รายงานสหกิจศึกษาฉบับสมบูรณ์

ระบบแชทสดสำหรับลูกค้าของบอทไอโอ

BOTIO LIVE CHAT

ภาคภูมิ บุญส่ง

PAKPUM BUNSONG

หลักสูตรวิศวกรรมสารสนเทศ

ภาควิชาวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

วิทยาเขตชุมพรเขตรอุดมศักดิ์ จังหวัดชุมพร

ปีการศึกษา 2564

รายงานสหกิจศึกษาฉบับสมบูรณ์

ระบบแชทสดสำหรับลูกค้าของบอทไอโอ

BOTIO LIVE CHAT

ภาคภูมิ บุญส่ง

PAKPUM BUNSONG

หลักสูตรวิศวกรรมสารสนเทศ

ภาควิชาวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

วิทยาเขตชุมพรเขตรอุดมศักดิ์ จังหวัดชุมพร

ปีการศึกษา 2564

รายงานสหกิจศึกษาฉบับสมบูรณ์

ประจำปีการศึกษา 2564

โครงงาน	ระบบแชทสดสำหรับลูกค้าของบอทไอโอ
ผู้จัดทำ	นาย ภาคภูมิ บุญส่ง รหัสนักศึกษา 61515017
ปฏิบัติงาน	บริษัท บอทไอโอ จำกัด
ที่อยู่	32-33 หมู่ ถ.บางนา-ตราด ต.บางพลีใหญ่ อ.บางพลี จ.สมุทรปราการ 10540
พนักงานที่ปรึกษา	นายปัณย์ ศักดิ์ธนากูล ตำแหน่ง Chief Technology Officer

รัตติกร สมบัติแก้ว

..... อาจารย์ที่ปรึกษา

(ดร.รัตติกร สมบัติแก้ว)

หนังสือส่งรายงานสหกิจศึกษาฉบับสมบูรณ์

เรื่อง ขอส่งรายงานสหกิจศึกษาฉบับสมบูรณ์

เรียน อาจารย์ที่ปรึกษาสหกิจศึกษา สาขาวิชาวิศวกรรมสารสนเทศ

ตามที่กระผม นาย ภาคภูมิ บุญส่ง นักศึกษาสาขาวิศวกรรมสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง วิทยาเขตชุมพรเขตรอุดมศักดิ์ จังหวัดชุมพร ได้ปฏิบัติงาน สหกิจศึกษาระหว่างวันที่ 2 สิงหาคม พ.ศ. 2562 ถึงวันที่ 30 พฤศจิกายน พ.ศ. 2564 ในตำแหน่ง Back-end developer ณ บริษัท บอทไอโอ จำกัด และได้รับมอบหมายจากพี่เลี้ยงที่ปรึกษาสหกิจศึกษาให้ศึกษาและจัดทำระบบแชทสดสำหรับลูกค้าของบอทไอโอ

บัดนี้ การปฏิบัติงานสหกิจศึกษาได้เสร็จสิ้นลงแล้ว จึงใคร่ขอส่งรายงานสหกิจศึกษาฉบับสมบูรณ์ จำนวน 1 ไฟล์

จึงเรียนมาเพื่อพิจารณา

ขอแสดงความเคารพอย่างสูง

นายภาคภูมิ บุญส่ง

นักศึกษาสหกิจศึกษาหลักสูตรวิศวกรรมสารสนเทศ

ภาควิชาวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

วิทยาเขตชุมพรเขตรอุดมศักดิ์ จังหวัดชุมพร

ใบรับรองสหกิจศึกษา

.....

หัวข้อสหกิจ ระบบแชทสดสำหรับลูกค้าของบอทไอโอ

Co-operative Title Botio Live Chat

ชื่อนักศึกษา นาย ภาคภูมิ บุญส่ง **รหัสประจำตัว** 61515017

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมสารสนเทศ

อาจารย์ที่ปรึกษาสหกิจ ดร.รัตติกร สมบัติแก้ว

คณะกรรมการสอบปริญญานิพนธ์			**ลายมือชื่อ**
ผศ.ดร.รัฐพงษ์	สุวลักษณ์	ประธานกรรมการสอบ	
อาจารย์อรรถศาสตร์	นาคเทวัญ	กรรมการสอบ	
อาจารย์นภัสรพี	สิทธิวัจน์	กรรมการสอบ	
ดร.รัตติกร	สมบัติแก้ว	อาจารย์ที่ปรึกษา	

วัน/เดือน/ปี ที่สอบ 24 ธันวาคม 2564 เวลา 09.00 – 16.00

สถานที่สอบ ออนไลน์ด้วยโปรแกรม Microsoft Team

ภาควิชาวิศวกรรมศาสตร์รับรองแล้ว

(ผู้ช่วยศาสตราจารย์ ดร.ปราโมทย์ กุศล)

หัวหน้าภาควิชาวิศวกรรมศาสตร์

วันที่ 21 กรกฎาคม พ.ศ. 2565

หัวข้อสหกิจ ระบบแชทสดสำหรับลูกค้าของบอทไอโอ

นักศึกษาผู้จัดทำ นายภาคภูมิ บุญส่ง **รหัสนักศึกษา** 61515017

หลักสูตร วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมสารสนเทศ

อาจารย์ที่ปรึกษา ดร.รัตติกร สมบัติแก้ว

พนักงานที่ปรึกษา นายปัถย์ ศักดิ์ธนากูล

ปีการศึกษา 2564

บทคัดย่อ

โครงงานสหกิจศึกษาฉบับนี้แสดงข้อมูลเกี่ยวกับระบบการสนทนาแบบเรียลไทม์ สำหรับผู้ใช้บริการหรือเจ้าของ-เพจเฟสบุ๊ค (Facebook) ที่ใช้บริการของบอทไอโอโดยเป็นตัวกลางหรือเอพีไอในการเชื่อมต่อระหว่างฐานข้อมูลกับแอปพลิเคชันที่ใช้ในการแสดงผล ซึ่งช่วยในการจัดการระบบการสนทนา ได้แก่ ประวัติการสนทนา ข้อความล่าสุด รวมถึงการส่งข้อความ โดยผู้ใช้บริการสามารถเข้าถึงและดำเนินการการสนทนาผ่านแพลตฟอร์มของบอทไอโอในหัวข้อการสนทนา ขั้นต้นหากผู้ใช้บริการเลือกหัวข้อการสนทนา ระบบจะทำการเก็บข้อมูลสำคัญทั้งหมดลงในฐานข้อมูล ตัวอย่างเช่น วันเวลาที่ทำการเชื่อมต่อ หมายเลขของผู้ที่ทำการเชื่อมต่อ และหมายเลขเพจที่ทำการเชื่อมต่อ เป็นต้น จากนั้นผู้ใช้งานจะสามารถเลือกบุคคลที่จะทำการสนทนาด้วย รวมถึงประเภทข้อความที่จะส่งได้

คำสำคัญ: เพจ Facebook, เอพีไอ, แพลตฟอร์ม, ฐานข้อมูล

Co-operative Title	Botio Live Chat	
Student	Mr. Pakpum Bunsong	**Student ID** 61515017
Degree	Bachelor of Engineering	
Program in	Information Engineering	
Advisor	Dr. Rattikorn Sombutkaew	
Mentor	Mr. Putt Sakdhnakool	
Academic Year	2021	

ABTRACT

This bachelor dissertation provides information about the real-time conversation system. For customers or owners of Facebook pages that use the services of Botio. It is an intermediary or API that connects between databases with the application used to display that helps to manage with the conversation system. For example, conversation history, last message including sending messages. Users can access and operate conversations through the Botio platform in the conversation thread. Initially, if the user selects a conversation topic The system will store all important information in the database, for example datetime stamp, user id, and page id. After that, Users will then be able to choose who to chat with. Including the types of messages that can be sent.

Keywords: Facebook pages, API, Platform, Database

กิตติกรรมประกาศ

การที่ข้าพเจ้าได้มาปฏิบัติงานสหกิจศึกษา ณ บริษัท บอทไอโอ จำกัด ตั้งแต่วันที่ 2 สิงหาคม พ.ศ. 2564 ถึงวันที่ 30 พฤศจิกายน พ.ศ. 2564 ส่งผลให้ข้าพเจ้าได้รับความรู้และประสบการณ์ต่าง ๆ ที่มีคุณค่าและประโยชน์อย่างมาก สำหรับรายงานวิชาสหกิจฉบับนี้ สำเร็จลุล่วงด้วยดีจากความช่วยเหลือและการสนับสนุนเป็นอย่างดีจากหลายฝ่าย

ขอขอบพระคุณคุณพ่อและคุณแม่และญาติพี่น้อง ผู้ซึ่งคอยให้การอบรมสั่งสอน เลี้ยงดู สนับสนุนเงินทุนในการศึกษา ตลอดจนให้กำลังใจเสมอมา

ขอขอบพระคุณ ดร.รัตติกร สมบัติแก้ว อาจารย์ที่ปรึกษา ผู้ซึ่งให้คำแนะนำต่าง ๆ คำปรึกษาในการแก้ไขปัญหาที่เกิดขึ้นทั้งในการออกแบบ ปัญหาที่เกิดขึ้นในระหว่างการทำโครงงาน ตลอดจนการติดตามเกี่ยวกับงานโครงงานตลอดมา ผู้เขียนจึงขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณคณะอาจารย์ที่เคารพทุกท่าน ที่ให้ความเอาใจใส่แนะนำ คอยช่วยเหลือเสมอมา แม้ว่าจะไม่ใช่อาจารย์ที่ปรึกษาก็ตาม

ขอขอบพระคุณนายปัถย์ ศักดิ์ธนากูล ที่ปรึกษาและบริษัท บอทไอโอ จำกัด ที่ได้เอื้อเฟื้อให้นักศึกษาได้มีโอกาสทำโครงงานร่วมกับบริษัทตลอดจนสถานที่ใช้ในการทำงานและอุปกรณ์ที่ใช้ในการปฏิบัติงาน และการช่วยเหลือของพนักงานที่ปรึกษาในทุก ๆ ด้านที่มีให้เสมอมา

ขอขอบพระคุณนายเสฏฐวุฒิ ลีลาวัฒนพาณิชย์ และ Mr. Mithlesh Meghwal ที่ได้ให้คำแนะนำ และช่วยเหลือเกี่ยวกับการทำโครงงานในครั้งนี้

ขอขอบพระคุณทุก ๆ ท่านที่เกี่ยวข้องและแม้ไม่ได้กล่าวในนามนี้ก็ตาม ที่เป็นที่รักและให้กำลังใจเสมอมาโดยตลอด

ขอน้อมรำลึกถึงคุณของทุก ๆ ท่านตลอดไป และความรู้ที่ได้จากการทำชิ้นงานในครั้งนี้ ผู้เขียนจะใช้ให้เป็นประโยชน์สูงสุด รวมถึงแบ่งปันให้กับผู้ที่สนใจต่อไป

ภาคภูมิ บุญส่ง

ธันวาคม 2564

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

ในบทนี้จะกล่าวถึง ความเป็นมาและความสำคัญ วัตถุประสงค์ของการทำโครงงาน ขอบเขตของการทำโครงงาน ประโยชน์ที่คาดว่าจะได้รับ แผนการดำเนินงาน และโครงสร้างของรายงานสหกิจศึกษา

1.1 ความเป็นมาและความสำคัญ

ในปัจจุบันบริษัทบอทไอโอ จำกัด เป็นบริษัท Startup ที่ทำธุรกิจเกี่ยวกับ Social commerce ที่ให้บริการทางด้านการบริหารจัดการร้านค้าออนไลน์บน Facebook ให้มีความสะดวกสบายมากยิ่งขึ้น โดยลูกค้าที่ใช้บริการของบริษัทสามารถใช้งานได้ผ่านเว็บไซต์ของบอทไอโอ เช่น การเพิ่มข้อมูลร้านค้า การปรับแต่งหน้าร้านค้าออนไลน์ การจัดการกับคลังสินค้า เป็นต้น แต่ในกรณีของการตอบแชทผู้ใช้บริการจะต้องใช้งานผ่านเว็บไซต์ของ Facebook ด้วยเหตุนี้ จึงมีการเพิ่มระบบสำหรับการตอบแชทให้สามารถใช้งานผ่านเว็บไซต์ของบอทไอโอได้

1.2 วัตถุประสงค์ของการทำโครงงาน

1. เพื่อศึกษาการเขียนโปรแกรมภาษา Golang
2. เพื่อศึกษาการใช้งาน Amazon Web Service
3. เพื่อศึกษาการใช้งาน Apache Kafka
4. เพื่อศึกษาการใช้งาน Facebook API
5. เพื่อพัฒนาระบบ Back-end สำหรับรองรับการสนทนาแบบ Real-time

1.3 ขอบเขตของการทำโครงงาน

1. สร้างระบบ Back-end สำหรับรองรับการสนทนาแบบ Real-time
2. สามารถดึงประวัติการสนทนาได้
3. สามารถเชื่อมต่อกับ Facebook API ได้
4. สามารถเชื่อมต่อกับ Apache Kafka ได้
5. สามารถบันทึก Template ข้อความเพื่อใช้สำหรับการส่งข้อความได้
6. สร้างระบบสำหรับเชื่อมต่อและส่งข้อความผ่าน WebSocket protocol
7. สามารถเชื่อมต่อเข้ากับ Amazon API Gateway ได้
8. สามารถเก็บประวัติการเชื่อมต่อของ WebSocket ได้
9. สร้างระบบสำหรับยืนยันตัวตนโดยใช้ JSON Web Token

1.3 ขอบเขตของการทำโครงงาน (ต่อ)

10. สามารถรองรับการใช้งานแบบ Multi tab ได้
11. สามารถรองรับการใช้งานร่วมกับ Instagram ได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. พัฒนาทักษะการเขียนโปรแกรม
2. ได้โอกาสในการเรียนรู้ ศึกษาหาประสบการณ์ใหม่เพิ่มเติมจากงานที่ได้รับมอบหมาย และนำความรู้ความสามารถมาประยุกต์ใช้กับงานได้จริง
3. มีมนุษย์สัมพันธ์กับบุคคลทั่วไปภายในบริษัท เรียนรู้ชีวิตในวัยทำงาน รวมทั้งปรับตัวและทำความเข้าใจระบบการทำงานในรูปแบบบริษัท
4. ได้พัฒนาทักษะการวางแผน รูปแบบการทำงาน และมีความรับผิดชอบในการทำงานมากขึ้น
5. ได้เรียนรู้ข้อผิดพลาดในการทำงานจริงของตนเอง และนำมาปรับปรุงเพื่อพัฒนาตนเอง
6. ระบบแชทสด (Live Chat) ที่ออกแบบและจัดทำสามารถนำไปใช้งานได้จริง

1.5 แผนการดำเนินงาน

ขั้นตอนการดำเนินงานที่ผู้จัดทำได้วางแผนและฝึกสหกิจศึกษาแสดงรายละเอียดไว้ในตารางดังนี้

ตารางที่ 1.1 ขั้นตอนและวิธีการดำเนินงาน

ขั้นตอนการดำเนินงาน	ระยะเวลาการดำเนินงาน															
	สิงหาคม				กันยายน				ตุลาคม				พฤศจิกายน			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. ศึกษาการใช้งาน Facebook Graph API	↔															
2. ศึกษาการใช้งาน Amazon Web Service		↔														
3. ออกแบบและพัฒนา Live Chat API			⟷													
4. ออกแบบและพัฒนา WebSocket					⟷											

ตารางที่ 1.1 ขั้นตอนและวิธีการดำเนินงาน (ต่อ)

ขั้นตอนการดำเนินงาน	ระยะเวลาการดำเนินงาน															
	สิงหาคม				กันยายน				ตุลาคม				พฤศจิกายน			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
5. ศึกษาการใช้งาน Docker							↔									
6. ศึกษาการใช้งาน Webhook								↔								
7. ออกแบบและพัฒนา Live Chat API สำหรับ Instagram									←	—	→					
8. ออกแบบและพัฒนา Tunnel สำหรับ Instagram												← — →				
9. ทดสอบและปรับปรุง API															← — →	

1.6 โครงสร้างของรายงานสหกิจศึกษา

ในรายงานสหกิจศึกษาฉบับนี้แบ่งออกเป็น 5 ส่วนหลัก ๆ คือ บทที่ 1-5 ซึ่งแต่ละบทจะอธิบายเนื้อหาที่เกี่ยวข้องกับการออกแบบระบบแชทสดสำหรับลูกค้าของบอทไอโอ โดยแบ่งออกเป็นบทต่าง ๆ มีรายละเอียด ดังต่อไปนี้

บทที่ 1 บทนำ ในบทนี้กล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของการทำโครงงาน ขอบเขตของการทำโครงงาน ประโยชน์ที่ได้รับ ขั้นตอนและวิธีการดำเนินงาน และโครงสร้างของรายงานสหกิจศึกษา

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง ในบทนี้กล่าวถึง ทฤษฎีที่เกี่ยวข้องกับระบบ Live Chat ได้แก่ เว็บแอปพลิเคชัน (Web Application) Application Programming Interface (API) โปรแกรม Visual Studio Code โปรแกรมฐานข้อมูล MongoDB โปรแกรม Postman และการใช้โค้ดในการเขียนโปรแกรมภาษา Golang

บทที่ 3 หลักการและการออกแบบ ในบทนี้จะกล่าวถึงขั้นตอนการดำเนินการออกแบบระบบแชทสดสำหรับลูกค้าของบอทไอโอประกอบด้วย ภาพรวมของระบบแชทสด ระบบการสนทนาของระบบแชทสด การยืนยันตัวตน ฐานข้อมูล และเอกสารอธิบาย API

บทที่ 4 การทดลองและผลการทดลอง ในบทนี้จะกล่าวถึงการทดลอง คือ การทดลองการทำงานของระบบหลังบ้าน (Back-end) การเชื่อมต่อเว็บซ็อกเก็ต (WebSocket) การส่งข้อความผ่านเว็บซ็อกเก็ต (WebSocket) การใช้งานเว็บฮุค (Webhook) และการเรียกใช้งาน API

บทที่ 5 สรุปผลการทดลองและข้อเสนอแนะ ในบทนี้กล่าวถึงการสรุปผลการทดลอง ปัญหาและอุปสรรคในการทำงาน และข้อเสนอแนะต่าง ๆ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในบทนี้กล่าวถึง ทฤษฎีที่เกี่ยวข้องกับระบบแชทสด (Live Chat) ได้แก่ เว็บแอปพลิเคชัน (Web Application) Application Programming Interface (API) โปรแกรม Visual Studio Code โปรแกรมฐานข้อมูล MongoDB โปรแกรม Postman การใช้โค้ดในการเขียนโปรแกรมภาษา Golang

2.1 เว็บแอปพลิเคชัน (Web Application)

เว็บแอปพลิเคชัน (Web Application) [1] เป็นแอปพลิเคชันที่ถูกสร้างขึ้นมาเพื่อเป็นเบราว์เซอร์ให้เหมาะสมสำหรับการใช้งานเว็บเพจต่าง ๆ ซึ่งถูกปรับเปลี่ยนให้แสดงผลเฉพาะส่วนที่จำเป็น เพื่อช่วยลดการประมวลผลของตัวเครื่องสมาร์ทโฟน หรือแท็บเล็ต ทำให้สามารถโหลดหน้าเว็บไซต์ได้เร็วยิ่งขึ้น อีกทั้งผู้ใช้งานยังสามารถเข้าใช้งานผ่านอินเทอร์เน็ตหรืออินทราเน็ตในความเร็วต่ำได้ หากนักพัฒนาได้เขียนเว็บแอปพลิเคชันตาม MVC (Model View Controller) แล้วสามารถแบ่งเว็บแอปพลิเคชันออกได้เป็น 3 ส่วนหลัก ๆ ได้ดังนี้ คือ

1. ส่วนที่ติดต่อกับผู้ใช้งานเพื่อรับข้อมูลและแสดงผล (View)
2. ส่วนที่ประมวลผลการทำงาน (Controller)
3. ส่วนที่ใช้ในการติดต่อและจัดการกับข้อมูลและฐานข้อมูล (Model)

ภาษาที่ใช้ในการพัฒนาเว็บแอปพลิเคชันแบ่งออกได้เป็น 2 ส่วนคือ Front-End Technology ที่ใช้สำหรับพัฒนาส่วนของการติดต่อกับผู้ใช้งาน และ Back-End Technology ใช้สำหรับพัฒนาส่วนของการประมวลผลข้อมูลและการจัดการข้อมูล

2.2 Application Programming Interface (API)

Application Programming Interface (API) [2] คือ ช่องทางการเชื่อมต่อช่องทางหนึ่ง ที่จะเชื่อมต่อกับเว็บไซต์ผู้ให้บริการ API จากที่อื่น เป็นตัวกลางที่ทำให้โปรแกรมประยุกต์เชื่อมต่อกับโปรแกรมประยุกต์อื่น หรือเชื่อมการทำงานเข้ากับระบบปฏิบัติการ กล่าวคือ API เป็นตัวกลางที่จะทำหน้าที่คอยรับคำสั่งต่าง ๆ ประมวลผล และทำการส่งคืนข้อมูลกลับไปยังผู้เรียกใช้งานโดยอัตโนมัติ

RESTful API คือ API รูปแบบหนึ่งที่ใช้เป็นอินเทอร์เฟซ (Interface) ที่ระบบคอมพิวเตอร์สองระบบใช้เพื่อแลกเปลี่ยนข้อมูลผ่านอินเทอร์เน็ตได้อย่างปลอดภัย แอปพลิเคชันทางธุรกิจส่วนใหญ่ต้องสื่อสารกับแอปพลิเคชันภายในอื่นๆ และของบุคคลที่สามเพื่อทำงานต่างๆ ตัวอย่างเช่น หากต้องการสร้างสลิปเงินเดือน ระบบบัญชีภายในของคุณต้องแบ่งปันข้อมูลกับระบบธนาคารของลูกค้าเพื่อออกใบแจ้งหนี้และสื่อสารกับแอปพลิเคชันบันทึกเวลาปฏิบัติงานภายในโดยอัตโนมัติ โดย

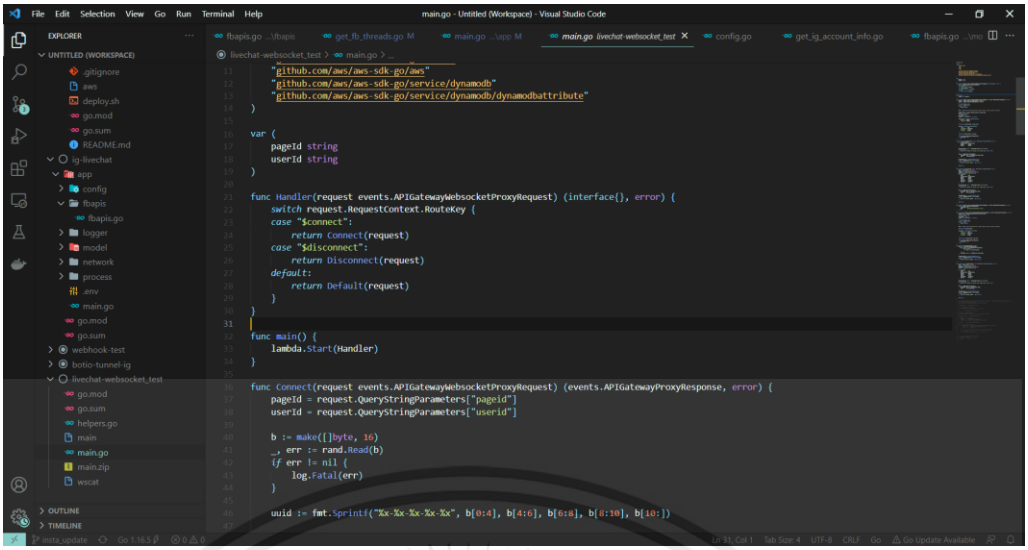
RESTful API รองรับการแลกเปลี่ยนข้อมูลนี้ เนื่องจากเป็นไปตามมาตรฐานการสื่อสารระหว่างซอฟต์แวร์ที่ปลอดภัย เชื่อถือได้ และมีประสิทธิภาพ

