Shop Bridge API

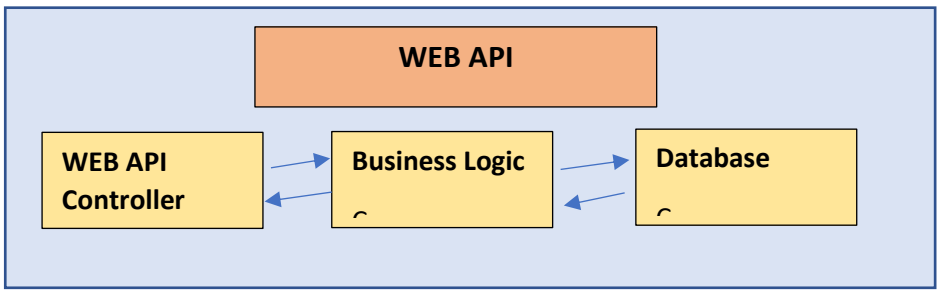
Database: SQL Server

Back end: .NET 6 Web API

ORM Frameworks used: Entity Framework core, Dapper

Required Software: SQL Server 19, Visual Studio 2022

Architecture Pattern followed:



Implementation Details:

1. Created the entity models and the dbContext class for ShopBridge API
2. Generated tables using Code First Approach in Entity Framework Core
3. Used migration to seed initial data for product and category table
4. Implemented Create, Update and Delete operations using Entity Framework core ORM.
5. Used Dapper for the GET operations as it is faster than Entity Framework Core and effective to implement different filter operations.
6. Used OAuth to implement Role-based access control for the endpoints.

Dependencies:

1. Microsoft.EntityFrameworkCore. -Version 7.0.2
2. Microsoft.EntityFrameworkCore.Tools -Version 7.0.2
3. Microsoft.EntityFrameworkCore.SqlServer – Version 7.0.2
4. Dapper - version (2.0.123)

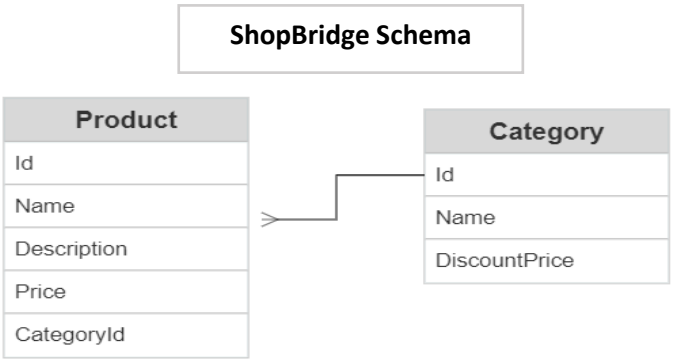
Other Nuget Packages used:

1. Fluent Validation (v11.4.0) – for validation the add and update request data
2. AutoMapper (v12.0.1) – to map between the entity model and the DTOs.
3. Newtonsoft.json (v13.0.2) – to handle json serialization operations

