



รายงาน

เรื่อง การวิเคราะห์และออกแบบเว็บแอปพลิเคชัน

เสนอ

ผศ. ดร. จิตาภา ไกรสังข์
อาจารย์ ดร. วุฒิชาติ แสรวงผล

จัดทำโดย

- นาย ปณัຍกร แพนใหญ่ชادา รหัสนักศึกษา 6687027
- นาย พุฒินันทน์ เตชะวรารณ์ รหัสนักศึกษา 6687038
- นาย วริศ มาลีวรรณ รหัสนักศึกษา 6687097
- นาย ศิริประภัส กิมปี รหัสนักศึกษา 6687098

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา เทคโนโลยีด้านเว็บและการประยุกต์ใช้
หลักสูตรวิทยาการและเทคโนโลยีดิจิทัล(ทสวด ๒๔๑)
คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล
ภาคเรียนที่ 1 ปีการศึกษา 2567

Project Description

Sasuke Ramen คือร้านอาหารที่เริ่มต้นจากการร้านราเมงขนาดเล็กในชุมชน โดยมีจุดเด่นที่น้ำซุปสูตรพิเศษ หลากหลายสูตร และเส้นราเมงที่เหนียวแน่น ทำให้ร้านได้รับความนิยมอย่างรวดเร็ว ด้วยการเติบโตของธุรกิจและ ความต้องการในการเพิ่มความสะดวกให้กับลูกค้า ร้านจึงได้พัฒนาเว็บไซต์ "Sasuke Ramen" เพื่อแสดงเมนูอาหาร ของร้านค้า ลูกค้าสามารถเลือกชมเมนูราเมงต่าง ๆ ของทางร้านได้อย่างง่ายดาย โดยเว็บไซต์ถูกออกแบบมาเพื่อให้ ลูกค้าสามารถเข้าถึงเมนูที่ต้องการได้อย่างรวดเร็ว ในด้านการค้นหาสินค้า และทำให้ลูกค้าสามารถเลือกชมราเมงที่ ชื่นชอบได้สะดวกจากทุกที่ รองรับการขยายตัวของธุรกิจในยุคดิจิทัลที่ต้องการความรวดเร็วและความสะดวกสบาย ในการใช้งาน ซึ่งในอนาคตจะนำระบบ e-commerce เข้ามาร่วมเพื่อขยายธุรกิจเพื่อเพิ่มความสะดวกสบายให้กับ ลูกค้าที่ต้องการความรวดเร็วหรือไม่มี

สำหรับการทำ Frontend Development ของโปรเจกต์นี้ เราใช้ React เป็นหลักในการพัฒนา ซึ่งเป็น ไลบรารี JavaScript ที่ช่วยในการสร้าง UI โดยเน้นที่การทำงานแบบ component-based ซึ่งช่วยให้โค้ดมีความ ยืดหยุ่นและสามารถใช้งานข้ามได้ นอกจากนี้ เราเลือกใช้ Vite ในการสร้างและจัดการโปรเจกต์ เนื่องจาก Vite ช่วยให้ การพัฒนาเร็วขึ้นด้วยระบบการโหลดโมดูลที่ทันสมัยและการคอมไพล์ที่มีประสิทธิภาพ ส่วนการออกแบบ UI นั้น เราเลือกใช้ Tailwind CSS ซึ่งเป็น utility-first CSS framework ที่ช่วยให้การจัดการสไตร์ล์ต่างๆ ง่ายและรวดเร็ว โดยไม่ต้องเขียน CSS ที่ซ้ำซ้อนหรือซับซ้อน ทำให้การพัฒนา Frontend ของโปรเจกต์มีความคล่องตัวและมี ประสิทธิภาพมากยิ่งขึ้น

สำหรับการพัฒนา Backend เราใช้ Node.js ร่วมกับ Express.js เพื่อสร้าง API endpoints ที่เชื่อมต่อ กับฐานข้อมูล โดยใช้ routes ในการดึงและส่งข้อมูลระหว่างเซิร์ฟเวอร์และ Frontend ซึ่งทำให้การจัดการข้อมูล และการตอบสนองของระบบมีประสิทธิภาพสูง

ในโปรเจกต์นี้ยังมีการใช้ JSON Web Token (JWT) เพื่อการยืนยันตัวตนและการควบคุมการเข้าถึงข้อมูล ของผู้ใช้ JWT ช่วยให้การจัดการ session ซึ่งทำให้สามารถตรวจสอบความถูกต้องของผู้ใช้ได้ง่ายและเร็วขึ้น โดย การสร้างและส่ง token ไปกับการตอบสนองจากเซิร์ฟเวอร์และให้ผู้ใช้ส่ง token กลับมาในแต่ละคำขอ (request) เพื่อยืนยันตัวตน และมีการใช้ reCAPTCHA ซึ่งเป็นเครื่องมือจาก Google ที่ช่วยในการป้องกันสแปมและการ โจมตีจากบอท โดยจะตรวจสอบว่าเป็นผู้ใช้จริงที่ทำการกรอกข้อมูลผ่านฟอร์มในเว็บไซต์หรือไม่ การใช้ reCAPTCHA เพิ่มความปลอดภัยให้กับแอปพลิเคชันโดยการป้องกันการโจมตีแบบบอทโนมัติ

Wireframe

1. Home Page



ในส่วนของ Home Page ประกอบด้วยสิ่งต่อไปนี้

- Header
- Navigation Bar เพื่อลิงค์ไปหน้าอื่น ๆ (รายละเอียดในข้อ 7)
- รูปภาพของอาหาร
- Footer

2. Menu Page

The screenshot shows the Sasuke Ramen menu page. At the top, there is a navigation bar with links for HOME, MENU, ABOUT US, CONTACT, and LOGIN. Below the navigation bar is the restaurant logo "Sasuke Ramen". A search bar is located at the top right. On the left side, there is a sidebar with a "Filter" section containing three dropdown menus: Rating (1★ to 5★★★★★), Spicy Level (Not spicy, 1🌶, 2🌶🌶), and Price Range (0 to 200). The main content area displays a grid of ramen bowls. Each bowl has a small green or red circular badge in the top right corner indicating its rating. The bowls are labeled with their names in Thai and English, along with their prices. The first row contains four bowls: "กุ้งเผา ไข่ゆ ราเมん" (Rating: ★★★★, Price: B108), "กุ้งเผา ราเมん" (Rating: ★★★, Price: B133), "ต้มยำ ชาชูมัน" (Rating: ★★★, Price: B108), and "ชาชูมัน" (Rating: ★★★★, Price: B83). The second row contains four more bowls: "ชาชูมัน" (Rating: ★★★★, Price: B108), "ต้มยำน้ำตก ราเม่น" (Rating: ★★★★, Price: B110), "มิโซะ ราเม่น" (Rating: ★★★★, Price: B100), and "มิโซะ ชาชูมัน" (Rating: ★★★★, Price: B115).

The screenshot shows the Sasuke Ramen menu page. At the top, there is a navigation bar with links for HOME, MENU, ABOUT US, CONTACT, and LOGIN. Below the navigation bar is the restaurant logo "Sasuke Ramen". A search bar is located at the top right. On the left side, there is a sidebar with a "Filter" section containing three dropdown menus: Rating (1★ to 5★★★★★), Spicy Level (Not spicy, 1🌶, 2🌶🌶), and Price Range (0 to 200). The main content area displays a grid of side dishes and desserts. Each item has a small green or red circular badge in the top right corner indicating its rating. The items are labeled with their names in Thai and English, along with their prices. The first row contains four items: "กุ้งเผา 6 ชิ้น" (Rating: ★★★★, Price: B96), "กุ้งเผา 12 ชิ้น" (Rating: ★★★★, Price: B108), "พิซซ่าไก่" (Rating: ★★★, Price: B53), and "พิซซ่า ชาชู" (Rating: ★★, Price: B53). The second row contains two items: "หนังวากิ ติ่มเป็ด" (Rating: ★★★, Price: B105) and "แซลมอนกรอบ" (Rating: ★★★★, Price: B83).

ในส่วนของ Menu Page ประกอบด้วยสิ่งต่อไปนี้

- Navigation Bar เพื่อลิงค์ไปหน้าอื่น ๆ (รายละเอียดในข้อ 7)
- Search Bar เพื่อ filter ข้อมูลสินค้า
- Filter เพื่อ filter ข้อมูลสินค้า
- ข้อมูลสินค้า
- Footer

3. About Us Page

The screenshot shows the 'About Us' page of a website for 'Sasuke Ramen'. At the top, there's a navigation bar with links for 'HOME', 'MENU', 'ABOUT US', and 'CONTACT', along with a 'LOGIN' button. Below the navigation is a large, dark image of a bowl of ramen with various toppings like sliced eggs and seaweed. To the right of this image is a section titled 'History' which includes a short paragraph of text and a small photo of a restaurant interior.