2.3 โปรแกรม Visual Studio Code

รูปที่ 2.1 สัญลักษณ์ของโปรแกรม Visual Studio Code
(**ที่มา**: https://icon-icons.com/icon/file-type-vscode/130084)

Visual Studio Code หรือ VSCode [3] สัญลักษณ์โปรแกรมดังแสดงในรูปที่ 2.1 เป็นโปรแกรมที่ใช้ในการแก้ไขและปรับแต่งโค้ดจากค่ายไมโครซอฟท์ มีการพัฒนาออกมาในรูปแบบของ Open Source จึงสามารถนำมาใช้งานได้แบบฟรี ๆ ที่ต้องการความเป็นมืออาชีพ

Visual Studio Code นั้นเหมาะสำหรับนักพัฒนาโปรแกรมที่ต้องการใช้งานข้ามแพลตฟอร์ม รองรับการใช้งานทั้งบน Windows, macOS และ Linux สนับสนุนทั้งภาษา JavaScript, TypeScript และ Node.js สามารถเชื่อมต่อกับ Git ได้ นำมาใช้งานได้ง่ายไม่ซับซ้อนมีเครื่องมือส่วนขยายต่าง ๆ ให้เลือกใช้อย่างมากมายไม่ว่าจะเป็นการเปิดใช้งานภาษาอื่น ๆ ทั้ง ภาษา C++, C#, JAVA, PYTHON และ GOLANG เป็นต้น ตัวอย่างการใช้งาน Visual Studio Code แสดงในรูปที่ 2.2

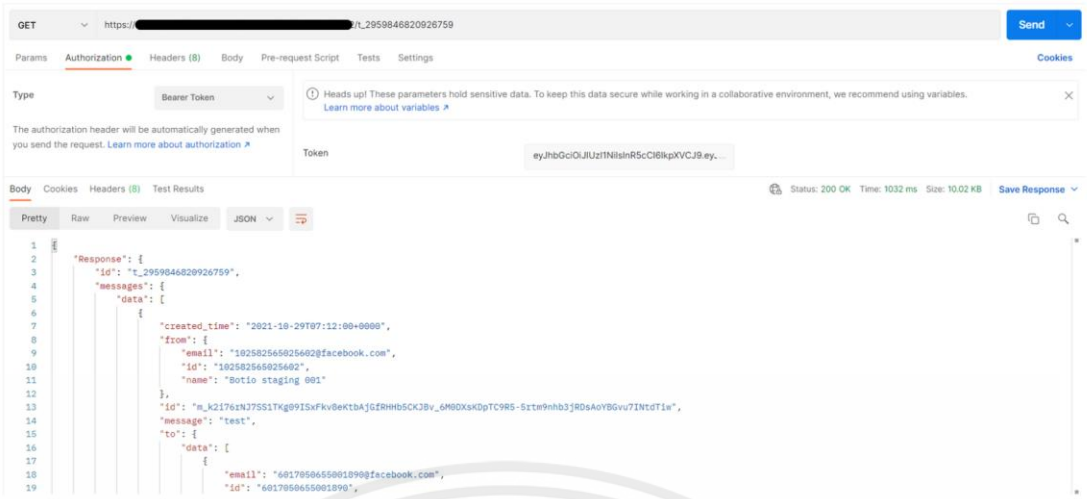


รูปที่ 2.2 ตัวอย่างการใช้งาน Visual Studio Code

2.4 โปรแกรม Postman

รูปที่ 2.3 สัญลักษณ์ของโปรแกรม Postman
(**ที่มา**: https://blog.qualys.com/product-tech/2019/10/07/enhanced-api-scanning-with-postman-support-in-qualys-was)

Postman [4] สัญลักษณ์โปรแกรมดังแสดงในรูปที่ 2.3 คือ แอปพลิเคชันที่ช่วยสำหรับการพัฒนาและทดสอบ API service โดย Postman สามารถจำลองข้อมูล รวมถึงบันทึกข้อมูลที่ใช้ในการทดสอบเก็บไว้ใช้งานต่อได้ ตัวอย่างแสดงดังรูปที่ 2.4



รูปที่ 2.4 ตัวอย่างการใช้งาน Postman

2.5 ภาษาโกแลง (Golang)

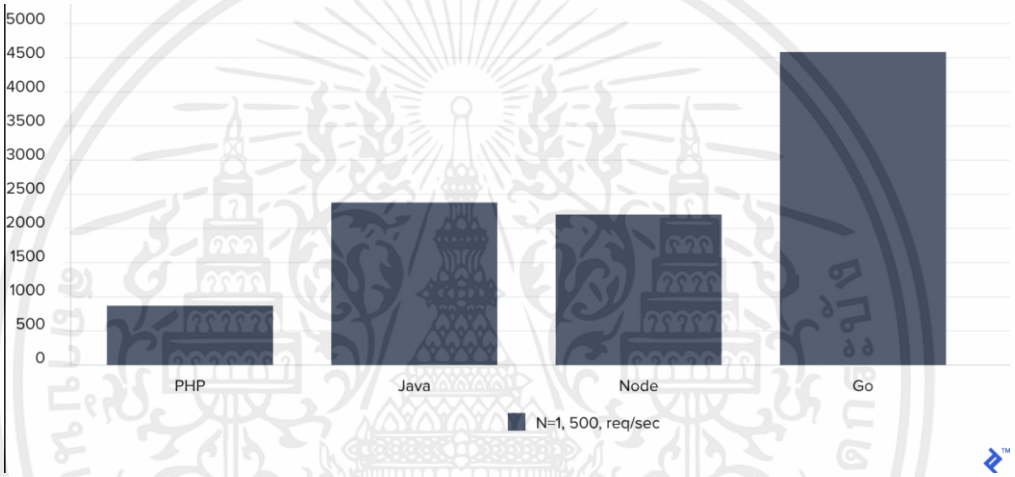
รูปที่ 2.5 สัญลักษณ์ของโปรแกรมภาษา Golang

(**ที่มา**: https://betterprogramming.pub/i-mastered-golang-basics-by-solving-these-15-project-euler-problems-1254a3897cf8)

ภาษาโกแลง (Golang) [5] สัญลักษณ์โปรแกรมดังแสดงในรูปที่ 2.5 คือ ภาษาโปรแกรมมิ่งที่ถูกสร้างโดยบริษัท Google โดยมุ่งเน้นไปที่การเขียนโปรแกรมในลักษณะ System programming และการเขียนโปรแกรมในฝั่ง Back-end ตัวอย่างเช่น การสร้าง API server หรือ Network application เป็นต้น

จุดเด่นของภาษาโกแลง

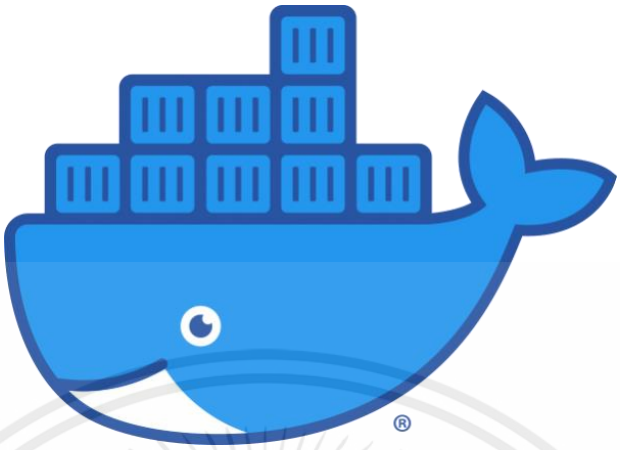
1. Compiler Language โกแลง (Golang) เป็นภาษาแบบคอมไพเลอร์ คือ มีการแปล Source code ทั้งหมดให้กลายเป็น Executable file (ภาษาเครื่อง) ซึ่งคอมพิวเตอร์สามารถนำไปใช้งานได้ทันที โดยไม่ต้องแปลคำสั่งใหม่ทุกรอบ ทำให้โปรแกรมสามารถทำงานได้รวดเร็วขึ้น
2. Static type โกแลง (Golang) เป็นภาษาแบบ Static type คือ ตัวแปรต้องกำหนดชนิดตั้งแต่แรก ทำให้คอมไพเลอร์ช่วยตรวจสอบข้อผิดพลาดเบื้องต้นได้ตั้งแต่เริ่มเขียนโปรแกรม
3. ใช้เวลาคอมไพล์น้อย
4. ออกแบบมาสำหรับการเขียนโปรแกรมแบบ Parallel โดยเฉพาะ
5. เหมาะสำหรับงานที่ต้องการรองรับ Request เป็นจำนวนมาก ตัวอย่างแสดงดังรูปที่ 2.6



รูปที่ 2.6 ตัวอย่างทดสอบการรองรับ Request ของภาษาคอมพิวเตอร์

(**ที่มา**: https://www.toptal.com/back-end/server-side-io-performance-node-php-java-go)

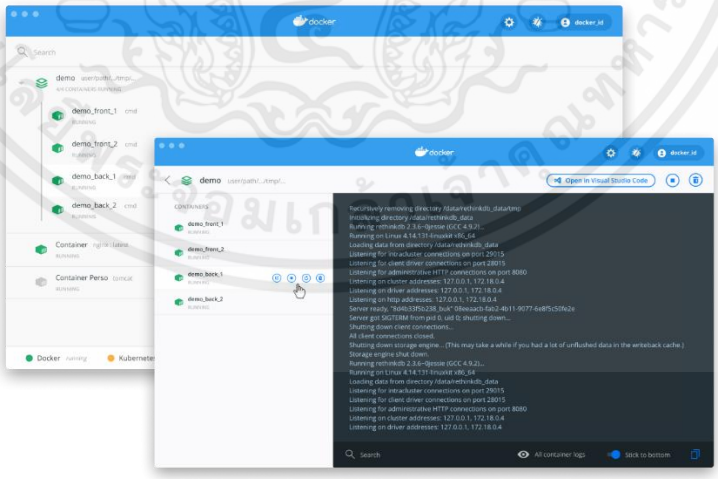
2.6 โปรแกรม Docker



รูปที่ 2.7 สัญลักษณ์ของโปรแกรม Docker

(**ที่มา**: https://www.docker.com/company/newsroom/media-resources)

Docker [6] สัญลักษณ์โปรแกรมดังแสดงในรูปที่ 2.7 คือแพลตฟอร์มซอฟต์แวร์ที่ช่วยในการสร้าง ทดสอบ และติดตั้งแอปพลิเคชันใช้จริงได้อย่างรวดเร็ว โดย Docker จะบรรจุซอฟต์แวร์ลงไปในหน่วยที่เป็นมาตรฐานเรียกว่า คอนเทนเนอร์ ซึ่งจะมีสิ่งที่ซอฟต์แวร์ต้องใช้ในการเรียกใช้ รวมทั้งไลบรารี เครื่องมือสำหรับระบบ โค้ด และรันไทม์ เมื่อใช้งาน Docker จะสามารถติดตั้ง ใช้งาน และปรับขนาดแอปพลิเคชันให้เหมาะกับทุกสภาพแวดล้อมได้ ตัวอย่างแสดงดังรูปที่ 2.8



รูปที่ 2.8 ตัวอย่างการใช้งาน Docker

(**ที่มา**: https://www.docker.com/products/docker-desktop)

2.7 Kubernetes

รูปที่ 2.9 สัญลักษณ์ของโปรแกรม Kubernetes
(**ที่มา**: https://uxwing.com/kubernetes-icon/)

คูเบอร์เนเตส (Kubernetes) หรือ K8s [7] สัญลักษณ์โปรแกรมดังแสดงในรูปที่ 2.9 คือ แพลตฟอร์มแบบโอเพ่นซอร์สที่ถูกพัฒนาโดยบริษัท Google สำหรับช่วยให้การปฏิบัติงานต่าง ๆ ที่เกี่ยวข้องกับ Linux container สามารถทำได้โดยอัตโนมัติ ลดกระบวนการติดตั้งหรือขยายแอปพลิเคชันที่รันบน Container ที่นักพัฒนาต้องลงมือทำด้วยตนเองให้เหลือน้อยที่สุด หรือกล่าวได้ว่าช่วยให้นักพัฒนาสามารถทำคลัสเตอร์กลุ่มของโฮสต์ที่รัน Linux Container ได้ และ Kubernetes เข้ามาช่วยบริหารจัดการคลัสเตอร์เหล่านั้นสำหรับใช้งานบน Public, Private และ Hybrid Cloud ได้อย่างสะดวกรวดเร็วและมีประสิทธิภาพ โดยมี

คุณสมบัติเด่นของ Kubernetes ดังนี้

1. ประสานการทำงานของ Containers ระหว่างแต่ละโฮสต์เข้าด้วยกัน
2. ช่วยให้ใช้ทรัพยากรของฮาร์ดแวร์ในการรันแอปพลิเคชันได้อย่างมีประสิทธิภาพสูงสุด
3. ควบคุมการติดตั้งและอัปเดตแอปพลิเคชันโดยอัตโนมัติ
4. ขยายแอปพลิเคชันที่รันบน Container และทรัพยากรต่างๆ ที่จำเป็นต้องใช้งานได้ตามต้องการ
5. เพิ่ม Storage สำหรับรองรับการรันแอปพลิเคชันแบบ Stateful ได้อย่างง่ายดาย
6. ช่วยการันตีว่าแอปพลิเคชันสามารถทำงานได้ตรงตามที่กำหนดไว้
7. มีระบบ Health-check และ Self-heal รวมไปถึง Auto replacement, Auto restart, Auto replication และ Auto scaling

2.8 บริการ Amazon Web Service (AWS)

Amazon Web Service (AWS) [8] คือ บริการในเครือของ Amazon ที่ให้บริการในด้าน Cloud computing ในรูปแบบ Infrastructure as a Service (IaaS) ต่าง ๆ เช่น เซิร์ฟเวอร์ เน็ตเวิร์ก ฯลฯ เพื่อใช้ในการประมวลผลใด ๆ เช่น เว็บไซต์ แอปพลิเคชันบนมือถือ ตลอดจนถึง Machine Learning โดยมี

จุดเด่นดังนี้

1. การรวบรวมระบบต่าง ๆ เข้าด้วยกันได้ง่าย
2. ค่าใช้จ่ายขึ้นอยู่กับการใช้งาน
3. ระบบรักษาความปลอดภัยที่ได้มาตรฐาน
4. สามารถขยับขยายระบบตามขนาดขององค์กรได้

2.9 บริการ Amazon Elastic Kubernetes Service (Amazon EKS)

รูปที่ 2.10 สัญลักษณ์ของบริการ Amazon EKS
(**ที่มา**: https://www.ambient-it.net/formation/formation-eks/)

Amazon Elastic Kubernetes Service (Amazon EKS) [9] สัญลักษณ์บริการดังแสดงในรูปที่ 2.10 คือ บริการ Kubernetes ที่ทำให้นักพัฒนาสามารถเรียกใช้งาน Kubernetes บน Amazon Web Service (AWS) และภายในองค์กรได้อย่างง่ายดาย

Amazon EKS ทำหน้าที่จัดการความพร้อมใช้งานและความสามารถในการปรับขนาดได้แบบอัตโนมัติของโหนดชั้นการควบคุม Kubernetes ที่รับผิดชอบในการกำหนดคอนเทนเนอร์ จัดการความพร้อมใช้งานของแอปพลิเคชัน จัดเก็บข้อมูลคลัสเตอร์ และภารกิจหลักอื่น ๆ

2.10 บริการ Amazon API Gateway

รูปที่ 2.11 สัญลักษณ์ของบริการ Amazon API Gateway
(**ที่มา**: https://aws.amazon.com/th/architecture/icons/)

Amazon API Gateway [10] สัญลักษณ์บริการดังแสดงในรูปที่ 2.11 เป็นบริการที่มีการจัดการเต็มรูปแบบ ซึ่งทำให้นักพัฒนาสามารถสร้าง เผยแพร่ บำรุงรักษา เฝ้าติดตาม และรักษาความปลอดภัยของ API ได้ทุกขนาด API ทำหน้าที่เป็น "ประตูหน้า" สำหรับแอปพลิเคชันเพื่อเข้าถึงข้อมูล ตรรกะทางธุรกิจ หรือการทำงานจากบริการแบ็คเอนด์ การใช้ API Gateway ทำให้นักพัฒนาสามารถสร้าง RESTful API และ WebSocket API ที่เปิดใช้งานแอปพลิเคชันการสื่อสารสองทางแบบเรียลไทม์ API Gateway รองรับปริมาณงานที่ใส่ในคอนเทนเนอร์และไร้เซิร์ฟเวอร์ รวมถึงเว็บแอปพลิเคชัน

API Gateway จัดการงานทั้งหมดที่เกี่ยวข้องในการรับและประมวลผลการเรียกใช้ API ที่เกิดขึ้นพร้อมกันหลายแสนรายการ รวมถึงการจัดการปริมาณการใช้งาน การรองรับ CORS การอนุมัติ และการควบคุมการเข้าถึง การเฝ้าติดตาม และการจัดการเวอร์ชัน API Gateway ไม่มีค่าธรรมเนียมขั้นต่ำหรือค่าใช้จ่ายเริ่มต้น นักพัฒนาสามารถจ่ายค่าบริการตามจำนวนการใช้งานได้

2.11 บริการ Amazon Web Service Lambda (AWS Lambda)

รูปที่ 2.12 สัญลักษณ์ของบริการ AWS Lambda
(**ที่มา**: https://commons.wikimedia.org/wiki/Amazon_Lambda_architecture_logo.svg)

Amazon Web Service Lambda (AWS Lambda) [11] สัญลักษณ์บริการดังแสดงในรูปที่ 2.12 คือ บริการประมวลผลแบบไร้เซิร์ฟเวอร์ในรูปแบบ Function as a Service (FaaS) ที่จะทำงานเมื่อมีการเรียกใช้งานเท่านั้น ซึ่งช่วยให้นักพัฒนาเรียกใช้โค้ดสำหรับบริการแอปพลิเคชันหรือแบ็คเอนด์แทบทุกประเภทได้โดยไม่ต้องมีการจัดเตรียมหรือจัดการเซิร์ฟเวอร์ นักพัฒนาสามารถเรียกใช้ Lambda ได้จากบริการของ Amazon Web Service (AWS)

2.12 บริการ Amazon DynamoDB

รูปที่ 2.13 สัญลักษณ์ของบริการ Amazon DynamoDB
(**ที่มา**: https://usefulangle.com/post/332/dynamodb-attribute-types)

Amazon DynamoDB [12] สัญลักษณ์บริการดังแสดงในรูปที่ 2.13 คือ ฐานข้อมูลประเภท NoSQL ที่รองรับโมเดลข้อมูลแบบคีย์-ค่าและโมเดลข้อมูลเอกสาร นักพัฒนาสามารถใช้ DynamoDB เพื่อสร้างแอปพลิเคชันแบบไร้เซิร์ฟเวอร์ที่ทันสมัยซึ่งสามารถเริ่มต้นได้จากขนาดเล็กและปรับขนาดเป็นระดับโลกเพื่อรองรับข้อมูลระดับเพตะไบต์และคำขอในการอ่านและเขียนกว่าสิบล้านคำขอต่อวินาที DynamoDB ถูกออกแบบมาเพื่อเรียกใช้แอปพลิเคชันประสิทธิภาพสูงในระดับอินเทอร์เน็ตที่อาจเป็นภาระที่ใหญ่เกินไปสำหรับระบบฐานข้อมูลเชิงสัมพันธ์แบบดั้งเดิม

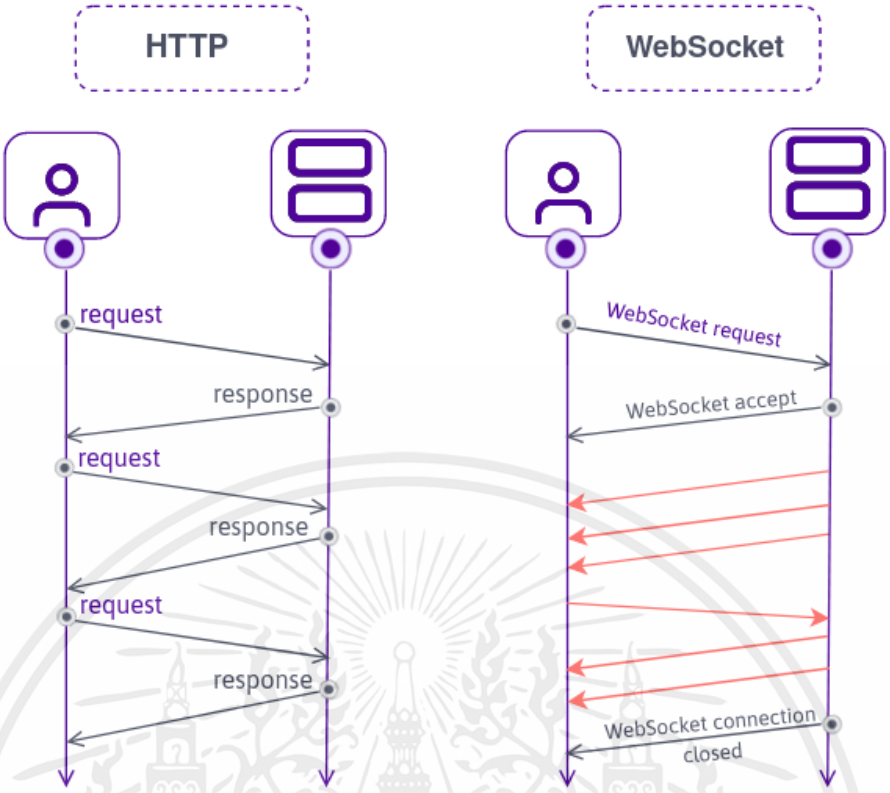
2.13 Redis

รูปที่ 2.14 สัญลักษณ์ของโปรแกรม Redis
(**ที่มา**: https://usefulangle.com/post/332/dynamodb-attribute-types)

Redis [13] สัญลักษณ์โปรแกรมดังแสดงในรูปที่ 2.14 คือ ซอฟต์แวร์โอเพ่นซอร์ส ที่ใช้ในการเก็บข้อมูลในรูปแบบโครงสร้างภายใน Memory หรือการเก็บข้อมูลใน RAM โดยผู้ใช้สามารถใช้งานเป็นฐานข้อมูลแบบชั่วคราวหรือใช้ในการเก็บแคช (Cache) Redis ถูกนับว่าเป็นฐานข้อมูลชนิด NoSQL ชนิดหนึ่งที่เก็บข้อมูลในลักษณะ key-value และมีจุดเด่นในด้านความเร็วในการทำงานเนื่องจากเป็นการเก็บข้อมูลใน RAM

