รายงานสหกิจศึกษาฉบับสมบูรณ์

การพัฒนาระบบชำระเงิน แบบชำระทันทีและชำระรายเดือน
IMPLEMENT MONTHLY AND RECURRING INSTANT BILLING SYSTEM

ทรงกิจ พาเจริญ

สาขาวิชาวิศวกรรมสารสนเทศ
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

รายงานสหกิจศึกษาฉบับสมบูรณ์

การพัฒนาระบบชำระเงิน แบบชำระทันทีและชำระรายเดือน

IMPLEMENT MONTHLY AND RECURRING INSTANT BILLING SYSTEM

ทรงกิจ พาเจริญ

สาขาวิชาวิศวกรรมสารสนเทศ

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ชื่อโครงงานสหกิจศึกษา การพัฒนาระบบชำระเงินแบบชำระทันทีและชำระรายเดือน
ชื่อ-สกุลนักศึกษา นายทรงกิจ พาเจริญ
คณะ วิศวกรรมศาสตร์บัณฑิต **ภาควิชา** วิศวกรรมสารสนเทศ
ชื่อ-สกุล อาจารย์นิเทศน์ ผศ.ดร.พิกุลแก้ว ตังติสานนท์
ชื่อ-สกุล ผู้นิเทศน์งาน ผศ.ดร.พิกุลแก้ว ตังติสานนท์
ชื่อสถานประกอบการ บริษัท มิวซ์ อินโนเวชั่น จำกัด

บทคัดย่อ

บริษัท มิวซ์ อินโนเวชั่น จำกัด เป็นบริษัทที่มีความเชี่ยวชาญในการสร้างระบบปฏิบัติการของพาณิชย์อิเล็กทรอนิกส์และโปรแกรมประยุกต์สำหรับอุปกรณ์เคลื่อนที่ ระบบ Pumpkin-CRM ซึ่งเป็นหนึ่งในผลิตภัณฑ์ของบริษัท เป็นระบบที่ช่วยสร้างความสัมพันธ์ระยะยาวระหว่างร้านค้ากับลูกค้า เพื่อให้ร้านค้าได้รับลูกค้าที่มีความภักดีต่อร้านค้ามากขึ้น และมี แผงควบคุม Pumpkin ซึ่งเป็นหนึ่งในผลิตภัณฑ์ของระบบ มีการแสดงข้อมูลเชิงลึก และภาพรวมต่าง ๆ ของร้านค้า ทำให้ร้านค้าสามารถเพิ่มยอดขายและกลุ่มลูกค้าใหม่ ๆ ได้ง่าย ซึ่งสามารถใช้งานได้โดยไม่เสียค่าใช้จ่าย แต่ปัจจุบันบริษัทต้องการเก็บค่าบริการจากการใช้งานความสามารถของ แผงควบคุม Pumpkin โครงงานนี้จัดทำขึ้นเพื่อพัฒนาระบบชำระค่าบริการการให้แก่ แผงควบคุม Pumpkin เพื่อให้บริษัทสามารถเก็บค่าบริการจากผู้ใช้งานระบบได้ โดยตั้งเป้าหมายไว้ว่า สามารถชำระค่าบริการรายเดือนและค่าบริการแบบชำระทันทีได้ สามารถคิดค่าบริการได้อย่างถูกต้องและมีประสิทธิภาพ ง่ายต่อการชำระค่าบริการเพื่อความพึงพอใจของลูกค้า มีการวิเคราะห์โครงสร้างระบบชำระค่าบริการเพื่อสร้างระบบที่มีประสิทธิภาพที่ดีที่สุด ทั้งนี้การดำเนินงานนี้จำเป็นต้องใช้ความรู้ทางด้าน การออกแบบพัฒนาเว็บไซต์แอปพลิเคชันในส่วนติดต่อผู้ใช้ การพัฒนาและจัดการระบบหลังบ้านของเว็บไซต์แอปพลิเคชัน และการเชื่อมต่อระบบชำระเงินกับผู้ให้บริการชำระเงินบุคคลที่สาม

คำสำคัญ : ระบบชำระค่าบริการ, โควตา, ฟีเจอร์, ทีมผู้พัฒนา, แดชบอร์ด, Pumpkin-CRM

Co-operative Title	Implement monthly and recurring instant billing system
Student Intern Name	Mr. Songkit Phachareon
Faculty	Engineering Department Information Engineering
Advisor Name	Asst.Prof.Dr. Pikulkaew Tangtisanon
Mentor Name	Asst.Prof.Dr. Pikulkaew Tangtisanon
Company	Muze Innovation Co., Ltd.

