**Week 4 - ASP.NET Core 8.0 Web API - Hands-on**

DEBANJAN KAURI(Superset ID-6362388)

**1: First Web API using .NET Core**

Scenario:

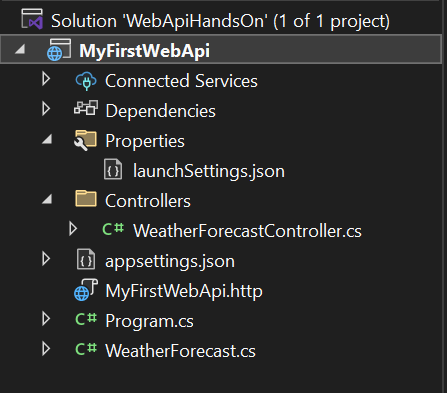
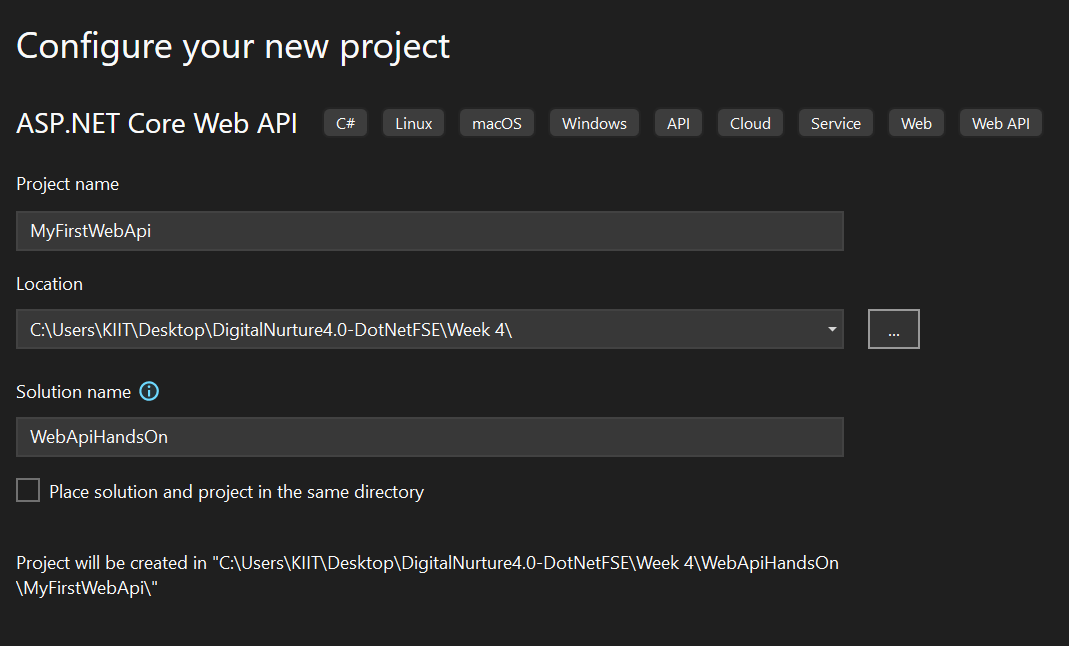
Create a basic .NET Core Web API and verify HTTP verb handling.

To achieve this:

1. Create a .NET Core Web API project using the API template.

2. Enable read/write controller actions (ValuesController).

3. Run the app and test the GET method endpoint in the browser.

**Application Creation:**

### **Controller:** *ValuesController.cs*

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebApi.Controllers

{

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

[ApiController]

public class ValuesController : ControllerBase

{

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

[HttpGet("{id}")]

public string Get(int id)

{

return $"value{id}";

}

[HttpPost]

public IActionResult Post([FromBody] string value)

{

return Ok($"Received: {value}");

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] string value)

{

return Ok($"Updated ID {id} with value: {value}");

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

return Ok($"Deleted item with ID: {id}");

}

}

}

### **Output:**

**2: Web API using .Net core with Swagger & Postman**

Scenario:

Integrate Swagger for API documentation and test using Swagger UI and Postman.

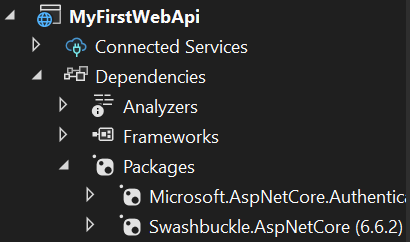
To achieve this:

1. Install and configure Swashbuckle.AspNetCore in the project.

2. Run the app and access /swagger to test the GET method.

3. Use Postman to test EmployeeController's GET method and check response/status.

4. Change controller route to Emp and re-test via Postman.

**Configuration:**

### **Code:**

*Program.cs*

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new OpenApiContact

{

Name = "Debanjan",

Email = "Deb@xyzmail.com",

Url = new Uri("https://www.example.com")

},

License = new OpenApiLicense

{

Name = "License Terms",

Url = new Uri("https://www.example.com")

}

});

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

});

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

*EmployeeController.cs*

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Http;

using System.Collections.Generic;

namespace MyFirstWebApi.Controllers

{

[ApiController]

[Route("api/emp")]

public class EmployeeController : ControllerBase

{

[HttpGet]

[ActionName("GetAll")]

[ProducesResponseType(StatusCodes.Status200OK)]

public IEnumerable<string> GetAll()

{

return new string[] { "Deb", "Adi", "Abhi" };

