Shopbridge

25.10.2021

Akhil Shukla
akhilshukla722@gmail.com
9653002960

Overview

ShopBridge is an e-commerce application. It is currently in development mode. It consists of a vast variety of products. It is built by exploiting Dot Net Core features which makes it consistent, scalable and efficient.

Goals

1. Create a set of API's for the product admin to manage the inventory.
2. Product Admin will be able to add, delete, modify and list all the products.

Technical Specifications

Technology In use - Entity Framework core, Asp Dot Net Core Web Api.

DataBase - Ms Sql server

Idea

The project is divided into 3 layers namely Data Access layer, Business Logic Layer and the Service Layer hence achieving the layered architecture. The Service layer or API layer will be wrapped around the Business Logic Layer to provide the functionalities. Finally the endpoints are exposed and those can be invoked in order to manage the inventory.

Milestones

I. DataBase Creation

Following the DB first approach. First I have created the database. The schema of the database is below and the database script is present under the Git repository.

Name of DataBase - ShopBridgeDB

Database Specifications and Schema

1) Product Schema

ProductId	Char(4), starts with "P101" Constraint - Primary Key
ProductName	Varchar(50) Constraint - Unique

CategoryId	TinyInt Constraint - Foreign Key → Categories (CategoryId)
Price	Float
QuantityAvailable	Int Constraint - Check (QuantityAvailable >= 0)

2) Categories

CategoryId	TinyInt Constraint - Primary Key
CategoryName	Varchar(20) Constraint - Unique

3) ProductImages

Sno	Int Constraint - Primary Key
ProductId	Char(4) Constraint - Foreign Key ---> Products(ProductId)
ImageLink	Varchar(2000)

4) Stored Procedures

usp_AddCategory	This stored procedure is invoked for adding categories.

usp_AddProduct	Invoked for adding products in the inventory.
usp_GetProductsOnCategoryId	Getting all the products based on the categoryId provided.

5) Functions -

ufn_GenerateNewProductId	Generating a new product Id Ex: P101, P102 Note: Product Id will always start with P and will be in the series of P101 and so on.
ufn_GetAllProductDetails	Lists down all the products.
ufn_GetCategories	Lists down all the available categories.
ufn_GenerateNewCategoryId	Generated a new category Id. Ex: 1,2 and so on.

II. DataStore and Business Logic Layer

EF core and database first approach is used to trigger the stored procedures or functions present in our database. Each and every function created in this layer is asynchronous. The Business Layer will invoke the functions available in the Data Access layer and after adding some logic it will pass the result to the Service Layer.

III. Service Layer

Asp Dot Net Core Web Api is used to create the Service Layer. The Service Layer consists of filters to perform Model Validation, Model validation are also performed through custom data annotations.

Finally the result is mapped to the DTO and sent as a response.

The exposed endpoints and their specifications -

1) AddProduct -

Type	POST
Functionality	Add a product based on the data provided.
Parameter Type	Json object from body

Example of object
 	
The product Id and the modified fields
	{
	“ProductName” : “Samsung M34”, //[Required]
	“CategoryId”: 5, //[Required]
	“Price” : 15000, //[Required]
	“QuantityAvailable”: 5 //[Required]
	}

Returns	For successful addition, returns the product object in Json format with product Id status Code.
	Example -
	{
	“ProductId”: “P158”, //[Required]
	“Status”: 1 //[Required]
	“ProductName” : “Samsung M34”, //[Required]
	“CategoryId”: 5, //[Required]
	“Price” : 15000, //[Required]
	“QuantityAvailable”: 5 //[Required]
	}
	All other status Codes - 1,-2,-3,-4,-5,-6, -7, -99. See Status Code for more details.

Url	http://localhost:46447/api/product/Modify Product

2) ModifyProduct -

Type	POST
Functionality	Modify a product based on its product Id and the data to be modified.
Parameter Type	Json object From Body

Example of object
 	
The product Id and the modified fields
	{
	“ProductId” : “P158” //mandatory
	“ProductName” : “Samsung M34”,
	“CategoryId”: 5,
	“Price” : 15000,
	“QuantityAvailable”: 5
	}

Url	http://localhost:46447/api/product/ModifyProduct

3) - DeleteProduct

Request Type	Delete
Functionality	Deletes a product from the repository.
Parameter	productId
Parameter type	Query String
Example URL	http://localhost:46447/api/DeleteProduct?productId=P157

4) Lists All products -

Request Type	Get
Functionality	Lists all the products in the inventory
Parameter	Not required
Example URL	http://localhost:46447/api/GetAllProducts

5) Lists All categories -

Request Type	Get
Functionality	Lists all the categories in the inventory
Parameter	Not required
Example URL	http://localhost:46447/api/GetAllCategories

Instructions -

1) Install Visual Studio
2) Open the solution.
3) Build and resolve the errors.
4) Download Microsoft SSMS
5) Run the provided scripts
6) Postman can be used to test the API