About Us

History

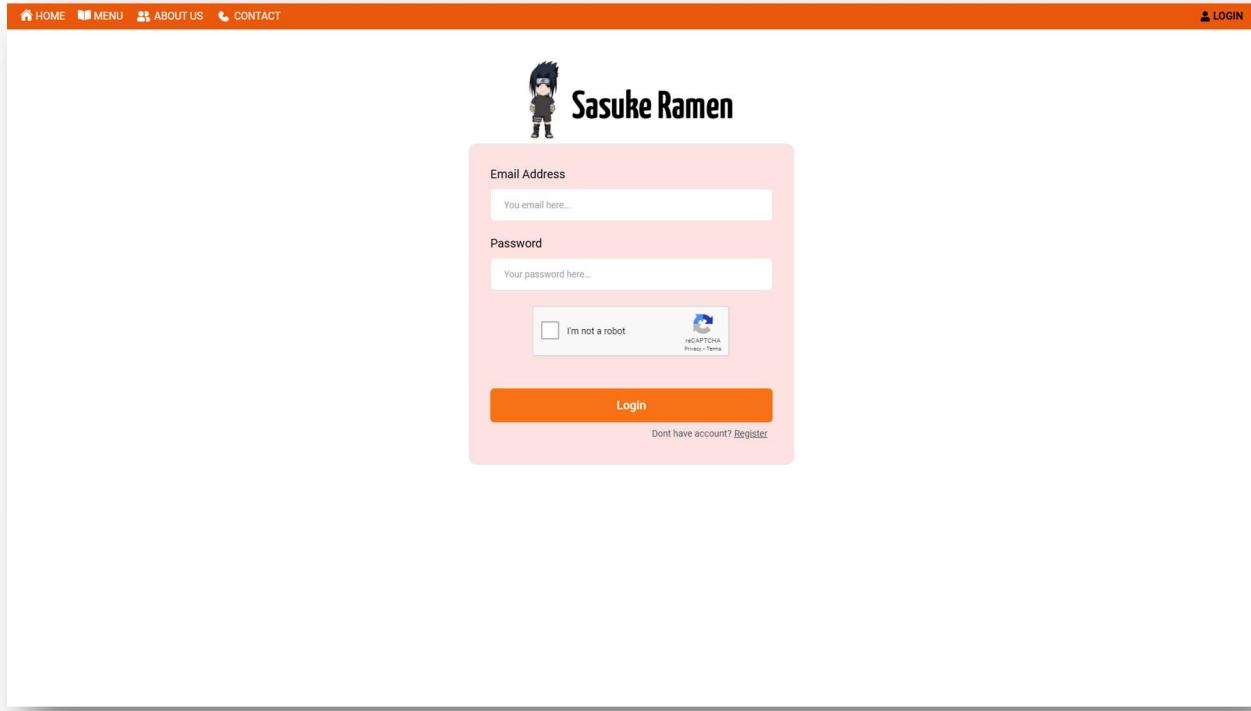
Sasuke Ramen ตั้งร้านอาหารที่มีน้ำใจรักเมืองพนมเพลศในปุณณ์ โดยครูดังแห่งนี้ ที่มาพร้อมกับความอร่อยและหัวใจที่ใส่มาให้กับอาหารที่มีอยู่ทุกจาน รวมถึง ต้นแบบเดิมของซุปเปอร์สุดยอด "Sasuke Ramen" เพื่อบรรลุภารกิจที่ต้องการให้ลูกค้าทุกคน ได้ลองลิ้มลองความอร่อยที่ต้องการมานานแล้ว ทุกจานอาหารที่เสิร์ฟขึ้นมา ล้วนเป็นเครื่องดูดใจที่ต้องการให้ลูกค้า ลิ้มลองและรักมากขึ้นเรื่อยๆ สำหรับเรา ในการให้บริการที่ดีเยี่ยมที่สุด ไม่ใช่แค่อาหาร แต่เป็นความรัก ความอบอุ่น ที่ต้องการจะสืบทอด下去 ให้ลูกค้าทุกท่าน ได้ลองลิ้มลองและรักมากขึ้นเรื่อยๆ

FOLLOW US ON [@](#) [f](#)

ในส่วนของ About Us page ประกอบด้วยสิ่งต่อไปนี้

- Navigation Bar เพื่อเลือกไปหน้าอื่น ๆ (รายละเอียดในข้อ 7)
- ประวัติของทางร้าน
- Footer

4. Login Page



ในส่วนของ Login page ประกอบด้วยสิ่งต่าง ๆ ดังนี้

- Navigation Bar เพื่อลิงค์ไปหน้าอื่น ๆ (รายละเอียดในข้อ 7)
- Form สำหรับกรอกข้อมูล
- ReCAPTCHA เพื่อยืนยันว่าไม่ใช่บอท
- ปุ่มสำหรับเข้าสู่ระบบ
- ปุ่มสำหรับลิงค์ไปหน้า Register

5. Register Page

The screenshot shows a web page for "Sasuke Ramen". At the top, there's a navigation bar with links for "HOME", "MENU", "ABOUT US", "CONTACT", and "LOGIN". Below the navigation is a logo featuring a character from the anime "Naruto" and the text "Sasuke Ramen". The main content area is a registration form with a pink background. It includes fields for "Email Address", "Password", "Confirm Password", "Firstname", "Lastname", "Date of Birth" (with a date input field and a calendar icon), "Tel.", and "Address". At the bottom of the form is a large orange "Register" button.

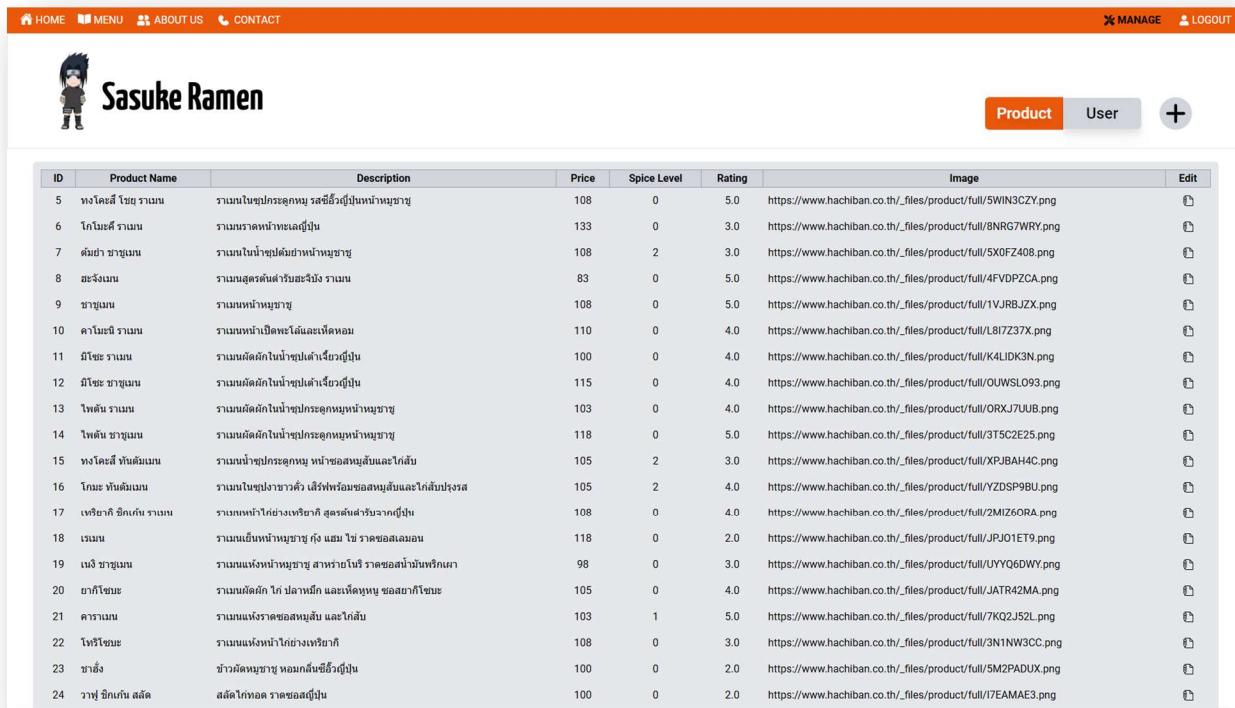
ในส่วนของ Register page ประกอบด้วยสิ่งต่าง ๆ ดังนี้

- Navigation Bar เพื่อลิงค์ไปหน้าอื่น ๆ (รายละเอียดในข้อ 7)
- Form สำหรับกรอกข้อมูล
- ปุ่ม Register เพื่อเพิ่มข้อมูลเข้าไปยัง Database

6. Management Page

ในส่วนของหน้า Management ต้องเป็น Admin เท่านั้นจึงสามารถเข้าหน้านี้ได้

6.1 Product Management



The screenshot shows a management interface for a ramen shop named "Sasuke Ramen". The top navigation bar includes links for Home, Menu, About Us, Contact, Manage, and Logout. The main content area displays a table of products with columns for ID, Product Name, Description, Price, Spice Level, Rating, Image, and Edit. The table lists 24 different ramen items, each with a unique ID, name, description, price, spice level, rating, image URL, and an edit icon.

ID	Product Name	Description	Price	Spice Level	Rating	Image	Edit
5	หงโคเจส์ ไข่крутายามัน	ราเม็นไข่крутายามัน ซอสมะเขือเทศเผ็ดร้อนมาก	108	0	5.0	https://www.hachiban.co.th/_files/product/full/5WIN3CZY.png	
6	โกโนเม็ต์ ราเม่น	ราเม็นคัตติ่งไก่เผ็ดร้อน	133	0	3.0	https://www.hachiban.co.th/_files/product/full/8NRG7WRY.png	
7	ผัดยำ ชาบูญี่ปุ่น	ราเม็นไข่เจียวเผ็ดร้อนมาก	108	2	3.0	https://www.hachiban.co.th/_files/product/full/5XOF240B.png	
8	แซลมอนน	ราเม็นแซลมอนเผ็ดร้อนไวๆ	83	0	5.0	https://www.hachiban.co.th/_files/product/full/4FVDPZCA.png	
9	ชาบูญี่ปุ่น	ราเม็นห้ามยาเสียหาย	108	0	5.0	https://www.hachiban.co.th/_files/product/full/1VJRBZJX.png	
10	คานิลี่ ราเม่น	ราเม็นไข่เจียวเผ็ดร้อนไวๆ	110	0	4.0	https://www.hachiban.co.th/_files/product/full/L8I7Z37X.png	
11	มีไซ ราเม่น	ราเม็นผัดคลีนไข่เจียวเผ็ดร้อน	100	0	4.0	https://www.hachiban.co.th/_files/product/full/K4LIDK3N.png	
12	มีไซ ชาบูญี่ปุ่น	ราเม็นผัดคลีนไข่เจียวเผ็ดร้อน	115	0	4.0	https://www.hachiban.co.th/_files/product/full/OUWSL093.png	
13	ไฟตัน ราเม่น	ราเม็นผัดคลีนไข่เจียวเผ็ดร้อนมาก	103	0	4.0	https://www.hachiban.co.th/_files/product/full/ORXJ7UUB.png	
14	ไฟตัน ชาบูญี่ปุ่น	ราเม็นผัดคลีนไข่เจียวเผ็ดร้อนมาก	118	0	5.0	https://www.hachiban.co.th/_files/product/full/3T5C2E25.png	
15	หงโคเจส์ ทาร์ต宦南น	ราเม็นไข่крутายามัน หน้าซอสมะเขือเทศเผ็ดร้อน	105	2	3.0	https://www.hachiban.co.th/_files/product/full/XPJBAH4C.png	
16	โกโน เมนต์宦南น	ราเม็นไข่крутายามัน เต็มไปด้วยซอสมะเขือเทศเผ็ดร้อน	105	2	4.0	https://www.hachiban.co.th/_files/product/full/YZDPPBUJ.png	
17	เบร์ยาก ชิงเก้น ราเม่น	ราเม็นหน้าไข่เจียวเผ็ดร้อน ซอสมะเขือเทศเผ็ดร้อน	108	0	4.0	https://www.hachiban.co.th/_files/product/full/2MIZ6ORA.png	
18	เนนน	ราเม็นเน็นหน้าหมูย่าง กุ้ง และ ไข่ ราดซอสมะลูบ	118	0	2.0	https://www.hachiban.co.th/_files/product/full/JPJO1ET9.png	
19	เนน ชาบูญี่ปุ่น	ราเม็นเน็นหน้าหมูย่าง สาหร่ายโรย ราดซอสมะลูบกับไข่เค็ม	98	0	3.0	https://www.hachiban.co.th/_files/product/full/UYYQ6DWY.png	
20	มาเกะชิม	ราเม็นผัดคลีนไข่เจียวเผ็ดร้อนมาก ซอสมะเขือเทศ	105	0	4.0	https://www.hachiban.co.th/_files/product/full/JATR42MA.png	
21	คาราเม่น	ราเม็นแห้งราดซอสมะลูบ และไข่ลวก	103	1	5.0	https://www.hachiban.co.th/_files/product/full/7KQ2J52L.png	
22	โนริเม่น	ราเม็นแห้งหน้าไก่ย่างเผ็ดร้อน	108	0	3.0	https://www.hachiban.co.th/_files/product/full/3INW3CC.png	
23	ชาสึ	ชาบูผัดหมูขาว หมูสามชั้นเผ็ดร้อน	100	0	2.0	https://www.hachiban.co.th/_files/product/full/5M2PAUDUX.png	
24	ราฟฟิเก้น สลัด	สลัดไก่สด ซอสมะลูบเผ็ดร้อน	100	0	2.0	https://www.hachiban.co.th/_files/product/full/I7EAMAE3.png	

