

Modul: Manual API Testing

Menggunakan Curl

Dalam dunia yang semakin digital, platform aplikasi populer seperti web dan mobile saling terhubung dan bertukar data melalui antarmuka pemrograman aplikasi (API). API testing menjadi sangat penting untuk memastikan komunikasi antar sistem ini berjalan lancar dan data yang dipertukarkan akurat dan aman.

Apa itu API Testing?

API testing adalah proses pengujian fungsionalitas, keamanan, performa, dan kehandalan API. Dilakukan secara sistematis untuk mengidentifikasi dan memperbaiki bug atau celah keamanan sebelum API tersebut dipublikasikan dan digunakan oleh aplikasi lain.

Mengapa API Testing Penting?

- **Mencegah Bug dan Error:** API yang tidak teruji dengan baik dapat menyebabkan aplikasi yang bergantung padanya mengalami crash, menampilkan data yang salah, atau tidak berfungsi sebagaimana mestinya.
- **Menjamin Keamanan Data:** API yang *vulnerable* dapat menjadi celah bagi *hacker* untuk menyusup ke sistem dan mencuri data sensitif. Pengujian API yang ketat membantu memastikan keamanan data yang dipertukarkan.
- **Meningkatkan Performa:** API yang lambat atau tidak responsif dapat berdampak negatif pada pengalaman pengguna aplikasi. Pengujian API membantu mengidentifikasi dan mengatasi masalah performa.
- **Memastikan Keandalan:** API yang handal memastikan ketersediaannya dan kemampuannya untuk menangani beban pengguna yang tinggi. Pengujian API berkontribusi terhadap ketahanan sistem secara keseluruhan.

Pada modul ini, kita akan berfokus tentang pengujian API testing dalam skala integration testing. Dalam skala integration testing, API testing berfokus pada bagaimana API berinteraksi dengan modul atau komponen lain dalam sistem. Hal ini tentunya bertujuan untuk memastikan bahwa:

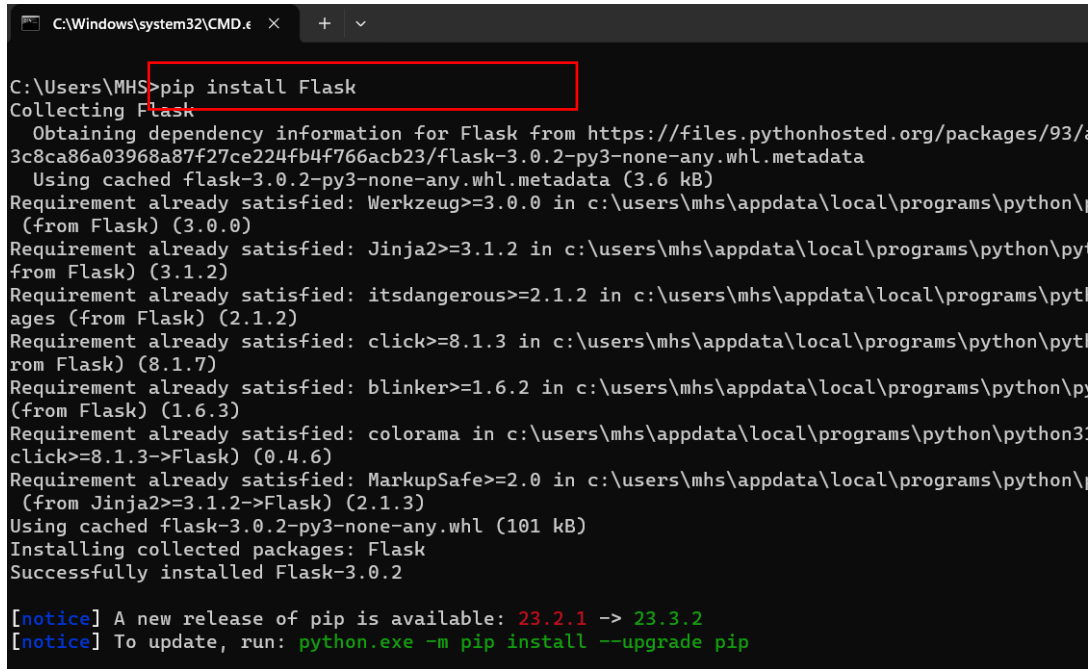
- **API dapat bertukar data dengan lancar:** API harus dapat mengirim dan menerima data dengan benar dari modul atau komponen lain yang terintegrasi dengannya.
- **Format data sesuai dengan yang diharapkan:** Struktur dan isi data yang dipertukarkan antar API dan komponen lain harus kompatibel dan sesuai dengan spesifikasi.
- **Perilaku API terintegrasi dengan baik:** API harus berperilaku sesuai dengan yang diharapkan ketika berinteraksi dengan modul atau komponen lain.