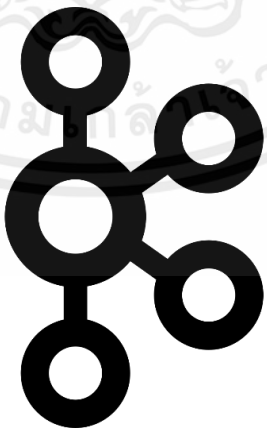
2.14 WebSocket Protocol

WebSocket protocol [14] คือ protocol ที่ทำงานอยู่บน Socket ที่เป็นการเชื่อมต่อแบบ Transmission Control Protocol (TCP) รองรับการใช้งานในรูปแบบ Full-duplex ซึ่งเป็นการสื่อสารแบบสองทิศทาง ทั้งฝั่งผู้ส่งและผู้รับ WebSocket protocol นิยมนำมาใช้กับระบบที่ต้องการการอัพเดทข้อมูลแบบ real-time เช่น ระบบ Chat ตัวอย่างการเชื่อมต่อของ WebSocket ดังแสดงในรูปที่ 2.15



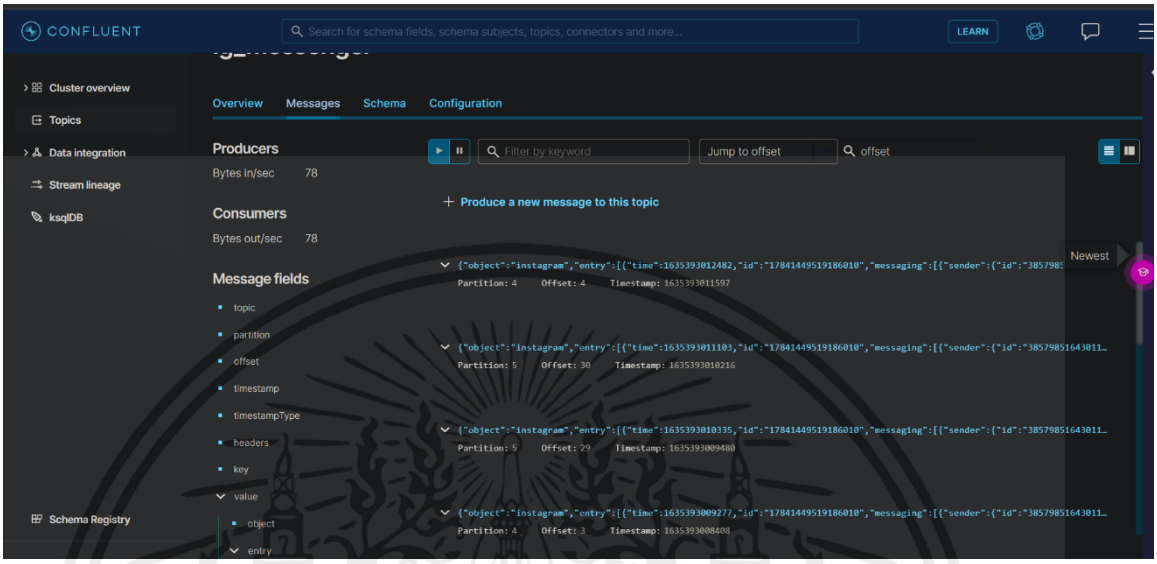
รูปที่ 2.15 ตัวอย่างแสดงความแตกต่างระหว่างการเชื่อมต่อแบบ HTTP และ WebSocket
(**ที่มา**: https://blog.scaleway.com/iot-hub-what-use-case-for-websockets)

2.15 Apache Kafka



รูปที่ 2.16 สัญลักษณ์ของโปรแกรม Apache Kafka
(**ที่มา**: https://commons.wikimedia.org/wiki/File:Apache_kafka-icon.svg)

Apache Kafka [15] สัญลักษณ์โปรแกรมดังแสดงในรูปที่ 2.16 คือ แพลตฟอร์มโอเพ่นซอร์ส ที่ใช้สำหรับการทำ Messaging queue ที่ถูกออกแบบให้เป็นศูนย์กลางของการส่งข้อความในองค์กร ถูกพัฒนาโดยบริษัท LinkedIn ตัวอย่างแสดงดังรูปที่ 2.17

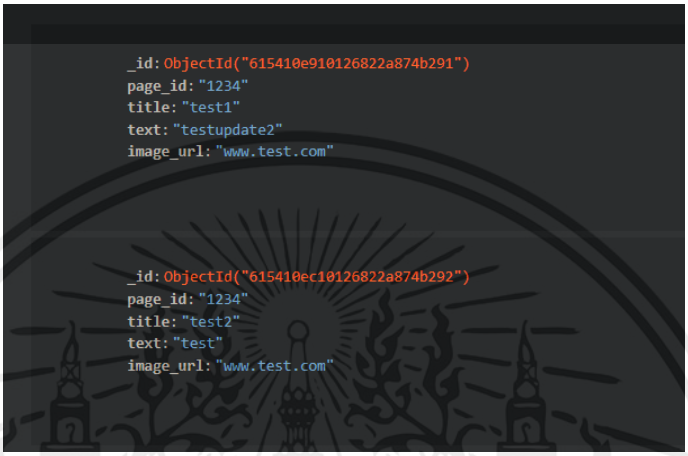


รูปที่ 2.17 ตัวอย่างการใช้งาน Apache Kafka

2.16 ฐานข้อมูล MongoDB

รูปที่ 2.18 สัญลักษณ์ของฐานข้อมูล MongoDB
(**ที่มา**: https://www.pngkey.com/maxpic/u2e6o0r5i1u2i1r5/)

MongoDB [16] สัญลักษณ์โปรแกรมดังแสดงในรูปที่ 2.18 คือ Open-Source Document Database รูปแบบหนึ่งที่ใช้เป็นฐานข้อมูลประเภท NoSQL หรือก็คือการไม่มีความสัมพันธ์ (Relation) ของตารางฐานข้อมูลแบบ SQL ทั่ว ๆ ไป แต่จะใช้วิธีการเก็บข้อมูลในรูปแบบ JSON (JavaScript Object Notation) โดยการบันทึกข้อมูลทุก ๆ Record ใน MongoDB ซึ่งจะเรียกว่าเป็น Document ที่จะเก็บค่าข้อมูลในลักษณะของ Key และ Value ตัวอย่างแสดงดังรูปที่ 2.19



รูปที่ 2.19 ตัวอย่างการใช้งาน MongoDB

2.17 แชทสดหรือไลฟ์แชท (Live Chat)

แชทสดหรือไลฟ์แชท (Live Chat) [17] คือ เครื่องมือสื่อสารทางออนไลน์ที่เพิ่มลงบนหน้าเว็บไซต์ เครื่องมือนี้จะช่วยให้ผู้ที่ใช้งานเว็บไซต์สามารถพูดคุยโต้ตอบ และสอบถามรายละเอียดต่าง ๆ กับร้านค้าออนไลน์ได้ ซึ่งไลฟ์แชทเป็นอีกช่องทางติดต่อสำคัญและเป็นที่นิยม เนื่องจากความสะดวกสบายในการใช้งาน สามารถถามตอบเสมือนการพูดคุยต่อหน้าได้ทันที อีกทั้งยังสามารถทิ้งข้อความไว้เพื่อให้เจ้าหน้าที่ติดต่อกลับได้ด้วย

ผลการสำรวจของ Forrester พบว่า 77% ของผู้ซื้อออนไลน์ ต้องการติดต่อกับผู้ขายก่อนซื้อสินค้าจริง และ 55% ของผู้ซื้อ จะไม่ซื้อสินค้าหากผู้ให้บริการตอบคำถามของพวกเขาช้าเกินไป

ดังนั้น สำหรับธุรกิจที่มีร้านค้าออนไลน์หรือเว็บไซต์ขายของออนไลน์ รวมถึงเว็บไซต์ที่เป็นสื่อกลางในการนำเสนอสินค้าและบริการไม่ควรพลาดที่จะติดตั้งระบบไลฟ์แชทไว้บนหน้าเว็บไซต์ เพราะระบบ Chat จะช่วยสนับสนุนการทำธุรกิจออนไลน์ผ่านเว็บไซต์ได้เป็นอย่างดี

ประโยชน์ของระบบไลฟ์แชทมีดังนี้

1. สามารถตอบข้อสงสัยและให้ข้อมูลได้ทันทีทำให้มีโอกาสปิดการขายได้มากกว่า เพราะหน้าที่หลักของระบบไลฟ์แชทเป็นช่องทางในการสื่อสารและตอบคำถามต่าง ๆ ซึ่ง

สามารถสนทนาโต้ตอบกันได้แบบทันทีทันใด (Real Time) ระหว่างเจ้าของร้านค้าออนไลน์กับลูกค้าที่กำลังชมสินค้าบนหน้าเว็บขายของออนไลน์ เมื่อลูกค้ามีข้อสงสัยเกี่ยวกับสินค้าชิ้นไหน ก็อาจสอบถามมาทางระบบไลฟ์แชท และหากเจ้าของร้านสามารถตอบข้อสงสัยหรือนำเสนอสิ่งที่ตรงใจให้กับลูกค้าได้ในขณะนั้น ย่อมมีผลต่อการตัดสินในการซื้อสินค้าและช่วยสร้างโอกาสในการซื้อขายให้เกิดได้มากยิ่งขึ้น

2. สร้างประสบการณ์ที่ดีเมื่อลูกค้าเข้าชมเว็บไซต์ อย่างที่กล่าวไปในข้างต้นว่า อีกหนึ่งข้อดีของระบบไลฟ์แชท คือช่วยสร้างประทับใจและประสบการณ์ที่ดีให้ลูกค้า เพราะในยุคปัจจุบันการดำเนินชีวิตและพฤติกรรมของผู้บริโภคเปลี่ยนแปลงไป ยิ่งบนโลกออนไลน์ด้วยแล้วผู้บริโภคคาดหวังความรวดเร็ว ว่องไวในการสื่อสารตอบถาม และการนำเสนอสินค้าและบริการให้ตรงกับความต้องการของผู้บริโภคที่ทั้งใจร้อน เบื่อง่าย ขี้ระแวง ฯลฯ ซึ่งระบบไลฟ์แชทสามารถเติมเต็ม และตอบสนองพฤติกรรมผู้บริโภคที่เปลี่ยนไปได้ ด้วยการพิมพ์ข้อความโต้ตอบลูกค้าให้ได้ข้อมูลที่ต้องการในทันที นั่นจึงเป็นสิ่งที่ช่วยสร้างความประทับใจและประสบการณ์ที่ดีแก่ลูกค้า นอกจากนั้นยังช่วยเพิ่มความมั่นใจแก่ผู้ชมหรือลูกค้าได้ว่าเว็บไซต์หรือร้านค้าออนไลน์นี้มีตัวตนจริง ๆ
3. ช่องทางผ่าน Messenger กำลังเติบโตขึ้นอย่างรวดเร็ว ประสบการณ์ของการแชทผ่านช่องทางออนไลน์เริ่มกระจายไปยังกลุ่มลูกค้าในช่วงอายุต่างๆ มากขึ้น ไม่ว่าจะเป็นเด็ก ผู้ใหญ่ รวมถึงผู้สูงอายุก็เริ่มมีประสบการณ์ในการแชทกันมากขึ้น จึงทำให้ลูกค้าไม่ต้องปรับตัวในการเรียนรู้ใหม่เพื่อใช้งาน ทำให้ระบบไลฟ์แชท เข้ามามีบทบาทช่วยเหลือในการให้ข้อมูลแก่ลูกค้าอย่างดีและมีประสิทธิภาพมากขึ้น
4. สามารถผสมผสานระบบไลฟ์แชทเข้ากับระบบ CRM ได้ ตัวอย่างเช่น Chatday แพลตฟอร์มบริหารแชททุกช่องทางในที่เดียว สามารถเชื่อมเข้ากับระบบ R-CRM แพลตฟอร์มบริหารทีมขาย ที่ออกแบบมาเพื่อธุรกิจไทย (ระบบที่สร้างขึ้นมาเพื่อเก็บข้อมูลลูกค้า สามารถช่วยติดตาม ตรวจสอบความต้องการและการสั่งซื้อสินค้าของลูกค้าเพื่อให้เกิดการจงรักภักดีต่อร้านค้าออนไลน์ บริษัทหรือองค์กร) การขอให้ลูกค้าใส่รายละเอียดข้อมูลการติดต่อเพิ่มเติมก่อนที่จะเริ่มต้นแชทผ่านทางไลฟ์แชท ด้วยข้อมูลส่วนนี้จะช่วยให้ร้านค้าสามารถปรับปรุงและพัฒนาสินค้าและบริการ เพื่อให้เหมาะสมกับลูกค้าแต่ละคนได้ ซึ่งข้อมูลพื้นฐานที่กล่าวมา เช่น ชื่อ เบอร์โทร หรืออีเมลของลูกค้า โดยสามารถเชื่อมต่อกับระบบ CRM เพื่อช่วยให้ร้านค้าสามารถค้นหารายละเอียดของลูกค้าในฐานข้อมูลได้อีกด้วย

2.18 ระบบฐานข้อมูลแบบ NoSQL

ระบบฐานข้อมูลแบบ Non SQL หรือ Not only SQL (NoSQL) [18] คือ ระบบฐานข้อมูลที่ไม่ใช่ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) ฐานข้อมูลนี้ได้ถูกคิดค้นขึ้นมาเพื่อแก้ไขปัญหาหลัก ๆ 2 อย่างที่มีใน Relational Database Management System (RDBMS) คือ

1) เพิ่มความสามารถในการจัดเก็บข้อมูลที่มีรูปแบบไม่แน่นอน (Unstructured data)

2) เพิ่มความสามารถในการขยายระบบในรูปแบบแนวนอน (Horizontal Scalability) เพื่อรองรับปริมาณข้อมูลที่มากขึ้นในปัจจุบัน

ข้อดีของระบบฐานข้อมูลแบบ NoSQL

ระบบฐานข้อมูลแบบ NoSQL เป็นที่นิยมมากสำหรับแอปพลิเคชันในปัจจุบัน ยกตัวอย่างเช่น อุปกรณ์เคลื่อนที่ เว็บไซต์ และเกมที่จำเป็นต้องมีฐานข้อมูลที่ยืดหยุ่น ปรับขนาดได้ ประสิทธิภาพสูง และทำงานได้ดีเยี่ยมเพื่อมอบประสบการณ์ที่ดีให้แก่ผู้ใช้งาน มีข้อดีดังนี้

1. ความยืดหยุ่น โดยทั่วไป ฐานข้อมูลแบบ NoSQL จะมีแบบแผนยืดหยุ่นที่ทำให้การพัฒนาเกิดขึ้นเร็ว และทำซ้ำคำสั่งได้ดียิ่งขึ้นกว่าเดิม โมเดลข้อมูลที่ยืดหยุ่นทำให้ฐานข้อมูลแบบ NoSQL เหมาะสมที่สุดสำหรับข้อมูลแบบกึ่งมีโครงสร้างและไม่มีโครงสร้าง
2. ความสามารถในการปรับขนาด โดยทั่วไป ฐานข้อมูลแบบ NoSQL มักถูกออกแบบมาให้ปรับขนาดออกได้โดยใช้คลัสเตอร์แบบกระจายของฮาร์ดแวร์แทนการปรับขนาดขึ้นโดยเพิ่มเซิร์ฟเวอร์ที่มีราคาแพง และมีประสิทธิภาพสูง
3. ประสิทธิภาพสูง ฐานข้อมูลแบบ NoSQL ได้รับการปรับปรุงประสิทธิภาพสำหรับโมเดลข้อมูลบางโมเดล และเข้าถึงรูปแบบที่เปิดใช้งานประสิทธิภาพที่สูงกว่าการพยายามดำเนินการทำงานที่คล้ายกันด้วยฐานข้อมูลเชิงสัมพันธ์
4. ทำงานได้ดีเยี่ยม ฐานข้อมูล NoSQL มี API การทำงานและประเภทข้อมูลที่สร้างตามวัตถุประสงค์สำหรับโมเดลข้อมูลแต่ละโมเดลที่สอดคล้องกัน

ระบบฐานข้อมูลแบบ NoSQL มี 4 ประเภทดังนี้

1. คีย์-ค่า (Key-value) ฐานข้อมูลคีย์-ค่า สามารถแบ่งพาร์ติชันได้ดีและสามารถปรับขนาดแนวนอนได้ตามขนาดที่ต้องการ ซึ่งฐานข้อมูลประเภทอื่นไม่สามารถทำได้ กรณีใช้งานตัวอย่างเช่น สำหรับเล่นเกม เทคโนโลยีโฆษณา และ IoT ทำให้ฐานข้อมูลประเภทนี้เหมาะสำหรับโมเดลข้อมูลแบบคีย์-ค่ามากอย่างยิ่ง

2. เอกสาร (Document) ในโค้ดของแอปพลิเคชันส่วนใหญ่ มักจะมีการแสดงข้อมูลเป็นวัตถุ หรือเอกสารที่คล้ายกับ JSON เนื่องจากเป็นโมเดลข้อมูลที่มีประสิทธิภาพ และใช้งานง่ายสำหรับผู้พัฒนา ฐานข้อมูลแบบเอกสารช่วยให้ผู้พัฒนาจัดเก็บ และสืบค้นข้อมูลในฐานข้อมูลได้ง่ายขึ้น โดยใช้รูปแบบโมเดลเอกสารเดียวกันที่ใช้ในโค้ดของแอปพลิเคชัน ลักษณะที่ยืดหยุ่น เป็นกึ่งโครงสร้าง และเป็นลำดับขั้นของเอกสาร และฐานข้อมูลเอกสาร
3. กราฟ (Graph) วัตถุประสงค์ของฐานข้อมูลแบบกราฟ คือ เพื่อให้การสร้าง และการเรียกใช้แอปพลิเคชันที่ทำงานกับชุดข้อมูลเกิดขึ้นได้อย่างง่ายดาย กรณีใช้งานโดยทั่วไปสำหรับฐานข้อมูลแบบกราฟ ตัวอย่างเช่น เครือข่ายทางสังคม กลไกข้อเสนอแนะ การตรวจจับการปลอมแปลง และกราฟความรู้
4. ในหน่วยความจำ (Cache) เหมาะสำหรับแอปพลิเคชันสำหรับเกมและเทคโนโลยีโฆษณา มีกรณีการใช้งาน ตัวอย่างเช่น บอร์ดผู้นำ การจัดเก็บเซสชัน และการวิเคราะห์แบบเรียลไทม์ที่จำเป็นต้องใช้เวลาในการตอบสนองเป็นมิลลิวินาที และอาจมีปริมาณการรับส่งข้อมูลที่เพิ่มขึ้นอย่างรวดเร็วเกิดขึ้นได้ตลอดเวลา

2.19 เว็บฮุค (Webhook)

เว็บฮุค (Webhooks) [19] หรือที่เรียกว่า “reverse API” เป็นเครื่องมือที่ช่วยให้ระบบหรือแอป-พลิเคชันหนึ่งสามารถส่งการแจ้งเตือนเกี่ยวกับเหตุการณ์เฉพาะไปยังระบบหรือแอปพลิเคชันอื่นได้ในแบบเรียลไทม์

แนวคิดของเว็บฮุค

เป็นการใช้งาน API ในรูปแบบหนึ่งที่ผู้ให้บริการจะทำการส่งข้อมูลมาให้เมื่อเกิด “เหตุการณ์” (event) ที่ผู้ใช้ต้องการ เนื่องจากผู้ให้บริการเป็นผู้ส่งข้อมูลมาให้ ทำให้ข้อมูลที่ได้จะถูกส่งผ่าน Webhooks แบบ real-time โดยส่วนมากจะส่งผ่าน HTTP POST และข้อมูลจะอยู่ในรูปแบบ JSON หรือ XML ขึ้นอยู่กับผู้ให้บริการ

ข้อแตกต่างระหว่าง Webhooks และ API

คือ API ทำงานโดยการที่เซิร์ฟเวอร์ของผู้ใช้งานส่งคำขอไปยังเซิร์ฟเวอร์ของผู้ให้บริการ API เพื่อตรวจสอบข้อมูล ในทางกลับกัน Webhooks จะส่งข้อมูลจากผู้ให้บริการกลับมาที่ผู้ใช้งานโดยอัตโนมัติเมื่อเกิดเหตุการณ์ขึ้น

2.20 JSON Web Token (JWT)

JSON Web Token (JWT) [20] คือ เป็นมาตรฐานเปิด (RFC 7519) รูปแบบหนึ่งใช้ในการสร้างรหัส token จากข้อมูลประเภท JSON เพื่อการส่งข้อมูลอย่างปลอดภัยระหว่างกัน โดยมีจุดเด่นทางด้านขนาดที่กระทัดรัด (compact) และเก็บข้อมูลภายในตัว (self-contained) ตัวอย่างแสดงดังรูปที่ 2.20

รูปที่ 2.20 ตัวอย่างการใช้งาน JSON Web Token
(**ที่มา**: https://medium.com/rootusercc/json-web-token-มาตรฐานใหม่-ในการทำ-authentication-b0760dd9acd1)

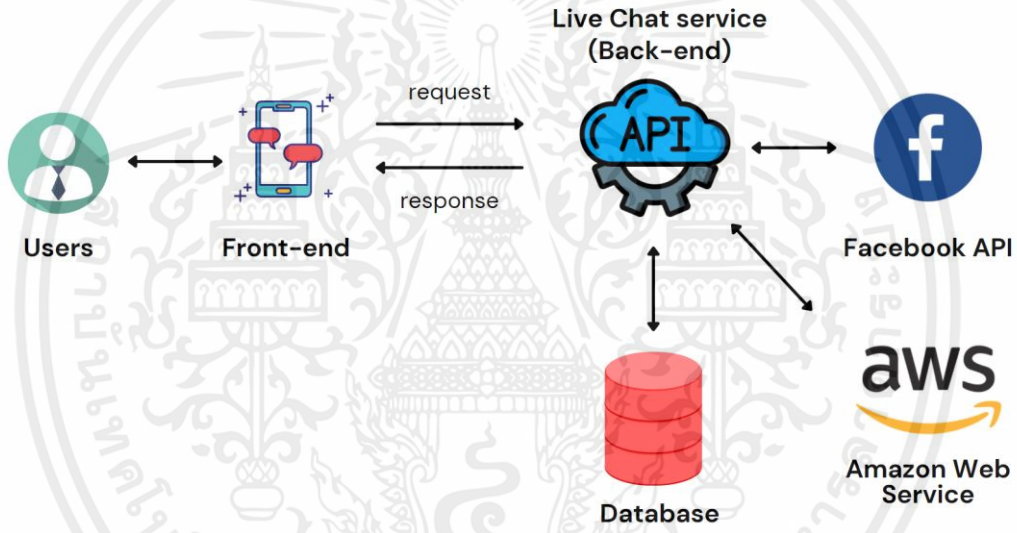
JSON Web Token (JWT) มีโครงสร้างแบ่งออกเป็น 3 ส่วน ดังนี้

1. Header ใช้ในการเก็บว่าข้อความชุดนี้เป็นการเข้ารหัสแบบใด ตัวอย่างเช่น SHA256, RSA
2. Payload ใช้ในการเก็บข้อมูลจริง เช่น User ID, Roles, E-mail เป็นต้น
3. Signature ส่วน Digital Signed ใช้ในการตรวจสอบความถูกต้องของ token

บทที่ 3
หลักการและการออกแบบ

ในบทนี้จะกล่าวถึงขั้นตอนการดำเนินการออกแบบระบบแชทสดสำหรับลูกค้าของบอทไอโอ ประกอบด้วย ภาพรวมของระบบแชทสดสำหรับลูกค้าของบอทไอโอ ระบบการสนทนาของระบบแชทสดสำหรับลูกค้าของบอทไอโอ การยืนยันตัวตน ฐานข้อมูล และเอกสารอธิบาย API ผู้จัดทำดำเนินงานตามขั้นตอนดังต่อไปนี้