}

[HttpGet("byid")]

[ActionName("GetById")]

[ProducesResponseType(StatusCodes.Status200OK)]

public string GetById()

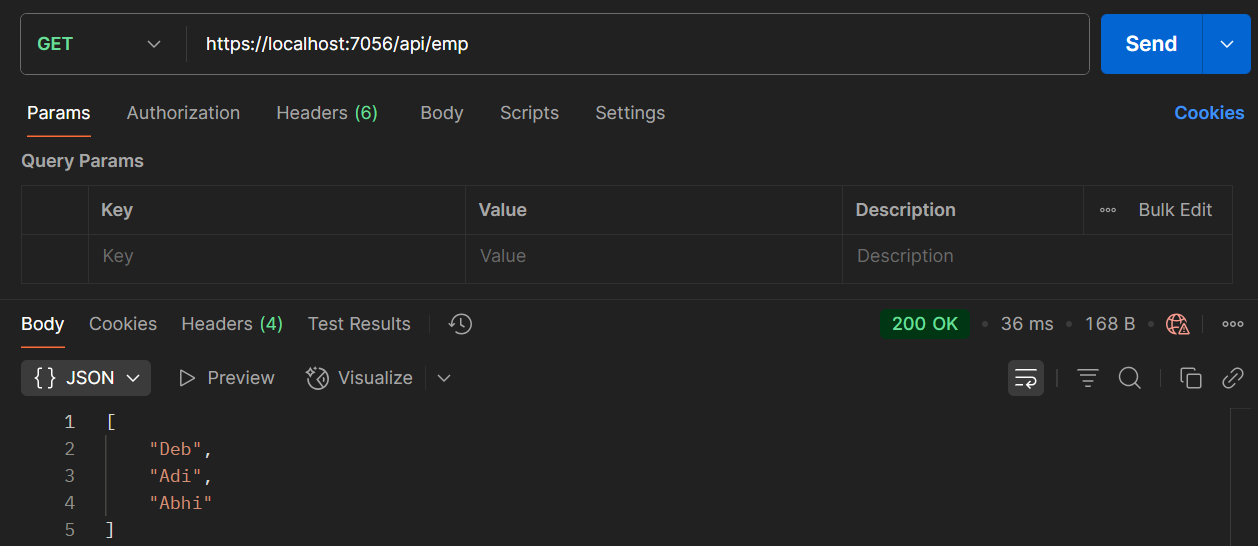
{

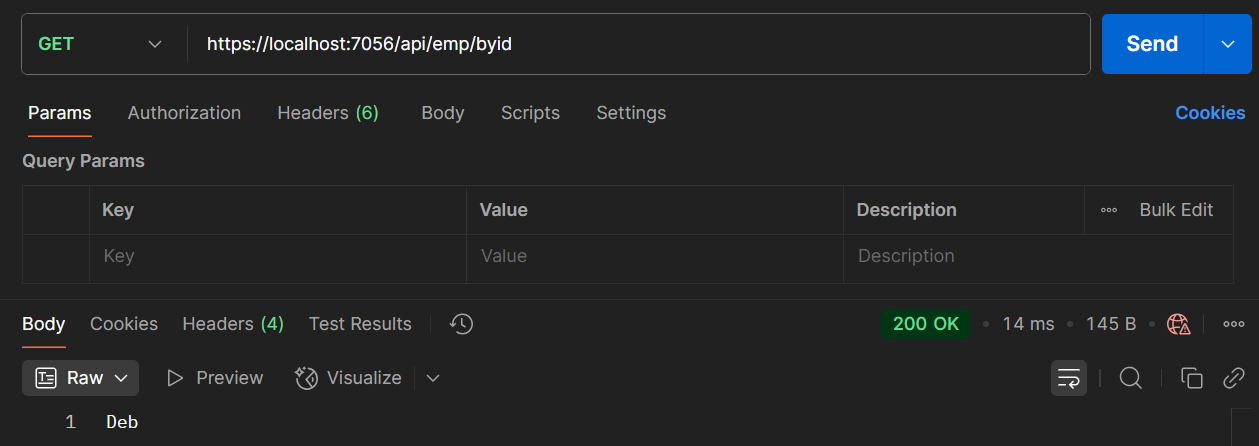
return "Deb";

}

}

}

**Output:**



**3: Web API using Custom Model, Auth Filter & Exception Filter**

Scenario:

Use a custom model in API, secure it with a filter, and handle errors with an exception filter.

To achieve this:

1. Create an Employee model and controller returning sample data.

2. Add ProducesResponseType(200) and verify in Swagger.

3. Create CustomAuthFilter to check for Authorization header (Bearer) and apply to controller.

4. Implement CustomExceptionFilter to log errors and return a custom response.

5. Throw an exception in the GET method, mark with ProducesResponseType(500), and test via Swagger.

### **Code: (In Models)**

*Employee.cs*

using System;

using System.Collections.Generic;

namespace MyFirstWebApi.Models

{

public class Employee

{

public int Id { get; set; }

public required string Name { get; set; }

public int Salary { get; set; }

public bool Permanent { get; set; }

public required Department Department { get; set; }

public required List<Skill> Skills { get; set; }

public DateTime DateOfBirth { get; set; }

}

}

*Department.cs*

namespace MyFirstWebApi.Models

{

public class Department

{

public int Id { get; set; }

public required string Name { get; set; }

}

}

*Skill.cs*

namespace MyFirstWebApi.Models

{

public class Skill

{

public int Id { get; set; }

public required string Name { get; set; }

}

}

### **Code: (In Filters)**

*CustomAuthFilter.cs*

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace MyFirstWebApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasAuthHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);

if (!hasAuthHeader)

