

Mai Bood

CS332/ 232: Introduction to Cloud Computing Technology



กลุ่มที่ 7

Section: 450002

นายทตต์ดนาย บุญเอม 6609611964

นายนวรตกร เนตรศิริ 6609612053

นายนทธพงค์ คันหมูล 6609612061

นายพันธนา บุราณรัตน์ 6609612145

นายสิริวิชญ์ ยิ่มศร瓦ล 6609612244

นายอภินัฐ สีบบุญ 6609612293

นายสวิศ บุญนี 6609681173

นายชิษณุชา ทนศิริ 6609681330

## สารบัญ

ที่มาและความสำคัญ.....	3
ความสำคัญของโปรเจกต์.....	3
กรณีการใช้งาน (Use cases) และประโยชน์สำหรับผู้ใช้.....	4
สถาปัตยกรรมระบบ .....	6
คำอธิบายการทำงานร่วมกันขององค์ประกอบต่าง ๆ ในสถาปัตยกรรม.....	6
หลักที่สำคัญในการสร้างและตั้งค่าแต่ละองค์ประกอบ .....	7
S3 .....	7
EC2 .....	9
RDS .....	10
Lambda .....	13
API Gateway.....	17
EventBridge.....	20
ผลการทดสอบการทำงานของระบบโซลูชันตามมุมมองของผู้ใช้ (end-to-end) .....	23
ไฟล์อื่น ๆ ในโครงการ .....	32
ผลการดำเนินงาน และแนวทางพัฒนาต่อในอนาคต .....	32

## ที่มาและความสำคัญ

ในปัจจุบันปัญหา food waste เป็นปัญหาใหญ่ในหลายประเทศ เพราะเป็นปัญหาที่ส่งผลกระทบต่อสิ่งแวดล้อมในหลายด้าน ข้อมูลจาก องค์การอาหารและเกษตรแห่งสหประชาชาติ (FAO) ระบุว่า ประมาณหนึ่งในสามของอาหารที่ผลิตขึ้นทั่วโลกถูกทิ้งโดยไม่ได้รับการบริโภค โดยส่วนใหญ่จะเกิดจาก อาหารหมดอายุ หรือใช้ได้ไม่ทันเวลา

สาเหตุของปัญหานี้เกิดจากการที่ผู้บริโภค จัดการกับวัตถุดิบได้ไม่ดีพอ ซึ่งวัตถุดิบมามากเกินความจำเป็น หรือ ไม่รู้ว่าจะนำของที่เหลืออยู่ในตู้เย็นไปใช้ทำอะไร ส่งผลให้อาหารเหลือทิ้งและกลายเป็น food waste

จากปัญหาดังกล่าว จึงเกิดการพัฒนาโครงการนี้ขึ้นมา โดยเป็นเว็บไซต์ที่จะทำการแจ้งเตือนผ่านแอปพลิเคชัน Line เมื่ออาหารใกล้หมดอายุ มีระบบช่วยให้ผู้ใช้ทำงานดูดิบที่ไม่ได้ใช้ข้าวแลกเปลี่ยน หรือแจกจ่ายกันได้ และ มีระบบช่วยแนะนำการประกอบอาหารจากวัตถุดิบที่เหลืออยู่ เพื่อให้เกิดการใช้ทรัพยากรอย่างคุ้มค่า และ ลดการสูญเสียอาหารที่ไม่จำเป็น

## ความสำคัญของโปรเจคนี้

- การจัดการวัตถุดิบอย่างมีประสิทธิภาพ : ช่วยให้สามารถจัดการกับวัตถุดิบให้หมดได้ก่อนวันหมดอายุ
- การประหยัดค่าใช้จ่ายในแต่ละครัวเรือน : การใช้วัตถุดิบได้หมดโดยไม่เหลือทิ้งช่วยให้ประหยัดค่าใช้จ่ายลงได้
- การลดปัญหา food waste : การใช้วัตถุดิบมาทำอาหารให้หมดก่อนวันหมดอายุจะช่วยลดขยะอาหารลงอย่างมาก

## กรณีการใช้งาน (Use cases) และประโยชน์สำหรับผู้ใช้

### Use Case ที่ 1: แสดงรายการวัตถุดิบใกล้หมดอายุ

Actor: ผู้ใช้งานทั่วไป

เป้าหมาย: ตรวจสอบวัตถุดิบที่ใกล้หมดอายุเพื่อใช้งานก่อนวันหมดอายุ

ขั้นตอนการใช้งาน:

- ผู้ใช้งานบันทึกข้อมูลวัตถุดิบพร้อมวันหมดอายุด้วยเว็บไซต์
- ระบบแสดงรายการวัตถุดิบที่ใกล้หมดอายุ
- ผู้ใช้งานตรวจสอบวัตถุดิบที่ใกล้หมดอายุผ่านแอปพลิเคชัน Line

ผลลัพธ์: ผู้ใช้งานรู้ว่าวัตถุดิบใดควรใช้ก่อน

ประโยชน์สำหรับผู้ใช้:

- บริหารจัดการวัตถุดิบได้อย่างมีประสิทธิภาพ
- ลดปริมาณ food waste ที่เกิดจากการลืมใช้วัตถุดิบ

### Use Case ที่ 2: แจกจ่ายวัตถุดิบให้ผู้ใช้งานในพื้นที่ใกล้เคียง

Actor: ผู้ใช้งานทั่วไป

เป้าหมาย: ส่งต่อวัตถุดิบที่ไม่ได้ใช้งานให้ผู้ที่อยู่ใกล้เคียง

ขั้นตอนการใช้งาน:

- ผู้ใช้งานกรอกรายละเอียดวัตถุดิบที่ต้องการแจกจ่าย
- ระบบแสดงรายการวัตถุดิบที่มีการแจกจ่ายจากผู้ใช้งานคนอื่นทั้งในพื้นที่ใกล้เคียงและพื้นที่อื่น ๆ ทั่วหมด
- ผู้ใช้งานสามารถเรียกดูรายการวัตถุดิบที่แจก และติดต่อกันภายนอกรอบหากสนใจ

ผลลัพธ์: วัตถุดิบที่เหลือใช้สามารถนำไปใช้โดยผู้ที่อยู่ในพื้นที่เดียวกัน

ประโยชน์สำหรับผู้ใช้:

- ลดของเสียจากวัตถุดิบที่ไม่ได้ใช้
- ช่วยให้ผู้ที่ต้องการสามารถเข้าถึงวัตถุดิบได้โดยไม่ต้องเสียค่าใช้จ่าย
- สนับสนุนแนวคิดแบ่งปันทรัพยากรในชุมชน

### Use Case ที่ 3: แนะนำเมนูจากวัตถุดิบที่ใกล้หมดอายุ

Actor: ผู้ใช้งานทั่วไป

เป้าหมาย: ค้นหาเมนูอาหารที่สามารถทำได้จากวัตถุดิบที่มีอยู่และใกล้หมดอายุ

ขั้นตอนการใช้งาน:

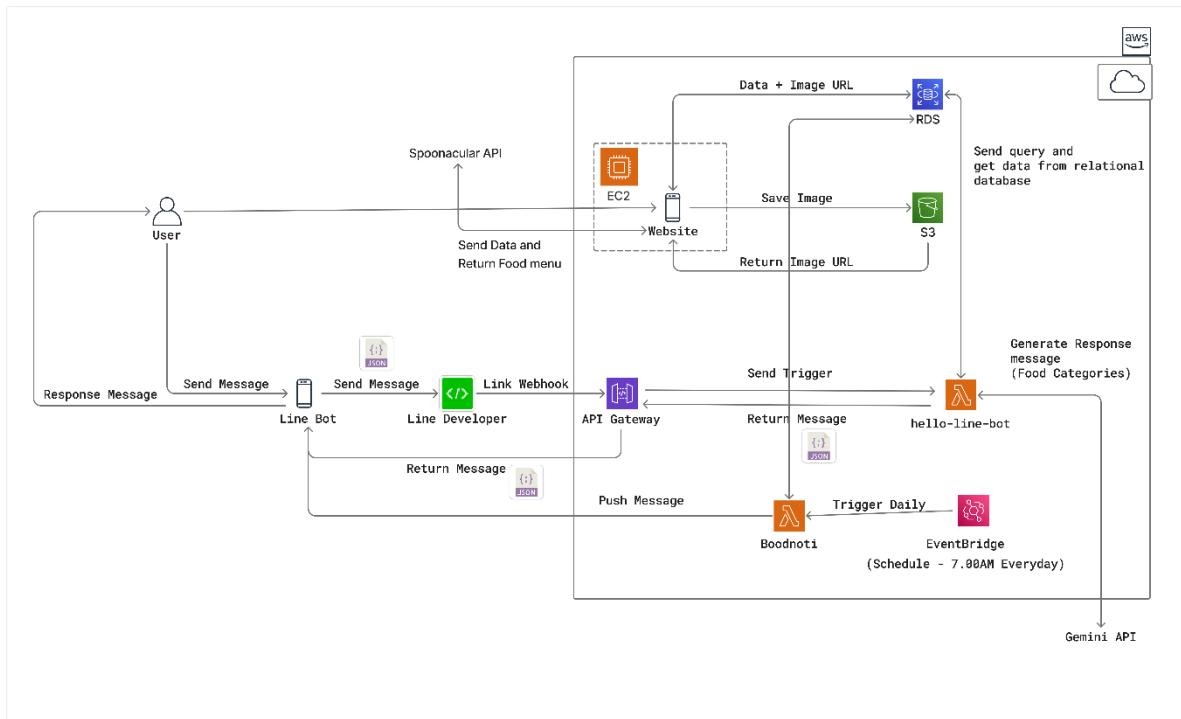
1. ผู้ใช้งานเลือกวัตถุดิบที่ตนเองมีอยู่
2. ระบบแสดงเมนูอาหารที่สามารถทำได้จากวัตถุดิบเหล่านั้น
3. ผู้ใช้งานเลือกเมนูที่สนใจและดูข้อมูล

