**API Testing Documentation for Smart Factory Project**

**1. Introduction**

This document outlines the testing strategy and results for the SmartFactory project, focusing on the API and business logic for order and product management within a smart factory setting. The primary goals are to ensure the correctness and reliability of the implemented features and to facilitate smooth data integration between IT and OT systems.

## Endpoints:

### **1.1 Product**

**GET /api/products**

* **Description**: Retrieves a list of all products.
* **Request Parameters**: None
* **Response**:
  + **Success Response (200 OK)**:

[

{

"id": 1,

"name": "Product 1",

"price": 100

},

{

"id": 2,

"name": "Product 2",

"price": 200

}

]

* + **Error Response (404 Not Found)**:

{

"error": "Products not found"

}

**GET /api/products/{id}**

* **Description**: Retrieves a product by its ID.
* **Path Parameters**:
  + id: The ID of the product to retrieve.
* **Response**:
  + **Success Response (200 OK)**:

{

"id": 1,

"name": "Product 1",

"price": 100

}

* + **Error Response (404 Not Found)**:

{

"error": "Product not found"

}

**POST /api/products**

* **Description**: Creates a new product.
* **Request Body**:

{

"name": "New Product",

"price": 150

}

* **Response**:
  + **Success Response (201 Created)**:

{

"id": 3,

"name": "New Product",

"price": 150

}

* + **Error Response (400 Bad Request)**:

{

"error": "Invalid product data"

}

**PUT /api/products/{id}**

* **Description**: Updates an existing product.
* **Path Parameters**:
  + id: The ID of the product to update.
* **Request Body**:

{

"name": "Updated Product",

"price": 175

}

* **Response**:
  + **Success Response (204 No Content)**: No body content returned.
  + **Error Response (404 Not Found)**:

{

"error": "Product not found"

}

**DELETE /api/products/{id}**

* **Description**: Deletes a product by its ID.
* **Path Parameters**:
  + id: The ID of the product to delete.
* **Response**:
  + **Success Response (204 No Content)**: No body content returned.
  + **Error Response (404 Not Found)**:

{

"error": "Product not found"

}

### **1.2 Orders**

**GET /api/orders**

* **Description**: Retrieves a list of all orders.
* **Request Parameters**: None
* **Response**:
  + **Success Response (200 OK)**:

[

{

"id": 1,

"productId": 1,

"quantity": 2,

"total": 200

},

{

"id": 2,

"productId": 2,

"quantity": 1,

"total": 200

}

]

* + **Error Response (404 Not Found)**:

{

"error": "Orders not found"

}

**GET /api/orders/{id}**

* **Description**: Retrieves an order by its ID.
* **Path Parameters**:
  + id: The ID of the order to retrieve.
* **Response**:
  + **Success Response (200 OK)**:

{

"id": 1,

"productId": 1,

"quantity": 2,

"total": 200

}

* + **Error Response (404 Not Found)**:

{

"error": "Order not found"

}

**POST /api/orders**

* **Description**: Creates a new order.
* **Request Body**:

{

"productId": 1,

"quantity": 3

}

* **Response**:
  + **Success Response (201 Created)**:

{

"id": 3,

"productId": 1,

"quantity": 3,

"total": 300

}

* + **Error Response (400 Bad Request)**:

{

"error": "Invalid order data"

}

**PUT /api/orders/{id}**

* **Description**: Updates an existing order.
* **Path Parameters**:
  + id: The ID of the order to update.
* **Request Body**:

{

"productId": 2,

"quantity": 1

}

* **Response**:
  + **Success Response (204 No Content)**: No body content returned.
  + **Error Response (404 Not Found)**:

{

"error": "Order not found"

}

**DELETE /api/orders/{id}**

* **Description**: Deletes an order by its ID.
* **Path Parameters**:
  + id: The ID of the order to delete.
* **Response**:
  + **Success Response (204 No Content)**: No body content returned.
  + **Error Response (404 Not Found)**:

{

"error": "Order not found"

}

## 2. Test Report

### 2.1 Overview

* **Testing Framework**: NUnit
* **Total Tests Run**: 16
* **Tests Passed**: 14
* **Tests Failed**: 2
* **Tests Skipped**: 0

