



รายงาน  
โครงการระดับที่ 2 การพัฒนาเว็บแอปพลิเคชัน dokmai4u

จัดทำโดย  
ธรรนากร คันศร 6787104  
ปกรณ์ นิมนานวล 6787051  
นิธิโขติ ไชยสิทธิ์ 6787048  
ศุภกิตติ สุวรรณ 6787080

เสนอ  
ผศ. ดร. จิตาภา ไกรสังข์  
อาจารย์ ดร. วุฒิชาติ แสงผล

รายงานนี้เป็นส่วนหนึ่งของรายวิชา  
ITDS242 Web Technologies Lab  
Semester 1/2025 Faculty of ICT, Mahidol University

## คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา ITDS242 Web Technologies Lab มีวัตถุประสงค์เพื่อพัฒนาเว็บแอปพลิเคชันและเว็บเซอร์วิสจากแบบจำลองที่ออกแบบไว้ในรายวิชา ITDS241 โดยระบบแบ่งออกเป็นส่วนหน้า (Front-end) ที่พัฒนาด้วย HTML, CSS และ JavaScript และส่วนหลังบ้าน (Back-end) ที่พัฒนาด้วย Node.js พร้อมเข้มต่อฐานข้อมูลและเว็บเซอร์วิสทั้งภายในและภายนอก โครงการนี้ช่วยฝึกทักษะการเขียนโปรแกรม การเชื่อมต่อระบบ และการพัฒนาเว็บแอปพลิเคชันให้สามารถใช้งานได้จริง

## สารบัญ

คำนำ	๗
สารบัญ	๘
สารบัญรูป	๙
บทที่ 1 คำอธิบายธุรกิจ	๑
บทที่ 2 Wireframe Design	๒
บทที่ 3 รายละเอียดหน้าเว็บต่าง ๆ ของผู้ดูแล	๙
บทที่ 4 รายละเอียดของเว็บเซอร์วิสและโค้ด	๑๘
4.1 account.controller.js	๑๘
4.2 auth.controller.js	๒๔
4.3 index.controller.js	๒๖
4.4 product.controller.js	๒๗
บทที่ 5 ผลการทดสอบของเว็บเซอร์วิส	๓๓
5.1 Login	๓๓
5.2 Account	๓๕
5.3 Product	๔๑
5.4 User Visit	๔๗
ข้อความเปิดเผยการใช้เทคโนโลยีปัญญาประดิษฐ์	๔๙
แหล่งอ้างอิง	๕๐

## สารบัญรูป

รูปที่ 2.1 Wireframe หน้า Home Page	2
รูปที่ 2.2 Wireframe หน้า Search Page	2
รูปที่ 2.3 Wireframe หน้า Detail Page	3
รูปที่ 2.4 Wireframe หน้า Login Page	3
รูปที่ 2.5 Wireframe หน้า Team Page	4
รูปที่ 2.6 Wireframe หน้า User Account Management Page	4
รูปที่ 2.7 Wireframe หน้า Add User Account Page	5
รูปที่ 2.8 Wireframe หน้า Update Information Account Page	5
รูปที่ 2.9 Wireframe หน้า Delete Account Page	6
รูปที่ 2.10 Wireframe หน้า Product Management Page	6
รูปที่ 2.11 Wireframe หน้า Add Product Page	7
รูปที่ 2.12 Wireframe หน้า Update Product Page	7
รูปที่ 2.13 Wireframe หน้า Delete Product Page	8
รูปที่ 3.1 Login Admin Page	9
รูปที่ 3.2 Product Management Page	10
รูปที่ 3.3 Product Add Page	11
รูปที่ 3.4 Product Update Page	12
รูปที่ 3.5 Product Delete Page	13
รูปที่ 3.6 Account Management Page	14
รูปที่ 3.7 Account Add Page	15
รูปที่ 3.8 Account Update Page	16
รูปที่ 3.9 Account Delete Page	17
รูปที่ 4.1.1 add admin	18
รูปที่ 4.1.2 Code add admin	18
รูปที่ 4.1.3 Update admin	19
รูปที่ 4.1.4 Code Update admin	19
รูปที่ 4.1.5 delete admin	20
รูปที่ 4.1.6 Code delete admin	20

## สารบัญรูป

รูปที่ 4.1.7 get all admin	21
รูปที่ 4.1.8 Code get all admin	21
รูปที่ 4.1.9 search admin	22
รูปที่ 4.1.10 Code search admin	22
รูปที่ 4.1.11 search by ID admin	23
รูปที่ 4.1.12 Code search by ID admin	23
รูปที่ 4.2.1 Login admin	24
รูปที่ 4.2.2 Code Login admin	24
รูปที่ 4.2.3 Login admin	25
รูปที่ 4.2.4 Code Logout admin	25
รูปที่ 4.3.1 Detail	26
รูปที่ 4.3.2 Code Detail	26
รูปที่ 4.4.1 Search all Product for User	27
รูปที่ 4.4.2 Code Search all Product for User	27
รูปที่ 4.4.3 Add Product	28
รูปที่ 4.4.4 Code Add Product	28
รูปที่ 4.4.5 Update Product	29
รูปที่ 4.4.6 Code Update Product	29
รูปที่ 4.4.7 Delete Product	30
รูปที่ 4.4.8 Code Delete Product	30
รูปที่ 4.4.9 Search Product by name for admin	31
รูปที่ 4.4.10 Code Search Product for admin	31
รูปที่ 4.4.11 Search Product by ID for admin	32
รูปที่ 4.4.12 Code Search by Product for admin	32

## สารบัญรูป

รูปที่ 5.1.1 Test Admin Login	33
รูปที่ 5.1.2 Test Admin Logout	34
รูปที่ 5.2.1 Test API to get account by ID	35
รูปที่ 5.2.2 Test API to get all accounts	36
รูปที่ 5.2.3 Test API to search accounts	37
รูปที่ 5.2.4 Test API to add new account	38
รูปที่ 5.2.5 Test API to update account by ID	39
รูปที่ 5.2.6 Test API to delete account by ID	40
รูปที่ 5.3.1 Test API to get product by ID	41
รูปที่ 5.3.2 Test API to get all products	42
รูปที่ 5.3.3 Test API to search products	43
รูปที่ 5.3.4 Test API to add new product	44
รูปที่ 5.3.5 Test API to update product by ID	45
รูปที่ 5.3.6 Test API to delete product by ID	46
รูปที่ 5.4.1 Test API to get product detail by ID	47
รูปที่ 5.4.2 Test API to search products	48

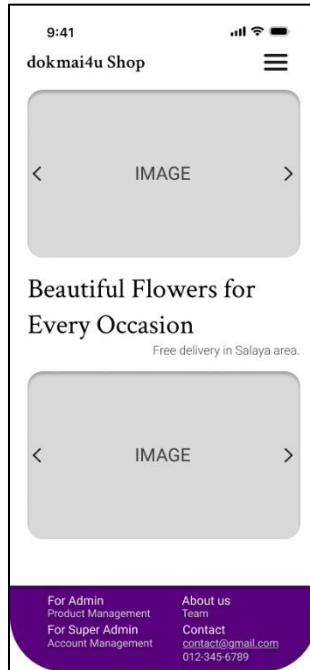
## บทที่ 1

### คำอธิบายธุรกิจ

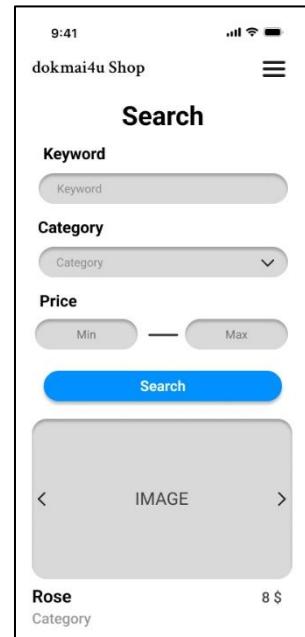
dokmai4u.com เป็นเว็บไซต์ขายดอกไม้ออนไลน์ที่ออกแบบมาให้ใช้งานง่ายบนทุกอุปกรณ์ โดยเฉพาะโทรศัพท์มือถือ สินค้าหลักประกอบด้วยช่อดอกไม้ พร้อมทั้งมีบริการกระเช้าและพวงดอกไม้สำหรับโอกาสต่าง ๆ เว็บไซต์ถูกพัฒนาภายใต้แนวคิดที่ต้องการให้ผู้ใช้สามารถเลือกดอกไม้ได้อย่างสะดวก รวดเร็ว และตรงกับความต้องการ นอกจากการนำเสนอสินค้าแล้ว dokmai4u ยังมีข้อมูลความหมายของดอกไม้ เพื่อช่วยให้ลูกค้าสามารถตัดสินใจเลือกสิ่งที่เหมาะสมสำหรับการมอบให้ครั้ง คุณใกล้ชิด หรือในเทศกาลสำคัญ จุดเด่นของเว็บไซต์คือการใช้งานที่ไม่ซับซ้อน การออกแบบในรูปแบบ mobile-first และบริการจัดส่งพรีวิวในพื้นที่ที่กำหนด เพื่อสร้างประสบการณ์ที่คุ้มค่าและประทับใจแก่ผู้ใช้บริการ

