Take Home Exercise (Backend Engineer)

Note: Please do not use AI coding agents to complete this assignment. You may refer to the internet for specific help or documentation.

Python is the preferred language for implementation.

Key Definitions

Marketplace

A *marketplace* is an online platform where sellers list and sell their products.

Examples: **Myntra, Flipkart, Amazon**.

Each marketplace has its own **product listing template** — a set of rules and attributes (like *product name*, *brand*, *price*, *size*, *color*) that every product must follow to be listed.

Think of it like a form you have to fill in:

- Flipkart might ask for "Title", "MRP", "Listing Price", "Image 1".
- Myntra might ask for "Product Name", "Gender", "Category", "Selling Price".

Seller File (CSV/Excel)

A *seller file* is the catalog provided by the merchant. It contains product attributes such as:

- SKU (unique code for each product)
- Name (product title)
- BrandName
- Color, Size
- Price, MRP
- Images (links to product photos)
- Quantity, Description

This file represents how the seller organizes their products, but it rarely matches marketplace templates directly.

Attribute Mapping

Because marketplaces have different naming conventions, we must **map** seller file columns → marketplace attributes.

Example:

- Seller's Name → Myntra's productName
- Seller's BrandName → Flipkart's brand
- Seller's Image1 + Image2 → Myntra's images

The mapping step is **critical** for successful uploads.

Assignment Goals

You'll build a **backend-only product-listing tool** that allows:

1. Uploading and saving a **marketplace template** (what attributes that marketplace expects).
2. Uploading a **seller products file** (CSV/Excel).
3. Mapping file columns → marketplace attributes.
4. Validating the mapped data (required fields, data types, rules).
5. Saving the mapping to a database.
6. Exposing APIs to view saved mappings and transformed data.

What to Build (Backend Flow)

1. Marketplace Template Management

- **API** to upload a JSON template for a marketplace.
- **API** to list and fetch saved marketplace templates.

2. Seller Product File Upload

- **API** to upload CSV/Excel.
- Parse and return discovered columns, sample rows, and row count.

3. Mapping API

- **API** to map seller columns → marketplace attributes.
- Store mapping in DB (marketplace + file + column map + timestamp).
- Apply validations such as required fields, enums, numeric checks (price ≤ mrp).

4. Saved Mappings

- **API** to list all mappings.
- **API** to fetch mapping details.

Marketplace Templates (Examples)

Myntra-like Marketplace Template

Required attributes

- productName (string, max 150 chars)
- brand (string)
- gender (enum: Men, Women, Boys, Girls, Unisex)
- category (enum: T-Shirts, Jeans, Dresses, Sarees, Shoes, Bags, Accessories)
- color (string)
- size (enum: XS, S, M, L, XL, XXL, numeric sizes like 32, 34, etc.)
- mrp (number ≥ 0)
- price (number ≥ 0, must be ≤ mrp)
- sku (unique string)
- images (array of URLs)
- description (string)
- material (string)

Flipkart-like Marketplace Template

Required attributes

- title (string, max 200 chars)
- brand (string)
- sellerSku (string)
- categoryPath (string, e.g., “Clothing > Men > T-Shirts”)
- listingPrice (number ≥ 0, must be ≤ mrp)
- mrp (number ≥ 0)
- color (string)
- image1 (URL)
- quantity (integer ≥ 0)
- size (string)
- gender (enum: Men, Women, Boys, Girls, Unisex)
- bulletPoints (array of up to 5 strings, | separated)
- image2, image3 (URL)

- description (string)
- countryOfOrigin (string, e.g., “India”)

Seller File Template (CSV/Excel)

Columns provided by the seller:

- SKU
- Name
- BrandName
- Gender
- Category
- Color
- Size
- MRP
- Price
- Material
- Image1
- Image2
- Quantity
- Description

Seller → Myntra Mapping

- productName ← Name
- brand ← BrandName
- gender ← Gender
- category ← Category
- color ← Color
- size ← Size
- mrp ← MRP
- price ← Price
- sku ← SKU
- description ← Description
- material ← Material
- images ← Image1

Deliverables

- A GitHub repo containing:
    - **Source code (Backend APIs)**
    - **Dockerfile and/or docker-compose.yml file** for containerized execution.
    - **README.md** with:
        - System design & DB schema
        - API documentation (endpoints, request/response format)
        - Setup and usage instructions (including Docker)
        - Instructions to run **unit tests**

Evaluation Criteria

- **Correctness**: Mapping & validation rules implemented properly.
- **Code Quality**: Clean, modular, testable, well-documented.
- **Unit Tests**: Coverage of core logic (CSV parsing, mapping, validation).
- **Dockerization**: App runs in a container across environments (Linux, macOS, Windows).
- **Extensibility**: Easy to add new marketplaces or attributes.