**Lab10-WebAPI**

**โจทย์: สร้าง Web API จัดการข้อมูลสินค้า (MySQL บน XAMPP)**

**1) เป้าหมาย**

พัฒนา RESTful API สำหรับตาราง appliances รองรับฟังก์ชัน:

* สร้างสินค้า (Create)
* อ่าน/ค้นหา (Read)
* แก้ไข (Update)
* ลบ (Delete)

ใช้ JSON เป็นรูปแบบรับ/ส่งข้อมูลทั้งหมด และรองรับ CORS

**2) โครงสร้างฐานข้อมูล (MySQL)**

**2.1 SQL สร้างฐานข้อมูลและตาราง (ตัวอย่าง)**

CREATE DATABASE IF NOT EXISTS webapi\_demo CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

USE webapi\_demo;

CREATE TABLE IF NOT EXISTS appliances (

id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

sku VARCHAR(32) NOT NULL UNIQUE,

name VARCHAR(150) NOT NULL,

brand VARCHAR(80) NOT NULL,

category VARCHAR(80) NOT NULL, -- เช่น 'ทีวี', 'ตู้เย็น', 'แอร์', 'เครื่องซักผ้า'

price DECIMAL(10,2) NOT NULL CHECK (price >= 0),

stock INT NOT NULL DEFAULT 0 CHECK (stock >= 0),

warranty\_months TINYINT UNSIGNED NOT NULL DEFAULT 12,

energy\_rating TINYINT UNSIGNED, -- 1-5 ดาว (อาจว่างได้)

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- ตัวอย่างข้อมูลเริ่มต้น (อย่างน้อย 10 รายการ)

INSERT INTO appliances (sku, name, brand, category, price, stock, warranty\_months, energy\_rating) VALUES

('TV-32A1', 'ทีวี 32 นิ้ว HD', 'Panaphonic', 'ทีวี', 4990.00, 12, 24, 3),

('TV-55U2', 'ทีวี 55 นิ้ว 4K', 'Sangsung', 'ทีวี', 16990.00, 7, 24, 5),

('FR-250S', 'ตู้เย็น 2 ประตู 250L', 'Hitano', 'ตู้เย็น', 8990.00, 10, 36, 5),

('AC-12000', 'แอร์ 12000 BTU อินเวอร์เตอร์', 'Daika', 'แอร์', 13990.00, 6, 60, 5),

('WM-8KG', 'เครื่องซักผ้า 8 กก.', 'Toshiha', 'เครื่องซักผ้า', 6990.00, 9, 24, 4),

('MW-23L', 'ไมโครเวฟ 23 ลิตร', 'Panaphonic', 'ไมโครเวฟ', 2490.00, 20, 12, 3),

('VA-1000', 'เครื่องดูดฝุ่น 1000W', 'Sangsung', 'เครื่องใช้ในบ้าน', 1590.00, 15, 12, 2),

('IH-2000', 'เตาแม่เหล็กไฟฟ้า 2000W', 'Sharpix', 'เครื่องครัว', 1290.00, 25, 12, 3),

('AR-5L', 'หม้อทอดไร้น้ำมัน 5 ลิตร', 'SmartCook', 'เครื่องครัว', 1790.00, 18, 12, 4),

('FR-180S', 'ตู้เย็น 1 ประตู 180L', 'Toshiha', 'ตู้เย็น', 6490.00, 8, 24, 4);

**3) สเปก API (เส้นทาง + วิธีเรียก)**

ให้วางไฟล์ไว้ภายใต้โฟลเดอร์โปรเจกต์ เช่น:

htdocs/

appliances\_api/

public/

.htaccess

index.php <- front controller/router

src/

Database.php

ApplianceController.php

Response.php

**3.1 Routing (แนะนำให้ใช้ URL สั้นด้วย .htaccess)**

# http://localhost/appliances\_api/public/api/appliances

# http://localhost/appliances\_api/public/api/appliances/3

**ตัวอย่าง .htaccess (ในโฟลเดอร์ public/)**

RewriteEngine On

RewriteBase /appliances\_api/public/

หมายเหตุ: สร้างโฟลเดอร์ public/api/ แล้ววาง index.php ข้างใน (หรือปรับเส้นทางตามถนัด)

