Project Description: Store Inventory Management

This project is a RESTful API built with NestJS and TypeScript for managing a store's inventory. It includes endpoints for creating, updating, deleting, and retrieving products. The application loads product data from a JSON file (`products.json`) and stores it in memory

Endpoints

The API exposes the following endpoints:

1. Create a New Product
    - Endpoint: POST /products
    - Description: Adds a new product to the inventory. The product name must be unique.
    - Body Parameters (json): { "name": "string", "description": "string", "price": "number", "quantity": "number"}
    - Request example: POST http://localhost:3000/products
    - Params example: { "name": "New Product11111", "description": "Description of the new product", "price": 100,"quantity": 20}
2. Update an Existing Product
    - Endpoint: PUT /products/:id
    - Description: Updates the details of an existing product. The product name must remain unique.
    - Path Parameters:
        - id (string): The ID of the product to update.
    - Body Parameters (json): { "name": "string", "description": "string", "price": "number", "quantity": "number"}
    - Request example: PUT http://localhost:3000/products/87fbbd02-ef24-4e25-a9c1-e8b45cd3450a
    - Params example: { "name": "New Product2222", "description": "Description of the new product", "price": 100,"quantity": 20}
3. Delete an Existing Product
    - Endpoint: DELETE /products/:id

- Description: Deletes a product from the inventory. Products with pending orders cannot be deleted.
- Path Parameters:
  - id (string): The ID of the product to delete.
- Request example: DELETE http://localhost:3000/products/87fbbd02-ef24-4e25-a9c1-e8b45cd3450a

4. Retrieve All Products
   - Endpoint: GET /products
   - Description: Retrieves a list of all products with options for sorting, pagination, and querying by name and description.
   - Query Parameters:
     - sortBy (string): Field to sort by (e.g., name, price, quantity).
     - page (number): Page number for pagination.
     - limit (number): Number of items per page (pagination takes place when both 'page' and 'limit' fields provided)
     - name (string): Query by product name.
     - description (string): Query by product description.
   - Request example: GET http://localhost:3000/products?limit=2&page=2
5. Retrieve low-stock products
   - Endpoint: GET /products/low-stock
   - Description: Retrieves a list of all products with options for sorting, pagination, and querying by name and description.
   - Query Parameters:
     - threshold(number): the highest 'quantity' to show products with
     - page (number): Page number for pagination.
     - limit (number): Number of items per page (pagination takes place when both 'page' and 'limit' fields provided)
   - Request example: GET http://localhost:3000/products/low-stock?limit=2&page=2
6. Retrieve most popular top N products
   - Endpoint: GET /products/low-stock
   - Description: Retrieves a list of all products with options for sorting, pagination, and querying by name and description.
   - Query Parameters:
     - top(number): 'top' number of products with highest 'sold' field is returned
     - page (number): Page number for pagination.
     - limit (number): Number of items per page (pagination takes place when both 'page' and 'limit' fields provided)
   - Request example: GET http://localhost:3000/products/most-popular?top=4

Pagination and caching implemented for all GET requests.Cache items default ttl is 60 sec. Generated request id is attached to every log entry (there is logger middleware that logs high-level request info into ‘request.log’ file in project’s root directory and log of product-service with output to console). Request ids are attached to response and all logs so we can easily correlate between log and response data during logs investigation.

How to run the project

Clone the project:

```
git clone https://github.com/mkopylenko/inabit_project.git
cd inabit_project
```

Install dependencies:

```
npm install
```

Start the application (it’ll be attached to port 3000 - see endpoint examples above):

```
npm run start
```

A few tests are implemented for product.service, they can be run with:

```
npm test products.service.spec.ts
```