ผลลัพธ์: ผู้ใช้งานรู้ว่าวัตถุดิบที่มีสามารถใช้ทำเมนูได้บ้าง

ประโยชน์สำหรับผู้ใช้:

- ลด food waste ด้วยการใช้วัตถุดิบอย่างคุ้มค่า
- ช่วยวางแผนมื้อาหารได้ง่ายขึ้น
- ลดค่าใช้จ่ายจากการซื้อวัตถุดิบใหม่โดยไม่จำเป็น

## สถาปัตยกรรมระบบ



### คำอธิบายการทำงานร่วมกันขององค์ประกอบต่าง ๆ ในสถาปัตยกรรม

ระบบของเราเปิดให้ผู้ใช้บริการสามารถใช้งานได้ทั้งผ่านเว็บไซต์และแอปพลิเคชัน Line โดยในส่วนของเว็บไซต์ ระบบถูกนำเข้ามาให้บริการบนเซิร์ฟเวอร์ Amazon EC2 เพื่อรับการใช้งาน และมีการทำงานร่วมกับ Amazon S3 และ Amazon RDS โดยเมื่อผู้ใช้ต้องการส่งข้อมูลต่าง ๆ เช่น ข้อมูลการสมัครใช้งานหรือข้อมูลอาหารที่ต้องการบันทึก ระบบจะจัดเก็บข้อมูลเหล่านี้ไว้ใน Amazon RDS ขณะที่ไฟล์รูปภาพจะถูกจัดเก็บใน Amazon S3 และนำ URL endpoint ของรูปภาพนั้นมาเก็บไว้ใน Amazon RDS แทน เมื่อเว็บไซต์ต้องแสดงผลข้อมูลต่าง ๆ ระบบจะดึงข้อมูลจาก Amazon RDS มาใช้แสดงในหน้าต่าง ๆ ให้แก่ผู้ใช้บริการ นอกจากนี้ยังมีการใช้งาน Spoonacular API เพื่อแนะนำเมนูอาหาร โดยเว็บไซต์จะส่งข้อมูลวัตถุต่างๆ ผ่าน API ดังกล่าว จากนั้นระบบจะรับข้อมูลเมนูอาหารที่แนะนำกลับมาแสดงแก่ผู้ใช้

และในส่วนของ Line Chatbot ระบบ Backend ทั้งหมดจะถูก Deploy ไว้บน AWS Lambda ซึ่งจะมีการดึงข้อมูลจาก Amazon RDS มาใช้ โดย AWS Lambda ในระบบแบ่งเป็น 2 Functions คือ พิงก์ชั่น ตอบกลับข้อความผู้ใช้ และ พิงก์ชั่นแจ้งเตือนอาหารใกล้หมดอายุด้วยข้อความแบบพุช โดยพิงก์ชั่นตอบกลับผู้ใช้จะต้องถูก Trigger ผ่าน API Gateway ที่เชื่อม Webhook ไว้กับ Line Developer Console ทุกรอบที่ผู้ใช้ส่งข้อความเข้ามา (Event) จะทำให้ Lambda function ถูก Trigger และส่งผ่านข้อมูลด้วย API Gateway และจะประมวลผลและเลือกข้อความตอบกลับส่งไปให้ผู้ใช้ตาม Reply token ต่อไป ส่วนใน พิงก์ชั่นแจ้งเตือนอาหารใกล้หมดอายุด้วยข้อความแบบพุช เราให้ Event ถูกสร้างขึ้นโดย Schedule ใน Amazon

EventBridge (แบบ Cron) ซึ่งจะส่ง Event มา Trigger Lambda ทุก ๆ 7.00 น. ของทุกวัน ทำให้มีการแจ้งเตือนส่งไปหาผู้ใช้ที่ผูกบัญชีไว้ในทุก ๆ วัน

## หลักที่สำคัญในการสร้างและตั้งค่าแต่ละองค์ประกอบ

### S3

ใช้ S3 เพื่อจัดเก็บรูปภาพที่ใช้ใน web application ห้องรูปภาพอาหารที่ถ่ายโดย user รวมไปถึง profile ของ user ทุกภาพจะถูกจัดเก็บใน AWS S3 และการเก็บเข้า database โดยตรง และเก็บเพียง URL ของภาพลงใน database และ เพื่อความมีประสิทธิภาพในการใช้งานในเรื่องความเร็วในการเข้าถึง การใช้ทรัพยากรอย่างคุ้มค่าและเหมาะสม