Database: ShopBridgeDatabase

Tables created: Product, Category

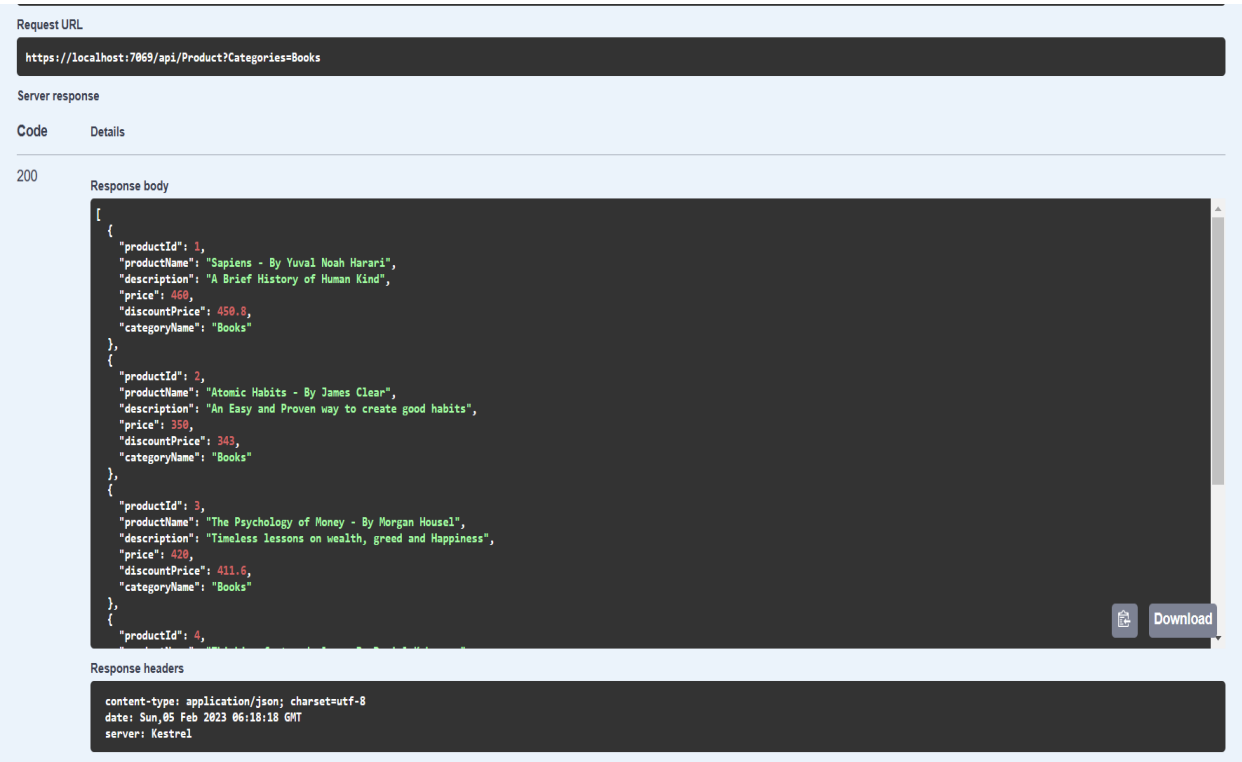
ER Diagram:



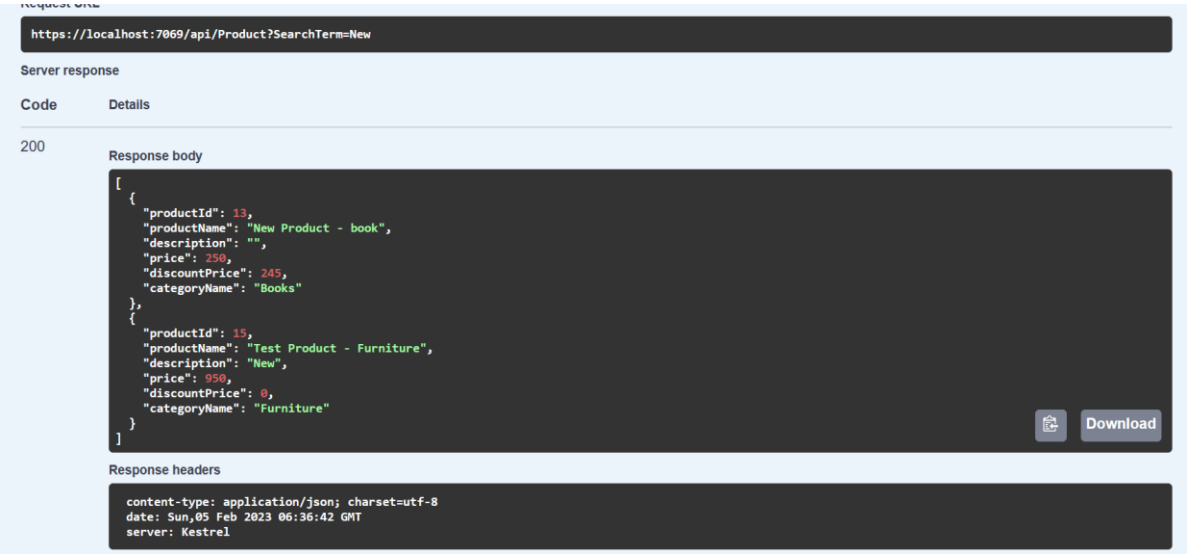
API Endpoints:

Get Products

1. Filter by category



2. Common Search



https://localhost:7069/api/Product?SearchTerm=New

Server response

Code | Details

200

Response body

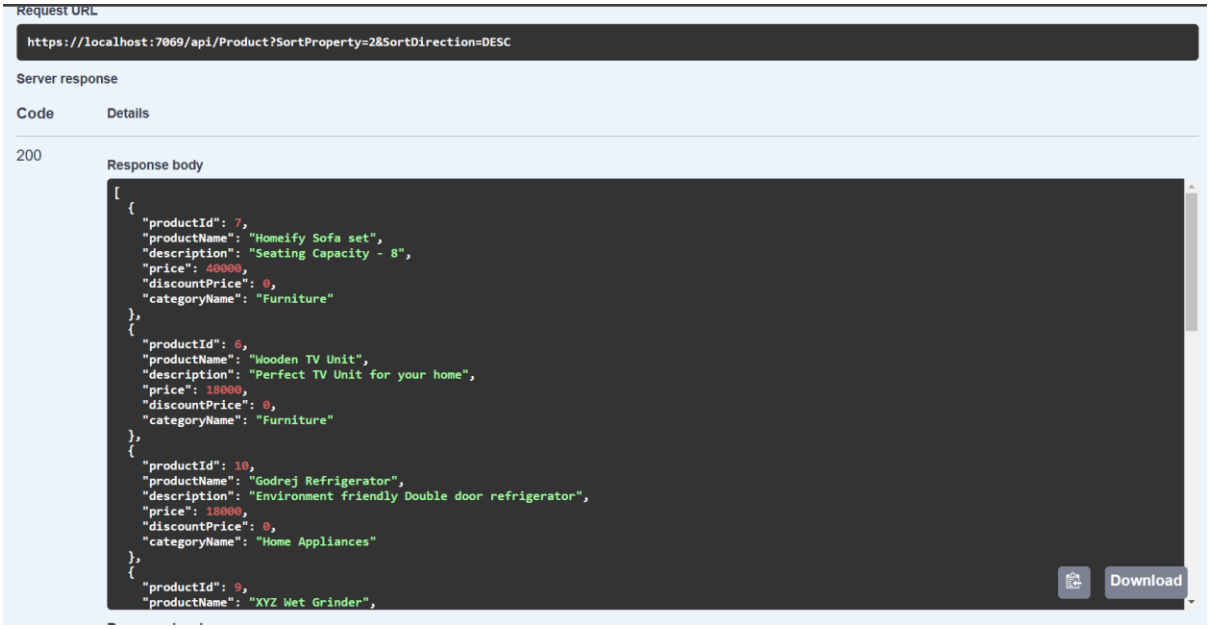

```
[
  {
    "productId": 13,
    "productName": "New Product - book",
    "description": "",
    "price": 250,
    "discountPrice": 245,
    "categoryName": "Books"
  },
  {
    "productId": 15,
    "productName": "Test Product - Furniture",
    "description": "New",
    "price": 950,
    "discountPrice": 0,
    "categoryName": "Furniture"
  }
]
```


Download

Response headers

```
content-type: application/json; charset=utf-8
date: Sun,05 Feb 2023 06:36:42 GMT
server: Kestrel
```

3. Sort by Price in Descending order



Request URL

https://localhost:7069/api/Product?SortProperty=2&SortDirection=DESC

Server response

Code | Details

200

Response body


```
[
  {
    "productId": 7,
    "productName": "Homeify Sofa set",
    "description": "Seating Capacity - 8",
    "price": 40000,
    "discountPrice": 0,
    "categoryName": "Furniture"
  },
  {
    "productId": 6,
    "productName": "Wooden TV Unit",
    "description": "Perfect TV Unit for your home",
    "price": 18000,
    "discountPrice": 0,
    "categoryName": "Furniture"
  },
  {
    "productId": 10,
    "productName": "Godrej Refrigerator",
    "description": "Environment friendly Double door refrigerator",
    "price": 18000,
    "discountPrice": 0,
    "categoryName": "Home Appliances"
  },
  {
    "productId": 9,
    "productName": "XYZ Wet Grinder",
```


Download

4. Simple Pagination

Curl

```
curl -X 'GET' \
  'https://localhost:7069/api/Product?PageSize=1&PageNumber=5' \
  -H 'accept: text/plain'
```