ในส่วนของ Product Management ประกอบด้วยสิ่งต่อไปนี้

- Navigation Bar เพื่อล็อกอินเข้าสู่ระบบ
- ปุ่มสำหรับสลับหน้าระหว่าง Product Management หรือ User Management
- ปุ่มสำหรับเพิ่มรายการสินค้า
- ข้อมูลสินค้าทั้งหมดที่อยู่ใน Database
- ปุ่มสำหรับแก้ไขสินค้า

6.1.1 Add Product Modal

The screenshot shows a web application interface for managing a ramen shop. At the top, there's a navigation bar with links for HOME, MENU, ABOUT US, and CONTACT, along with MANAGEMENT and LOGOUT buttons. The main header features a cartoon character and the text "Sasuke Ramen". Below the header is a table listing various ramen products with columns for ID, Product Name, and Description. In the center, a modal window titled "Add Product" is open, containing input fields for Name, Description, Price, Rating, Spicy Level, and Image, each with a placeholder text. At the bottom of the modal are "Cancel" and "Confirm" buttons. To the right of the modal, a table lists existing products with columns for ID, Name, Description, Price, Rating, Spicy Level, and Image, with an "Edit" link next to each row.

ID	Product Name	Description
5	หงโ律เดี๋ยวๆ ราเมน	ราเมนในไก่กระอกหมู ซอซึชิวะญี่ปุ่นหนานุญาญี่ปุ่น
6	โกโนเมต์ ราเมน	ราเมนราด้าห้ามเหลือง
7	ผัดชา ราฐมน	ราเมนบนน้ำชาอีสานเผาหัวหมากญี่ปุ่น
8	กระเจนนน	ราเมนต้มตุ๋นคั่วชิวะญี่ปุ่น
9	ชาญมน	ราเมนห้ามเผาญี่ปุ่น
10	คานินี้ ราเมน	ราเมนห้ามเผาเพื่อให้เป็นสีเขียวญี่ปุ่น
11	มีไซ ราเมน	ราเมนผัดกิ้นน้ำซุปเดี๋ยวๆ
12	มีไซ ชาญมน	ราเมนผัดกิ้นน้ำซุปเดี๋ยวๆ ญี่ปุ่น
13	ไฟล์นี้ ราเมน	ราเมนผัดกิ้นน้ำซุปกระอกหมูหัวหมากญี่ปุ่น
14	ไฟล์นี้ ชาญมน	ราเมนผัดกิ้นน้ำซุปกระอกหมูหัวหมากญี่ปุ่น
15	หงโ律เดี๋ยวห้ามบ่น	ราเมนบนไก่กระอกหมูหัวหมากญี่ปุ่น
16	โกน หันห้ามบ่น	ราเมนบนไก่กระอกหมูหัวหมากญี่ปุ่น เดี๋ยวที่ร้อนของลูกน้ำเด็กและไข่ลับไปร้อน
17	เบร์เกอร์ วิงกัน ราเมน	ราเมนห้ามเผาไว้ร้อนเบร์เกอร์ ซอสผ่านฟาร์มวากิญี่ปุ่น
18	เนนน	ราเมนเนยหัวหมากญี่ปุ่น เมนู ไฟ ขาดซอสมะลูน
19	เนน ชาญมน	ราเมนแห้งหัวหมากญี่ปุ่น สาหร่ายโนริ ราดซอสน้ำมันพริกเผา
20	ชา ก็ใจช	ราเมนผัดกิ้น กิ้น ปลาร้าเผา และเตือกหมู ซอสสาเกใจช
21	คานมน	ราเมนแห้งราดซอสมะลูน และกิ้นลัน
22	ไฟริชช	ราเมนแห้งหัวไก่เผาไฟริชช
23	ชาลีส	ชาวดหูหมากญี่ปุ่นเดี๋ยวๆ
24	ราชา อินกัน สลัด	สลัดไก่ทอด ราดซอสญี่ปุ่น

ในส่วนของ Add Product Modal ประกอบด้วยสิ่งต่อไปนี้

- ปุ่มสำหรับบันทึกการเพิ่มสินค้า
- Form สำหรับกรอกข้อมูล
- ปุ่มสำหรับการยกเลิกการเพิ่มสินค้า

6.1.2 Edit Product Modal

HOME	MENU	ABOUT US	CONTACT	MANAGE	LOGOUT
 Sasuke Ramen					
ID	Product Name	Description	Image	Edit	
5	ฟางโคลีฟี่ ไข่ ราเมน	ราเมนในถ้วยกระถางหมุน ซอซิชิริปูนหน้าหมูชาชู	https://www.hachiban.co.th/_files/product/full/5WIN3CZY.png		
6	ไก่เผือก ราเมน	ราเมนเคราฟ์ไก่เผือกญี่ปุ่น	https://www.hachiban.co.th/_files/product/full/5NRPG7WRY.png		
7	ต้มยำ ชาบูมัน	ราเมนในถ้วยกระถางหมุนหน้าหมูชาชู	https://www.hachiban.co.th/_files/product/full/5XOFZ40B.png		
8	สะจิลัน	ราเมนสูตรเด็ดต้มยำชี้มิ้ง ราเมน	https://www.hachiban.co.th/_files/product/full/4FVDPZCA.png		
9	ชาบูมัน	ราเมนหน้าหมูชาชู	https://www.hachiban.co.th/_files/product/full/1VJRJBZK.png		
10	คานิล่า ราเมน	ราเม็นหน้าเป็นเพลทและไฟหอยแมลงภู่	https://www.hachiban.co.th/_files/product/full/8J737X.png		
11	มีโซะ ราเมน	ราเมนตัดผักในถ้วยปลั๊กเต้าเจี๊ยบญี่ปุ่น	https://www.hachiban.co.th/_files/product/full/K4LIDK3N.png		
12	มีโซะ ชาบูมัน	ราเมนตัดผักในถ้วยปลั๊กเต้าเจี๊ยบญี่ปุ่น	https://www.hachiban.co.th/_files/product/full/OUWSL093.png		
13	ไฟเผา ราเมน	ราเมนตัดผักในถ้วยกระถางหมุนหน้าหมูชาชู	https://www.hachiban.co.th/_files/product/full/0RJKJUU8.png		
14	ไฟเผา ชาบูมัน	ราเมนตัดผักในถ้วยกระถางหมุนหน้าหมูชาชู	https://www.hachiban.co.th/_files/product/full/3TSC2E25.png		
15	ฟางโคลีฟี่ ห้ามลิ้มหนึ่น	ราเมนหน้าไข่กระถางหมุน หน้าของสองถ้วยแล้วก้าวลิ้น	https://www.hachiban.co.th/_files/product/full/XPJBAH4C.png		
16	ไก่เผา ห้ามลิ้มหนึ่น	ราเมนในถ้วยปลาคาร์ด เต้าเจี๊ยบรวมซอสมะเขือเทศและไก่ลิ้มพริกเผา	https://www.hachiban.co.th/_files/product/full/YZDSP9BU.png		
17	เนริยา ข้าวคั่ว ราเมน	ราเม็นหน้าไก่คั่วเผาเต้าเจี๊ยบ สูตรเด็ดต้มยำชาชู	https://www.hachiban.co.th/_files/product/full/2M7Z60RA.png		
18	เรบิน	ราเมนหน้าหมูชาชู กุ้ง แคน ไข่ ราชเทวสถานหนึ่งเดียว	https://www.hachiban.co.th/_files/product/full/JPJ01ET9.png		
19	เนริ ชาบูมัน	ราเม็นหน้าหมูชาชู สาหร่ายโนริ ราชเทวสถานหน้าหมูชาชู	https://www.hachiban.co.th/_files/product/full/UYYQ6DWY.png		
20	ถากิโซะนะ	ราเม็นตัดผัก ไก่ มั่นมาก และเนื้อหมู ซอสยาเกิร์ชแบบญี่ปุ่น	https://www.hachiban.co.th/_files/product/full/JATR42MA.png		
21	คาราเมน	ราเม็นหน้ากระเทียมหน้าหมู หน้าเผา	https://www.hachiban.co.th/_files/product/full/7KQ2J52L.png		
22	ไฟริโซะนะ	ราเม็นแพลงก์หน้าไก่เผาเต้าเจี๊ยบ	https://www.hachiban.co.th/_files/product/full/3N1NW3CC.png		
23	ชาบูสึ	ชาบูเดือนพฤษภาคม ซอสยาเกิร์ชญี่ปุ่น	https://www.hachiban.co.th/_files/product/full/5M2PADUX.png		
24	ราชา กี๊กคั่ว สลัด	สลัดคั่กออล ราชเทวสถานหนึ่งเดียว	https://www.hachiban.co.th/_files/product/full/17EAMAE3.png		