#### 1. สร้าง S3 Bucket โดยใช้ชื่อ mai-bood

#### 2. อนุญาต public access

3. กด create bucket
4. แก้ไข permission bucket policy โดยใช้ policy

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowPublicRead",  
      "Effect": "Allow",  
      "Principal": "*",  
      "Action": "s3:GetObject",  
      "Resource": "arn:aws:s3:::mai-bood /*"  
    }  
  ]  
}
```

5. เชื่อมต่อ S3 กับ Expressjs server โดยใช้ aws-sdk/client-s3

## EC2

ใช้ EC2 เพื่อ deploy web application เนื่องจากเป็นวิธีที่พื้นฐานที่สุดและสามารถจัดการทรัพยากรของ server ได้ด้วยตัวเอง

### 1. สร้าง ec2 instant ใช้ OS Ubuntu

The screenshot shows the 'Launch an instance' wizard. Step 1: 'Name and tags' (Name: mai-bood). Step 2: 'Application and OS Images (Amazon Machine Image)' (Search bar: Q. Search our full catalog including 1000s of application and OS images, Quick Start tab selected, recent AMIs: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian). Step 3: 'Amazon Machine Image (AMI)' (Selected: Ubuntu Server 24.04 LTS (HVM), SSD Volume Type, ami-01938df366ac2d954 (64-bit (x86))), Free tier eligible. Step 4: 'Summary' (Number of instances: 1, Software Image (AMI): Canonical, Ubuntu, 24.04, amd64, ami-01938df366ac2d954, Virtual server type (instance type): t2.micro, Firewall (security group): New security group, Storage (volumes): 1 volume(s) - 8 GiB). Step 5: 'Free tier' information (In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet).

### 2. สร้าง key pair ขึ้นมาและเก็บไว้ในคอมพิวเตอร์ส่วนตัว

The screenshot shows the 'Instances' launch wizard. Step 1: 'Description' (Ubuntu Server 24.04 LTS (HVM), EBS General Purpose (SSD) Volume Type, Support available from Canonical (<http://www.ubuntu.com/cloud/services>)). Step 2: 'Instance type' (t2.micro, Family: t2, 1 vCPU, 1 GiB Memory, Current generation: true, On-Demand Ubuntu Pro base pricing: 0.0164 USD per Hour, On-Demand RHEL base pricing: 0.0146 USD per Hour, On-Demand SUSE base pricing: 0.0146 USD per Hour, Free tier eligible, All generations, Compare instance types). Step 3: 'Key pair (login)' (mai-bood-key, Create new key pair). Step 4: 'Summary' (Number of instances: 1, Software Image (AMI): Canonical, Ubuntu, 24.04, ami-01938df366ac2d954, Virtual server type (instance type): t2.micro, Firewall (security group): New security group, Storage (volumes): 1 volume(s) - 8 GiB). Step 5: 'Free tier' information (In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet).

### 3. เลือก

- Allow SSH traffic from
- Allow HTTPS traffic from the internet
- Allow HTTP traffic from the internet

**Firewall (security groups) Info**

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group
  Select existing security group

We'll create a new security group called "launch-wizard-3" with the following rules:

- Allow SSH traffic from Anywhere
 

0.0.0.0/0
- Allow HTTPS traffic from the internet
 

To set up an endpoint, for example when creating a web server
- Allow HTTP traffic from the internet
 

To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Cancel
Launch Instance
Preview code

- กด Launch instant
  - เมื่อทำตามขั้นตอนข้างต้นทั้งหมดจะสามารถเข้าควบคุม VM ได้โดยใช้ key pair จากนั้นก็เพียงแค่ clone project จาก GitHub และทำการติดตั้ง library ต่าง ๆ เพิ่มไฟล์ .env เพื่อทำการตั้งค่า project

ใช้ PM2 ในการจัดการควบคุมการ run ของ backend และใช้ NGINX ในการเป็น static file server สำหรับแสดงผล frontend

RDS

ใช้เป็น Database เพื่อเก็บข้อมูล Member และอาหารใน Fridge โดยจะมีการทำเป็น Multi-AZ เพื่อป้องกันเวลาที่ AZ หนึ่งใช้งานไม่ได้ จะสลับไปอีก AZ เพื่อให้ระบบทำงานต่อได้

1. สร้าง RDS โดยเลือก Engine options เป็น MySQL

**Create database** [Info](#)

**Choose a database creation method**

**Standard create**  
You set all of the configuration options, including ones for availability, security, backups, and maintenance.

**Easy create**  
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

---

**Engine options**

**Engine type** [Info](#)

<input type="radio"/> Aurora (MySQL Compatible) 	<input type="radio"/> Aurora (PostgreSQL Compatible) 	<input checked="" type="radio"/> MySQL 
<input type="radio"/> PostgreSQL 	<input type="radio"/> MariaDB 	<input type="radio"/> Oracle  <b>ORACLE®</b>
<input type="radio"/> Microsoft SQL Server 	<input type="radio"/> IBM Db2  <b>IBM Db2</b>	

**Edition**

MySQL Community

## 2. เลือก templates เป็นแบบ Free tier

**Templates**  
Choose a sample template to meet your use case.

**Production**  
Use defaults for high availability and fast, consistent performance.

**Dev/Test**  
This instance is intended for development use outside of a production environment.

**Free tier**  
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. [Info](#)

**Availability and durability**  
Deployment options [Info](#)  
Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you choose. Learn more in the [Amazon RDS service level agreement \(SLA\)](#).

**Multi-AZ DB cluster deployment (3 instances)**  
Creates a primary DB instance with two readable standbys in separate Availability Zones. This setup provides:

- 99.99% uptime
- Redundancy across Availability Zones
- Increased read capacity
- Reduced write latency

**Multi-AZ DB instance deployment (2 instances)**  
Creates a primary DB instance with a non-readable standby instance in a separate Availability Zone. This setup provides:

- 99.95% uptime
- Redundancy across Availability Zones

**Single-AZ DB instance deployment (1 instance)**  
Creates a single DB instance without standby instances. This setup provides:

- 99.5% uptime
- No data redundancy

The diagram illustrates three deployment options for Amazon RDS:

- Multi-AZ DB cluster deployment (3 instances):** Shows a primary instance in AZ 1 and two readable standbys in AZ 2 and AZ 3. The primary instance is labeled "Primary instance + SSD". The standbys are labeled "Readable standby + SSD".
- Multi-AZ DB instance deployment (2 instances):** Shows a primary instance in AZ 1 and one non-readable standby in AZ 2. The primary instance is labeled "Primary instance". The standby is labeled "Standby".
- Single-AZ DB instance deployment (1 instance):** Shows a single primary instance in AZ 1. It is labeled "Primary instance".

## 3. ในส่วน Settings ให้กรอกชื่อ RDS, Master username, และ Master password

**Settings**

**DB instance identifier** [Info](#)  
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

**Credentials Settings**

**Master username** [Info](#)  
Type a login ID for the master user of your DB instance.

1 to 16 alphanumeric characters. The first character must be a letter.

**Credentials management**  
You can use AWS Secrets Manager or manage your master user credentials.

**Managed in AWS Secrets Manager - most secure**  
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

**Self managed**  
Create your own password or have RDS create a password that you manage.

**Master password** [Info](#)  
  
Password strength **Strong**  
Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / \ ^ @  
**Confirm master password** [Info](#)

## 4. กด Create Database

5. ในหน้า Database ให้เลือก RDS ที่สร้าง จากนั้นกด Modify
- a. ในหน้า Modify ส่วนของ Availability & durability - Multi-AZ deployment ให้เลือก Create a standby instance (recommended for production usage)

#### Availability & durability

##### Multi-AZ deployment [Info](#)

- Create a standby instance (recommended for production usage)

Creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

- Do not create a standby instance

b. กด Continue

c. ในหน้า Schedule modifications ให้เลือก Apply immediately

#### Modify DB instance: mai-bood

##### Summary of modifications

You are about to submit the following modifications. Only values that will change are displayed. Carefully verify your changes and click Modify DB Instance.

Attribute	Current value	New value
No modifications selected		

##### Schedule modifications

###### When to apply modifications

- Apply during the next scheduled maintenance window

Current maintenance window: June 01, 2025 16:46 - 17:16 (UTC+07:00)

- Apply immediately

The modifications in this request and any pending modifications will be asynchronously applied as soon as possible, regardless of the maintenance window setting for this database instance.

d. กด Modify DB instance

เมื่อทำการสร้าง database สำเร็จสามารถใช้ endpoint, ชื่อผู้ใช้ และรหัสผ่านที่ตั้งในการเข้าใช้งาน

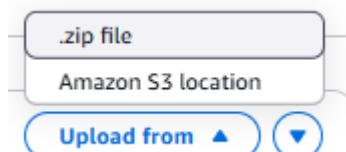
## Lambda

ฟังก์ชันสำหรับตอบกลับข้อความผู้ใช้

1. สร้างฟังก์ชันใหม่โดยตั้งค่าดังนี้
  - a. เลือก Author from scratch
  - b. ตั้งชื่อ Function
  - c. Runtime ใช้ Node.js 20.X
  - d. Architecture ใช้ x86\_64
  - e. Execution role เลือก Labrole

The screenshot shows the 'Create function' wizard in the AWS Lambda console. It has three main options at the top: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. Below these are sections for 'Basic information' (Function name: myFunctionName, Runtime: Node.js 22.x, Architecture: x86\_64), 'Permissions' (Info: By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs.), and 'Change default execution role'. At the bottom right are links for CloudShell, Feedback, and a copyright notice.

2. อัปโหลด zip file source code ของฟังก์ชัน ที่มา: <https://github.com/Chisanucha-Thonsiri/LineChatbotWithAWS.git>



3. Deploy function ที่อัปโหลด

The screenshot shows the 'Code source' tab in the AWS Lambda console. It displays the deployed function code (hello-line-bot) and a deployment history panel on the right. The deployment history shows a successful deployment with a timestamp of 2023-09-11T10:45:00Z and a log link.

## 4. เพิ่ม API Gateway เป็น Trigger สำหรับฟังก์ชันนี้

The screenshot shows the AWS Lambda console interface. At the top, it displays the Lambda service icon, the function name 'hello-line-bot', and the region 'United States (N. Virginia)'. On the left, there's a navigation bar with 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Configuration' tab is currently active. In the main content area, there's a diagram showing the function 'hello-line-bot' connected to an 'API Gateway' trigger. Below the diagram, there are buttons for '+ Add destination' and '+ Add trigger'. To the right of the diagram, there's a 'Description' section with details like 'Last modified 20 hours ago', 'Function ARN arn:aws:lambda:us-east-1:869310007132:function:hello-line-bot', and a 'Function URL' link. The 'Triggers' section shows one entry: 'API Gateway: hello-chatbot' with the ARN 'arn:aws:execute-api:us-east-1:869310007132:ird17hkb09/\*POST/' and endpoint 'https://ird17hkb09.execute-api.us-east-1.amazonaws.com/hello-line-bot/'. There are also 'Edit', 'Delete', and 'Add trigger' buttons for this trigger.

## 5. เพิ่ม Environment Variable ที่ใช้ในฟังก์ชัน

The screenshot shows the 'Configuration' tab of the AWS Lambda console for the 'hello-line-bot' function. On the left, there's a sidebar with options like 'General configuration', 'Triggers', 'Permissions', 'Destinations', 'Function URL', 'Environment variables' (which is currently selected), 'Tags', 'VPC', 'RDS databases', 'Monitoring and operations tools', 'Concurrency and recursion detection', 'Asynchronous invocation', and 'Code signing'. The main content area shows the 'Environment variables' section with a table titled 'Environment variables (8)'. The table has two columns: 'Key' and 'Value'. The keys listed are AI, DB, HOST, LINE\_BOT\_TOKEN, LINE\_SECRET, PASSWORD, SpoonAPIKEY, and USER. The values are partially obscured by a black redaction box. There is also an 'Edit' button at the top right of the table.

ฟังก์ชันพุ่มข้อความแจ้งเตือนใกล้หมดอายุ

### 1. สร้างฟังก์ชันใหม่โดยตั้งค่าดังนี้

- e. เลือก Author from scratch
- f. ตั้งชื่อ Function
- g. Runtime ใช้ Node.js 22.X
- h. Architecture ใช้ x86\_64
- i. Execution role เลือก Labrole

Screenshot of the AWS Lambda 'Create function' wizard.

**Create function** Info

Choose one of the following options to create your function.

- Author from scratch
 Start with a simple Hello World example.
- Use a blueprint
 Build a Lambda application from sample code and configuration presets for common use cases.
- Container image
 Select a container image to deploy for your function.

**Basic information**

**Function name**  

Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

**Runtime** Info  

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

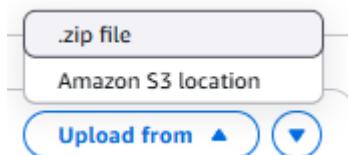
**Architecture** Info  
 arm64
  x86\_64

**Permissions** Info  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

[Change default execution role](#)

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

2. อัปโหลด zip file source code ของฟังก์ชัน ที่มา: <https://github.com/Chisanucha-Thonsiri/LineChatbotWithAWS.git> (ไฟล์เดอร์: Expire Notification)



3. Deploy function ที่อัปโหลด

Screenshot of the AWS Lambda 'Code' tab for the 'BoodNoti' function.

**Code source** Info

**EXPLORER**

- BOODNOTI
  - node\_modules
  - index.mjs
  - package.json
  - package-lock.json
- DEPLOY
  - Deploy (Ctrl+Shift+U)
  - Test (Ctrl+Shift+I)

**Code Editor**