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

if (!token.ToString().Contains("Bearer"))

{

context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

}

}

}

}

*CustomExceptionFilter.cs*

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

using System.IO;

namespace MyFirstWebApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string message = $"Exception: {context.Exception.Message}";

var logPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "error\_log.txt");

File.AppendAllText(logPath, message + Environment.NewLine);

context.Result = new ObjectResult("An unexpected error occurred")

{

StatusCode = 500

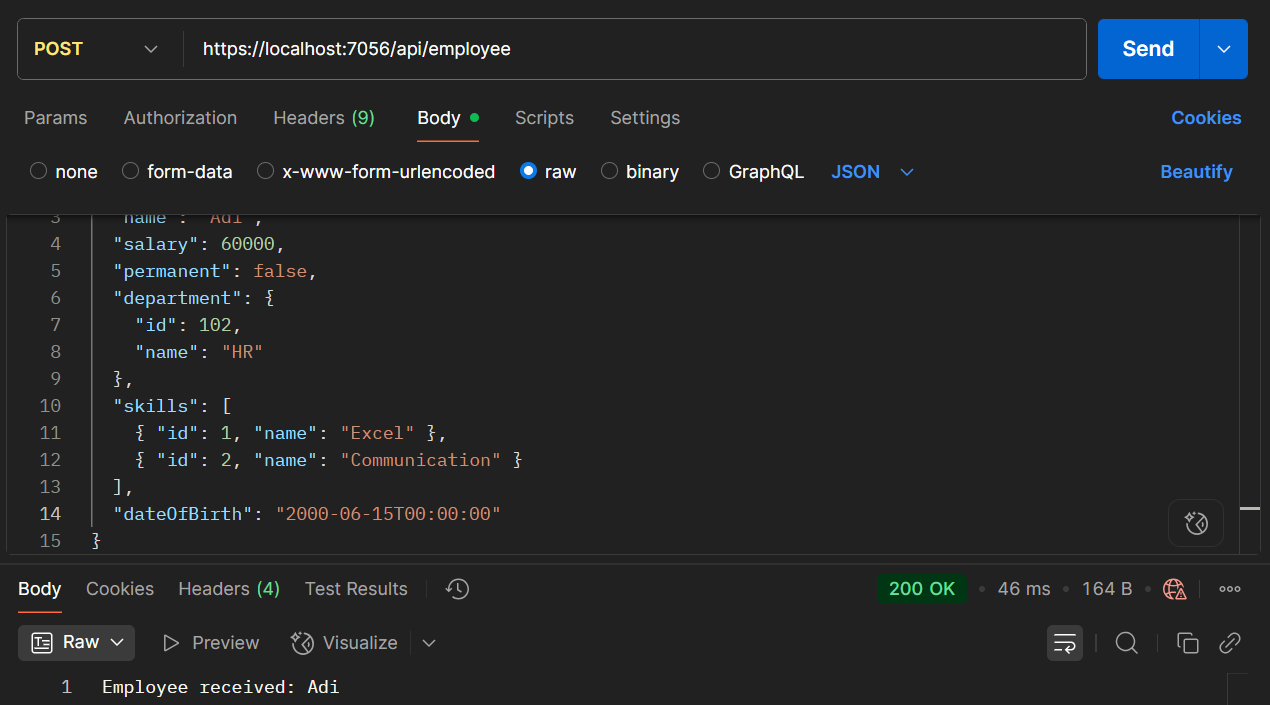
};