ABSTRACT

Muze Innovation Co., Ltd. is specialized in e-commerce system and mobile application development. Pumpkin–CRM system is one of the company's products that has been used to make a long-term relationship between stores and its customer. The system has also been used to ensures customer loyalty to the stores. Pumpkin-Dashboard, a dashboard program shipped with the system, shows an insight data and provide overview of the store therefore the store can boost their sales volume and reach new customer groups with ease. The dashboard was free-to-use but now the company wants to collect the service charge from it. The objective of this project is to develop a payment system for user to pay for the Pumpkin-Dashboard service. The goal of this system is to enable user to pay monthly or instant service charge and to calculate service charge precisely and effectively in order to maximize customer satisfaction. The infrastructure of the payment system is analyzed, developed and improved to get the best performance. Therefore, the development of this system needs knowledge in designing and developing user interface (Front-end web development), developing and managing databases and core systems of the application website (Back-end web development) and connecting payment system to a third-party (3rd-party payment gateway).

Keywords: Billing payment system, Quota, Feature, Developer team, Dashboard, Pumpkin-CRM

กิตติกรรมประกาศ

การจัดทำโครงงานพัฒนาเว็บแอปพลิเคชันเพื่อสร้างระบบชำระค่าบริการสินค้าแบบชำระทันทีและชำระรายเดือนนี้ เป็นโครงงานของบริษัท มิวซ์ อินโนเวชั่น จำกัด (Muze Innovation Co., Ltd) เพื่อพัฒนาระบบของโครงงานให้เสร็จสมบูรณ์มากขึ้น ตลอดช่วงระยะเวลาตั้งแต่วันที่ 1 มิถุนายน 2560 จนถึง 24 พฤศจิกายน 2560 ที่ผู้จัดทำรับผิดชอบและปฏิบัติหน้าที่ ได้รับความรู้ ความเข้าใจและประสบการณ์ในการทำงานที่เป็นประโยชน์อย่างมาก ทั้งด้านวิชาการและวัฒนธรรมองค์กรตลอดระยะเวลาการทำงาน

การปฏิบัติงานในโครงงานนี้สามารถบรรลุวัตถุประสงค์ได้เพราะได้รับการชี้แนะดูแลและความช่วยเหลือต่าง ๆ จาก คุณศิริวัชร์ คุณาพร, คุณธีรัตน์ จึงธนาเจริญเลิศ และคุณธรรมวัฒน์ บงการณ์ รวมทั้งบุคลากรท่านอื่น ๆ ที่ไม่ได้กล่าวนามทุกท่าน ที่ได้มีส่วนเกี่ยวข้องในการเป็นที่ปรึกษาและให้คำแนะนำช่วยเหลือในการปฏิบัติงานให้สำเร็จลุล่วงไปด้วยดี

ขอขอบคุณ ผศ.ดร. พิกุลแก้ว ตังติสานนท์ อาจารย์นิเทศโครงการสหกิจศึกษา ที่คอยให้การสนับสนุนข้อเสนอแนะ และแนวทางแก้ไขปัญหาและข้อบกพร่องต่าง ๆ ตลอดจนการติดตามความคืบหน้าของผลงาน

สุดท้ายนี้ขอขอบพระคุณผู้มีส่วนเกี่ยวข้องทุกท่านที่มีส่วนร่วมในการให้ข้อมูล เป็นที่ปรึกษาทั้งทางตรงและทางอ้อมจนการทำรายงานเล่มนี้สำเร็จลงด้วยดี ขอขอบพระคุณไว้ ณ ที่นี้ด้วย

