

## PART 3: TEST EXECUTION AND REPORTING

### 1. Test Execution – Backend (C# APIS):

#### 1. TC001: Login with Valid Credentials - POST <http://localhost:5044/api/User/login>

The screenshot shows a Postman interface for a POST request to `http://localhost:5044/api/User/login`. The request body is a JSON object with `username: "testuser"` and `password: "password"`. The response is a 200 OK status with a response time of 2 ms and a body size of 171 B. The response body is a JSON object containing `token: "sampletoken"`.

```
1 {
2   "username": "testuser",
3   "password": "password"
4 }
```

Body

```
1 {
2   "token": "sampletoken"
3 }
```

#### 2. TC002: Login with invalid Credentials - POST

##### <http://localhost:5044/api/User/login>


The screenshot shows a Postman interface for a POST request to `http://localhost:5044/api/User/login`. The request body is a JSON object with `username: "username"` and `password: "wrongpassword"`. The response is a 401 Unauthorized status with a response time of 14 ms and a body size of 331 B. The response body is a JSON object containing error details: `type: "https://tools.ietf.org/html/rfc9110#section-15.6.2"`, `title: "Unauthorized"`, `status: 401`, and `traceId: "00-c46a8e68e94078c9326c8855aacf7bee-d25aa619b0ca72ae-00"`.

```
1 {
2   "username": "username",
3   "password": "wrongpassword"
4 }
```

Body

```
1 {
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.6.2",
3   "title": "Unauthorized",
4   "status": 401,
5   "traceId": "00-c46a8e68e94078c9326c8855aacf7bee-d25aa619b0ca72ae-00"
6 }
```

### 3. TC003: Login with both fields empty POST <http://localhost:5044/api/User/login>

 Login / TC003 - Login with both fields empty.

POST

http://localhost:5044/api/User/login

Send

Params Auth Headers (11) **Body** Scripts Settings Cookies Beautify

raw JSON

```
1 {
2   "username": "",
3   "password": ""
4 }
```

Body 401 Unauthorized 5 ms 331 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.5.2",
3   "title": "Unauthorized",
4   "status": 401,
5   "traceId": "00-d4d95ac5930f84c9118d2f4ccf9adca1-7d0b61040b1216eb-00"
6 }
```

#### 4. TC004: Login with empty user field POST <http://localhost:5044/api/User/login>

HTTP Login / TC004 - Login with empty user field Save Share

**POST** ▼  Send ▼

Params Auth Headers (11) Body ● Scripts Settings Cookies

raw ▼ **JSON** ▼ Beautify

```
1 {
2   "username": "",
3   "password": "password"
4 }
```

Body ▼ 401 Unauthorized 5 ms 331 B Save as example ⋮

Pretty Raw Preview Visualize **JSON** ▼ ↺ 📄 🔍

```
1 {
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.5.2",
3   "title": "Unauthorized",
4   "status": 401,
5   "traceId": "00-525d2e103bf2904522ba725989ab4603-8cf194c953a2ca1c-00"
6 }
```

## 5. TC005: Login with empty password field POST

<http://localhost:5044/api/User/login>

HTTP Login / TC005 - Login with empty password field Save Share

POST ▼  Send ▼

Params Auth Headers (11) Body ● Scripts Settings Cookies

raw ▼ JSON ▼ Beautify

```
1 {
2   "username": "user",
3   "password": ""
4 }
```

Body ▼ 401 Unauthorized 8 ms 331 B Save as example ⋮

Pretty Raw Preview Visualize JSON ▼ ↺ 📄 🔍

```
1 {
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.5.2",
3   "title": "Unauthorized",
4   "status": 401,
5   "traceId": "00-fa972c50bfb3ca60eafa6e47d4567729-62b655f7045bc8ff-00"
6 }
```

## 6. TC006: Create Product POST <http://localhost:5044/api/Product>

[HTTP](#) Product / TC006 - Create Product Save Share

POST

http://localhost:5044/api/Product

Send

Params Auth Headers (9) **Body** Scripts Settings Cookies

raw JSON Beautify