```
JS index.mjs
1 import { createConnection } from 'mysql2/promise';
2
3 const HOST = process.env.HOST;
4 const USER = process.env.USER;
5 const PASSWORD = process.env.PASSWORD;
6 const DB = process.env.DB;
7 const noIMG = "https://maibood.s3.us-east-1.amazonaws.com/noImage.png";
8
9 export const handler = async (event) => {
10   const pushMessage = await foodExpsoon(connection) => {
11     try {
12       const loading... = await connection.execute(
13         'SELECT user_line FROM members WHERE id = ?;', [foodExpsoon.owner]
14       );
15       const date = new Date(foodExpsoon.exp);
16       const thote = new Intl.DateTimeFormat('th-TH', {
17         year: 'numeric',
18         month: 'long',
19         day: 'numeric'
20       ).format(date);
21       pushMessage(`Loading... ${thote}`);
22     } catch (error) {
23       pushMessage(`Error: ${error.message}`);
24     }
25   };
26 }
```

Upload from ▼

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

4. ตั้งค่า Timeout เป็น 10 – 30 วินาที เพื่อรองรับการรันโค้ดซึ่งใช้เวลานานในกรณีที่มีข้อมูลใน Database มาก และเพิ่ม Environment Variable ที่ใช้ในฟังก์ชัน

The image shows two screenshots of the AWS Lambda console. The top screenshot displays the 'General configuration' settings for a function named 'BoodNoti'. It shows Memory set to 128 MB and Ephemeral storage set to 512 MB. The Timeout setting is listed as 0 min 30 sec. The bottom screenshot shows the 'Edit environment variables' screen for the same function. It lists several environment variables with their corresponding values and 'Remove' buttons: AI (value redacted), DB (value redacted), HOST (value redacted), LINE\_CHANNEL\_ACCESS\_TOKEN (value redacted), PASSWORD (value redacted), and USER (value redacted). A button for 'Add environment variable' is visible at the bottom left. The 'Encryption configuration' section is collapsed. At the bottom right of the screenshot, there are 'Cancel' and 'Save' buttons, along with links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

## API Gateway

- เลือก Create API และในหน้านี้เลือก Build REST API

The screenshot shows the 'Create API' section of the AWS API Gateway console. It lists three types of APIs:

- WebSocket API**: Described as building a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards. It works with Lambda, HTTP, and AWS Services. A 'Build' button is present.
- REST API**: Described as developing a REST API where you gain complete control over the request and response along with API management capabilities. It works with Lambda, HTTP, and AWS Services. Buttons for 'Import' and 'Build' are shown.
- REST API Private**: Described as creating a REST API that is only accessible from within a VPC. It works with Lambda, HTTP, and AWS Services.

At the bottom, there are links for CloudShell and Feedback, and standard AWS footer links for Privacy, Terms, and Cookie preferences.

- สร้าง Rest API (กำหนดเพียงชื่อ ไม่ต้องใส่ URL ที่ส่วนต่อไปให้เป็น Default)

The screenshot shows the 'Create REST API' configuration page. The 'API details' section includes the following fields:

- API name**: Set to "My REST API".
- Description - optional**: An empty text area.
- API endpoint type**: Set to "Regional".
- IP address type**: Set to "IPv4".

Below these, there is a note: "Supports only edge-optimized and Regional API endpoint types." At the bottom, there are links for CloudShell and Feedback, and standard AWS footer links for Privacy, Terms, and Cookie preferences.

### 3. เลือก Create Method สำหรับ API นี้

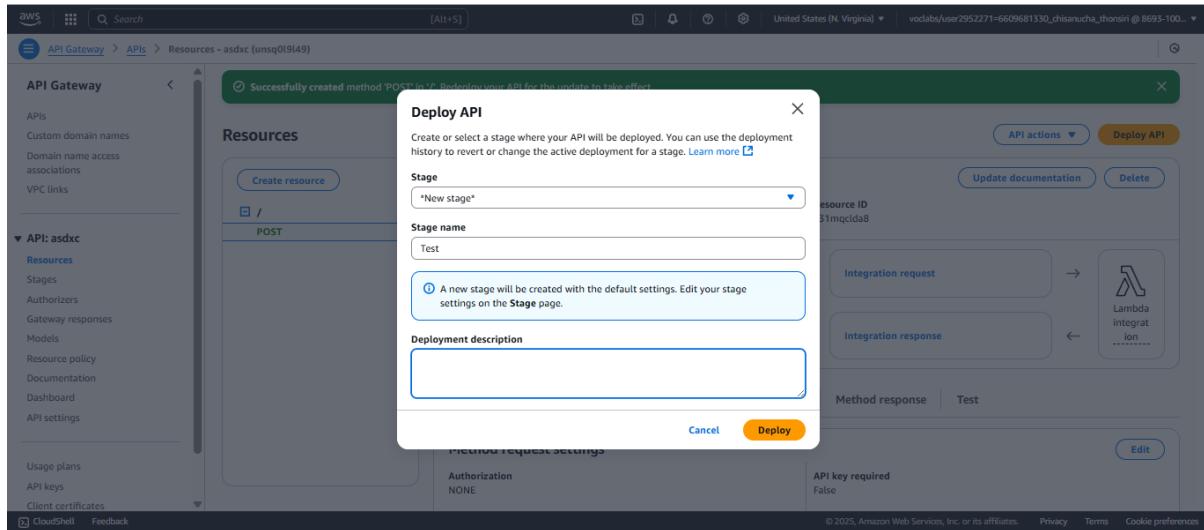
The screenshot shows the AWS API Gateway Resources page. On the left, there's a sidebar with 'API Gateway' navigation and a detailed 'API: asdxc' section. The main area is titled 'Resources' and shows a single resource with the path '/'. The 'Resource details' panel shows the Resource ID as k31mqclda8. The 'Methods (0)' panel indicates 'No methods defined.' There are buttons for 'Update documentation', 'Enable CORS', 'Delete', and 'Create method'.

### 4. ตั้งค่า Method ดังนี้

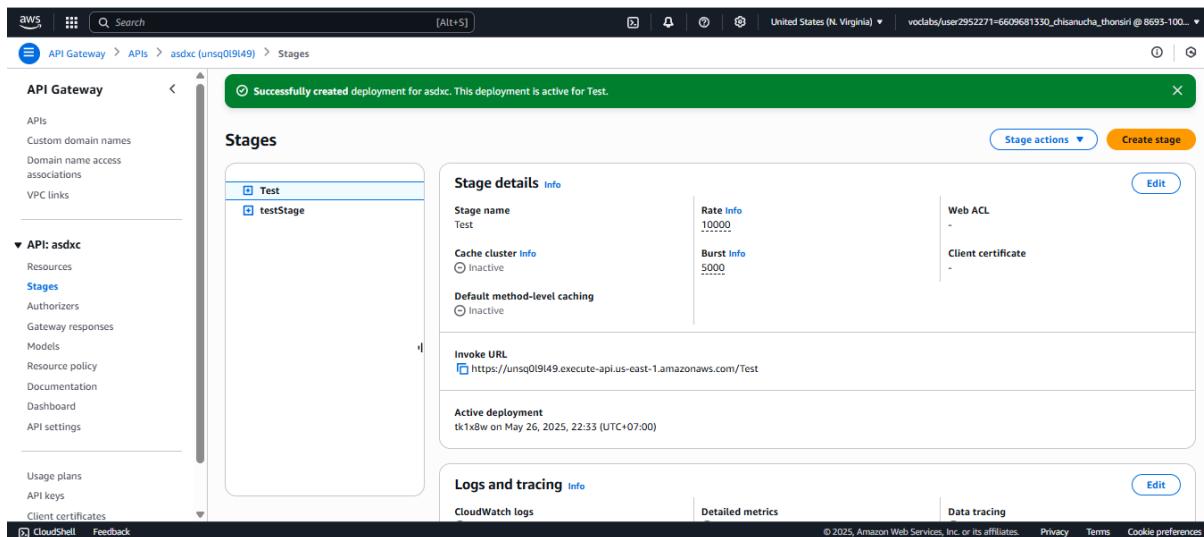
- Method type: POST
- Integration type: Lambda function
- Lambda Function: เลือก พังก์ชันที่ตอบ Response ของผู้ใช้

The screenshot shows the 'Method details' configuration page for a POST method. Under 'Method type', 'POST' is selected. Under 'Integration type', 'Lambda function' is selected, which is highlighted with a blue border. Other options like 'HTTP', 'Mock', 'AWS service', and 'VPC link' are shown without borders. Below this, there's a section for 'Lambda proxy integration' with a note about sending requests to a Lambda function. At the bottom, there's a dropdown for 'Lambda function' set to 'us-east-1' and a text input field containing 'arn:aws:lambda:us-east-1:869310007132:function:BoolNoti'.