Request URL

```
https://localhost:7069/api/Product?PageSize=1&PageNumber=5
```

Server response

Code	Details
200	Response body

```
[
  {
    "productId": 5,
    "productName": "Wooden Wall Shelf",
    "description": "Decorative set of shelves",
    "price": 2000,
    "discountPrice": 0,
    "categoryName": "Furniture"
  }
]
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Sun,05 Feb 2023 06:39:09 GMT
server: Kestrel
```

Get By Id

Request URL

```
https://localhost:7069/api/Product/2
```

Server response

Code	Details
200	Response body

```
{
  "productId": 2,
  "productName": "Atomic Habits - By James Clear",
  "description": "An Easy and Proven way to create good habits",
  "price": 350,
  "discountPrice": 343,
  "categoryName": "Books"
}
```

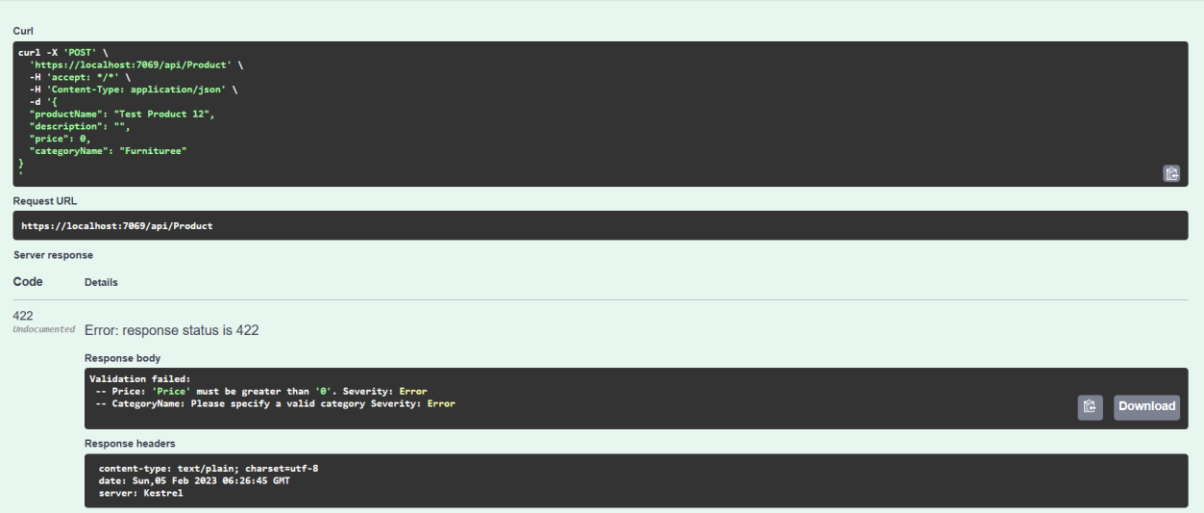
Download

Response headers

```
content-type: application/json; charset=utf-8
date: Sun,05 Feb 2023 06:20:32 GMT
server: Kestrel
```

Create Product

1. Given invalid request



2. Given valid request

Update Product

1. Invalid Put Request body

2. Invalid Product Id:

3. Valid Put Request

Curl

```
curl -X 'PUT' \
  'https://localhost:7069/api/Product/1001' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "productName": "Test Product 12",
  "description": "New Product",
  "price": 1200,
  "categoryName": "Furniture"
}'
```

Request URL

```
https://localhost:7069/api/Product/1001
```

Server response

Code	Details
204	Response headers

```
date: Sun,05 Feb 2023 06:33:44 GMT
server: Kestrel
```

Delete Product

1. Invalid Product Id

Curl

```
curl -X 'DELETE' \
  'https://localhost:7069/api/Product/10015' \
  -H 'accept: */*'
```

Request URL

```
https://localhost:7069/api/Product/10015
```

Server response

Code	Details
404	Error: response status is 404

Response body

```
Product not found for the given Id
```

Download

Response headers

2. Valid Product Id

Curl

```
curl -X 'DELETE' \
  'https://localhost:7069/api/Product/1001' \
  -H 'accept: */*'
```