ในส่วนของ Edit Product Modal ประกอบด้วยสิ่งต่าง ๆ ดังนี้

- ปุ่มสำหรับยืนยันการแก้ไขสินค้า
 - Form สำหรับกรอกข้อมูล
 - ปุ่มสำหรับการยกเลิกการแก้ไขสินค้า
 - ปุ่มสำหรับการลบสินค้า

6.2 User Management

The screenshot shows a web application interface for managing users. At the top, there is a navigation bar with links: HOME, MENU, ABOUT US, and CONTACT. To the right of the navigation bar are buttons for MANAGE, LOGOUT, and a user profile icon. The main header features a cartoon character icon and the text "Sasuke Ramen". Below the header, there are two tabs: "Product" (grayed out) and "User" (highlighted in orange). A small circular button with a plus sign is located to the right of the tabs. The central part of the page is a table listing user data:

ID	Firstname	Lastname	Date of Birth	Phone	Address	Email	Password	isAdmin	Edit
2	Siriprapas	Kimpee	11/11/1111	123124		siriprapas.kim@student.mahidol.edu	Image0123456	✓	edit
25	Varich	Maleevan	11/20/2004	0846454399	asgjhngwa	varich.mal@student.mahidol.edu	taetar2016	✓	edit
31	User	User	11/11/2000			user@user.com	user	✗	edit
32	Admin	Admin	11/11/2000			admin@admin.com	admin	✓	edit

ในส่วนของ User Management ประกอบด้วยสิ่งต่าง ๆ ดังนี้

- Navigation Bar เพื่อเลือกไปหน้าอื่น ๆ (รายละเอียดในข้อ 7)
- ปุ่มสำหรับสลับหน้าระหว่าง Product Management หรือ User Management
- ปุ่มสำหรับเพิ่ม User
- ข้อมูล User ทั้งหมดที่อยู่ใน Database
- ปุ่มสำหรับแก้ไข User

6.2.1 User Add Modal

The screenshot shows the Sasuke Ramen website interface. At the top, there is a navigation bar with links for HOME, MENU, ABOUT US, and CONTACT. On the right side of the header, there are links for MANAGE, LOGOUT, Product, User, and a plus sign icon for adding new items.

The main content area features a logo of a ramen character and the text "Sasuke Ramen". Below this is a table listing users with columns for ID, Firstname, Lastname, and Date of Birth. The table contains four rows of data:

ID	Firstname	Lastname	Date of Birth
2	Siriprapas	Kimpee	11/11/1111
25	Varich	Maleevan	11/20/2000
31	User	User	11/11/2000
32	Admin	Admin	11/11/2000

A modal window titled "Add User" is displayed in the center. It contains fields for First Name, Last Name, Date of Birth (with a date picker), Phone Number, Email, Password, Address, and a dropdown for "Is Admin?". The "Is Admin?" dropdown has "No" selected. At the bottom of the modal are "Cancel" and "Confirm" buttons.

To the right of the modal, there is a table titled "User" with columns for Password, isAdmin, and Edit. It lists four users with their respective details:

Password	isAdmin	Edit
Image0123456	✓	edit icon
taetar2016	✓	edit icon
user	✗	edit icon
admin	✓	edit icon

ในส่วนของ Add User Modal ประกอบด้วยสิ่งต่อไปนี้

- ปุ่มสำหรับบันทึกการเพิ่ม User
- Form สำหรับกรอกข้อมูล
- ปุ่มสำหรับการยกเลิกการเพิ่ม User

6.2.2 User Edit Modal

The screenshot shows the Sasuke Ramen website's user management interface. At the top, there is a navigation bar with links for HOME, MENU, ABOUT US, and CONTACT. On the right side of the header, there are links for MANAGE, LOGOUT, Product, and User, along with a plus sign icon for adding new users.

The main content area features a logo of a ramen character and the text "Sasuke Ramen". Below this is a table listing users with columns for ID, Firstname, Lastname, and Date of Birth. The table contains four rows of data:

ID	Firstname	Lastname	Date of Birth
2	Siriprapas	Kimpee	11/11/1111
25	Varich	Maleevan	11/20/2000
31	User	User	11/11/2000
32	Admin	Admin	11/11/2000

A modal window titled "Edit User" is open in the center. It contains fields for First Name (Siriprapas), Last Name (Kimpee), Date of Birth (11/11/1111), Phone Number (123124), Email (siriprapas.kim@student.mahidol.edu), Password (*****), Address (empty), and Is Admin? (Yes). There are "Cancel" and "Confirm" buttons at the bottom of the modal.

To the right of the modal, there is a table titled "User" with columns for Password, isAdmin, and Edit. It lists four users:

Password	isAdmin	Edit
Image0123456	✓	edit icon
taetar2016	✓	edit icon
user	✗	edit icon
admin	✓	edit icon

ในส่วนของ Edit User Modal ประกอบด้วยสิ่งต่อไปนี้

- ปุ่มสำหรับบันทึกการแก้ไข User
- Form สำหรับกรอกข้อมูล
- ปุ่มสำหรับการยกเลิกการแก้ไข User
- ปุ่มสำหรับการลบ User

7. Navigation Bar

ในส่วนของ Navigation Bar ประกอบไปด้วย 3 ส่วน

7.1 No account Navigation Bar



ในส่วนของ Navigation Bar ที่ไม่มีการ Login ประกอบด้วยสิ่งต่าง ๆ ดังนี้

- ปุ่ม HOME เพื่อนำไปหน้า Home Page
- ปุ่ม MENU เพื่อนำไปหน้า Menu Page
- ปุ่ม ABOUT US เพื่อนำไปหน้า About Us Page
- ปุ่ม CONTACT เพื่อนำไปหน้า Contact Page
- ปุ่ม LOGIN เพื่อนำไปหน้า Home Page

7.2 User Navigation Bar



ในส่วนของ User Navigation Bar ประกอบด้วยสิ่งต่าง ๆ ดังนี้

- ปุ่ม HOME เพื่อนำไปหน้า Home Page
- ปุ่ม MENU เพื่อนำไปหน้า Menu Page
- ปุ่ม ABOUT US เพื่อนำไปหน้า About Us Page
- ปุ่ม CONTACT เพื่อนำไปหน้า Contact Page
- ปุ่ม LOGOUT เพื่อ Logout และนำไปหน้า Home

7.2 Admin Navigation Bar

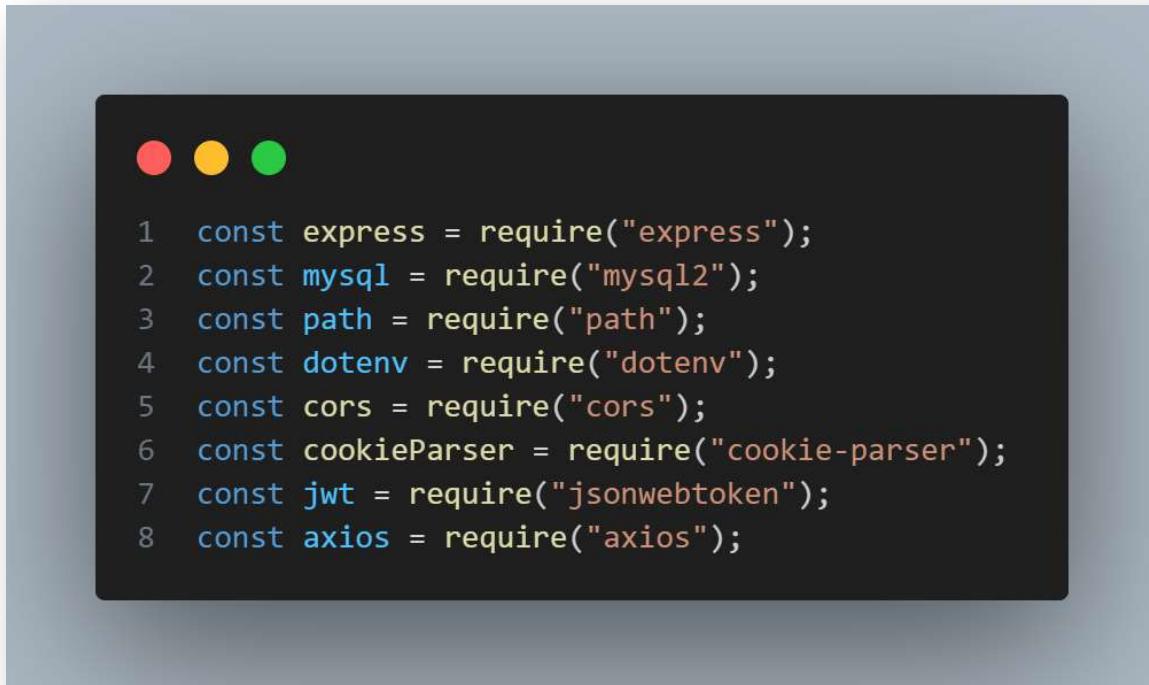


ในส่วนของ User Navigation Bar ประกอบด้วยสิ่งต่าง ๆ ดังนี้

- ปุ่ม HOME เพื่อนำไปหน้า Home Page
- ปุ่ม MENU เพื่อนำไปหน้า Menu Page
- ปุ่ม ABOUT US เพื่อนำไปหน้า About Us Page
- ปุ่ม CONTACT เพื่อนำไปหน้า Contact Page
- ปุ่ม MANAGE เพื่อนำไปหน้า Management Page
- ปุ่ม LOGOUT เพื่อ Logout และนำกลับไปหน้า Home Page