## 5. Deploy API และสร้าง Stage ใหม่สำหรับการ Deploy API นี้



## 6. เมื่อ Deploy เสร็จสิ้นแล้ว สามารถนำ Invoke URL ไปเชื่อมต่อในส่วนของ Webhook ของ Line Messaging API



# EventBridge

## 1. เลือก Create Schedule

The screenshot shows the Amazon EventBridge Scheduler interface. On the left, there's a sidebar with navigation links like Dashboard, Developer resources, Buses, Pipes, Scheduler, and Integration. The main area has four sections: 'Templated targets', 'Universal targets', 'Flexible time windows', and 'Retries'. Below these is a table titled 'Schedules (2)' with columns for Schedule name, Schedule group, Status, Target, Target type, and Last modified. Two entries are listed: 'BoodNotical' (Enabled, Lambda\_Invoke, May 17, 2025) and 'BoodNoti' (Disabled, Lambda\_Invoke, May 16, 2025). At the bottom right of the main area, there's a 'Create schedule' button.

## 2. Schedule pattern ตั้งค่าดังนี้

- Occurrence: Recurring Schedule
- Schedule type: Cron-based schedule

### Schedule pattern

**Occurrence** | [Info](#)  
You can define an one-time or recurrent schedule.

One-time schedule

Recurring schedule

**Time zone**  
The time zone for the schedule.

(UTC+07:00) Asia/Bangkok

**Schedule type**  
Choose the schedule type that best meets your needs.

**Cron-based schedule**

A schedule set using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.

**Rate-based schedule**

A schedule that runs at a regular rate, such as every 10 minutes.

**Cron expression** | [Info](#)  
Define the cron expression for the schedule.

cron ( 0 Minutes 7 Hours \* Day of month \* Month ? Day of the week \* Year )

[Copy](#) [Clear](#)

### 3. Target detail เลือกเป็น AWS Lambda (Invoke)

The screenshot shows the 'Select target' step of creating a schedule. On the left, a sidebar lists steps: Step 1 (Specify schedule detail), Step 2 (Select target - highlighted with a blue circle), Step 3 - optional (Settings), and Step 4 (Review and create schedule). The main area is titled 'Target detail' and contains a 'Target API' section with 'Info' and a note: 'Select an API that will be invoked as a target for your schedule.' Below this is a grid of 'Templated targets' and 'All APIs'. The 'Templated targets' section includes: CodeBuild StartBuild, CodePipeline StartPipelineExecution, Amazon ECS RunTask, Amazon EventBridge PutEvents, Amazon Inspector V1 StartAssessmentRun, Kinesis Data Firehose PutRecord, Kinesis Data Streams PutRecord, AWS Lambda Invoke (highlighted with a blue border), Amazon SNS Publish, Amazon SQS SendMessage, SageMaker StartPipelineExecution, and AWS Step Functions StartExecution. At the bottom, there's an 'Invoke' section with a dropdown set to 'NONE', a 'Universal target definition' toggle switch (unchecked), and a 'Create new Lambda function' button.

### 4. ในส่วนของ Invoke เลือก Lambda function ที่ต้องการ Trigger

This screenshot shows the 'Invoke' configuration page. It features a 'Lambda function' dropdown menu with 'BoodNoti' selected, a 'Create new Lambda function' button, and a 'Universal target definition' toggle switch (unchecked). Below these are sections for 'Action after schedule completion' (set to 'NONE') and 'Permissions'. The 'Permissions' section includes a note: 'EventBridge Scheduler requires permission to send events to the target, and based on the preferences you select, integrate with other AWS services such as AWS KMS and Amazon SQS.' It has two options: 'Create new role for this schedule' (unchecked) and 'Use existing role' (checked). A 'Select an existing role' dropdown menu is open, showing 'LabRole'.

5. Review Schedule ที่สร้าง และกด Create Schedule (ไม่ต้องตั้งค่า Trigger ใน Lambda เพิ่มเติม)

The screenshot shows the Amazon EventBridge console with the following details:

**Schedule detail**

- Schedule name:** Noti
- Status:** Enabled
- Description:** +
- Schedule group name:** default
- Action after completion:** NONE
- Schedule ARN:** arn:aws:scheduler:us-east-1:869310007132:schedule/default/Noti
- Schedule start time:** -
- Schedule end time:** -
- Execution time zone:** Asia/Bangkok
- Flexible time window:** -
- Created date:** May 26, 2025, 22:50:18 (UTC+07:00)
- Last modified date:** May 26, 2025, 22:50:18 (UTC+07:00)

**Schedule**

Cron expression: [Info](#)

0	7	*	*	?	*
Minutes	Hours	Day of month	Month	Day of week	Year

Bottom navigation: Schedule | Target | Retry policy | Dead-letter queue | Encryption

Page footer: © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

ผลการทดสอบการทำงานของระบบโซลูชันตามมุ่งมอของผู้ใช้ (end-to-end)

## สถานการณ์ที่ 1

ผู้ใช้ทำการเพิ่มวัตถุดิบ แก้ไขข้อมูล และออกจากระบบ

### ขั้นตอนการทดสอบ :

- ผู้ใช้ทำการ Log in
- เพิ่มวัตถุดิบชื่อ “ขนมปังเนย” วันที่ 28 พ.ค. 2568 โดยระบบแสดงผลว่าสำเร็จ
- แก้ไขข้อมูลส่วนตัว เช่น ชื่อร้าน
- Logout

### ผลลัพธ์ที่คาดหวัง :

- มีข้อมูลของ user ที่จะทำการ login ไว้ก่อนหน้านี้แล้ว
- ใน database ของตู้เย็น มี “ขนมปังเนย” วันที่ 28 พ.ค. 68
- ใน database มีการอัพเดทข้อมูลส่วนตัว และไม่ส่งผลกระทบกับของมูลส่วนอื่น ๆ

## ผลลัพธ์ที่ได้จริง :

- ผู้ใช้สามารถ Log in เข้าใช้งานได้จริง (login ด้วยเบอร์โทรศัพท์ 0927789568)  
ภาพของ user ทั้งหมดที่สามารถ login ได้

**Result Grid:**

ID	phone_number	password	fname	lname	zip_code	address	ig	line	is_store
37	0939761545	\$2b\$10\$BKMey\$fmwvKcaGm1t1eW4Scph...	Sorawit	B.	12120	Modis us	sorawit.s	only_...	-
38	2	\$2b\$10\$BKMey\$fmwvKcaGm1t1eW4Scph...	Tops	ก.	12120	สำนักงาน	TopsTU	tops...	-
39	5477777777	\$2b\$10\$BKMey\$fmwvKcaGm1t1eW4Scph...	นาย	วิจิตร	12120	ก.	TopTU	Karb...	-
40	0927789568	\$2b\$10\$BKMey\$fmwvKcaGm1t1eW4Scph...	นาย	วิจิตร	12120	TU Dome	TopTU	Karb...	-
41	000000	\$2b\$10\$BKMey\$fmwvKcaGm1t1eW4Scph...	Farmhouse	ก.	10510	ธรรมชาติสัมภាន	-	-	-