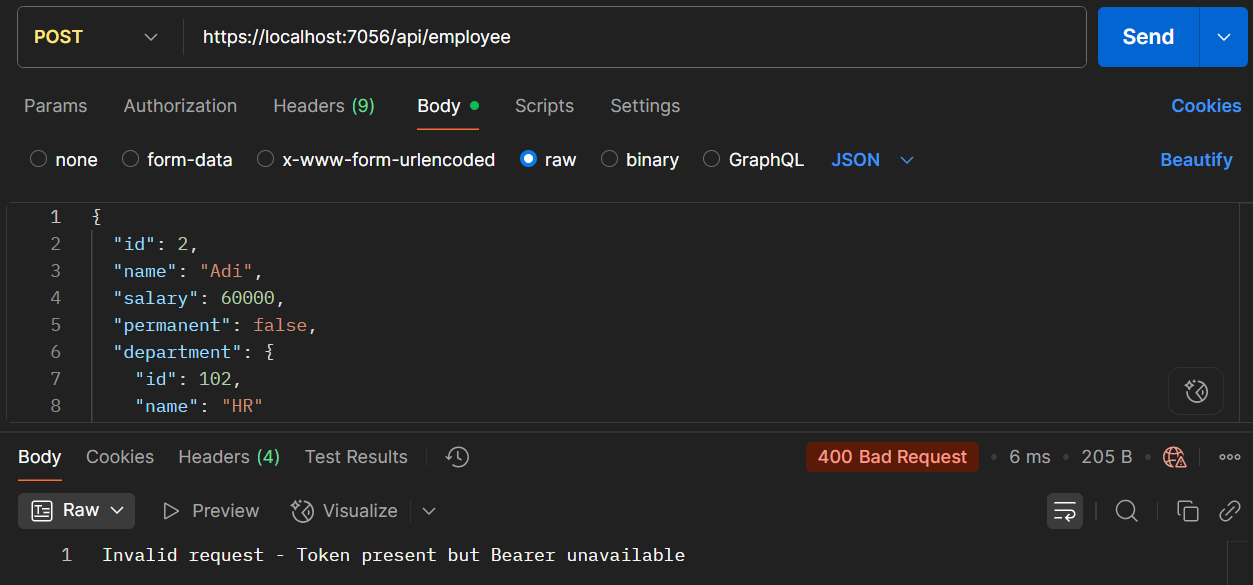
}

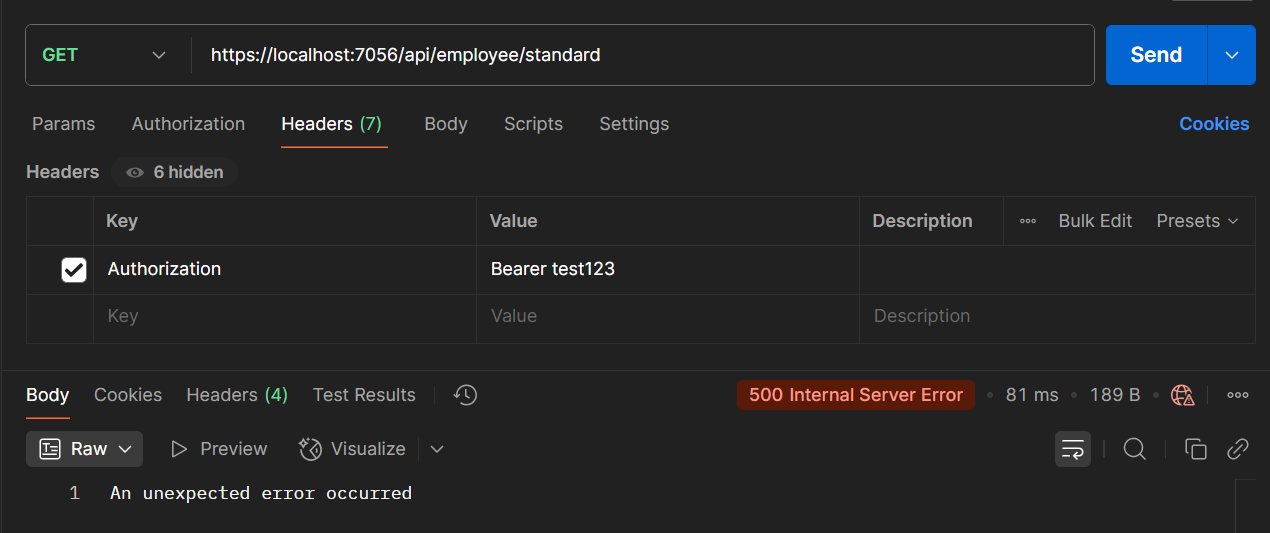
}

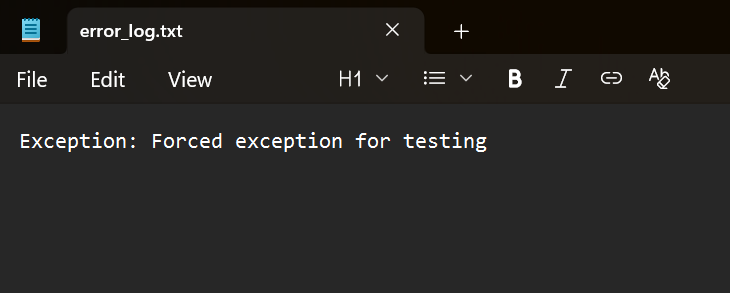
}

### **Output:**









**4: Web Api CRUD operation**

Scenario:

Implement an update operation for employee data using HTTP PUT in a .NET Core Web API.

To achieve this:

1. Use Swagger to call the PUT method of EmployeeController.

2. Accept JSON input to update employee details in a hardcoded list.

3. Return the updated employee using ActionResult<Employee>.

4. If id <= 0 or not found in the list, return BadRequest with "Invalid employee id".

5. If valid, update the record and return the modified employee in the response.

### **Code:**

*EmployeeController.cs*

using MyFirstWebApi.Filters;

using MyFirstWebApi.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebApi.Controllers

{

[ApiController]

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

[AllowAnonymous]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private List<Employee> employeeList;

public EmployeeController()

{

employeeList = new List<Employee>

{

new Employee

{

Id = 1,

Name = "Deb",

Salary = 60000,

Permanent = true,

Department = new Department { Id = 1, Name = "IT" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "ASP.NET Core" }

},

DateOfBirth = new DateTime(2002, 10, 17)

},

new Employee

{

Id = 2,

Name = "Adi",

Salary = 50000,

Permanent = false,

Department = new Department { Id = 2, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "Excel" },

new Skill { Id = 4, Name = "Communication" }

},

DateOfBirth = new DateTime(2001, 8, 5)

}

};

}

[HttpGet("standard")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

// throw new Exception("Forced exception for testing");

return Ok(employeeList);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

return Ok($"Employee received: {emp.Name}");

}

[HttpPut("{id}")]

public ActionResult<Employee> Put(int id, [FromBody] Employee updatedEmp)