```
1 {
2   "id": 0,
3   "name": "Product1",
4   "price": 1
5 }
```

Body 201 Created 9 ms 236 B Save as example

Pretty Raw Preview Visualize

JSON

Copy Search

```
1 {
2   "id": 0,
3   "name": "Product1",
4   "price": 1
5 }
```

## 7. TC007: Consult Product List GET <http://localhost:5044/api/Product>

Product / TC007 - Consult Product List

Save

Share

GET

http://localhost:5044/api/Product

Send

Params Auth Headers (9) Body • Scripts Settings

Cookies

### Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body



200 OK

9 ms

224 B



Save as example

...

Pretty

Raw

Preview


Visualize

JSON



```
1  [
2    {
3      "id": 0,
4      "name": "Product1",
5      "price": 1
6    },
7    {
8      "id": 1,
9      "name": "Product2",
10     "price": 10
11   }
12 ]
```

## 8. TC008: Create Product with Existing ID POST <http://localhost:5044/api/Product>

 Product / TC008 - Create Product with Existing ID Save Share

POST

http://localhost:5044/api/Product

Send

Params Auth Headers (9) **Body** Scripts Settings Cookies

raw

JSON



Beautify

```
1 {
2   "id": 0,
3   "name": "Product3",
4   "price": 30
5 }
```

Body 201 Created 7 ms 237 B Save as example

Pretty Raw Preview Visualize

JSON

```
1 {
2   "id": 0,
3   "name": "Product3",
4   "price": 30
5 }
```

## 9. TC009: Browse Product by ID GET <http://localhost:5044/api/Product/{Id}>

GET TC009 - Browse Produc

+

⌵

No environment ⌵

Product / TC009 - Browse Product by ID

Save ⌵

Share

GET ⌵

http://localhost:5044/api/Product/1

Send ⌵

Params

Auth

Headers (9)

Body ●

Scripts

Settings

Cookies

Query Params

	Key	Value	Description	⋮ Bulk Edit
	Key	Value	Description	

Body ⌵

200 OK 14 ms 185 B

Save as example ⋮

Pretty

Raw

Preview

Visualize

JSON ⌵

⌵

📄

🔍

```
1 {
2   "id": 1,
3   "name": "Product2",
4   "price": 10
5 }
```



## 10.TC010: Edit Product by ID PUT <http://localhost:5044/api/Product/{Id}>

PUT TC010 - Edit Product by

+

⌵

No environment ⌵

HTTP Product / TC010 - Edit Product by ID

Save ⌵

Share

PUT ⌵

http://localhost:5044/api/Product/1

Send ⌵

Params

Auth

Headers (9)

Body ●

Scripts

Settings

Cookies

raw ⌵

JSON ⌵

Beautify


```
1 {
2   "id": 0,
3   "name": "Product1Edit",
4   "price": 100
5 }
```



Body ⌵


🌐 204 No Content 2 ms 81 B


Save as example ⋮

## 11.TC011: Delete Product by ID DELETE <http://localhost:5044/api/Product/{Id}>


 Product / TC011 - Delete Product by ID Copy

 Save 

 Share

DELETE 

http://localhost:5044/api/Product/0

Send 

Params

Auth


Headers (7)


Body

Scripts

Settings


Cookies




raw 

JSON 

Beautify

1

Body 

 204 No Content 6 ms 81 B  Save as example 

## 12. TC012: Product Upgrade with Nonexistent ID PUT

<http://localhost:5044/api/Product/{id}>

Product / TC012 - Product Upgrade with Nonexistent ID

PUT <http://localhost:5044/api/Product/20> Send

Params Auth Headers (9) Body • Scripts Settings Cookies

raw JSON Beautify

```
1 {
2   "id": 0,
3   "name": "Product1Edit",
4   "price": 100
5 }
```

Body 404 Not Found 4 ms 325 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.5.5",
3   "title": "Not Found",
4   "status": 404,
5   "traceId": "00-c0a7c83aa1f0e6d7ed9d144eee210d84-a060a9dea086d6b3-00"
6 }
```

### 2. Documentation of Results:

- Results Format:
  - Test Case ID:** Identification of the test case.
  - Test Case Title:** Title of the test case to be executed.
  - Result:** Pass if the test case has been successful, Fail if it does not meet expectations.
  - Comments:** Any additional details, such as if there were unexpected behaviors or if the result was not as expected.