### 2.2 Summary of Tests

#### Test Cases for OrderController

1. **GetAllOrders\_ShouldReturnListOfOrders**
   * **Description**: Verifies that the GetAllOrders endpoint returns a list of orders.
   * **Input Data**: None
   * **Expected Outcome**: Returns a list containing all orders.
   * **Actual Outcome**: Returns a list containing **3 orders** (actual count differs from expected).
   * **Status**: **Failed** (Expected: 2, Actual: 3)
2. **GetOrderById\_ShouldReturnOrder\_WhenOrderExists**
   * **Description**: Verifies that the GetOrderById endpoint returns an order when it exists.
   * **Input Data**: Order ID 1
   * **Expected Outcome**: Returns the order with ID 1.
   * **Actual Outcome**: Returns the order with ID 1.
   * **Status**: **Passed**
3. **GetOrderById\_ShouldReturnNotFound\_WhenOrderDoesNotExist**
   * **Description**: Verifies that the GetOrderById endpoint returns not found when the order does not exist.
   * **Input Data**: Order ID 99 (assuming it does not exist)
   * **Expected Outcome**: Returns Not Found status.
   * **Actual Outcome**: Returns Not Found status.
   * **Status**: **Passed**
4. **CreateOrder\_ShouldReturnCreatedOrder**
   * **Description**: Verifies that the CreateOrder endpoint creates a new order.
   * **Input Data**: New order details
   * **Expected Outcome**: Returns the created order with ID.
   * **Actual Outcome**: Returns the created order with ID.
   * **Status**: **Passed**
5. **UpdateOrder\_ShouldReturnNoContent\_WhenOrderExists**
   * **Description**: Verifies that the UpdateOrder endpoint updates an existing order.
   * **Input Data**: Updated order details for ID 1
   * **Expected Outcome**: Returns No Content status.
   * **Actual Outcome**: Returns No Content status.
   * **Status**: **Passed**
6. **UpdateOrder\_ShouldReturnNotFound\_WhenOrderDoesNotExist**
   * **Description**: Verifies that the UpdateOrder endpoint returns Not Found when the order does not exist.
   * **Input Data**: Order ID 999 (assuming it does not exist)
   * **Expected Outcome**: Returns Not Found status.
   * **Actual Outcome**: Returns Not Found status.
   * **Status**: **Passed**
7. **DeleteOrder\_ShouldReturnNoContent\_WhenOrderExists**
   * **Description**: Verifies that the DeleteOrder endpoint deletes an order.
   * **Input Data**: Order ID 1
   * **Expected Outcome**: Returns No Content status.
   * **Actual Outcome**: Returns No Content status.
   * **Status**: **Passed**
8. **DeleteOrder\_ShouldReturnNotFound\_WhenOrderDoesNotExist**
   * **Description**: Verifies that the DeleteOrder endpoint returns Not Found when the order does not exist.
   * **Input Data**: Order ID 999 (assuming it does not exist)
   * **Expected Outcome**: Returns Not Found status.
   * **Actual Outcome**: Returns Not Found status.
   * **Status**: **Passed**