Request URL

```
https://localhost:7069/api/Product/1001
```

Server response

Code	Details
204	Response headers

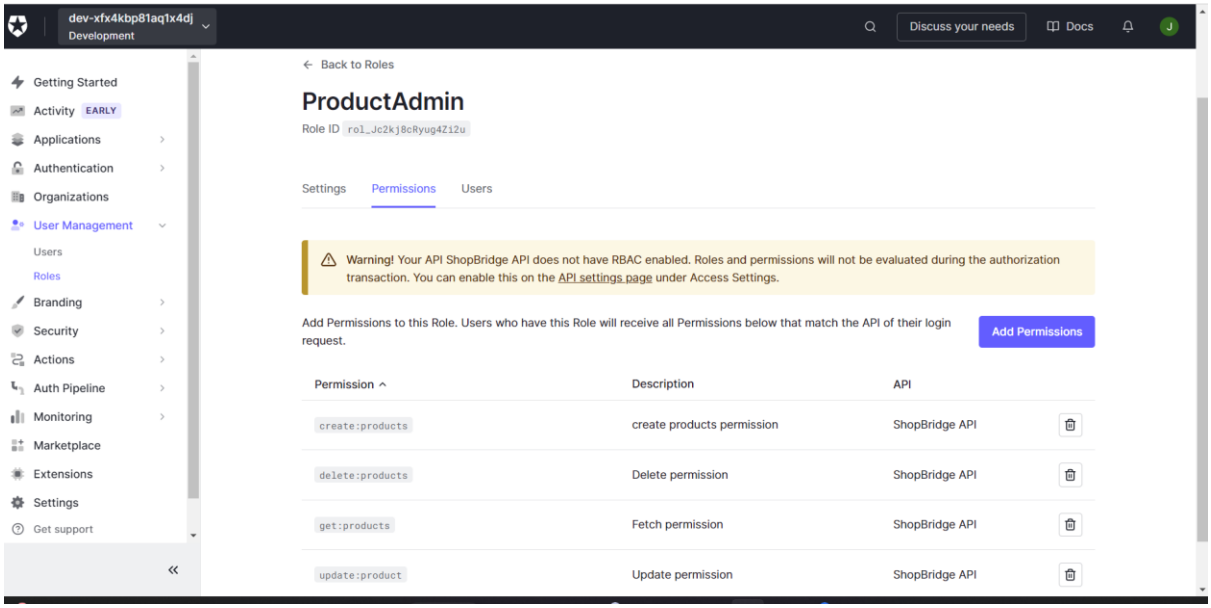
```
date: Sun,05 Feb 2023 06:35:11 GMT
server: Kestrel
```

