



รายงาน

เรื่อง การออกแบบเว็บแอปพลิเคชัน และพัฒนาระบบธุรกิจ Puma

เสนอ

ผู้ช่วยศาสตราจารย์ ดร. จิตภา ไกรสังข์

อาจารย์ ดร. วุฒิชชาติ แสงผล

จัดทำโดย

นาย นวตล สมบูรณ์กุล	รหัสนักศึกษา 6687026
นาย วัฒนชัย บุญไชย	รหัสนักศึกษา 6687045
นาย สิทธา สีลาเขตต์	รหัสนักศึกษา 6687054
นาย กิตติคุณ พวงสุวรรณ	รหัสนักศึกษา 6687059

รายงานนี้เป็นส่วนหนึ่งของรายวิชาเทคโนโลยีด้านเว็บและการประยุกต์ใช้ (ทสวด 241 และ ทสวด 242)

มหาวิทยาลัยมหิดล

ภาคเรียนที่ 1 ปีการศึกษา 2567

1.ประวัติและความเป็นมาของธุรกิจ

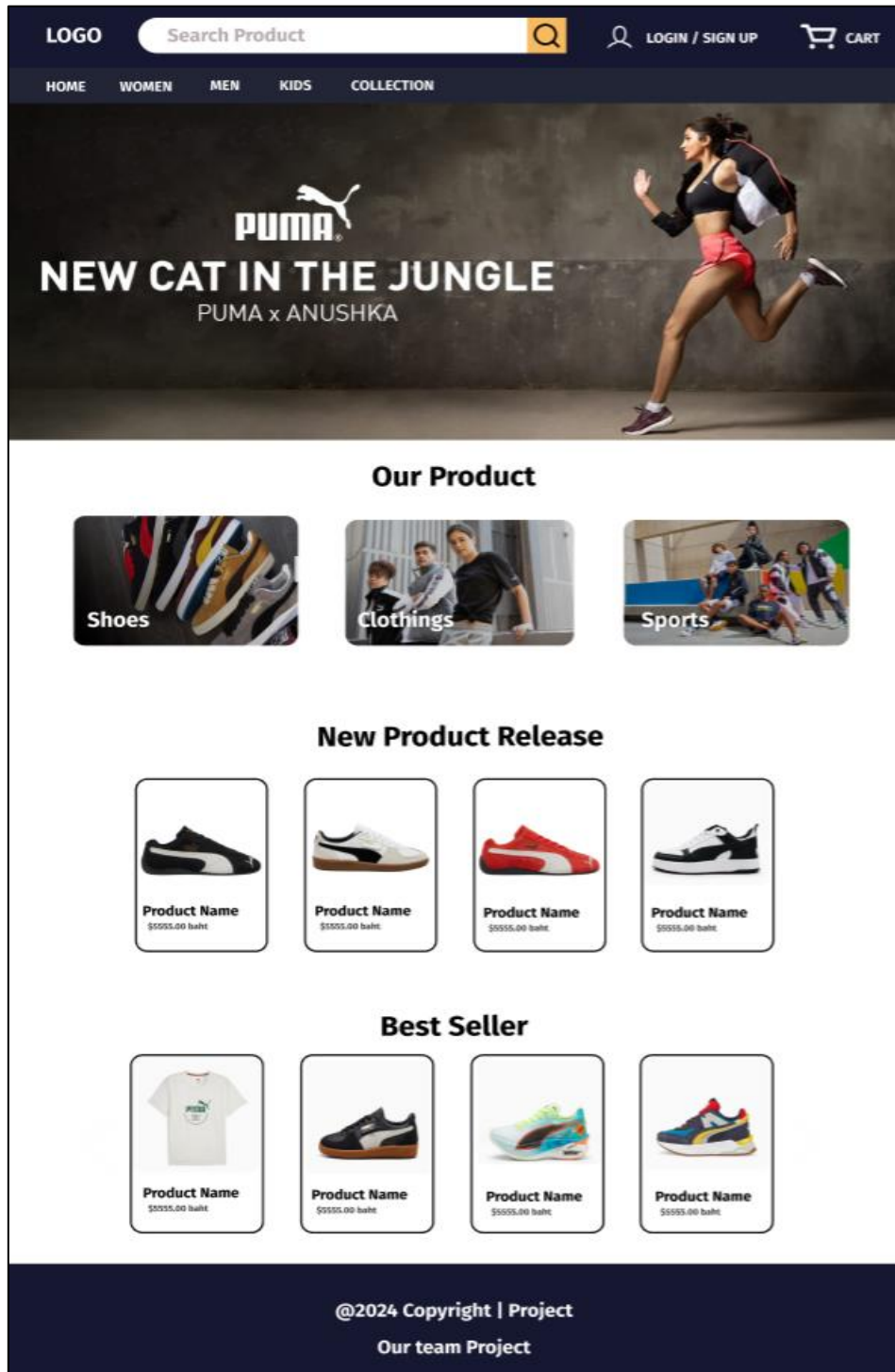
บริษัทพูมา (Puma) เริ่มต้นจากธุรกิจครอบครัวโดยสองพี่น้องตระกูลแดสเลอร์ คือ รูดอล์ฟ แดสเลอร์ และ อ돌ฟ์ แดสเลอร์ โดยในครั้งแรกได้จดทะเบียนบริษัทในชื่อ อาดิดาส (Adidas) แต่เนื่องจากความขัดแย้งในครอบครัวและผลประโยชน์ทางธุรกิจ ทำให้รูดอล์ฟ แดสเลอร์แยกตัวออกมาก่อตั้งบริษัทใหม่ในปี ค.ศ. 1948 ภายใต้ชื่อ "พูมา" (Puma) โดยใช้เงาของเสือพูมาเป็นสัญลักษณ์หลักของบริษัท

หลังจากก่อตั้งขึ้นไม่นาน พูมาได้รับความนิยมอย่างมาก โดยเฉพาะในช่วงปี ค.ศ. 1970 จากการที่ได้รับการสนับสนุนจากนักกีฬาและนักออกแบบรองเท้าชื่อดัง ทำให้พูมาเป็นที่รู้จักอย่างแพร่หลายในวงการกีฬาทั่วโลกปัจจุบัน พูมาได้ขยายธุรกิจของตนไปทั่วโลก ไม่เพียงแต่การเปิดสาขาในห้างสรรพสินค้าเท่านั้น แต่ยังมีการจำหน่ายสินค้าผ่านช่องทางออนไลน์หรือระบบอีคอมเมิร์ซ (E-Commerce) ทำให้ลูกค้าทั่วโลกสามารถเข้าถึงสินค้าและบริการของพูมาได้อย่างสะดวก

2. แบบจำลองหน้าเว็บแอปพลิเคชัน

2.1 Homepage

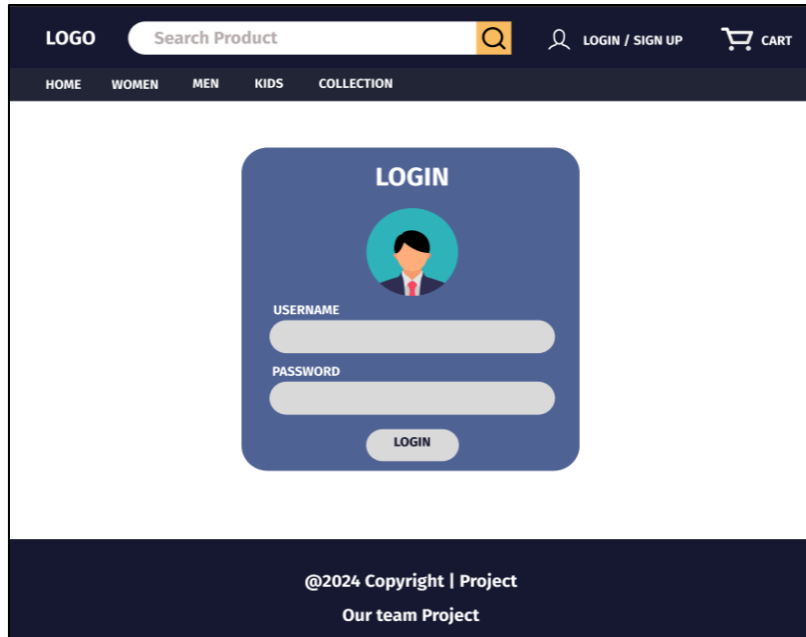
ในหน้าของ Homepage มีปุ่ม Login Page เพื่อให้เชื่อมไปหน้า Login สำหรับ user และ administrator และมี Navbar ที่จะใช้แสดงประเภทสินค้าต่างๆในหน้าเว็บไซต์ และมีการแสดงสินค้าต่างๆบางส่วนในหน้าหลักของเว็บไซต์



รูปแสดงหน้า Homepage

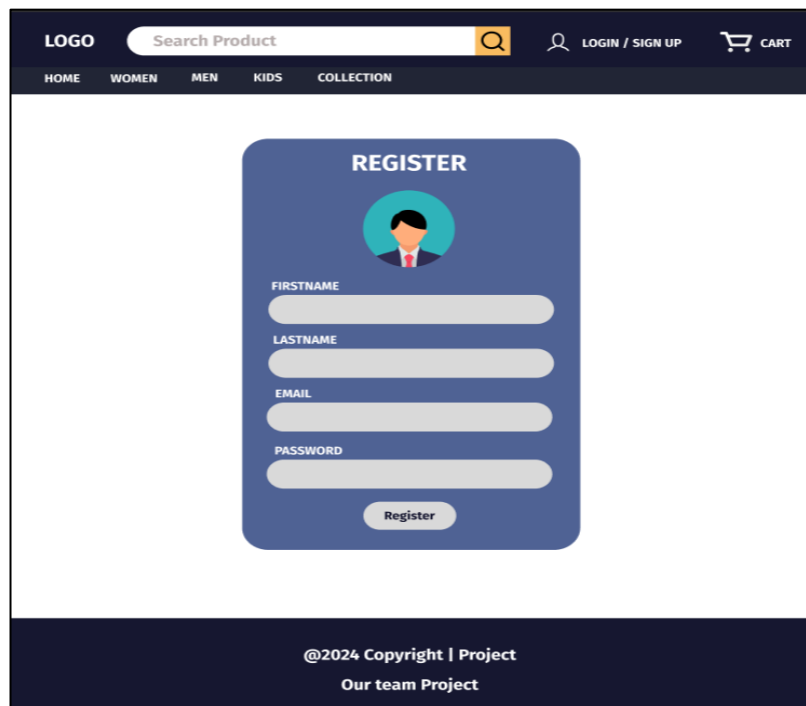
2.2 Login Page & Register Page

หน้าของ Login สามารถเข้าถึงได้โดยการกดปุ่ม Login บน Header โดยในหน้า Login นั้น ผู้ใช้ระบบจะต้องทำการกรอกข้อมูล Username และ Password ให้ถูกต้อง และในหน้าของ Register ผู้ใช้สามารถกรอกข้อมูลตามที่ปรากฏเพื่อลงทะเบียนเข้าใช้ระบบ



The screenshot shows the Login page of an e-commerce application. At the top, there is a dark blue header with a 'LOGO' on the left, a 'Search Product' bar in the center, and a user icon with 'LOGIN / SIGN UP' and a shopping cart icon on the right. Below the header is a navigation bar with links for 'HOME', 'WOMEN', 'MEN', 'KIDS', and 'COLLECTION'. The main content area features a blue login card with a user icon, the title 'LOGIN', and two input fields labeled 'USERNAME' and 'PASSWORD'. A 'LOGIN' button is at the bottom of the card. The footer is dark blue with the text '@2024 Copyright | Project' and 'Our team Project'.

รูปแสดงหน้า Login Page



The screenshot shows the Register page of the same e-commerce application. The header and navigation bar are identical to the Login page. The main content area features a blue register card with a user icon, the title 'REGISTER', and four input fields labeled 'FIRSTNAME', 'LASTNAME', 'EMAIL', and 'PASSWORD'. A 'Register' button is at the bottom of the card. The footer is dark blue with the text '@2024 Copyright | Project' and 'Our team Project'.

รูปแสดงหน้า Register Page

2.3 Search Page

หน้า Searching Page จะทำหน้าที่ในการรับ Input จากผู้ใช้ในการค้นหาสินค้าตามที่ใช้
นั้นได้ระบุลงใน Form ต่างๆ

LOGO Search Product LOGIN / SIGN UP CART

HOME WOMEN MEN KIDS COLLECTION

Product Name Value x

Product Type ☐ male ☐ female ☐ unisex

Gender ☐ male ☐ female ☐ unisex

Sport ☐ football ☐ basketball ☐ running ☐ training ☐ outdoor

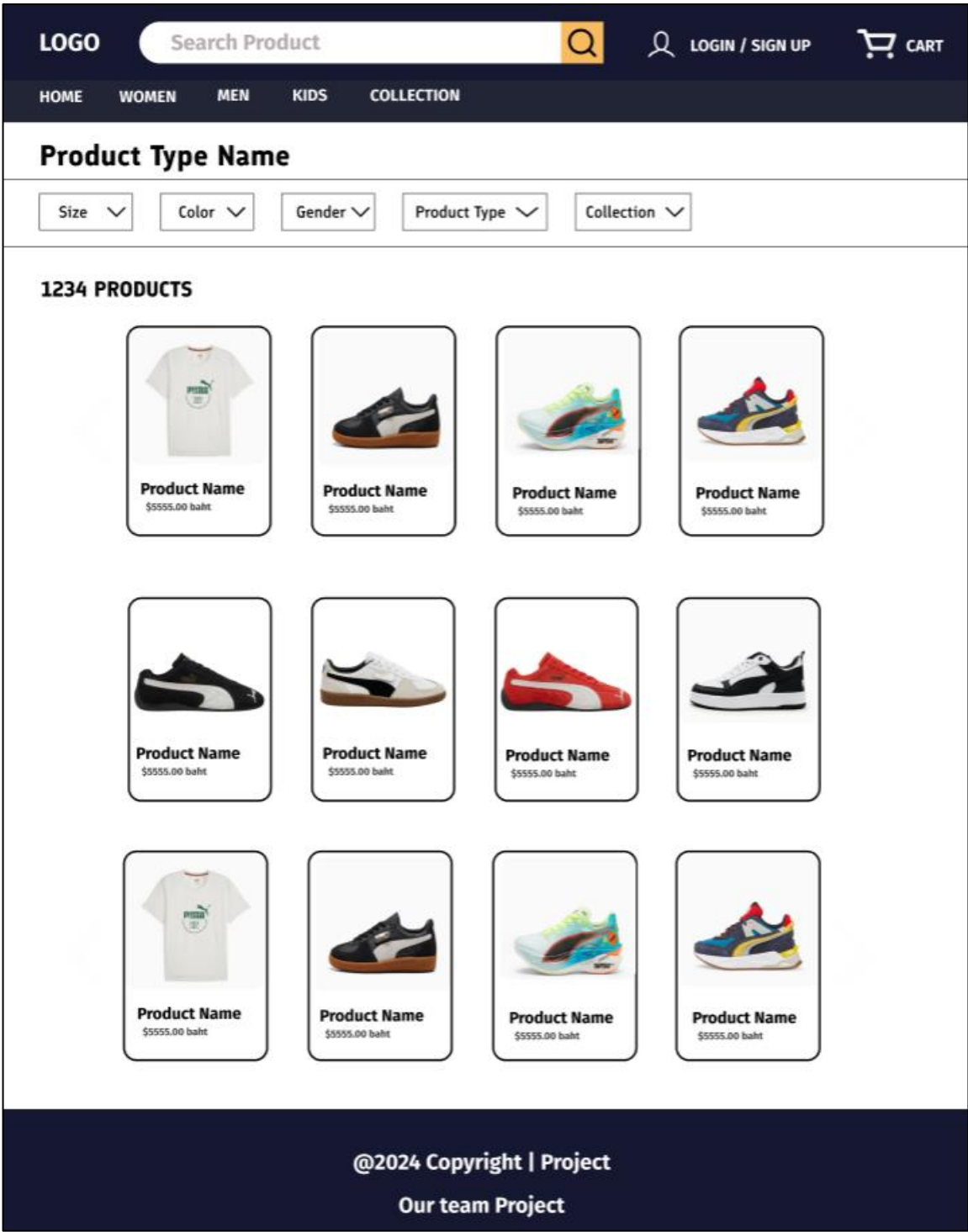
Collection ☐ AF#1 ☐ AF#2 ☐ AF#3 ☐ AF#4 ☐ AF#5 ☐ AF#6