Percobaan 1: Persiapan API Server dengan Flask sebagai System-under-Test (SUT)

Software yang anda butuhkan sebelum memulai percobaan ini adalah:

- Python 3.11: <https://www.python.org/downloads/release/python-3118/>
- Laragon: <https://laragon.org/download/index.html>
- Visual Studio Code: <https://code.visualstudio.com/download>

Instalasi Flask dengan menggunakan pip:



```
C:\Windows\system32\CMD.exe X + v
C:\Users\MHS>pip install Flask
Collecting Flask
  Obtaining dependency information for Flask from https://files.pythonhosted.org/packages/93/3c/3c8ca86a03968a87f27ce224fb4f766acb23/flask-3.0.2-py3-none-any.whl.metadata
  Using cached flask-3.0.2-py3-none-any.whl.metadata (3.6 kB)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\mhs\appdata\local\programs\python\python311\Scripts\werkzeug.exe (from Flask) (3.0.0)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\mhs\appdata\local\programs\python\python311\Scripts\jinja2.exe (from Flask) (3.1.2)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\mhs\appdata\local\programs\python\python311\Scripts\itsdangerous.exe (from Flask) (2.1.2)
Requirement already satisfied: click>=8.1.3 in c:\users\mhs\appdata\local\programs\python\python311\Scripts\click.exe (from Flask) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\mhs\appdata\local\programs\python\python311\Scripts\blinker.exe (from Flask) (1.6.3)
Requirement already satisfied: colorama in c:\users\mhs\appdata\local\programs\python\python311\Scripts\colorama.exe (from Flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\mhs\appdata\local\programs\python\python311\Scripts\MarkupSafe.exe (from Jinja2>=3.1.2->Flask) (2.1.3)
Using cached flask-3.0.2-py3-none-any.whl (101 kB)
Installing collected packages: Flask
Successfully installed Flask-3.0.2

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Selanjutnya, lakukan pula instalasi Flask-JWT-Extended untuk menggunakan JSON Web Token dalam proses autentikasi dan authorization di aplikasi Flask. Jalankan perintah berikut ini:

```
C:\Windows\system32\CMD.exe X + v
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MHS>pip install flask_jwt_extended
Requirement already satisfied: flask_jwt_extended in c:\users\mhs\appdata\local\
ges (4.6.0)
Requirement already satisfied: Werkzeug>=0.14 in c:\users\mhs\appdata\local\prog
(from flask_jwt_extended) (3.0.0)
Requirement already satisfied: Flask<4.0,>=2.0 in c:\users\mhs\appdata\local\pro
(from flask_jwt_extended) (3.0.2)
Requirement already satisfied: PyJWT<3.0,>=2.0 in c:\users\mhs\appdata\local\pro
(from flask_jwt_extended) (2.8.0)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\mhs\appdata\local\progr
from Flask<4.0,>=2.0->flask_jwt_extended) (3.1.2)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\mhs\appdata\local
ages (from Flask<4.0,>=2.0->flask_jwt_extended) (2.1.2)
Requirement already satisfied: click>=8.1.3 in c:\users\mhs\appdata\local\program
rom Flask<4.0,>=2.0->flask_jwt_extended) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\mhs\appdata\local\prog
(from Flask<4.0,>=2.0->flask_jwt_extended) (1.6.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\mhs\appdata\local\p
es (from Werkzeug>=0.14->flask_jwt_extended) (2.1.3)
Requirement already satisfied: colorama in c:\users\mhs\appdata\local\programs\p
click>=8.1.3->Flask<4.0,>=2.0->flask_jwt_extended) (0.4.6)