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

var employee = employeeList.FirstOrDefault(e => e.Id == id);

if (employee == null)

{

return BadRequest("Invalid employee id");

}

employee.Name = updatedEmp.Name;

employee.Salary = updatedEmp.Salary;

employee.Permanent = updatedEmp.Permanent;

employee.Department = updatedEmp.Department;

employee.Skills = updatedEmp.Skills;

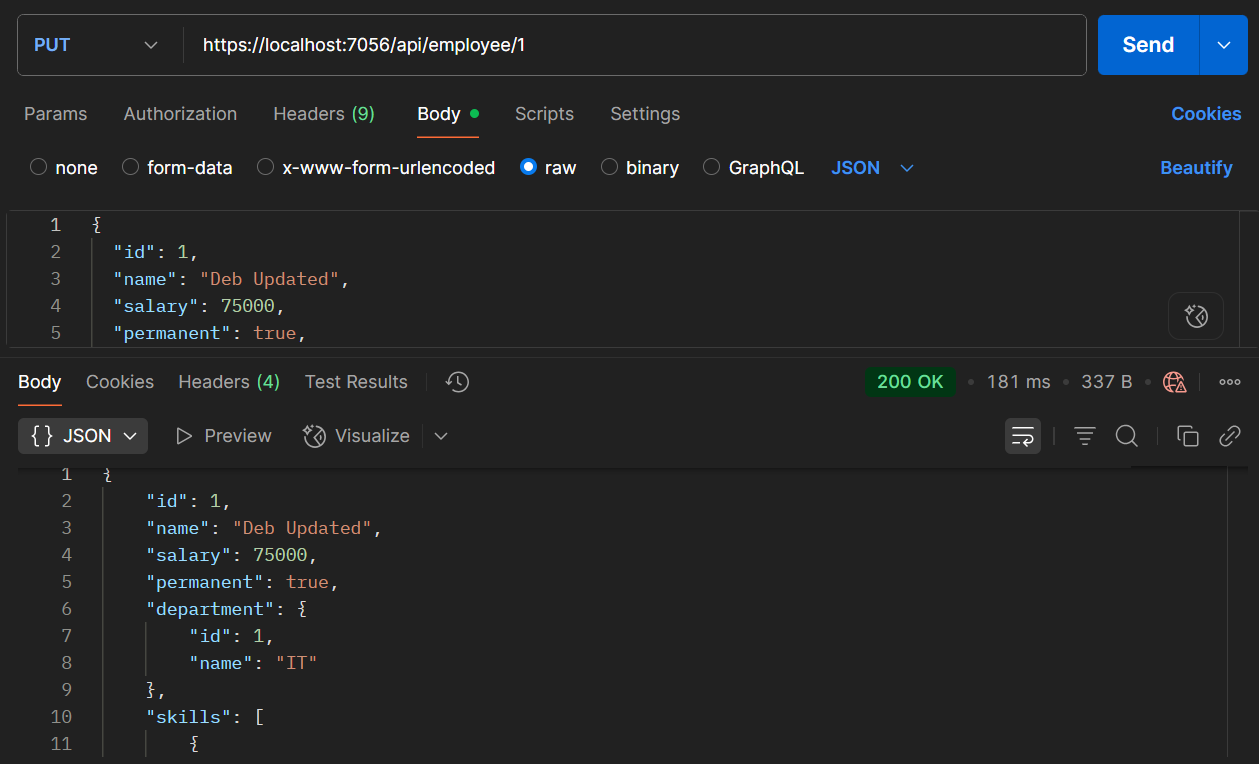
employee.DateOfBirth = updatedEmp.DateOfBirth;

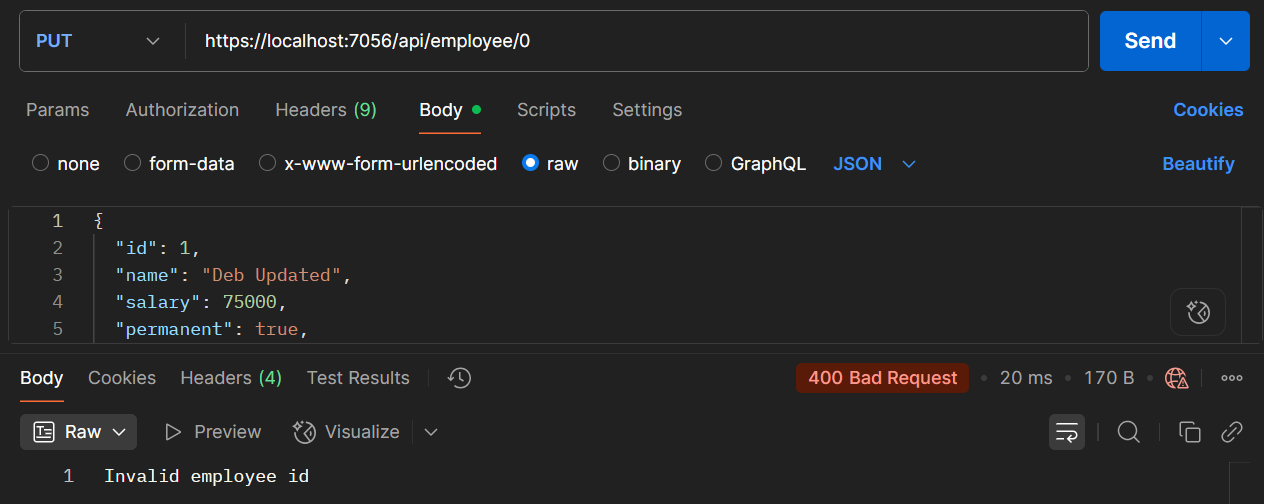
return Ok(employee);