Web Service

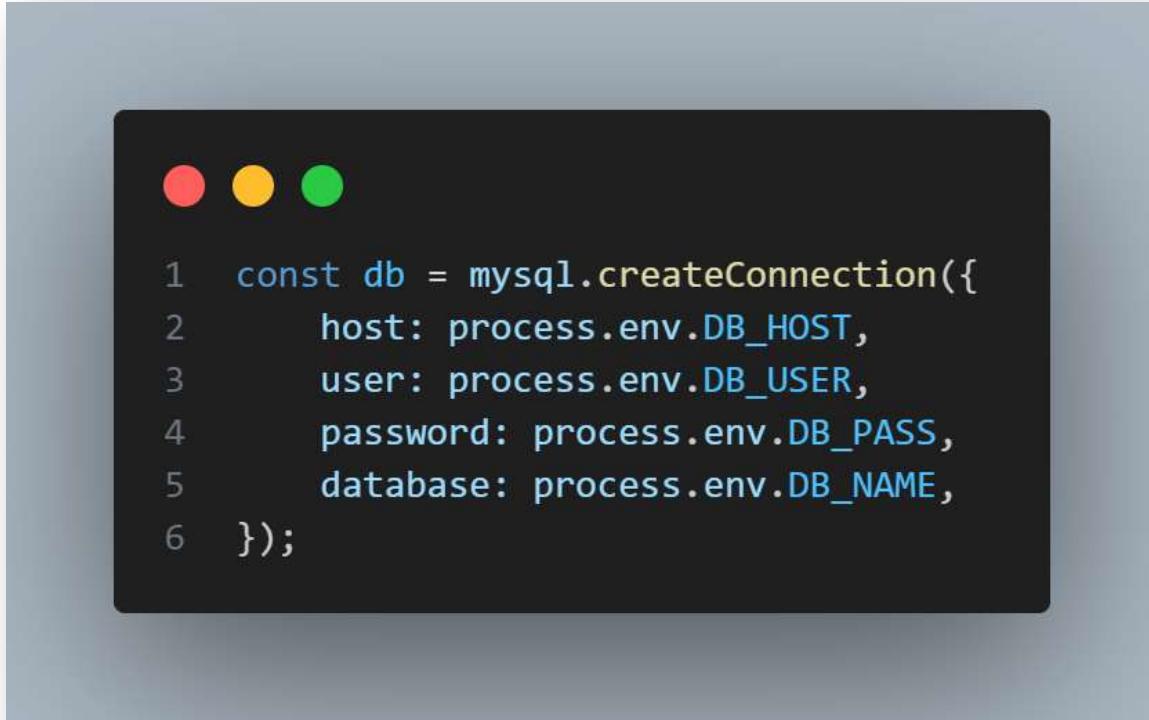
1. Modules



```
1 const express = require("express");
2 const mysql = require("mysql2");
3 const path = require("path");
4 const dotenv = require("dotenv");
5 const cors = require("cors");
6 const cookieParser = require("cookie-parser");
7 const jwt = require("jsonwebtoken");
8 const axios = require("axios");
```

- **express:** ใช้เพื่อสร้างเว็บเซอร์วิส (API) ที่ให้บริการการรับและส่งข้อมูล
- **mysql2:** ใช้เพื่อเชื่อมต่อและทำงานกับฐานข้อมูล MySQL
- **dotenv:** ใช้เพื่อโหลดค่าต่าง ๆ ที่กำหนดในไฟล์ .env ซึ่งมักจะเก็บข้อมูลที่เกี่ยวกับการตั้งค่าระบบ เช่น DB_HOST, DB_USER, และ DB_PASS โดยไม่ต้องระบุค่าเหล่านี้โดยตรงในโค้ด
- **cors:** ใช้เพื่อเปิดให้การสื่อสารจากโดเมนอื่น (Cross-Origin Resource Sharing) สามารถเข้าถึง API ได้
- **cookie-parser:** ใช้ในการจัดการคุกกี้จากคำร้องขอ
- **jsonwebtoken:** ใช้ในการสร้างและตรวจสอบ JSON Web Token (JWT) สำหรับการพิสูจน์ตัวตนและการรักษาความปลอดภัย
- **axios:** ใช้ในการทำคำร้องขอ HTTP ไปยัง API หรือเซอร์วิสอื่นๆ

2. Database Connection



```
● ● ●
1 const db = mysql.createConnection({
2     host: process.env.DB_HOST,
3     user: process.env.DB_USER,
4     password: process.env.DB_PASS,
5     database: process.env.DB_NAME,
6 });
```

ส่วนนี้คือการตั้งค่าการเชื่อมต่อกับฐานข้อมูล MySQL โดยใช้ข้อมูลที่ได้จากไฟล์ .env เช่น โ伊斯ต์, ชื่อผู้ใช้, รหัสผ่าน และชื่อฐานข้อมูล
หลังจากนั้นการเชื่อมต่อกับฐานข้อมูลจะถูกทดสอบด้วย:



```
● ● ●
1 db.connect((err) => {
2     if (err) throw err;
3     console.log(`Database Connected on : ${process.env.DB_NAME}`);
4 });
```

ถ้าเชื่อมต่อได้สำเร็จ จะมีข้อความ "Database Connected on : ชื่อฐานข้อมูล" แสดงขึ้นมาบน Console

3. Express Setup



```
● ● ●
1 const app = express();
2 app.use(express.urlencoded({ extended: true }));
3 app.use(cookieParser());
4 app.use(express.json());
5 app.use(
6   cors({
7     origin: ["http://localhost:3000", "localhost:3000"],
8     credentials: true,
9   })
10 );
```

- `cors()`: เพื่อนำเข้า API ที่ต้องการจากโดเมนอื่น
- `cookieParser()`: ใช้ในการแปลงข้อมูลคุกกี้ในคำร้องขอเป็นอ็อบเจกต์ที่สามารถใช้งานได้
- `express.json()`: ใช้เพื่อให้สามารถรับข้อมูลในรูปแบบ JSON ใน body ของ request ได้
- `express.urlencoded()`: ทำให้ Express สามารถแปลงข้อมูลจากฟอร์ม (form data) ที่ถูกส่งใน POST request ให้เป็นอ็อบเจกต์ใน req.body ซึ่งสามารถใช้ข้อมูลนี้ต่อไปในแอปพลิเคชันได้
- `extended: true`: หมายความว่าเราจะใช้ qs library เพื่อแปลงข้อมูลที่มีโครงสร้างซับซ้อน เช่น อาร์เรย์หรืออ็อบเจกต์ในฟอร์ม ส่งผ่าน URL-encoded ข้อมูลออกมาได้อย่างถูกต้อง

4. Routes

a. User Register



```
1 app.post("/register", (req, res) => {
2   const query =
3     "INSERT INTO Users (Fname, Lname, DoB, Phone, Address, Email, Password) VALUES (?, ?, ?, ?, ?, ?, ?)";
4   const { fname, lname, dob, phone, address, email, password } = req.body;
5
6   db.query(
7     query,
8     [fname, lname, dob, phone, address, email, password],
9     (err, result) => {
10       if (err) return res.status(500).json({ error: err });
11       res.json({ message: "Register successfully", result: result });
12     }
13   );
14 });

15 
```

1. รับข้อมูลจาก req.body
2. สร้างคำสั่ง SQL
3. เรียกใช้งาน db.query: ฟังก์ชัน db.query จะทำการเชื่อมต่อฐานข้อมูล MySQL และรันคำสั่ง SQL
4. หากมีข้อผิดพลาด (err) ระบบจะแจ้งข้อผิดพลาดกลับไปยังผู้ใช้ผ่าน หากคำสั่งสำเร็จจะส่ง response กลับไปยัง Front-end

b. User login & generate token

```
1 // User login & generate token
2 app.post("/login", async (req, res) => {
3   const query =
4     "SELECT Email, Password, isAdmin FROM Users WHERE Email = ? AND Password = ?";
5   const { email, password, recaptchaToken } = req.body;
6
7   try {
8     const verifyURL = `https://www.google.com/recaptcha/api/siteverify?secret=${process.env.RECAPTCHA_SECRET}&response=${recaptchaToken}`;
9     const { data } = await axios.post(verifyURL);
10
11    if (!data.success) {
12      return res.status(400).json({ error: "Invalid reCAPTCHA token" });
13    }
14  } catch (error) {
15    console.error("reCAPTCHA validation error:", error);
16    return res.status(500).json({ error: "reCAPTCHA validation failed" });
17  }
18
19  db.query(query, [email, password], (err, result) => {
20    if (err) return res.status(500).json({ error: err });
21    if (result.length === 0)
22      return res.status(404).json({ error: "No user found" });
23
24    const token = jwt.sign(
25      { email, isAdmin: result[0].isAdmin },
26      process.env.TOKEN_PASS,
27      {
28        expiresIn: "12h",
29      }
30    );
31    res.cookie("key", token);
32    res.json({ message: "Generate key successfully!", result: result[0] });
33  });
34});
```

1. รับข้อมูลจาก req.body
2. ตรวจสอบ reCAPTCHA: ใช้ axios เพื่อส่งคำขอ POST ไปยัง API ของ Google reCAPTCHA ที่ URL <https://www.google.com/recaptcha/api/siteverify> โดย ส่งค่า secret (ซึ่งได้จากการตั้งค่าใน Google reCAPTCHA Console) และ response (ซึ่งเป็นโทเค็นจากฟอร์ม) ในคำขอเพื่อยืนยันว่าเป็นการยืนยันโดยมนุษย์
3. เชื่อมต่อฐานข้อมูล MySQL: เพื่อดึงข้อมูลผู้ใช้จากฐานข้อมูลที่มีอีเมลและรหัสผ่าน ตรงกับที่ผู้ใช้กรอกมา
4. สร้าง JWT Token: เมื่อการตรวจสอบสำเร็จ (อีเมลและรหัสผ่านถูกต้อง) จะมีการ สร้าง JSON Web Token (JWT) เพื่อให้ผู้ใช้สามารถใช้มันในการเข้าถึงข้อมูลที่ ต้องการในส่วนอื่นๆ ของระบบ ซึ่งจะมีการส่ง token นี้กลับไปยังผู้ใช้เพื่อใช้ในการเข้า สู่ระบบในอนาคต
5. การตอบกลับ: ถ้าการเข้าสู่ระบบสำเร็จ จะส่งข้อมูลที่เกี่ยวข้องกับการเข้าสู่ระบบหรือ การสร้าง JWT token

c. Check Token

```
1 app.post("/checktoken", (req, res) => {
2   const token = req.cookies.key;
3
4   if (!token) return res.status(401).json({ message: "Session Expired" });
5
6   jwt.verify(token, process.env.TOKEN_PASS, (err, result) => {
7     if (err) return res.status(500).json({ error: err });
8     if (!result)
9       return res
10      .status(403)
11      .json({ message: "เปลี่ยน Token ทำไม่ครับที่รัก" });
12   return res
13   .status(200)
14   .json({ message: "มีผ่าน ยินดีด้วย", result: result });
15 });
16});
```