Value

Search

@2024 Copyright | Project
Our team Project

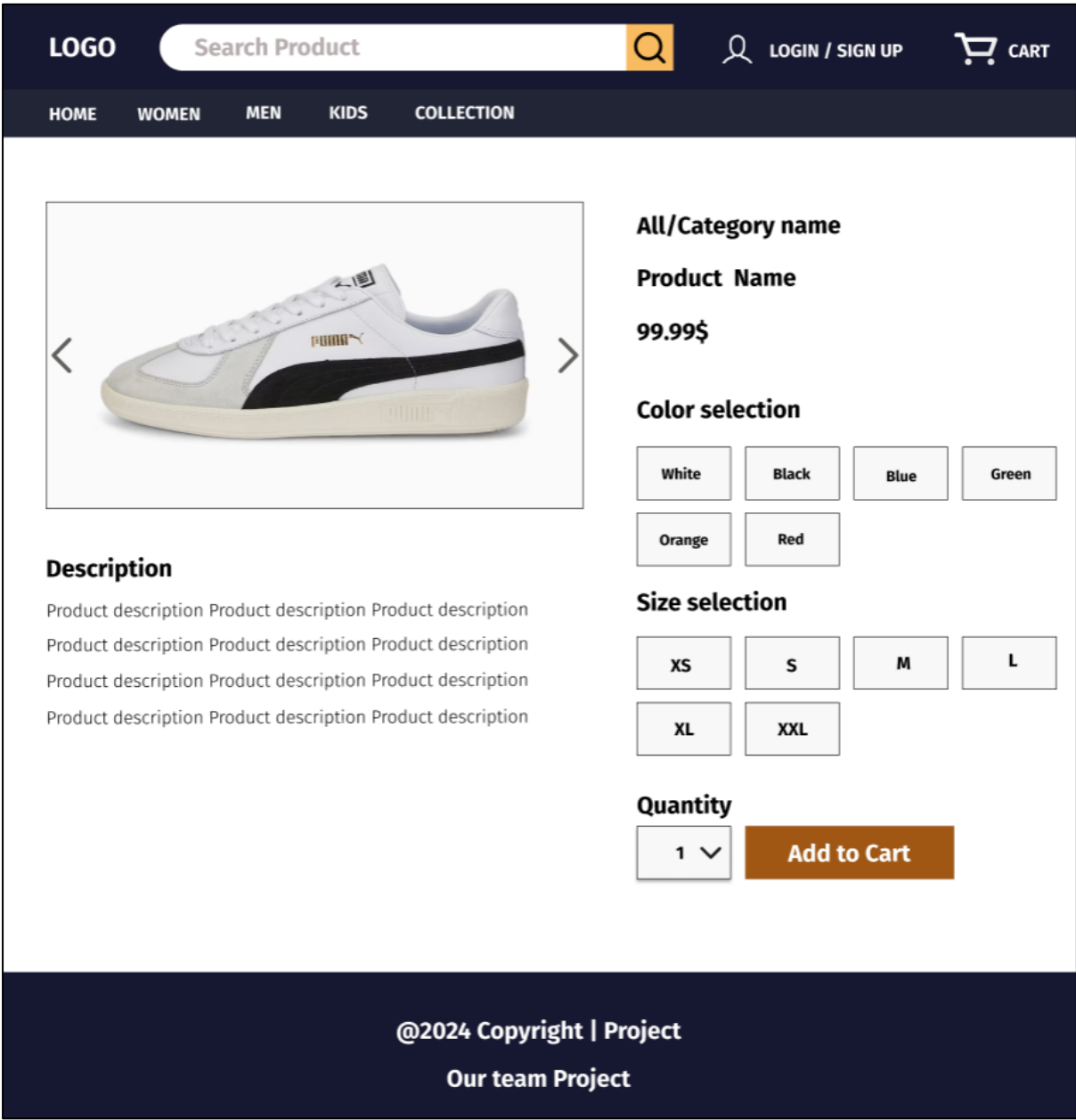
รูปแสดงหน้า Search Page



รูปแสดงหน้า Result of Search Page

2.4 Detail Product Page

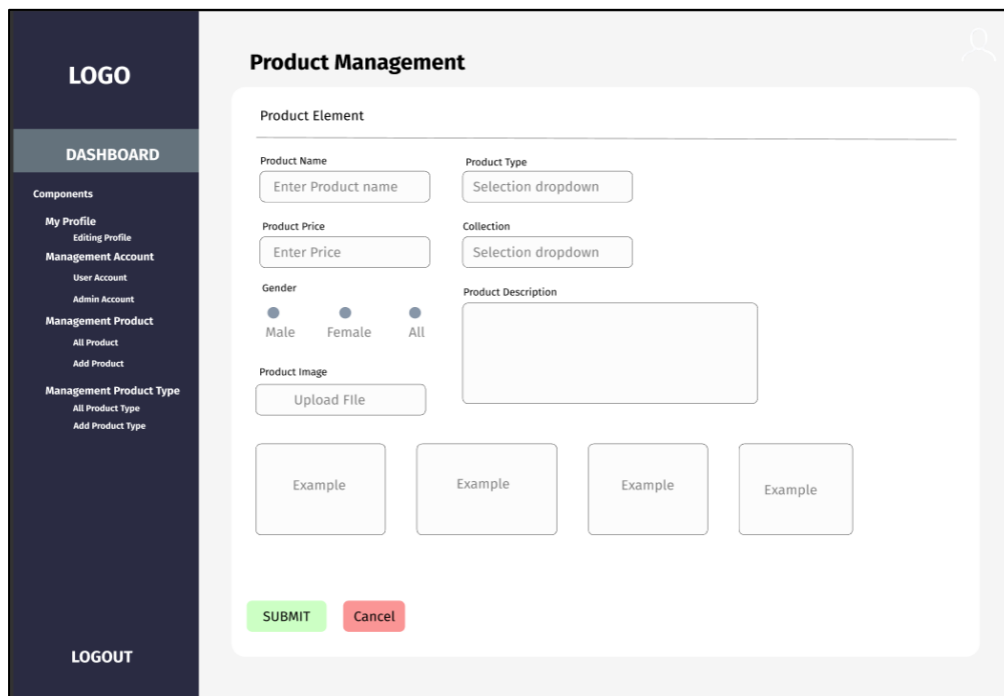
หน้าของ Detail จะมีการแสดงชื่อ รูป ราคา และคำอธิบายของสินค้าให้ผู้ใช้งานสามารถ เลือกขนาด สี และจำนวนสินค้า



รูปแสดงหน้า Detail Product Page

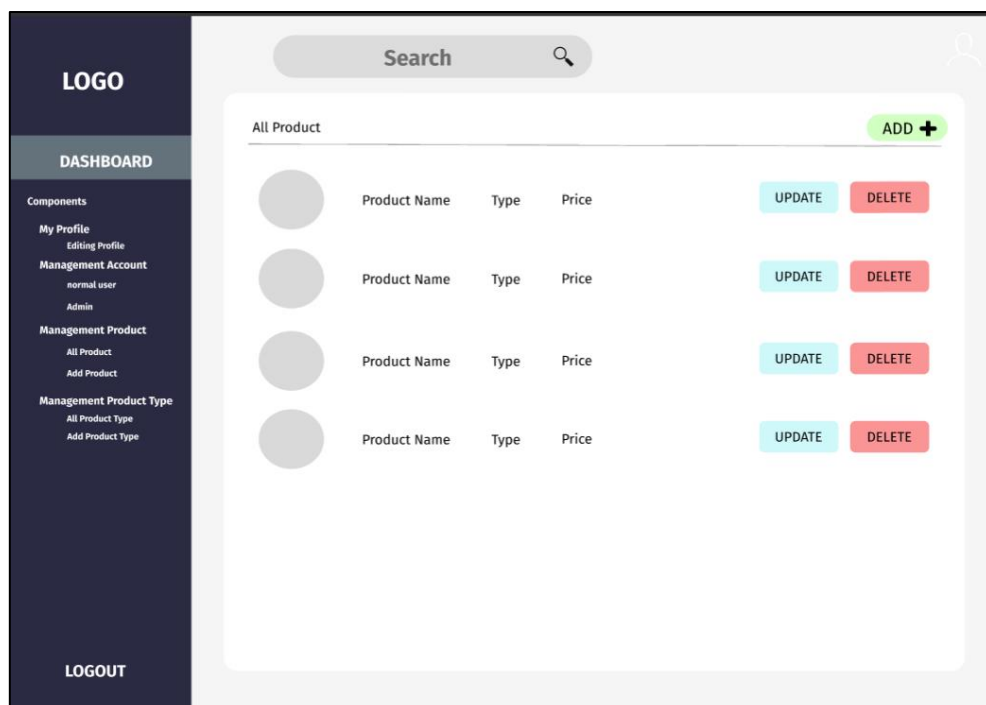
2.5 Product / Service Management Page

หน้าสำหรับจัดการสินค้าโดยที่มีทั้งหน้าแสดงสินค้าทั้งหมด และเพิ่มสินค้า โดยการใส่ข้อมูล และรูปและประเภทสินค้า ราคา



The image shows a 'Product Management' form within a dashboard. The dashboard has a dark sidebar with a 'LOGO' at the top, a 'DASHBOARD' section, and a list of components including 'My Profile', 'Management Account', 'Management Product', and 'Management Product Type'. The main content area is titled 'Product Management' and contains a 'Product Element' form. The form has several input fields: 'Product Name' (text input), 'Product Type' (selection dropdown), 'Product Price' (text input), 'Collection' (selection dropdown), 'Gender' (radio buttons for Male, Female, All), 'Product Description' (text area), and 'Product Image' (upload file button). Below the form are four placeholder boxes labeled 'Example'. At the bottom of the form are 'SUBMIT' and 'Cancel' buttons.

รูปแสดงหน้าเพิ่มสินค้า(Add Product)



The image shows a table titled 'All Product' within the same dashboard. The table has columns for 'Product Name', 'Type', 'Price', 'UPDATE', and 'DELETE'. There are four rows of product data. The 'UPDATE' buttons are cyan and the 'DELETE' buttons are red. A green 'ADD +' button is located at the top right of the table.

	Product Name	Type	Price	UPDATE	DELETE
	Product Name	Type	Price	UPDATE	DELETE
	Product Name	Type	Price	UPDATE	DELETE
	Product Name	Type	Price	UPDATE	DELETE
	Product Name	Type	Price	UPDATE	DELETE

รูปแสดงหน้าแสดงสินค้า(All Product)

2.6 User Account Management Page

หน้าสำหรับการบริหารจัดการบัญชีผู้ใช้ของผู้ดูแลระบบ (administrators)
โดยผู้ดูแลระบบสามารถค้นหาบัญชีผู้ใช้ เพิ่มบัญชีผู้ใช้ แก้ไขบัญชีผู้ใช้ และลบข้อมูลบัญชีผู้ใช้

Account Management

ADD USER

USERNAME:

PASSWORD:


FIRSTNAME:

LASTNAME:

Gender: ☒ Male ☐ Female ☐ All

TEL-NUMBER:

ADDRESS:

PROFILE PICTURE: 

รูปแสดงหน้าเพิ่มข้อมูลบัญชีผู้ใช้ (Add User Page)

Search

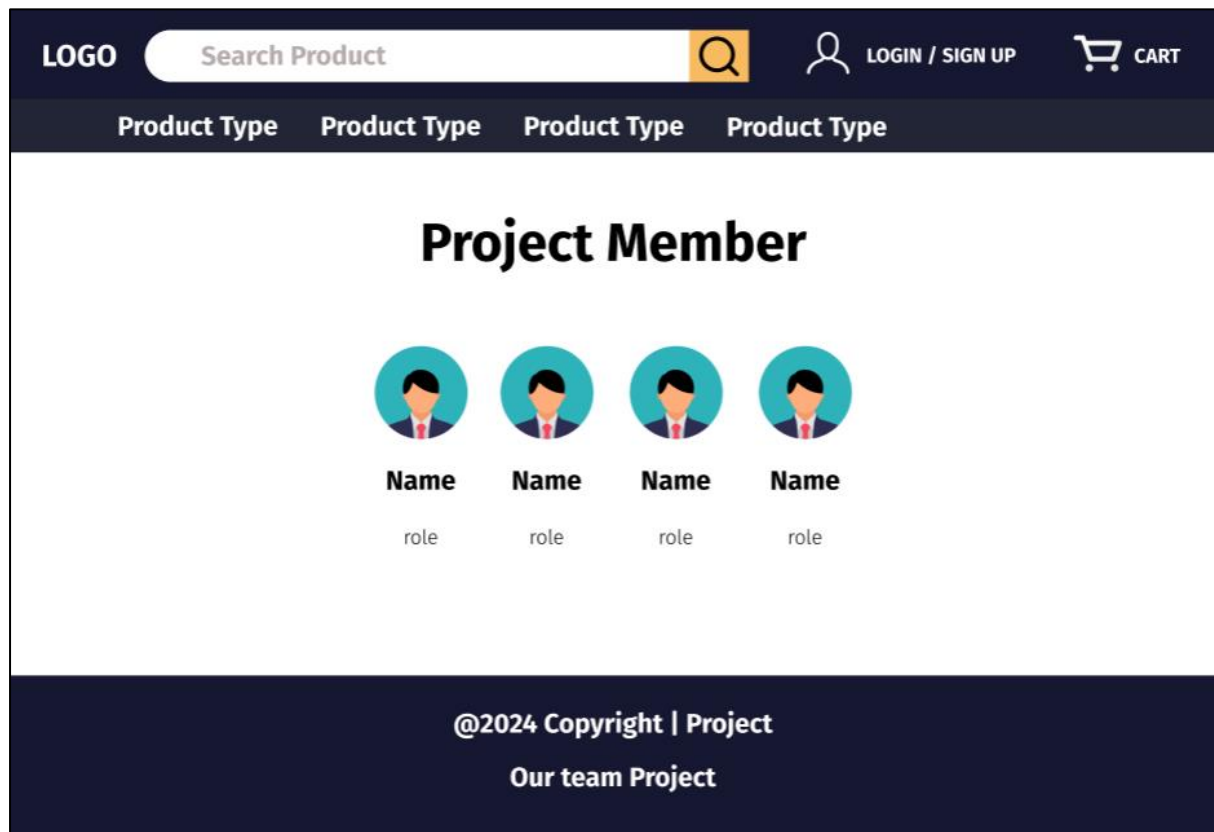
USER

username	active		
USER_1	in 13 hours	<input type="button" value="UPDATE"/>	<input type="button" value="DELETE"/>
USER_2	in 18 hours	<input type="button" value="UPDATE"/>	<input type="button" value="DELETE"/>