3.1 ภาพรวมระบบแชทสดสำหรับลูกค้าของบอทไอโอ



รูปที่ 3.1 ภาพรวมของระบบแชทสดสำหรับลูกค้าของบอทไอโอ

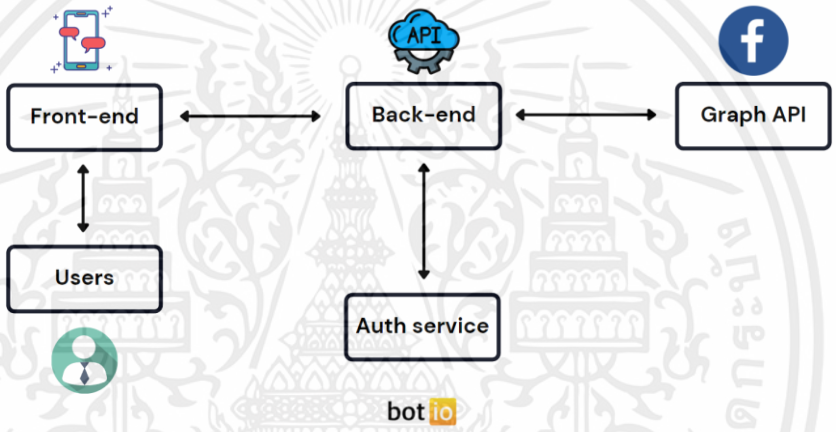
ผู้จัดทำได้ออกแบบระบบแชทสดสำหรับลูกค้าของบอทไอโอดังรูปที่ 3.1 โดยผู้ใช้งาน (Users) จะไม่ได้ใช้งานระบบที่จัดทำขึ้นโดยตรง แต่จะสามารถใช้งานระบบได้ผ่านแอปพลิเคชันในส่วนที่เรียกว่า Front-end หรือส่วนติดต่อผู้ใช้ (User interface) จากนั้นแอปพลิเคชันในส่วน Front-end จะทำการเรียกใช้งาน (Request) ระบบที่ได้จัดทำขึ้นเพื่อตอบสนองความต้องการของผู้ใช้งาน (Users) ยกตัวอย่างเช่น การแสดงข้อมูลประวัติการสนทนา ระบบที่จัดทำขึ้นจะทำการติดต่อกับทาง Facebook API เพื่อทำการดึงข้อมูลประวัติการสนทนา จากนั้นระบบจะทำการส่งคืนข้อมูล (Response) ที่ได้จาก Facebook API ไปแสดงผลบนแอปพลิเคชันในส่วน Front-end

กล่าวคือระบบที่จัดทำขึ้นจะเป็นตัวกลางในการเชื่อมต่อระหว่างแอป-พลิเคชันในส่วน Front-end ฐานข้อมูล บริการ Amazon Web Service (AWS) และ Facebook API

3.2 ระบบการสนทนาของระบบแชทสดสำหรับลูกค้าของบอทไอโอ

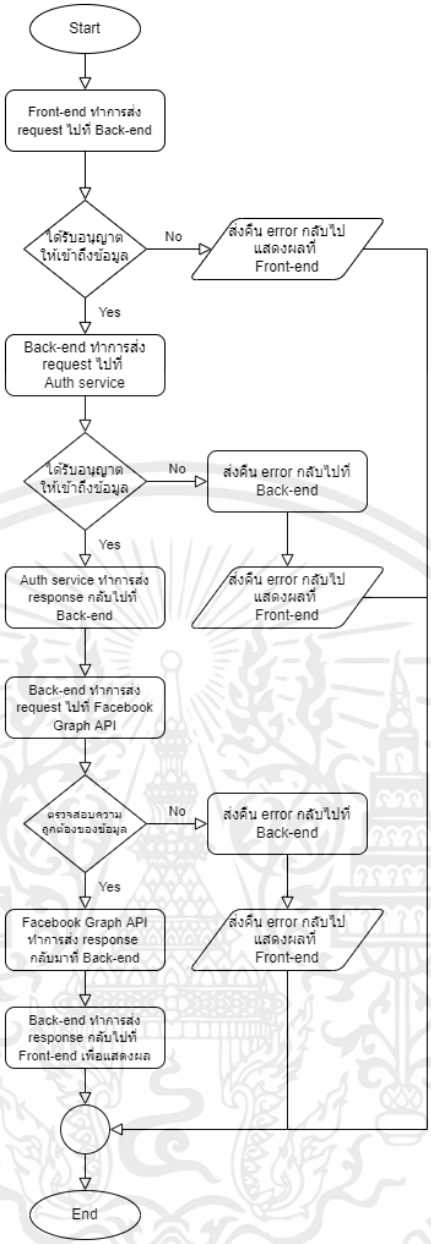
ระบบการสนทนาของระบบแชทสดสำหรับลูกค้าของบอทไอโอ เกิดจากการรวบรวมระบบย่อย ๆ ตัวอย่างเช่น API การสนทนา (Conversation API) เว็บฮุค (Webhook) การเชื่อมต่อเว็บซ็อกเก็ต (WebSocket) และการส่งข้อความผ่านเว็บซ็อกเก็ต เข้าไว้ด้วยกัน โดยสามารถแบ่งรายละเอียดการทำงานของแต่ละส่วนได้ดังนี้

3.2.1 API การสนทนา (Conversation API)



รูปที่ 3.2 บล็อกไดอะแกรมการทำงานของ API การสนทนา (Conversation API)

จากรูปที่ 3.2 บล็อกไดอะแกรมการทำงานของ API การสนทนา (Conversation API) จะประกอบด้วย Live Chat API หรือที่เรียกว่าระบบหลังบ้าน (Back-end) ซึ่งเป็นระบบที่ผู้จัดทำสร้างขึ้นเพื่อเป็นตัวการในการเชื่อมต่อกับส่วนติดต่อกับผู้ใช้งานหรือ Front-end เมื่อผู้ใช้เริ่มใช้งานระบบแชทสด (Live Chat) ส่วนติดต่อกับผู้ใช้งาน (Front-end) จะทำการเรียกใช้งาน Live Chat API เพื่อทำการดึงข้อมูลการสนทนา จากนั้นระบบ Live Chat API จะทำการตรวจสอบสิทธิ์ของผู้ใช้ผ่านทางบริการการยืนยันตัวตน หรือ Authentication service จากนั้น Live Chat API จะทำการเรียกใช้งาน Facebook Graph API เพื่อขอข้อมูลการสนทนา จากนั้นจะส่งคืนข้อมูลที่ได้ไปแสดงที่ส่วนติดต่อกับผู้ใช้งาน



รูปที่ 3.3 แผนผังการทำงานของ API การสนทนา (Conversation API)

การทำงานของระบบแชทสดสำหรับลูกค้าของบอทไอโอในส่วนของ API การสนทนา (Conversation API) ดังแสดงในรูปที่ 3.3 ซึ่งมีขั้นตอนการทำงานดังนี้

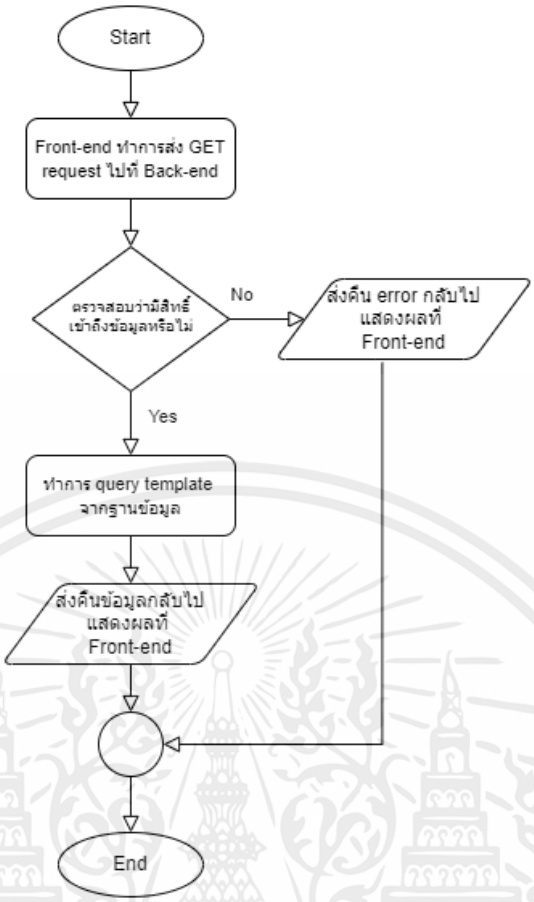
1. เริ่มต้นจากการที่แอปพลิเคชันในส่วนติดต่อผู้ใช้งาน หรือที่เรียกว่าส่วนหน้าบ้าน (Front-end) ทำการส่งคำร้องขอ (Request) ไปที่ระบบที่ผู้จัดทำได้สร้างขึ้นหรือที่เรียกว่าส่วนหลังบ้าน (Back-end)
2. จากนั้นระบบยืนยันตัวตนของส่วนหลังบ้าน (Back-end) จะทำการตรวจสอบสิทธิ์การเข้าถึงของผู้ที่ส่งคำร้องขอ (request) ที่ได้รับมา

3. กรณีที่ผู้ที่ส่งคำร้องขอ (request) ไม่มีสิทธิ์เข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะทำการส่งคืนข้อความ error กลับไปแสดงผลที่ส่วนหน้าบ้าน (Front-end) และจบการทำงาน
4. กรณีที่ผู้ที่ส่งคำร้องขอ (request) มีสิทธิ์เข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะทำการส่งคำร้องขอ (request) ไปที่ระบบยืนยันตัวตนของบอทไอโอ (Botio Authentication service)
5. ระบบยืนยันตัวตนของบอทไอโอจะทำการตรวจสอบสิทธิ์การเข้าถึงของผู้ที่ส่งคำร้องขอ (request) ที่ได้รับมา ซึ่งในที่นี้คือระบบหลังบ้าน (Back-end)
6. กรณีที่ระบบหลังบ้าน (Back-end) ไม่มีสิทธิ์เข้าถึงข้อมูล ระบบยืนยันตัวตนของบอทไอโอจะทำการส่งคืนข้อมูล error กลับไปที่ระบบหลังบ้าน (Back-end)
7. จากนั้นระบบหลังบ้าน (Back-end) จะทำการส่งคืนข้อความ error กลับไปแสดงผลที่ส่วนหน้าบ้าน (Front-end) และจบการทำงาน
8. กรณีที่ระบบหลังบ้าน (Back-end) มีสิทธิ์เข้าถึงข้อมูล ระบบยืนยันตัวตนของบอทไอโอจะทำการส่งคืนข้อมูล (response) กลับไปที่ระบบหลังบ้าน (Back-end)
9. ระบบหลังบ้าน (Back-end) ทำการส่งคำร้องขอ (request) ไปที่ Facebook Graph API โดยใช้ข้อมูลที่ได้มาจากระบบยืนยันตัวตนของบอทไอโอ
10. จากนั้น Facebook Graph API จะทำการตรวจสอบคำร้องขอ (request) และข้อมูลที่ส่งมาจากระบบหลังบ้าน (Back-end)
11. กรณีที่ข้อมูลที่ได้รับมาไม่ถูกต้อง Facebook Graph API จะทำการส่งคืนข้อความ error กลับไปที่ระบบหลังบ้าน (Back-end)
12. จากนั้นระบบหลังบ้าน (Back-end) จะทำการส่งคืนข้อความ error กลับไปแสดงผลที่ส่วนหน้าบ้าน (Front-end) และจบการทำงาน
13. กรณีที่ข้อมูลที่ได้รับมาถูกต้อง Facebook Graph API จะทำการส่งคืนข้อมูล (response) กลับไปที่ระบบหลังบ้าน (Back-end)
14. จากนั้นระบบหลังบ้าน (Back-end) จะทำการส่งคืนข้อมูล (response) ที่ได้รับมาจาก Facebook Graph API กลับไปที่ส่วนหน้าบ้าน (Front-end) เพื่อใช้ในการแสดงผล

3.2.2 ระบบข้อความแบบ Template

รูปที่ 3.4 บล็อกไดอะแกรมการทำงานของระบบข้อความแบบ Template

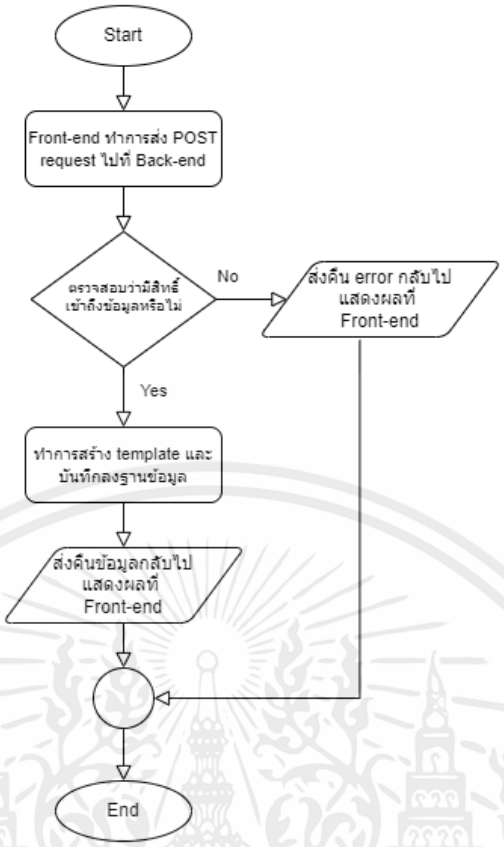
จากรูปที่ 3.4 บล็อกไดอะแกรมการทำงานของระบบข้อความแบบ Template จะประกอบด้วยระบบหลังบ้าน (Back-end) เป็นตัวกลางในการเชื่อมต่อระหว่างส่วนหน้าบ้าน (Front-end) และฐานข้อมูล MongoDB โดยส่วนหน้าบ้าน (Front-end) จะสามารถดำเนินการสืบค้น สร้าง แก้ไข และลบข้อมูลในฐานข้อมูลได้ด้วย HTTP Method (GET POST PUT DELETE) ซึ่งมีรายละเอียดในการออกแบบแผนผังการทำงานของแต่ละส่วนดังต่อไปนี้



รูปที่ 3.5 แผนผังการทำงานของการสืบค้นข้อมูลข้อความแบบ Template

การทำงานของระบบการสืบค้นข้อมูลข้อความแบบ Template แสดงในรูปที่ 3.5 มีขั้นตอนการทำงานดังนี้

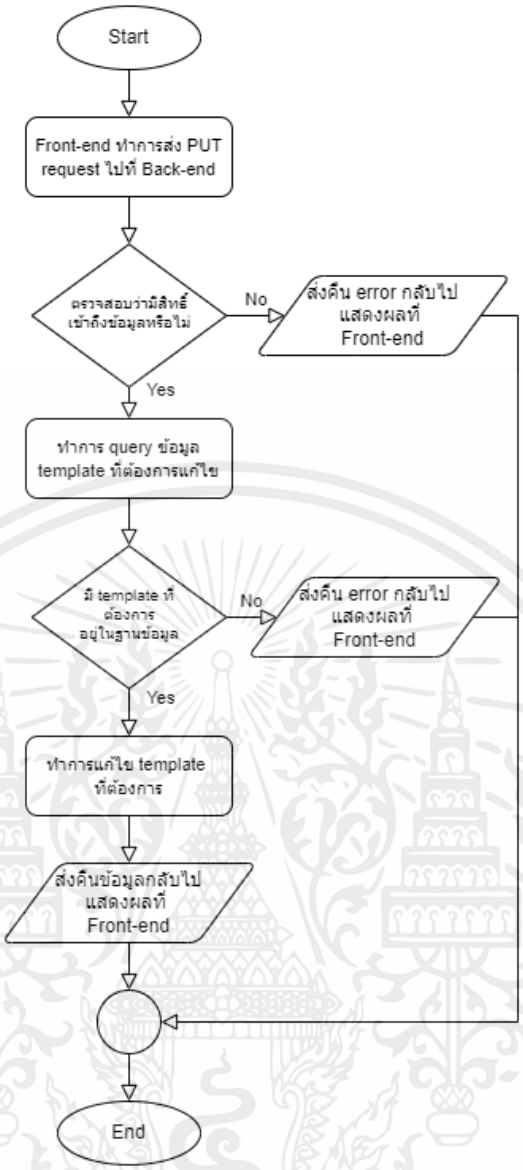
1. เริ่มต้นจากส่วนหน้าบ้าน (Front-end) ทำการส่งคำร้องขอ (request) GET ไปยังระบบหลังบ้าน (Back-end)
2. ระบบหลังบ้าน (Back-end) จะทำการตรวจสอบสิทธิ์การเข้าถึงข้อมูล
3. กรณีที่ไม่มีสิทธิ์ในการเข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะส่งคืน error กลับไปแสดงผลในส่วนหน้าบ้าน (Front-end) และจบการทำงาน
4. กรณีที่มีสิทธิ์ในการเข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะทำการสืบค้นข้อมูลข้อความแบบ template
5. จากนั้นจะส่งคืนข้อมูลกลับไปแสดงผลในส่วนหน้าบ้าน (Front-end)



รูปที่ 3.6 แผนผังการทำงานของการสร้างข้อความแบบ Template

การทำงานของระบบการสร้างข้อความแบบ Template แสดงในรูปที่ 3.6 มีขั้นตอนการทำงานดังนี้

1. เริ่มต้นจากส่วนหน้าบ้าน (Front-end) ทำการส่งคำร้องขอ (request) POST ไปยังระบบหลังบ้าน (Back-end)
2. ระบบหลังบ้าน (Back-end) จะทำการตรวจสอบสิทธิ์การเข้าถึงข้อมูล
3. กรณีที่ไม่มีสิทธิ์ในการเข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะส่งคืน error กลับไปแสดงผลในส่วนหน้าบ้าน (Front-end) และจบการทำงาน
4. กรณีที่มีสิทธิ์ในการเข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะทำการสร้าง template และบันทึกลงฐานข้อมูล MongoDB
5. จากนั้นจะส่งคืนข้อมูลกลับไปแสดงผลในส่วนหน้าบ้าน (Front-end)

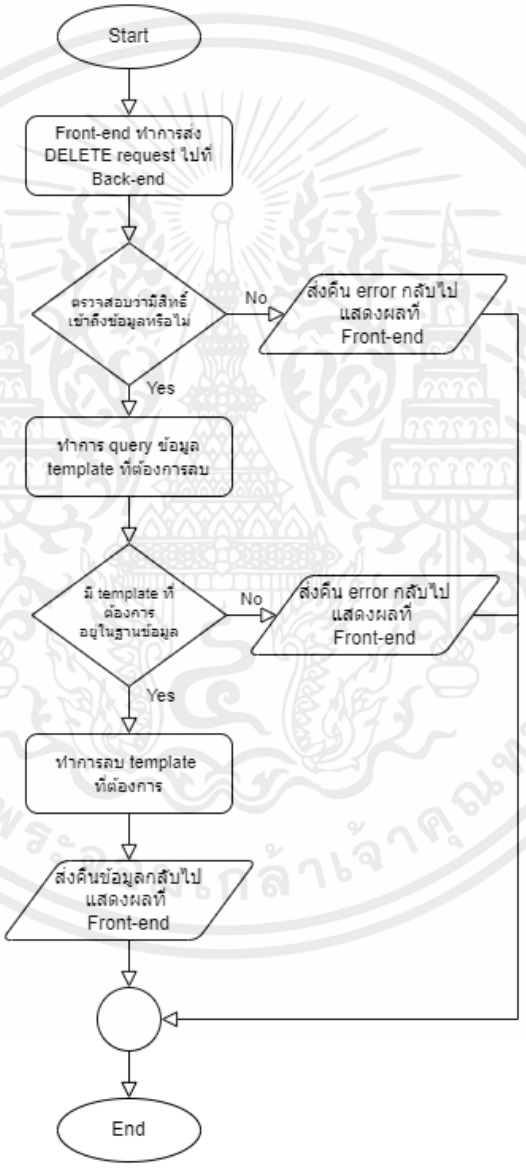


รูปที่ 3.7 แผนผังการทำงานของการแก้ไขข้อความแบบ Template

การทำงานของระบบการแก้ไขข้อความแบบ Template แสดงในรูปที่ 3.7 มีขั้นตอนการทำงานดังนี้

1. เริ่มต้นจากส่วนหน้าบ้าน (Front-end) ทำการส่งคำร้องขอ (request) PUT ไปยังระบบหลังบ้าน (Back-end)
2. ระบบหลังบ้าน (Back-end) จะทำการตรวจสอบสิทธิ์การเข้าถึงข้อมูล
3. กรณีที่ไม่มีสิทธิ์ในการเข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะส่งคืน error กลับไปแสดงผลในส่วนหน้าบ้าน (Front-end) และจบการทำงาน
4. กรณีที่มีสิทธิ์ในการเข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะทำการสืบค้นข้อมูลข้อความแบบ template ที่ต้องการจะแก้ไขในฐานข้อมูล

5. จากนั้นตรวจสอบว่าในฐานข้อมูลมีข้อความแบบ template ตรงกับที่ต้องการจะแก้ไขหรือไม่
6. กรณีที่ไม่มีข้อความ template ตรงกัน ระบบหลังบ้าน (Back-end) จะทำการส่งคืน error กลับไปแสดงผลในส่วนหน้าบ้าน (Front-end) และจบการทำงาน
7. กรณีที่มีข้อความ template ที่ตรงกัน ระบบหลังบ้าน (Back-end) จะทำการแก้ไขข้อมูลในฐานข้อมูล
8. จากนั้นจะส่งคืนข้อมูลกลับไปแสดงผลในส่วนหน้าบ้าน (Front-end)

