Test Cases - Ingrid Garcia

**August 4th, 2025**

# **Backend Test Cases**

#### Login

**http://localhost:5044/api/user/login**

### Test Case: Successful login with valid credentials

Section: User Authentication  
API Method: POST  
JSON Payload:  
{  
'username': 'string',   
'password': 'string'  
}

Expected Result: 200 OK

### Test Case: Empty JSON body

Section: User Authentication  
API Method: POST  
JSON Payload:  
{  
'username': 'string',   
'password': 'string'  
}

Expected Result: 400 Bad Request

### Test Case: Invalid Content-Type: text/plain

Section: User Authentication  
API Method: POST  
JSON Payload:  
{  
'username': 'string',   
'password': 'string'  
}

Expected Result: 415 Unsupported Media Type

### Test Case: Missing password field

Section: User Authentication  
API Method: POST  
JSON Payload:  
{  
'username': 'string',  
 'password': 'string'}

Expected Result: 400 Bad Request

### Test Case: Missing username field

Section: User Authentication  
API Method: POST  
JSON Payload:  
{  
'username': 'string',   
'password': 'string'  
}

Expected Result: 400 Bad Request

### Test Case: Invalid URL Login

Section: User Authentication  
API Method: POST  
JSON Payload:  
{  
'username': 'string',   
'password': 'string'  
}

Expected Result: 404 Not Found

### Test Case: Incorrect password (i.e psswors) with valid username

Section: User Authentication  
API Method: POST  
JSON Payload:  
{  
'username': 'string',   
'password': 'string'  
}

Expected Result: 401 Unauthorized

### Test Case: Incorrect username (i.e test) with valid password

Section: User Authentication  
API Method: POST  
JSON Payload:  
{  
'username': 'string',   
'password': 'string'  
}

Expected Result: 401 Unauthorized

### Test Case: Incorrect username and password

Section: User Authentication  
API Method: POST  
JSON Payload:  
{  
'username': 'string',   
'password': 'string'  
}

Expected Result: 401 Unauthorized

#### Product Management

**http://localhost:5044/api/product**

## **Test Case: Retrieve product list when database is empty**

Section: Product Management  
API Method: GET  
JSON Payload:

Expected Result: 200

## **Test Case: Retrieve product list when products exist**

Section: Product Management  
API Method: GET  
JSON Payload:

Expected Result: 200 OK

## **Test Case: Send unexpected query param (i.e /api/product?23?**

Section: Product Management  
API Method: GET

Expected Result: 400 Bad Request

## **Test Case: Get all products with invalid token**

Section: Product Management  
API Method: GET

Expected Result: 401 Unauthorized

## **Test Case: Recently deleted product does not appear in full list**

Section: Product Management  
API Method: GET

Expected Result: 200 OK

## **Test Case: Single Product - Get a product by valid ID**

Section: Product Management  
API Method: GET

Expected Result: 200 OK

## **Test Case: Single Product - Get a product by a non-existent ID**

Section: Product Management  
API Method: GET  
  
Expected Result: 404 Not Found

## **Test Case: Single Product - Use invalid ID format (i.e string)**

Section: Product Management  
API Method: GET

Expected Result: 400 Bad Request

## **Test Case: Single Product - Get a recently created product**

Section: Product Management  
API Method: GET

Expected Result: 200 OK

## **Test Case: Single Product - Get a deleted product**

Section: Product Management  
API Method: GET

Expected Result: 204 No Content

## **Test Case: Create Product with all valid fields**

Section: Product Management  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 201 Created

## **Test Case: Create Product with missing name**

Section: Product Management  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 400 Bad Request

## **Test Case: Create Product with negative price**

Section: Product Management  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 400 Bad Request

## **Test Case: Create Product with invalid characters in name field**

Section: Product Management  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 400 Bad Request

## **Test Case: Create Product with exact same name/fields**

Section: Product Management  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 409 Conflict

## **Test Case: Create Product with same ID, different name**

Section: Product Management  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 409 Conflict

## **Test Case: Create Product with invalid token**

Section: Product Management  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 401 Unauthorized

## **Test Case: Update name and price of existing product**

Section: Product Management  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 200 OK

## **Test Case: Update only one field - id**

Section: Product Management  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 400 Bad Request or 405 Method Not Allowed

## **Test Case: Update only one field - name**

Section: Product Management  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 200 OK

## **Test Case: Update only one field - price**

Section: Product Management  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 200 OK

## **Test Case: Update product with invalid ID**

Section: Product Management  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 404 Not Found

## **Test Case: Update with invalid types (i.e string instead of number for price)**

Section: Product Management  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 400 Bad Request

## **Test Case: Update with empty or missing fields**

Section: Product Management  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'name': 'string',  
 'price': 0  
}

Expected Result: 400 Bad Request

## **Test Case: Update same product from two tabs simultaneously**

Section: Product Management  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 400 Bad Request

## **Test Case: Update with invalid token**

Section: Product Management  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'name': 'string',   
'price': 0  
}

Expected Result: 401 Unauthorized

## **Test Case: Delete product by valid ID**

Section: Product Management  
API Method: DELETE

Expected Result: 204 No Content

## **Test Case: Delete product with non-existent ID**

Section: Product Management  
API Method: DELETE

Expected Result: 204 No Content

## **Test Case: Attempt to delete same product twice**

Section: Product Management  
API Method: DELETE

Expected Result: 204 No Content

#### Order Processing

**http://localhost:5044/api/order**

## **Test Case: Retrieve all existing orders**

Section: Order Processing  
API Method: GET

Expected Result: 200 OK

## **Test Case: Retrieve order list when none exist**

Section: Order Processing  
API Method: GET

Expected Result: 200 OK

## **Test Case: Invalid Token**

Section: Order Processing  
API Method: GET

Expected Result: 401 Unauthorized

## **Test Case: Rapid repeated calls (looking for throttling behavior)**

Section: Order Processing  
API Method: GET

Expected Result: 429 Too Many Requests or 200 OK (if no throttling logic)