42	0612238933	\$2b\$10\$BKMey\$fmwvKcaGm1t1eW4Scph...	นาย	วิจิตร	12120	TU Dome	-	-	-

**Action Output:**

#	Time	Action	Message	Duration / Fetch
41	10:13:39	SELECT COUNT(*) AS count FROM fridge WHERE owner = 42 and is_store in (0,1) LIMIT 0, 1000	1 row(s) returned	0.265 sec / 0.000 sec
42	10:32:25	SELECT * FROM members LIMIT 0, 1000	6 row(s) returned	0.451 sec / 0.000 sec
43	10:43:39	SELECT * FROM fridge LIMIT 0, 1000	13 row(s) returned	0.281 sec / 0.000 sec
44	22:37:24	SELECT * FROM members LIMIT 0, 1000	11 row(s) returned	0.257 sec / 0.000 sec
45	19:38:50	SELECT * FROM fridge LIMIT 0, 1000	57 row(s) returned	0.281 sec / 0.000 sec
46	19:40:47	SELECT * FROM members LIMIT 0, 1000	11 row(s) returned	0.343 sec / 0.000 sec

- มีวัตถุดิบชื่อ “ขนมปังเนย” วันที่ 28 พ.ค. 2568 ในฐานข้อมูลอยู่จริง

**Result Grid:**

ID	material	exp	owner	image	type	price	is_store
169	ขนมปัง 20 ...	2025-05-30	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	3	500	1
170	เกลือหินอ่อน	2025-05-28	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	4	500	1
171	ผักกาดหอมสด	2025-05-31	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	3	500	1
172	ขนมปัง 20 ...	2025-05-29	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	3	500	3
173	ขนมปัง	2025-05-28	42	https://mai-bood.s3.ap-southeast-1.amazonaws...	2	500	2
...	...	...	...	...	...	...	...
51	...	...	...	...	...	...	...

**Action Output:**

#	Time	Action	Message	Duration / Fetch
46	19:40:47	SELECT * FROM members LIMIT 0, 1000	11 row(s) returned	0.343 sec / 0.000 sec
47	19:42:52	SELECT * FROM members LIMIT 0, 1000	11 row(s) returned	0.313 sec / 0.000 sec
48	19:43:59	SELECT * FROM fridge LIMIT 0, 1000	57 row(s) returned	0.359 sec / 0.000 sec
49	19:46:23	SELECT * FROM members LIMIT 0, 1000	11 row(s) returned	0.311 sec / 0.000 sec
50	19:48:46	SELECT * FROM fridge LIMIT 0, 1000	58 row(s) returned	0.332 sec / 0.000 sec
51	19:49:15	SELECT * FROM members LIMIT 0, 1000	58 row(s) returned	0.328 sec / 0.000 sec

3. ผู้ใช้สามารถแก้ไขข้อมูลส่วนตัวได้จริง  
 (ผู้ใช้ทำการเปลี่ยนชื่อจาก อภินันท์ สีบุญ เป็น Aphinut Suebboon)

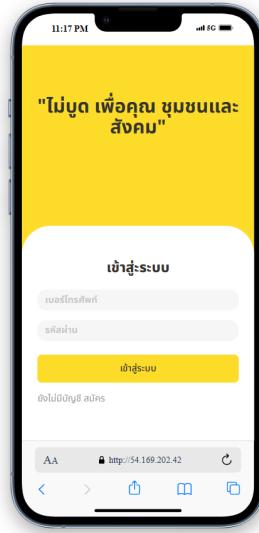
ID	phone_number	password	frame	name	zip_code	address	id	line
37	0939761545	\$2b\$10\$9MMy@bfmw..VcazCemRt1m40Scphv...	baf	B	12120	Modis se	soawiss.s	only_J
38	2	\$2b\$10\$9nTQf6R4ifZP2fV0fP6OgGmsa...	Tops	us จิ๊ฟ	12120	อะพาร์ทเม้น	TopTU	topter
39	0647777777	\$2b\$10\$9habJdQ5QsUA.a8f9MOY9EVw3o...	Kim	Kim	12120	B7 103	right_with_firework	Rainbow
40	0927799568	\$2b\$10\$9R4tV259Pj3/CisksXcp..guIsSwb7d4l...	Aphinut	Suebboon	12120	TU Done		
41	000000	\$2b\$10\$7A432HeXkPH7caRYTQAOCVcV9tTD...	Farmhouse	农舍	10510	โรงแรมฟาร์ม	-	-
42	0612238933	\$2b\$10\$4LRu7osgh1WxtRWjH0LMnT84nE2...	เขียวชา	ชาโยก้า	12120	TU Done		

Action Output:

#	Time	Action	Message	Duration / Fetch
48	19:43:59	SELECT * FROM fridge LIMIT 0, 1000	57 row(s) returned	0.359 sec / 0.000 sec
49	19:46:23	SELECT * FROM members LIMIT 0, 1000	11 row(s) returned	0.311 sec / 0.000 sec
50	19:48:46	SELECT * FROM fridge LIMIT 0, 1000	58 row(s) returned	0.320 sec / 0.000 sec
51	19:49:15	SELECT * FROM fridge LIMIT 0, 1000	58 row(s) returned	0.320 sec / 0.000 sec
52	19:50:05	SELECT * FROM fridge LIMIT 0, 1000	59 row(s) returned	0.297 sec / 0.000 sec
53	20:47:04	SELECT * FROM members LIMIT 0, 1000	11 row(s) returned	0.281 sec / 0.000 sec

4. ผู้ใช้สามารถ Log out ออกจากระบบได้จริง

(ระบบทำการกลับมาหน้า login)



ข้อสรุป :

ระบบทำงานได้อย่างปกติทั้งผึ้งของหน้าต่างการใช้งานของผู้ใช้ และผึ้งระบบ

## สถานการณ์ที่ 2

ทดสอบการแจ้งเตือนอาหารใกล้หมดอายุผ่านแอปพลิเคชัน Line

### ขั้นตอนการทดสอบ :

- ผู้ใช้ทำการเพิ่มเพื่อนแซทบอทด้วย LINE ID: @750oenhz
- ผู้ใช้ทำการพิมพ์ “ล็อกอิน”
- ผู้ใช้กรอก ID และรหัสผ่านเพื่อ Login
- ผู้ใช้เพิ่มรายการอาหารเข้าไปในตู้เย็นกำหนดวันที่หมดอายุเป็นวันที่ 23 พฤษภาคม 2025
- เมื่อถึงวันที่ 21 พฤษภาคม 2025 จะมีข้อความแจ้งเตือนอาหารใกล้หมดอายุพร้อมรายละเอียดของอาหารที่ใกล้หมดอายุ

### ผลลัพธ์ที่คาดหวัง :

ผู้ใช้งานมีเพื่อนใน Line ชื่อว่า Mai-Bood

ระบบมีการส่งแซทให้ผู้ใช้งานกรอก ID และรหัสผ่านส่วนตัว

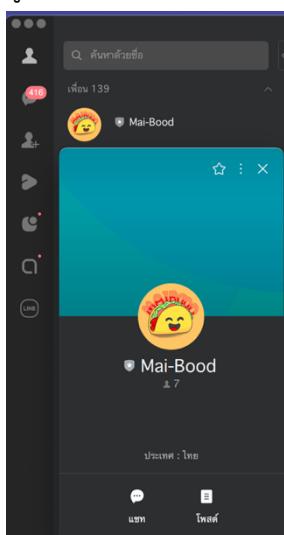
ระบบมีการแจ้งเตือนว่าล็อกอินสำเร็จ

ผู้ใช้งานสามารถเพิ่มอาหารเข้าไปในตู้เย็นได้

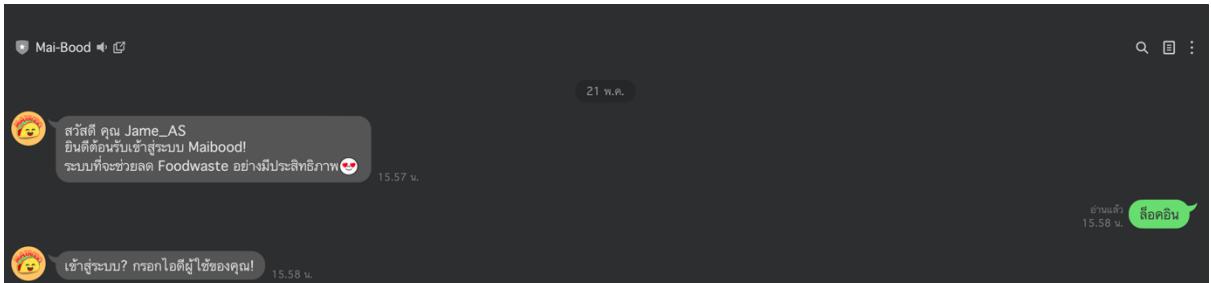
ระบบมีการส่งแจ้งเตือนว่าของใกล้หมดอายุ

### ผลลัพธ์ที่ได้จริง :

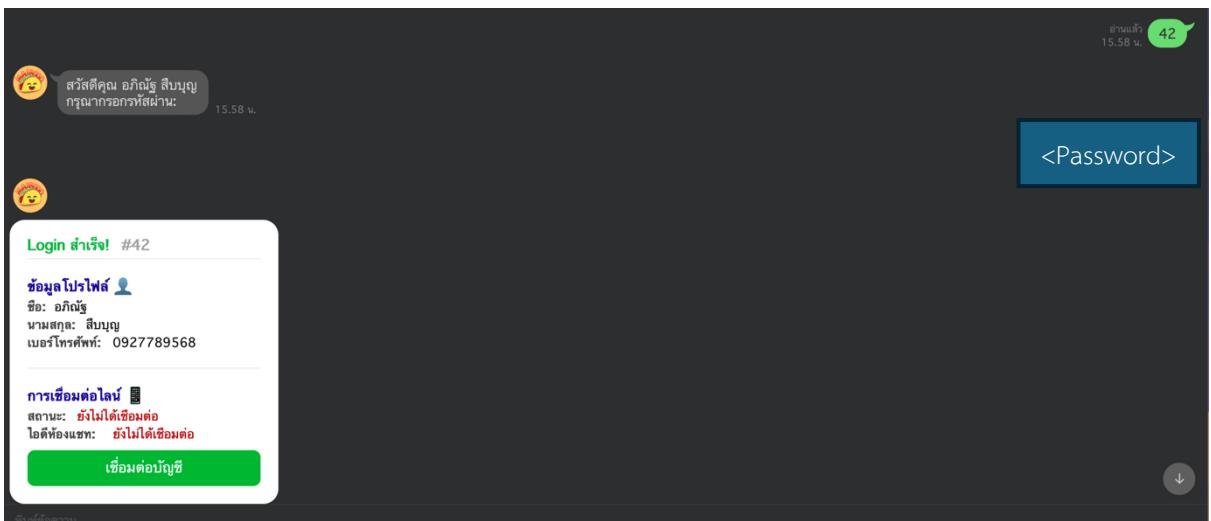
- ผู้ใช้งานมีเพื่อนใน Line ชื่อว่า Mai-Bood จริง



2. ระบบมีการส่งแจ้งให้ผู้ใช้งานกรอก ID และรหัสผ่านส่วนตัวเมื่อผู้ใช้งานคำว่า “ล็อกอิน”



3. ระบบมีการแจ้งเตือนว่าล็อกอินสำเร็จ



4. ในฐานข้อมูลมีข่องที่ทำการเพิ่มมาใหม่ตามที่ได้คาดหวัง ก่อนทำการเพิ่มของเข้าไป

ID	MATERIAL	EXP	OWNER	IMAGE	TYPE	PRICE	IS_STORE
169	ขบวนรำ 20...	2025-05-30	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	3	HULL	1
170	เกล็ดหูเสือ	2025-05-28	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	4	HULL	1
171	แมลงศีรษะสัก	2025-05-31	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	3	HULL	1
172	ขบวนรำ 20...	2025-05-29	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	3	HULL	3
173	ขบวนรำ	2025-05-28	42	HULL	HULL	HULL	2

## หลังทำการเพิ่มของเข้าไป

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the 'Query' tab is selected. Below it, the 'fridge' table is chosen from the 'members' schema. The main area displays the results of the query:

