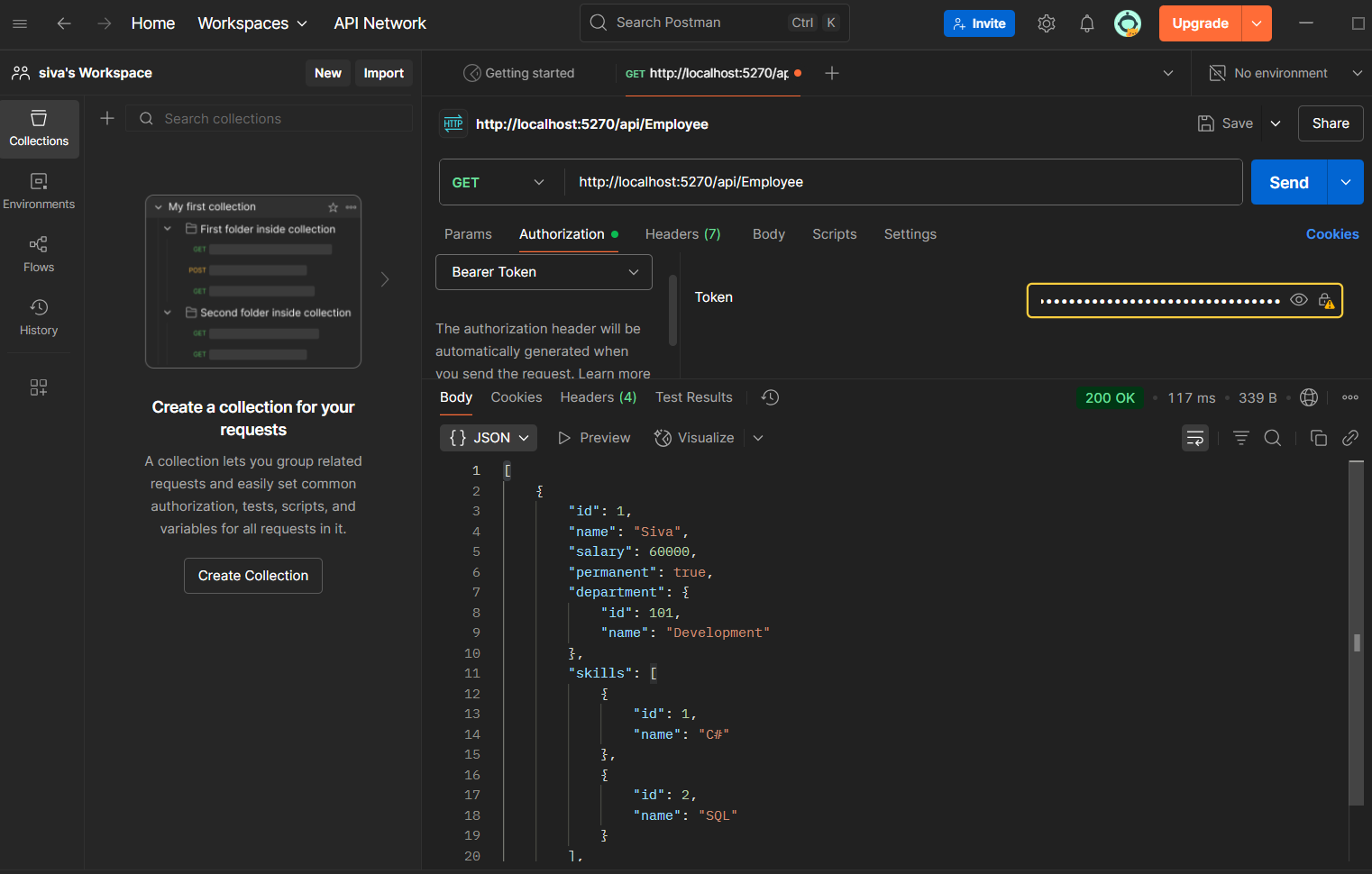
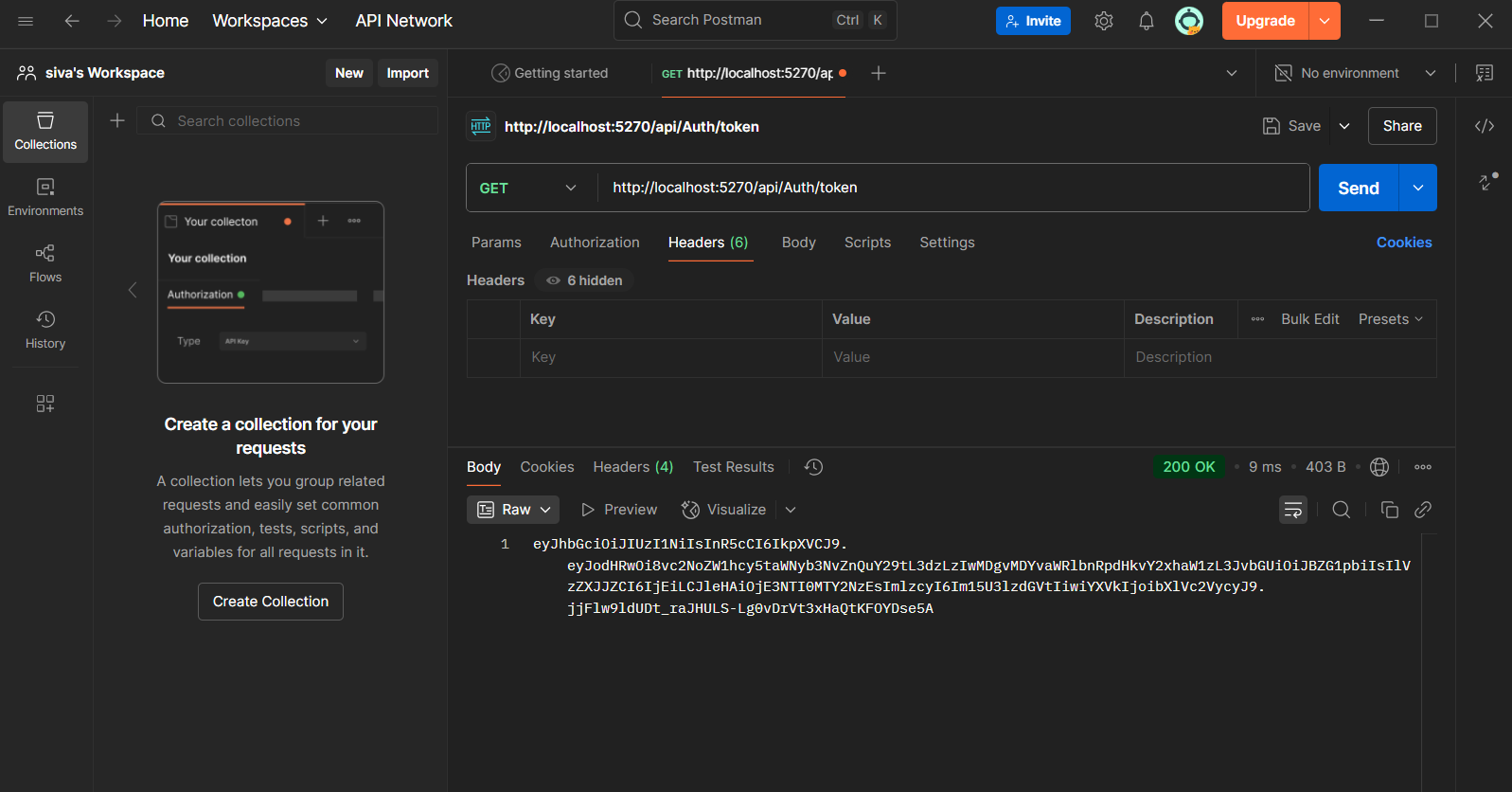