ทรงกิจ พาเจริญ

สารบัญ

บทคัดย่อ I

ABSTRACT II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VII

สารบัญรูปภาพ VIII

บทที่ 1 บทนำ 1

1.1 ข้อมูลสถานประกอบการที่เข้าร่วมปฏิบัติงานสหกิจศึกษา 1

1.2 ความเป็นมาและความสำคัญ 2

1.3 วัตถุประสงค์ของการปฏิบัติงาน 3

1.4 วิธีการดำเนินงาน 3

1.5 ขอบเขตของงาน 4

1.6 ประโยชน์ที่คาดว่าจะได้รับ 4

บทที่ 2 แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง 5

2.1 ความหมายของแดชบอร์ด (Dashboard) 5

2.2 ความหมายของค่าบริการรายเดือนและค่าบริการแบบชำระทันที 6

2.3 เทคโนโลยีที่เกี่ยวข้อง 6

2.3.1 รีแอค (React) 6

2.3.2 อีโมชัน เจเอส (Emotion JS) 8

2.3.3 สตอรี่บุ๊ค (Storybook) 8

2.3.4 แมททีเรียล-ยูไอ ไลบรารี่ (MATERIAL-UI Library) 8

2.3.5 คลาวด์ไฟร์สโตร (Cloud Firestore) 9

2.3.6 โหนด เจเอส (Node.js) 11
2.3.7 เอ็กซ์เพรส (Express) 11
2.3.8 โลแดช (Lodash) 11
2.4 เครื่องมือที่ใช้ในการพัฒนาระบบ 11
2.4.1 วิชวลสตูดิโอโค้ด (Visual Studio Code) 11
2.4.2 ซอร์สทรี (Source Tree) 12
2.4.3 กูเกิลโครม (Google Chrome) 12
2.4.4 เซปลิ้น (Zeplin) 13
2.4.5 โพสแมน (Postman) 14
บทที่ 3 ขั้นตอนการดำเนินงาน 15
3.1 จัดเตรียมเครื่องมือและอุปกรณ์ที่ใช้ 15
3.2 การออกแบบและวิเคราะห์ระบบ 16
3.2.1 สาเหตุของปัญหา 16
3.2.2 สืบค้นข้อมูล (Research) 16
3.2.3 วิเคราะห์ข้อมูลที่สืบค้น 17
3.2.4 ออกแบบระบบสินค้า (Pumpkin-Dashboard's products) 18
3.2.5 ออกแบบระบบชำระค่าบริการ 19
3.2.5.1 ออกแบบระบบชำระค่าบริการสินค้าระบบ Package 19
3.2.5.2 ออกแบบระบบชำระค่าบริการสินค้าระบบ Products 20
3.2.6 การรวมกับระบบเดิม 23
3.3 การดำเนินการพัฒนาระบบ 25
3.3.1 ศึกษาและทำความเข้าใจความต้องการของระบบ 25
3.3.2 ออกแบบ UI/UX 26
3.3.3 การพัฒนาระบบหน้าบ้าน (Front-end) 29

3.3.3 การพัฒนาระบบหลังบ้าน (Back-end) 30
3.4 ทดสอบระบบ 31
3.4.1 การทดสอบระบบชำระค่าแบบชำระทันที (Instant Billing System) 31
3.4.2 การทดสอบระบบชำระค่าบริการแบบรายเดือน (Monthly Billing System) 34
3.4.2.1 ปัญหาที่เกิดขึ้น 34
3.4.2.1.1 ข้อจำกัดด้านเทคนิค 34
3.4.2.1.1 ข้อจำกัดด้านเวลา 34
3.4.2.2 การแก้ปัญหา 34
3.4.2.2.1 การแก้ปัญหาด้านเทคนิค 34
3.4.2.2.2 การแก้ปัญหาด้านเวลา 35
บทที่ 4 ผลการทำงาน 36
4.1 ระบบชำระค่าบริการแบบชำระทันที (Instant billing system) 36
4.2 ระบบชำระค่าบริการแบบรายเดือน (Monthly billing system) 46
4.3 ระบบเปลี่ยนแพ็กเกจ (Package) 51
บทที่ 5 สรุปผลการวิจัย และข้อเสนอแนะ 53
5.1 สรุปผลวิจัย 53
5.2 ปัญหาและอุปสรรคในการดำเนินงาน 53
5.3 วิธีการแก้ปัญหา 53
5.4 ข้อเสนอแนะ 54
ภาคผนวก 55
ภาคผนวก ก การพัฒนา RESTful API และการทดสอบการทำงานโดยใช้ Postman 57
ภาคผนวก ข วิธีติดตั้ง JS library ลงในโครงงาน 69
เอกสารอ้างอิง 71

สารบัญตาราง

ตารางที่ 3.1 ตารางคำนวณตัวอย่างการคิดค่าบริการรายเดือนระบบ Package 19
ตารางที่ 4.1 ตารางคำนวณค่าบริการที่เกิดขึ้นจากการจำลองสถานการณ์ที่ 1 48
ตารางที่ 4.2 ตารางคำนวณค่าบริการที่เกิดขึ้นจากการจำลองสถานการณ์ที่ 2 50