Test Case ID	Test Case Title	Result	Comments
TC001	Login with Valid Credentials	Pass	Authentication was successful and a token was received in the response body.
TC002	Login with Invalid Credentials	Pass	The system correctly rejected the authentication and returned a 401 (Unauthorized) error.

TC003	Login with both fields empty.	Pass	The system correctly returned a 401 (Unauthorized) error when attempting to log in with both fields empty.
TC004	Login with empty user field.	Pass	The system correctly returned a 401 (Unauthorized) error when trying to start the system.
TC005	Login with empty password field.	Pass	The system correctly returned a 401 (Unauthorized) error when attempting to log in with only the password field.
TC006	Create Product	Pass	The system successfully created the product with the data provided and returned a status 201 along with the details of the product created.
TC007	Consult Product List	Pass	The system successfully returned a list of products with status 200. The list contains the products with their respective id, name, and price.
TC008	Create Product with Existing ID	Fail	The system allowed the creation of a product with an ID that already existed, which is incorrect. A status 201 was received when a status 409 is expected.
TC009	Browse Product by ID	Pass	The product query with ID 1 was successful. A status 200 was received.
TC010	Edit Product by ID	Pass	The edition of the product with ID 0 was successful. A status 204 was received, indicating that the operation was completed successfully with no additional content.
TC011	Delete Product by ID	Pass	The deletion of the product with the specified ID was successful. A status

			204 was received, indicating that the operation was completed successfully with no additional content.
TC012	Product Upgrade with Nonexistent ID	Pass	Product update with a non-existent ID was handled correctly. A 404 status was received.
TC013	Elimination of Product with Inexistent ID		
TC014	View all orders		
TC015	Search for an order using an existing ID.		
TC016	Search an order for a non-existent ID.		
TC017	Search order with invalid ID		
TC018	Create order with valid data		
TC019	Create order with empty productName		
TC020	Create order with quantity at 0		
TC021	Create order with negative quantity		
TC022	Create order with empty status		
TC023	Create order with duplicate ID		
TC024	Create order with all empty fields		
TC025	Create order with incorrect data type		
TC026	Update order with valid data (without changing the ID)		
TC027	Update the ID of an order		
TC028	Update order with empty productName		
TC029	Update order with quantity at 0		
TC030	Update order with negative quantity		
TC031	Update the order with an empty status		
TC032	Update order with non-existent ID		
TC033	Update order with all fields empty		
TC034	Update order with incorrect data type		

TC035	Delete order with existing ID		
TC036	Delete order with non-existent ID		
TC037	Delete order with non-numeric ID		
TC038	Delete order with negative ID		
TC039	Delete order without authorization		

### 3. Defect Reporting:

- Defect Details:
  - **Defect ID:** A unique identifier for each defect.
  - **Associated Test Case ID:** Identify in which test case the defect occurs.
  - **Description:** Clear and concise explanation of the defect found.
  - **Steps to Replicate:** Detailed step-by-step to replicate the defect.
  - **Actual Result:** What happens when the scenario is run.
  - **Expected Result:** What should happen according to the requirements.
  - **Severity:** Classification of the defect (e.g., Critical, Major, Minor).

Defect ID	Associated Test Case ID	Description	Steps to Replicate	Actual Result	Expected Result	Severity
DEF001	TC003, TC004, TC005	The system does not display a specific error message for empty fields in the login.	1. Access the login page. 2. Leave both fields empty, only fill in the user name, or only fill in the password. 3. Click on "Login".	The system returns a 401 (Unauthorized) error.	The system should display a specific message indicating that all fields are mandatory.	Minor
DEF002	TC008	The system allows the creation of a product with an ID that already exists, which	1. Send a POST request to /api/Product with an existing product ID.	The system returns a 201 status.	The system should return an error with a 409 status and a message	Major (as it allows the creation of duplicates, which can cause problems in

		should not be allowed.	2. Observe the system response which is a code 201.		indicating that the ID already exists.	product management).