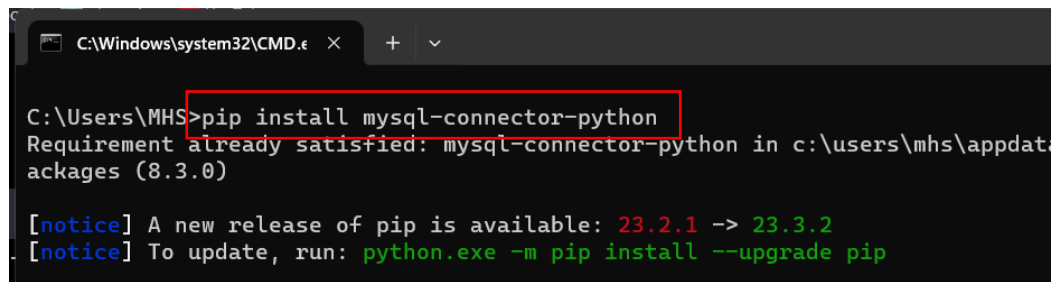
[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

API endpoints umumnya juga digunakan oleh sistem lainnya yang berada di luar lingkup dari API server, sehingga kemampuan dalam menangani CORS harus dimanajemen dengan baik. Oleh karena itu, Instalasi modul Flask-CORS seperti berikut ini:

```
C:\Windows\system32\CMD.exe X + v
C:\Users\MHS>pip install Flask-CORS
Collecting Flask-CORS
  Obtaining dependency information for Flask-CORS from https://files.pythonhosted.
88c32d6258219e9d1ff7c131abf74249ef2031279/Flask_Cors-4.0.0-py2.py3-none-any.whl.me
  Using cached Flask_Cors-4.0.0-py2.py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: Flask>=0.9 in c:\users\mhs\appdata\local\programs\p
m Flask-CORS) (3.0.2)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\mhs\appdata\local\progr
(from Flask>=0.9->Flask-CORS) (3.0.0)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\mhs\appdata\local\program
from Flask>=0.9->Flask-CORS) (3.1.2)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\mhs\appdata\local\p
ages (from Flask>=0.9->Flask-CORS) (2.1.2)
Requirement already satisfied: click>=8.1.3 in c:\users\mhs\appdata\local\programs
rom Flask>=0.9->Flask-CORS) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\mhs\appdata\local\progra
(from Flask>=0.9->Flask-CORS) (1.6.3)
Requirement already satisfied: colorama in c:\users\mhs\appdata\local\programs\pyt
click>=8.1.3->Flask>=0.9->Flask-CORS) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\mhs\appdata\local\progr
(from Jinja2>=3.1.2->Flask>=0.9->Flask-CORS) (2.1.3)
Using cached Flask_Cors-4.0.0-py2.py3-none-any.whl (14 kB)
Installing collected packages: Flask-CORS
Successfully installed Flask-CORS-4.0.0

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Untuk koneksi antara kode program dan DBMS (pada contoh ini adalah MySQL), Instalasi pula mysql-connector-python seperti dibawah ini:



```
C:\Windows\system32\CMD.exe X + v

C:\Users\MHS>pip install mysql-connector-python
Requirement already satisfied: mysql-connector-python in c:\users\mhs\appdata\local\programs\python\python311\python.exe\lib\site-packages (8.3.0)