รูปแสดงหน้าจัดการบัญชีผู้ใช้ (User Account Management Page)

2.7 Team Page

หน้านี้เป็นหน้าสำหรับแสดงข้อมูลของคนในทีมทั้งหมดรวมไปถึงหน้าที่ต่างๆของคนในทีม

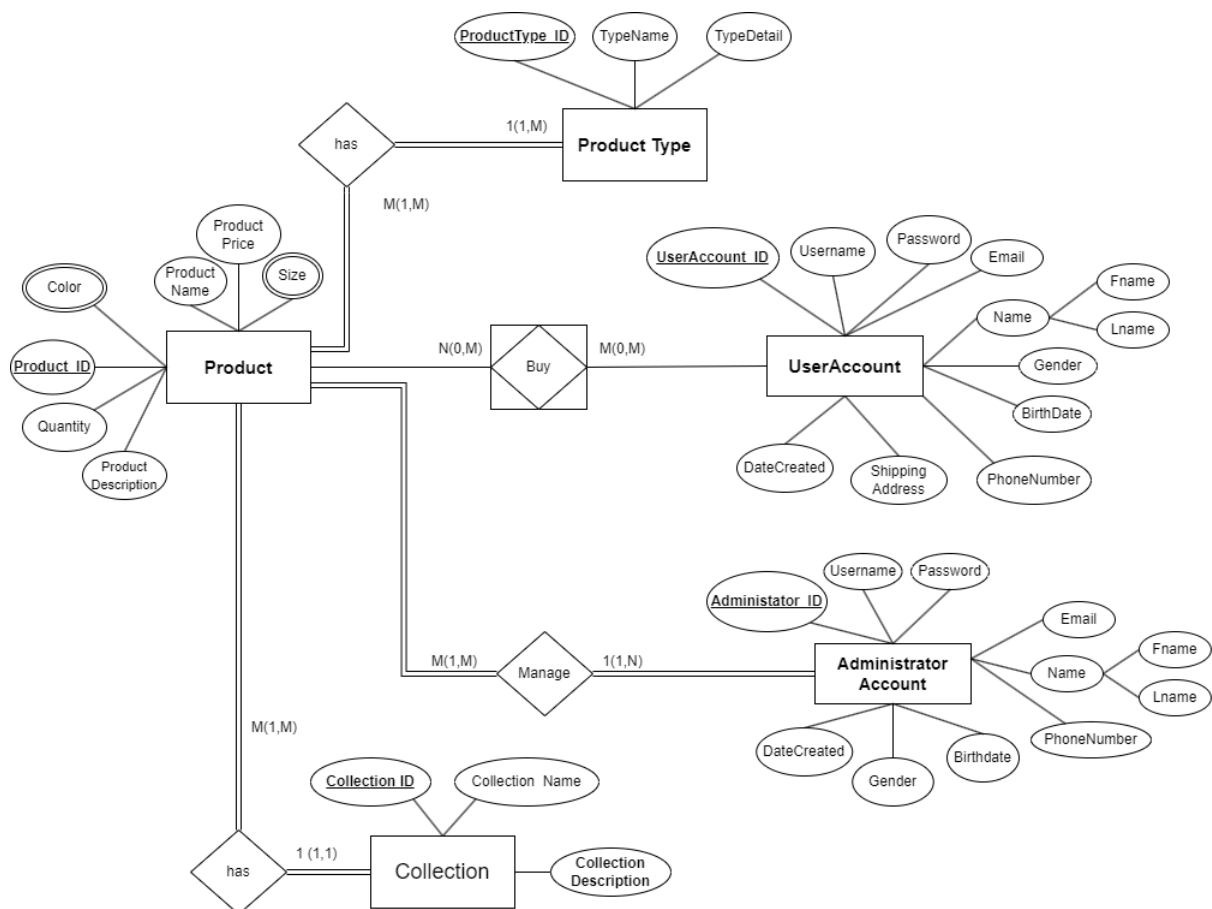


รูปแสดงหน้าสมาชิกกลุ่ม (Team page)

3.แบบจำลองข้อมูลหรือ Data Model

มี Entity หลักทั้งหมด 5 Entity ได้แก่

1. Product เก็บข้อมูลรายละเอียดต่างๆเกี่ยวกับสินค้าและรายละเอียดสินค้า
ได้แก่ รหัสสินค้า ชื่อสินค้า ขนาด สี ราคา รายละเอียดสินค้า และจำนวนสินค้า
2. Product Type เก็บข้อมูลรายละเอียดต่างๆของประเภทสินค้า
ได้แก่ รหัสประเภทสินค้า ชื่อประเภทสินค้า รายละเอียดประเภทสินค้า
3. Collection เก็บข้อมูลรายละเอียดของชุดคอลเล็กชันต่างๆ
ได้แก่ คอลเล็กชันไอดี ชื่อคอลเล็กชัน รายละเอียดคอลเล็กชัน
4. User Account เก็บข้อมูลของผู้ใช้ที่ได้ทำการสมัครสมาชิกเว็บไซต์
ได้แก่ ไอดีบัญชีผู้ใช้ ยูเซอร์เนม รหัสผ่าน อีเมล ชื่อจริงนามสกุล เพศ วันเกิด เบอร์โทรศัพท์
ที่อยู่สำหรับจัดส่งสินค้า วันที่สร้างบัญชี
5. Administrator Account เก็บข้อมูลของผู้ดูแลระบบ
ได้แก่ ไอดีผู้ดูแลระบบ ยูเซอร์เนม รหัสผ่าน อีเมล ชื่อนามสกุล เพศ วันเกิด เบอร์โทรศัพท์
วันที่สร้างบัญชี



4.Data dictionary

User Account

Table Name	Attribute Name	Contents	Type	Format	Nullable	Range	Key	FK referenced Table
UserAccount	UserAccount_ID	User ID	int	UIDxxxxxx			PK	
	Fname	firstname of user	varchar(50)	xxxxxx				
	Lname	lastname of user	varchar(50)	xxxxxx				
	BirthDate	birthday of user	date	yyyy-mm-dd				
	PhoneNumber	phone number of user	varchar(10)	xxxxxxxxxx				
	Gender	gender of user	varchar(10)	xxxxxxxxxx				
	DateCreated	created account day	date	yyyy-mm-dd				
	ShippingAddress	address of user	varchar(50)	xxxxxx				
	Username	Username for login website	varchar(16)	xxxxxx				
	Password	Password for login website	varchar(16)	xxxxxxxxxx				
	Email	email of user	varchar(50)	xxx@xxx.xxx				

Administrator Account

Table Name	Attribute Name	Contents	Type	Format	Nullable	Range	Key	FK referenced Table
Administrator Account	Administrator_ID	User ID	int	ADIDxxxxxx			PK	
	Fname	firstname of Administrator	varchar(50)	xxxxxx				
	BirthDate	birthday of Administrator	date	yyyy-mm-dd				
	Lname	lastname of Administrator	varchar(50)	xxxxxx				
	Gender	gender of Administrator	varchar(10)	xxxxxxxxxx				
	PhoneNumber	phone number of Administrator	varchar(10)	xxxxxxxxxx				
	Email	Email of Administrator	varchar(50)	xxx@xxx.xxx				
	DateCreated	created account day	date	yyyy-mm-dd				
	Username	Username for login website	varchar(16)	xxxxxxxxxx				
	Password	Password for login website	varchar(16)	xxxxxxxxxx				

Collection

Table Name	Attribute Name	Contents	Type	Format	Nullable	Range	Key	FK Referenced Table
Collection	Collection_ID	Collection ID	varchar(20)	COLLxxxxxxx			PK	
	COLL_Name	Collection name	varchar(30)	Xxxxxxxx				
	COLL_Detail	Detail of Collection	varchar(225)	XXXXXXXXXXXXXX				

Product type

Table Name	Attribute Name	Contents	Type	Format	Nullable	Range	Key	FK Referenced Table
Product_Type	Type_ID	Product Type ID	varchar(20)	PDTxxxx			PK	
	PDT_Name	Product name	varchar(30)	Xxxxxxxx				
	PDT_Detail	Detail of Product Type	varchar(225)	XXXXXXXXXXXXXX				

Product

Table Name	Attribute Name	Contents	Type	Format	Nullable	Range	Key	FK Referenced Table
Product	Product_ID	Product ID	varchar(20)	SIDxxxxxxx			PK	
	Qauntity	number of product	int	10	Y			
	Product Description	detail	varchar(225)	XXxxxxxxx				
	Color	color of product	varchar(10)	red				
	Product Name	name of product	varchar(20)	xxxxxxxxxxx				
	Product Price	price of product	Int	3999				
	Size	size of product	varchar(2)	39	Y			
	Type_ID	Type of Product	varchar(20)	PDTxxxx			FK	Type_ID (Product Type)
	Collection_ID	Collection of Product	Varchar(20)	COLLxxxx	Y		FK	Collection_ID (Collection)

Buy (User Buy Product)

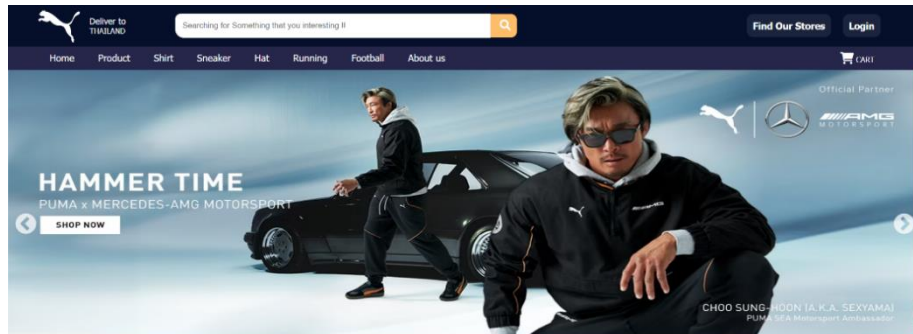
Table Name	Attribute Name	Contents	Type	Format	Nullable	Range	Key	FK referenced Table
BUY	UserAccount_ID	User ID	int	UIDxxxxxx			PK,FK	UserAccount_ID(User Account)
	Product_ID	Product ID	varchar(20)	SIDxxxxxxx			PK,FK	Product_ID (Product)

Web Application Project Phase 2

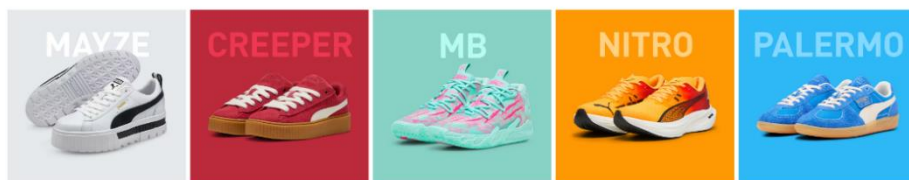
รายละเอียดหน้าเว็บของเว็บแอปพลิเคชัน

1. หน้าหลัก

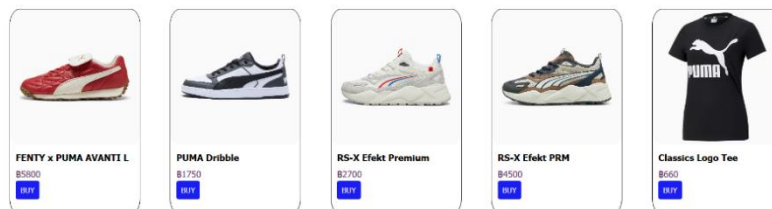
จะแสดงหน้านี้เมื่อผู้ใช้งานที่ปุ่ม Home จะแสดงข้อมูลต่างๆเกี่ยวกับระบบหรือ Service ที่ทำ



Our Example Product



Product Release

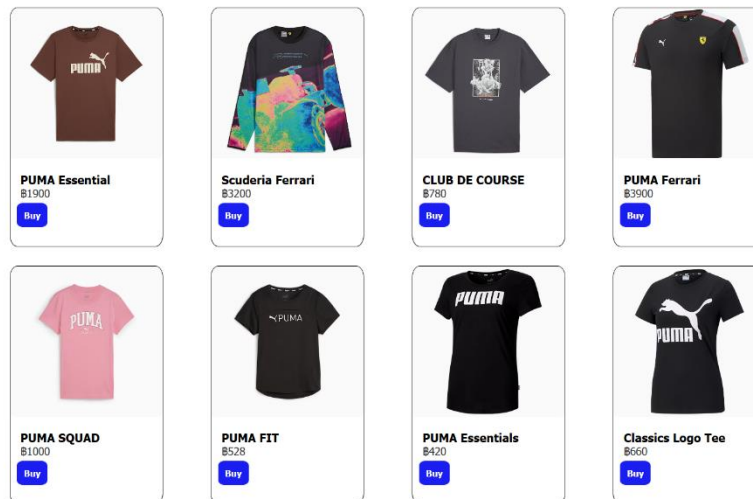


EXPLORE YOUR SELF WITH YOUR STYLE

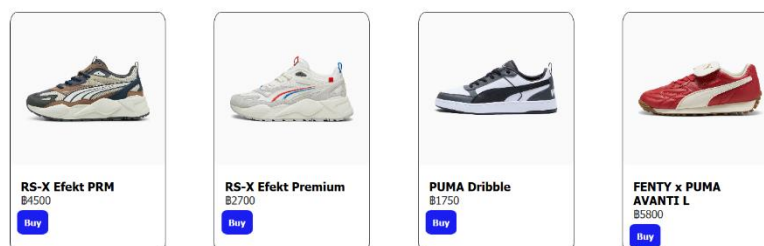


2. หน้าแสดงข้อมูลรายการสินค้า

เมื่อผู้ใช้งานที่ประเภทสินค้าต่าง ๆ ข้อมูลจะแสดงสินค้าตามประเภทสินค้าที่เลือก




© Copyright by Group 11 Section 2
Contact Our team Click here



© Copyright by Group 11 Section 2
Contact Our team Click here

3. หน้าแสดงข้อมูลรายละเอียดสินค้า

เมื่อผู้ใช้กดซื้อที่สินค้า เว็บไซต์จะทำการแสดงรายละเอียดข้อมูลของสินค้า



Deliver to
THAILAND

Searching for Something that you interesting !!