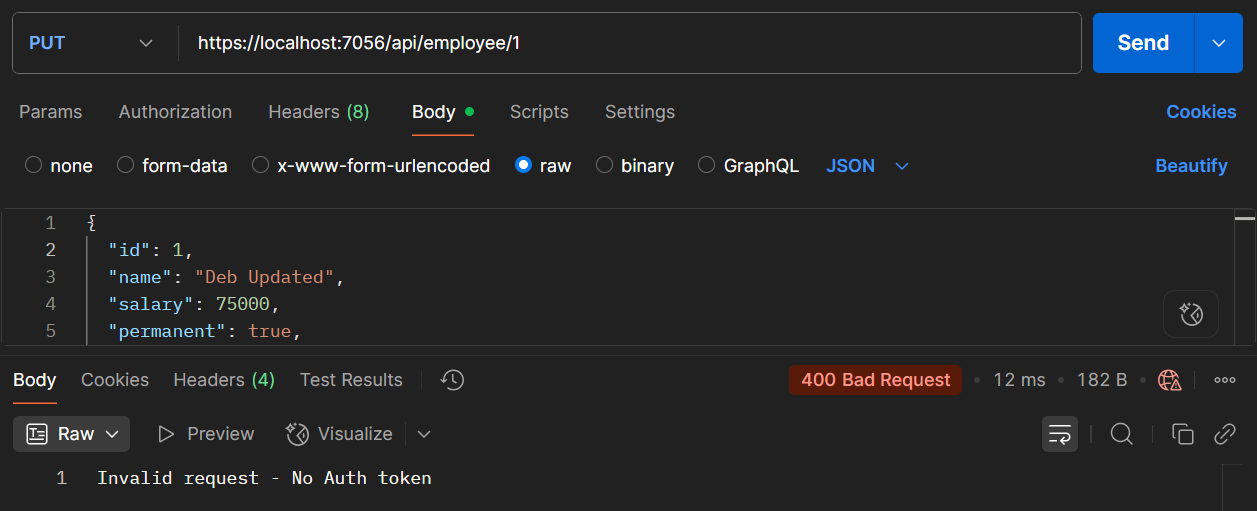
}

}

}

**Output:**





**5: JWT Authentication in .NET Core Web API**

Scenario:

Secure your Web API using JWT (Json Web Token) authentication and test it via Postman.

To achieve this:

1. Configure JWT authentication in Program.cs using a symmetric security key.

2. Create an AuthController with a method GenerateJSONWebToken(userId, userRole) to return a token.

3. Use AllowAnonymous for AuthController and Authorize on EmployeeController.

4. Generate a JWT and use it in Postman’s Authorization header to access secured endpoints.

5. Test for 401 Unauthorized by:

* Omitting the token
* Using an invalid or expired token
* Setting expiry to 2 minutes and retrying after delay
* Using mismatched roles (e.g., POC only vs Admin,POC) in the Authorize attribute.

**Code:**

*Program.cs*

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using MyFirstWebApi.Filters;

using System.Text;

using System.IdentityModel.Tokens.Jwt;

var builder = WebApplication.CreateBuilder(args);

string securityKey = "mysuperdupersecrettokenkey1234567890";

var symmetricKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));

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 = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricKey,

};

options.Events = new JwtBearerEvents

{

OnAuthenticationFailed = context =>

{

context.Response.StatusCode = 401;

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

if (context.Exception.GetType() == typeof(SecurityTokenExpiredException))

{

return context.Response.WriteAsync("{\"error\": \"Token expired\"}");

}

return context.Response.WriteAsync("{\"error\": \"Authentication failed\"}");

}

};

});

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new OpenApiContact

{

Name = "Debanjan",

Email = "Deb@xyzmail.com",

Url = new Uri("https://www.example.com")

},

License = new OpenApiLicense

{

Name = "License Terms",

Url = new Uri("https://www.example.com")

}

});

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer {your-token}'"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

});

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

*AuthController.cs*

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace MyFirstWebApi.Controllers

{

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

try

{

var token = GenerateJSONWebToken(101, "Admin");

return Ok(new { token });

}

catch (Exception ex)

{

return StatusCode(500, $"Token generation failed: {ex.Message}");

}

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecrettokenkey1234567890"));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, userRole),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(2),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

*EmployeeController.cs*

using MyFirstWebApi.Filters;

using MyFirstWebApi.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebApi.Controllers

{

[ApiController]

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

[Authorize(Roles = "Admin,POC")]

public class EmployeeController : ControllerBase

{

private List<Employee> employeeList;

public EmployeeController()

{

employeeList = new List<Employee>

{

new Employee

{

Id = 1,

Name = "Deb",

Salary = 60000,

Permanent = true,

Department = new Department { Id = 1, Name = "IT" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "ASP.NET Core" }

},

DateOfBirth = new DateTime(2002, 10, 17)

},

new Employee

{

Id = 2,

Name = "Adi",

Salary = 50000,

Permanent = false,

Department = new Department { Id = 2, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "Excel" },

new Skill { Id = 4, Name = "Communication" }

},

DateOfBirth = new DateTime(2001, 8, 5)