1. ดึง Token จาก Cookies: ดึง JWT token ที่ถูกเก็บไว้ในคุกกี้ โดยคุกกี้นั้นมีชื่อว่า key
2. ตรวจสอบว่า Token มีอยู่หรือไม่: ถ้าไม่มี Token จะส่งกลับการตอบกลับ HTTP status 401 พร้อมข้อความ "Session Expired" เพื่อบอกว่าผู้ใช้ไม่มีการเข้าสู่ระบบ หรือ Session หมดอายุแล้ว
3. ตรวจสอบความถูกต้องของ Token:
 1. ถ้ามี Token จะใช้ jwt.verify() เพื่อตรวจสอบความถูกต้องของ Token โดยใช้ Secret Key ที่เก็บในตัวแปร process.env.TOKEN_PASS ซึ่งเป็นคีย์สำหรับ การตรวจสอบ Token หากการตรวจสอบไม่สำเร็จ เช่น Token หมดอายุ, ปลอมแปลง, หรือไม่ตรงกับ Secret Key
 - a. จะตอบกลับด้วย HTTP status 500 และข้อผิดพลาดที่เกิดขึ้น
 - b. หาก jwt.verify() ไม่สามารถยืนยันผลลัพธ์ได้ (Token ไม่ถูกต้อง) จะตอบกลับด้วย HTTP status 403 และข้อความ "เปลี่ยน Token ทำไม่ครับที่รัก"
4. ถ้า Token ถูกต้องและยังไม่หมดอายุ จะตอบกลับด้วย HTTP status 200 และ ข้อความ "Token Verified" พร้อมกับผลลัพธ์ที่ได้รับจากการตรวจสอบ Token

d. User Logout

```
1 app.post("/logout", (req, res) => {
2     res.clearCookie("key");
3     res.json({ message: "Logged out successfully" });
4 });
```

- ลบคุกกี้ที่ชื่อว่า key ซึ่งเก็บ JWT token ที่ใช้ในการตรวจสอบตัวตนของผู้ใช้
- หลังจากที่ลบคุกกี้แล้วจะตอบกลับด้วย HTTP status 200 และข้อความ “Logged out successfully”

e. Add Product

```
1 app.post("/addproduct", (req, res) => {
2     const query =
3         "INSERT INTO Products (Pname, Desp, Price, Star, Spice, Img) VALUES (?, ?, ?, ?, ?, ?)";
4     const { pname, desp, price, star, spice, img } = req.body;
5     db.query(query, [pname, desp, price, star, spice, img], (err, result) => {
6         if (err) return res.status(500).json({ error: err });
7         res.json({ message: "Product added successfully" });
8     });
9 });
```

- รับข้อมูลที่ส่งมาจากการผู้ใช้ผ่าน req.body
- สร้างคำสั่ง SQL เพื่อใช้ในการแทรกข้อมูลใหม่เข้าไปในตาราง Products
- ทำการบันทึกข้อมูลลงในฐานข้อมูล
 - หากมีข้อผิดพลาดจะตอบกลับด้วยสถานะ HTTP 500 พร้อมข้อความแสดงข้อผิดพลาด
 - หากคำสั่ง SQL สำเร็จ จะตอบกลับข้อความ "Product added successfully" ในรูปแบบ JSON เพื่อยืนยันว่าได้เพิ่มสินค้าลงในฐานข้อมูลเรียบร้อยแล้ว

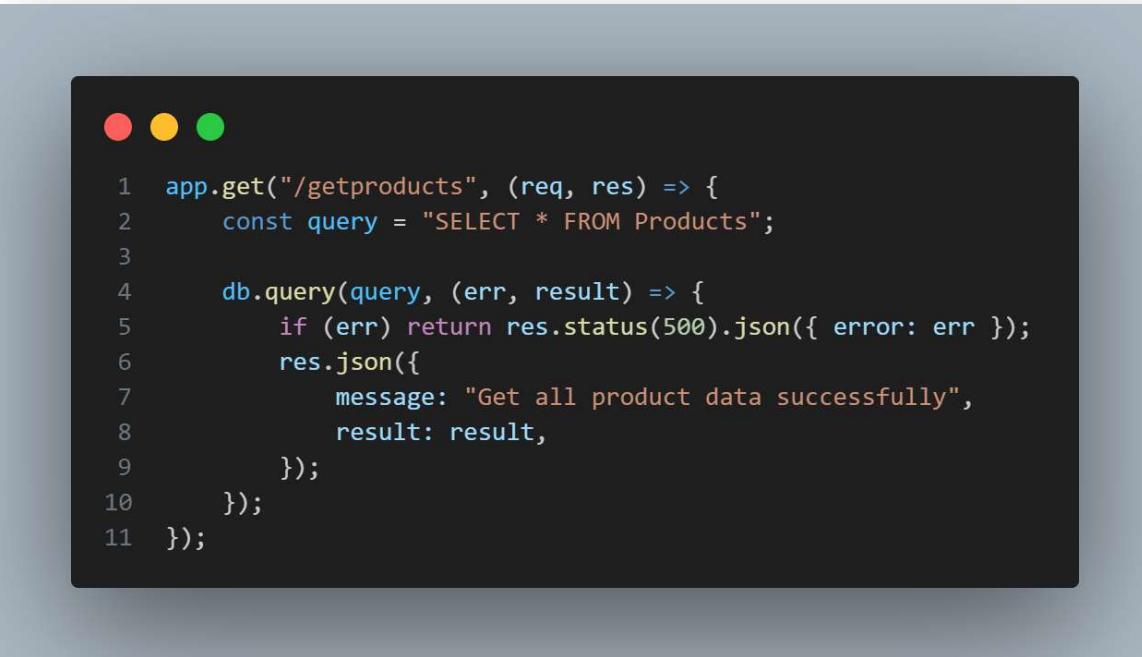
f. Add User

```
1 app.post("/adduser", (req, res) => {
2   const query =
3     "INSERT INTO Users (Fname, Lname, Dob, Phone, Address, Email, Password, isAdmin) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
4   const { fname, lname, dob, phone, address, email, password, isAdmin } =
5     req.body;
6   db.query(
7     query,
8     [fname, lname, dob, phone, address, email, password, isAdmin],
9     (err, result) => {
10       if (err) return res.status(500).json({ error: err });
11       res.json({ message: "User added successfully" });
12     }
13   );
14 });


```

1. รับข้อมูลจาก req.body
2. สร้างคำสั่ง SQL เพื่อเพิ่มข้อมูลผู้ใช้ใหม่ในตาราง Users
3. ใช้ db.query() เพื่อดำเนินการคำสั่ง SQL ในฐานข้อมูล MySQL
4. ตรวจสอบผลลัพธ์:
 1. หากเกิดข้อผิดพลาดในกระบวนการเพิ่มผู้ใช้ จะมีการตอบกลับเป็น status 500 พร้อมข้อความแสดงข้อผิดพลาด (error)
 2. หากการเพิ่มผู้ใช้สำเร็จ, ระบบจะส่งการตอบกลับเป็น status 200 พร้อมข้อความ "User added successfully" เพื่อยืนยันว่าได้เพิ่มผู้ใช้ใหม่ลงในฐานข้อมูลเรียบร้อยแล้ว

g. Get all products



```
1 app.get("/getproducts", (req, res) => {
2     const query = "SELECT * FROM Products";
3
4     db.query(query, (err, result) => {
5         if (err) return res.status(500).json({ error: err });
6         res.json({
7             message: "Get all product data successfully",
8             result: result,
9         });
10    });
11});
```

1. สร้างคำสั่ง SQL เพื่อดึงข้อมูลทั้งหมดจากตาราง Products ซึ่งจะได้ข้อมูลทุกคอลัมน์ ในตาราง
2. รันคำสั่ง SQL บนฐานข้อมูล MySQL และรับผลลัพธ์ที่ได้จากฐานข้อมูลมาเก็บในตัวแปร result
3. หากคำสั่ง SQL สำเร็จ ระบบจะส่งข้อมูลสินค้าทั้งหมดที่ดึงมาจากฐานข้อมูลกลับไปยังผู้ใช้ในรูปแบบ JSON

h. Get product by ID

```
● ● ●  
1 app.get("/getproduct/:id", (req, res) => {  
2   const query = "SELECT * FROM Products WHERE PID = ?";  
3   const id = req.params.id;  
4  
5   db.query(query, [id], (err, result) => {  
6     if (err) return res.status(500).json({ error: err });  
7     res.json({  
8       message: `Get product data | ID: ${id} Name: ${result[0].Pname} | successfully`,  
9       result: result,  
10    });  
11  });  
12});
```

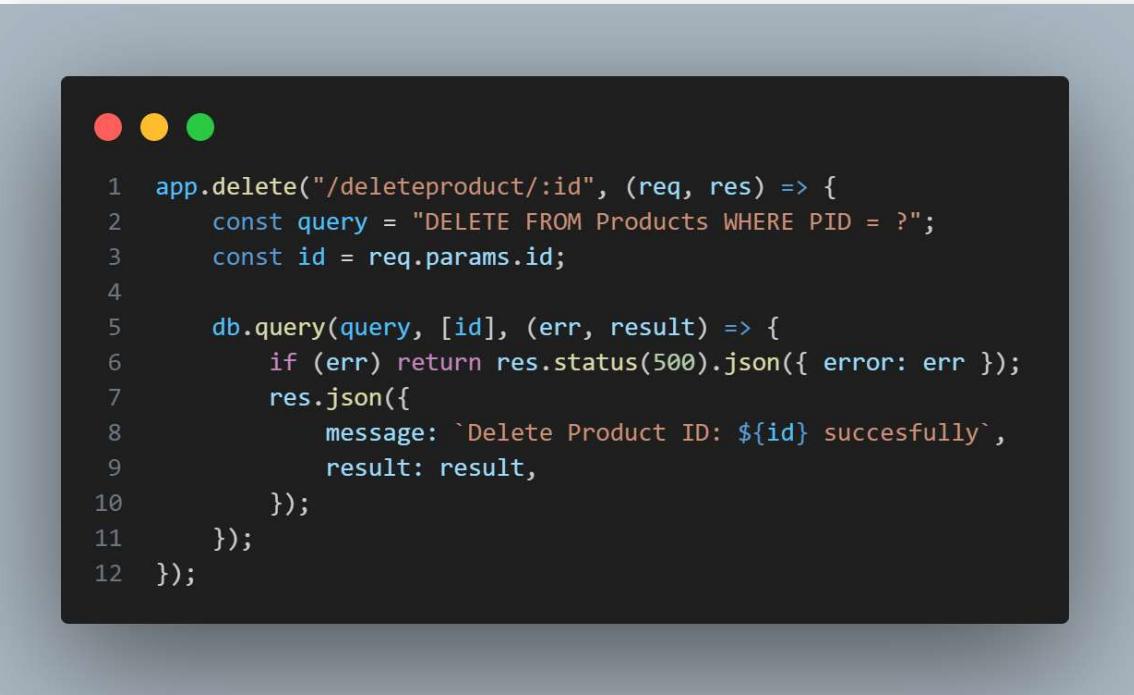
1. รับค่า id จาก URL เพื่อดึงค่าของ id ที่ส่งมาจาก URL โดยใช้พารามิเตอร์ :id
2. สร้างคำสั่ง SQL เพื่อดึงข้อมูลสินค้าเฉพาะตัวจากตาราง Products ที่มีค่า PID ตรงกับค่า id ที่ได้รับ
3. ทำการ query กับฐานข้อมูล:
 1. ถ้ามีข้อผิดพลาดในระหว่างการค้นหาข้อมูล จะตอบกลับสถานะ 500 พร้อมข้อความข้อผิดพลาด
 2. ถ้าข้อมูลถูกดึงมาได้สำเร็จ จะส่งข้อมูลกลับไปในรูปแบบ JSON พร้อมข้อความว่า "Get product data | ID: {id} Name: {product name} | successfully" และข้อมูลสินค้าที่ดึงมา