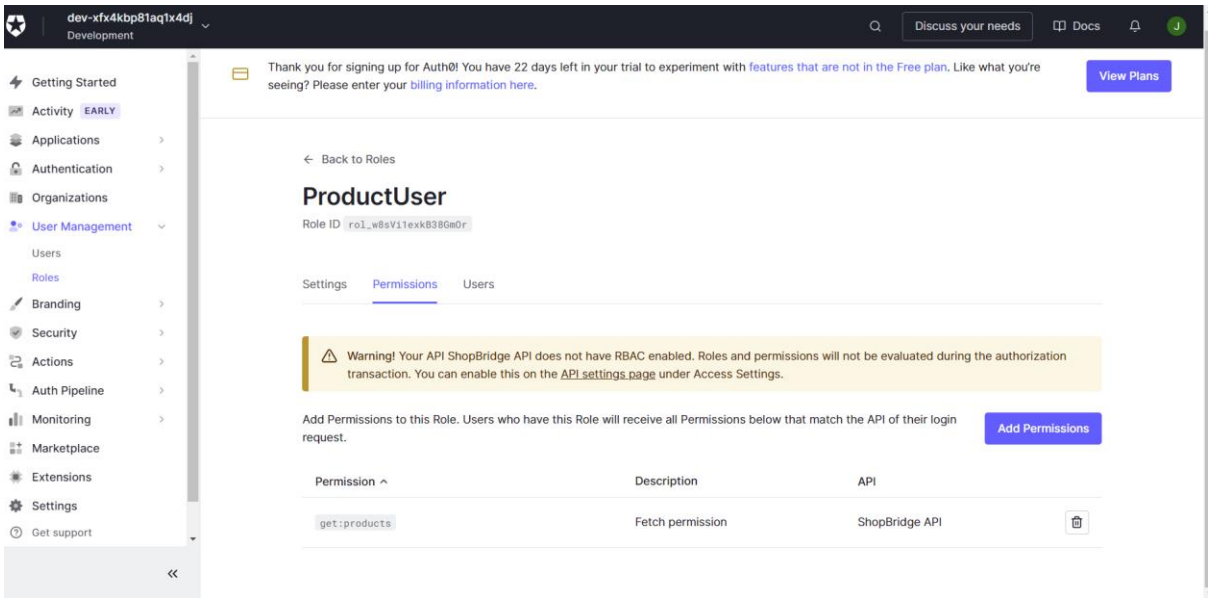
Implementation In Progress

Add role based authorization to allow only users with **ProductAdmin** Role to access the CREATE, UPDATE and DELETE endpoints.

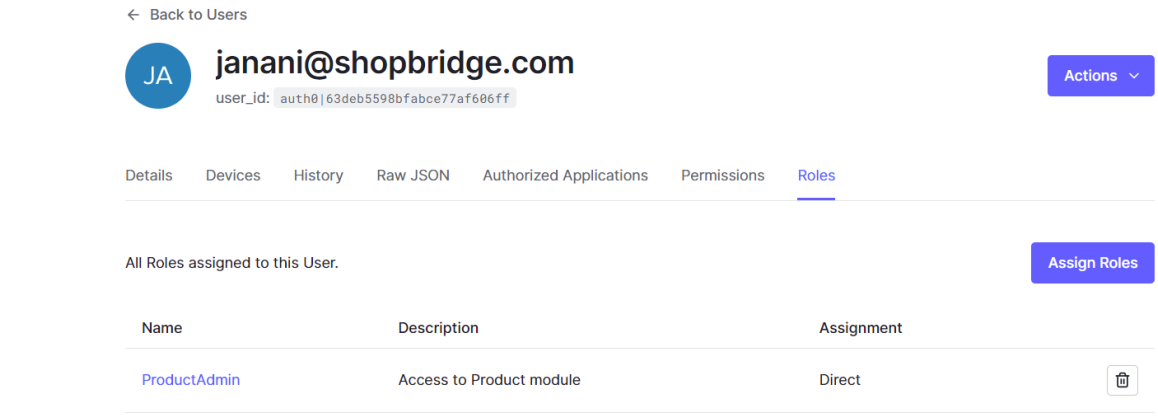
Users with **ProductUser** can only access the GET endpoints.

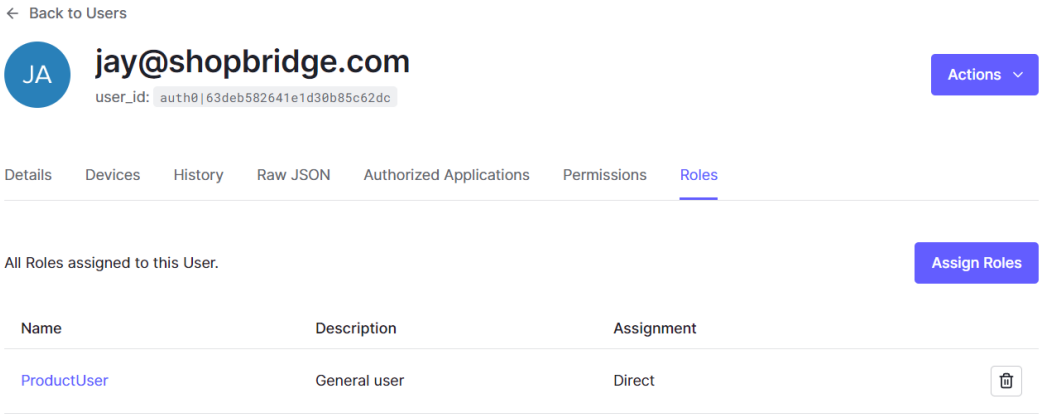
Roles Created in OAuth:



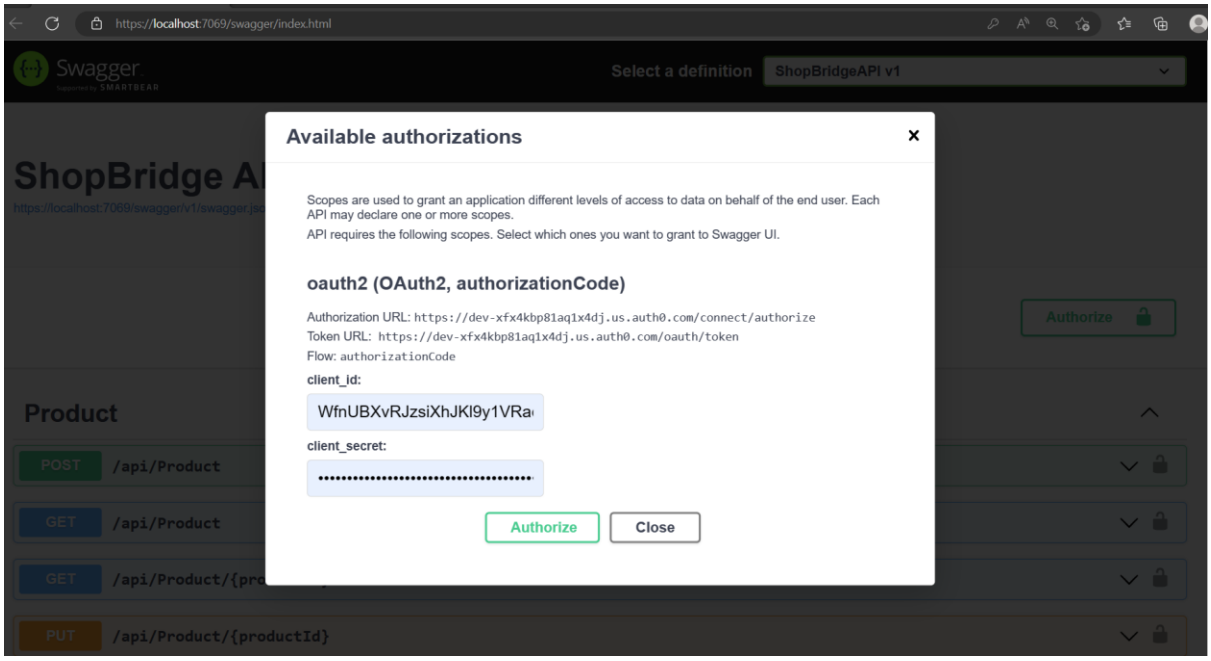


Created Users and Assigned them roles:





Incorporated OAuth2 Swagger Login Flow:



Configuring the OAuth Authorization in .net Application:

```
builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("CreateAccess", policy =>
                        policy.RequireClaim("permissions", "create:products"));
    options.AddPolicy("UpdateAccess", policy =>
                      policy.RequireClaim("permissions", "update:product"));
    options.AddPolicy("DeleteAccess", policy =>
                      policy.RequireClaim("permissions", "delete:products"));
    options.AddPolicy("GetAccess", policy =>
                      policy.RequireClaim("permissions", "get:products"));
});
```

```
// Fetch product by Id
[Authorize(Roles = "ProductUser")]
//[Authorize(Policy = "GetAccess")]
[HttpGet("{productId:int}")]
[ProducesResponseType(typeof(Product), StatusCodes.Status200OK)]
[ProducesResponseType(typeof(string), StatusCodes.Status404NotFound)]
[ProducesResponseType(typeof(string), StatusCodes.Status500InternalServerError)]
0 references
public async Task<ActionResult> GetAsync(int productId)
{
    try
    {
        if (productId == 0)
        {
            return BadRequest(ErrorMessages.InvalidProductId);
        }

        var result = await _getProductService.GetByIdAsync(productId);
        return Ok(result);
    }
    catch (EntityNotFoundException ex)
    {
        _logger.LogError(ex.Message);
        return NotFound(ex.Message);
    }
    catch (Exception ex)
    {
        return StatusCode(StatusCodes.Status500InternalServerError, ex.Message);
    }
}
```

I am facing some issues with validating the JWT token generated from the login flow in the ShopBridge API. I may need to spend some more time on this to make it work.