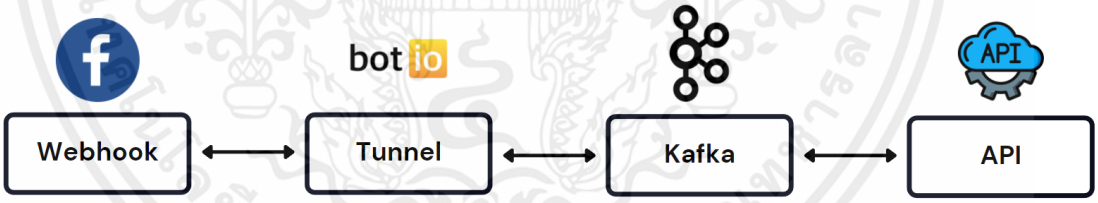


รูปที่ 3.8 แผนผังการทำงานของการลบข้อความแบบ Template

การทำงานของระบบการลบข้อความแบบ Template แสดงในรูปที่ 3.8 มีขั้นตอนการทำงานดังนี้

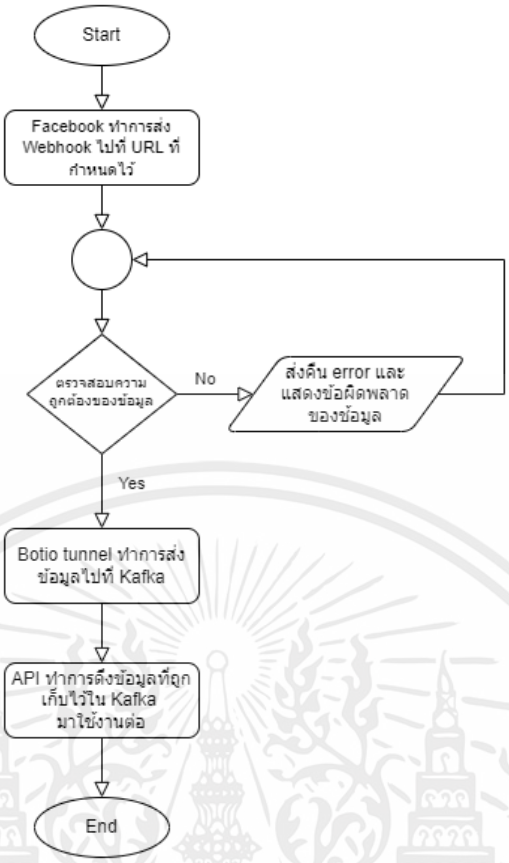
1. เริ่มต้นจากส่วนหน้าบ้าน (Front-end) ทำการส่งคำร้องขอ (request) DELETE ไปยังระบบหลังบ้าน (Back-end)
2. ระบบหลังบ้าน (Back-end) จะทำการตรวจสอบสิทธิ์การเข้าถึงข้อมูล
3. กรณีที่ไม่มีสิทธิ์ในการเข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะส่งคืน error กลับไปแสดงผลในส่วนหน้าบ้าน (Front-end) และจบการทำงาน
4. กรณีที่มีสิทธิ์ในการเข้าถึงข้อมูล ระบบหลังบ้าน (Back-end) จะทำการสืบค้นข้อมูลข้อความแบบ template ที่ต้องการจะลบในฐานข้อมูล
5. จากนั้นตรวจสอบว่าในฐานข้อมูลมีข้อความแบบ template ตรงกับที่ต้องการจะลบหรือไม่
6. กรณีที่ไม่มีข้อความ template ตรงกัน ระบบหลังบ้าน (Back-end) จะทำการส่งคืน error กลับไปแสดงผลที่ส่วนหน้าบ้าน (Front-end) และจบการทำงาน
7. กรณีที่มีข้อความ template ตรงกัน ระบบหลังบ้าน (Back-end) จะทำการลบข้อมูลในฐานข้อมูล
8. จากนั้นจะส่งคืนข้อมูลกลับไปแสดงผลในส่วนหน้าบ้าน (Front-end)

3.2.3 เว็บฮุค (Webhook)



รูปที่ 3.9 บล็อกไดอะแกรมการทำงานของเว็บฮุค (Webhook)

จากรูปที่ 3.9 บล็อกไดอะแกรมการทำงานของเว็บฮุค (Webhook) ประกอบด้วยระบบตัวหนึ่งของบอทไอโอที่เรียกว่า Botio Tunnel ซึ่งใช้ในการจัดการกับ Webhook โดยเริ่มต้นจากการที่ Facebook ทำการส่ง Webhook มาที่ Tunnel จากนั้น Tunnel จะทำการตรวจสอบความถูกต้องและผลิตข้อมูลที่ได้รับมาจาก Webhook และส่งต่อไปยัง Apache Kafka เพื่อให้ Apache Kafka ทำการจัดการกับ Messaging queue เพื่อความสมบูรณ์ของข้อมูลและความสะดวกของ API อื่น ๆ ในการนำข้อมูลจาก Apache Kafka ไปใช้งานต่อได้



รูปที่ 3.10 แผนผังการทำงานของเว็บฮุค (Webhook)

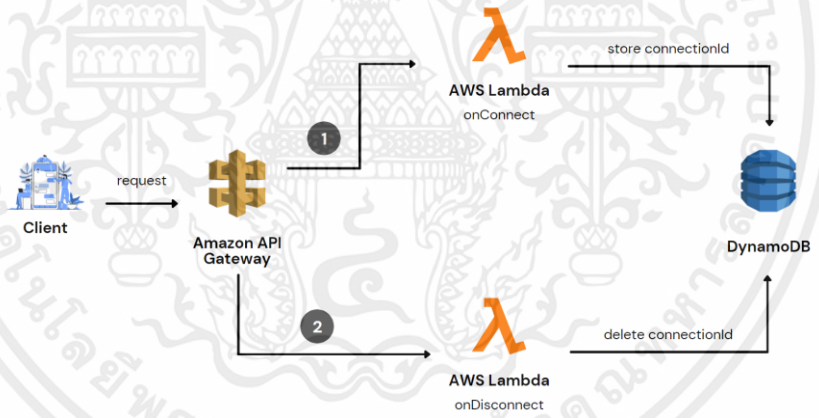
การทำงานของระบบแชทสดสำหรับลูกค้าของบอทไอโอในส่วนของเว็บฮุค (Webhook) แสดงในรูปที่ 3.10 มีขั้นตอนการทำงานดังนี้

1. เริ่มต้นจากการที่ Facebook ทำการส่งข้อมูล Webhook ไปยัง URL ที่ได้ทำการกำหนดไว้ ในกรณีนี้คือ Botio tunnel
2. เมื่อ Botio tunnel ได้รับข้อมูล Webhook ระบบจะทำการตรวจสอบความถูกต้องและความสมบูรณ์ของข้อมูล
3. กรณีที่ข้อมูล Webhook ที่ได้รับมาไม่ถูกต้อง ระบบจะทำการส่งคืน error และแสดงข้อผิดพลาดในลักษณะของ log message
4. กรณีที่ข้อมูล Webhook ที่ได้รับมาถูกต้อง ระบบจะทำการสร้างผู้ผลิต (producer) และผลิตข้อมูล Webhook ที่ได้ลงไปเก็บไว้ใน Apache Kafka
5. จากนั้นบริการ API อื่น ๆ จะสามารถเรียกใช้ข้อมูลจาก Webhook ได้ผ่านทาง Apache Kafka

3.2.4 การเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

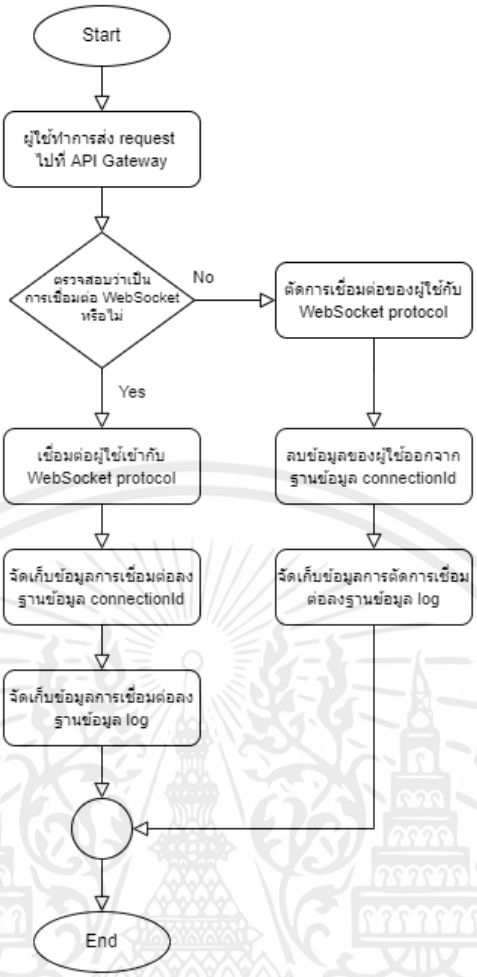
รูปที่ 3.11 บล็อกไดอะแกรมการทำงานของการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

จากรูปที่ 3.11 บล็อกไดอะแกรมการทำงานของการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket) จะประกอบด้วย Amazon API Gateway เป็นตัวกลางในการเชื่อมต่อระหว่างส่วนหน้าบ้าน (Front-end) และหลังบ้าน (Back-end) เนื่องจากผู้จัดทำได้ใช้ Amazon API Gateway ในการสร้างเว็บซ็อกเก็ต เพื่อให้สามารถส่งข้อมูลระหว่างส่วนหน้าบ้านและหลังบ้านได้แบบเรียวไทม์ผ่านเว็บซ็อกเก็ตโปรโตคอล (WebSocket protocol)



รูปที่ 3.12 แผนภาพการทำงานของ API Gateway ในการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

จากรูปที่ 3.12 แผนภาพการทำงานของ API Gateway ในการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket) โดยภายในประกอบด้วย AWS Lambda 2 ตัว เพื่อช่วยในการจัดการเชื่อมต่อเว็บซ็อกเก็ต และการตัดการเชื่อมต่อเว็บซ็อกเก็ต กรณีที่มีการเชื่อมต่อกับเว็บซ็อกเก็ต ระบบจะทำการจัดเก็บข้อมูลการเชื่อมต่อลงในฐานข้อมูล DynamoDB และหากมีการตัดการเชื่อมต่อกับเว็บซ็อกเก็ตระบบจะทำการลบข้อมูลออกจากฐานข้อมูล DynamoDB

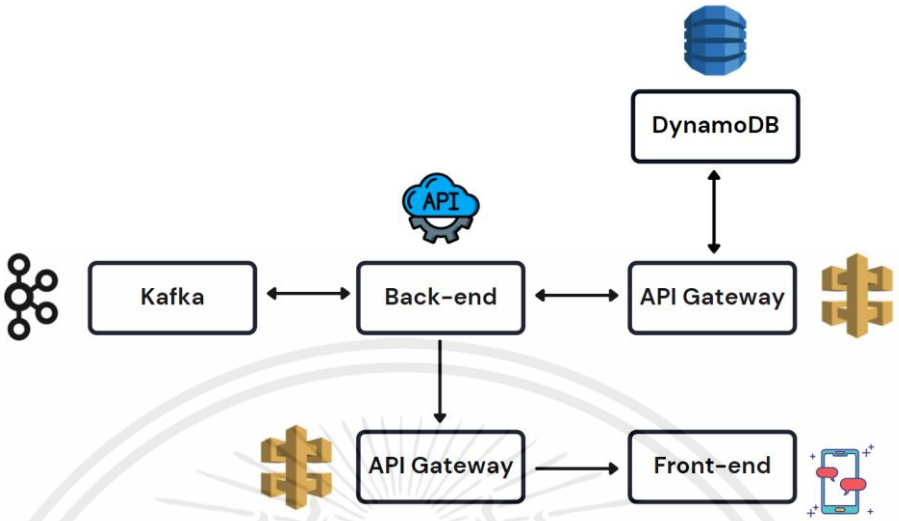


รูปที่ 3.13 แผนผังการทำงานของการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

การทำงานของระบบแชทสดสำหรับลูกค้าของบอทไอโอในส่วนของการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket) แสดงในรูปที่ 3.13 ซึ่งมีขั้นตอนการทำงานดังนี้

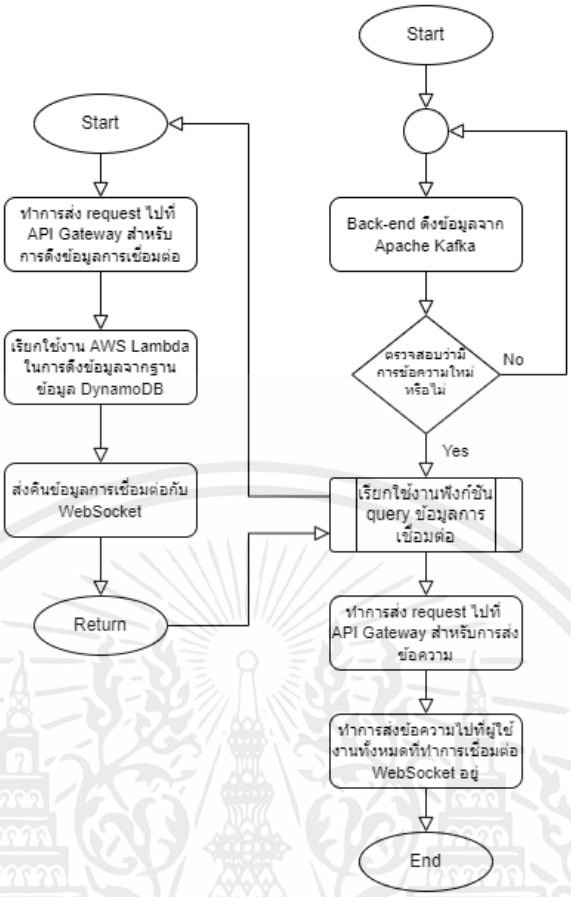
1. เริ่มต้นจากการที่ผู้ใช้งานทำการส่งคำร้องขอ (request) ไปที่ API Gateway Endpoint จากนั้นระบบจะทำการตรวจสอบว่าผู้ใช้งานต้องการเชื่อมต่อหรือตัดการเชื่อมต่อกับเว็บซ็อกเก็ต
2. ในกรณีที่เป็นการเชื่อมต่อกับเว็บซ็อกเก็ต ระบบจะทำการเชื่อมต่อผู้ใช้งานเข้ากับเว็บซ็อกเก็ตโปรโตคอล (WebSocket protocol)
3. ระบบจะจัดเก็บข้อมูลการเชื่อมต่อลงฐานข้อมูล connectionId
4. ระบบจะจัดเก็บข้อมูลการเชื่อมต่อลงฐานข้อมูลสำหรับใช้เก็บประวัติการเชื่อมต่อ
5. ในกรณีที่ไม่เป็นการเชื่อมต่อกับเว็บซ็อกเก็ต ระบบจะทำการตัดการเชื่อมต่อของผู้ใช้งานกับเว็บซ็อกเก็ต
6. ระบบจะทำการลบข้อมูลของผู้ใช้งานออกจากฐานข้อมูล DynamoDB
7. ระบบจะจัดเก็บข้อมูลการตัดการเชื่อมต่อลงฐานข้อมูลสำหรับใช้เก็บประวัติการเชื่อมต่อ

3.2.5 การส่งข้อความผ่านเว็บซ็อกเก็ต (WebSocket)



รูปที่ 3.14 บล็อกไดอะแกรมการทำงานของการส่งข้อความเว็บซ็อกเก็ต (WebSocket)

จากรูปที่ 3.14 บล็อกไดอะแกรมการทำงานของการส่งข้อความเว็บซ็อกเก็ต (WebSocket) จะประกอบด้วย ระบบหลังบ้าน (Back-end) เป็นตัวกลางในการส่งข้อความที่ได้รับมาจากเว็บฮุค (Webhook) จากนั้นระบบหลังบ้านจะทำการส่งคำร้องขอ (Request) ไปที่ Amazon API Gateway เพื่อทำการดึงข้อมูลการเชื่อมต่อ WebSocket ของทางฝั่งหน้าบ้าน (Front-end) ที่เก็บไว้ในฐานข้อมูล DynamoDB จากนั้นระบบหลังบ้านจะทำการส่งคำร้องขอไปที่ Amazon API Gateway อีกหนึ่งตัว เพื่อทำการส่งข้อมูลที่ได้รับมาจากเว็บฮุคไปที่ส่วนหน้าบ้านผ่านทาง WebSocket Protocol เพื่อนำไปใช้ในการแสดงผล



รูปที่ 3.15 แผนผังการทำงานของการส่งข้อความผ่านเว็บซ็อกเก็ต (WebSocket)

การทำงานของระบบแชทสดสำหรับลูกค้าของบอทไอโอในส่วนของการส่งข้อความผ่านเว็บ-ซ็อกเก็ตดังแสดงในรูปที่ 3.15 มีขั้นตอนการทำงานดังนี้

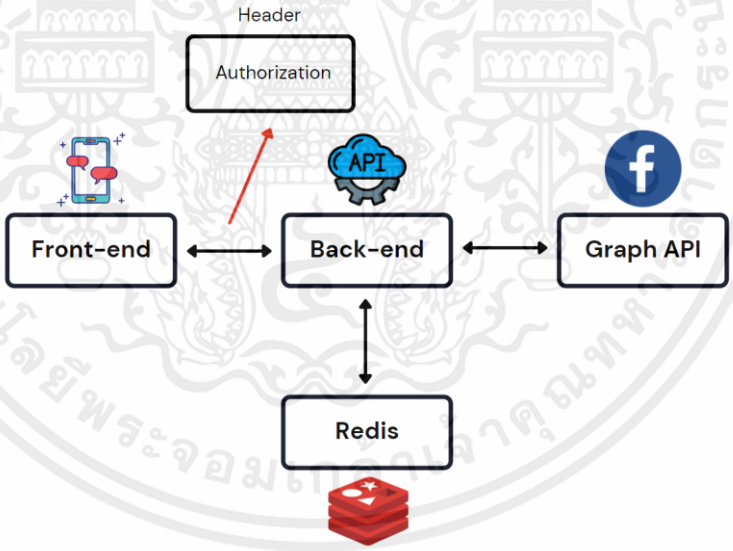
1. เริ่มต้นด้วยการที่ระบบหลังบ้าน (Back-end) จะทำการดึงข้อมูลจาก Apache Kafka
2. จากนั้นระบบจะทำการตรวจสอบว่ามีการส่ง Webhook ใหม่มาหรือไม่ หรือเป็นการตรวจสอบว่ามีข้อความใหม่ส่งมาหรือไม่
3. กรณีที่ไม่มีข้อความใหม่เข้ามา ระบบจะกลับไปทำการตรวจสอบข้อมูลจาก Apache Kafka ใหม่อีกครั้ง
4. กรณีที่มีข้อความใหม่เข้ามา ระบบจะการเรียกใช้งานฟังก์ชันในการ query ข้อมูลการเชื่อมต่อ WebSocket ของทางส่วนหน้าบ้าน (Front-end)
5. ฟังก์ชันในการ query ข้อมูลการเชื่อมต่อ จะเริ่มต้นด้วยการส่งคำร้องขอ (request) ไปที่ API Gateway ในส่วนของการดึงข้อมูลการเชื่อมต่อ
6. จากนั้น API Gateway จะเรียกใช้งาน AWS Lambda ในการดึงข้อมูลการเชื่อมต่อจากฐานข้อมูล DynamoDB

7. จากนั้น API Gateway จะทำการส่งคืนข้อมูลการเชื่อมต่อกลับไปที่ระบบหลังบ้าน (Back-end)
8. ระบบหลังบ้าน (Back-end) จะนำข้อมูลการเชื่อมต่อที่ได้รับมาจาก API Gateway มาใช้ในการส่งคำร้องขอ (request) ไปที่ API Gateway อีกตัวหนึ่ง เพื่อใช้ในการส่งข้อความ
9. API Gateway ที่ใช้ในการส่งข้อความจะทำการส่งข้อความไปยังส่วนหน้าบ้าน (Front-end) ทั้งหมดที่ทำการเชื่อมต่อเว็บซ็อกเก็ตอยู่

3.3 ระบบการยืนยันตัวตน (Authentication)

ระบบการยืนยันตัวตน (Authentication) ของระบบแชทสดสำหรับลูกค้าของบอทไอโอเป็นส่วนสำคัญในการทำงานของระบบ เนื่องจากข้อมูลการสนทนานั้นเป็นความลับ จึงจำเป็นต้องมั่นใจว่าผู้ใช้ที่ทำการเรียกใช้งานระบบหลังบ้านหรือ API นั้น ต้องเป็นผู้ที่ได้รับอนุญาตให้ใช้งานเท่านั้น มิเช่นนั้นอาจเกิดการรั่วไหลของข้อมูล รวมถึงเกิดการแทรกแซงระบบจากผู้ไม่หวังดี โดยระบบการยืนยันตัวตนของระบบแชทสดสำหรับลูกค้าของบอทไอโอ สามารถแบ่งออกเป็น 2 ประเภท ดังนี้

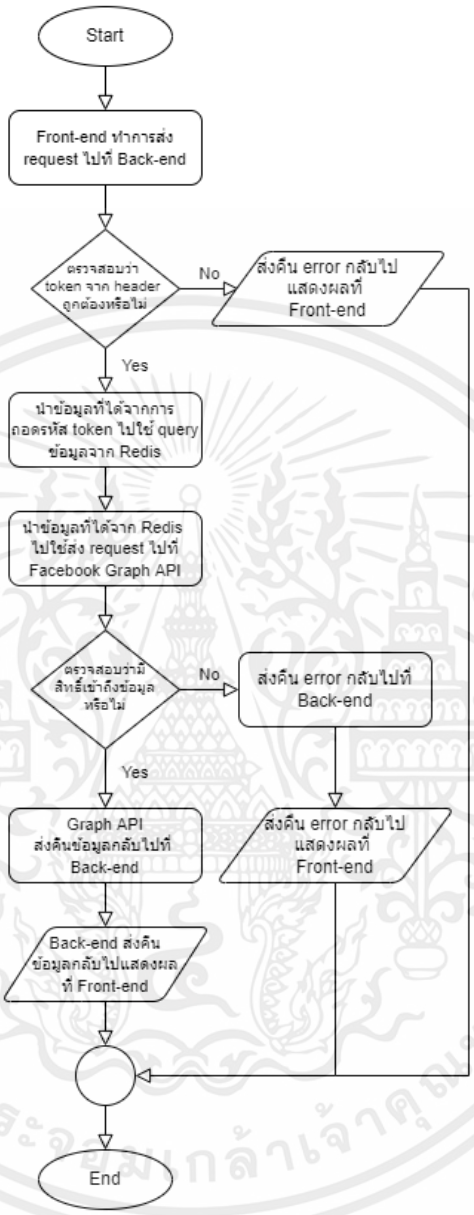
3.3.1 ระบบการยืนยันตัวตนเมื่อเรียกใช้งาน API



รูปที่ 3.16 บล็อกไดอะแกรมการทำงานของระบบยืนยันตัวตนเมื่อเรียกใช้งาน API

ระบบการยืนยันตัวตนเมื่อเรียกใช้งาน API จะใช้ JSON Web Token (JWT) ในการยืนยันตัวตน ดังรูปที่ 3.16 เป็นบล็อกไดอะแกรมการทำงานของระบบยืนยันตัวตนเมื่อเรียกใช้งาน API ประกอบด้วย ระบบหลังบ้านเป็นตัวกลางในการเชื่อมต่อ เมื่อส่วนหน้าบ้านจะทำการส่งคำร้องขอการใช้งาน API มาที่ระบบหลังบ้านจำเป็นต้องมีการส่งข้อมูล token ที่ใช้ในการยืนยันตัวตนมาในลักษณะ

ของ header จากนั้นระบบหลังบ้านจะนำ token ที่ได้มาถอดรหัสเพื่อนำไปตรวจสอบสิทธิ์ในการใช้งานโดยใช้ Facebook Graph API



รูปที่ 3.17 แผนผังการทำงานของการยืนยันตัวตนเมื่อเรียกใช้งาน API

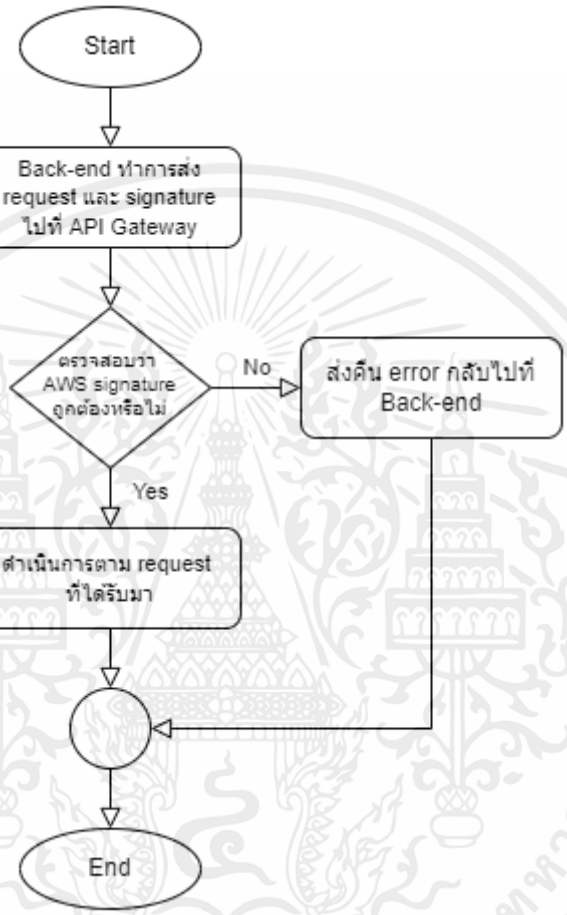
การทำงานของระบบแชทสดสำหรับลูกค้าของบอทไอโอในส่วนของการยืนยันตัวตนเมื่อเรียกใช้งาน API แสดงในรูปที่ 3.17 มีขั้นตอนการทำงานดังนี้

