# System Requirements Specification Index

For

# E-Commerce Warehouse Console

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

# Table of Contents

# E-Commerce Warehouse Console
## System Requirements Specification

## 1 Project Abstract

The **E-Commerce Warehouse Console Application** is a pure Java-based application leveraging Java Collections, Lambda Expressions, Stream API, and Method References to enable seamless management of products in a warehouse. This application provides users with the capability to perform CRUD (Create, Read, Update, Delete) operations and execute essential inventory management functionalities.

The Warehouse Management System empowers users to create new product entries, update product details, delete products, and retrieve products based on specific criteria like price. Additionally, the system calculates the total inventory value.

## 2 Business Requirements:

| Screen Name | Console input screen |
|---|---|
| Problem Statement | 1. User needs to enter into the application.<br>2. The user should be able to do the below particular operations.<br>3. The console should display the menu:<br>1. Add Product<br>2. Get Products Sorted by Price<br>3. Get Products Below Specific Price<br>4. Update Product<br>5. Delete Product<br>6. Calculate Total Inventory Value<br>7. Exit |

# 3 Constraints

## 3.1 Add Product

1. Unique ID for Product:
    - Each product must have a unique ID generated using **UUID**.
2. Validate Product Data:
    - Use **Lambda Expression** to validate the product data.
    - The product's price must be greater than `0`, and the quantity must be greater than `0`.
3. Check for Duplicate Product IDs:
    - Use **Stream API** to check if the product ID already exists in the inventory.
4. Add Product to Inventory:
    - Use a **Method Reference** to add the product to the inventory list.

**Common Constraints:**

1. When attempting to add a product with invalid data (e.g., negative price or quantity), the `addProduct` method should throw an `InvalidProductDataException` with the message:

    "Invalid product data provided: Price and quantity must be positive."

2. When attempting to add a product with the same ID already exists in the inventory, the `addProduct` method should throw a `DuplicateProductException` with the message:

    "Duplicate ID detected. Cannot add product."

## 3.2 Update Product

1. Validate Updated Data:
    - Use **Lambda Expression** to validate new data.
    - The updated product's price must be greater than `0`, and the quantity must not be negative.
2. Locate and Update Product:
    - Use the **Stream API** to locate the product in the inventory by its ID.

**Common Constraints:**

1. When updating a product with invalid data (e.g., negative price or quantity), the method should throw an `InvalidProductDataException` with the message:

   “Invalid update data: Price must be positive, and quantity cannot be negative."

2. When attempting to update a product that does not exist in the inventory, the method should throw an `IllegalArgumentException` with the message:

   "Product with ID: " + id + " not found."

### 3.3 Delete Product

1. Check Product Exists:
   - Use the **Stream API** to check if the product with the given ID exists in the inventory.

**Common Constraints::**

1. When attempting to delete a product that does not exist in the inventory, the method should throw an `IllegalArgumentException` with the message:

   "Product with ID: " + id + " not found."

### 3.4 Other Constraints

1. When attempting to retrieve products with a price limit that is negative, the method should return an empty list.
2. When calculating the total inventory value and there are no products in the inventory, the method should return `0.0`.

# 4 Template Code Structure

## 4.1 Package: com.ecommerce

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **InventoryManagementApp.java(class)** | **This represents bootstrap class i.e class with Main method, that shall contain all console interaction with the user.** | **Partially implemented** |

## 4.2 Package: com.ecommerce.models

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **Product (class)** | • This class contains all the properties of the Product class. | Already implemented. |

## 4.3 Package: com.ecommerce.inventory

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **InventoryService (class)** | • This class contains all the methods which are used to write the business logic for the application.<br>• You can create any number of private methods in the class. | Partially implemented. |

### 4.4 Package: com.ecommerce.exception

**Resources**

| Class/Interface | Description | Status |
|---|---|---|
| **DuplicateProductException (Class)** | ● Custom Exception to be thrown when trying to add a product that already exists in the inventory, identified by a duplicate product id. | Already created. |
| **InvalidProductDataException (Class)** | ● Custom Exception to be thrown when trying to add or update a product with invalid data. | Already created. |

# 5 Execution Steps to Follow

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **This editor Auto Saves the code.**
4. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
5. **To run your project use command:**
   mvn clean install exec:java -Dexec.mainClass="com.ecommerce.InventoryManagementApp"
7. **To test your project, use the command**
   mvn test