Q

Find Our Stores

Login

Home

Product

Shirt

Sneaker


Hat

Running

Football

About us

CART



All/Category: Shirt

Scuderia Ferrari

\$ 3200

Color selection

White

Black

Blue

Green

Black

Size selection

XS

S

M

L

XL

XXL

Quantity

1

Add to Cart

Product Description

ยกระดับสไตล์ของคุณด้วยเสื้อยืดแขนยาว Scuderia Ferrari Race Statement สุดเรียบหรู แสดงความรักต่อรูปม้าด้วยกราฟิกแลนโดลด์ เด่นที่จะขับเคลื่อนคุณไปสู่อันดับที่หนึ่ง

© Copyright by Group 11 Section 2

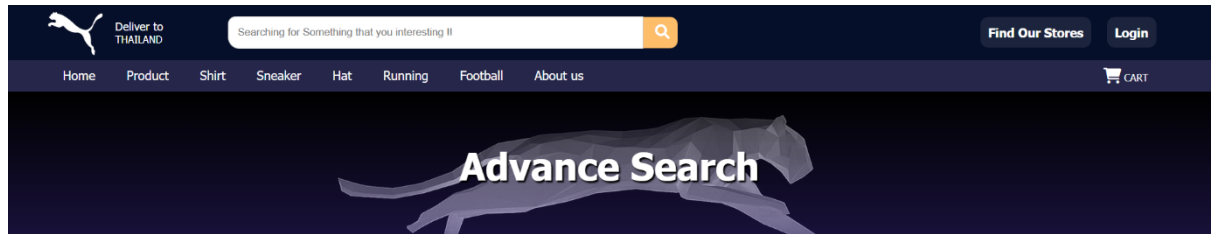
Contact Our team Click here

4. หน้าการค้นหาสินค้า

เมื่อผู้ใช้กดที่ปุ่มการค้นหา จะแสดงหน้านี้ และ เมื่อผู้ใช้ใส่ข้อมูลจะแสดงสินค้าตามที่ใช้คัดกรอง หากไม่มีการคัดกรองสินค้าจะแสดงข้อมูลสินค้าทั้งหมดในระบบ เช่นดังภาพ

Product Type: shirt

Collection: Summer




Product Name

Product Type:


Gender ☐ Male ☐ Female ☐ Unisex

Collection


Value Min - Max



PUMA Essential
฿1900



CLUB DE COURSE
฿780



PUMA FIT
฿528

5. หน้าการเข้าสู่ระบบ สำหรับผู้ดูแลระบบ

เมื่อผู้ใช้ทำการกดที่ปุ่ม Login ระบบจะทำการแสดงหน้าเข้าสู่ระบบ โดยจะมีการเชื่อมต่อกับ Web service เพื่อตรวจสอบการเข้าสู่ระบบของผู้ดูแลระบบ

The image shows a screenshot of a Puma website. At the top, there is a dark blue header with the Puma logo on the left, a search bar in the center, and navigation links on the right. Below the header is a dark blue navigation bar with links for Home, Product, Shirt, Sneaker, Hat, Running, Football, and About us. In the center of the page, there is a white login form with a blue border. The form has a title 'Login' and two input fields labeled 'Username' and 'Password'. Below the input fields is a blue 'Login' button. At the bottom of the form, there is a link that says 'Don't have an account? Register'. At the very bottom of the page, there is a dark blue footer with copyright information.

Deliver to THAILAND

Searching for Something that you interesting !!

Find Our Stores Login

Home Product Shirt Sneaker Hat Running Football About us

CART

Login

Username

Username

Password

Password

Login

Don't have an account? Register

© Copyright by Group 11 Section 2
Contact Our team Click here

6. หน้าสำหรับการจัดการข้อมูลสินค้า ของผู้ดูแลระบบ

1. หน้า Product management ได้แก่ ADD , EDIT , DELETE , Search & View

MENU

Dashboard

Homepage

Account Management

All Admin

Search Admin

Product Management

All Product

Search Product

Logout

Product Management

+ ADD

PID	ProductName	ProductType	Price	ProductIMG	ACTION
28	PUMA Essential	Shirt	1900		<div>EDIT</div> <div>DELETE</div>
29	Scuderia Ferrari	Shirt	3200		<div>EDIT</div> <div>DELETE</div>
30	CLUB DE COURSE	Shirt	780		<div>EDIT</div> <div>DELETE</div>
31	PUMA Ferrari	Shirt	3900		<div>EDIT</div> <div>DELETE</div>
32	PUMA SQUAD	Shirt	1000		<div>EDIT</div> <div>DELETE</div>
33	PUMA FIT	Shirt	528		<div>EDIT</div> <div>DELETE</div>

ADD Product

เมื่อผู้ดูแลระบบกดที่ปุ่ม ADD ด้านขวบนจะแสดงหน้าสำหรับการเพิ่มข้อมูลสินค้า

MENU

Dashboard

Homepage

Account Management

All Admin

Search Admin

Product Management

All Product

Search Product

Logout

Add Product Element

Back

Product Name

Enter ProductName

Product Type

select product type

Product Price

Enter ProductPrice

Collection

select collection

Gender

☐ Male

☐ Female

☐ Unisex

Product Description

Enter Product Description


100 x 100

เลือกไฟล์ | ไม่ได้อัปโหลด

CREATE PRODUCT

UPDATE & EDIT Product

เมื่อผู้ดูแลระบบกดที่ปุ่ม EDIT ของสินค้านั้นๆ จะแสดงหน้าสำหรับการแก้ไขข้อมูล พร้อมทั้งแสดงข้อมูลเดิมของสินค้าที่กดเลือกแก้ไข

 PUMA DASHBOARD

MENU

Dashboard

Homepage

Account Management

All Admin

Search Admin

Product Management

All Product

Search Product

Logout

Edit Product Element

Back

Product Name

Scuderia Ferrari

Product Type

Shirt

Product Price

3200

Collection

Winter

Gender


☒ Male

☐ Female

☐ Unisex

Product Description

ยกรดับได้ออกหนาวของคณะวินส์กีตแซนยาว Scuderia Ferrari R



เลือกไฟล์

ไม่ได้เลือกไฟล์ใด

EDIT PRODUCT

DELETE



เมื่อผู้ดูแลระบบกดปุ่ม Delete ระบบจะแสดงคำแจ้งเตือนการลบสินค้า หากกดยืนยัน ระบบจะทำการเรียก Web service เพื่อทำการลบข้อมูล

localhost:5000 บอกว่า
Are you sure to delete this product !!!

ตกลง


ยกเลิก

+ ADD


PID	ProductName	ProductType	Price	ProductIMG	ACTION
28	PUMA Essential	Shirt	1900		<div><div>EDIT</div><div>DELETE</div></div>
29	Scuderia Ferrari	Shirt	3200		<div><div>EDIT</div><div>DELETE</div></div>

SEARCH & VIEW


เมื่อผู้ดูแลระบบกด Search Product ที่ Sidebar ระบบจะแสดงหน้าสำหรับการค้นหาข้อมูลและสามารถดูสินค้าที่ค้นพบได้

PUMA DASHBOARD

MENU


Dashboard

Homepage

Account Management


All Admin

Search Admin

Product Management

All Product

Search Product

Logout

Search Product

Search by Name:

Search by Product Type:





T-Shirt


Search by Gender:

Female


Search

Search Results


PID	Product Name	Product Type	Price	Image	ACTION
32	PUMA SQUAD	Shirt	1000		VIEW
33	PUMA FIT	Shirt	528		VIEW
34	PUMA Essentials	Shirt	420		VIEW
35	Classics Logo Tee	Shirt	660		VIEW

PUMA DASHBOARD

MENU


Dashboard

Homepage

Account Management


All Admin

Search Admin

Product Management

All Product

Search Product

Logout

Detail Product

Product Name:

Product Type:

Product Price:

Collection:

Gender:


☐ Male

☒ Female


☐ Unisex

Product Description:


สร้างจากโพลีเอสเตอร์สีชมพูตัดเย็บด้วยวิธีพิเศษ โทนสีแบบสตรี PUM





2. หน้า Admin Account management ได้แก่ ADD , EDIT , DELETE , Search & View

 PUMA DASHBOARD

MENU


 Dashboard

 Homepage

 Account Management


All Admin

Search Admin

 Product Management



All Product

Search Product

 Logout


Admin Management

+ ADD


Admin_ID	IMAGES	Username	ACTION
39		MxnloodyX	<div>EDITDELETE</div>
81		Navadol	<div>EDITDELETE</div>


ADD Account


เมื่อผู้ดูแลระบบกดที่ปุ่ม Add จะแสดงหน้าสำหรับเพิ่มข้อมูลผู้ดูแลระบบ

 PUMA DASHBOARD

MENU


 Dashboard

 Homepage

 Account Management


All Admin

Search Admin

 Product Management

All Product

Search Product

 Logout

Add Admin Account

Back

Email

Enter Email

Username

Enter Username

Password

Enter Password

First Name

Enter Firstname

Last Name

Enter Lastname

Gender

☐ Male

☐ Female

Tel-number

Enter Tel-Number

Address

Enter Address

Position

Enter Position

100 x 100

เลือกไฟล์

ไม่จำเป็นต้องใส่ไฟล์

CREATE DATA

UPDATE & EDIT Account

เมื่อผู้ดูแลระบบกดที่ปุ่ม EDIT ของผู้ดูแลระบบคนนั้น จะแสดงหน้าสำหรับการแก้ไขข้อมูล พร้อมทั้งแสดงข้อมูลเดิมของของผู้ดูแลระบบที่กดเลือกแก้ไข

PUMA DASHBOARD

MENU

- Dashboard
- Homepage
- Account Management
 - All Admin
 - Search Admin
- Product Management
 - All Product
 - Search Product
- Logout

Edit Admin Account Back

Email: empmanzat150@hotmail.com Username: MxnloodyX Password: 12345

First Name: Watanachai Last Name: Boonchai

Gender: ☒ Male ☐ Female Tel-number: 0834745429

Address: 8306 mahidol dormitory Position: FullStack Developer

เลือกไฟล์ ลบไฟล์เลือกไฟล์ใหม่

EDIT DATA

DELETE

เมื่อผู้ดูแลระบบกดปุ่ม Delete ระบบจะแสดงคำแจ้งเตือนการลบข้อมูลผู้ดูแลระบบ หากกดยืนยัน ระบบจะทำการเรียก Web service เพื่อทำการลบข้อมูล

localhost:5000 บอกว่า
Are you sure to delete this admin account !!!

ตกลง ยกเลิก + ADD

Admin_ID	IMAGES	Username	ACTION
39		MxnloodyX	EDIT DELETE
81		Navadol2	EDIT DELETE

SEARCH & VIEW

เมื่อผู้ดูแลระบบกด Search Admin ที่ Sidebar ระบบจะแสดงหน้าสำหรับการค้นหาข้อมูลและสามารถดูข้อมูลเข้าสู่ระบบที่ค้นพบได้

MENU

Dashboard

Homepage

Account Management

All Admin

Search Admin

Product Management

All Product

Search Product

Logout

Search Admin

Search by Name:

Enter name to search

Search by Email:

Enter email

Search by Username:

Enter username

Search

Search Results

ID	Username	Email	Full Name	Actions
39	MxnloodyX	empmanza1150@hotmail.com	Wattanachai Boonchai	VIEW
81	Navadol2	nice123@mu.com	Navadol Somboonkul	VIEW

MENU

Dashboard

Homepage

Account Management

All Admin

Search Admin

Product Management

All Product

Search Product

Logout

Admin Account

Back

Email

empmanza1150@hotmail.com

Username

MxnloodyX

Password

12345

First Name

Wattanachai

Last Name

Boonchai

Gender

☒ Male ☐ Female

Tel-number

0834745429

Address

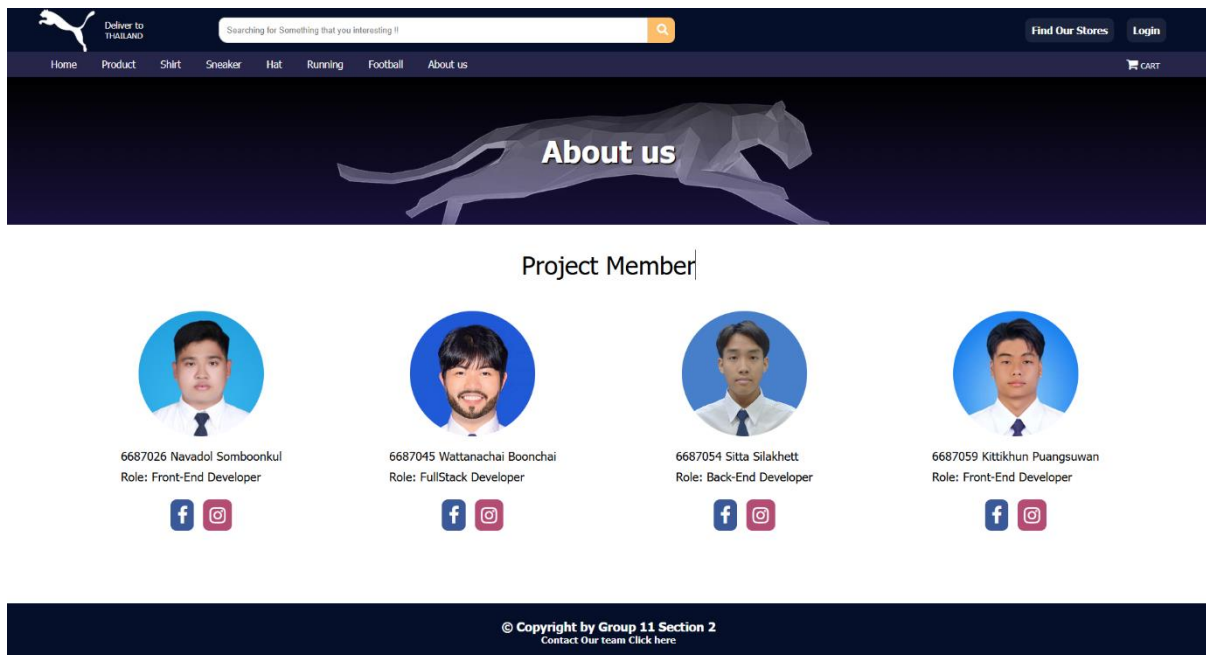
8306 mahidol dormitory

Position

FullStack Developer

7. About us

เมื่อกดที่ปุ่ม About ข้อมูลของผู้จัดทำโครงการ และมีปุ่ม Social link ของผู้จัดทำแต่ละคน



