### ****Product Service for Fashion and Apparel Store****

The **Product Service** in the Fashion and Apparel Store is responsible for managing the product catalog, including hierarchical categories, reusable product types, individual products, product variants, and seasonal collections. This service ensures flexibility for admins to define product templates and manage variations while allowing customers to browse and search products effectively.

## ****Core Functionalities****

### ****1. Category Management****

* Maintain a **hierarchical classification** of product categories (e.g., Men > Shirts > Casual Shirts).
* Allow products to be assigned to specific categories.
* Enable APIs for creating, updating, retrieving, and deleting categories.

### ****2. Product Type Management****

* Define reusable templates (ProductType) for products with **attributes** specific to the type (e.g., Shirt attributes: Size, Color, Material).
* Enforce type-specific constraints like allowed values for attributes (e.g., Size can only be S, M, L, XL).
* Enable admins to manage product types via CRUD operations.

### ****3. Product Management****

* Manage **products** linked to a category and a product type.
* Support **custom attributes** at the product level for additional flexibility (e.g., Fit: Slim, Material: Cotton).
* Link products to one or more **seasonal collections** (e.g., Winter 2024).
* Provide filtering and searching by attributes, category, price range, or seasonal collection.

### ****4. Product Variant Management****

* Enable management of product variants based on specific attributes (e.g., Size and Color).
* Maintain stock quantities and optional price adjustments for each variant.
* Allow admins to add, update, or delete variants dynamically.

### ****5. Seasonal Collection Management****

* Associate products with specific **seasonal collections** (e.g., Winter 2024, Spring 2025).
* Enable filtering of products by season to support seasonal promotions.

## ****Relationships Between Entities****

### ****Entity Relationships****

| **Entity** | **Relationship** |
| --- | --- |
| **Category** | One-to-Many: A category can have multiple child categories; a product belongs to one category. |
| **ProductType** | One-to-Many: A product type defines attributes for multiple products. |
| **Product** | One-to-Many: A product can have multiple variants; belongs to one category and one product type. |
| **ProductVariant** | Many-to-One: A product variant belongs to one product. |
| **Season** | Many-to-Many: A product can belong to multiple seasons; a season can include multiple products. |

### ****Entity Descriptions****

#### **1. Category**

* Represents a hierarchical classification of products.
* **Attributes**:
  + id: Unique identifier.
  + name: Name of the category (e.g., Shirts, Casual Shirts).
  + parentCategory: Reference to the parent category for hierarchical relationships.
  + description: Optional description of the category.

#### **2. ProductType**

* Represents the **template** for a product, defining its attributes.
* **Attributes**:
  + id: Unique identifier.
  + name: Name of the product type (e.g., Shirt, Jacket).
  + attributes: List of attributes (e.g., Size, Color, Material).

#### **3. Product**

* Represents an individual product linked to a product type and a category.
* **Attributes**:
  + id: Unique identifier.
  + name: Name of the product (e.g., Slim Fit Casual Shirt).
  + price: Base price of the product.
  + customAttributes: Additional product-specific attributes (e.g., Fit: Slim).
  + categoryId: Reference to the category.
  + productTypeId: Reference to the product type.
  + seasonIds: List of associated seasons.

#### **4. ProductVariant**

* Represents specific variations of a product based on attributes.
* **Attributes**:
  + id: Unique identifier.
  + productId: Reference to the product.
  + variantAttributes: Attributes specific to the variant (e.g., Size: M, Color: Blue).
  + stock: Stock quantity for the variant.
  + additionalPrice: Price adjustment for this variant.

#### **5. Season**

* Represents a specific **seasonal collection**.
* **Attributes**:
  + id: Unique identifier.
  + name: Name of the season (e.g., Winter 2024).
  + startDate: Start date of the season.
  + endDate: End date of the season.

#### **1. Category Management APIs**

##### **1.1 Create Category**

* **Endpoint**: POST /categories
* **Description**: Create a new category with an optional parent category to establish a hierarchy.
* **Request Body**:

json

Copy code

{

"name": "Shirts",

"parentCategoryId": 1,

"description": "All types of shirts for men."

}

* **Response**:
  + **Status**: 201 Created
  + **Body**:

json

Copy code

{

"id": 2,

"name": "Shirts",

"parentCategoryId": 1,

"description": "All types of shirts for men."