```
1 • SELECT * FROM fridge;
```

	id	material	exp	owner	image	type	price	is_store
170	เกลือห้องน้ำ	2025-05-28	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	4	HOLE	1	
171	แม่น้ำตุบูลส์	2025-05-31	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	3	HOLE	1	
172	ขบวนร้าน 20 ...	2025-05-29	41	https://mai-bood.s3.ap-southeast-1.amazonaws...	3	HOLE	3	
173	ขบวนร้าน	2025-05-28	42	HOLE	HOLE	HOLE	2	
174	ขบวนร้าน	2025-05-29	42	HOLE	HOLE	HOLE	0	
	MNULL	HOLE	HOLE	HOLE	HOLE	HOLE	HOLE	HOLE

Below the results, the 'Output' pane shows the history of actions taken on the 'fridge' table:

#	Time	Action	Message	Duration / Fetch
47	19:42:52	SELECT * FROM members LIMIT 0, 1000	11 row(s) returned	0.313 sec / 0.000 sec
48	19:43:59	SELECT * FROM fridge LIMIT 0, 1000	57 row(s) returned	0.359 sec / 0.000 sec
49	19:46:23	SELECT * FROM members LIMIT 0, 1000	11 row(s) returned	0.313 sec / 0.000 sec
50	19:48:46	SELECT * FROM fridge LIMIT 0, 1000	58 row(s) returned	0.328 sec / 0.000 sec
51	19:49:15	SELECT * FROM fridge LIMIT 0, 1000	58 row(s) returned	0.328 sec / 0.000 sec
52	19:50:05	SELECT * FROM fridge LIMIT 0, 1000	59 row(s) returned	0.297 sec / 0.000 sec

## 5. ระบบในการส่งการแจ้งเตือนผ่าน Line ได้ตามที่คาดหวัง

▶ 2025-05-25T00:00:32.510Z	2025-05-25T00:00:32.510Z 4468325d-80ff-4177-a2ec-8310099891fc INFO	Push message success for U9354ecff90e07546c6f8810bce6f7bd3!
▶ 2025-05-25T00:00:32.585Z	2025-05-25T00:00:32.585Z 4468325d-80ff-4177-a2ec-8310099891fc INFO	Push message success for Ua587502b58c2062c869868d092c6fff71!
▶ 2025-05-25T00:00:32.769Z	2025-05-25T00:00:32.769Z 4468325d-80ff-4177-a2ec-8310099891fc INFO	Push message success for U9354ecff90e07546c6f8810bce6f7bd3!
▶ 2025-05-25T00:00:32.889Z	2025-05-25T00:00:32.889Z 4468325d-80ff-4177-a2ec-8310099891fc INFO	Push message success for Ua587502b58c2062c869868d092c6fff71!

ข้อสรุป :

ระบบทำงานได้อย่างปกติทั้งฝั่งของหน้าต่างการใช้งานของผู้ใช้ และฝั่งระบบ

### สถานการณ์ที่ 3

ทดสอบการค้นหาเมนูจากวัตถุดิบในตู้เย็น

ขั้นตอนการทดสอบ :

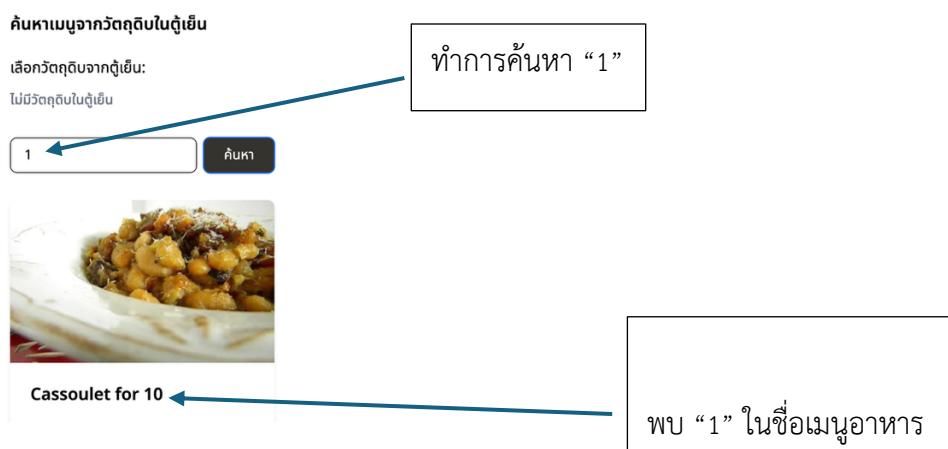
- ผู้ใช้งานทำการเข้าไปที่หน้าของตู้เย็น และทำการกดที่ข้อความ “ไม่รู้จะทำไรกิน กดตรงนี้”  
ผู้ใช้งานทำการค้นหาวัตถุดิบจากสิ่งที่มีในตู้เย็น หรือทำการพิมพ์ในช่องค้นหา  
กดช่องค้นหา  
ผู้ใช้งานจะเห็นเมนูอาหารพร้อมวัตถุดิบ และวิธีทำ

ผลลัพธ์ที่คาดหวัง :

เมนูอาหารที่มีวัตถุดิบนั้นจะทำการแสดงผลออกมา

ผลลัพธ์ที่ได้จริง :

มีเมนูอาหารที่ทำการแสดงผลจริง แต่มีข้อผิดพลาด



ข้อผิดพลาดที่เกิดขึ้น :

เกิดจาก logic การค้นหาที่ทำการ search จากข้อความทั้งหมดในแต่ละเมนู ไม่ใช่จากวัตถุดิบนั้น ๆ

ระบบที่รับมือกับปัญหานี้ :

ทำการแก้ไข logic ให้ scope ลงไปเฉพาะในวัตถุดิบ และตัดส่วนของตัวเลข และ white space ออก

ข้อสรุป :

ระบบเกิดนี้เกิดข้อผิดพลาดขึ้นจาก logic ของการค้นหา ซึ่งสามารถแก้ไขได้ตามปกติ

# บทวิเคราะห์ข้อดีและข้อเสียของระบบโดยอิงตาม AWS Well-Architected Framework

## 1. ด้านการปฏิบัติงาน (Operational Excellence)

**วัตถุประสงค์:** ดำเนินงานระบบให้มีประสิทธิภาพ และสามารถปรับปรุงพัฒนาได้อย่างต่อเนื่อง

**หลักการ:** ทำการปรับปรุงระบบอย่างสม่ำเสมอ นำข้อผิดพลาดมาเรียนรู้และปรับปรุงระบบ

**ข้อดี:** การ deploy แบบย่ออย่างบ่อย ช่วยลดความเสี่ยงจากการเปลี่ยนแปลงขนาดใหญ่ การใช้ IoT ทำให้สามารถวิเคราะห์ปัญหาได้ดีขึ้น และระบบมีความน่าเชื่อถือจากการทดสอบก่อนการใช้งานจริง

**ข้อเสีย:** ถ้าใช้ผลจากการทดสอบมากเกินไปอาจทำให้ตรวจเจอข้อผิดพลาดที่ระเอียดอ่อนข้าวไป การปรับปรุงอย่างต่อเนื่องต้องอาศัยการร่วมมือของสมาชิก และต้องใช้เวลาในการเรียนรู้การตั้งค่า CloudFormation

## 2. ด้านความปลอดภัย (Security)

**วัตถุประสงค์:** เพื่อทำการปกป้องข้อมูลส่วนบุคคลจากผู้ใช้งาน รวมทั้งฐานข้อมูลของระบบ โดยที่ไม่ให้บุคคลที่ไม่มีสิทธิในการเข้าถึง เข้าถึงข้อมูลเป็นอันขาด

**หลักการ:** ระบบต้องมีการตรวจสอบตัวตน (Authentication) และกำหนดสิทธิ์การเข้าถึง เช่นบุคคล หรือบัญชีผู้ใช้งานนั้น ๆ

**ข้อดี:** ข้อมูลส่วนบุคคลของบัญชี และฐานข้อมูลมีความปลอดภัยยิ่งขึ้น และสามารถเข้าถึงจากแหล่งที่ไม่เหมาะสมได้ยากมากขึ้น

**ข้อเสีย:** จะเกิดความยุ่งยากในการระบุตัวตนของเจ้าของบัญชีเอง หรือในกรณีที่ผู้ใช้งานไม่ได้ทำการยืนยันตัวตนไว้ก่อนหน้า ส่งผลให้การเข้าสู่ระบบในครั้งถัดไปอาจทำให้ไม่สามารถเข้าสู่ระบบได้

## 3. ด้านความน่าเชื่อถือ (Reliability)