**4) รายละเอียด Endpoint**

ทุก response เป็น application/json; charset=utf-8  
ส่งและรับ body เป็น JSON เท่านั้น

**4.1 SELECT (อ่าน/ค้นหา)**

* **GET** /api/appliances

**ตัวอย่าง**  
GET /api/appliances?category=ทีวี&min\_price=4000&max\_price=20000&sort=price\_asc&page=1&per\_page=5

**เรียกใช้สำเร็จ --> 200 OK (ตัวอย่าง)**

{

"data": [

{

"id": 2,

"sku": "TV-55U2",

"name": "ทีวี 55 นิ้ว 4K",

"brand": "Sangsung",

"category": "ทีวี",

"price": 16990.00,

"stock": 7,

"warranty\_months": 24,

"energy\_rating": 5,

"created\_at": "...",

"updated\_at": "..."

}

* **GET** /api/appliances/{id}

**200 OK**

{ "data": { "...เขตข้อมูลสินค้า..." } }

**404 Not Found**

{ "error": "Not found" }

**4.2 INSERT (สร้างใหม่)**

* **POST** /api/appliances

**Request Body (JSON)**

{

"sku": "AC-9000",

"name": "แอร์ 9000 BTU อินเวอร์เตอร์",

"brand": "Daika",

"category": "แอร์",

"price": 11990,

"stock": 5,

"warranty\_months": 48,

"energy\_rating": 5

}

**201 Created**

{ "message": "Created", "data": { "...สินค้าใหม่..." } }

**400 Bad Request** (กรณีข้อมูลไม่ผ่าน validation)

{ "error": "Validation failed", "details": { "price": "must be >= 0" } }

**409 Conflict** (กรณี sku ซ้ำ)

{ "error": "SKU already exists" }

**4.3 UPDATE (แก้ไข)**

* **PUT/PATCH** /api/appliances/{id}

**Request Body (JSON)** — อนุญาตแก้ไขเฉพาะฟิลด์ที่ส่งมา

**200 OK**

{ "message": "Updated", "data": { "...ข้อมูลล่าสุด..." } }

**404 Not Found** | **400 Bad Request** | **409 Conflict (sku ซ้ำ)**

**4.4 DELETE (ลบ)**

* **DELETE** /api/appliances/{id}

**200 OK**

{ "message": "Deleted" }

**404 Not Found**

{ "error": "Not found" }

**5) ข้อกำหนด (ต้องทำ)**

1. ใช้ PHP (Native) + MySQL (XAMPP) \*\* หรือภาษาที่ Xampp รองรับ
2. ใช้ PDO และ Prepared Statements ทุกจุด
3. ส่ง/รับข้อมูลเป็น JSON, ตั้ง header Content-Type: application/json
4. มีโครงสร้างโค้ดแยกชั้น (ไฟล์เชื่อม DB, Controller, Helper สำหรับ response)
5. จัดการ Error/Exception และส่งสถานะ HTTP ที่เหมาะสม (200/201/400/404/409/500)