}

##### **1.2 Get All Categories**

* **Endpoint**: GET /categories
* **Description**: Retrieve all categories, including their hierarchy.
* **Response**:
  + **Status**: 200 OK
  + **Body**:

json

Copy code

[

{

"id": 1,

"name": "Men",

"subcategories": [

{

"id": 2,

"name": "Shirts",

"subcategories": [

{ "id": 3, "name": "Casual Shirts" },

{ "id": 4, "name": "Formal Shirts" }

]

}

]

}

]

##### **1.3 Update Category**

* **Endpoint**: PUT /categories/{id}
* **Description**: Update the details of an existing category.
* **Request Body**:

json

Copy code

{

"name": "Updated Shirts",

"parentCategoryId": null,

"description": "Updated description for shirts."

}

* **Response**:
  + **Status**: 200 OK
  + **Body**:

json

Copy code

{

"id": 2,

"name": "Updated Shirts",

"parentCategoryId": null,

"description": "Updated description for shirts."

}

##### **1.4 Delete Category**

* **Endpoint**: DELETE /categories/{id}
* **Description**: Delete a category by ID. If the category has subcategories or associated products, return a 400 Bad Request.
* **Response**:
  + **Status**: 204 No Content (Success)
  + **Error**: 400 Bad Request if deletion fails:

json

Copy code

{

"error": "Cannot delete category with subcategories or associated products."

}

#### **2. Product Type Management APIs**

##### **2.1 Create Product Type**

* **Endpoint**: POST /product-types
* **Description**: Define a new product type with attributes.
* **Request Body**:

json

Copy code

{

"name": "Shirt",

"attributes": [

{ "name": "Size", "type": "Enum", "allowedValues": ["S", "M", "L", "XL"] },

{ "name": "Color", "type": "Enum", "allowedValues": ["Red", "Blue", "Green"] },

{ "name": "Material", "type": "Text" },

{ "name": "Fit", "type": "Enum", "allowedValues": ["Slim", "Regular"] },

{ "name": "Occasion", "type": "Enum", "allowedValues": ["Casual", "Formal"] }

]

}

* **Response**:
  + **Status**: 201 Created
  + **Body**:

json

Copy code

{

"id": 1,

"name": "Shirt",

"attributes": [

{ "name": "Size", "type": "Enum", "allowedValues": ["S", "M", "L", "XL"] },

{ "name": "Color", "type": "Enum", "allowedValues": ["Red", "Blue", "Green"] },

{ "name": "Material", "type": "Text" },

{ "name": "Fit", "type": "Enum", "allowedValues": ["Slim", "Regular"] },

{ "name": "Occasion", "type": "Enum", "allowedValues": ["Casual", "Formal"] }

]

}

##### **2.2 Get All Product Types**

* **Endpoint**: GET /product-types
* **Description**: Retrieve all product types with their attributes.
* **Response**:
  + **Status**: 200 OK
  + **Body**:

json

Copy code

[

{

"id": 1,

"name": "Shirt",

"attributes": [

{ "name": "Size", "type": "Enum", "allowedValues": ["S", "M", "L", "XL"] },

{ "name": "Color", "type": "Enum", "allowedValues": ["Red", "Blue", "Green"] },

{ "name": "Material", "type": "Text" },

{ "name": "Fit", "type": "Enum", "allowedValues": ["Slim", "Regular"] },

{ "name": "Occasion", "type": "Enum", "allowedValues": ["Casual", "Formal"] }

]

}

]

##### **2.3 Update Product Type**

* **Endpoint**: PUT /product-types/{id}
* **Description**: Update the attributes or name of a product type.
* **Request Body**:

json

Copy code

{

"name": "Updated Shirt",

"attributes": [

{ "name": "Size", "type": "Enum", "allowedValues": ["S", "M", "L", "XL", "XXL"] },

{ "name": "Material", "type": "Text" },

{ "name": "Occasion", "type": "Enum", "allowedValues": ["Casual", "Formal", "Party"] }

]

}

* **Response**:
  + **Status**: 200 OK
  + **Body**:

json

Copy code

{

"id": 1,

"name": "Updated Shirt",

"attributes": [

{ "name": "Size", "type": "Enum", "allowedValues": ["S", "M", "L", "XL", "XXL"] },

{ "name": "Material", "type": "Text" },

{ "name": "Occasion", "type": "Enum", "allowedValues": ["Casual", "Formal", "Party"] }

]

}

##### **2.4 Delete Product Type**

* **Endpoint**: DELETE /product-types/{id}
* **Description**: Delete a product type by ID. If there are products associated with this type, return a 400 Bad Request.
* **Response**:
  + **Status**: 204 No Content (Success)
  + **Error**: 400 Bad Request if deletion fails:

json

Copy code

{

"error": "Cannot delete product type with associated products."