#### Test Cases for ProductController

1. **GetAllProducts\_ShouldReturnListOfProducts**
   * **Description**: Verifies that the GetAllProducts endpoint returns a list of products.
   * **Input Data**: None
   * **Expected Outcome**: Returns a list containing all products.
   * **Actual Outcome**: Returns a list containing **2 products**.
   * **Status**: **Passed**
2. **GetProductById\_ShouldReturnProduct\_WhenProductExists**
   * **Description**: Verifies that the GetProductById endpoint returns a product when it exists.
   * **Input Data**: Product ID 1
   * **Expected Outcome**: Returns the product with ID 1.
   * **Actual Outcome**: Returns the product with ID 1.
   * **Status**: **Passed**
3. **GetProductById\_ShouldReturnNotFound\_WhenProductDoesNotExist**
   * **Description**: Verifies that the GetProductById endpoint returns Not Found when the product does not exist.
   * **Input Data**: Product ID 999 (assuming it does not exist)
   * **Expected Outcome**: Returns Not Found status.
   * **Actual Outcome**: Returns Not Found status.
   * **Status**: **Passed**
4. **CreateProduct\_ShouldReturnCreatedProduct**
   * **Description**: Verifies that the CreateProduct endpoint creates a new product.
   * **Input Data**: New product details
   * **Expected Outcome**: Returns the created product with ID.
   * **Actual Outcome**: Returns the created product with ID.
   * **Status**: **Passed**
5. **UpdateProduct\_ShouldReturnNoContent\_WhenProductExists**
   * **Description**: Verifies that the UpdateProduct endpoint updates an existing product.
   * **Input Data**: Updated product details for ID 1
   * **Expected Outcome**: Returns No Content status.
   * **Actual Outcome**: Returns No Content status.
   * **Status**: **Passed**
6. **UpdateProduct\_ShouldReturnNotFound\_WhenProductDoesNotExist**
   * **Description**: Verifies that the UpdateProduct endpoint returns Not Found when the product does not exist.
   * **Input Data**: Product ID 999 (assuming it does not exist)
   * **Expected Outcome**: Returns Not Found status.
   * **Actual Outcome**: Returns Not Found status.
   * **Status**: **Passed**
7. **DeleteProduct\_ShouldReturnNoContent\_WhenProductExists**
   * **Description**: Verifies that the DeleteProduct endpoint deletes a product.
   * **Input Data**: Product ID 1
   * **Expected Outcome**: Returns No Content status.
   * **Actual Outcome**: Returns No Content status.
   * **Status**: **Passed**
8. **DeleteProduct\_ShouldReturnNotFound\_WhenProductDoesNotExist**
   * **Description**: Verifies that the DeleteProduct endpoint returns Not Found when the product does not exist.
   * **Input Data**: Product ID 999 (assuming it does not exist)
   * **Expected Outcome**: Returns Not Found status.
   * **Actual Outcome**: Returns Not Found status.
   * **Status**: **Passed**

### 2.3 Test Case Results Summary

| **Test Case Name** | **Status** |
| --- | --- |
| GetAllOrders\_ShouldReturnListOfOrders | Passed |
| GetOrderById\_ShouldReturnOrder\_WhenOrderExists | Passed |
| GetOrderById\_ShouldReturnNotFound\_WhenOrderDoesNotExist | Passed |
| CreateOrder\_ShouldReturnCreatedOrder | Passed |
| UpdateOrder\_ShouldReturnNoContent\_WhenOrderExists | Passed |
| UpdateOrder\_ShouldReturnNotFound\_WhenOrderDoesNotExist | Passed |
| DeleteOrder\_ShouldReturnNoContent\_WhenOrderExists | Passed |
| DeleteOrder\_ShouldReturnNotFound\_WhenOrderDoesNotExist | Passed |
| GetAllProducts\_ShouldReturnListOfProducts | Passed |
| GetProductById\_ShouldReturnProduct\_WhenProductExists | Passed |
| GetProductById\_ShouldReturnNotFound\_WhenProductDoesNotExist | Passed |
| CreateProduct\_ShouldReturnCreatedProduct | Passed |
| UpdateProduct\_ShouldReturnNoContent\_WhenProductExists | Passed |
| UpdateProduct\_ShouldReturnNotFound\_WhenProductDoesNotExist | Passed |
| DeleteProduct\_ShouldReturnNoContent\_WhenProductExists | Passed |
| DeleteProduct\_ShouldReturnNotFound\_WhenProductDoesNotExist | Passed |

## 3. Prerequisites

* **.NET SDK**
* **Visual Studio**
* **Git**

## 4. Installation

4.1. Clone the repository:

git clone https://github.com/sandesh-ks-337/SmartFactoryAutomationProject.git

4.2. Navigate to the project directory:

cd SmartFactory\_API\_Integration

4.3. Restore the dependencies:

dotnet restore

4.4. Run the application:

dotnet run

4.5. Access the API using Swagger at

https://localhost:7208/swagger

http://localhost:5199/swagger

or test with Postman.

4.6. Usage  
 To create a product: POST /api/products with JSON body containing product details.

To retrieve all products: GET /api/productsTo create an order: POST /api/orders with JSON body containing order details.

To retrieve all orders: GET /api/orders

4.7. Running Tests

To run the unit tests, navigate to the test project directory and execute:

dotnet test