**วัตถุประสงค์:** เพื่อให้ระบบสามารถทำงานได้อย่างถูกต้องและเป็นไปอย่างสม่ำเสมอ แม้ว่าตัวเซิร์ฟเวอร์จะหยุดชะงักลงก็ตาม

**หลักการ:** ควรมีอย่างน้อย 2 Availability Zone เพื่อไว้สำหรับการกู้คืนข้อมูลเมื่อเกิดเหตุที่ทำให้เซิร์ฟเวอร์หยุดชะงัก

**ข้อดี:** ระบบและตัวเซิร์ฟเวอร์ยังสามารถทำงานได้อย่างปกติ แม้จะมีอาการหยุดชะงักก็ตาม

**ข้อเสีย:** ในกรณีที่ Availability Zone ที่เซิร์ฟเวอร์ใช้บริการอยู่นั้นเกิดความเสียหายพร้อมกัน ส่งผลให้ตัวเซิร์ฟเวอร์เกิดความเสียหายอย่างหนักและอาจไม่สามารถทำงานได้อย่างปกติ

#### 4. ด้านประสิทธิภาพการทำงาน (Performance Efficiency)

**วัตถุประสงค์:** เพื่อให้ระบบจัดการกับทรัพยากรด้านการประมวลผล เครือข่าย และพื้นที่จัดเก็บอย่างมีประสิทธิภาพสูงสุด

**หลักการ:** ใช้ serverless เพื่อให้สามารถจัดการทรัพยากรและทำให้ระบบมีประสิทธิภาพได้

**ข้อดี:** ลดภาระของ server ใช้ทรัพยากรเฉพาะเมื่อมีการใช้งานจริง ทำให้ไม่เปลืองทรัพยากร

**ข้อเสีย:** การออกแบบระบบให้มีประสิทธิภาพอาจต้องใช้ต้นทุนที่สูงและต้องอาศัยความรู้ในการเลือกใช้บริการต่าง ๆ

#### 5. ด้านค่าใช้จ่าย (Cost Optimization)

**วัตถุประสงค์:** ลดต้นทุนในการดำเนินการ และ ใช้ทรัพยากรเท่าที่จำเป็น

**หลักการ:** จัดการบริหารต้นทุนหลักเลี้ยงค่าใช้จ่ายที่ไม่จำเป็น

**ข้อดี:** ปิด EC2 เมื่อไม่ได้ใช้งาน ช่วยลดค่าใช้จ่ายโดยรวมของระบบ และใช้ S3 Lifecycle Policy ทำให้ข้อมูลที่ไม่ถูกเข้าถึงเป็นเวลานานถูกย้ายไปใน storage ที่มีต้นทุนต่ำกว่า

**ข้อเสีย:** จะเกิดปัญหาจากการปิด EC2 ที่ผิดเวลา และการย้ายข้อมูลไป S3 Glacier ทำให้การเข้าถึงข้อมูลช้าลง และจะมีค่าใช้จ่ายเพิ่มขึ้นถ้าต้องกู้คืนข้อมูลบ่อย ๆ

#### 6. ความยั่งยืน (Sustainability)

**วัตถุประสงค์:** ออกแบบระบบให้ส่งผลต่อสิ่งแวดล้อมน้อยที่สุด

**หลักการ:** ใช้ทรัพยากรอย่างคุ้มค่า

**ข้อดี:** การใช้ serverless และการปรับ workload ให้เหมาะสมกับช่วงเวลาที่จำเป็นต้องใช้จะเป็นการลดการใช้ทรัพยากรโดยไม่จำเป็นช่วยให้ระบบมีความยั่งยืน

**ข้อเสีย:** ต้องการวางแผนที่ซับซ้อนและการทำให้ระบบมีความยั่งยืนและลดการส่งผลกระทบต่อสิ่งแวดล้อมมีโอกาสทำให้ประสิทธิภาพของระบบลดลงและอาจใช้ค่าใช้จ่ายมากขึ้น

## ไฟล์อื่น ๆ ในโครงการ

GitHub: <https://github.com/sorawiss/MaiBood>

GitHub (Chatbot) : <https://github.com/Chisanucha-Thonsiri/LineChatbotWithAWS>

Demo video การทำงานของระบบ: [https://tuipied-my.sharepoint.com/personal/sorawit\\_boonn\\_dome\\_tu\\_ac\\_th/\\_layouts/15/stream.aspx?id=%2Fpersonal%2Fsorawit%5Fboonn%5Fdome%5Ftu%5Fac%5Fth%2FDocuments%2FCSTU%2FCS232%2FDemo%2DMAibood%2Emp4&nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAIoJPbmVEcmI2ZUZvckJ1c2luZXNzliwicmVmZXJyYWxBcHBObGF0Zm9ybSI6IlldlYilsInJLZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOjJNeUZpbGVzTGlua0NvcHkifX0&ga=1&referrer=StreamWebApp%2EWeb&referrerScenario=AddressBarCopied%2Eview%2E29765704%2D1e76%2D485f%2Da1cc%2D05f6ef0b4241](https://tuipied-my.sharepoint.com/personal/sorawit_boonn_dome_tu_ac_th/_layouts/15/stream.aspx?id=%2Fpersonal%2Fsorawit%5Fboonn%5Fdome%5Ftu%5Fac%5Fth%2FDocuments%2FCSTU%2FCS232%2FDemo%2DMAibood%2Emp4&nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAIoJPbmVEcmI2ZUZvckJ1c2luZXNzliwicmVmZXJyYWxBcHBObGF0Zm9ybSI6IlldlYilsInJLZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOjJNeUZpbGVzTGlua0NvcHkifX0&ga=1&referrer=StreamWebApp%2EWeb&referrerScenario=AddressBarCopied%2Eview%2E29765704%2D1e76%2D485f%2Da1cc%2D05f6ef0b4241)

Trello:

<https://trello.com/invite/b/6826e613cef68ae22cf3b907/ATTlb8b97a6fa5c641996da89aa3dd685c9d3076D414/cs232>

## ผลการดำเนินงาน และแนวทางพัฒนาต่อในอนาคต

จากการดำเนินงานพบว่า เว็บไซต์ และตัวบอท ที่มีฟังก์ชันสำหรับการตรวจสอบอาหาร/ วัตถุดิบที่ผู้ใช้ทำการเพิ่มเข้าไป พร้อมทำการแจ้งเตือนเมื่ออาหารใกล้หมดอายุภายใน 3 วันและทำการแนะนำอาหารจากวัตถุดิบที่มีอยู่นั้น ทำงานได้อย่างมีประสิทธิภาพ อีกทั้งระบบจะแจ้งจ่ายอาหาร/ วัตถุดิบให้กับผู้ใช้งานในบริเวณใกล้เคียงก์สามารถทำงานได้ดีทั้งในสภาวะที่ปกติ หรือเมื่อปัญหาเล็กน้อย

สำหรับแนวทางที่จะพัฒนาต่อในอนาคตจะเป็นในส่วนของระบบแนะนำเมนูอาหารที่มีหลากหลายมากขึ้น หรือการทำระบบเครดิตสำหรับการซื้อขาย และเปลี่ยน หรือเป็นการสะสม carbon credit เพื่อสำหรับเป็นช่องทางการหารายได้จากการลดปริมาณ food waste ให้กับตลาดซื้อขาย และเพื่อความยั่งยืนในอนาคต นอกจากนี้เรายังสามารถ Deploy เว็บแอพพลิเคชันเป็น LIFF APP หรือ MINI APP และใช้ระบบ Login เป็นของ Line โดยตรงเพื่อให้สามารถใช้งานได้สะดวกมากขึ้น