}

#### **Create Product**

* **Endpoint**: POST /products
* **Description**: Add a new product linked to a product type and category, with optional variants.
* **Request Body**:

json

Copy code

{

"name": "Slim Fit Casual Shirt",

"categoryId": 3,

"productTypeId": 1,

"price": 50.00,

"season": "Winter 2024",

"customAttributes": {

"Material": "Cotton",

"Fit": "Slim"

},

"variants": [

{

"variantAttributes": { "Size": "M", "Color": "Blue" },

"stock": 50,

"additionalPrice": 0.00,

"sku": "SFS-M-BLUE-001"

},

{

"variantAttributes": { "Size": "L", "Color": "Green" },

"stock": 30,

"additionalPrice": 2.00,

"sku": "SFS-L-GREEN-002"

}

]

}

* **Response**:
  + **Status**: 201 Created
  + **Body**:

json

Copy code

{

"id": 101,

"name": "Slim Fit Casual Shirt",

"category": "Casual Shirts",

"price": 50.00,

"customAttributes": {

"Material": "Cotton",

"Fit": "Slim"

},

"season": "Winter 2024",

"variants": [

{

"id": 201,

"variantAttributes": { "Size": "M", "Color": "Blue" },

"stock": 50,

"additionalPrice": 0.00,

"sku": "SFS-M-BLUE-001"

},

{

"id": 202,

"variantAttributes": { "Size": "L", "Color": "Green" },

"stock": 30,

"additionalPrice": 2.00,

"sku": "SFS-L-GREEN-002"

}

]

}

#### **2. Add Product Variant**

* **Endpoint**: POST /products/{productId}/variants
* **Description**: Add a new variant to an existing product. Each variant must have a unique SKU.
* **Request Body**:

json

Copy code

{

"variantAttributes": { "Size": "XL", "Color": "Black" },

"stock": 20,

"additionalPrice": 3.00,

"sku": "SFS-XL-BLACK-003"

}

* **Response**:
  + **Status**: 201 Created
  + **Body**:

json

Copy code

{

"id": 203,

"variantAttributes": { "Size": "XL", "Color": "Black" },

"stock": 20,

"additionalPrice": 3.00,

"sku": "SFS-XL-BLACK-003"

}

#### **3. Get Product Details**

* **Endpoint**: GET /products/{productId}
* **Description**: Retrieve details of a product along with its variants, including SKUs.
* **Response**:
  + **Status**: 200 OK
  + **Body**:

json

Copy code

{

"id": 101,

"name": "Slim Fit Casual Shirt",

"category": "Casual Shirts",

"price": 50.00,

"customAttributes": {

"Material": "Cotton",

"Fit": "Slim"

},

"season": "Winter 2024",

"variants": [

{

"id": 201,

"variantAttributes": { "Size": "M", "Color": "Blue" },

"stock": 50,

"additionalPrice": 0.00,

"sku": "SFS-M-BLUE-001"

},

{

"id": 202,

"variantAttributes": { "Size": "L", "Color": "Green" },

"stock": 30,

"additionalPrice": 2.00,

"sku": "SFS-L-GREEN-002"

}

]

}

#### **4. Update Product Variant**

* **Endpoint**: PUT /products/{productId}/variants/{variantId}
* **Description**: Update the details of an existing variant, including stock or SKU.
* **Request Body**:

json

Copy code

{

"variantAttributes": { "Size": "L", "Color": "Green" },

"stock": 40,

"additionalPrice": 1.50,

"sku": "SFS-L-GREEN-002"

}

* **Response**:
  + **Status**: 200 OK
  + **Body**:

json

Copy code

{

"id": 202,

"variantAttributes": { "Size": "L", "Color": "Green" },

"stock": 40,

"additionalPrice": 1.50,

"sku": "SFS-L-GREEN-002"

}

#### **5. Remove Product Variant**

* **Endpoint**: DELETE /products/{productId}/variants/{variantId}
* **Description**: Remove an existing variant by its ID.
* **Response**:
  + **Status**: 204 No Content