**Lab10-WebAPI**

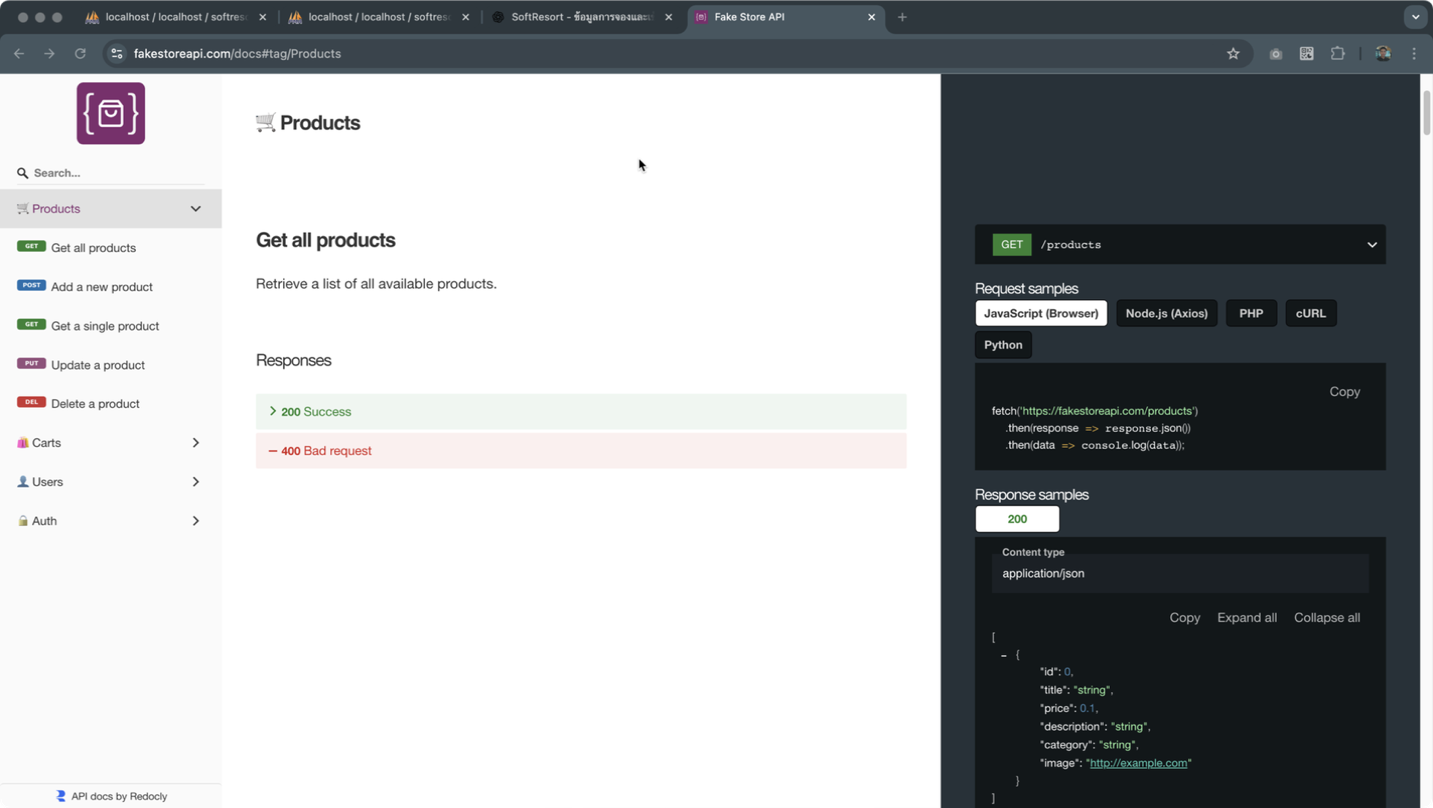
* สร้างฐานข้อมูล MySQL ขึ้นมาใหม่
* สร้างตารางข้อมูลสินค้า (products) เป็นสินค้าอะไรก็ได้ ระบุคอลัมน์ตามรายละเอียดของสินค้าที่เลือกมา
* เพิ่มข้อมูลสินค้าลงในตาราง 20-30 รายการ
* จากนั้นสร้าง web api ด้วยภาษาที่ Xampp รองรับ
  + ที่สามารถเรียกใช้งาน api ได้เหมือนกับตัวอย่าง <https://fakestoreapi.com/docs#tag/Products> ให้ทำงานเฉพาะสินค้า สามารถ ดูข้อมูลทั้งหมด, ดูที่ละรายการ, เพิ่มสินค้า, แก้ไขสินค้า และลบสินค้าได้
  + ข้อมูลอยู่ในรูปแบบ json

**[ งานที่ต้องส่ง ให้เอาขึ้น GitHub ]**

1. ไฟล์ฐานข้อมูล .sql
2. ไฟล์โปรเจ็ก api โค้ด
3. readme : อธิบายการทำงาน และการเรียกใช้งาน api พร้อมภาพประกอบ

**ตัวอย่างเว็บ API**

<https://fakestoreapi.com/docs#tag/Products>



**เว็บ API**

http://localhost/php-products-api/public/api.php

1.**SELECT (อ่าน/ค้นหา)**

โครงสร้าง OOP (index.php→Controller) ไปเป็น **โครงสร้าง Procedural** (api.php) เพื่อแก้ไขปัญหา Routing และปัญหา Output **"1"** ซึ่งตอนนี้ทำงานได้อย่างสมบูรณ์