Administrators Login Web Service

```
// Login Admin
router.post("/adminLogin", (req, res) => {
  //รับข้อมูลจาก client
  const {adminUsername, adminPassword} = req.body;
  if (!adminUsername || !adminPassword) {
    //หากขาดข้อมูลให้ตอบกลับด้วย status 400 และข้อความ "Please Provide adminUsername and adminPassword"
    return res.status(400).json({success: false, message: "Please Provide adminUsername and adminPassword"});
  }
  //คำสั่ง query
  dbConnection.query("SELECT * FROM adminAccount WHERE username = ? AND password = ?", [adminUsername, adminPassword], function (err, results) {
    if (err) {
      //หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"
      return res.status(500).json({success: false, message: "Database Error"});
    }
    if (results.length === 0) {
      //หากไม่พบข้อมูล admin ให้ตอบกลับด้วย status 401 และข้อความ "Invalid adminUsername or adminPassword"
      return res.status(401).json({success: false, message: "Invalid adminUsername or adminPassword"});
    }
    const adminData = results[0];
    const jwtToken = jwt.sign({adminUsername: adminData.username, Admin_ID: adminData.Admin_ID.toString()}, process.env.JWT_SECRET, {
      expiresIn: "1h",
    });
    //ส่ง token และ ข้อมูล
    return res.status(200).json({success: true, token: jwtToken, adminData});
  });
});
```

เป็นการรับ request มาจาก client นำมาเช็คค่า json format ตรงหรือไม่

*หากขาดข้อมูลหรือรูปแบบไม่ตรงจะส่ง respond status 400 และข้อความ "Please Provide adminUsername and adminPassword" กลับไป

-เรียกใช้คำสั่ง query ที่เขียนกำหนดไว้

*หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

*หากไม่พบข้อมูล admin ให้ตอบกลับด้วย status 401 และข้อความ "Invalid adminUsername or adminPassword"

-สร้างตัวแปรจัดเก็บข้อมูล Admin

-สร้างตัวแปรเพื่อ เก็บ token admin ให้แก่ รหัส admin นั้นๆ

-ส่ง respond status 200, token และ ข้อมูล Admin

Admin account management Web Service API

Insert Admin

```
// เพิ่ม Admin
router.post("/InsertAdmin", upload.single("profile_image"), (req, res) => {
  //รับข้อมูลจาก client
  const u = ({email, username, password, first_name, last_name, gender, tel_number, address, position} = req.body);
  const profile_image = req.file ? req.file.filename : null;
  //หากขาดข้อมูลให้ตอบกลับด้วย status 400 และข้อความ "All fields are required"
  if (!email || !username || !password || !first_name || !last_name || !gender || !tel_number || !address || !position || !profile_image) {
    return res.status(400).json({success: false, message: "All fields are required"});
  }
  //คำสั่ง query
  const insertQuery =
    "INSERT INTO adminAccount (email, username, password, f_name, l_name, gender, tel_number, address, position, profile_image) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
  dbConnection.query(
    insertQuery,
    [email, username, password, first_name, last_name, gender, tel_number, address, position, profile_image],
    (err, results) => {
      if (err) {
        //หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"
        return res.status(500).json({success: false, message: "Database Error"});
      }
      //ส่งข้อมูล
      return res.status(200).json({success: true, message: "Admin successfully added!", data: results});
    }
  );
});
```

เป็นการรับ request มาจาก client และสร้างตัวแปรเพื่อจัดเก็บรูปภาพ

*หากขาดข้อมูลหรือรูปแบบไม่ตรงจะส่ง respond status 400 และข้อความ "All fields are required"

-สร้างตัวแปรเพื่อจัดเก็บคำสั่ง query

-เรียกใช้คำสั่ง query เขียนที่กำหนดไว้

*หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

-ส่ง respond status 200, ข้อความ "Admin successfully added!" และ ข้อมูล Admin

Updated Admin

```
// Update Admin
router.put("/UpdateAdmin/:Admin_ID", upload.single("profile_image"), (req, res) => {
  //โดยจะดึงรหัส admin ผ่าน parameter ใน URL
  const Admin_ID = req.params.Admin_ID;
  //รับข้อมูลจาก client
  const {email, username, password, first_name, last_name, gender, tel_number, address, position} = req.body;
  const profile_image = req.file ? req.file.filename : null;
  //สร้าง query
  let updateQuery = `UPDATE adminAccount SET email = ?, username = ?, password = ?, f_name = ?, l_name = ?, gender = ?, tel_number = ?, address = ?, position = ?`;
  const attribute = [email, username, password, first_name, last_name, gender, tel_number, address, position];
  //เพิ่ม query parameters ที่ใส่เข้ามาเพื่อ select ข้อมูล
  if (profile_image) {
    updateQuery += `, profile_image = ?`;
    attribute.push(profile_image);
  }
  updateQuery += ` WHERE Admin_ID = ?`;
  attribute.push(Admin_ID);
  dbConnection.query(updateQuery, attribute, (err, results) => {
    if (err) {
      //หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"
      return res.status(500).json({success: false, message: "Database Error", error: err.message});
    }
    if (results.affectedRows === 0) {
      return res.status(404).json({
        success: false,
        message: "Admin not found.",
      });
    }
    //ส่ง respond และข้อความ "Admin updated successfully"
    return res.status(200).json({
      success: true,
      message: "Admin updated successfully!",
    });
  });
});
```

-สร้างตัวแปรเก็บรหัส admin ผ่าน parameter ใน URL

เป็นการรับ request มาจาก client และสร้างตัวแปรเพื่อจัดเก็บรูปภาพ

-สร้างตัวแปรเพื่อจัดเก็บคำสั่ง query

-เพิ่ม query parameters ที่ใส่เข้ามาเพื่อ select ข้อมูล

-เรียกใช้คำสั่ง query เขียนที่กำหนดไว้

*หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

*หากไม่พบข้อมูล admin ให้ตอบกลับไปด้วย status 404 และข้อความ "Admin not found."

-ส่ง respond status 200 และข้อความ "Admin updated successfully!"

Delete Admin

```
// Delete Admin
router.delete("/adminDL/:Admin_ID", (req, res) => {
  // ดึงรหัส admin จาก parameter ใน URL
  const Admin_ID = req.params.Admin_ID;

  if (!Admin_ID) {
    // หากขาดข้อมูลให้ตอบกลับด้วย status 400 และข้อความ "Please provide Admin ID"
    return res.status(400).json({success: false, message: "Please provide Admin ID"});
  }
  // หลังจากลบไฟล์แล้ว, ลบข้อมูลจากฐานข้อมูล
  dbConnection.query("DELETE FROM adminAccount WHERE Admin_ID = ?", [Admin_ID], (err, results) => {
    if (err) {
      return res.status(500).json({success: false, message: "Database Error"});
    }
    if (results.affectedRows === 0) {
      return res.status(404).json({success: false, message: "Admin not found"});
    }
    return res.status(200).json({success: true, message: "Admin deleted successfully"});
  });
});
```

-สร้างตัวแปรเก็บรหัส admin ผ่าน parameter ใน URL

*หากขาดข้อมูลหรือรูปแบบไม่ตรงจะส่ง respond status 400 และข้อความ "Please provide Admin ID" กลับไป

-เรียกใช้คำสั่ง query ที่เขียนกำหนดไว้

*หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

*หากไม่พบข้อมูล admin ให้ตอบกลับไปด้วย status 404 และข้อความ "Admin not found."

-ส่ง respond status 200 และข้อความ "Admin deleted successfully!"

Get admin info

```
// ค้นหาข้อมูล Admin
router.get("/adminInfo/", (req, res) => {
  //คำสั่ง query
  const query = "SELECT * FROM adminAccount ";
  dbConnection.query(query, (err, results) => {
    if (err) {
      //หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"
      return res.status(500).json({success: false, message: "Database Error"});
    }
    //ส่งข้อมูล
    return res.status(200).json({success: true, data: results});
  });
});
```

-สร้างตัวแปรเพื่อจัดเก็บคำสั่ง query

-เรียกใช้คำสั่ง query เขียนที่กำหนดไว้

*หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

-ส่ง respond status 200, success: true และ ข้อมูล Admin

Get admin info by ID

```
// ค้นหาข้อมูล Admin จากรหัส Admin
router.get("/adminInfo/:Admin_ID", (req, res) => {
  //โดยจะดึงรหัส admin ผ่าน parameter ใน URL
  const Admin_ID = req.params.Admin_ID;
  //คำสั่ง query
  const query = "SELECT * FROM adminAccount WHERE Admin_ID = ?";
  dbConnection.query(query, Admin_ID, (err, results) => {
    if (err) {
      //หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"
      return res.status(500).json({success: false, message: "Database Error"});
    }
    if (results.length === 0) {
      return res.status(404).json({success: false, message: "Data not foundd"});
    }
    //ส่งข้อมูล
    return res.status(200).json({success: true, data: results});
  });
});
```

-สร้างตัวแปรเก็บรหัส admin ผ่าน parameter ใน URL

-สร้างตัวแปรเพื่อจัดเก็บคำสั่ง query

-เรียกใช้คำสั่ง query เขียนที่กำหนดไว้

*หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

*หากไม่พบข้อมูล admin ให้ตอบกลับไปด้วย status 404 และข้อความ "Data not found"

-ส่ง respond status 200, success: true และ ข้อมูล Admin

Search Admin

```
//ค้นหา Admin
router.get("/searchAdmin", (req, res) => {
  //รับข้อมูลจาก client
  const {name, email, username} = req.query;

  //คำสั่ง query
  let query = "SELECT * FROM adminAccount WHERE 1=1";
  //สร้างตัวแปรสำหรับจัดเก็บ parameter ในการเรียกข้อมูลจาก database
  const params = [];

  //เพิ่ม query parameters ที่ใส่เข้ามาเพื่อ select ข้อมูล
  if (name) {
    query += " AND (f_name LIKE ? OR l_name LIKE ?)";
    params.push(`%${name}%`, `%${name}%`);
  }

  if (email) {
    query += " AND email LIKE ?";
    params.push(`%${email}%`);
  }

  if (username) {
    query += " AND username LIKE ?";
    params.push(`%${username}%`);
  }

  // เรียกใช้ SQL query
  dbConnection.query(query, params, (err, results) => {
    if (err) {
      //หากเกิดข้อผิดพลาด (error) ในการเชื่อมต่อนข้อมูล ให้ตอบกลับไปด้วย status 500 และข้อความ "Database error"
      console.error("Error fetching data:", err);
      return res.status(500).json({
        success: false,
        message: "Database error",
      });
    }
    //หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 200 และข้อความ "No data found"
    if (results.length === 0) {
      return res.status(404).json({
        success: false,
        message: "No data found",
        data: [],
      });
    }
    //ส่งข้อมูล
    res.status(200).json({
      success: true,
      data: results,
    });
  });
});
```

เป็นการรับ request มาจาก client

-สร้างตัวแปรเพื่อจัดเก็บคำสั่ง query

-สร้างตัวแปรสำหรับจัดเก็บ parameter ในการเรียกข้อมูลจาก database

-เพิ่ม query parameters ที่ใส่เข้ามาเพื่อ select ข้อมูล

-เรียกใช้คำสั่ง query เขียนที่กำหนดไว้

*หากเกิดข้อผิดพลาดในการเชื่อมต่อฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

*หากไม่พบข้อมูล admin ให้ตอบกลับไปด้วย status 404 และข้อความ "No data found"

-ส่ง respond status 200, success: true และ ข้อมูล Admin

Product management Web Service API

Add product

```
//เพิ่มข้อมูลสินค้าลงในฐานข้อมูล โดยใช้ HTTP POST เพื่อรับข้อมูลจาก client และบันทึกลงในตาราง Product ในฐานข้อมูล
router.post("/addProduct", upload.single("product_image"), authorize, (req, res) => {
  //รับข้อมูลจาก client
  const {product_name, product_type, product_price, product_collection, product_gender, product_description} = req.body;
  const product_image = req.file ? req.file.filename : null;

  if (!product_name || !product_type || !product_price || !product_collection || !product_gender || !product_description || !product_image) {
    //หากขาดข้อมูลให้ตอบกลับด้วย status 400 และข้อความ "Please provide product information"
    return res.status(400).json({
      success: false,
      message: "Please provide product information",
    });
  }
  //คำสั่ง query
  const insertQuery =
    "INSERT INTO Product (product_name, product_type, product_price, product_collection, gender, product_description, image) VALUES (?, ?, ?, ?, ?, ?, ?)";
  //เพิ่มข้อมูลสินค้าลงในฐานข้อมูล
  dbConnection.query(
    insertQuery,
    [product_name, product_type, product_price, product_collection, product_gender, product_description, product_image],
    (err, results) => {
      if (err) {
        //หากเกิดข้อผิดพลาดในการเพิ่มข้อมูลสินค้าลงในฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"
        console.error("Error fetching data:", err);
        return res.status(500).json({
          success: false,
          message: "Database error",
        });
      }
      //หากการเพิ่มข้อมูลสินค้าสำเร็จให้ส่งข้อมูลสินค้า และตอบกลับด้วยข้อความ "New product has been created successfully."
      return res.send({
        success: true,
        message: "New product has been created successfully.",
        data: results,
      });
    }
  );
});
```

เป็นการเพิ่มข้อมูลสินค้าโดยการรับข้อมูลที่กรอกใน form ที่มาจากฝั่ง client และประกาศเป็นตัวแปรใหม่ โดยมีการรับทั้งข้อมูลสินค้า และภาพสินค้า

*หากไม่มีการกรอก form ในบางส่วนจะมีการตอบกลับด้วย status 400 และข้อความ “Please provide product information”

หากมีการกรอกข้อมูลครบถ้วนจะมีการประกาศ InsertQuery เพื่อจะทำการเพิ่มข้อมูลลงในฐานข้อมูลตามที่ประกาศตัวแปรไว้ก่อนหน้านี้

*หากเกิดข้อผิดพลาดในการเพิ่มข้อมูลสินค้าลงในฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ “Database error”

หากการเพิ่มข้อมูลสินค้าสำเร็จให้ส่งข้อมูลสินค้า และตอบกลับด้วยข้อความ “New product has been created successfully.” ไปที่ client

Show product

```
//แสดงสินค้าทั้งหมดจากฐานข้อมูล
router.get("/showProduct", (req, res) => {
  dbConnection.query("SELECT * FROM product", (err, results) => {
    if (err) {
      //หากเกิดข้อผิดพลาด (error) ในการเชื่อมต่อฐานข้อมูล ให้ตอบกลับไปด้วย status 500 และข้อความ "Database error"
      console.error("Error fetching product:", err);
      return res.status(500).json({
        success: false,
        message: "Database error",
      });
    }
    //ส่งข้อมูลสินค้า
    return res.status(200).json({
      success: true,
      results: results,
    });
  });
});
```

เป็นการแสดงรายการสินค้าทั้งหมดจากฐานข้อมูลเมื่อได้รับการเรียก req จาก route /showProduct
*หากเกิดข้อผิดพลาดในการแสดงข้อมูลสินค้าจากฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ
“Database error”

หากการส่งข้อมูลสินค้าสำเร็จให้ส่งข้อมูลสินค้าไปด้วยตัวแปร results และตอบกลับด้วย status 200
ไปที่ client

