**Dukkantek Assignment**

**Problem Statement:**

We have an inventory with :

● products(name, barcode, description, weight, status(sold, inStock, damaged))

● categories (name)

Using DDD, SQL server, EF Core for the database provider, .Net core, no UI needed just

the APIs

● Create 3 APIs

1) Count the number of products sold, damaged and inStock

2) Change the status of a product

3) Sell a product

Solution:

1. Created a Web API targeting .NET 5.0 Framework, Entity Framework core and using SQL Server Database.
2. Clean code with Domain Driven Development – Entities created for each object.
3. Generic Repository pattern created for each object – targeting each Database table and performing CRUD operations.
4. Added dependency injection.
5. Error handling and null checks.
6. SOLID principals are kept in mind when building this project.
7. Two projects have been created – one is database layer and other is WEB API layer to encourage loose coupling and make sure each layer is independent.
8. Swagger documentation is implemented.
9. Database scripts are also included

Steps to run project :

1. Clone the repo in your local Visual Studio : <https://github.com/Solid-Programmer/WebAPI-EFCore-Assignment.git>
2. Click Build and make sure “WebAPILayer” project is set as Start Up project.
3. Validate connection string provided in appSettings.json file : "InventoryDB": "Data Source=localhost;Initial Catalog=Inventory; Integrated Security=true;"
4. Run the project by clicking on : IIS Express
5. Swagger UI will open which will help in testing the API End points

Graphical user interface, application, email

Description automatically generated

1. Endpoint : GetProductByStatus
   1. Returns the count of products under each Status category.
   2. If passed a status for e.g., “Damaged”, it will return the no of products Damaged.
2. Endpoint : ChangeProductStatus
   1. Changes the status of already existing product
3. Endpoint : SellProduct
   1. It takes product ID as input and marks the Status as “Sold”
   2. In case of invalid ID, it returns Bad Request with a message.

I have used GET, PUT and POST just to demonstrate all types of WEB API request:

1. Simple GET
2. POST – using body parameters
3. PUT to update a specific value

Things which are not implemented.

1. Logging
2. Exception handling
3. Integration testing
4. No business layer present but it can be added based on requirement.