i. Update product by ID

```
1 app.put("/updateproduct/:id", (req, res) => {
2   const query =
3     "UPDATE Products SET Pname = ?, Desp = ?, Price = ?, Star = ?, Spice = ?, Img = ? WHERE PID = ?";
4   const { Pname, Desp, Price, Star, Spice, Img } = req.body;
5   const id = req.params.id;
6
7   db.query(
8     query,
9     [Pname, Desp, Price, Star, Spice, Img, id],
10    (err, result) => {
11      if (err) return res.status(500).json({ error: err });
12      res.json({
13        message: `Update product | ID: ${id} Name: ${Pname} | successfully`,
14        result: result,
15      });
16    }
17  );
18});
```

1. รับข้อมูลจาก req.body
2. ใช้คำสั่ง SQL เพื่ออัปเดตข้อมูลสินค้าตาม PID ที่ระบุใน URL
3. ถ้ามีข้อผิดพลาดในการดำเนินการ คำตอบจะมีสถานะ 500 และข้อความผิดพลาด ถ้าข้อมูลถูกดึงมาได้สำเร็จ จะส่งข้อมูลกลับไปในรูปแบบ JSON พร้อมข้อความว่า "Update product | ID: {id} Name: {product name} | successfully" และข้อมูลสินค้าที่ดึงมา

j. Delete product by ID



```
1 app.delete("/deleteproduct/:id", (req, res) => {
2   const query = "DELETE FROM Products WHERE PID = ?";
3   const id = req.params.id;
4
5   db.query(query, [id], (err, result) => {
6     if (err) return res.status(500).json({ error: err });
7     res.json({
8       message: `Delete Product ID: ${id} successfully`,
9       result: result,
10    });
11  });
12});
```

1. ดึงค่า id จาก URL
2. ส่งคำสั่ง SQL DELETE เพื่อทำการลบข้อมูลสินค้า
3. ถ้าการลบสำเร็จ ระบบจะตอบกลับด้วยข้อความสำเร็จ พร้อมข้อมูล result

k. Get all users

```
1 app.get("/getusers", (req, res) => {
2   const query =
3     "SELECT UID, Fname, Lname, DATE_FORMAT(DoB, '%Y-%m-%d') AS DoB, Phone, Address, Email, Password, isAdmin FROM Users";
4
5   db.query(query, (err, result) => {
6     if (err) return res.status(500).json({ error: err });
7     res.json({ message: "Get all user data successfully", result: result });
8   });
9 });
```

- สร้างคำสั่ง SQL ที่ดึงข้อมูลจากตาราง Users, แต่จะทำการปรับรูปแบบวันที่ (DoB) โดยใช้ DATE_FORMAT เพื่อให้ผลลัพธ์วันที่แสดงในรูปแบบ YYYY-MM-DD
- รัน db.query() เพื่อส่งคำสั่ง SQL ไปยังฐานข้อมูล MySQL
- ถ้ามีข้อผิดพลาดในการดึงข้อมูล จะส่งกลับข้อความผิดพลาดด้วยรหัสสถานะ 500 ถ้าคำสั่ง SQL สำเร็จ จะส่งข้อมูลผู้ใช้ทั้งหมดที่ได้จากการดึงข้อมูลกลับไปยังผู้ใช้ในรูปแบบ JSON

I. Get user by ID



```
1 app.get("/getuser/:id", (req, res) => {
2   const query =
3     "SELECT UID, Fname, Lname, DATE_FORMAT(DoB, '%Y-%m-%d') AS DoB, Phone, Address, Email, Password, isAdmin FROM Users WHERE UID = ?";
4   const id = req.params.id;
5
6   db.query(query, [id], (err, result) => {
7     if (err) return res.status(500).json({ error: err });
8     res.json({
9       message: `Get user data | ID: ${id} Name: ${result[0].Fname} ${result[0].Lname} | successfully`,
10      result: result,
11    });
12  });
13});
```

- รับค่า id จาก URL เพื่อดึงค่าของ id ที่ส่งมาจาก URL โดยใช้พารามิเตอร์ :id
- คำสั่ง SQL นี้จะเลือกข้อมูลของผู้ใช้จากตาราง Users โดยใช้ UID ที่ตรงกับค่า id ที่ส่งมาใน URL. นอกจากนี้ยังใช้ฟังก์ชัน DATE_FORMAT เพื่อแปลงวันที่เกิด (DoB) ให้อยู่ในรูปแบบ YYYY-MM-DD
- ทำการเรียกคำสั่ง SQL และส่งค่าของ id ไปในฐานข้อมูล
- หากมีข้อผิดพลาดในการดึงข้อมูล (เช่น การเชื่อมต่อฐานข้อมูลล้มเหลว) ระบบจะส่งข้อความผิดพลาดกลับไปยังผู้ใช้ หากคำขอสำเร็จ ระบบจะส่งข้อมูลผู้ใช้ที่ดึงมาได้ในรูปแบบ JSON พร้อมข้อความที่ระบุว่าเรียกข้อมูลผู้ใช้ได้สำเร็จ

m. Update user by ID



```
1 app.put("/updateuser/:id", (req, res) => {
2     const query =
3         "UPDATE Users SET Fname = ?, Lname = ?, DoB = ?, Phone = ?, Email = ?, Password = ?, isAdmin = ? WHERE UID = ?";
4
5     const { Fname, Lname, DoB, Phone, Email, Password, isAdmin } = req.body;
6     const id = req.params.id;
7
8     db.query(
9         query,
10        [Fname, Lname, DoB, Phone, Email, Password, isAdmin, id],
11        (err, result) => {
12            if (err) return res.status(500).json({ error: err });
13            res.json({
14                message: `Update user | ID: ${id} Name: ${Fname} ${Lname} | successfully`,
15                result: result,
16            });
17        }
18    );
19 });
20
```

1. รับข้อมูลจาก req.body
2. ดึงค่า id ของผู้ใช้จาก URL โดยใช้พารามิเตอร์ :id
3. คำสั่ง SQL นี้จะอัปเดตข้อมูลของผู้ใช้ในตาราง Users โดยใช้ค่าใหม่จาก req.body และจะอัปเดตแค่ผู้ใช้ที่มี UID ตรงกับ id
4. ทำการส่งคำสั่ง SQL พร้อมค่าที่จะอัปเดตไปยังฐานข้อมูล หากมีข้อผิดพลาดในคำสั่ง SQL จะส่งสถานะ 500 และข้อความผิดพลาด หากอัปเดตสำเร็จจะส่งข้อความยืนยันกลับไป

n. Delete user by ID

```
1 app.delete("/deleteuser/:id", (req, res) => {
2   const query = "DELETE FROM Users WHERE UID = ?";
3   const id = req.params.id;
4
5   db.query(query, [id], (err, result) => {
6     if (err) return res.status(500).json({ error: err });
7     res.json({
8       message: `Delete user ID: ${id} successfully`,
9       result: result,
10    });
11  });
12});
```

1. ดึงค่าของ id ที่ส่งมาจาก URL โดยใช้พารามิเตอร์ :id
2. คำสั่ง SQL นี้จะลบแถว (row) ในตาราง Users ที่มีค่า UID ตรงกับ id ที่ผู้ใช้ส่งมา
3. ถ้ามีข้อผิดพลาดในการดำเนินการ (err), จะส่งกลับสถานะ 500 พร้อมข้อความแสดงข้อผิดพลาด ถ้าการลบสำเร็จ, จะส่งกลับสถานะ 200 พร้อมข้อความ Delete user ID: \${id} successfully และผลลัพธ์ที่อยู่ในตัวแปร result

5. Start Server

```
● ● ●  
1 app.listen(process.env.PORT, (err) => {  
2   if (err) throw err;  
3   console.log(`Server is running on port: ${process.env.PORT}`);  
4 });
```

ในส่วนนี้คือการตั้งค่าให้แอปพลิเคชัน Express ฟังคำร้องขอบนพอร์ตที่กำหนดไว้ใน .env

Test with Postman

User Register URL: <http://localhost:3001/register>

The screenshot shows a Postman request to `localhost:3001/register` using the `POST` method. The `Body` tab contains the following JSON payload:

```
1 {
2     "fname": "test1",
3     "lname": "test",
4     "dob": "2012-07-04",
5     "phone": "0987654321",
6     "address": "asof",
7     "email": "test1@test.com",
8     "password": "test"
9 }
```

The response status is `200 OK` with a response time of 15 ms and a body size of 443 B. The response body is:

```
1 {
2     "message": "Register successfully",
3     "result": {
4         "fieldCount": 0,
5         "affectedRows": 1,
6         "insertId": 36,
7         "info": "",
8         "serverStatus": 2,
9         "warningStatus": 0,
10        "changedRows": 0
11    }
12 }
```

User Login (ນໍາ ReCAPTCHA ອອກ) URL: <http://localhost:3001/login>

The screenshot shows a Postman request to `http://localhost:3001/login` using the `POST` method. The `Body` tab contains the following JSON payload:

```
1 {
2     "email": "test1@test.com",
3     "password": "test"
4 }
```

The response status is `200 OK` with a response time of 37 ms and a body size of 596 B. The response body is:

```
1 {
2     "message": "Logged in successfully!",
3     "result": {
4         "Email": "test1@test.com",
5         "Password": "test",
6         "isAdmin": 0
7     }
8 }
```

Token Check URL: <http://localhost:3001/checktoken>

The screenshot shows a Postman request to `http://localhost:3001/checktoken`. The request method is POST, and the body is empty. The response status is 200 OK, with a response time of 10 ms, a content length of 400 B, and a session duration of 10 ms. The response body is a JSON object:

```
1 {  
2   "message": "Token Verified",  
3   "result": {  
4     "email": "test@test.com",  
5     "isAdmin": 0,  
6     "iat": 1732954799,  
7     "exp": 1732997999  
8   }  
9 }
```

The screenshot shows a Postman request to `http://localhost:3001/checktoken`. The response includes a cookie named `key` with the value `eyJhbGciOiJIUzI1NiIsInR5cCI6I...` . The cookie is set for the domain `localhost` and has a path of `/`, an expiration of `Session`, `false` for `HttpOnly`, and `false` for `Secure`.