## บทที่ 2

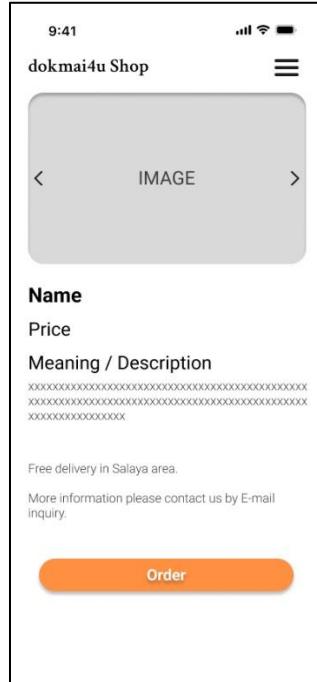
### Wireframe Design



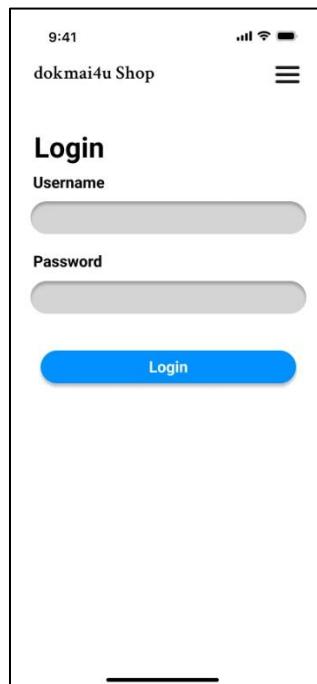
รูปที่ 2.1 Wireframe หน้า Home Page



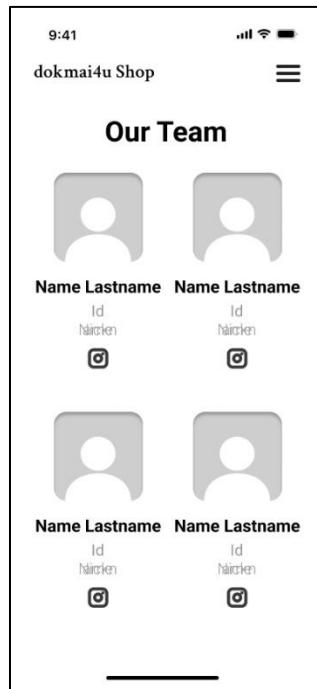
รูปที่ 2.2 Wireframe หน้า Search Page



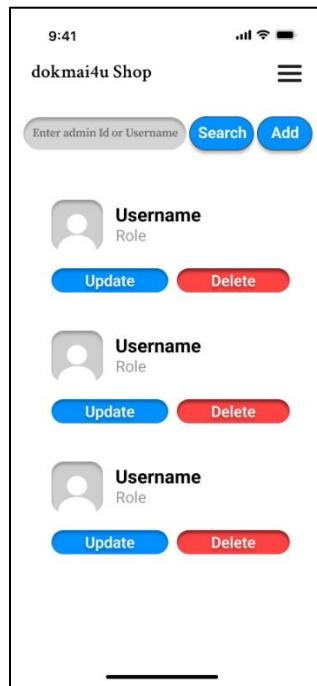
รูปที่ 2.3 Wireframe หน้า Detail Page



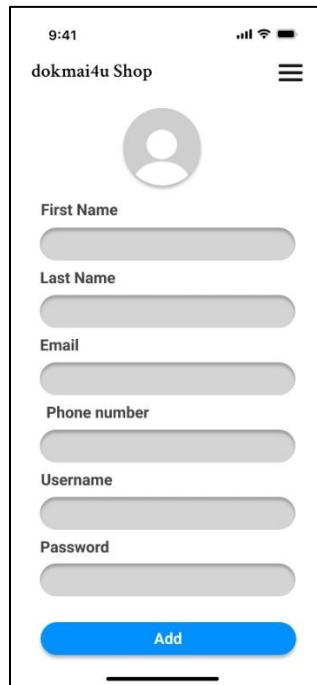
รูปที่ 2.4 Wireframe หน้า Login Page



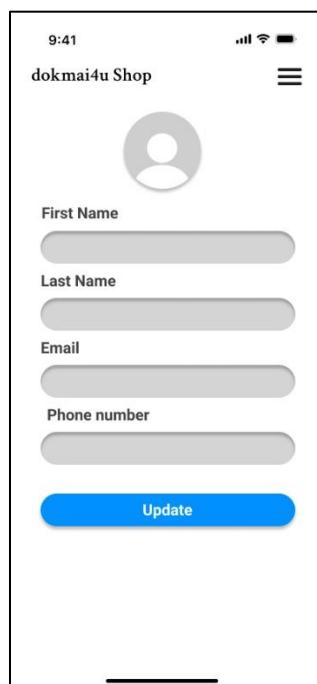
รูปที่ 2.5 Wireframe หน้า Team Page



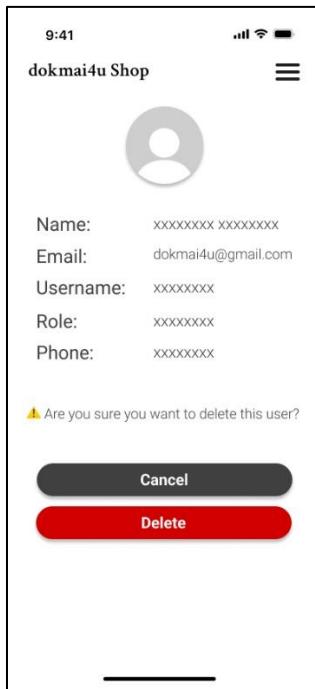
รูปที่ 2.6 Wireframe หน้า User Account Management Page



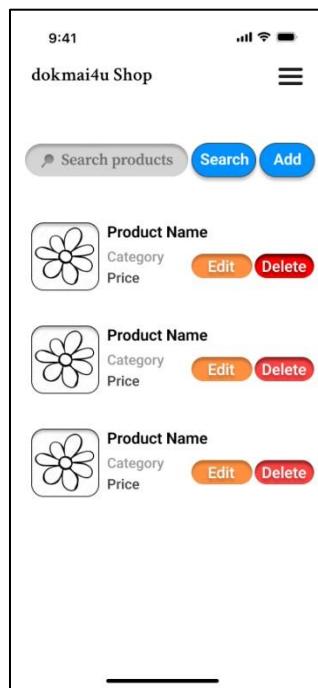
รูปที่ 2.7 Wireframe หน้า Add User Account Page



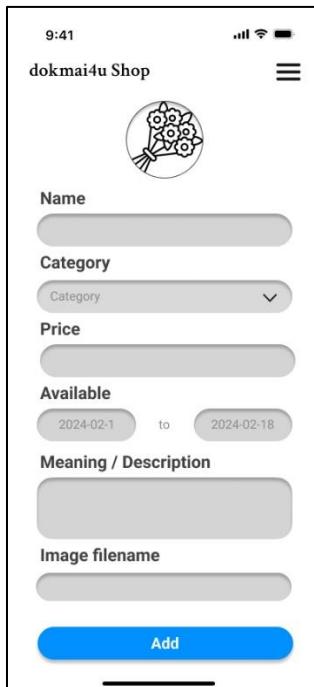
รูปที่ 2.8 Wireframe หน้า Update Information Account Page



រូបទី 2.9 Wireframe หน้า Delete Account Page



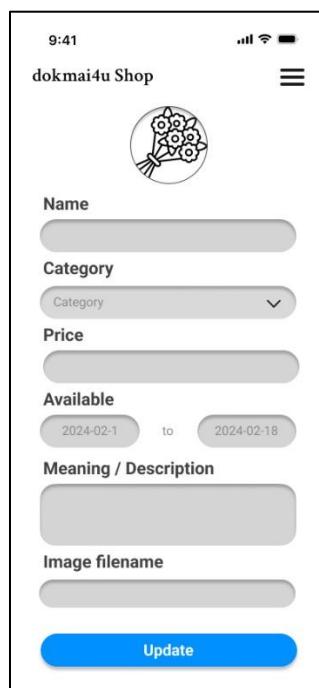
រូបទី 2.10 Wireframe หน้า Product Management Page



Wireframe for the Add Product Page. The interface includes:

- Header: dokmai4u Shop
- Image placeholder: A circular icon containing a small bouquet of flowers.
- Form fields:
  - Name: Text input field
  - Category: A dropdown menu labeled "Category".
  - Price: Text input field
  - Available: Date range selector showing "2024-02-1" to "2024-02-18".
  - Meaning / Description: Text input field
  - Image filename: Text input field
- Action button: A blue "Add" button at the bottom.

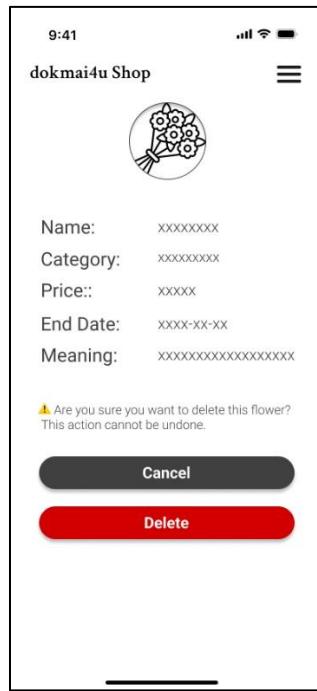
รูปที่ 2.11 Wireframe หน้า Add Product Page



Wireframe for the Update Product Page, similar to the Add Product Page but with an "Update" button instead of "Add". The interface includes:

- Header: dokmai4u Shop
- Image placeholder: A circular icon containing a small bouquet of flowers.
- Form fields:
  - Name: Text input field
  - Category: A dropdown menu labeled "Category".
  - Price: Text input field
  - Available: Date range selector showing "2024-02-1" to "2024-02-18".
  - Meaning / Description: Text input field
  - Image filename: Text input field
- Action button: A blue "Update" button at the bottom.

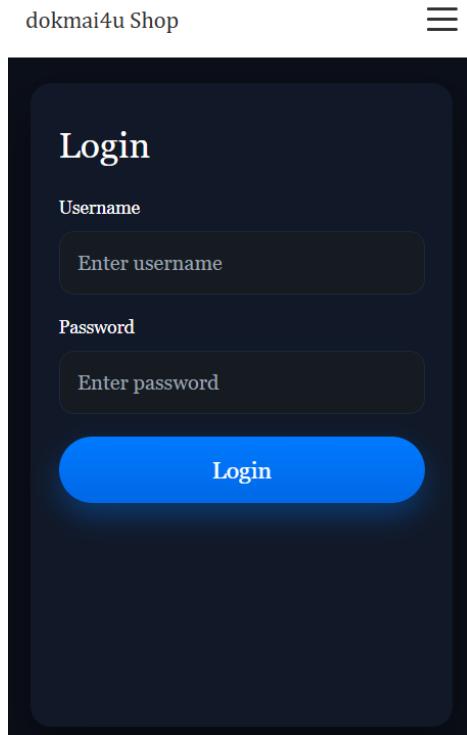
รูปที่ 2.12 Wireframe หน้า Update Product Page



รูปที่ 2.13 Wireframe หน้า Delete Product Page

## บทที่ 3

### รายละเอียดหน้าเว็บต่าง ๆ ของผู้ดูแล



รูปที่ 3.1 Login Admin Page

หน้า login admin page สามารถ login ได้เฉพาะ admin เท่านั้น เมื่อ login แล้ว admin สามารถจัดการข้อมูล product และ user

The screenshot shows a product management interface for 'dokmai4u Shop'. At the top, there is a search bar with the placeholder 'Enter product name' and a blue 'Search' button. To the right of the search bar are three buttons: 'Add', 'Edit', and 'Delete'. Below the search bar, there is a list of four products, each with a small thumbnail image, the product name, its category, and price.

Product Name	Category	Price
Rose	Bouquet	199
Tulip	Bouquet	249
Sunflower	Gift Set	299
Mali	Bouquet	200

Each product row contains two buttons: 'Edit' (orange) and 'Delete' (red).

รูปที่ 3.2 Product Management Page

หน้า Product Management Admin Page ภายใต้หน้าผู้ดูแล admin สามารถเพิ่ม แก้ไข ลบ และค้นหา product ที่ต้องการ

dokmai4u Shop 



**Name**  
Enter flower name

**Category**  
Category ▾

**Price**  
Enter flower price

**Available**  
22/00/2023  to 22/00/2023 

**Meaning**  
Enter flower meaning

**Image filename**  
e.g., rose.jpg

**Add**

รูปที่ 3.3 Product Add Page

หน้า Product Add Page ภายในหน้านี้ admin สามารถเพิ่มข้อมูล product ที่ต้องการ

dokmai4u Shop ≡



**Name**  
Rose

**Category**  
Bouquet

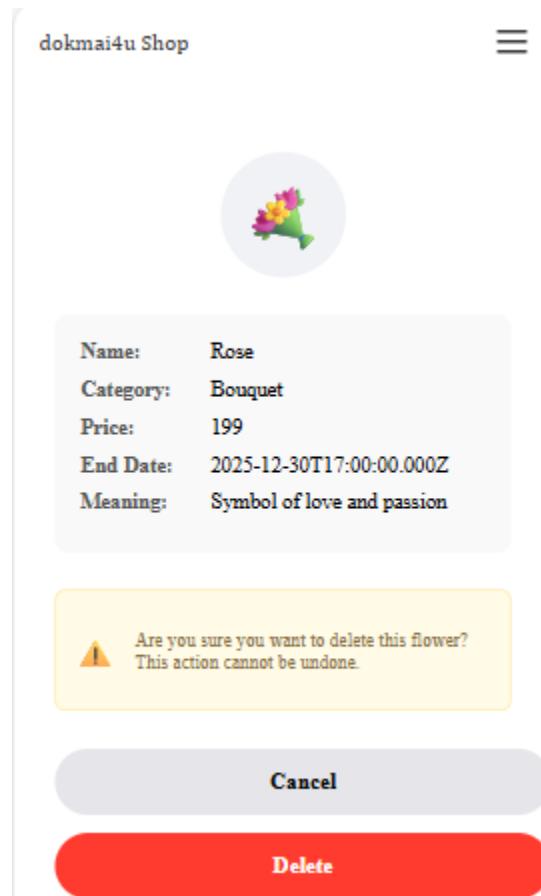
**Price**  
199

**Available**  
31/12/2024 to 30/12/2025

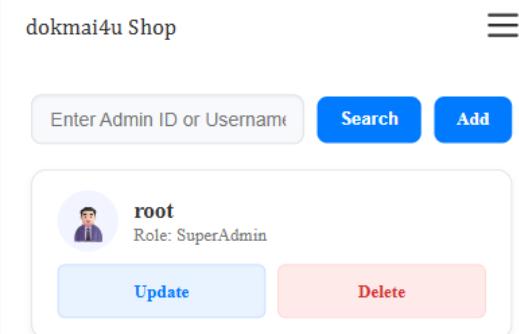
**Meaning**  
Symbol of love and passion