1. เริ่มต้นด้วยการที่ส่วนหน้าบ้านทำการส่งคำร้องขอไปยังระบบหลังบ้าน
2. ระบบหลังบ้านจะทำการตรวจสอบ token ที่ได้รับมาจาก header ว่ามีความถูกต้องหรือไม่

3. กรณีที่ token ไม่ถูกต้อง ระบบหลังบ้านจะทำการส่งคืน error กลับไปที่ส่วนหน้าบ้าน และจบการทำงาน
4. กรณีที่ token ถูกต้อง ระบบหลังบ้านจะนำข้อมูลที่ได้จากการถอดรหัส token ไปใช้ในการ query ข้อมูลผู้ใช้จาก Redis
5. จากนั้นระบบหลังบ้านจะนำข้อมูลที่ได้จาก Redis ไปใช้ในการส่งคำร้องขอไปที่ Facebook Graph API
6. Facebook Graph API จะทำการตรวจสอบข้อมูลว่าผู้ใช้งานมีสิทธิ์เข้าถึงข้อมูลที่ร้องขอหรือไม่
7. กรณีที่ผู้ใช้ไม่มีสิทธิ์เข้าถึง Facebook Graph API จะทำการส่งคืน error กลับไปที่ส่วนหลังบ้าน
8. จากนั้นระบบหลังบ้านจะทำการส่งคืน error กลับไปที่ส่วนหน้าบ้าน และจบการทำงาน
9. กรณีที่ผู้ใช้งานมีสิทธิ์เข้าถึง Facebook Graph API จะทำการส่งคืนข้อมูลกลับไปที่ระบบหลังบ้าน
10. จากนั้นระบบหลังบ้านจะส่งคืนข้อมูลกลับไปแสดงผลในส่วนหน้าบ้าน

3.3.2 ระบบบการยืนยันตัวตนเมื่อเรียกใช้งาน Amazon API Gateway

ระบบการยืนยันตัวตนเมื่อเรียกใช้งาน Amazon API Gateway จะต้องทำการสร้าง AWS Signature และแนบไปกับ header เพื่อใช้ในการส่งคำร้องขอเมื่อเรียกใช้งาน Amazon API Gateway



รูปที่ 3.18 แผนผังการทำงานของการยืนยันตัวตนเมื่อเรียกใช้งาน API Gateway

การทำงานของระบบแชทสดสำหรับลูกค้าของบอทไอโอในส่วนของการยืนยันตัวตนเมื่อเรียกใช้งาน API แสดงในรูปที่ 3.18 มีขั้นตอนการทำงานดังนี้

1. เริ่มต้นด้วยการที่ระบบหลังบ้านทำการส่งคำร้องขอไปที่ API Gateway และแนบ AWS signature ไปในส่วน header
2. API Gateway จะทำการตรวจสอบ AWS signature ว่าถูกต้องหรือไม่
3. กรณีที่ AWS signature ไม่ถูกต้อง API Gateway จะทำการส่งคืน error กลับไปที่ระบบหลังบ้าน
4. กรณีที่ AWS signature ถูกต้อง API Gateway จะดำเนินการตามคำร้องขอที่ได้รับมา

3.4 ฐานข้อมูล (Database)

จากขั้นตอนการทำงานของระบบการสนทนาของระบบแชทสดสำหรับลูกค้าของบอทไอโอในหัวข้อที่ 3.2 สามารถนำมาออกแบบฐานข้อมูลได้ 2 แบบ ดังนี้

3.4.1 ฐานข้อมูลแบบคีย์-ค่า (Key-Value Database)

connecti...	pageid	userid
Ksn1BeLMS...	395420184...	10162194455210324
Ksn2JcKvy...	102582565...	1022783324790773
Ksn5fdvwS...	102582565...	1022783324790773
Ksnr_fWQy...	102582565...	1022783324790773
JC9IFf_byQ...	102427815...	1022783324790773

รูปที่ 3.19 ตัวอย่างฐานข้อมูลสำหรับเก็บข้อมูลการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

ฐานข้อมูลสำหรับเก็บข้อมูลการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket) ดังรูปที่ 3.19 ประกอบไปด้วยรายละเอียดดังนี้

1. connectionId เป็นคีย์พาร์ทิชัน (Partition Key) ชนิด string ที่ใช้ในการเก็บค่า connectionId สำหรับใช้ส่งข้อมูลผ่านเว็บซ็อกเก็ต
2. pageid เป็นคีย์ชนิด string ที่ใช้ในการเก็บค่า id ของเพจที่เชื่อมต่อกับเว็บซ็อกเก็ต
3. userid เป็นคีย์ชนิด string ที่ใช้ในการเก็บค่า id ของผู้ใช้งานที่เชื่อมต่อกับเว็บซ็อกเก็ต

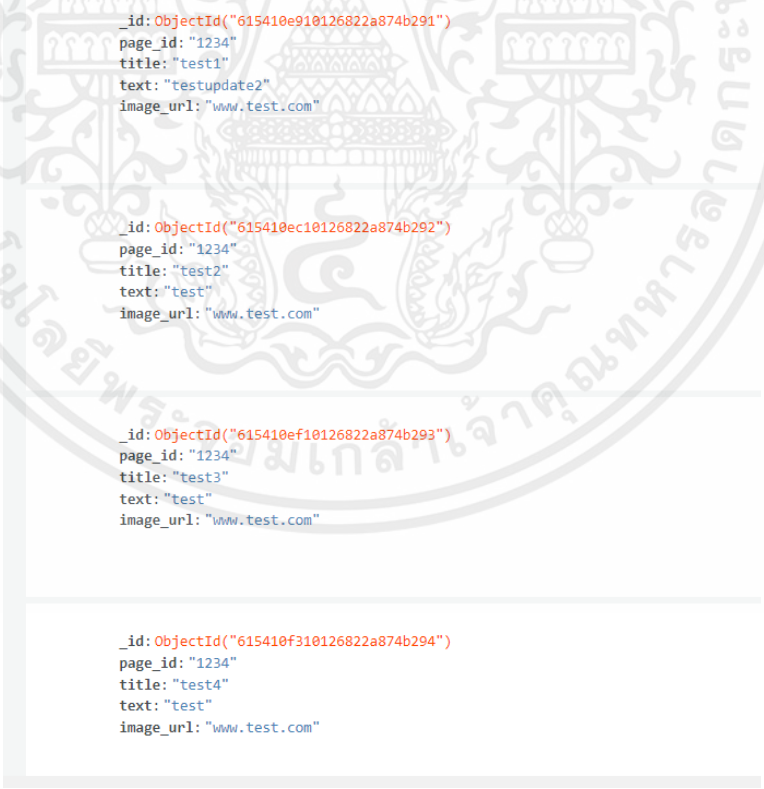
uuid	action	connecti...	ex_time ...	page_id	timestamp	user_id
7de827b2-...	disconnect	Ko6sOdbcy...	1640248614	100740101...	1639989414	2124671427686092
9c1890c9-5...	disconnect	KpS1Vdtby...	1640257921	158326827...	1639998721	107829654681865
bc91cb0f-3...	connect	KoQ25fE5y...	1640230879	100740101...	1639971679	10162194455210324
597f8a19-6...	connect	Ksl-WfBwy...	1640344386	102109478...	1640085186	2510299029223235
b849f0e9-5...	disconnect	Kse8xd09S...	1640341572	102109478...	1640082372	2510299029223235
a51055b9-...	connect	KsjbLcmny...	1640343341	102109478...	1640084141	2510299029223235

รูปที่ 3.20 ตัวอย่างฐานข้อมูลสำหรับเก็บข้อมูลประวัติการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

ฐานข้อมูลสำหรับเก็บข้อมูลประวัติการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket) ดังรูปที่ 3.20 ประกอบไปด้วยรายละเอียด ดังนี้

1. uuid เป็นคีย์พาท์ทิชัน (Partition Key) ชนิด string ที่ใช้ในการเก็บค่า Unique Key
2. action เป็นคีย์ชนิด string ที่ใช้ในการเก็บค่า action ว่าเป็นการ connect หรือ disconnect
3. connectionId เป็นคีย์ชนิด string ที่ใช้ในการเก็บค่า connectionId สำหรับใช้ส่งข้อมูลผ่านเว็บซ็อกเก็ต
4. ex_time เป็นคีย์ชนิด timestamp ที่ใช้ในการเก็บค่า Time to Live (TTL) เพื่อเป็นตัวกำหนดระยะเวลาที่ข้อมูลจะถูกลบออกจากฐานข้อมูล
5. pageid เป็นคีย์ชนิด string ที่ใช้ในการเก็บค่า id ของเพจที่เชื่อมต่อกับเว็บซ็อกเก็ต
6. timestamp เป็นคีย์ชนิด timestamp ที่ใช้ในการเก็บค่าเวลา Unix
7. userid เป็นคีย์ชนิด string ที่ใช้ในการเก็บค่า id ของผู้ใช้งานที่เชื่อมต่อกับเว็บซ็อกเก็ต

3.4.2 ฐานข้อมูลแบบเอกสาร (Document Database)



รูปที่ 3.21 ตัวอย่างฐานข้อมูลสำหรับเก็บข้อมูลข้อความแบบ Template

ฐานข้อมูลสำหรับเก็บข้อมูลข้อความแบบ Template ดังรูปที่ 3.21 ประกอบไปด้วยรายละเอียด ดังนี้

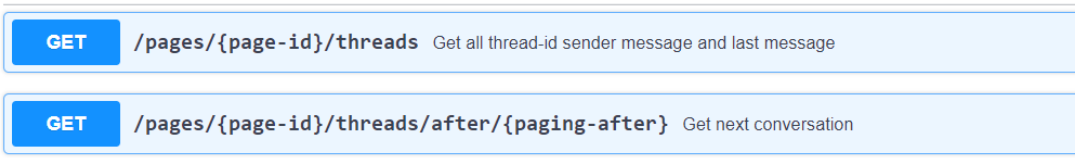
1. _id เป็น Unique Key สำหรับเก็บข้อมูล id ของแต่ละ document
2. page_id เป็นคีย์ชนิด string ที่ใช้ในการเก็บข้อมูลว่าข้อความแบบ template แต่ละ template เป็นของเพจไหน
3. title เป็นคีย์ชนิด string ที่ใช้ในการเก็บข้อมูลหัวเรื่องของ template
4. text เป็นคีย์ชนิด string ที่ใช้ในการเก็บข้อมูลข้อความของ template
5. image_url เป็นคีย์ชนิด string ที่ใช้ในการเก็บข้อมูลที่อยู่ของรูปภาพ

3.5 เอกสารอธิบาย API (API Document)

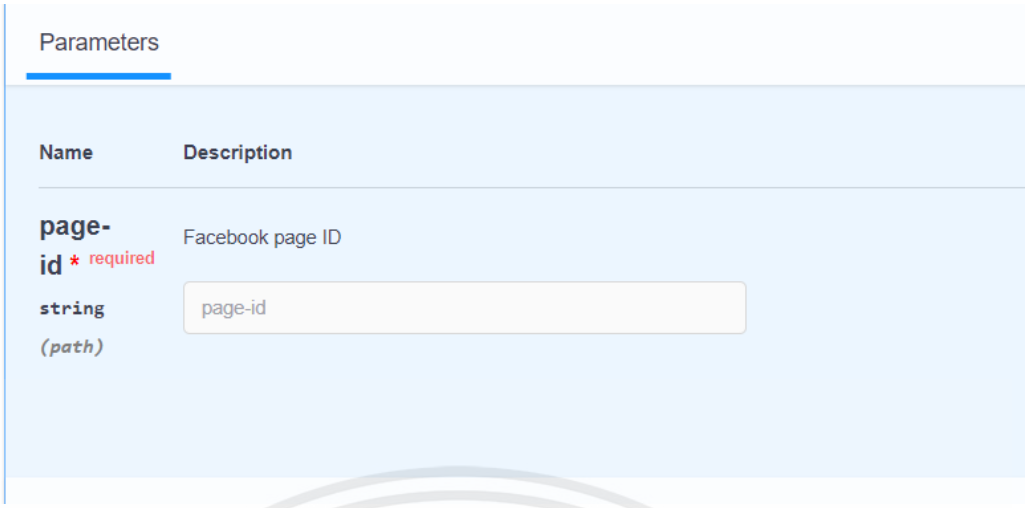
เอกสารอธิบาย API (API Document) เป็นเอกสารที่ช่วยในการสื่อสารระหว่างส่วนหน้าบ้านและหลังบ้าน ให้มีความสะดวก และเข้าใจง่ายขึ้น

รูปที่ 3.22 ตัวอย่างการสร้าง API Document ด้วย SwaggerHub

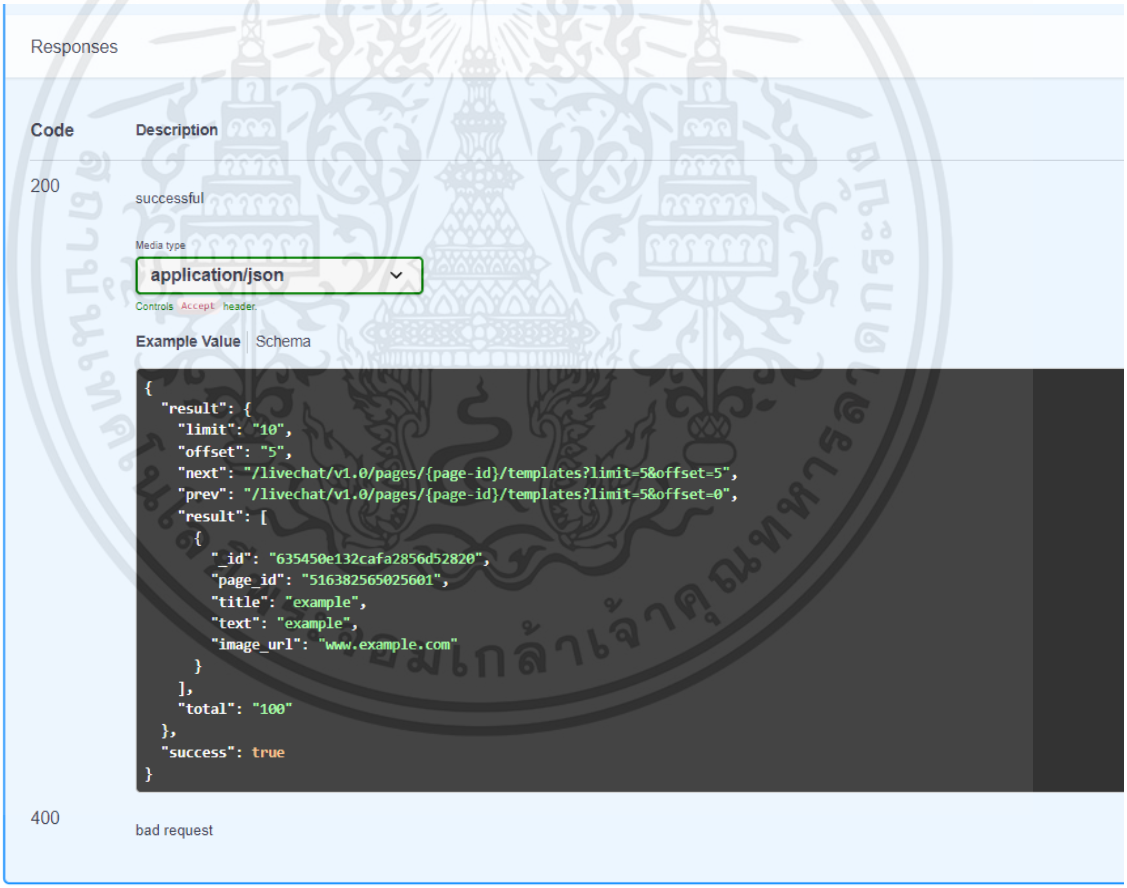
เอกสารอธิบาย API (API Document) ของระบบแชทสดสำหรับลูกค้าของบอทไอโอ จะประกอบไปด้วยคำอธิบาย URL endpoint, method, path, parameters และ response



รูปที่ 3.23 ตัวอย่างคำอธิบาย method และ path



รูปที่ 3.24 ตัวอย่างคำอธิบาย parameters



รูปที่ 3.25 ตัวอย่าง response

บทที่ 4
การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลอง คือ ในบทนี้จะกล่าวถึงการทดลองการทำงานของระบบหลังบ้าน ถึงขั้นตอนการทำงานตามหัวข้อดังนี้ การเชื่อมต่อเว็บซ็อกเก็ต การส่งข้อความผ่านเว็บซ็อกเก็ต การใช้งานเว็บฮุค และการเรียกใช้งาน API โดยมีรายละเอียดดังนี้

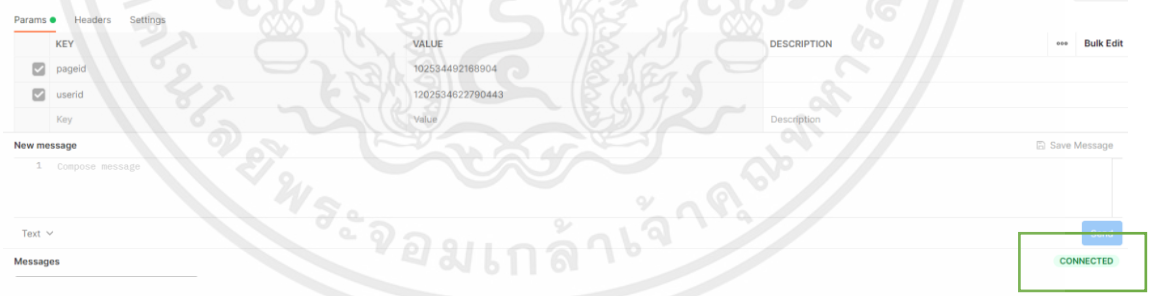
4.1 การทดสอบการทำงานของระบบหลังบ้าน (Back-end)

การทดสอบนี้เป็นการทดสอบเพื่อตรวจสอบคุณสมบัติและการทำงานของระบบแชทสดสำหรับลูกค้าของบอทไอโอ ประกอบด้วย การเชื่อมต่อเว็บซ็อกเก็ต (WebSocket) การส่งข้อความผ่านเว็บซ็อกเก็ต (WebSocket) การใช้งานเว็บฮุค (Webhook) และการเรียกใช้งาน API เพื่อทดสอบการทำงานของระบบแชทสดสำหรับลูกค้าของบอทไอโอ

มีขั้นตอนและผลการทดสอบการทำงานของระบบหลังบ้านดังนี้

4.1.1 การเชื่อมต่อเว็บซ็อกเก็ต

ในการทดลองนี้เป็นการแสดงผลการทำงานของการเชื่อมต่อกับเว็บซ็อกเก็ต โดยมีรายละเอียดการทดลองดังนี้

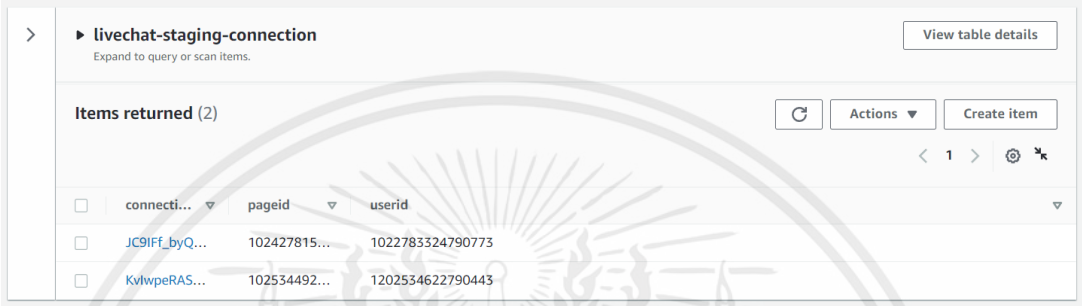


รูปที่ 4.1 ตัวอย่างการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

จากรูปที่ 4.1 เป็นการทดสอบการเชื่อมต่อกับเว็บซ็อกเก็ตด้วยโปรแกรม Postman โดยทำการเชื่อมต่อกับ API Gateway endpoint (wss://xxxxx.amazon.com/xxx) และส่ง Parameter สองตัว ได้แก่ pagid และ userid เพื่อเป็นการบอกกับระบบว่าเพจใด และผู้ใช้งานระบบคนใดเป็นผู้เชื่อมต่อกับเว็บซ็อกเก็ต โดยสามารถสังเกตได้จากสถานะการเชื่อมต่อ (Connected) ดังกรอบสี่เหลี่ยมด้านล่าง

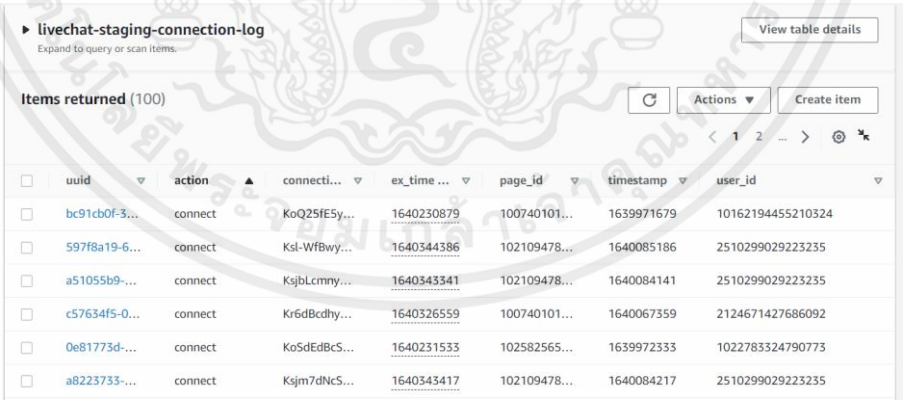
หลังจากทำการเชื่อมต่อเว็บซ็อกเก็ต ในรูปที่ 4.1 ระบบจะทำการเก็บข้อมูลการเชื่อมต่อเว็บ-ซ็อกเก็ตลงในฐานข้อมูลสำหรับเก็บข้อมูลการเชื่อมต่อดังรูปที่ 4.2 โดยยกตัวอย่างข้อมูลที่แสดงรายละเอียดดังนี้

1. connectionId เช่น KvJXzeiPyQ0CH1g= เป็นต้น
2. pageid เช่น 102534492168904 เป็นต้น
3. userid เช่น 1202534622790443 เป็นต้น



รูปที่ 4.2 ผลลัพธ์การเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

หลังจากทำการเก็บข้อมูลการเชื่อมต่อเว็บซ็อกเก็ตลงในฐานข้อมูลสำหรับเก็บข้อมูลการเชื่อมต่อเรียบร้อยแล้ว ระบบจะทำการเก็บประวัติการเชื่อมต่อเว็บซ็อกเก็ตลงในฐานข้อมูลสำหรับเก็บประวัติการเชื่อมต่อดังรูปที่ 4.3