}

};

}

[HttpGet("standard")]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employeeList);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

return Ok($"Employee received: {emp.Name}");

}

[HttpPut("{id}")]

public ActionResult<Employee> Put(int id, [FromBody] Employee updatedEmp)

{

if (id <= 0)

return BadRequest("Invalid employee id");

var employee = employeeList.FirstOrDefault(e => e.Id == id);

if (employee == null)

return BadRequest("Invalid employee id");

employee.Name = updatedEmp.Name;

employee.Salary = updatedEmp.Salary;

employee.Permanent = updatedEmp.Permanent;

employee.Department = updatedEmp.Department;

employee.Skills = updatedEmp.Skills;

employee.DateOfBirth = updatedEmp.DateOfBirth;

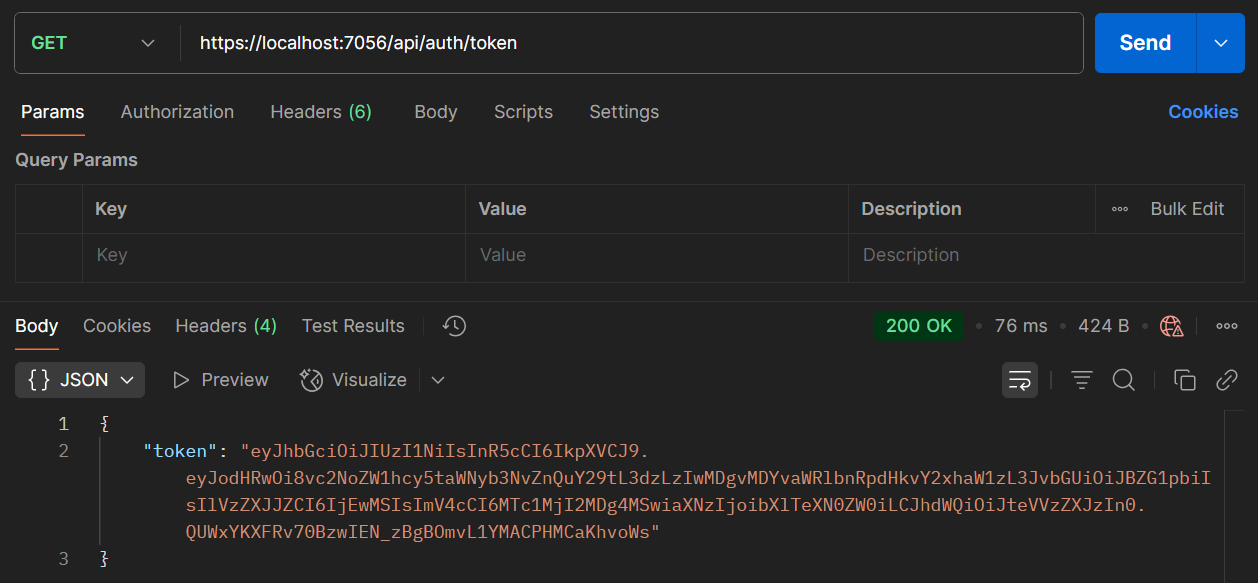
return Ok(employee);