นี่คือคำอธิบายการทำงานของโค้ด API

### 1. การตั้งค่า Header และการเชื่อมต่อ (Initialization)

* **Header Setup**: โค้ดจะส่ง HTTP Header ไปยัง Client (Postman) เป็นอันดับแรกสุด เพื่อบอกว่า Response จะเป็น JSON (Content-Type: application/json) และอนุญาตให้ Client อื่น ๆ สามารถเข้าถึงได้ (CORS)
* **Database Connection**: โค้ดจะสร้างการเชื่อมต่อกับฐานข้อมูล **products\_db** ทันที โดยใช้ค่าที่กำหนดไว้ (localhost, root, "", products\_db) ถ้าเชื่อมต่อสำเร็จ จะได้ Object การเชื่อมต่อ ($conn) สำหรับใช้ในการทำ Query
* **การแก้ปัญหา "1"**: ในโค้ด Procedural ที่ทำงานได้ การเชื่อมต่อที่สำเร็จจะไม่มีการ echo ค่า true (เลข **1**) ออกมาอีกแล้ว ทำให้ JSON Header ไม่ถูกขัดขวาง

### 2. การจัดการ Request Method (Routing Logic)

โค้ดจะตรวจสอบ HTTP Method ที่ Postman ส่งมา (GET, POST, PUT, DELETE) และใช้คำสั่ง switch เพื่อเลือก Logic ที่จะรัน

### 3. การทำงานของ GET All (Read All)

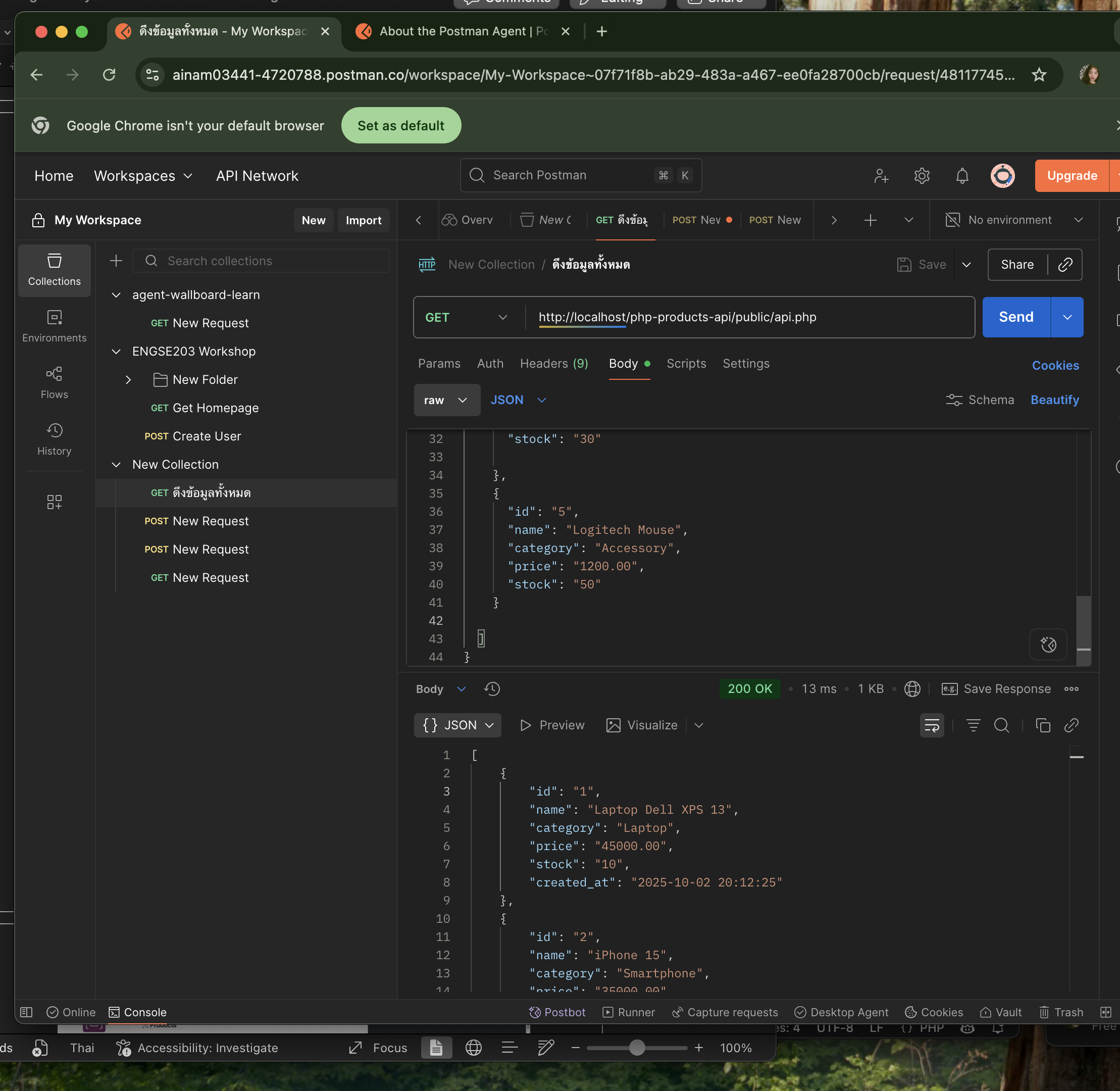
นี่คือส่วนที่ทำให้คุณเห็นข้อมูลทั้งหมดใน Postman