**Update**

รูปที่ 3.4 Product Update Page  
หน้า Product Update Page ภายในหน้านี้ admin สามารถแก้ไข product ตามข้อมูลที่ต้องการ update



รูปที่ 3.5 Product Delete Page  
หน้า Product Delete Page ภายใต้หน้าผู้ดูแล admin สามารถลบ product ที่ต้องการ



รูปที่ 3.6 Account Management Page

หน้า Account Management Page ภายใต้หน้านี้ admin สามารถเพิ่ม แก้ไข ลบ และค้นหา account ที่ต้องการ

dokmai4u Shop

First Name

Enter your first name

Last Name

Enter your last name

Email

Enter your email

Phone number

Enter your phone number

Username

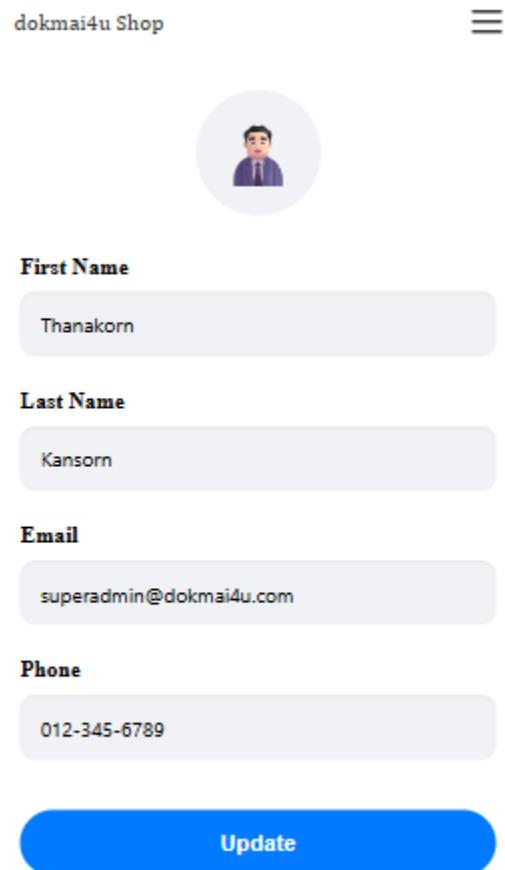
Enter your username

Password

Enter your password

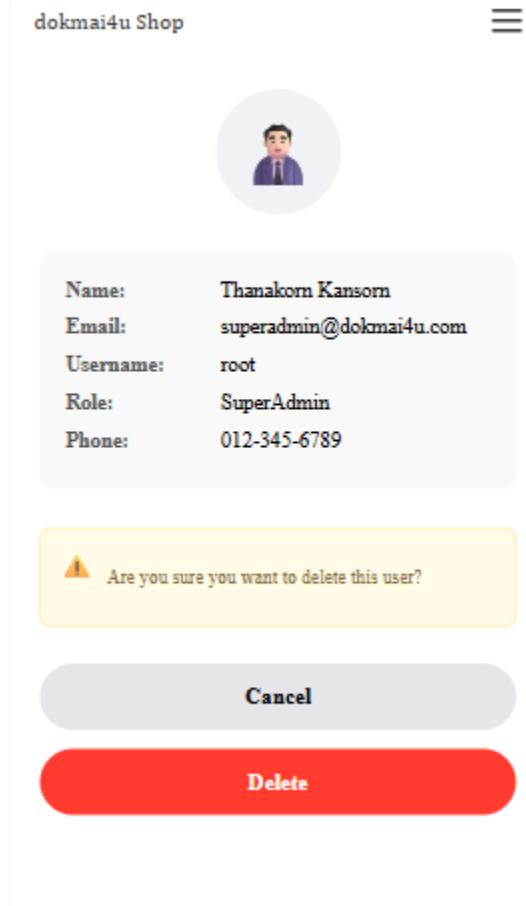
Add

รูปที่ 3.7 Account Add Page  
หน้า Account Add Page ภายใต้หน้าที่ admin สามารถเพิ่มข้อมูล account ที่ต้องการ



รูปที่ 3.8 Account Update Page

หน้า Account Update Page ภายใต้หน้าผู้ดูแล admin สามารถแก้ไข account ตามข้อมูลที่ต้องการ update



รูปที่ 3.9 Account Delete Page  
หน้า Account Delete Page ภายใต้หน้าผู้ดูแล admin สามารถลบ account ที่ต้องการ

## บทที่ 4

### รายละเอียดของเว็บเซอร์วิส และโค้ด

(controller)

4.1 account.controller.js :

Add admin

The screenshot shows a web application interface for adding a new admin user. The page title is 'dokmai4u Shop'. At the top right, there are navigation links: Home, Snack, Account, and Product. Below the title, there is a placeholder image of a person. The main form consists of five input fields: 'First Name' (containing 'admin'), 'Last Name' (placeholder 'Enter your last name'), 'Email' (placeholder 'Enter your email'), 'Phone number' (placeholder 'Enter your phone number'), and 'Username' (placeholder 'Enter your username').

(รูปที่ 4.1.1 add admin)

หน้า Account add admin จะทำการเพิ่ม admin

```
exports.add = async (req, res) => {
  try {
    const fName = (String(req.body.FName) || '').trim();
    const lName = (String(req.body.LName) || '').trim();
    const Email = (String(req.body.Email) || '').trim();
    const PhoneNumber = (String(req.body.PhoneNumber) || '').trim();
    const Username = (String(req.body.Username) || '').trim();
    const Password = (String(req.body.Password) || '').trim();
    if (!fName, lName, Email, PhoneNumber, Username, Password).every(Boolean)) {
      return res.status(400).json({ message: 'Missing required fields' });
    }
    const [duplicate] = await db.query(
      `SELECT 1 FROM Admin WHERE Username = ? OR Email = ? LIMIT 1`,
      [Username, Email]
    );
    if (duplicate.length > 0) {
      return res.status(409).json({ message: 'Username or Email already exists' });
    }
    const hashedPassword = await bcrypt.hash>Password, 10;
    const sql = `
      INSERT INTO Admin(FName, LName, Email, PhoneNumber, Username, Password)
      VALUES(?, ?, ?, ?, ?, ?)
    `;
    const params = [fName, lName, Email, PhoneNumber, Username, hashedPassword];
    const result = await db.query(sql, params);
    res.status(201).json({
      message: 'Inserted successfully',
      AdminID: result.insertId,
      account: { AdminID: result.insertId, FName: fName, LName: lName, Email: Email, PhoneNumber: PhoneNumber, Username: Username }
    });
  } catch (error) {
    if (error && error.error_code === 'ER_DUP_ENTRY') {
      return res.status(409).json({ message: 'Duplicate entry', error: error.sqlMessage });
    }
    console.error('Insert error:', error);
    res.status(500).json({ message: 'Insert error', error: error.message });
  }
};
```

(รูปที่ 4.1.2 Code add admin)

หน้า Account add admin จะทำการเพิ่ม admin

หน้า Account add admin จะทำการเพิ่ม admin พังก์ชัน add ทำหน้าที่เพิ่มบัญชีผู้ดูแลระบบใหม่เข้าฐานข้อมูล โดยรับข้อมูลจากผู้ใช้งาน req.body และตรวจสอบความครบถ้วนของข้อมูล จากนั้นเช็คกว่า Username หรือ Email ซ้ำหรือไม่ก่อนดำเนินการต่อ หากข้อมูลถูกต้องจะเข้ารหัสรหัสผ่านด้วย bcrypt เพื่อความปลอดภัย และสร้าง SQL แบบ INSERT เพื่อบันทึกลงฐานข้อมูล และส่งผลลัพธ์พร้อม AdminID กลับไปยังผู้ใช้ หากเกิดข้อผิดพลาดจะส่งข้อความแจ้งเตือนสถานะที่เหมาะสมกลับไป.

## Update admin

(รูปที่ 4.1.3 Update admin)

หน้า Account update admin จะทำการupdate admin

```
exports.update = async (req, res) => {
  try {
    const { id } = req.params;
    const { FName, LName, Email, Phone } = req.body;

    if (!FName || !LName || !Email) {
      return res.status(400).json({ message: 'First Name, Last Name, Email are required' });
    }

    const [result] = await db.query(
      `UPDATE Admin SET FName = ?, LName = ?, Email = ?, PhoneNumber = ? WHERE AdminID = ?`,
      [FName, LName, Email, Phone || "-", id]
    );

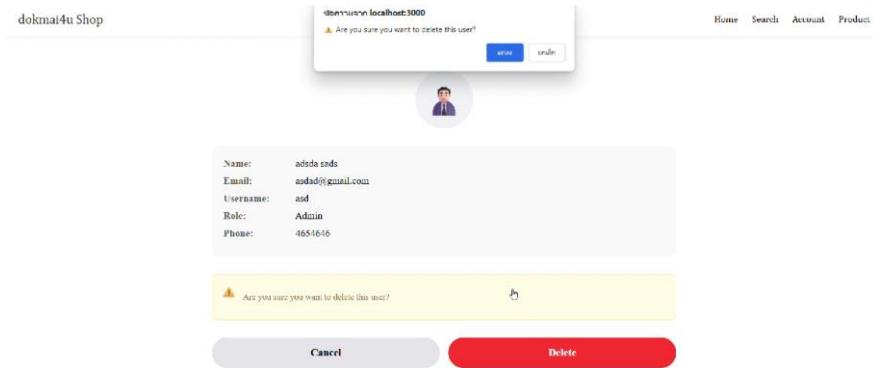
    if (result.affectedRows === 0) {
      return res.status(404).json({ message: 'Account not found' });
    }
    return res.status(200).json({ ok: true, updatedId: id });
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: 'Database error' });
  }
};
```

(รูปที่ 4.1.4 Code Update admin)

หน้า Account update admin จะทำการupdate admin

พิมพ์คำสั่ง update มีหน้าที่อัปเดตข้อมูลบัญชีผู้ดูแลระบบตาม AdminID ที่ส่งผ่าน URL โดยรับข้อมูลใหม่จาก req.body เช่น FName, LName, Email และ Phone จากนั้นตรวจสอบว่าข้อมูลถูกคุณครับหรือไม่ ก่อนส่งคำสั่ง SQL แบบ UPDATE ไปยังฐานข้อมูล หากพบว่ามีการแก้ไขจริงจะส่งผลสำเร็จลับไป แต่ถ้าไม่พบบัญชีจะส่งสถานะ 404 และหากเกิดข้อผิดพลาดจากระบบฐานข้อมูลจะส่งสถานะ 500 แจ้งข้อผิดพลาดกลับไป.

## Delete admin



(รูปที่ 4.1.5 delete admin)

หน้า Account delete admin จะทำการ delete admin

```
exports.delete = async (req, res) => {
  try {
    const adminId = req.params.id;
    if (!adminId) return res.status(400).json({ message: 'Missing Admin ID' });

    const [result] = await db.query('DELETE FROM Admin WHERE AdminID = ?', [adminId]);
    if (result.affectedRows === 0) return res.status(404).json({ message: 'Account not found' });

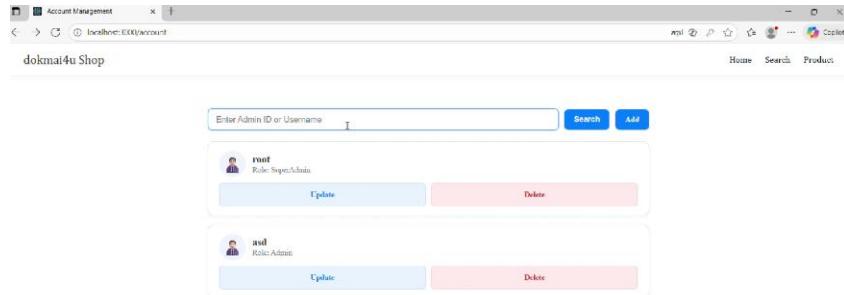
    res.status(200).json({ message: 'Account deleted successfully', adminId });
  } catch (error) {
    console.error('Delete error:', error);
    res.status(500).json({ message: 'Database error during deletion' });
  }
};
```

(รูปที่ 4.1.6 Code delete admin)

หน้า Account delete admin จะทำการ delete admin

ฟังก์ชัน delete ทำหน้าที่ลบบัญชีผู้ดูแลระบบตาม AdminID ที่รับมาจาก URL โดยตรวจสอบก่อนว่ามีการส่งรหัสเข้ามาหรือไม่ จากนั้นรันคำสั่ง SQL แบบ DELETE เพื่อลบข้อมูลออกจากฐานข้อมูล หากไม่พบบัญชีจะส่งสถานะ 404 แจ้งว่าไม่พบข้อมูล แต่ถ้าลบสำเร็จจะส่งข้อความยืนยันกลับไป พร้อม adminId และหากเกิดข้อผิดพลาดจากฐานข้อมูลจะตอบกลับด้วยสถานะ 500.