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Setelah semua modul yang dibutuhkan telah terinstall, pada Workspace dengan nama warungif pada VScode anda, lalu buat file dengan nama **app.py**. Salin kode dibawah ini:

```
from flask import Flask, request, jsonify
from flask_jwt_extended import JWTManager, jwt_required, create_access_token
from flask_cors import CORS
import mysql.connector

app = Flask(__name__)
CORS(app)

app.config['JWT_SECRET_KEY'] = 'Raha$ia1234567890'
jwt = JWTManager(app)

db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'dbwarungif'
}

db_conn = mysql.connector.connect(**db_config)
db_cursor = db_conn.cursor(dictionary=True)

@app.route('/', methods=['GET'])
def home():
    return "Selamat datang di WARUNGIF API"

@app.route('/login', methods=['POST'])
def login():
    username = request.json.get('username', None)
    password = request.json.get('password', None)

    query = "SELECT * FROM users WHERE username = %s AND password = %s"
    db_cursor.execute(query, (username, password))
    user = db_cursor.fetchone()

    if user:
        access_token = create_access_token(identity=username)
        response = {
            "success": "Login berhasil",
            "access_token": access_token
        }
        return jsonify(response), 200
    else:
        return jsonify({"error": "Kredensial tidak valid!"}), 401
```

```

@app.route('/products', methods=['GET'])
def get_all_products():
    query = "SELECT id, nama_barang, harga, stok FROM products"
    db_cursor.execute(query)
    products = db_cursor.fetchall()
    return jsonify(products)

@app.route('/products/<int:product_id>', methods=['GET'])
def get_product(product_id):
    query = "SELECT * FROM products WHERE id = %s"
    db_cursor.execute(query, (product_id,))
    product = db_cursor.fetchone()

    if product:
        return jsonify(product)
    else:
        return jsonify({"error": f"Produk {product_id} tidak ditemukan"}), 404

@app.route('/products', methods=['POST'])
@jwt_required()
def create_product():
    new_product = request.get_json()

    query = "INSERT INTO products (nama_barang, deskripsi, harga, stok) VALUES (%s, %s, %s, %s)"
    db_cursor.execute(query, (new_product['nama_barang'], new_product['deskripsi'], new_product['harga'], new_product['stok']))
    db_conn.commit()

    new_product_id = db_cursor.lastrowid
    new_product['id'] = new_product_id

    db_cursor.execute("SELECT * FROM products WHERE id = %s", (new_product_id,))
    new_product_details = db_cursor.fetchone()

    response_data = {
        "success": f"Produk {new_product_id} berhasil ditambahkan",
        "added_product": new_product_details
    }

    return jsonify(response_data), 201

@app.route('/products/<int:product_id>', methods=['PATCH'])
@jwt_required()
def update_product(product_id):
    updated_data = request.get_json()

    query_check_product = "SELECT * FROM products WHERE id = %s"
    db_cursor.execute(query_check_product, (product_id,))
    existing_product = db_cursor.fetchone()

    if not existing_product:
        return jsonify({"error": f"Produk {product_id} tidak ditemukan"}), 404

    query_update_product = "UPDATE products SET nama_barang = %s, deskripsi = %s, harga = %s, stok = %s WHERE id = %s"
    db_cursor.execute(query_update_product, (updated_data['nama_barang'], updated_data['deskripsi'], updated_data['harga'], updated_data['stok'], product_id))
    db_conn.commit()

    return jsonify({"success": f"Produk {product_id} berhasil diubah"})

```

```

@app.route('/products/<int:product_id>', methods=['DELETE'])
@jwt_required()
def delete_product(product_id):
    query_check_product = "SELECT * FROM products WHERE id = %s"
    db_cursor.execute(query_check_product, (product_id,))
    existing_product = db_cursor.fetchone()

    if not existing_product:
        return jsonify({"error": f"Produk {product_id} tidak ditemukan"}), 404

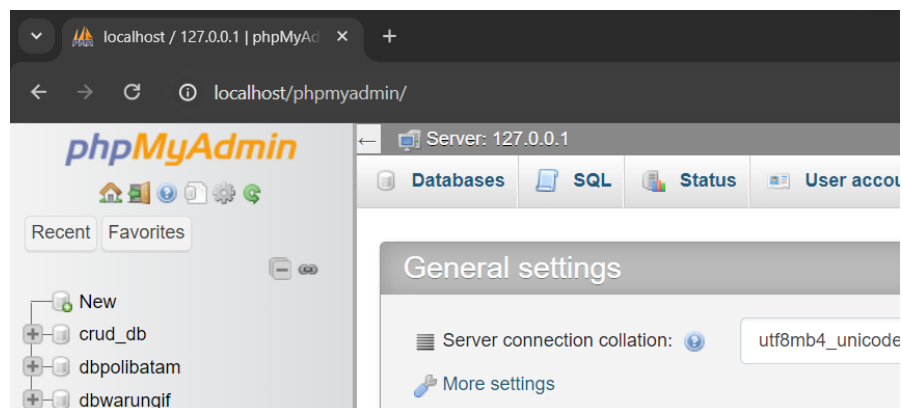
    query_delete_product = "DELETE FROM products WHERE id = %s"
    db_cursor.execute(query_delete_product, (product_id,))
    db_conn.commit()

    return jsonify({"success": f"Produk {product_id} berhasil dihapus"})

if __name__ == '__main__':
    app.run(debug=True)

```

Nyalakan apache dan MySQL pada XAMPP atau Laragon anda, lalu masuk ke localhost/phpmyadmin:



Buat sebuah database baru dengan nama **dbwarungif** lalu buat dua buah tabel dengan spesifikasi sebagai berikut:

dbwarungif products
id : int(11)
nama_barang : varchar(255)
deskripsi : varchar(255)
harga : decimal(10,2)
stok : int(11)

dbwarungif users
id : int(11)
username : varchar(255)
password : varchar(255)

Note: pastikan kolom id adalah primary key dan auto increment

Untuk masing-masing tabel diatas, tambahkan isi record data-data berikut ini:

- **Data users:** (silahkan tambahkan jika perlu)

	id	username	password
<input type="checkbox"/> Edit Copy Delete	1	admin	admin123
<input type="checkbox"/> Edit Copy Delete	2	fulan	fulan123

- **Data products:** (silahkan tambahkan jika perlu)

	id	nama_barang	deskripsi	harga	stok
<input type="checkbox"/> Edit Copy Delete	1	Mouse	Tetikus andalan Informatika	100000.00	10
<input type="checkbox"/> Edit Copy Delete	2	Keyboard	Keyboard murah tapi lucu	200000.00	20

Jalankan app.py diatas dengan perintah:

```
python app.py
```

Percobaan 2: Pengujian API manual dengan menggunakan Curl

ID	Judul	Method & Endpoint	Input (Payload)	Header	Expected Output (Response)	Status
T01	Pengujian dengan <i>valid</i> akun	POST /login	User: admin Password: admin123 {format JSON}	-	Mendapatkan access_token	..
T02	Pengujian dengan <i>invalid</i> akun	POST /login	User: admin Password: salah	-	Error invalid credentials	..

			{format JSON}			
T03	Menampilkan seluruh produk	GET /products	-	-	Seluh data produk yang ada di tabel productst ditampilkan. Kolomnya terdiri dari: id, nama_barang, deskripsi, dan harga	
T04	Melihat detail produk berdasarkan ID	GET /products/1	-	-	Data spesifik dari produk Id 1 ditampilkan. Kolomnya terdiri dari: id, nama_barang, deskripsi, harga, stok	
T04	Menambahkan produk tanpa token user	POST /products	nama_barang: Earphone deskripsi: kanan semua harga: 150000 stok: 14 {format JSON}		Muncul pesan error missing auth header	
T06	Menambahkan produk dengan token user	POST /products	nama_barang: deskripsi: harga: stok: {format JSON}	Authorization: Bearer <<JWT>>	Data produk tersimpan ke database	
T07	Mengubah isi produk berdasarkan ID tanpa token user	PUT /products/1	nama_barang: deskripsi: harga: stok: {format JSON}		Muncul pesan error missing auth header	
T08	Mengubah isi produk berdasarkan ID dengan token user	PUT /products/1	nama_barang: deskripsi: harga: stok: {format JSON}	Authorization: Bearer <<JWT>>	Data produk berhasil diubah sesuai dengan payload	
T09	Menghapus produk tanpa token user	DELETE /products/2	-	-	Muncul pesan error missing auth header	
T10	Menghapus produk dengan token user	DELETE /products/2	-	Authorization: Bearer <<JWT>>	Data produk dengan ID 2 terhapus dari database	

Lengkapi status (Sukses / Gagal) diatas setelah anda menyelesaikan percobaan **curl**. Pada terminal / command prompt / powershell anda masing-masing, jalankan perintah-perintah berikut ini:

T01: Pengujian dengan Akun Valid

```
curl -X POST -H "Content-Type: application/json" -d '{"username\":"admin\","password\":"admin123\"}' http://localhost:5000/login
```

Screenshot hasil pengujian anda dibawah ini:

Simpan isi access_token yang didapatkan dari response diatas karena akan dimanfaatkan pada pengujian yang membutuhkan Bearer Token.

JWT memiliki waktu expired yang dapat dikonfigurasi. Namun secara default memiliki waktu akses yaitu 15 menit.

Bagaimana jika saat pengujian token expired???

Jalankan kembali pengujian ini dan gunakan access_token yang baru hingga waktu expired selanjutnya.

T02: Pengujian dengan Akun Invalid

```
curl -X POST -H "Content-Type: application/json" -d '{"username\":"admin\","password\":"salah\"}' http://localhost:5000/login
```

Screenshot hasil pengujian anda dibawah ini:

T03: Menampilkan Seluruh Produk

```
curl -X GET http://localhost:5000/products
```

Screenshot hasil pengujian anda dibawah ini:

T04: Melihat Detail Produk Berdasarkan ID

```
curl -X GET http://localhost:5000/products/1
```

Screenshot hasil pengujian anda dibawah ini:

T05: Menambahkan Produk Tanpa Token User

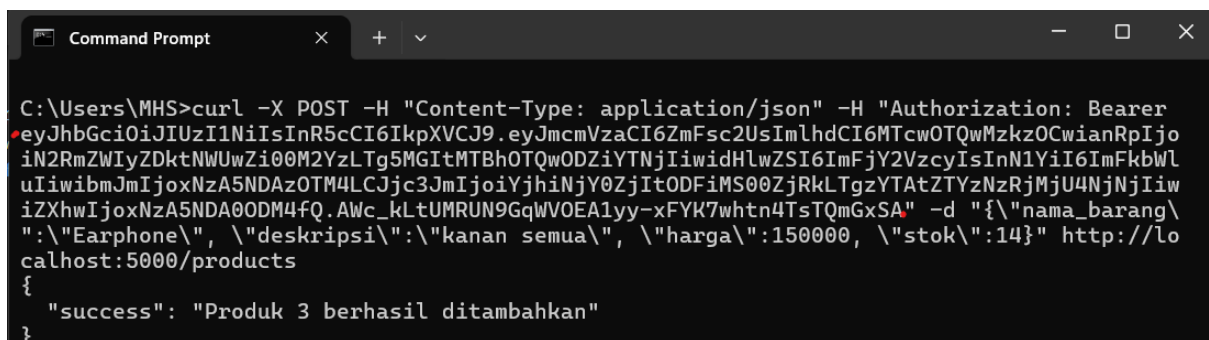
```
curl -X POST -H "Content-Type: application/json" -d '{"nama_barang\":"Earphone\","deskripsi\":"kanan semua\","harga\":"150000","stok\":"14\"}' http://localhost:5000/products
```

Screenshot hasil pengujian anda dibawah ini:

T06: Menambahkan Produk dengan Token User

```
curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer access_token" -d '{"nama_barang":"Earphone", "deskripsi":"kanan semua", "harga":150000, "stok":14}' http://localhost:5000/products
```

Karena JWT yang di-generate tentunya akan berbeda satu sama lain, oleh karena itu silahkan paste JWT yang telah disalin sebelumnya pada percobaan pengujian login. Sebagai contoh, letakkan token yang disimpan dalam pengujian login diantara posisi yang ditandai titik merah dibawah ini:



```
Command Prompt
C:\Users\MHS>curl -X POST -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsImhhdCI6MTcwOTQwMzkzOCwianRpIjoiaN2RmZWlyZDktNWUwZi00M2YzLTg5MGItMTBhOTQwODZiYTJiIiwiaWF0IjY2VzcyIsInN1YiI6ImFkbWluIiwibmJmIjoxNzA5NDZOTM4LCJjc3JmIjoieYjhiNjY0ZjItODFiMS00ZjRkLTgzYTAtZTYzNzRjMjU4NjNjIiwiaXhwIjoxNzA5NDZOTM4fQ.AWc_kLtUMRUN9GqWVOEA1yy-xFYK7whtn4TsTQmGxSA" -d '{"nama_barang":"Earphone", "deskripsi":"kanan semua", "harga":150000, "stok":14}' http://localhost:5000/products
{
  "success": "Produk 3 berhasil ditambahkan"
}
```

Screenshot hasil pengujian anda dibawah ini:

T07: Mengubah Isi Produk Berdasarkan ID Tanpa Token User


```
curl -X PATCH -H "Content-Type: application/json" -d '{"harga":updated_product_price, "stok":updated_product_stock}' http://localhost:5000/products/3
```

Screenshot hasil pengujian anda dibawah ini:

T08: Mengubah Isi Produk Berdasarkan ID dengan Token User

```
curl -X PATCH -H "Content-Type: application/json" -H "Authorization: Bearer  
access_token" -d '{"harga":updated_product_price,  
"stok":updated_product_stock}' http://localhost:5000/products/3
```

Screenshot hasil pengujian anda dibawah ini:



T09: Menghapus Produk Tanpa Token User

```
curl -X DELETE http://localhost:5000/products/3
```

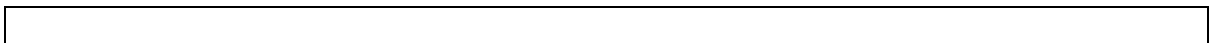
Screenshot hasil pengujian anda dibawah ini:



T10: Menghapus Produk dengan Token User

```
curl -X DELETE \  
-H "Authorization: Bearer  
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InVzZXJ1YWllIiwiaWF0IjoxNjQ2OTUyODAwLCJleHAiOjE2NDY5NTM0MDB9.s1234567890" \  
http://localhost:5000/products/3
```

Screenshot hasil pengujian anda dibawah ini:



- Selamat Mengerjakan -