* **Condition**: เมื่อ Method เป็น GET และไม่มี ID ถูกส่งมาใน URL (!isset($GET[′id′]))
* **SQL Query**: PHP จะรันคำสั่ง SQL **SELECT \* FROM products** เพื่อดึงข้อมูลสินค้าทั้งหมด 5 รายการ
* **Data Aggregation**: โค้ดจะใช้ loop เพื่อดึงข้อมูลแต่ละแถว ($row=$result→fetch\_assoc())) และเก็บไว้ใน PHP Array ($data[]).
* **Response**: PHP จะตั้งค่า Status **200 OK** และใช้ echo json\_encode($data) เพื่อแปลง PHP Array ทั้งหมดนั้นให้เป็น JSON Array และส่งกลับไปยัง Postman

### 4. การทำงานของ CRUD ที่เหลือ

โค้ดของคุณยังมี Logic สำหรับการจัดการข้อมูลอื่น ๆ:

* **GET Single**: หากมี ID ใน URL (?id=...) จะรัน Query **SELECT \* FROM products WHERE id=$id** และส่งข้อมูลรายการเดียวกลับไป
* **POST (Create)**: รับข้อมูล JSON Body จาก Postman และรันคำสั่ง **INSERT INTO products** เพื่อเพิ่มสินค้าใหม่
* **PUT (Update)**: รับข้อมูล JSON Body ที่มี ID และรันคำสั่ง **UPDATE products** เพื่อแก้ไขข้อมูลสินค้าที่มีอยู่
* **DELETE**: รับ ID จาก URL (?id=...) และรันคำสั่ง **DELETE FROM products** เพื่อลบสินค้า
* **Cleanup**: ท้ายที่สุด คำสั่ง $conn→close() จะถูกเรียกใช้เพื่อปิดการเชื่อมต่อฐานข้อมูลอย่างเป็นระเบียบ



2. **INSERT (สร้างใหม่)**

### 1. โครงสร้างและการจัดการ Path (Routing)

* **ปัญหา 404 (Not Found)**: ในช่วงแรกเมื่อคุณพยายามใช้ URL ที่สวยงาม (.../public/api/products) คุณพบ Error 404 ซึ่งเกิดจาก Apache Web Server ไม่ได้ประมวลผลไฟล์ **.htaccess** ที่อยู่ในโฟลเดอร์ public/
* **ทางแก้ (Working URL)**: เพื่อให้โค้ด PHP รันได้ คุณจึงเปลี่ยนไปใช้ URL ที่ชี้ตรงไปยังไฟล์ PHP หลัก (.../public/index.php/api/...) หรือไฟล์ .../public/api.php ซึ่งเป็นการข้ามปัญหา .htaccess ทำให้ PHP สามารถเริ่มประมวลผลได้