## Get all admin



(รูปที่ 4.1.7 get all admin)

หน้า Account get all admin จะทำการ get all admin

```
exports.getall = async (req, res) => {
  try {
    const [rows] = await db.query(
      'SELECT AdminID, Role, Username FROM Admin'
    );
    res.json(rows);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Database error' });
  }
};
```

(รูปที่ 4.1.8 Code get all admin)

หน้า Account get all admin จะทำการ get all admin

ฟังก์ชัน getall ทำหน้าที่ดึงข้อมูลบัญชีผู้ดูแลระบบทั้งหมดจากฐานข้อมูล โดยเลือกเฉพาะข้อมูลสำคัญคือ AdminID, Role และ Username หากดึงข้อมูลสำเร็จจะส่งกลับในรูปแบบ JSON แต่หากเกิดข้อผิดพลาดระหว่างการเข้ามายังฐานข้อมูลจะส่งสถานะ 500 แจ้งข้อผิดพลาดกลับไป.

## Search admin



(รูปที่ 4.1.9 search admin)

หน้า Account search admin จะทำการ search admin

```
exports.search = async (req, res) => {
  try {
    const searchKey = (req.body.searchKey || '').trim();
    const searchID = (req.body.searchID || '').trim();
    const searchRole = (req.body.searchRole || '').trim();

    const where = [];
    const params = [];

    if (searchID) {
      where.push('AdminID = ?');
      params.push(searchID);
    }

    if (searchKey) {
      where.push('Username LIKE ?');
      params.push(`%${searchKey}%`);
    }

    if (searchRole) {
      where.push('Role = ?');
      params.push(searchRole);
    }

    let sql = 'SELECT AdminID, Role, Username FROM Admin';

    if (where.length) {
      sql += ` WHERE ${where.join(' OR ')}`;
    }

    const [rows] = await db.query(sql, params);
    res.json(rows);
  } catch (error) {
    console.error('Search error:', error);
    res.status(500).json({ message: 'Database error' });
  }
};
```

(รูปที่ 4.1.10 Code search admin)

หน้า Account search admin จะทำการ search admin

ฟังก์ชัน search ทำหน้าที่ค้นหาข้อมูลผู้ดูแลระบบจากฐานข้อมูล โดยรับเงื่อนไขการค้นหาจากผู้ใช้ เช่น AdminID, Username หรือ Role และสร้างคำสั่ง SQL ตามเงื่อนไขที่ให้มา จากนั้นนำไปค้นหาและส่งผลลัพธ์กลับในรูปแบบ JSON หากไม่พบข้อมูลจะส่งผลลัพธ์ว่าง และหากเกิดข้อผิดพลาดจากฐานข้อมูลจะส่งสถานะ 500 และข้อผิดพลาดกลับไป.

## Search by ID



(รูปที่ 4.1.11 search by ID admin)

หน้า Account search by ID admin จะทำการ search by ID admin

โดยจะ search ตามเลข ID ของadmin

```
exports.get = async (req, res) => {
  try {
    const { id } = req.params;
    const [rows] = await db.query(
      'SELECT * FROM Admin WHERE AdminID = ?',
      [id]
    );
    if (!rows.length) return res.status(404).json({ message: 'Not found' });
    return res.json(rows[0]);
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: 'Database error' });
  }
};
```

(รูปที่ 4.1.12 Code search by ID admin)

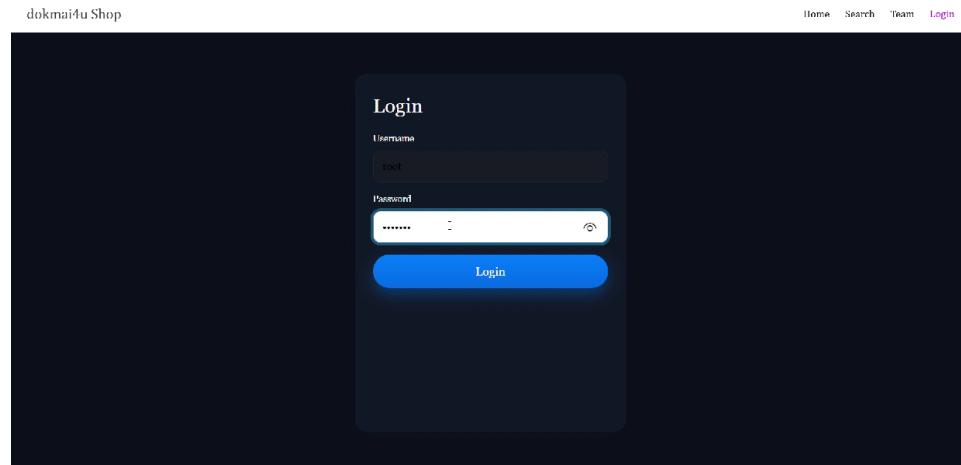
หน้า Account search by ID admin จะทำการ search by ID admin

โดยจะ search ตามเลข ID ของadmin

ฟังก์ชัน get ทำหน้าที่ดึงข้อมูลบัญชีผู้ดูแลระบบจากฐานข้อมูลตาม AdminID ที่รับมาทาง URL โดยใช้คำสั่ง SQL เพื่อค้นหา หากพบข้อมูลจะส่งรายละเอียดของบัญชีนั้นกลับไปในรูปแบบ JSON และถ้าไม่พบจะส่งสถานะ 404 และถ้ามีเพลช้อมูล และหากเกิดข้อผิดพลาดระหว่างการ query จะส่งสถานะ 500 พร้อมข้อความแจ้งข้อผิดพลาด.

## 4.2 auth.controller.js :

Login admin



(รูปที่ 4.2.1 Login admin)

หน้า Account Login admin จะทำการ Login admin

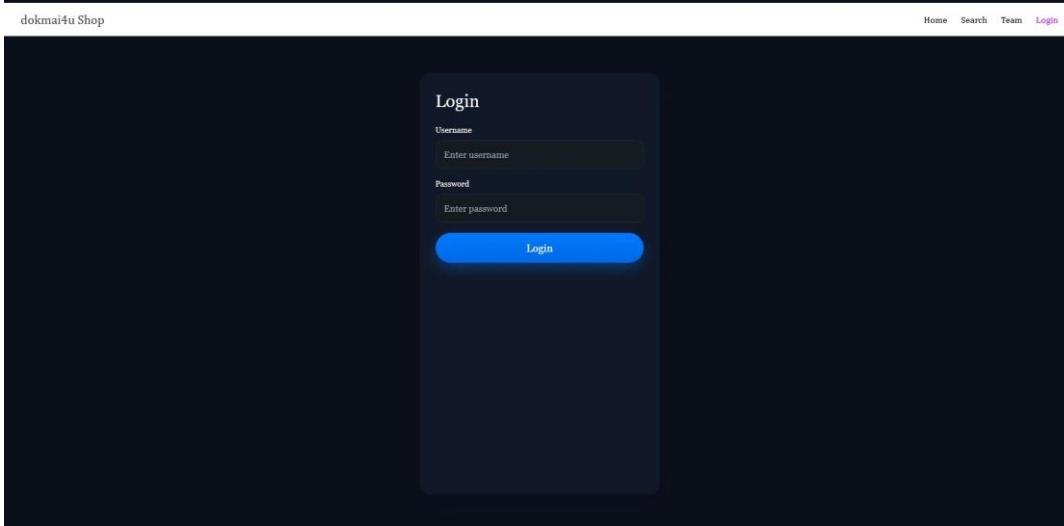
```
exports.login = async (req, res) => {
  try {
    const { username, password } = req.body;
    if (!username || !password) {
      return res.status(400).json({ message: 'Username and password are required' });
    }
    const [rows] = await db.query(
      'SELECT * FROM Admin WHERE Username = ?',
      [username]
    );
    if (!rows.length) {
      return res.status(401).json({ message: 'Invalid username or password' });
    }
    const user = rows[0];
    const ok = await bcrypt.compare(password, user.Password);
    if (!ok) {
      return res.status(401).json({ message: 'Invalid username or password' });
    }
    req.session.user = {
      id: user.AdminID,
      username: user.Username,
      role: user.Role
    };
    await db.query(
      'INSERT INTO Login (AdminID) VALUES (?)',
      [user.AdminID]
    );
    console.log('Login session:', req.session);
    return res.json({
      message: 'Login successful',
      user: {
        id: user.AdminID,
        username: user.Username,
        role: user.Role
      }
    });
  } catch (e) {
    console.error(e);
    return res.status(500).json({ message: 'Login error' });
  }
};
```

(รูปที่ 4.2.2 Code Login admin)

หน้า Account Login admin จะทำการ Login admin

ฟังก์ชัน login ทำหน้าที่ตรวจสอบการเข้าสู่ระบบของผู้ใช้ โดยรับ username และ password จาก req.body และนำไปตรวจสอบในฐานข้อมูล หากพบบัญชีจะทำการเปรียบเทียบรหัสผ่านกับข้อมูลที่เข้ารหัสไว้ด้วย bcrypt หากถูกต้องจะสร้าง session เก็บข้อมูลผู้ใช้ เช่น ID, username และ role พร้อมบันทึกเวลาการเข้าสู่ระบบลงในตาราง Login และส่งข้อความแจ้งว่าเข้าสู่ระบบสำเร็จ แต่หากข้อมูลไม่ถูกต้องหรือไม่พบผู้ใช้ จะส่งสถานะ 401 แจ้งว่าไม่สามารถเข้าสู่ระบบได้.

Logout admin



(รูปที่ 4.2.3 Login admin)

หน้า Account Login admin จะทำการ แสดงหน้า Login admin เมื่อทำการ Logout

```
exports.logout = (req, res) => {
  req.session.destroy(() => {
    res.clearCookie('connect.sid');
    res.redirect('http://localhost:3000/login');
  });
}
```

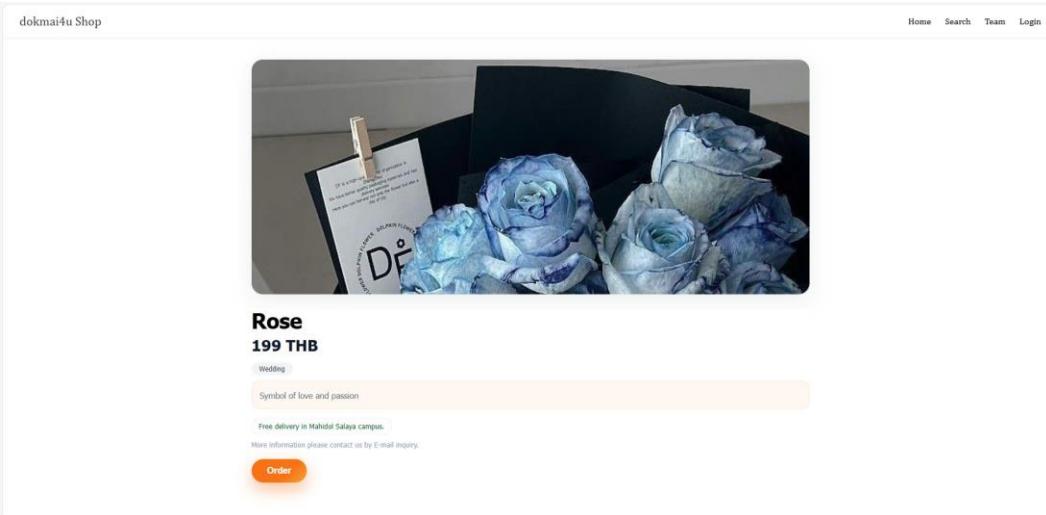
(รูปที่ 4.2.4 Code Logout admin)