Name	Value	Domain	Path	Expires	HttpOnly	Secure
key	eyJhbGciOiJIUzI1NiIsInR5cCI6I...	localhost	/	Session	false	false

User Logout URL: <http://localhost:3001/logout>

The screenshot shows a Postman request to `http://localhost:3001/logout`. The request method is POST, and the body is empty. The response status is 200 OK, with a response time of 14 ms, a content length of 391 B, and a session duration of 14 ms. The response body is a JSON object:

```
1 {  
2   "message": "Logged out successfully"  
3 }
```

Add Product URL: <http://localhost:3001/addproduct>

The screenshot shows a Postman request for `http://localhost:3001/addproduct`. The request method is `POST`. The body is set to `raw JSON` with the following payload:

```
1 {
2     "pname": "testproduct",
3     "desp": "asdfasfasfa",
4     "price": 2000000,
5     "star": 5,
6     "spice": 0,
7     "img": "asdfasfasdf"
8 }
```

The response status is `200 OK` with a response time of 29 ms and a size of 329 B. The response body is:

```
1 {
2     "message": "Product added successfully"
3 }
```

Add User URL: <http://localhost:3001/adduser>

The screenshot shows a Postman request for `http://localhost:3001/adduser`. The request method is `POST`. The body is set to `raw JSON` with the following payload:

```
1 {
2     "fname": "test2",
3     "lname": "testest",
4     "dob": "2000-01-01",
5     "phone": "0999998888",
6     "address": "asdfs",
7     "email": "test2@test.com",
8     "password": "test",
9     "isAdmin": 0
10 }
```

The response status is `200 OK` with a response time of 32 ms and a size of 326 B. The response body is:

```
1 {
2     "message": "User added successfully"
3 }
```

Get all Products

URL: <http://localhost:3001/getproducts>

Postman screenshot showing the results of a GET request to <http://localhost:3001/getproducts>. The response is a 200 OK with a total time of 19 ms and a size of 7.97 KB. The JSON body contains a message "Get all product data successfully" and a list of products (PID 5, 6, 7, 8) with their details: Pname, Desp, Price, Spice, Star, and Img.

```
1 {
2   "message": "Get all product data successfully",
3   "result": [
4     {
5       "PID": 5,
6       "Pname": "ໜຳເລີດ ໂອງ ລາຍນ",
7       "Desp": "ກົມພະນັກງານຂອງລາຍນ ລັບລົງທຶນນິ້ນມາຫຼາຍ",
8       "Price": 100,
9       "Spice": 0,
10      "Star": "5.0",
11      "Img": "https://www.hachiban.co.th/_files/product/full/5WIN3CZY.png"
12    },
13    {
14      "PID": 6,
15      "Pname": "ເຖິງເພື່ອ ລາຍນ",
16      "Desp": "ກົມພະນັກງານຂອງລາຍນ ລັບລົງທຶນນິ້ນມາຫຼາຍ",
17      "Price": 133,
18      "Spice": 0,
19      "Star": "3.0",
20      "Img": "https://www.hachiban.co.th/_files/product/full/8NRG7wRY.png"
21    },
22    {
23      "PID": 7,
24      "Pname": "ໜຳ ແກ້ວມາ",
25      "Desp": "ກົມພະນັກງານຂອງລາຍນ ລັບລົງທຶນນິ້ນມາຫຼາຍ",
26      "Price": 100,
27      "Spice": 2,
28      "Star": "3.0",
29      "Img": "https://www.hachiban.co.th/_files/product/full/5X0FZ488.png"
30    },
31    {
32      "PID": 8,
33      "Pname": "ຂອງລາຍນ",
34    }
35  ]
36 }
```

Get product by ID

URL: <http://localhost:3001/getproduct/:id>

Postman screenshot showing the results of a GET request to <http://localhost:3001/getproduct/5>. The response is a 200 OK with a total time of 12 ms and a size of 732 B. The JSON body contains a message "Get product data | ID: 5 Name: ຜຳເລີດ ໂອງ ລາຍນ | successfully," and a single product (PID 5) with its details: Pname, Desp, Price, Spice, Star, and Img.

```
1 {
2   "message": "Get product data | ID: 5 Name: ຜຳເລີດ ໂອງ ລາຍນ | successfully",
3   "result": [
4     {
5       "PID": 5,
6       "Pname": "ໜຳເລີດ ໂອງ ລາຍນ",
7       "Desp": "ກົມພະນັກງານຂອງລາຍນ ລັບລົງທຶນນິ້ນມາຫຼາຍ",
8       "Price": 100,
9       "Spice": 0,
10      "Star": "5.0",
11      "Img": "https://www.hachiban.co.th/_files/product/full/5WIN3CZY.png"
12    }
13  ]
14 }
```

Update Product

URL: <http://localhost:3001/updateproduct/:id>

The screenshot shows a POST request to <http://localhost:3001/updateproduct/37>. The body is a JSON object:

```
PUT http://localhost:3001/updateproduct/37
{
  "Pname": "testupdateproduct",
  "Desp": "asdasdf",
  "Price": 28,
  "Spice": 0,
  "Star": 6.0,
  "Img": "asdfdasdfdasdf"
}
```

The response status is 200 OK, with a response time of 25 ms, size of 523 B, and a detailed message: "Update product | ID: 37 Name: testupdateproduct | successfully".

Delete Product

URL: <http://localhost:3001/deleteproduct/:id>

The screenshot shows a DELETE request to <http://localhost:3001/deleteproduct/37>. The query parameters are:

Key	Value	Description
Key	Value	Description

The response status is 200 OK, with a response time of 16 ms, size of 454 B, and a detailed message: "Delete Product ID: 37 succesfully".

Get all users URL: <http://localhost:3001/getusers>

GET http://localhost:3001/getusers

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	...

Body Cookies Headers (9) Test Results ⚡

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Get all user data successfully",
3   "result": [
4     {
5       "UID": 2,
6       "Fname": "Siriprapas",
7       "Lname": "Kimpree",
8       "Dob": "1111-11-11",
9       "Phone": "123124",
10      "Address": "",
11      "Email": "siriprapas.kim@student.mahidol.edu",
12      "Password": "Image0123456",
13      "isAdmin": 1
14    },
15    {
16      "UID": 25,
17      "Fname": "Varich",
18      "Lname": "Maleavan",
19      "Dob": "2804-11-28",
20      "Phone": "0846454399",
21      "Address": "asgjhngawa",
22      "Email": "varich.mal@student.mahidol.edu",
23      "Password": "taetaz2016",
24      "isAdmin": 1
25    },
26    {
27      "UID": 31,
28      "Fname": "User",
29      "Lname": "User",
30      "Dob": "2808-11-11",
31      "Phone": "",
32      "Address": "",
33      "Email": "user@user.com",
34      "Password": "user"
35    }
36  ]
37 }
```

200 OK 13 ms 1.28 KB ⚡ Save Response

Get User by ID URL: <http://localhost:3001/getuser/:id>

GET http://localhost:3001/getuser/36

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	...

Body Cookies Headers (9) Test Results ⚡

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Get user data | ID: 36 Name: test2 testest | successfully",
3   "result": [
4     {
5       "UID": 36,
6       "Fname": "test2",
7       "Lname": "testest",
8       "Dob": "2808-01-01",
9       "Phone": "0999999888",
10      "Address": "asdif",
11      "Email": "test@test.com",
12      "Password": "test",
13      "isAdmin": 0
14    }
15  ]
16 }
```

200 OK 13 ms 529 B ⚡ Save Response

Update User URL: <http://localhost:3001/updateuser/:id>

The screenshot shows a POST request to <http://localhost:3001/updateuser/36>. The request body is a JSON object containing user information:

```
1 {
2   "Fname": "test2",
3   "Lname": "testestestestest",
4   "DOB": "2000-01-01",
5   "Phone": "0988776666",
6   "Email": "test2@test.com",
7   "Password": "test",
8   "Address": "asdi",
9   "isAdmin": 1
10 }
```

The response status is 200 OK, with a response time of 34 ms and a size of 525 B. The response body is:

```
1 {
2   "message": "Update user | ID: 36 Name: test2 testestestestest | successfully",
3   "result": {
4     "fieldCount": 0,
5     "affectedRows": 1,
6     "insertId": 0,
7     "info": "Rows matched: 1  Changed: 1  Warnings: 0",
8     "serverStatus": 2,
9     "warningStatus": 0,
10    "changedRows": 1
11  }
12 }
```

Delete User URL: <http://localhost:3001/deleteuser/:id>

The screenshot shows a DELETE request to <http://localhost:3001/deleteuser/36>. The request body is a JSON object with a single key 'Key' set to 'Value':

Key	Value	Description	Bulk Edit
Key	Value	Description	...

The response status is 200 OK, with a response time of 16 ms and a size of 451 B. The response body is:

```
1 {
2   "message": "Delete user ID: 36 sucessfully",
3   "result": {
4     "fieldCount": 0,
5     "affectedRows": 1,
6     "insertId": 0,
7     "info": "",
8     "serverStatus": 2,
9     "warningStatus": 0,
10    "changedRows": 0
11  }
12 }
```

แหล่งอ้างอิง

- <https://www.hachiban.co.th/th/home/>