}

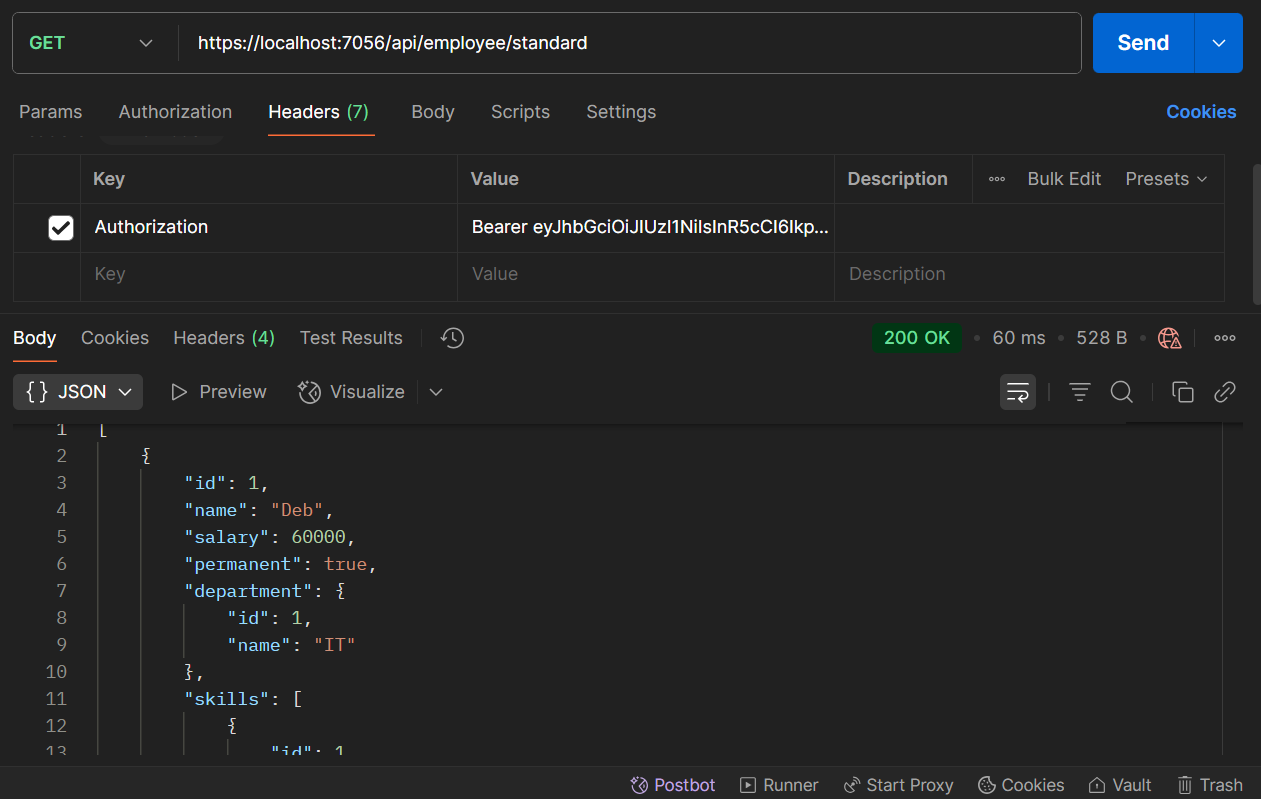
}

}

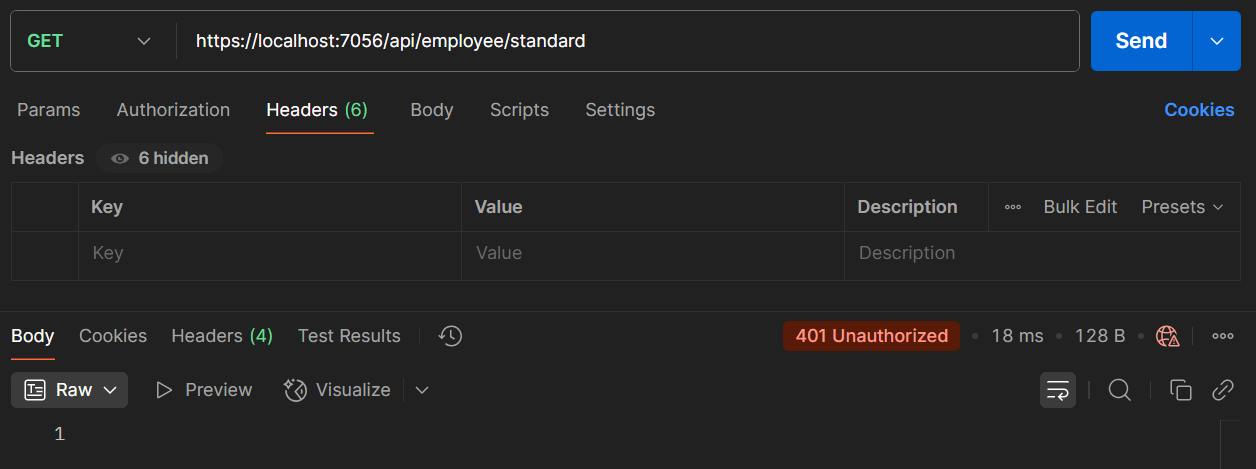
### **Output:**

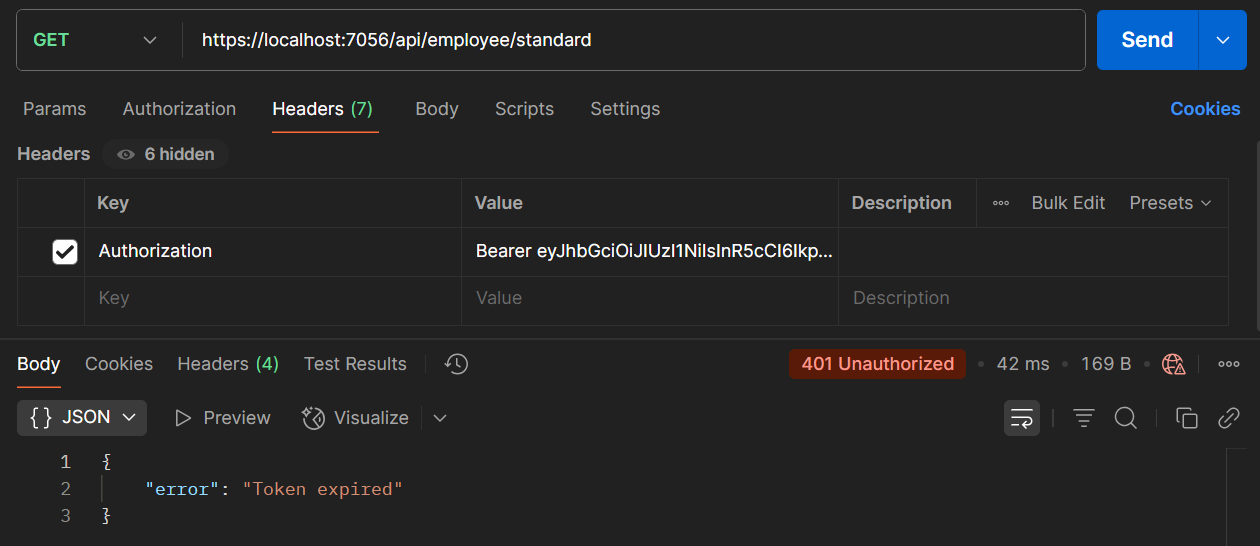
Get JWT Token

Use Token in Authorized Request



No Authorization Header





Expired Token

Wrong Role (optional)