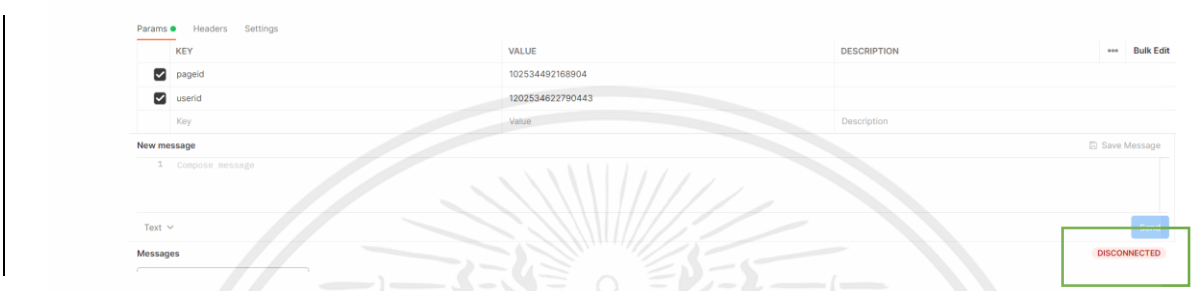


รูปที่ 4.3 ผลลัพธ์การเก็บประวัติการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

จากรูปที่ 4.3 ระบบจะทำการเก็บประวัติการเชื่อมต่อเว็บซ็อกเก็ตลงในฐานข้อมูลสำหรับเก็บประวัติการเชื่อมต่อ โดยยกตัวอย่างข้อมูลที่แสดงรายละเอียดดังนี้

1. uuid เช่น bc91cb0f-3d40-5813-5128-116e893e3c0a เป็นต้น

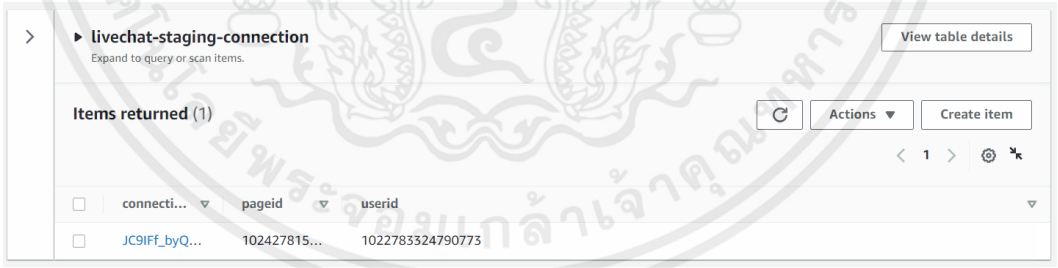
2. action เช่น connect เป็นต้น
3. connectionId เช่น KoQ25fE5yQ0CFOg= เป็นต้น
4. ex_time เช่น 1640230879 เป็นต้น
5. page_id เช่น 102534492168904 เป็นต้น
6. timestamp เช่น 1639971679 เป็นต้น
7. user_id เช่น 1202534622790443 เป็นต้น



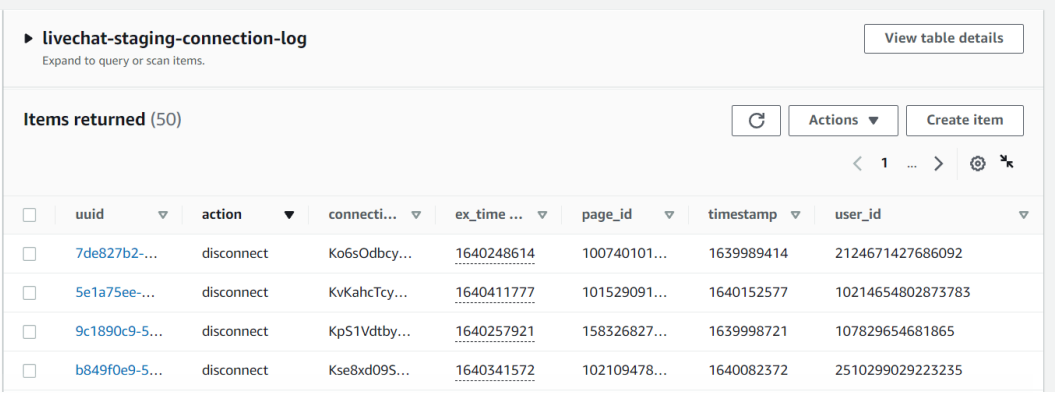
รูปที่ 4.4 ตัวอย่างการตัดการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

จากรูปที่ 4.4 เป็นการทดสอบการตัดการเชื่อมต่อเว็บซ็อกเก็ตด้วยโปรแกรม Postman โดยสามารถสังเกตได้จากสถานะการตัดการเชื่อมต่อ (Disconnected) ดังกรอบสี่เหลี่ยมด้านล่างของรูป

หลังจากทำการตัดการเชื่อมต่อเว็บซ็อกเก็ตในรูปที่ 4.5 ระบบจะทำการลบข้อมูลการเชื่อมต่อเว็บซ็อกเก็ตออกจากฐานข้อมูลสำหรับเก็บข้อมูลการเชื่อมต่อ ดังรูปที่ 4.6



รูปที่ 4.5 ผลลัพธ์การตัดการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)



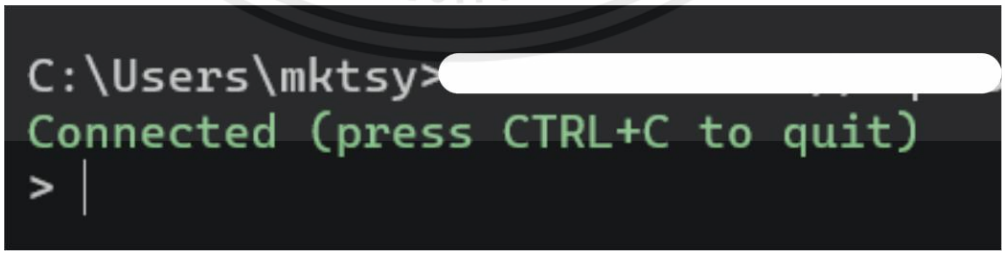
รูปที่ 4.6 ผลลัพธ์การเก็บประวัติการตัดการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

จากรูปที่ 4.6 ระบบจะทำการเก็บประวัติการตัดการเชื่อมต่อเว็บซ็อกเก็ตลงในฐานข้อมูลสำหรับเก็บประวัติการเชื่อมต่อ โดยยกตัวอย่างข้อมูลที่แสดงรายละเอียดดังนี้

1. uuid เช่น bc91cb0f-3d40-5813-5128-116e893e3c0a เป็นต้น
2. action เช่น disconnect เป็นต้น
3. connectionId เช่น KoQ25fE5yQ0CFOg= เป็นต้น
4. ex_time เช่น 1640230879 เป็นต้น
5. page_id เช่น 102534492168904 เป็นต้น
6. timestamp เช่น 1639971679 เป็นต้น
7. user_id เช่น 1202534622790443 เป็นต้น

4.1.2 การส่งข้อความผ่านเว็บซ็อกเก็ต

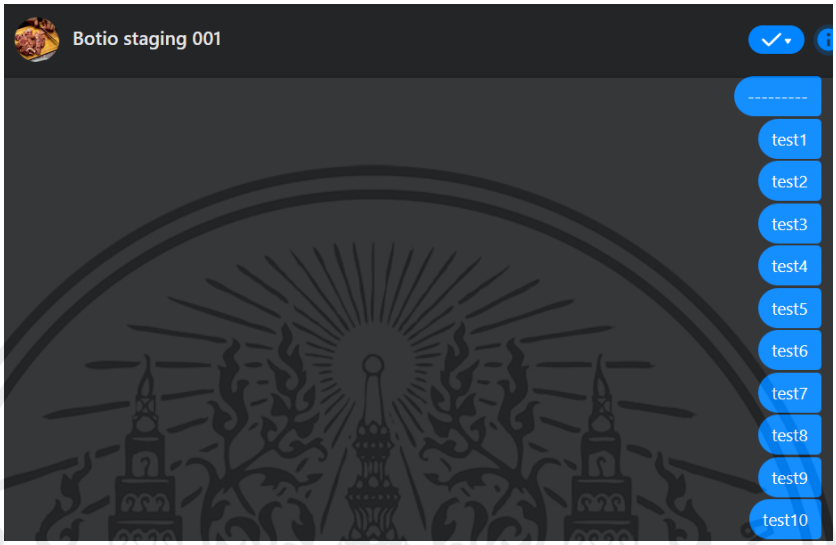
ในการทดลองนี้เป็นการแสดงผลการทำงานของการส่งข้อความผ่านเว็บซ็อกเก็ต โดยมีรายละเอียดการทดลองดังนี้



รูปที่ 4.7 ตัวอย่างการเชื่อมต่อเว็บซ็อกเก็ต (WebSocket)

จากรูปที่ 4.7 เป็นการเริ่มต้นการทดสอบโดยการเชื่อมต่อกับเว็บซ็อกเก็ตด้วย WebSocket Client (WSCAT) เข้ากับเพจ Facebook ที่ต้องการจะทดสอบ

หลังจากทำการเชื่อมต่อเว็บซ็อกเก็ต จะทำการส่งข้อความไปที่ inbox ของเพจ Facebook ที่ต้องการจะทดสอบดังรูปที่ 4.8



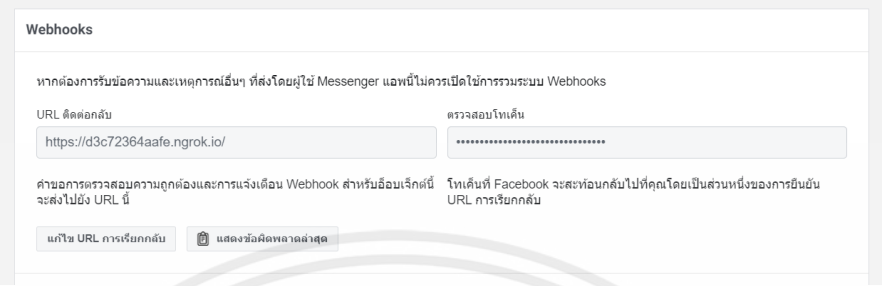
รูปที่ 4.8 ตัวอย่างการส่งความไปที่ inbox ของเพจ Facebook

รูปที่ 4.9 ผลลัพธ์การส่งข้อความผ่านเว็บซ็อกเก็ต (WebSocket)

หลังจากที่ได้รับข้อความจากเพจ Facebook ระบบจะทำการส่งข้อความที่ได้รับมาจากเพจ Facebook ผ่านเว็บซ็อกเก็ตเพื่อนำไปแสดงผล โดยระบบจะส่งข้อความผ่านเว็บซ็อกเก็ตไปที่ connectionId ทั้งหมดที่เชื่อมต่อเว็บซ็อกเก็ตอยู่ดังรูปที่ 4.9

4.1.3 การใช้งานเว็บฮุค (Webhook)

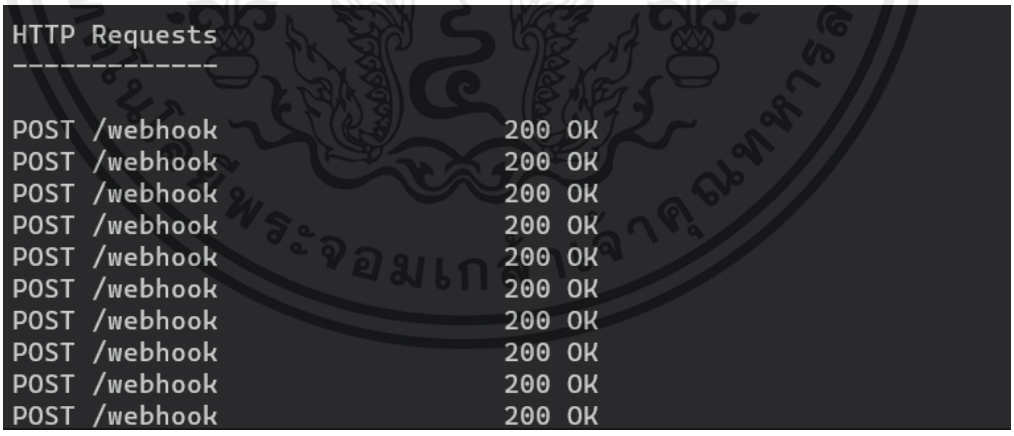
ในการทดลองนี้เป็นการแสดงผลการทำงานของการใช้งานเว็บฮุค โดยมีรายละเอียดการทดลองดังนี้



รูปที่ 4.10 ตัวอย่างการตั้งค่าเว็บฮุค (Webhook)

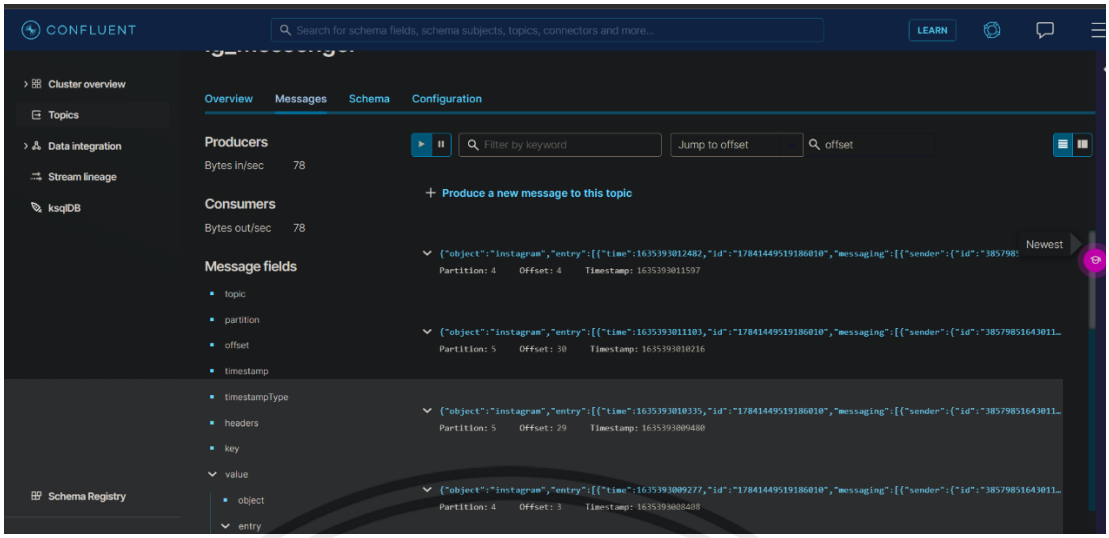
จากรูปที่ 4.10 เป็นการตั้งค่าการรับเว็บฮุคบนแอปพลิเคชัน Facebook สำหรับผู้พัฒนา โดยจะประกอบไปด้วย URL ติดต่อกลับ และโทเค็นที่ใช้ในการตรวจสอบ

หลังจากที่ทำการตั้งค่าการรับเว็บฮุคบนแอปพลิเคชัน Facebook สำหรับผู้พัฒนาเรียบร้อยแล้ว จะเป็นการทดสอบการส่งเว็บฮุค โดยจะเป็นการส่ง HTTP POST กลับมาที่ URL ติดต่อกลับดังรูปที่ 4.11



รูปที่ 4.11 ผลลัพธ์การใช้งานเว็บฮุค (Webhook)

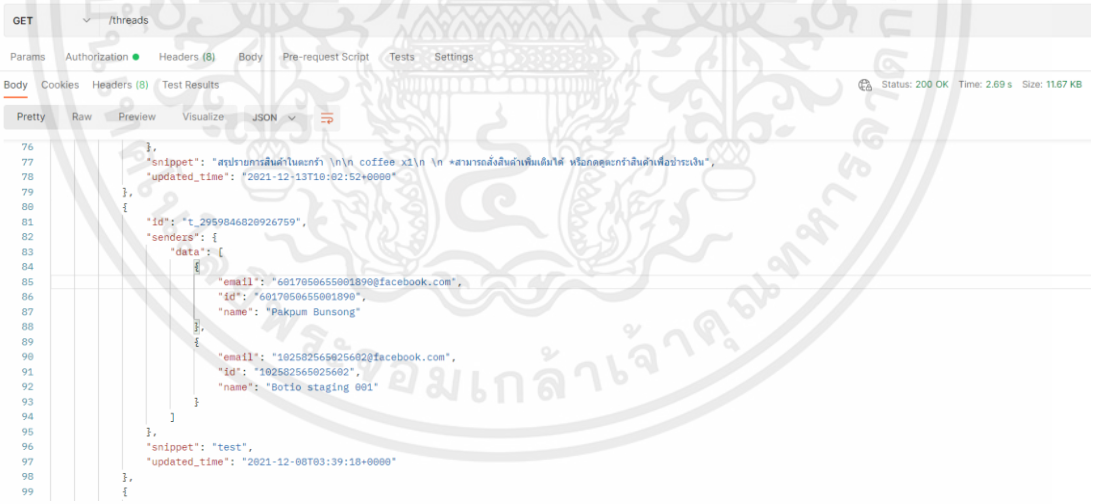
จากนั้นเมื่อได้รับเว็บฮุค จะทำการทดสอบสร้างผู้ผลิต (Producer) เพื่อใช้ในการส่งข้อความที่ได้รับจากเว็บฮุคไปที่ Apache Kafka ดังรูปที่ 4.12



รูปที่ 4.12 ผลลัพธ์การผลิตข้อความลง Apache Kafka

4.1.4 การเรียกใช้งาน API

ในการทดลองนี้เป็นการแสดงผลการทำงานของการเรียกใช้งาน API โดยมีรายละเอียดการทดลองดังนี้

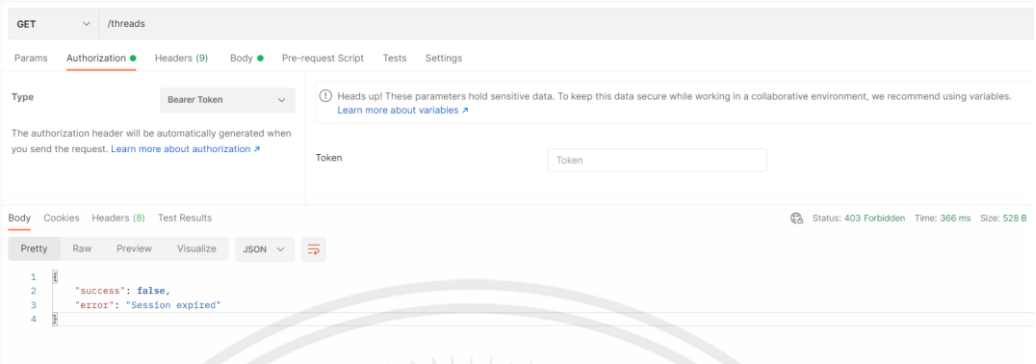


รูปที่ 4.13 ผลลัพธ์การเรียกใช้งาน API ดึงข้อมูลการสนทนาทั้งหมด

จากรูปที่ 4.13 เป็นการทดสอบการเรียกใช้งาน API ด้วยโปรแกรม Postman ในการดึงข้อมูลการสนทนาทั้งหมด โดยยกตัวอย่างข้อมูลที่แสดงรายละเอียดดังนี้

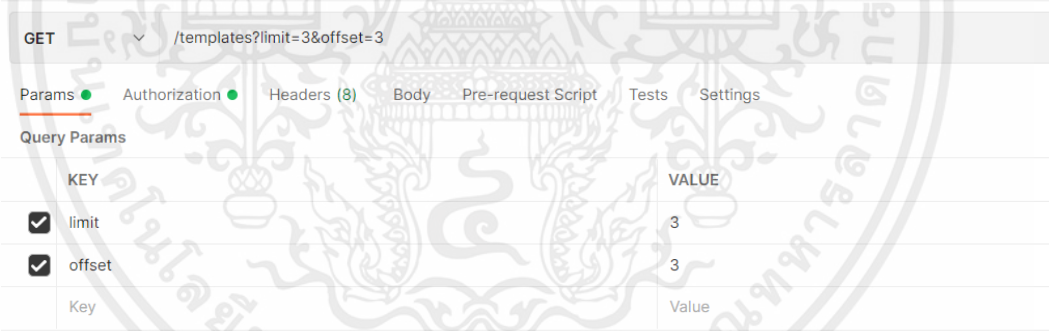
1. id เช่น t_2959846820926759 เป็นต้น
2. sender จะประกอบด้วย email, id และ name ของผู้ส่งและผู้รับ

3. snippet เช่น test เป็นต้น
4. updated_time 2021-12-08T03:39:18+0000 เป็นต้น



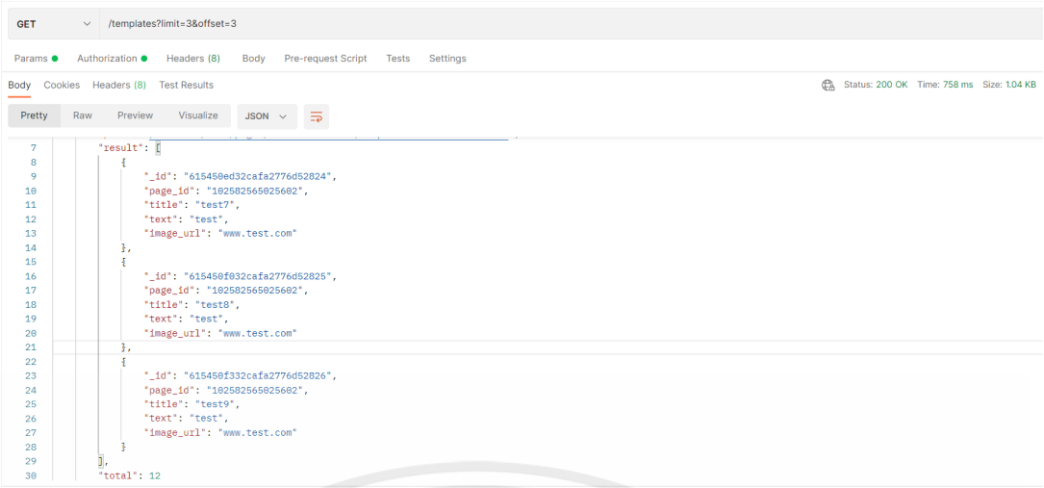
รูปที่ 4.14 ผลลัพธ์การเรียกใช้งาน API ในกรณีที่ไม่มี token หรือ token ไม่ถูกต้อง

จากรูปที่ 4.14 เป็นการทดลองการเรียกใช้งาน API ในกรณีที่ไม่มี token หรือ token ที่ส่งมาไม่ถูกต้อง ระบบจะทำการคืนค่า error กลับไปยังผู้ที่เรียกใช้งาน API