หน้า Account Login admin จะทำการ แสดงหน้า Login admin เมื่อทำการ Logout

ฟังก์ชัน logout มีหน้าที่จัดการกระบวนการออกจากระบบของผู้ใช้ โดยเริ่มจากการทำลายข้อมูล Session ที่ผูกกับผู้ใช้งานคำสั่ง req.session.destroy() เพื่อยกเลิก session ออกจากระบบ จากนั้นระบบจะลบคุกกี้ที่ใช้เก็บ session id ด้วยคำสั่ง res.clearCookie('connect.sid') เพื่อให้เบราว์เซอร์ไม่เก็บ session เดิมไว้อีกต่อไป เมื่อเสร็จสิ้นกระบวนการออกจากระบบแล้ว ระบบจะทำการเปลี่ยนเส้นทาง (Redirect) ผู้ใช้กลับไปยังหน้าเข้าสู่ระบบ (Login Page) ด้วย res.redirect('http://localhost:3000/login') เพื่อให้ผู้ใช้สามารถเข้าสู่ระบบใหม่ได้หากต้องการ

### 4.3 index.controller.js :

Datail



(รูปที่ 4.3.1 Detail)

หน้า Account Login admin จะทำการ Login admin

```
const db = require('../db');

exports.get = async (req, res) => {
    try {
        const id = req.params.id;
        const sql = `SELECT * FROM Flower JOIN Category ON Flower.CID = Category.CID WHERE FlowerID = ?`;
        const [rows] = await db.query(sql, id);

        if (!rows.length) {
            return res.status(404).json({ message: 'No data found' });
        }

        return res.json(rows[0]);
    } catch (error) {
        console.error(error);
        return res.status(500).json({ message: 'Database error' });
    }
};
```

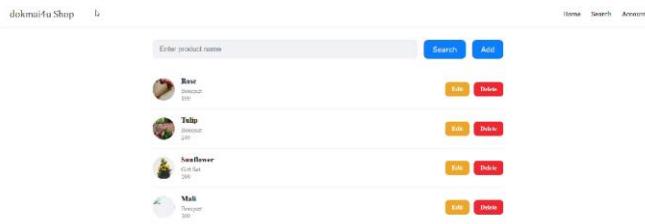
(รูปที่ 4.3.2 Code Detail)

หน้า Account Login admin จะทำการ Login admin

ฟังก์ชัน `get` ทำหน้าที่ดึงข้อมูลรายละเอียดของดอกไม้จากฐานข้อมูลโดยใช้ `FlowerID` ที่รับผ่าน URL จากนั้นใช้คำสั่ง SQL แบบ `JOIN` เพื่อรวมข้อมูลของตาราง `Flower` และ `Category` เข้าด้วยกัน หากพบข้อมูลจะส่งกลับในรูปแบบ JSON แต่ถ้าไม่พบจะส่งสถานะ 404 แจ้งว่าไม่พบข้อมูล และหากเกิดข้อผิดพลาดระหว่างการทำงานจะส่งสถานะ 500 พร้อมข้อความแจ้งข้อผิดพลาด.

#### 4.4 product.controller.js :

##### Search Product for User



(รูปที่ 4.4.1 Search all Product) for User

หน้า Search product จะทำการ Search all product

```
exports.search = async (req, res) => {
  try {
    const FlowerName = (req.query.FlowerName ?? '').toString().trim();
    const CID = (req.query.Category ?? '').toString().trim();
    const MinPrice = (req.query.MinPrice ?? '').toString().trim();
    const MaxPrice = (req.query.MaxPrice ?? '').toString().trim();

    const where = [];
    const params = [];

    if (FlowerName) {
      where.push(`f.FlowerName LIKE ?`);
      params.push(`%${FlowerName}%`);
    }

    if (CID !== '') {
      const cid = Number(CID);
      if (!Number.isNaN(cid)) {
        where.push(`f.CID = ?`);
        params.push(cid);
      }
    }

    if (MinPrice !== '') {
      const min = Number(MinPrice);
      if (!Number.isNaN(min)) {
        where.push(`f.Price >= ?`);
        params.push(min);
      }
    }

    if (MaxPrice !== '') {
      const max = Number(MaxPrice);
      if (!Number.isNaN(max)) {
        where.push(`f.Price <= ?`);
        params.push(max);
      }
    }

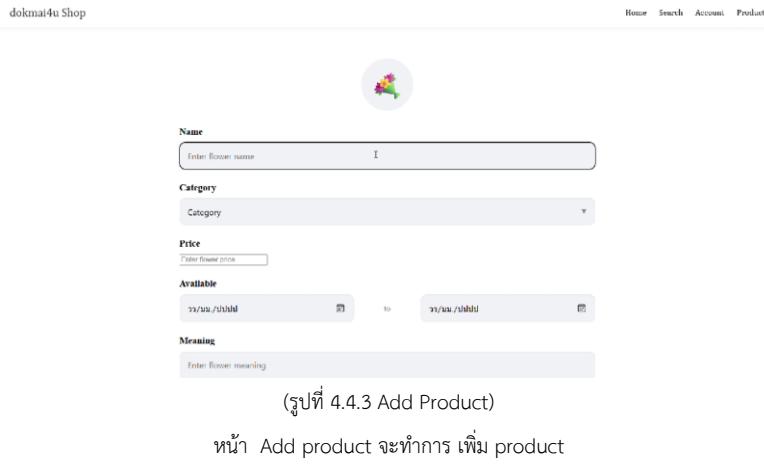
    const whereSql = where.length ? `WHERE ${where.join(' AND ')}` : '';
    const sql = `SELECT * FROM Flower AS f JOIN Category AS c ON f.CID = c.CID ${whereSql}`;
    const [rows] = await db.query(sql, params);
    res.type('application/json').status(200).send(JSON.stringify(rows));
  } catch (err) {
    console.error(`Search error: ${err}`);
    res.status(500).json({ message: 'Database error' });
  }
}
```

(รูปที่ 4.4.2 Code Search all Product for User)

หน้า Search product จะทำการ Search all product for User

พิมพ์ชั้น search ทำหน้าที่ค้นหาข้อมูลดอกไม้จากฐานข้อมูล โดยใช้เงื่อนไขที่ผู้ใช้ส่งมาทาง query เช่นชื่อดอกไม้ หมวดหมู่ หรือช่วงราคา หากค้นหาได้จะส่งข้อมูลกลับในรูปแบบ JSON แต่หากเกิดข้อผิดพลาดจะส่งสถานะ 500 แจ้งข้อผิดพลาดจากฐานข้อมูล.

## Add Product



```
exports.add = async (req, res) => {
  try {
    const FlowerName = (req.body.FlowerName || '').trim();
    const CID = Number(req.body.CID || '');
    const Price = Number(req.body.Price || 0);
    const StartDate = req.body.StartDate ? String(req.body.StartDate).slice(0, 10) : null;
    const EndDate = req.body.EndDate ? String(req.body.EndDate).slice(0, 10) : null;
    const Meaning = (req.body.Meaning || '').trim();
    const srcImage = String(req.body.srcImage).trim() : null;

    if (!FlowerName || isNaN(Price) || !Meaning) {
      return res.status(400).json({ message: 'Flower Name, Price, Meaning are required' });
    }

    const sql = `
      INSERT INTO Flower(FlowerName, CID, Price, StartDate, EndDate, Meaning, srcImage)
      VALUES(?, ?, ?, ?, ?, ?, ?)`;

    const params = [FlowerName, CID, Price, StartDate, EndDate, Meaning, srcImage];

    const [result] = await db.query(sql, params);

    return res.status(201).json({
      message: 'Inserted successfully',
      insertId: result.insertId
    });
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: 'Database error' });
  }
};
```

(รูปที่ 4.4.4 Code Add Product)

หน้า Add product จะทำการ เพิ่ม product

ฟังก์ชัน add ทำหน้าที่เพิ่มข้อมูลดอกไม้ใหม่เข้าสู่ฐานข้อมูล โดยรับข้อมูลจาก req.body เช่น FlowerName, CID, Price, Meaning, วันที่เริ่ม-สิ้นสุดจำหน่าย และรูปภาพ จากนั้นตรวจสอบว่าข้อมูลไม่มี ราคา และความหมายถูกกรอกครบหรือไม่ หากข้อมูลถูกต้องจะใช้คำสั่ง SQL แบบ INSERT เพื่อบันทึกข้อมูลลงในตาราง Flower และส่งผลลัพธ์กลับด้วยสถานะ 201 พร้อม insertId แต่หากเกิดข้อผิดพลาดจะส่งสถานะ 500 และข้อผิดพลาดจากฐานข้อมูล.

## Update Product

The screenshot shows a web-based application interface for updating a product. At the top, there's a navigation bar with links for Home, Search, Account, and Product. Below the navigation, there's a search bar with placeholder text 'enreq'. The main area contains several input fields: 'Name' (set to 'enreq'), 'Category' (set to 'Bouquet'), 'Price' (set to '200'), 'Available' (date range from '29/10/2025' to '04/11/2025'), and 'Meaning' (set to 'dead'). At the bottom of the form, a green success message 'Update Success!' is visible above a blue 'Update' button.

(รูปที่ 4.4.5 Update Product)

หน้า Edit product จะทำการ Update product

```
exports.update = async (req, res) => {
  try {
    const { id } = req.params;
    let { FlowerName, CID, Price, StartDate, EndDate, Meaning, srcImage } = req.body;

    CID = CID ? Number(CID) : null;
    Price = Number(Price);

    if (!FlowerName || Number.isNaN(Price) || !Meaning) {
      return res.status(400).json({ message: 'Flower Name, Price, and Meaning are required' });
    }

    const sql = `
      UPDATE Flower SET FlowerName = ?, CID = ?, Price = ?, StartDate = ?, EndDate = ?, Meaning = ?, srcImage = ?
      WHERE FlowerID = ?`;
    const params = [FlowerName, CID, Price, StartDate || null, EndDate || null, Meaning, srcImage || null, id];

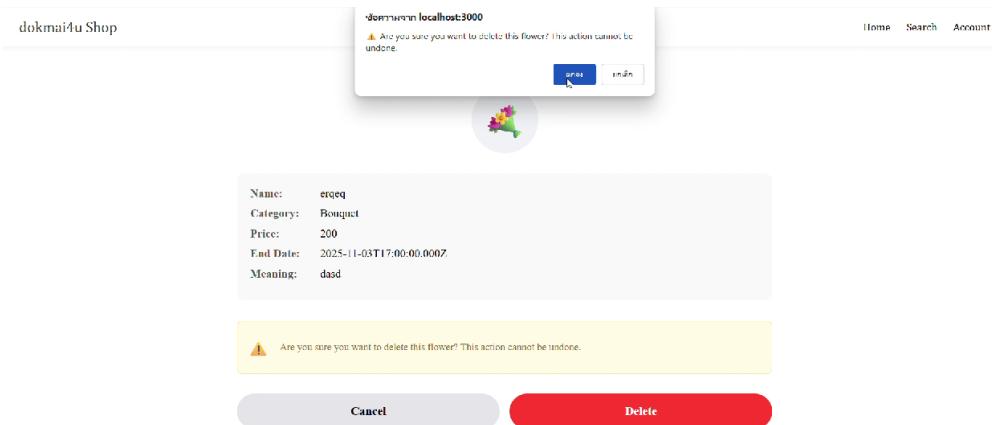
    const [result] = await db.query(sql, params);
    if (result.affectedRows === 0) {
      return res.status(404).json({ message: 'Flower not found' });
    }
    return res.status(200).json({ ok: true, updatedId: id });
  } catch (error) {
    console.error(error);
    return res.status(500).json({ message: 'Database error' });
  }
};
```

(รูปที่ 4.4.6 Code Update Product)

หน้า Edit product จะทำการ Update product

ฟังก์ชัน update ทำหน้าที่แก้ไขข้อมูลเดิมตาม FlowerID ที่รับจาก URL โดยอ่านข้อมูลใหม่จาก req.body เช่น FlowerName, CID, Price, Meaning, วันที่ และรูปภาพ จากนั้นตรวจสอบความถูกต้องของข้อมูลที่จำเป็น ก่อนใช้ SQL แบบ UPDATE เพื่อแก้ไขข้อมูลในฐานข้อมูล ถ้าอัปเดตสำเร็จจะส่งสถานะ 200 และ updatedId กลับไป แต่หากไม่พบข้อมูลจะส่งสถานะ 404 และกรณีเกิดข้อผิดพลาดจากฐานข้อมูลจะส่งสถานะ 500.