Show product by ID

```
//แสดงสินค้าจากฐานข้อมูล และ ส่งผลลัพธ์ของสินค้าที่มีรหัสสินค้าตรงตามกำหนด
router.get("/showProduct/:product_ID", (req, res) => {
  //โดยจะดึงรหัสสินค้าผ่าน parameter ใน URL
  const product_ID = req.params.product_ID;
  if (!product_ID) {
    //หากไม่ได้รับ parameter จะส่ง status 400 และข้อความ "Product ID is required"
    return res.status(400).json({
      success: false,
      message: "Product ID is required",
    });
  }
  const query = "SELECT * FROM product WHERE product_ID = ?";
  dbConnection.query(query, [product_ID], (err, results) => {
    if (err) {
      //หากเกิดข้อผิดพลาด (error) ในการเชื่อมต่อฐานข้อมูล ให้ตอบกลับไปด้วย status 500 และข้อความ Database error"
      console.error("Error fetching product:", err);
      return res.status(500).json({
        success: false,
        message: "Database error",
      });
    }
    if (results.length === 0) {
      //หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 404 และข้อความ "Product not found"
      return res.status(404).json({
        success: false,
        message: "Product not found",
        data: [],
      });
    }
    //ส่งข้อมูลสินค้า
    return res.status(200).json({
      success: true,
      data: results,
    });
  });
});
```

เป็นการแสดงรายการสินค้าตาม product_ID ที่ได้รับจาก client เพื่อดึงจากฐานข้อมูลเมื่อได้รับการเรียก request จาก route /showProduct/:product_ID

ประกาศตัวแปร product_ID เพื่อรับ parameter จาก client

* // หากไม่ได้รับ parameter จะส่ง status 400 และข้อความ "Product ID is required"

หากมีการกรอกข้อมูลครบถ้วนจะมีการประกาศ query เพื่อจะทำการเรียกข้อมูลจากฐานข้อมูลตาม product_ID ที่ request

*หากเกิดข้อผิดพลาดในการแสดงข้อมูลสินค้าจากฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

*หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 404 และข้อความ "Product not found"

หากการส่งข้อมูลสินค้าสำเร็จให้ส่งข้อมูลสินค้าไปด้วยตัวแปร results และตอบกลับด้วย status 200 ไปที่ client

Show new products release

```
//แสดงข้อมูลโดยมีการเรียงลำดับสินค้าที่เข้าสู่ฐานข้อมูล 5 อันล่าสุด
router.get("/shownewrelease", (req, res) => {
  //Query แสดงสินค้าโดยเรียงลำดับจาก5อันล่าสุด
  dbConnection.query("SELECT * FROM product ORDER BY product_ID DESC LIMIT 5", (err, results) => {
    if (err) {
      //หากเกิดข้อผิดพลาด (error) ในการเชื่อมต่อฐานข้อมูล ให้ตอบกลับไปด้วย status 500 และข้อความ "Database error"
      console.error("Error fetching product:", err);
      return res.status(500).json({
        success: false,
        message: "Database error",
      });
    }
    //ส่งข้อมูลสินค้า
    return res.status(200).json({
      success: true,
      results: results,
    });
  });
});
```

เป็นการแสดงรายการสินค้า 5 ชิ้นล่าสุดจากฐานข้อมูลเมื่อได้รับการเรียก req จาก route

/shownewrelease โดยมีการ query เพื่อรับสินค้า 5 ชิ้นล่าสุดจากฐานข้อมูล

*หากเกิดข้อผิดพลาดในการแสดงข้อมูลสินค้าจากฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ
“Database error”

หากการส่งข้อมูลสินค้าสำเร็จให้ส่งข้อมูลสินค้าไปด้วยตัวแปร results และตอบกลับด้วย status 200
ไปที่ client

Show product by type

```
// แสดงสินค้าจากประเภทสินค้าที่เลือก
router.get("/showProduct/type/:product_type", (req, res) => {
  // โดยจะดึงประเภทสินค้าผ่าน parameter ใน URL
  const product_type = req.params.product_type;
  if (!product_type) {
    // หากไม่ได้รับ parameter จะส่ง status 400 และข้อความ "Product type is required"
    return res.status(400).json({
      success: false,
      message: "Product type is required",
    });
  }
  // คำสั่ง query
  const query = "SELECT * FROM product WHERE product_type = ?";
  dbConnection.query(query, [product_type], (err, results) => {
    if (err) {
      // หากเกิดข้อผิดพลาด (error) ในการเชื่อมต่อนับฐานข้อมูล ให้ตอบกลับไปด้วย status 500 และข้อความ "Database error"
      console.error("Error fetching product:", err);
      return res.status(500).json({
        success: false,
        message: "Database error",
      });
    }
    // หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 404 และข้อความ "Product not found"
    if (results.length === 0) {
      return res.status(404).json({
        success: false,
        message: "Product not found",
        data: [],
      });
    }
    // ส่งข้อมูลสินค้า
    return res.status(200).json({
      success: true,
      data: results,
    });
  });
});
```

เป็นการแสดงรายการสินค้าตาม product_type ที่ได้รับจาก client เพื่อดึงจากฐานข้อมูลเมื่อได้รับการเรียก request จาก route /showProduct/type/:product_type และประกาศตัวแปร product_type เพื่อรับ parameter จาก client

*หากไม่ได้รับ parameter จะส่ง status 400 และข้อความ "Product type is required"

หากมีการกรอกข้อมูลครบถ้วนจะมีการประกาศ query เพื่อจะทำการเรียกข้อมูลจากฐานข้อมูลตาม product_type ที่ request

*หากเกิดข้อผิดพลาดในการแสดงข้อมูลสินค้าจากฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

*หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 404 และข้อความ "Product not found"

หากการส่งข้อมูลสินค้าสำเร็จให้ส่งข้อมูลสินค้าไปด้วยตัวแปร results และตอบกลับด้วย status 200 ไปที่ client

Delete product by product id

```
//ลบสินค้าจากรหัสสินค้า
router.delete("/productDL/:PID", authorize, (req, res) => {
  //โดยจะดึงรหัสสินค้าผ่าน parameter ใน URL
  let PID = req.params.PID;
  if (!PID) {
    //หากไม่ได้รับ parameter จะส่ง status 400 และข้อความ "Please provide Product id"
    return res.status(400).send({
      success: false,
      message: "Please provide Product id",
    });
  }
  //คำสั่ง query
  dbConnection.query("DELETE FROM Product WHERE product_ID = ?", PID, function (error, result) {
    //หากเกิดข้อผิดพลาด (error) ให้แสดง error กลับมา
    if (err) {
      //หากเกิดข้อผิดพลาด (error) ในการเชื่อมต่อนข้อมูล ให้ตอบกลับไปด้วย status 500 และข้อความ "Database error"
      console.error("Error fetching product:", err);
      return res.status(500).json({
        success: false,
        message: "Database error",
      });
    }
    return res.status(200).send({
      //ส่ง respond และข้อความ "Product has been deleted successfully."
      success: true,
      data: result.affectedRows,
      message: "Product has been deleted successfully.",
    });
  });
});
```

เป็นการลบสินค้าจากฐานข้อมูลตาม product_ID ที่มีการใส่เข้ามาเมื่อได้รับการ req จาก route

/showProduct

ประกาศตัวแปร PID เพื่อรับ parameter จาก client

*หากไม่ได้รับ parameter จะส่ง status 400 และข้อความ "Please provide Product id"

เรียกใช้คำสั่ง query เพื่อลบสินค้าตาม product_id

*หากเกิดข้อผิดพลาดในการลบข้อมูลสินค้าจากฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ

"Database error"

หากการส่งข้อมูลสินค้าสำเร็จให้ส่ง respond และข้อความ "Product has been deleted successfully."

ไปที่ client

Search products

```
//ค้นหาสินค้า
router.get("/searchProduct", (req, res) => {
  //กำหนดค่าใน request
  const {productname, producttype, gender} = req.query;

  //คำสั่ง query
  let query = "SELECT * FROM product WHERE 1=1";
  //สร้างตัวแปรสำหรับจัดเก็บ parameter ในการเรียกข้อมูลจาก database
  const params = [];

  //เพิ่ม query parameters ที่ใส่เข้ามาเพื่อ select ข้อมูล
  if (productname) {
    query += " AND product_name LIKE ?";
    params.push(`%${productname}%`);
  }
  if (producttype) {
    query += " AND product_type LIKE ?";
    params.push(`%${producttype}%`);
  }
  if (gender) {
    query += " AND gender LIKE ?";
    params.push(`%${gender}%`);
  }
  // เรียกใช้ SQL query
  dbConnection.query(query, params, (err, results) => {
    if (err) {
      //หากเกิดข้อผิดพลาด (error) ในการเชื่อมต่องานข้อมูล ให้ตอบกลับไปด้วย status 500 และข้อความ "Database error"
      console.error("Error fetching data:", err);
      return res.status(500).json({
        success: false,
        message: "Database error",
      });
    }
    if (results.length === 0) {
      //หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 200 และข้อความ "No data found"
      return res.status(200).json({
        success: false,
        message: "No data found",
        data: [],
      });
    }
    //ส่งข้อมูลสินค้า
    res.status(200).json({
      success: true,
      data: results,
    });
  });
});
```

เป็นการค้นหาสินค้าตาม parameters ที่ได้รับจาก client เพื่อดึงจากฐานข้อมูลเมื่อได้รับการเรียก request จาก route /searchProduct และประกาศตัวแปรตาม parameter ที่ได้รับจาก client และเพิ่ม query parameters ที่ใส่เข้ามาเพื่อทำการ select ข้อมูลจากนั้นทำการเรียกใช้ SQL query

*หากเกิดข้อผิดพลาดในการลบข้อมูลสินค้าจากฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

*หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 404 และข้อความ "Product not found"

หากการส่งข้อมูลสินค้าสำเร็จให้ส่งข้อมูลสินค้าไปด้วยตัวแปร results และตอบกลับด้วย status 200 ไปที่ client

Update product

```
// อัปเดตสินค้าจากกร็อสสินค้า
router.put("/UpdateProduct/:product_ID", upload.single("image"), authorize, (req, res) => {
  // รับ product_ID จาก URL
  let product_ID = req.params.product_ID;
  if (!product_ID) {
    // หากไม่ได้รับ parameter จะส่ง status 400 และข้อความ "Please provide Product id"
    return res.status(400).json({
      error: true,
      message: "Please provide a Product ID.",
    });
  }
  // รับข้อมูล product จาก client
  const {product_name, product_type, product_price, product_collection, gender, product_description} = req.body;
  // ตรวจสอบไฟล์ภาพ
  const image = req.file ? req.file.filename : null;
  // สร้างคำสั่ง query
  let updateQuery = `UPDATE Product SET `;
  const attribute = [];
  // เพิ่ม query parameters ที่ใส่เข้ามาเพื่อ update ข้อมูล
  if (product_name) {
    updateQuery += `product_name = ?, `;
    attribute.push(product_name);
  }
  if (product_type) {
    updateQuery += `product_type = ?, `;
    attribute.push(product_type);
  }
  if (product_price) {
    updateQuery += `product_price = ?, `;
    attribute.push(product_price);
  }
  if (product_collection) {
    updateQuery += `product_collection = ?, `;
    attribute.push(product_collection);
  }
  if (gender) {
    updateQuery += `gender = ?, `;
    attribute.push(gender);
  }
  if (product_description) {
    updateQuery += `product_description = ?, `;
    attribute.push(product_description);
  }
  if (image) {
    updateQuery += `image = ?, `;
    attribute.push(image);
  }

  // รวม WHERE Clause:
  updateQuery = updateQuery.slice(0, -2); // ตัด ", " ตัวสุดท้ายออก
  updateQuery += ` WHERE product_ID = ?`;
  attribute.push(product_ID);

  // เชื่อมต่อ SQL query
  dbConnection.query(updateQuery, attribute, (err, results) => {
    if (err) {
      // หากเกิดข้อผิดพลาด (error) ในการเชื่อมต่องานข้อมูล ให้ตอบกลับไปด้วย status 500 และข้อความ "Database error"
      console.error("Error updating product data: ", err);
      return res.status(500).json({
        success: false,
        message: "Error updating product data",
        error: err.message,
      });
    }
    if (results.affectedRows === 0) {
      // หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 200 และข้อความ "Product not found."
      return res.status(404).json({
        success: false,
        message: "Product not found.",
      });
    }
    // ส่ง respond และข้อความ "Product has been updated successfully."
    return res.status(200).json({
      success: true,
      message: "Product data updated successfully!",
      data: results,
    });
  });
});
```

เป็นการอัปเดตสินค้าตาม parameters ที่ได้รับจาก client เพื่ออัปเดตสินค้าในฐานข้อมูลเมื่อได้รับการเรียก request จาก route /UpdateProduct/:product_ID และประกาศตัวแปรตาม parameter ที่ได้รับจาก client และเพิ่ม query parameters ที่ใส่เข้ามาเพื่อทำการอัปเดตข้อมูลตาม product_ID จากนั้นทำการเรียกใช้ SQL query

*หากเกิดข้อผิดพลาดในการอัปเดตข้อมูลสินค้าจากฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ “Error updating product data”

*หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 404 และข้อความ "Product not found"

หากการส่งข้อมูลสินค้าสำเร็จให้ส่งข้อมูลสินค้าไปด้วยตัวแปร results ตอบกลับด้วย status 200 ไปที่ client และข้อความ “Product data updated successfully!”

