### Product Management

#### Test Cases

##### 1. Creating a New Product

* **Description**: Verify that a new product can be created with valid data.
* **Steps**:
  1. Send a POST request to /api/Product with a JSON containing the details of the new product.
  2. Verify that the response has a status code of 201 Created.
  3. Ensure that the product has been created correctly by checking the response or querying the product list.
* **Test Data**:

json

{

"productName": "New Product",

"quantity": 10,

"status": "Available"

}

* **Expected Result**: Response with status code 201 and the new product created correctly.

##### 2. Reading a Product

* **Description**: Verify that the information of an existing product can be read.
* **Steps**:
  1. Send a GET request to /api/Product/{id} with the ID of the existing product.
  2. Verify that the response has a status code of 200 OK.
  3. Ensure that the response contains the correct product details.
* **Test Data**:

json

{

"id": 1

}

* **Expected Result**: Response with status code 200 and correct product details.

##### 3. Updating a Product

* **Description**: Verify that an existing product can be updated.
* **Steps**:
  1. Send a PUT request to /api/Product/{id} with a JSON containing the updated product data.
  2. Verify that the response has a status code of 200 OK.
  3. Ensure that the product has been updated correctly by checking the response or querying the product list.
* **Test Data**:

json

{

"productName": "Updated Product",

"quantity": 20,

"status": "Unavailable"

}

* **Expected Result**: Response with status code 200 and the product updated correctly.

##### 4. Deleting a Product

* **Description**: Verify that an existing product can be deleted.
* **Steps**:
  1. Send a DELETE request to /api/Product/{id} with the ID of the product to be deleted.
  2. Verify that the response has a status code of 204 No Content.
  3. Ensure that the product has been deleted correctly by checking the product list or attempting to access the deleted product.
* **Test Data**:

json

{

"id": 1

}

* **Expected Result**: Response with status code 204 and the product deleted correctly.

##### 5. Error Handling for Invalid Product Data

* **Description**: Verify that invalid data is handled correctly when creating or updating a product.
* **Steps**:
  1. Send a POST or PUT request to /api/Product with invalid product data.
  2. Verify that the response has a status code of 400 Bad Request.
  3. Ensure that the error message is appropriate for the invalid data.
* **Test Data**:

json

{

"productName": "",

"quantity": -10,

"status": ""

}

* **Expected Result**: Response with status code 400 and appropriate error message.

##### 6. Search and Filter Functionalities

* **Description**: Verify that the search and filter functionalities for products work correctly.
* **Steps**:
  1. Send a GET request to /api/Product/search with search and filter parameters.
  2. Verify that the response has a status code of 200 OK.
  3. Ensure that the results match the search and filter criteria.
* **Test Data**:

json

{

"searchTerm": "Product",

"status": "Available"

}

* **Expected Result**: Response with status code 200 and products matching the search and filter criteria.