## Delete Product



(รูปที่ 4.4.7 Delete Product)

หน้า delete product จะทำการ delete product

```
exports.delete = async (req, res) => {
  try {
    const FlowerID = req.params.id;
    if (!FlowerID) return res.status(400).json({ message: 'Missing Flower ID' });

    const [result] = await db.query('DELETE FROM Flower WHERE FlowerID = ?', [FlowerID]);
    if (result.affectedRows === 0) return res.status(404).json({ message: 'Product not found' });

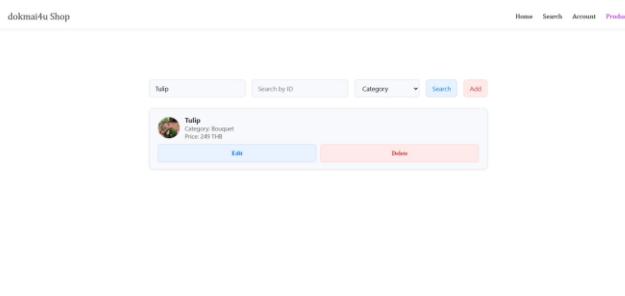
    res.status(200).json({ message: 'Product deleted successfully', FlowerID });
  } catch (error) {
    console.error('Delete error:', error);
    res.status(500).json({ message: 'Database error during deletion' });
  }
};
```

(รูปที่ 4.4.8 Code Delete Product)

หน้า delete product จะทำการ delete product

พังก์ชัน delete ทำหน้าที่ลบข้อมูลดอกไม้ออกจากฐานข้อมูลโดยใช้ FlowerID ที่ส่งมาทาง URL เริ่มจากตรวจสอบว่าได้รับ ID หรือไม่ หากนั้นใช้ SQL แบบ DELETE เพื่อลบข้อมูล หากไม่พบดอกไม้ตาม ID จะตอบกลับด้วยสถานะ 404 แต่ถ้าลับสำเร็จจะส่งข้อความยืนยันพร้อม ID คืนกลับ และหากเกิดปัญหาในฐานข้อมูลจะส่งสถานะ 500 แจ้งข้อผิดพลาด.

## Search Product by name for admin



(รูปที่ 4.4.9 Search Product by name for admin)

หน้า Search product จะทำการ Search product by name

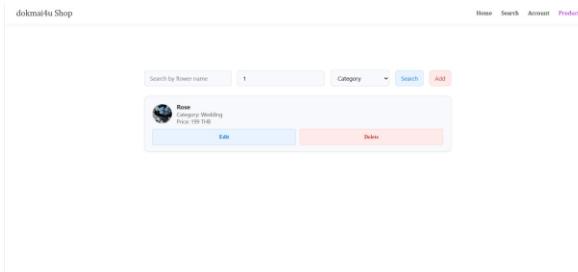
```
exports.searchs = async (req, res) => {
  try {
    const searchKey = (req.body.searchKey || '').toString().trim();
    const searchID = (req.body.searchID || '').toString().trim();
    const cid = [req.body.cid || ''].toString().trim();
    const where = [];
    const params = [];
    if (searchID) {
      where.push('f.FlowerID = ?');
      params.push(searchID);
    }
    if (searchKey) {
      where.push('f.FlowerName LIKE ?');
      params.push(`%${searchKey}%`);
    }
    if (cid) {
      where.push('f.CID = ?');
      params.push(cid);
    }
    const whereSql = where.length ? `WHERE ${where.join(' OR ')}` : '';
    const sql = `
      SELECT
        f.FlowerID,
        f.FlowerName,
        f.Price,
        f.srcImage,
        f.CID,
        c.CName
      FROM Flower AS f
      JOIN Category AS c ON f.CID = c.CID
      ${whereSql}
    `;
    const [rows] = await db.query(sql, params);
    res.json(rows);
  } catch (error) {
    console.error('Search error:', error);
    res.status(500).json({ message: 'Database error' });
  }
};
```

(รูปที่ 4.4.10 Code Search Product for admin)

หน้า Search product จะทำการ Search product

ฟังก์ชันค้นหาด้วยชื่อดอกไม้มีหน้าที่ค้นหารายการดอกไม้จากฐานข้อมูลโดยใช้ชื่อดอกไม้ (FlowerName) เป็นเงื่อนไขหลัก ฟังก์ชันจะรับข้อความค้นหาจากผู้ใช้ แล้วนำไปสร้างเงื่อนไขในคำสั่ง SQL ด้วยรูปแบบ FlowerName LIKE '%...%' เพื่อให้สามารถค้นหาชื่อดอกไม้ที่ตรงบางส่วนหรือคล้ายกับคำค้นหาได้ จากนั้นจะใช้คำสั่ง JOIN ระหว่างตาราง Flower และ Category เพื่อแสดงข้อมูลทั้งตัวดอกไม้และหมวดหมู่ร่วมกัน ผลลัพธ์ที่ได้อาจมีหลายรายการและจะถูกส่งกลับในรูปแบบ JSON หากไม่มีข้อมูลที่ตรงกับเงื่อนไข ระบบจะส่งผลลัพธ์เป็นรายการว่าง และหากเกิดข้อผิดพลาดจากการทำงานของฐานข้อมูลจะส่งข้อความแจ้งข้อผิดพลาดให้ผู้ใช้ทราบ เช่นเดียวกัน

## Search Product by ID for admin



(รูปที่ 4.4.11 Search Product by ID for admin)

หน้า Search product จะทำการ Search product by ID

```
exports.searchs = async (req, res) => {
  try {
    const searchKey = (req.body.searchKey || '').toString().trim();
    const searchID = (req.body.searchID || '').toString().trim();
    const cid = [req.body.cid || []].toString().trim();
    const where = [];
    const params = [];
    if (searchID) {
      where.push('f.FlowerID = ?');
      params.push(searchID);
    }
    if (searchKey) {
      where.push('f.FlowerName LIKE ?');
      params.push(`%${searchKey}%`);
    }
    if (cid) {
      where.push('f.CID = ?');
      params.push(cid);
    }
    const whereSql = where.length ? `WHERE ${where.join(' OR ')}` : '';
    const sql = `
      SELECT
        f.FlowerID,
        f.FlowerName,
        f.Price,
        f.srImage,
        f.CID,
        c.CName
      FROM Flower AS f
      JOIN Category AS c ON f.CID = c.CID
      ${whereSql}
    `;
    const [rows] = await db.query(sql, params);
    res.json(rows);
  } catch (error) {
    console.error('Search error:', error);
    res.status(500).json({ message: 'Database error' });
  }
};
```

(รูปที่ 4.4.12 Code Search by Product for admin)

หน้า Search product จะทำการ Search product by ID

ฟังก์ชันค้นหาด้วยรหัสดอกไม้มีหน้าที่ใช้รหัสประจำตัวดอกไม้ (FlowerID) เป็นเงื่อนไขหลักในการค้นหาข้อมูลจากฐานข้อมูล ตารางที่เกี่ยวข้องคือ Flower และ Category โดยฟังก์ชันจะรับค่า FlowerID จากคำขอของผู้ใช้ (เช่น จากพารามิเตอร์หรือข้อมูลใน body) จากนั้นนำค่าไปใช้ในคำสั่ง SQL แบบ WHERE FlowerID = ? เพื่อดึงข้อมูลดอกไม้เพียงรายการเดียวที่ตรงกับรหัสดังกล่าว พ้อ้มเขื่อมกับตารางหมวดหมู่ (Category) ผ่านคำสั่ง JOIN เมื่อค้นหาสำเร็จ ระบบจะส่งข้อมูลรายละเอียดของดอกไม้กลับไปในรูปแบบ JSON แต่หากไม่พบข้อมูลจะส่งข้อความแจ้งว่าไม่พบข้อมูล และหากเกิดข้อผิดพลาดจากฐานข้อมูลจะส่งสถานะข้อผิดพลาดให้ผู้ใช้ทราบ

## บทที่ 5

### ผลการทดสอบของเว็บเซอร์วิส

#### 5.1 Login

POST http://localhost:3001/api/auth/login

Params Authorization Headers (9) Body Scripts Settings

Body (raw JSON)

```

1 {
2   "username": "nitichot",
3   "password": "6787048"
4 }
```

Body Cookies (1) Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "Login successful",
3   "user": {
4     "id": 3,
5     "username": "nitichot",
6     "role": "Admin"
7   }
8 }
```

(รูปที่ 5.1.1 Test Admin Login)

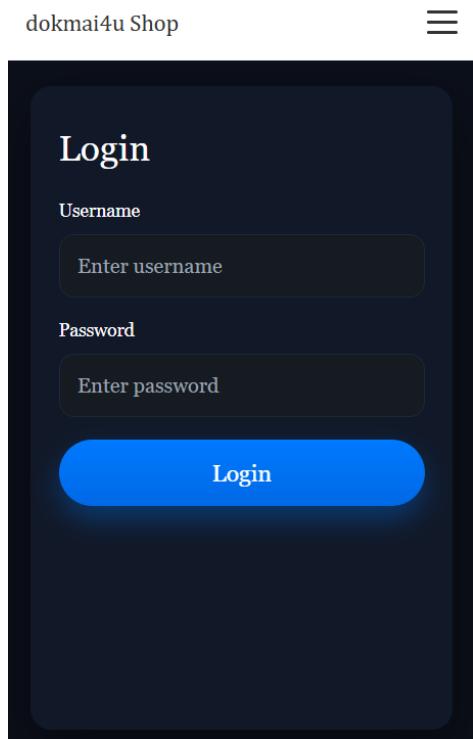
Test Admin Login

method: post

URL: http://localhost:3001/api/auth/login

body: raw JSON