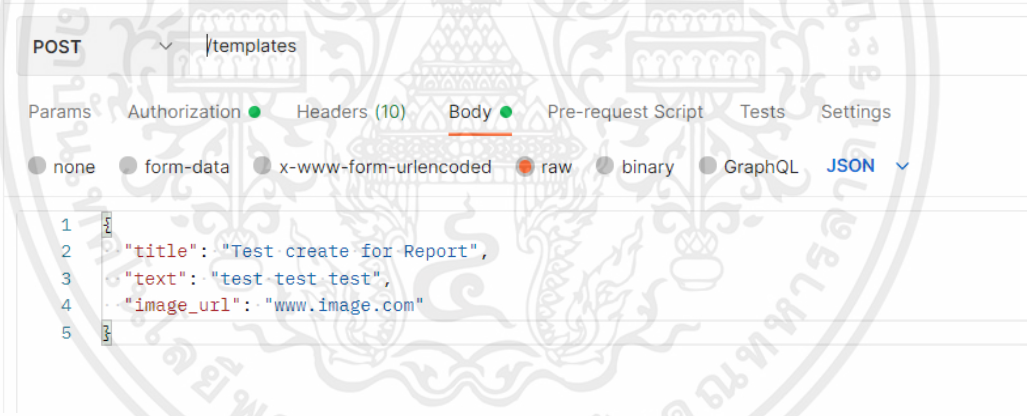
รูปที่ 4.15 ตัวอย่างการเรียกใช้งาน API ในการดึงข้อมูลข้อความแบบ Template

จากรูปที่ 4.15 เป็นการทดสอบการเรียกใช้งาน API ในการดึงข้อมูลข้อความแบบ Template โดยการส่ง parameters limit และ offset เพื่อเป็นการกำหนดจำนวนข้อมูลที่ API จะส่งคืนกลับมา



รูปที่ 4.16 ผลลัพธ์การเรียกใช้งาน API ดึงข้อมูลข้อความแบบ Template

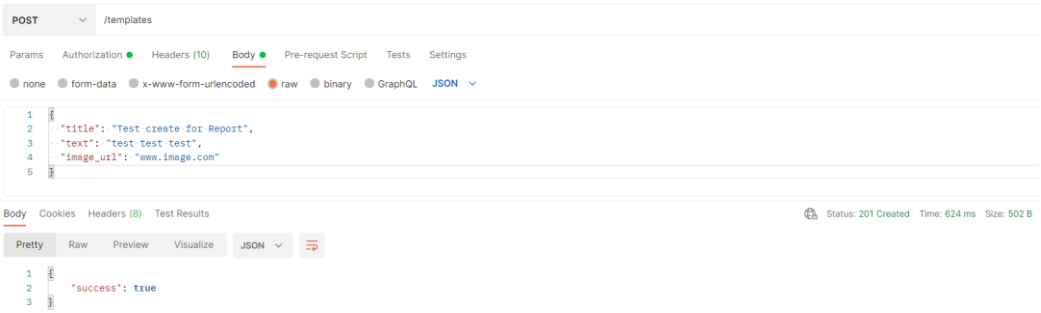
จากรูปที่ 4.16 เป็นผลลัพธ์ของการเรียกใช้งาน API ดึงข้อมูลข้อความแบบ Template โดยการกำหนดค่า limit และ offset ไว้ที่ 3 ระบบจึงมีการส่งคืนข้อมูลกลับมา 3 ชุด ดังรูปที่ 4.17



รูปที่ 4.17 ตัวอย่างการเรียกใช้งาน API ในการสร้างข้อความแบบ Template

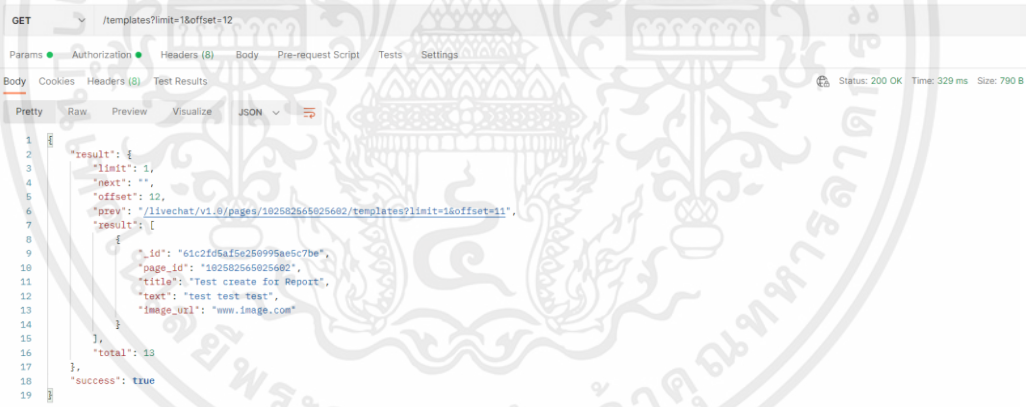
จากรูปที่ 4.17 เป็นการทดสอบการเรียกใช้งาน API ในการสร้างข้อความแบบ Template ด้วย HTTP method POST โดยการส่ง body ไปในลักษณะของ JSON โดยยกตัวอย่างข้อมูลที่แสดงรายละเอียดดังนี้

1. title เช่น Test create for Report เป็นต้น
2. text เช่น test test test เป็นต้น
3. image_url เช่น www.image.com เป็นต้น

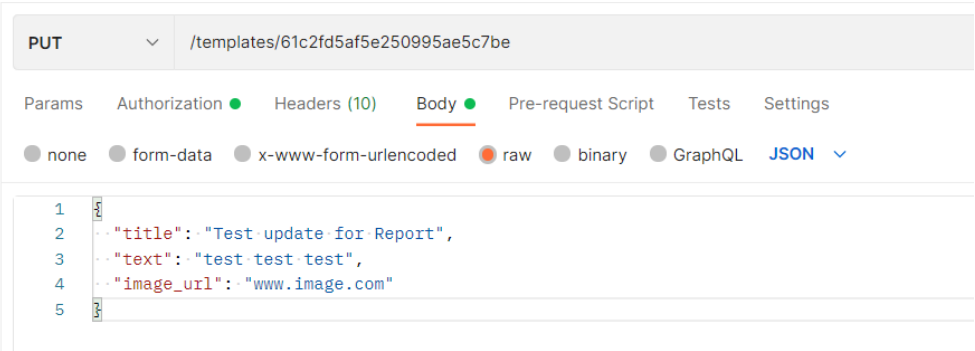


รูปที่ 4.18 ผลลัพธ์การเรียกใช้งาน API ในการสร้างข้อความแบบ Template

จากรูปที่ 4.18 ผลลัพธ์ของการเรียกใช้งาน API ในการสร้างข้อความแบบ Template เมื่อคำร้องขอสำเร็จ ระบบจะทำการส่งคืนข้อมูลสถานะการสร้างกลับมา และเมื่อเรียกใช้งาน API สำหรับการดึงข้อมูลข้อความแบบ Template อีกครั้ง จะเห็นได้ว่าระบบได้ทำการสร้างข้อความแบบ Template ขึ้นมาเพิ่มดังรูปที่ 4.19

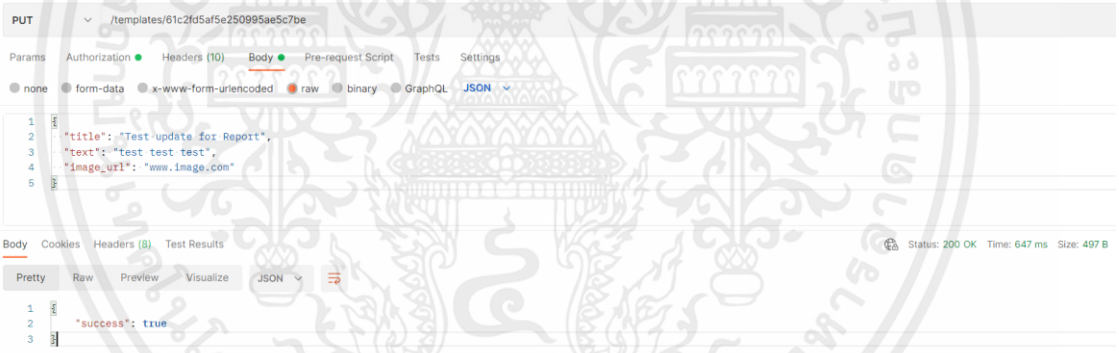


รูปที่ 4.19 ผลลัพธ์การเรียกใช้งาน API ในการดึงข้อมูลข้อความแบบ Template



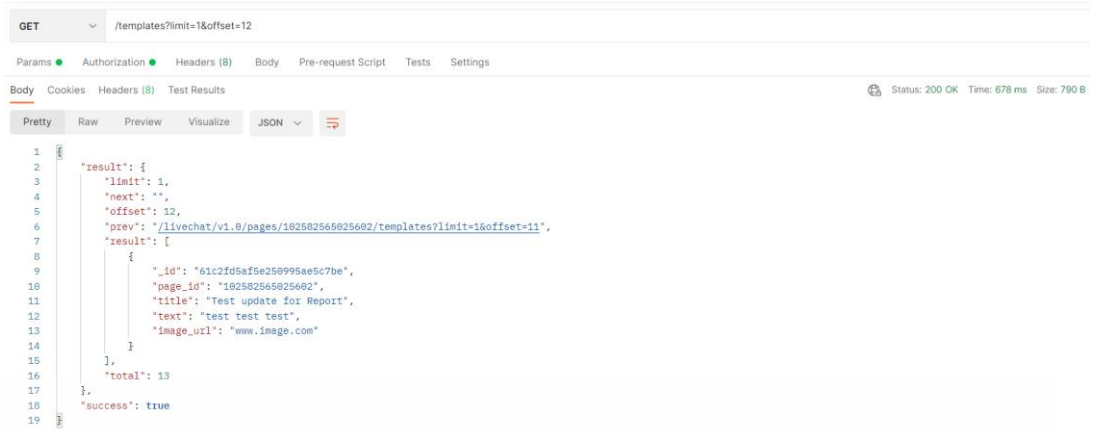
รูปที่ 4.20 ตัวอย่างการเรียกใช้งาน API ในการแก้ไขข้อความแบบ Template

จากรูปที่ 4.20 เป็นการทดสอบการเรียกใช้งาน API ในการแก้ไขข้อความแบบ Template ด้วย HTTP method PUT โดยการส่ง body ไปในลักษณะเดียวกับการเรียกใช้งาน API ในการสร้างข้อความแบบ Template แต่จะเพิ่ม id ของข้อความแบบ Template ที่ต้องการจะแก้ไขไปใน path ของ URL endpoint

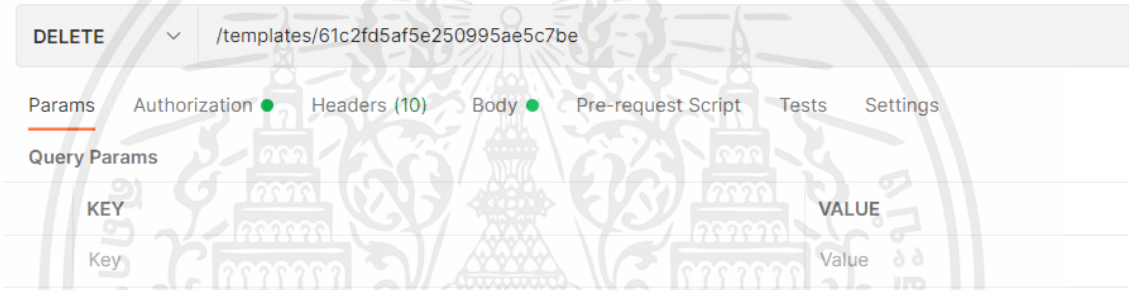


รูปที่ 4.21 ผลลัพธ์การเรียกใช้งาน API ในการแก้ไขข้อความแบบ Template

จากรูปที่ 4.21 ผลลัพธ์ของการเรียกใช้งาน API ในการแก้ไข้ข้อความแบบ Template เมื่อคำร้องขอสำเร็จ ระบบจะทำการส่งคืนข้อมูลสถานะการแก้ไขกลับมา และเมื่อเรียกใช้งาน API สำหรับการดึงข้อมูลข้อความแบบ Template อีกครั้ง จะเห็นได้ว่าระบบได้ทำการแก้ไขข้อความแบบ Template ตาม id ที่อยู่ใน path ของ URL endpoint ดังรูปที่ 4.22

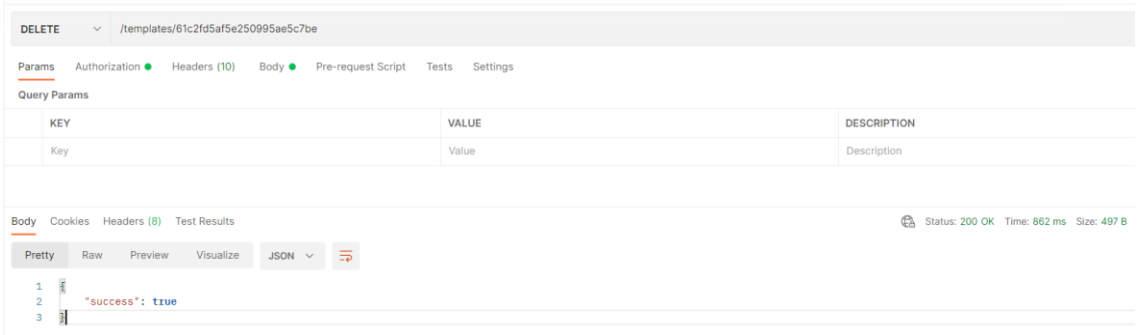


รูปที่ 4.22 ผลลัพธ์การเรียกใช้งาน API ในการดึงข้อมูลข้อความแบบ Template



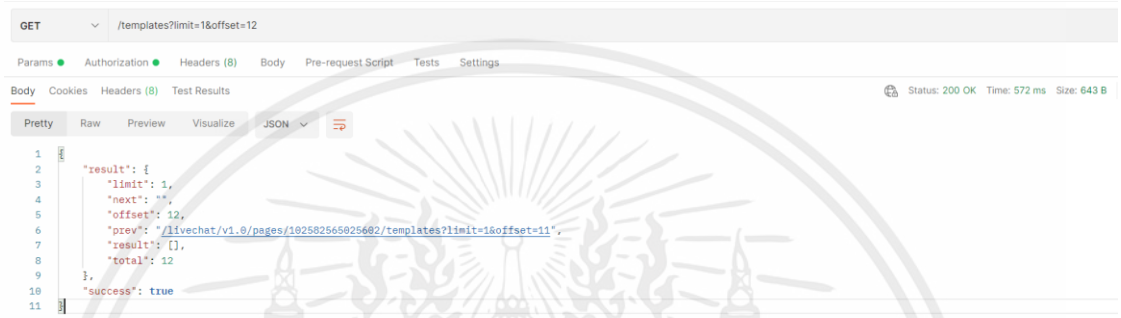
รูปที่ 4.23 ตัวอย่างการเรียกใช้งาน API ในการแก้ไขข้อความแบบ Template

จากรูปที่ 4.23 เป็นการทดสอบการเรียกใช้งาน API ในการลบข้อความแบบ Template ด้วย HTTP method DELETE โดยการเพิ่ม id ของข้อความแบบ Template ที่ต้องการจะลบไปใน path ของ URL endpoint



รูปที่ 4.24 ผลลัพธ์การเรียกใช้งาน API ในการลบข้อความแบบ Template

จากรูปที่ 4.24 ผลลัพธ์ของการเรียกใช้งาน API ในการลบข้อความแบบ Template เมื่อคำร้องขอสำเร็จ ระบบจะทำการส่งคืนข้อมูลสถานะการลบกลับมา และเมื่อเรียกใช้งาน API สำหรับการดึงข้อมูลข้อความแบบ Template อีกครั้ง จะเห็นได้ว่าระบบได้ทำการลบข้อความแบบ Template ตาม id ที่อยู่ใน path ของ URL endpoint สังเกตได้จากจำนวน total ที่ลดลงดังรูปที่ 4.25



รูปที่ 4.25 ผลลัพธ์การเรียกใช้งาน API ในการดึงข้อมูลข้อความแบบ Template

บทที่ 5
สรุปผลการทดลองและข้อเสนอแนะ

ในบทนี้เป็นการสรุปผลการทดลอง ปัญหาและอุปสรรคในการทำงาน และข้อเสนอแนะต่าง ๆ ซึ่งมีรายละเอียดดังนี้

5.1 สรุปผลการทดลอง

จากการดำเนินงานโครงงานสหกิจศึกษาในการสร้างระบบแชทสดสำหรับลูกค้าของบอทไอโอ ได้ทำระบบการสนทนา การยืนยันตัวตน รวมถึงระบบข้อความแบบ Template เพื่อช่วยในการสนับสนุนการสร้างบริการแชทสดของบอทไอโอให้สามารถใช้งานระบบแชทได้ผ่านเว็บไซต์ของบอทไอโอ ช่วยลดปัญหาที่เกิดความยุ่งยากและซับซ้อนกับลูกค้าของบอทไอโอ ทำให้ได้เกิดความสะดวกและรวดเร็วในการทำงานมากขึ้น

5.2 ปัญหาและอุปสรรค

1. ต้องใช้ความรู้ความชำนาญในด้านการพัฒนา API ฐานข้อมูล การใช้งานบริการของ Amazon Web Service (AWS) รวมถึงบริการอื่น ๆ ที่เกี่ยวข้อง ทำให้ต้องใช้ระยะเวลาในการศึกษาเพิ่มเติม

2. ไม่สามารถเข้าไปทำงานที่บริษัทได้ เนื่องจากสถานการณ์โควิดในปัจจุบัน ทำให้มีบางครั้งที่การสื่อสารภายในบริษัทเกิดข้อผิดพลาด

3. หากนักพัฒนาในส่วนหน้าบ้านต้องการใช้งาน API จำเป็นต้องสอบถามทางผู้จัดทำเพื่อความเข้าใจในการใช้งาน API

5.3 วิธีการแก้ไขปัญหา

1. ผู้จัดทำได้ศึกษาขั้นตอนการพัฒนา API ฐานข้อมูล รวมถึงบริการต่าง ๆ ที่เกี่ยวข้องอย่างละเอียดเพื่อนำมาออกแบบและพัฒนาระบบให้สอดคล้องกับการใช้งานจริง

2. เนื่องจากสถาณการณ์โควิดในปัจจุบัน ทำให้ไม่สามารถเข้าไปทำงานที่บริษัทได้ จึงจำเป็นต้องนัดประชุมออนไลน์กันภายในบริษัทเพื่อความเข้าใจที่ตรงกัน

3. ผู้จัดทำได้จัดทำเอกสารอธิบาย API (API Document) ในการอธิบายขั้นตอน และองค์ประกอบต่าง ๆ ในการใช้งาน API เพื่อให้นักพัฒนาในส่วนหน้าบ้าน หรือผู้อื่นที่ต้องการใช้งาน API ได้เข้าใจวิธีการใช้งาน

5.4 ข้อเสนอแนะ

1. ควรมีการแก้ไขปัญหาระยะเวลาที่ใช้ในการดึงข้อมูลการสนทนา
2. ควรเพิ่มการส่งสติ๊กเกอร์ในส่วนของ send API

เอกสารอ้างอิง

[1] บทความเกี่ยวกับเว็บแอปพลิเคชัน, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://www.mindphp.com/3664-web-application.html

[2] บทความเกี่ยวกับ Application Programming Interface, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://www.mulesoft.com/resources/api/what-is-an-api

[3] บทความเกี่ยวกับ Application Programming Interface, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://www.thaibulksms.com/blog/post/what-is-an-api-explain-like-someone-who-do-not-know-about-it/

[4] บทความเกี่ยวกับ Visual Studio Code โปรแกรม text editor, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://code.visualstudio.com/docs/editor/whyvscode

[5] บทความเกี่ยวกับโปรแกรม Postman, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://www.encora.com/insights/what-is-postman-api-test

[6] บทความเกี่ยวกับโปรแกรม Postman, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://www.digitalcrafts.com/blog/student-blog-what-postman-and-why-use-it

[7] บทความเกี่ยวกับ Golang ภาษาคอมพิวเตอร์, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://betterprogramming.pub/i-mastered-golang-basics-by-solving-these-15-project-euler-problems-1254a3897cf8

[8] บทความเกี่ยวกับ Golang ภาษาคอมพิวเตอร์, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://dev.to/centrilliontech/golang-101-21ko

[9] บทความเกี่ยวกับ Golang ภาษาคอมพิวเตอร์, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://blog.skooldio.com/what-is-golang/

[10] บทความเกี่ยวกับโปรแกรม Docker, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://aws.amazon.com/th/docker/

[11] บทความเกี่ยวกับโปรแกรม Kubernetes, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://blog.openlandscape.cloud/what-is-kubernetes

[12] บทความเกี่ยวกับโปรแกรม Kubernetes, [ออนไลน์], ค้นหาเมื่อ 22 ธันวาคม 2564, จาก https://aws.amazon.com/th/eks/features/

เอกสารอ้างอิง (ต่อ)

[13] บทความเกี่ยวกับ Amazon Web Service, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://aws.amazon.com/th/what-is-aws/

[14] บทความเกี่ยวกับ Amazon Elastic Kubernetes Service, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://aws.amazon.com/th/eks/

[15] บทความเกี่ยวกับ Amazon API Gateway, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://aws.amazon.com/th/api-gateway/features/

[16] บทความเกี่ยวกับ Amazon Web Service Lambda, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

[17] บทความเกี่ยวกับ Amazon DynamoDB, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/

[18] บทความเกี่ยวกับโปรแกรม Redis, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://redis.io/docs/about/

[19] บทความเกี่ยวกับโปรแกรม Redis, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://www.borntodev.com/2020/07/14/redis-101/

[20] บทความเกี่ยวกับ WebSocket Protocol, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://www.jittagornp.me/blog/what-is-websocket/

[21] บทความเกี่ยวกับ WebSocket Protocol, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/

[22] บทความเกี่ยวกับโปรแกรม Apache Kafka, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://aws.amazon.com/th/msk/what-is-kafka/

[23] บทความเกี่ยวกับโปรแกรม Apache Kafka, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://kafka.apache.org/documentation/#introduction

[24] บทความเกี่ยวกับโปรแกรมฐานข้อมูล MongoDB, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://www.mongodb.com/docs/

[25] บทความเกี่ยวกับระบบ LiveChat เครื่องมือสื่อสารทางออนไลน์, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://blog.readyplanet.com/17660661/live-chat-in-online-store

เอกสารอ้างอิง (ต่อ)

[26] บทความเกี่ยวกับฐานข้อมูลประเภท NoSQL, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://aws.amazon.com/th/nosql/?nc1=h_ls

[27] บทความเกี่ยวกับ Webhook เครื่องมือสำหรับการส่งแจ้งเตือน, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://sendgrid.com/blog/whats-webhook/

[28] บทความเกี่ยวกับ JSON Web Token (JWT), [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://jwt.io/introduction

[29] บทความเกี่ยวกับการเชื่อมต่อ WebSocket protocol, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://blog.scaleway.com/iot-hub-what-use-case-for-websockets/

[30] บทความเกี่ยวกับ RESTful API, [ออนไลน์], ค้นหาเมื่อ 23 ธันวาคม 2564, จาก https://aws.amazon.com/th/what-is/restful-api/

ประวัติผู้เขียน

ชื่อ-นามสกุล นายภาคภูมิ บุญส่ง

วัน เดือน ปีเกิด 24 กรกฎาคม พ.ศ. 2541

ที่อยู่ปัจจุบัน 8/91 หมู่ 7 ตำบลบ่อวิน อำเภอศรีราชา จังหวัดชลบุรี
รหัสไปรษณีย์ 20230

อีเมล 61515017@kmitl.ac.th

ประวัติการศึกษา ระดับมัธยมศึกษาตอนต้น โรงเรียนแหลมสิงห์วิทยาคม จันทบุรี
ระดับมัธยมศึกษาตอนปลาย สายวิทย์คณิต
โรงเรียนแหลมสิงห์วิทยาคม
เบอร์โทรศัพท์ : 0948681365