### 2. ปัญหา Output ขัดขวาง (Body เป็น "1")

นี่คือปัญหาที่ยากที่สุดที่คุณแก้ไขได้สำเร็จ

* **อาการ**: เมื่อคุณส่ง Request GET ไปยัง API Status เป็น 200 OK แต่ Body กลับแสดงเป็น HTML เพียงเลข **"1"**
* **สาเหตุ**: ในโค้ด PHP แบบ Procedural ที่คุณเคยใช้อยู่ (เช่น ในไฟล์ db.php ที่ถูก include หรือ require) เมื่อการเชื่อมต่อฐานข้อมูลสำเร็จ จะมีค่า true กลับมา ซึ่ง PHP จะตีความเป็นเลข **"1"** และพิมพ์ออกมาเป็น Output ทันที
* **ผลกระทบ**: เมื่อ Output เกิดขึ้นแล้ว PHP จะไม่สามารถส่ง HTTP Header ใหม่ (Content-Type: application/json) ได้อีก ทำให้ Postman แสดง Body เป็น **HTML**

### 3. การทำงานของ GET All (Read All)

หลังจากแก้ไขปัญหา Output แล้ว API จึงทำงานตาม Logic ที่เขียนไว้:

1. **การรับ Request**: Postman ส่ง Method GET มาที่ URL ที่ทำงานได้
2. **การเชื่อมต่อ**: โค้ดจะสร้างการเชื่อมต่อ MySQL กับฐานข้อมูล **products\_db** ทันที
3. **การดึงข้อมูล**: โค้ดจะรัน SQL **SELECT \* FROM products** เพื่อดึงข้อมูลสินค้าทั้งหมด 5 รายการ
4. **การประมวลผล**: โค้ดจะวน Loop ดึงข้อมูลแต่ละแถวและนำไปสร้าง PHP Array
5. **การตอบกลับ**: โค้ดจะตั้งค่า Status **200 OK** และใช้ echo json\_encode() เพื่อแปลง PHP Array นั้นให้เป็น JSON Array และส่งกลับไปยัง Postman
6. **ผลลัพธ์**: Postman แสดง JSON Array ที่มีข้อมูลสินค้าครบถ้วน

A screenshot of a computer

AI-generated content may be incorrect.

3. **UPDATE (แก้ไข)**

### 1.การทำงานของ GET All (Read All)

นี่คือ Logic ที่ทำงานได้สำเร็จและทำให้คุณเห็นข้อมูลทั้งหมดใน Postman

1. **การรับ Request**: โค้ดจะตรวจสอบว่า HTTP Method คือ GET และไม่มี ID ถูกส่งมาใน URL
2. **การดึงข้อมูล**: โค้ดจะรัน SQL **SELECT \* FROM products** เพื่อดึงข้อมูลสินค้าทั้งหมด 5 รายการจากตาราง products ในฐานข้อมูล products\_db
3. **การตอบกลับ**: ข้อมูลที่ได้จะถูกนำมาสร้างเป็น PHP Array และถูกแปลงเป็น **JSON Array** ด้วย json\_encode() พร้อม Status **200 OK**
4. **ผลลัพธ์**: Postman แสดง JSON ที่มีสินค้าครบถ้วนอย่างถูกต้อง

A screenshot of a computer

AI-generated content may be incorrect.

**4.DELETE (ลบ)**

### การทำงานของ CRUD ที่เหลือ

โค้ด Procedural ที่สมบูรณ์ของคุณยังมี Logic สำหรับการจัดการข้อมูลอื่น ๆ:

* **GET Single**: ตรวจสอบ ID ใน URL (?id=...) และรัน Query **SELECT \* FROM products WHERE id=$id**
* **POST (Create)**: รับ JSON Body จาก Postman และรันคำสั่ง **INSERT INTO products**
* **PUT (Update)**: รับ JSON Body ที่มี ID และรันคำสั่ง **UPDATE products**
* **DELETE**: รับ ID จาก URL และรันคำสั่ง **DELETE FROM products**

A screenshot of a computer

AI-generated content may be incorrect.