Advance search product

```
//ค้นหาสินค้าผ่าน form จากหน้า Advance Search
router.get("/advanceSearchProduct", (req, res) => {
  //รับข้อมูลจาก client
  const {productname, producttype, gender, collection, min, max} = req.query;
  //คำสั่ง query
  let query = "SELECT * FROM product WHERE 1=1";
  //สร้างตัวแปรสำหรับจัดเก็บ parameter ในการเรียกข้อมูลจาก database
  const params = [];
  //เพิ่ม query parameters ที่ใส่เข้ามาเพื่อ select ข้อมูล
  if (productname) {
    query += " AND product_name LIKE ?";
    params.push(`%${productname}%`);
  }
  if (producttype) {
    query += " AND product_type LIKE ?";
    params.push(`%${producttype}%`);
  }
  if (gender) {
    query += " AND gender LIKE ?";
    params.push(`%${gender}%`);
  }
  if (collection) {
    query += " AND product_collection LIKE ?";
    params.push(`%${collection}%`);
  }
  if (min) {
    query += " AND product_price >= ?";
    params.push(Number(min));
  }
  if (max) {
    query += " AND product_price <= ?";
    params.push(Number(max));
  }

  // เรียกใช้ the query
  dbConnection.query(query, params, (err, results) => {
    if (err) {
      //หากเกิดข้อผิดพลาด (error) ในการเชื่อมต่องานข้อมูล ให้ตอบกลับไปด้วย status 500 และข้อความ "Database error"
      console.error("Error fetching data:", err);
      return res.status(500).json({
        success: false,
        message: "Database error",
      });
    }

    if (results.length === 0) {
      //หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 200 และข้อความ "Product not found."
      return res.status(404).json({
        success: false,
        message: "No products found",
        data: [],
      });
    }

    //หากค้นหาสินค้าสำเร็จให้ส่ง status 200 และส่ง results กลับไป
    return res.status(200).json({
      success: true,
      data: results,
    });
  });
});
```

เป็นการค้นหาสินค้าตาม parameters ที่ได้รับจาก client เพื่อดึงจากฐานข้อมูลเมื่อได้รับการเรียก request จาก route /advanceSearchProduct และประกาศตัวแปรตาม parameter ที่ได้รับจาก client เพิ่ม query parameters ที่ใส่เข้ามาเพื่อทำการ select ข้อมูลจากนั้นทำการเรียกใช้ SQL query

*หากเกิดข้อผิดพลาดในการลบข้อมูลสินค้าจากฐานข้อมูลให้ตอบกลับด้วย status 500 และข้อความ "Database error"

*หากไม่พบข้อมูลสินค้า ให้ตอบกลับไปด้วย status 404 และข้อความ "Product not found"

หากการส่งข้อมูลสินค้าสำเร็จให้ส่งข้อมูลสินค้าไปด้วยตัวแปร results และตอบกลับด้วย status 200 ไปที่ client

Public API : LONGDO MAP API

LONGDO MAP เป็น API ที่ให้บริการในเรื่องของเทคโนโลยีแผนที่ออนไลน์

การประยุกต์ใช้ในโครงงาน

นำมาใช้ในการดึงข้อมูลแผนที่ในประเทศไทย และทำการทำ Marker ของสาขาร้านค้า PUMA ในประเทศไทย ในตำแหน่งต่างๆ

วิธีการเรียก API มาใช้เบื้องต้น

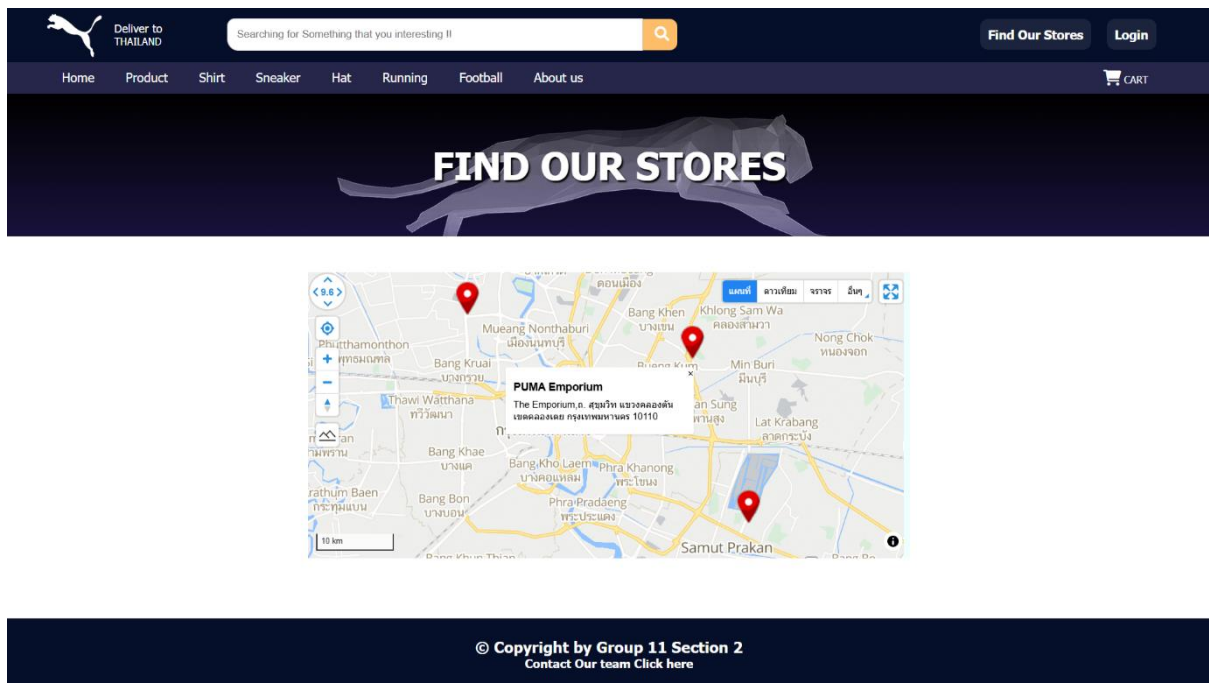
1.เรียก API ด้วย TAG พร้อมใส่ src เป็นลิ้งค์ API ที่ตามด้วย token ของผู้ใช้

```
</style>
<script type="text/javascript" src="https://api.longdo.com/map3/?key=199cd971899dde4b509b5c73a6645d2d"></script>
</script>
```

2.สร้าง div tag ที่ id = map เพื่อเรียกแสดงผลแผนที่

```
</section>
<div id="map"></div>
</section>
```

ภาพตัวอย่างของการนำ API มาประยุกต์ใช้กับโครงงาน



ผลการทดสอบของเว็บเซอร์วิสทั้งหมด โดยใช้โปรแกรม Postman

Setting Auth Type ของทุก ๆ API = Bearer Token

นำ token จากการ Login ไปใส่ในทุกๆ API ที่ทำการ Testing

Admin account management API Testing

Administrator Login

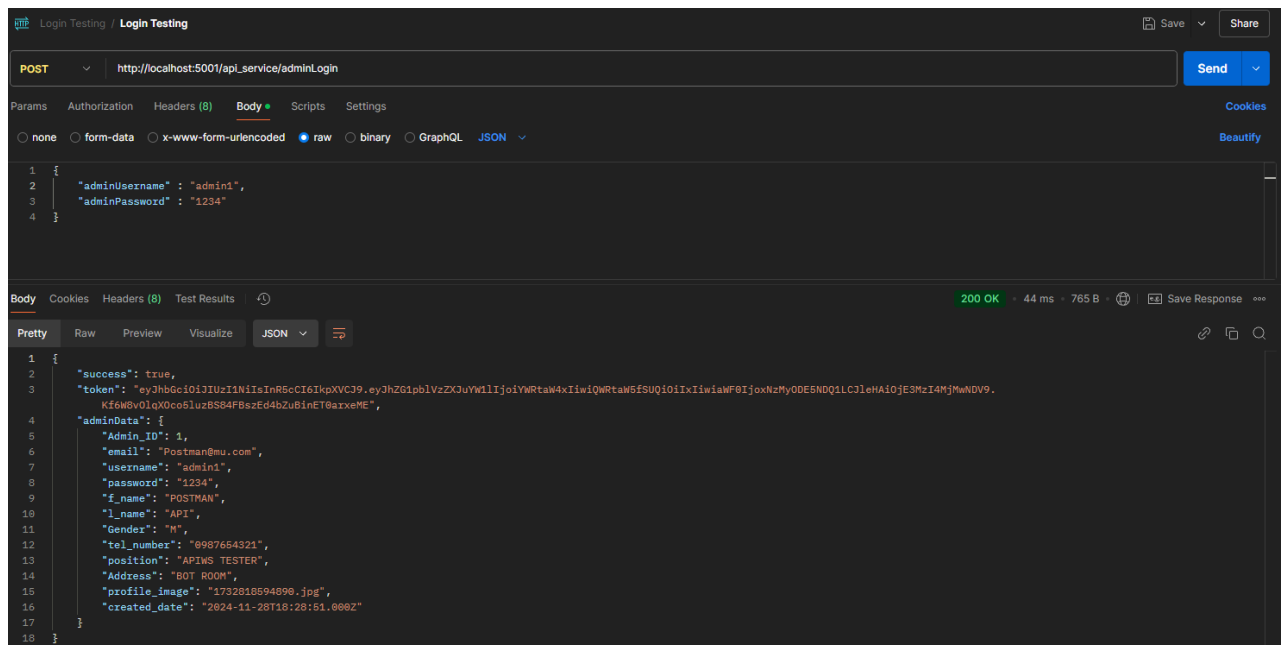
Method: POST

Auth Type: bearer Token

URL : http://localhost:5001/api_service/adminLogin

Body : raw JSON

```
{
    "adminUsername": "MxnldyX",
    "adminPassword": "1234"
}
```

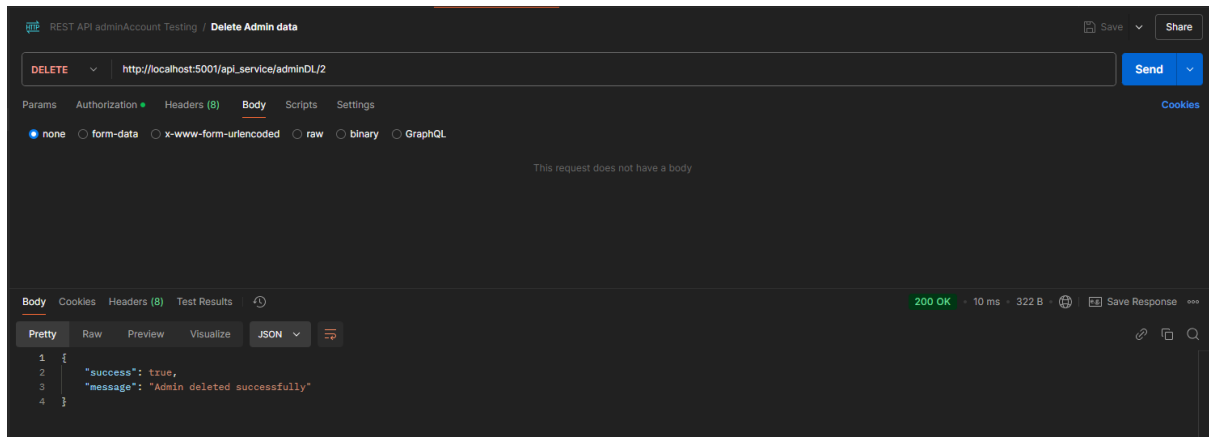


Delete Admin Account

Method: DELETE

Auth Type: bearer Token

URL `http://localhost:5001/api_service/adminDL/:AdminID`



Add Admin Account

Method: POST

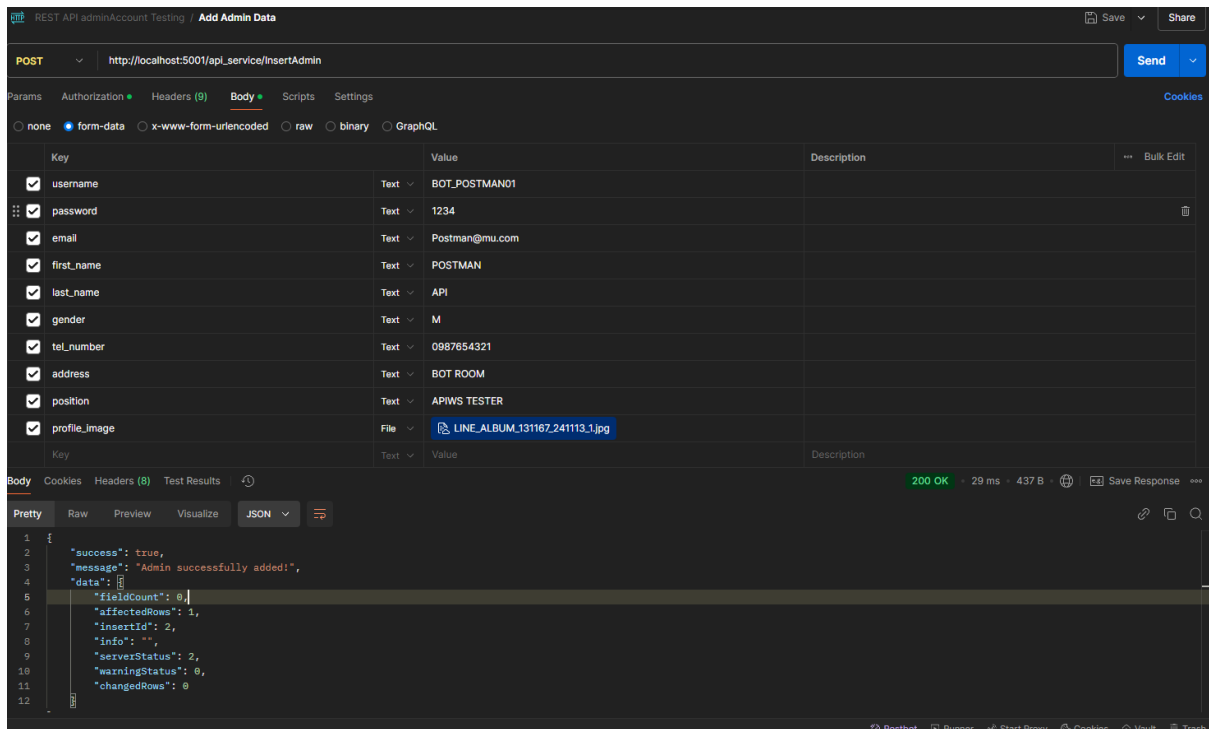
Auth Type: bearer Token

URL http://localhost:5001/api_service/InsertAdmin

Body : form-data

{username, password, email, first_name, last_name, gender, tel_number, address, position} Body type = Text

{Profile_image} Body type = File

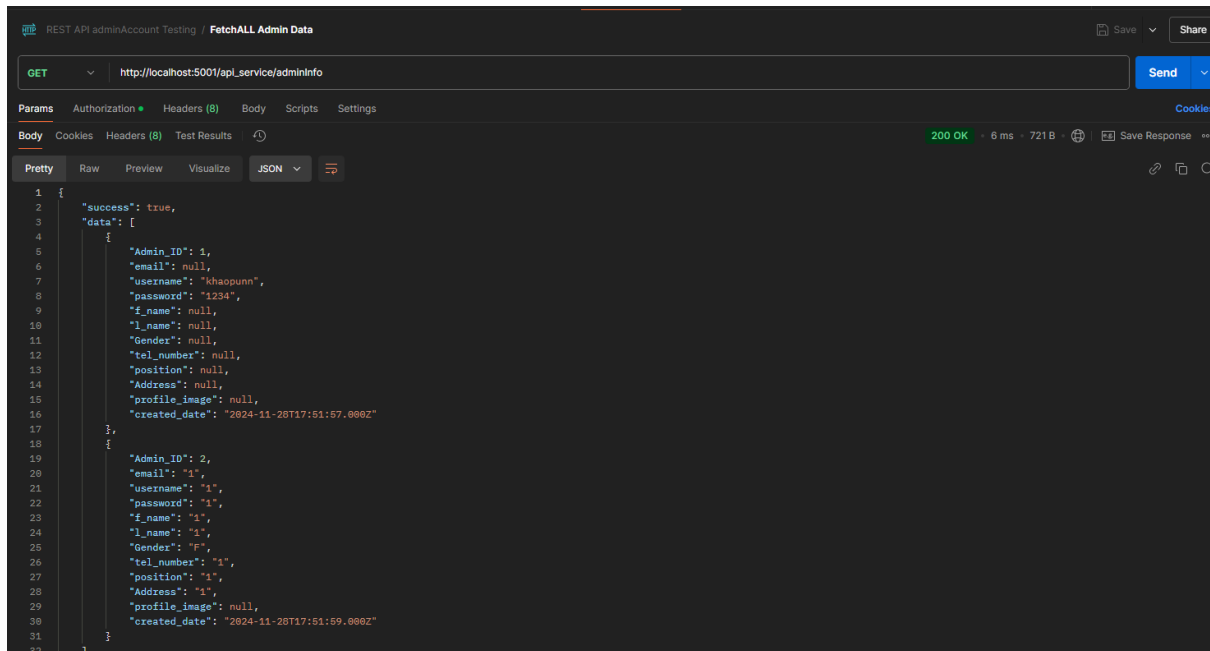


Show Admin Account

Method: Get

Auth Type: bearer Token

URL: http://localhost:5001/api_service/adminInfo



Update Admin Account

Method: PUT

Auth Type: bearer Token

URL : http://localhost:5001/api_service/UpdateAdmin/:Admin_ID

Body: form-data

Request Body เลือกแค่ attribute ที่ต้องการแก้ไข ตัวอย่างดังภาพเช่น แก้ไข password

PUT http://localhost:5001/api_service/UpdateAdmin/1

Params Authorization Headers (9) **Body** Scripts Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	Key		Value
<input type="checkbox"/>	username	Text	MxnIodyX
<input checked="" type="checkbox"/>	password	Text	12345
<input type="checkbox"/>	first_name	Text	Wattanachai
<input type="checkbox"/>	last_name	Text	Boonchai
<input type="checkbox"/>	gender	Text	M
<input type="checkbox"/>	tel_number	Text	0834745429
<input type="checkbox"/>	position	Text	Full-Stack Developer
<input type="checkbox"/>	address	Text	Mahidol University
<input type="checkbox"/>	email	Text	Wattanachai.bon@student.mahidol.ac.th
	Key	Text	Value

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true,
3   "message": "Admin updated successfully!"
4 }
```

Show Admin Account By Admin_ID

Method: GET

Auth Type: bearer Token

URL : http://localhost:5001/api_service/adminInfo/:adminID

The screenshot displays a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:5001/api_service/adminInfo/1
- Body:** This request does not have a body.
- Response:** 200 OK, 5 ms, 623 B. The response is displayed in JSON format.