```
{
  "username": "nitichot",
  "password": "6787048"
}
```



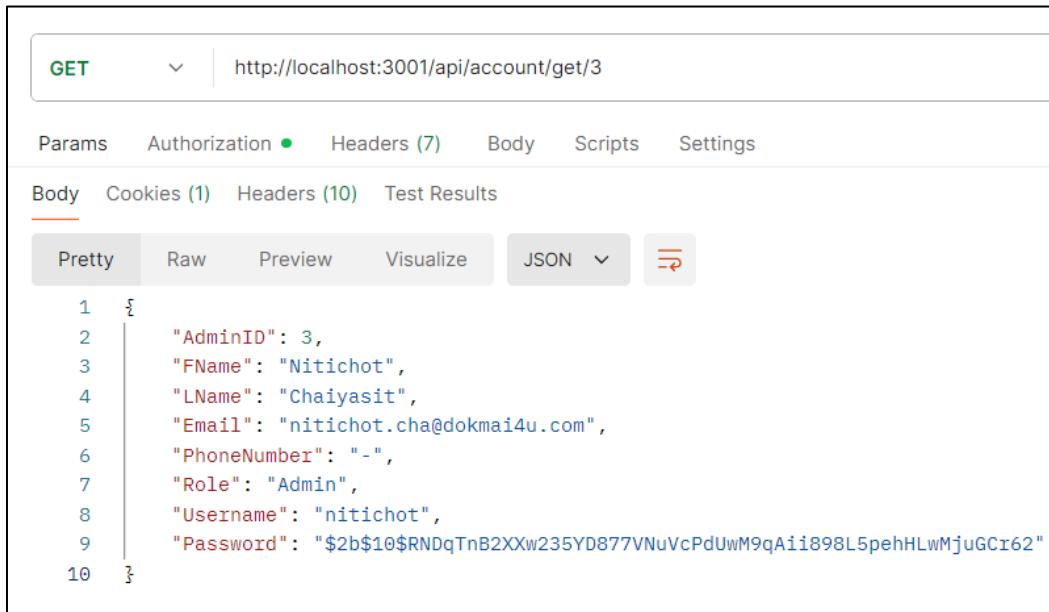
(รูปที่ 5.1.2 Test Admin Logout)

Test Admin Logout

method: post

URL: <http://localhost:3001/api/auth/logout>

## 5.2 Account



The screenshot shows the Postman application interface. At the top, it displays a green "GET" button and the URL "http://localhost:3001/api/account/get/3". Below the URL, there are tabs for "Params", "Authorization", "Headers (7)", "Body", "Scripts", and "Settings". The "Body" tab is currently selected, showing sub-tabs for "Pretty", "Raw", "Preview", and "Visualize", with "Pretty" selected. A dropdown menu next to "JSON" shows "Text" and "CSV". The main content area displays a JSON response with line numbers from 1 to 10 on the left. The JSON object contains the following fields:

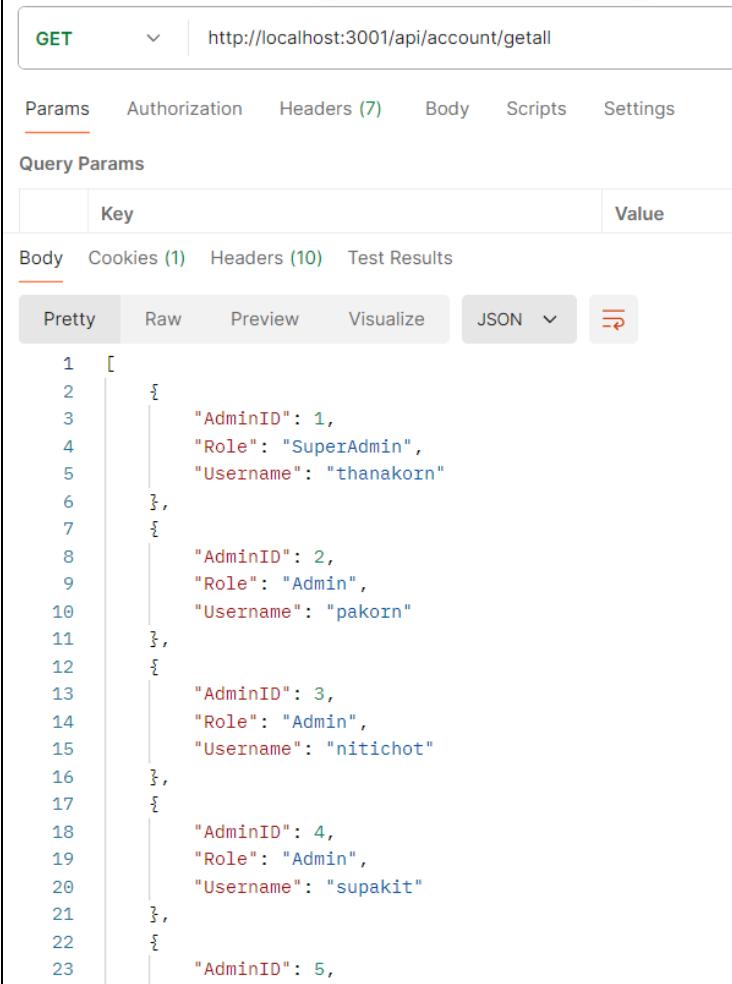
```
1 {  
2   "AdminID": 3,  
3   "FName": "Nitichot",  
4   "LName": "Chaiyasit",  
5   "Email": "nitichot.cha@dokmai4u.com",  
6   "PhoneNumber": "- ",  
7   "Role": "Admin",  
8   "Username": "nitichot",  
9   "Password": "$2b$10$RNDqTnB2XXw235YD877VNuVcPdUwM9qAii898L5pehHLwMjuGCr62"  
10 }
```

(ສູ່ປໍ່ 5.2.1 Test API to get account by ID)

Test API to get account by ID

method: get

URL: <http://localhost:3001/api/account/get/3>



The screenshot shows the Postman application interface. At the top, it displays a 'GET' method and the URL 'http://localhost:3001/api/account/getall'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Scripts', and 'Settings', with 'Params' being the active tab. Under 'Query Params', there is a table with columns 'Key' and 'Value', which is currently empty. Below the table, there are tabs for 'Body', 'Cookies (1)', 'Headers (10)', and 'Test Results', with 'Body' being the active tab. Under 'Body', there are four options: 'Pretty' (selected), 'Raw', 'Preview', and 'Visualize'. To the right of these options is a 'JSON' dropdown set to 'Pretty' and a copy icon. The main content area shows a JSON response with line numbers from 1 to 23. The JSON data represents a list of accounts:

```
1 [  
2 {  
3     "AdminID": 1,  
4     "Role": "SuperAdmin",  
5     "Username": "thanakorn"  
6 },  
7 {  
8     "AdminID": 2,  
9     "Role": "Admin",  
10    "Username": "pakorn"  
11 },  
12 {  
13     "AdminID": 3,  
14     "Role": "Admin",  
15     "Username": "hitichot"  
16 },  
17 {  
18     "AdminID": 4,  
19     "Role": "Admin",  
20     "Username": "supakit"  
21 },  
22 {  
23     "AdminID": 5,
```

(รูปที่ 5.2.2 Test API to get all accounts)

Test API to get all accounts

method: get

URL: <http://localhost:3001/api/account/getall>

POST http://localhost:3001/api/account/search

Params Authorization Headers (9) Body Scripts Setting

none  form-data  x-www-form-urlencoded  raw  binary

```

1  {
2    "searchKey": "nitichot",
3    "searchId": "",
4    "searchRole": ""
5  }

```

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```

1  [
2    {
3      "AdminID": 3,
4      "Role": "Admin",
5      "Username": "nitichot"
6    }
7  ]

```

(รูปที่ 5.2.3 Test API to search accounts)

Test API to search accounts

method: post

URL: <http://localhost:3001/api/account/search>

body: raw JSON

```
{
  "searchKey": "nitichot",
  "searchId": "",
  "searchRole": ""
}
```

The screenshot shows the Postman application interface. At the top, it says "POST" and the URL "http://localhost:3001/api/account/add". Below this, there are tabs for "Params", "Authorization", "Headers (9)", "Body", "Scripts", and "Sets". The "Body" tab is selected and has a radio button for "raw" selected. The JSON body is:

```

1  {
2   "FName": "Jidapa",
3   "LName": "abc",
4   "Email": "jida.pa@mahidol.com",
5   "PhoneNumber": "-",
6   "Username": "jidapa",
7   "Password": "jidapa"
8 }

```

Below the body, there are tabs for "Body", "Cookies (1)", "Headers (10)", and "Test Results". The "Body" tab is selected. It shows the response in "Pretty" format:

```

1  {
2   "message": "Inserted successfully",
3   "AdminID": 6,
4   "account": {
5     "AdminID": 6,
6     "FName": "Jidapa",
7     "LName": "abc",
8     "Email": "jida.pa@mahidol.com",
9     "PhoneNumber": "-",
10    "Username": "jidapa"
11  }
12 }

```

(រូបថត 5.2.4 Test API to add new account)

Test API to add new account

method: post

URL: <http://localhost:3001/api/account/add>

body: raw JSON

```
{
  "FName": "Jidapa",
  "LName": "abc",
  "Email": "jida.pa@mahidol.com",
  "PhoneNumber": "-",
  "Username": "jidapa",
  "Password": "jidapa"
}
```

The screenshot shows the Postman interface with a PUT request to `http://localhost:3001/api/account/update/6`. The body is set to raw JSON with the following content:

```

1  {
2   "FName": "Wudthichart",
3   "LName": "abc",
4   "Email": "wud.cha@mahidol.com",
5   "PhoneNumber": "0000000000"
6 }
```

The response body is also raw JSON, showing the result of the update:

```

1  {
2   "ok": true,
3   "updatedId": "6"
4 }
```

(រូបថត 5.2.5 Test API to update account by ID)

Test API to update account by ID

method: put

URL: `http://localhost:3001/api/account/update/6`

body: raw JSON

```
{
  "FName": "Wudthichart",
  "LName": "abc",
  "Email": "wud.cha@mahidol.com",
  "PhoneNumber": "0000000000"
}
```

The screenshot shows the Postman application interface. At the top, it displays a **DELETE** method and a URL <http://localhost:3001/api/account/delete/6>. Below the URL, there are tabs for **Params**, **Authorization**, **Headers (7)**, **Body**, **Scripts**, and **Settings**. The **Body** tab is selected, showing sub-tabs for **Pretty**, **Raw**, **Preview**, **Visualize**, **JSON** (with a dropdown arrow), and a copy icon. The **Pretty** tab is active, displaying the following JSON response:

```
1  {
2    "message": "Account deleted successfully",
3    "adminId": "6"
4 }
```

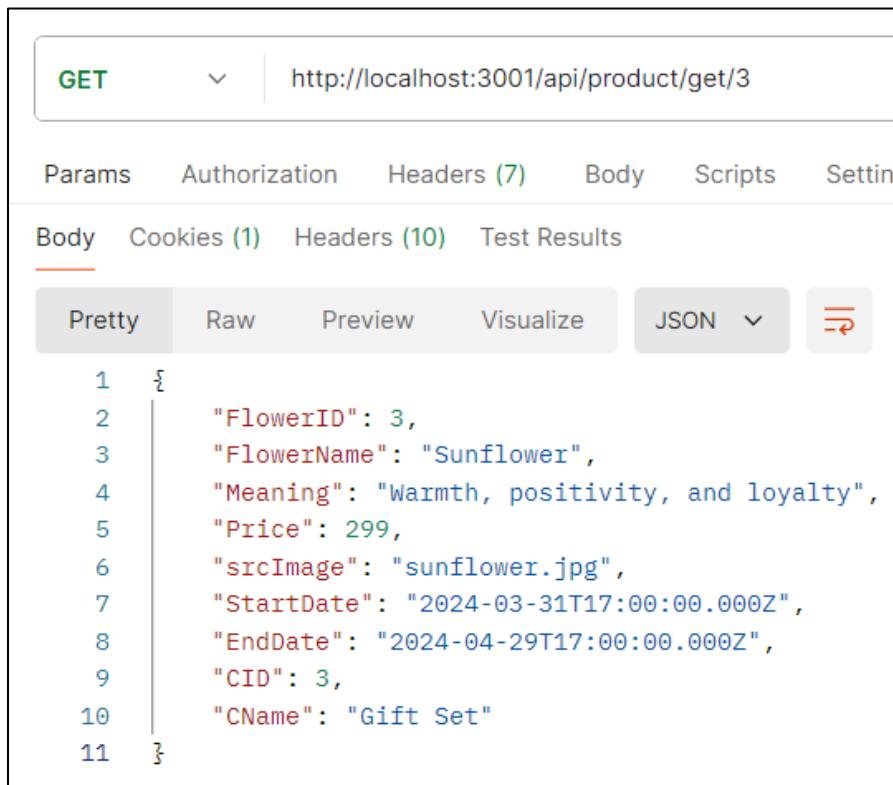
(รูปที่ 5.2.6 Test API to delete account by ID

Test API to delete account by ID

method: delete

URL: <http://localhost:3001/api/account/delete/6>

### 5.3 Product



The screenshot shows the Postman interface with a successful API call. The method is GET, and the URL is <http://localhost:3001/api/product/get/3>. The response body is displayed in a pretty-printed JSON format:

```
1 {  
2   "FlowerID": 3,  
3   "FlowerName": "Sunflower",  
4   "Meaning": "Warmth, positivity, and loyalty",  
5   "Price": 299,  
6   "srcImage": "sunflower.jpg",  
7   "StartDate": "2024-03-31T17:00:00.000Z",  
8   "EndDate": "2024-04-29T17:00:00.000Z",  
9   "CID": 3,  
10  "CName": "Gift Set"  
11 }
```

(ສູບທີ 5.3.1 Test API to get product by ID)