## **Test Case: Deleted Order does not appear in full list**

Section: Order Processing  
API Method: GET

Expected Result: 200 OK

## **Test Case: Single Order - Get order by valid ID**

Section: Order Processing  
API Method: GET

Expected Result: 200 OK

## **Test Case: Single Order - Use non-existent order ID**

Section: Order Processing  
API Method: GET

Expected Result: 404 Not Found

**Test Case: Single Order - Use string value as ID**

Section: Order Processing  
API Method: GET

Expected Result: 400 Bad Request

## **Test Case: Single Order - Get an order right after creation**

Section: Order Processing  
API Method: GET

Expected Result: 200 OK

## **Test Case: Single Order - Get a deleted order**

Section: Order Processing  
API Method: GET

Expected Result: 404 Not Found

## **Test Case: Single Order - Invalid Token**

Section: Order Processing  
API Method: GET

Expected Result: 401 Unauthorized

**Test Case: Create an order with valid product name, quantity, and status**Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 201 Created

## **Test Case: Create multiple orders in sequence**

Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 201 Created

## **Test Case: Create order with missing field - id**

Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 400 Bad Request

## **Test Case: Create order with missing field - productName**

Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 400 Bad Request

## **Test Case: Create order with missing field - quantity**

Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 400 Bad Request

## **Test Case: Create order with missing field - status**

Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 400 Bad Request

## **Test Case: Create order with invalid fields**

Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 400 Bad Request

## **Test Case: Create order invalid token**

Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 401 Unauthorized

## **Test Case: Create order with malformed JSON**

Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 201 Created

## **Test Case: Create duplicate order**

Section: Order Processing  
API Method: POST  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 201 Created

## **Test Case: Update order status**

Section: Order Processing  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
 }

Expected Result: 200 OK

## **Test Case: Update only the quantity**

Section: Order Processing  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 200 OK

## **Test Case: Update with non-existent ID**

Section: Order Processing  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 404 Not Found

## **Test Case: Update with empty request body**

Section: Order Processing  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 400 Bad Request

## **Test Case: Update with Invalid Token**

Section: Order Processing  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 401 Unauthorized

## **Test Case: Update with invalid status value**

Section: Order Processing  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 400 Bad Request

## **Test Case: Update same order from 2 clients simultaneously**

Section: Order Processing  
API Method: PUT  
JSON Payload:  
{  
'id': 0,   
'productName': 'string',   
'quantity': 0,   
'status': 'string'  
}

Expected Result: 409 Conflict or 500 Internal Server Error (depending on handling logic)

## **Test Case: Delete an order with valid ID**

Section: Order Processing  
API Method: DELETE

Expected Result: 204 No Content

## **Test Case: Delete with malformed ID (i.e string)**

Section: Order Processing  
API Method: DELETE

Expected Result: 400 Bad Request

## **Test Case: Delete with non-existent ID**

Section: Order Processing  
API Method: DELETE

Expected Result: 400 Bad Request

## **Test Case: Confirm deletion by trying to GET**

Section: Order Processing  
API Method: DELETE

Expected Result: 200 OK

## **Test Case: Delete already deleted order**

Section: Order Processing  
API Method: DELETE

Expected Result: 400 Bad Request

# **Frontend Test Cases**

## **Verify user can log in with valid credentials**

Section: User Authentication  
Steps:

1. Navigate to the login page
2. Enter valid username and password
3. Click the login button
4. Verify user is redirected to the dashboard

Expected Result: User should be redirected to dashboard successfully

## **Verify error message appears with invalid credentials**

Section: User Authentication  
Steps:

1. Navigate to the login page
2. Enter invalid username or password
3. Click the login button
4. Verify error message is displayed

Expected Result: Error message should indicate invalid credentials

## **Verify password input is masked**

Section: User Authentication  
Steps:

1. Navigate to the login page
2. Enter password into password field
3. Verify that entered characters are displayed as dots or asterisks

Expected Result: Password should be masked in the input field

## **Verify error message when password is missing**

Section: User Authentication  
Steps:

1. Navigate to the login page
2. Enter username only
3. Click login
4. Check for validation error

Expected Result: Error message should indicate that password is required

## **Verify error message when username is missing**

Section: User Authentication  
Steps:

1. Navigate to the login page
2. Enter password only
3. Click login
4. Check for validation error

Expected Result: Error message should indicate that username is required

## **Verify error message when both username and password are missing**

Section: User Authentication  
Steps:

1. Navigate to the login page
2. Click login without entering any credentials
3. Check for validation errors

Expected Result: Error message should indicate that both fields are required

## **Verify dashboard loads upon successful login**

Section: User Authentication  
Steps:

1. Log in with valid credentials
2. Verify that the dashboard loads with user-specific content

Expected Result: Dashboard should display relevant user data

## **Check UI responsiveness of login page on different screen sizes**

Section: User Authentication  
Steps:

1. Open login page on desktop, tablet, and mobile devices
2. Verify layout adjusts and remains usable

Expected Result: Login page should be responsive and usable across devices

## **Verify logout button is visible and functional**

Section: User Authentication  
Steps:

1. Log in with valid credentials
2. Click the logout button
3. Verify user is redirected to login screen

Expected Result: User should be successfully logged out