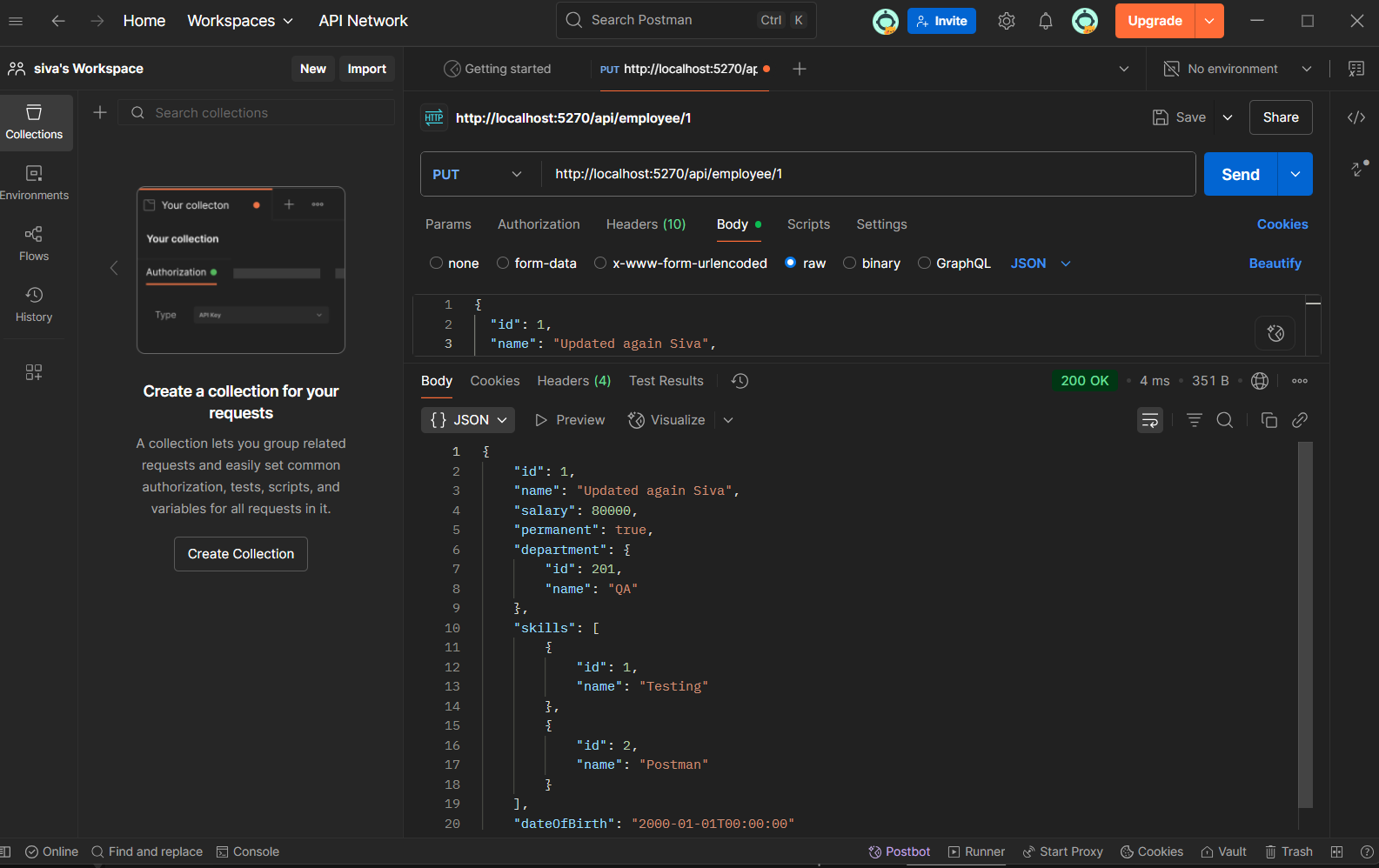
• Wait 2 mins and try again to get 401 Unauthorized

• Modify token roles and retest with mismatched roles

## Screenshots





Output:

Thank you