```
1 {
2   "success": true,
3   "data": [
4     {
5       "Admin_ID": 1,
6       "email": "Wattanachai.bon@student.mahidol.ac.th",
7       "username": "MxnloodyX",
8       "password": "12345",
9       "f_name": "Wattanachai",
10      "l_name": "Boonchai",
11      "Gender": "M",
12      "tel_number": "8834745429",
13      "position": "Full-Stack Developer",
14      "Address": "Mahidol University",
15      "profile_image": "1732818594898.jpg",
16      "created_date": "2024-11-26T18:28:51.000Z"
17    }
18  ]
19 }
```

Search Admin Account

METHOD: GET

Auth Type: bearer Token

URL: http://localhost:5001/api_service/searchAdmin

Params : name , email , username

เช่น ค้นหาด้วย username = Mxn

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:5001/api_service/searchAdmin?name=&email=&username=Mxn
- Params:** name, email, username (all checked)
- Response:** 200 OK, 11 ms, 623 B
- Body (JSON):**

```
{
  "success": true,
  "data": [
    {
      "Admin_ID": 1,
      "email": "Wattanachai.bon@student.mahidol.ac.th",
      "username": "MxnlodyX",
      "password": "12345",
      "f_name": "Wattanachai",
      "l_name": "Boonchai",
      "Gender": "M",
      "tel_number": "0834745429",
      "position": "Full-Stack Developer",
      "Address": "Mahidol University",
      "profile_image": "1732818594898.jpg",
      "created_date": "2024-11-26T18:28:51.098Z"
    }
  ]
}
```

Product management API Testing

Show Product

Method: GET

URL: http://localhost:5001/api_service/showProduct

REST API productManagement Testing / FetchALL Product Data

GET http://localhost:5001/api_service/showProduct

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (6) Test Results

200 OK • 5 ms • 1.1 KB

Save Response

Pretty Raw Preview Visualize JSON

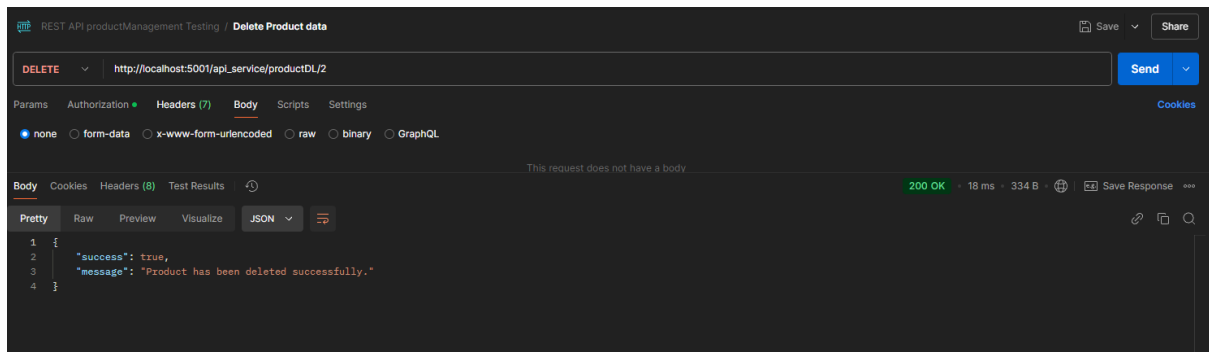
```
1 {
2   "success": true,
3   "results": [
4     {
5       "product_ID": 1,
6       "product_name": "PUMA Essential",
7       "product_type": "Shirt",
8       "product_price": 1900,
9       "product_collection": "Summer",
10      "gender": "W",
11      "product_description": "PUMA x Essential",
12      "image": "1732839401412.avif"
13    },
14    {
15      "product_ID": 2,
16      "product_name": "Classics Logo Tee",
17      "product_type": "Sneaker",
18      "product_price": 1900,
19      "product_collection": "Summer",
20      "gender": "F",
21      "product_description": "ชุดเสื้อโปโลสีฟ้าของแบรนด์เสื้อกีฬามืออาชีพ Scuderia Ferrari Race Statement ชุดเสื้อโปโล
22      แดงขาวที่มีลวดลายด้านหน้าจากศิลปินโคเคนที่กำลังเปลี่ยนชุดไปสู่อันสืบที่หนึ่ง",
23      "image": "1732839425335.avif"
24    }
25  ]
26 }
```

Delete product by product id

Method: DELETE

Auth Type: Bearer Token

URL: http://localhost:5001/api_service/productDL/{:Product_ID}



Add product

METHOD: POST

Auth: Bearer Token

URL: http://localhost:5001/api_service/addProduct

Body: Form-data

{product_name, product_type, product_type, product_collection, product_gender
, product_description} Body type = Text

{product_image) Body type = File

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:5001/api_service/addProduct
- Body Type:** form-data
- Form Data:**

Key	Value	Description
product_name	PUMA-Cilla-patent	
product_type	Sneaker	
product_price	1900	
product_collection	Winter	
product_gender	F	
product_description	SNEAKER MATCH WITH EVERY LOOK	
product_image	รูปสินค้ารองเท้า-Patent.avif	
- Response:** 200 OK, 43 ms, 454 B. The response body is a JSON object:

```
{  "success": true,  "message": "New product has been created successfully.",  "data": {    "fieldCount": 0,    "affectedRows": 1,    "insertId": 6,    "info": "",    "serverStatus": 2,    "warningStatus": 0,    "changedRows": 0  }  }
```

Update product

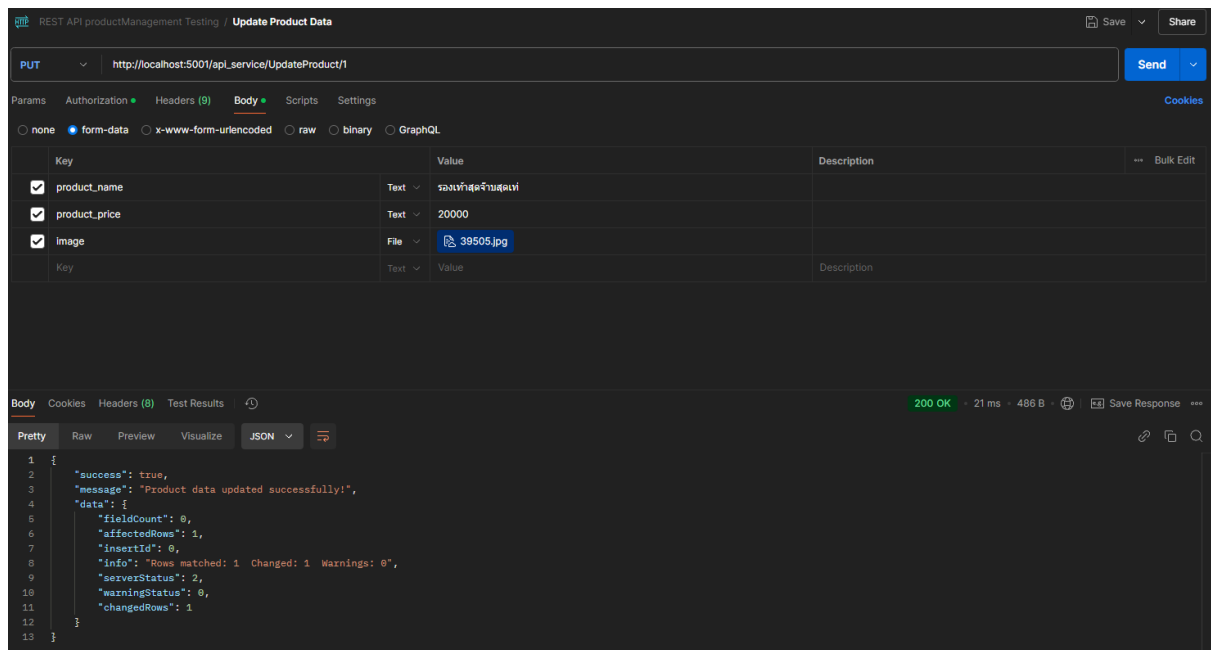
METHOD: PUT

AUTH: Bearer Token

URL: `http://localhost:5001/api_service/UpdateProduct/: {product_Id}`

Body: Form-data

Request Body เลือกแค่ attribute ที่ต้องการแก้ไข



REST API productManagement Testing / Update Product Data

PUT `http://localhost:5001/api_service/UpdateProduct/1` Send

Params Authorization Headers (9) Body Scripts Settings Cookies

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

Key	Value	Description
<input checked="" type="checkbox"/> product_name	Text รองเท้าสุดเจ๋งสุดเหิ	
<input checked="" type="checkbox"/> product_price	Text 20000	
<input checked="" type="checkbox"/> image	File 39505.jpg	

Body Cookies Headers (8) Test Results 200 OK · 21 ms · 486 B Save Response

Pretty Raw Preview Visualize .JSON

```
1 {
2   "success": true,
3   "message": "Product data updated successfully!",
4   "data": {
5     "fieldCount": 0,
6     "affectedRows": 1,
7     "insertId": 0,
8     "info": "Rows matched: 1 Changed: 1 Warnings: 0",
9     "serverStatus": 2,
10    "warningStatus": 0,
11    "changedRows": 1
12  }
13 }
```

Show Product By Product_ID

METHOD: Get

URL: http://localhost:5001/api_service/showProduct/1

The screenshot displays a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:5001/api_service/showProduct/1
- Body:** This request does not have a body.
- Response:** 200 OK, 6 ms, 546 B
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "data": [
4     {
5       "product_ID": 1,
6       "product_name": "เสื้อยืดฤดูร้อน",
7       "product_type": "Shirt",
8       "product_price": 28888,
9       "product_collection": "Summer",
10      "gender": "M",
11      "product_description": "PUMA x Essential",
12      "image": "1732831376474.jpg"
13    }
14  ]
15 }
```

Search Product By params

METHOD: Get

URL: http://localhost:5001/api_service/searchProduct/?productname=&producttype=Shirt&gender=

Params: productname, producttype, gender

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:5001/api_service/searchProduct/?productname=&producttype=Shirt&gender=`
- Params:** productname, producttype, gender
- Status:** 200 OK
- Response Body (JSON):**

```
{
  "success": true,
  "data": [
    {
      "product_ID": 1,
      "product_name": "รองเท้าสุดจ๊ามสุดเท่",
      "product_type": "Shirt",
      "product_price": 20000,
      "product_collection": "Summer",
      "gender": "M",
      "product_description": "PUMA x Essential",
      "image": "1732831376474.jpg"
    },
    {
      "product_ID": 7,
      "product_name": "Classics Logo Tee",
      "product_type": "Shirt",
      "product_price": 20000,
      "product_collection": "Rainny",
      "gender": "M",
      "product_description": "ยกระฉับสโสดักหนาวของคุดเนด้วยลือยัดแชนขาว Scuderia Ferrari Race Statement สุดเจี๊ยมพหุ  
แสดงความรักล่อปไม่าล้วยกกราฟักแสบโคตเศนที่จะขับเคื่อนคุดเนไปสู้วันฉับทั้งหนึ่ง",
      "image": "1732832946886.jpg"
    }
  ]
}
```

Advance Search Product

METHOD: GET

URL: http://localhost:5001/api_service/advanceSearchProduct?productname&producttype&gender&collection&min&max

Params: productname, producttype, gender, collection, min, max

กรอก params ที่ต้องการใช้ในการค้นหา เช่นดังตัวอย่าง Gender == M

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:5001/api_service/advanceSearchProduct?productname&producttype&gender=M&collection&min&max`
- Params:** A table with 6 rows: productname, producttype, gender (value: M), collection, min, and max.
- Body:** The response is in JSON format, showing a successful status (200 OK) and a list of two products. The first product is a 'Shirt' from the 'Summer' collection, and the second is a 'Tee' from the 'Rainy' collection.

```
1 {
2   "success": true,
3   "data": [
4     {
5       "product_ID": 1,
6       "product_name": "เสื้อกีฬาฤดูร้อน",
7       "product_type": "Shirt",
8       "product_price": 28000,
9       "product_collection": "Summer",
10      "gender": "M",
11      "product_description": "PUMA x Essential",
12      "image": "1732831376474.jpg"
13    },
14    {
15      "product_ID": 7,
16      "product_name": "Classic Logo Tee",
17      "product_type": "Tee",
18      "product_price": 28000,
19      "product_collection": "Rainy",
20      "gender": "M",
21      "product_description": "รถแข่งในศึกสูตรหนึ่งจากทีม Scuderia Ferrari Race Statement สุดร้อนแรง
22      แสดงความกล้าหาญผ่านโลโก้ที่ขับเคลื่อนด้วยนวัตกรรม",
23      "image": "1732832946086.jpg"
24    }
25  ]
26 }
```

แหล่งอ้างอิง

1. "The History of Adidas and Puma". Newsweek. 13 เมษายน 2551
สืบค้นวันที่ 20 กันยายน 2567. <https://www.newsweek.com/history-adidas-and-puma-86373>
2. "Puma Website" สืบค้นวันที่ 11 กันยายน 2567 <https://th.puma.com/>