Product & Cart Management API

This project is an **ASP.NET Core Web API** built using **Clean Architecture**, **MediatR (CQRS)**, **Entity Framework Core**, and **MSTest**. It manages **Products, Clients, Carts**, and related business rules.

Tech Stack

- .NET (ASP.NET Core Web API)
- Entity Framework Core
- MediatR (CQRS Pattern)
- SQL Server
- Swagger (OpenAPI)
- MSTest & Moq

Solution Structure

```
Solution
│
├── API
│   ├── Controllers
│   ├── DependencyInjection
│   ├── Program.cs
│   └── appsettings.json
│
├── Application
│   ├── Commands
│   ├── Queries
│   ├── Handlers
│   └── DependencyInjection
│
├── Domain
│   ├── Entities
│   ├── Enums
│   └── Interfaces
│
├── Infrastructure
│   ├── Repositories
│   └── AppDBContext
│
├── Database
│   └── SeedData.sql
```

```
│
└── API.Tests
    ├── Handlers
    └── Domain
```

Prerequisites

- .NET SDK (matching the project target framework)
- SQL Server / SQL Server Express
- Visual Studio 2022 (recommended)

Configuration

Update `` ` in the `` API` project:

```
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=.;Database=ProductDB;Trusted_Connection=True;TrustServerCertificate=True;"
  }
}
```

Database Setup & Seeding

> ⚠️**Mandatory:** The application requires initial seeded data.

1️⃣Create / Update Database

```
dotnet ef database update
```

2️⃣Seed Data from SQL File

The SQL seed file is located at:

```
/Database/SeedData.sql
```

Execute using:

SQL Server Management Studio (Recommended)

1. Open SSMS and connect to your server
2. Select the database
3. Open `SeedData.sql`
4. Click **Execute**

Command Line

```
sqlcmd -S <ServerName> -d <DatabaseName> -i SeedData.sql
```

Example:

```
sqlcmd -S . -d ProductDB -i SeedData.sql
```

3️⃣ Verify Seeded Data

```
SELECT * FROM Products;
SELECT * FROM Clients;
SELECT * FROM Carts;
```

Run the Application

```
dotnet run --project API
```

Swagger UI:

```
https://localhost:<port>/swagger
```

Available Endpoints

Method	Endpoint	Description
POST	/api/products	Add product
PUT	/api/products/{id}	Update product
GET	/api/products	Get all products

Method	Endpoint	Description
GET	/api/products/{id}	Get product by id
DELETE	/api/products/{id}	Delete product

Unit Tests

Run all tests:

```
dotnet test
```

Testing Strategy:

- Controllers → Mock `ISender`
- Handlers → Mock repositories
- Domain → Pure unit tests
- Data-driven tests with `[DataTestMethod]` & `[DataRow]`

Architectural Patterns

- Clean Architecture
- CQRS with MediatR
- Repository Pattern
- Dependency Injection
- Domain-driven validation

Notes

- Do not run without seeding data
- Seed script must run once per environment
- Re-run only after database reset

Author

Ramakrishnan Ramar
Senior .NET Developer / IT Architect