**Inventory API Documentation**

**Highlights:**

* Layering Design Pattern
* Dependency Injection
* XUnit Test Cases
* Asynchronous API Call
* Exception handling
* Data Validations handled
* Logging enabled - SeriLog
* Coding Standards followed
* Versioning
* Entity Framework
* Used Migrations to create database - Please use "add-migration initial" and "update-database" in package management console for localdb
* Asynchronous Methods
* Proper user messages have provided with status code in response
* Custom Action filters

**EndPoints**

**GET:** **/v1/api/Inventory**

Getting all items in the Inventory

**POST:** **/v1/api/Inventory**

Add an item in the inventory by passing Item object

**Sample Request:**

{

“name": "Shampoo",

"description": "Pure Natural 100% organic",

"inStock": true,

"price": 5.00

}

**Validations:**

* + Name and Description should not be impty
  + ItemId is not allowed
  + Price should be greater than 0.00
  + Same item is not allowed to add again based on Name

**PUT:** **v1/api/Inventory/delete**

Delete the existing item from Inventory

**Sample Request:**

{

"itemId":12,

"name": "ShampooWhite",

"description": "Pure Natural 100% organic",

"inStock": true,

"price": 5.00

` }

Only existing Id will delete else thrown message to user (Item Id is mandatory)

**PUT (Update):** **v1/api/Inventory**

Update the existing item in the inventory (Item Id is mandatory)

**Sample Request:**

{

"itemId":12,

"name": "ShampooWhite",

"description": "Pure Natural 100% organic",

"inStock": true,

"price": 5.00

` }

1. Data store design - 1 hour (Tried migrations)

2. API and service logic – 1 hour

3. Unit Test Coverage – 30 mins