Test API to get product by ID

method: get

URL: <http://localhost:3001/api/product/get/3>

```

GET      | http://localhost:3001/api/product/getall
Params   Authorization Headers (7) Body Scripts Settings
Body   Cookies (1) Headers (10) Test Results
Pretty Raw Preview Visualize JSON ▾
1  [
2   {
3     "FlowerID": 1,
4     "FlowerName": "Rose",
5     "Meaning": "Symbol of love and passion",
6     "Price": 199,
7     "srcImage": "rose.jpg",
8     "StartDate": "2023-12-31T17:00:00.000Z",
9     "EndDate": "2025-12-30T17:00:00.000Z",
10    "CID": 5,
11    "CName": "Wedding"
12  },
13  {
14    "FlowerID": 2,
15    "FlowerName": "Tulip",
16    "Meaning": "Perfect love and happiness",
17    "Price": 249,
18    "srcImage": "tulips.jpg",
19    "StartDate": "2023-12-31T17:00:00.000Z",
20    "EndDate": "2024-02-13T17:00:00.000Z",
21    "CID": 2,
22    "CName": "Bouquet"
23  },
24  {
25    "FlowerID": 3,
26    "FlowerName": "Sunflower",
27    "Meaning": "Warmth, positivity, and loyalty"
}

```

(รูปที่ 5.3.2 Test API to get all products)

Test API to get all products

method: get

URL: <http://localhost:3001/api/product/getall>

POST | http://localhost:3001/api/product/search

Params | Authorization | Headers (9) | **Body** | Scripts | Set

none  form-data  x-www-form-urlencoded  raw

```

1  {
2    "searchKey": "Rose",
3    "searchId": "",
4    "CID": ""
5  }

```

Body | Cookies (1) | Headers (10) | Test Results

Pretty | Raw | Preview | Visualize | **JSON** |

```

1  [
2    {
3      "FlowerID": 1,
4      "FlowerName": "Rose",
5      "Price": 199,
6      "srcImage": "rose.jpg",
7      "CID": 5,
8      "CName": "Wedding"
9    }
10 ]

```

(รูปที่ 5.3.3 Test API to search products)

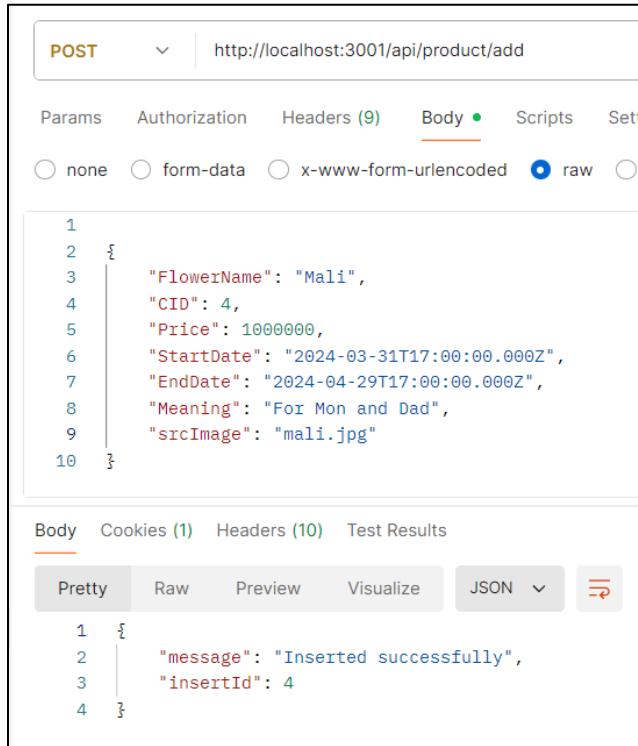
Test API to search products

method: post

URL: http://localhost:3001/api/product/search

body: raw JSON

```
{
  "searchKey": "Rose",
  "searchId": "",
  "CID": ""
}
```



(ຮູບທີ 5.3.4 Test API to add new product)

Test API to add new product

method: post

URL: http://localhost:3001/api/product/add

body: raw JSON

```
{
    "FlowerName": "Mali",
    "CID": 4,
    "Price": 1000000,
    "StartDate": "2024-03-31T17:00:00.000Z",
    "EndDate": "2024-04-29T17:00:00.000Z",
    "Meaning": "For Mon and Dad",
    "srcImage": "mali.jpg"
}
```

The screenshot shows a POSTMAN interface with the following details:

- Method:** PUT
- URL:** http://localhost:3001/api/product/update/4
- Body:** raw JSON (selected)
- JSON Content:**

```

1  {
2    "FlowerName": "Malai",
3    "CID": 4,
4    "Price": 1000000,
5    "StartDate": "2024-03-31",
6    "EndDate": "2024-04-29",
7    "Meaning": "For Mon and Dad",
8    "srcImage": "mali.jpg"
9  }

```
- Response Body:** JSON (Pretty selected)

```

1  {
2    "ok": true,
3    "updatedId": "4"
4  }

```

(ຮູບທີ 5.3.5 Test API to update product by ID)

Test API to update product by ID

method: put

URL: http://localhost:3001/api/product/update/4

body: raw JSON

```
{
  "FlowerName": "Malai",
  "CID": 4,
  "Price": 1000000,
  "StartDate": "2024-03-31",
  "EndDate": "2024-04-29",
  "Meaning": "For Mon and Dad",
  "srcImage": "mali.jpg"
}
```

The screenshot shows the Postman application interface. At the top, it displays a **DELETE** method and the URL <http://localhost:3001/api/product/delete/4>. Below the URL, there are tabs for **Params**, **Authorization**, **Headers (7)**, **Body**, **Scripts**, and **Setting**. The **Body** tab is selected and contains sub-tabs for **Pretty**, **Raw**, **Preview**, **Visualize**, **JSON** (with a dropdown arrow), and a copy icon. The **Pretty** tab is active, showing the following JSON response:

```
1  {
2    "message": "Product deleted successfully",
3    "FlowerID": "4"
4 }
```

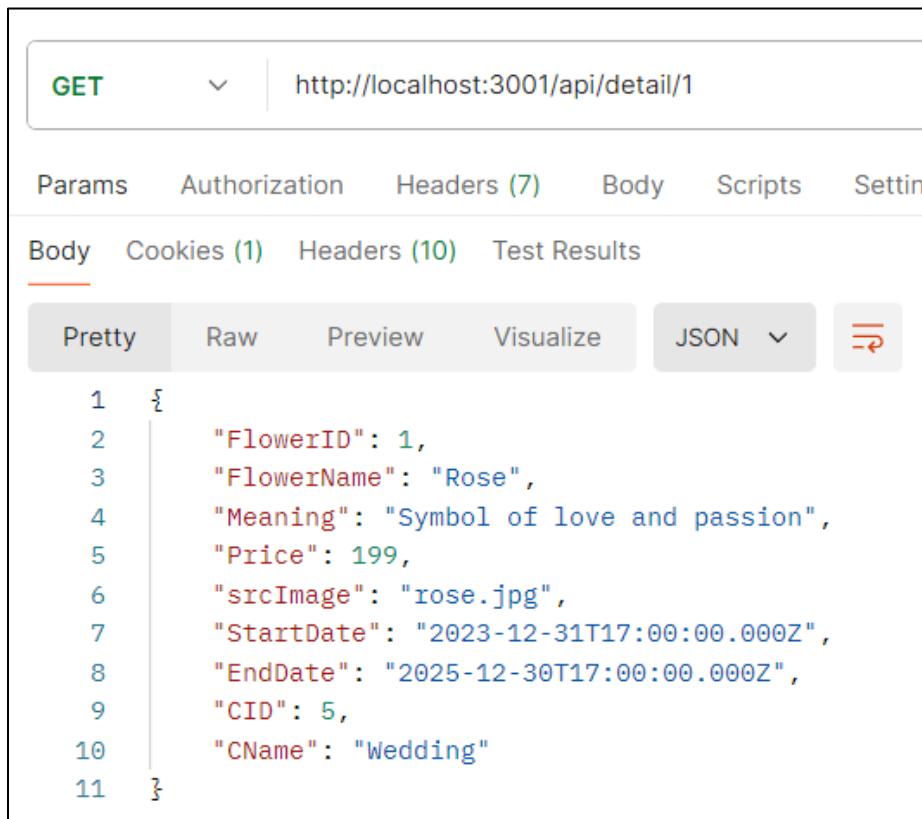
(รูปที่ 5.3.6 Test API to delete product by ID)

Test API to delete product by ID

method: delete

URL: <http://localhost:3001/api/product/delete/4>

## 5.4 User Visit



The screenshot shows the Postman application interface. At the top, it displays a green "GET" button and the URL "http://localhost:3001/api/detail/1". Below the URL, there are tabs for "Params", "Authorization", "Headers (7)", "Body", "Scripts", and "Settings". The "Body" tab is currently selected, showing sub-tabs for "Body", "Cookies (1)", "Headers (10)", and "Test Results". Under the "Body" tab, there are buttons for "Pretty", "Raw", "Preview", "Visualize", and "JSON" (with a dropdown arrow). The main content area shows a JSON response with line numbers from 1 to 11. The JSON data is as follows:

```
1  {
2    "FlowerID": 1,
3    "FlowerName": "Rose",
4    "Meaning": "Symbol of love and passion",
5    "Price": 199,
6    "srcImage": "rose.jpg",
7    "StartDate": "2023-12-31T17:00:00.000Z",
8    "EndDate": "2025-12-30T17:00:00.000Z",
9    "CID": 5,
10   "CName": "Wedding"
11 }
```

(ຮູບທີ 5.4.1 Test API to get product detail by ID)

Test API to get product detail by ID

method: get

URL: <http://localhost:3001/api/detail/1>

GET http://localhost:3001/api/search?FlowerName=&Category=&MaxPrice=250&MinPrice=200

Params • Authorization Headers (9) Body • Scripts Settings

Query Params

	Key	Value
<input checked="" type="checkbox"/>	FlowerName	
<input checked="" type="checkbox"/>	Category	
<input checked="" type="checkbox"/>	MaxPrice	250
<input checked="" type="checkbox"/>	MinPrice	200

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON ↻

```

1  [
2   {
3     "FlowerID": 2,
4     "FlowerName": "Tulip",
5     "Meaning": "Perfect love and happiness",
6     "Price": 249,
7     "srcImage": "tulips.jpg",
8     "StartDate": "2023-12-31T17:00:00.000Z",
9     "EndDate": "2024-02-13T17:00:00.000Z",
10    "CID": 2,
11    "CName": "Bouquet"
12  }
13 ]

```

(รูปที่ 5.4.2 Test API to search products)

Test API to search products

method: get

URL: http://localhost:3001/api/search?FlowerName=&Category=&MaxPrice=&MinPrice=

## ข้อความเปิดเผยการใช้เทคโนโลยีปัญญาประดิษฐ์

ข้าพเจ้าขอรับรองว่าได้มีการใช้เทคโนโลยี Generative AI จากเครื่องมือ Chatgpt, Gemini

ในรายวิชา ITDS241\_Web Technologies and Applications

ในการจัดทำกิจกรรมหรือชิ้นงานนี้ โดยนำเครื่องมือ Generative AI มาใช้ในระดับ (เลือกระดับ 2 - 5)

เพื่อวัตถุประสงค์ดังต่อไปนี้:

- ระดมความคิด
- ร่างชิ้นงานเบื้องต้น
- ตรวจสอบและแก้ไขภาษา ไวยากรณ์ หรือการแปล
- สรุปและถอดความเนื้อหา
- แก้ไขและตรวจสอบโค้ด
- เขียนโค้ด
- ใช้ AI เป็นส่วนหนึ่งในผลงาน
- อื่น ๆ (โปรดระบุ): \_\_\_\_\_

ธรรนากร คันศร 6787104

ปรรณ์ นิมนานล 6787051

นิธิโชค ไชยสิทธิ 6787048

ศุภกิตติ สุวรรณ 6787080

## แหล่งอ้างอิง

Potico. (n.d.). Honey Bee [Product image]. Retrieved from <https://potico.co.th/product/honey-bee>

Potico. (n.d.). Sofia Pink Tulips Bouquet [Product image]. Retrieved from <https://potico.co.th/product/sofia-pink-tulips-bouquet>

Potico. (n.d.). Beautiful You [Product image]. Retrieved from  
<https://potico.co.th/product/beautiful-you>