สารบัญรูปภาพ

รูปที่ 1.1 สัญลักษณ์ของบริษัท มิวซ์ อินโนเวชั่น จำกัด 1
รูปที่ 2.1 Pumpkin-Dashboard 5
รูปที่ 2.2 การส่งค่าของ React 6
รูปที่ 2.3 Component คือส่วนประกอบเล็ก ๆ ที่ประกอบกันจนกลายเป็นหน้าเว็บไซต์ 7
รูปที่ 2.4 การเก็บข้อมูลของ Realtime Database 9
รูปที่ 2.5 การเก็บข้อมูลของ Cloud Firestore Database 10
รูปที่ 2.6 สัญลักษณ์ VSCode 12
รูปที่ 2.7 สัญลักษณ์ Source Tree 12
รูปที่ 2.8 สัญลักษณ์ Google Chrome 13
รูปที่ 2.9 สัญลักษณ์ Zeplin 14
รูปที่ 2.10 สัญลักษณ์ Postman 14
รูปที่ 3.1 เว็บไซต์ของ LnwSHOP 16
รูปที่ 3.2 เว็บไซต์ของ Netflix 17
รูปที่ 3.3 เว็บไซต์ของ SellSUKI 17
รูปที่ 3.4 ระบบ Package ของ SellSUKI 18
รูปที่ 3.5 แผนภาพแสดงเหตุการณ์ตัวอย่างการคิดค่าบริการรายเดือนระบบ Package 20
รูปที่ 3.6 Flowchart แสดง Checkout flow ของระบบ e-commerce 22
รูปที่ 3.7 Block diagram การทำงานของ Notification feature 24
รูปที่ 3.8 แผนภาพแสดงการส่ง Notification ไปหาลูกค้า 24
รูปที่ 3.9 Block diagram การทำงานของ Notification feature แบบใหม่ 25
รูปที่ 3.10 แผนภาพแสดงการส่ง Notification ไปหาลูกค้าแบบใหม่ 25
รูปที่ 3.11 Wireframe ที่ใช้ในการอธิบาย Workflow 26
รูปที่ 3.12 งาน Design ที่ออกแบบจาก Wireframe 28
รูปที่ 3.13 Component ที่ถูกเขียนด้วย React แสดงผลใน Storybook 29
รูปที่ 3.14 การพัฒนา Front-end โดยปรับใช้ Atomic structure ผ่านเครื่องมือ VSCode 30
รูปที่ 3.15 Cloud Firestore Dashboard 30
รูปที่ 3.16 การพัฒนา API บนเครื่องมือ VSCode 31
รูปที่ 3.17 Instant billing system's workflow 33

รูปที่ 4.1 หน้าสร้าง Customer group ที่มีการสร้างกลุ่มไปแล้ว 3 กลุ่ม 36
รูปที่ 4.2 สร้างกลุ่มลูกค้าใหม่ชื่อกลุ่มว่า “silvers” 37
รูปที่ 4.3 รอกระบวนการสร้างกลุ่มใหม่ 37
รูปที่ 4.4 แจ้งเตือนว่าไม่สามารถสร้างกลุ่มใหม่ได้ ต้องซื้อโควตาเพิ่ม 38
รูปที่ 4.5 หน้า “แพ็กเกจของฉัน” 38
รูปที่ 4.6 หน้า “Market” ที่รวบรวม feature ที่สามารถซื้อโควตาเพิ่มได้ 39
รูปที่ 4.7 เมื่อเลือกกด Add to cart ที่ Product card จะเพิ่มจำนวนสินค้าที่อยู่ในตะกร้า 39
รูปที่ 4.8 หน้า Checkout แสดงรายการสินค้าที่เลือกไว้ก่อนชำระค่าสินค้า 40
รูปที่ 4.9 แบบฟอร์มสำหรับเก็บข้อมูลใบกำกับภาษี 41
รูปที่ 4.10 แบบฟอร์มสำหรับบัตรเครดิต/เดบิต 41
รูปที่ 4.11 รอระบบทำรายการ 42
รูปที่ 4.12 หน้ากรอก OTP ของ OMISE ที่ถูกข้ามขั้นตอนกรอกข้อมูลใน OMISE-Dev mode 42
รูปที่ 4.13 หน้า “แพ็กเกจของฉัน” ที่ถูก Redirect มาจาก OMISE 43
รูปที่ 4.14 หน้า “แพ็กเกจของฉัน” ส่วนแสดง Products ที่ซื้อจากระบบ 43
รูปที่ 4.15 หน้า “แพ็กเกจของฉัน” ส่วนแสดง Order list 44
รูปที่ 4.16 หน้า “OMISE Dashboard” แสดงข้อมูลธุรกรรมที่เกิดขึ้น 44
รูปที่ 4.17 หน้า “Customer Group” แสดงข้อมูลลูกค้าทั้งหมดที่มีในระบบของร้านค้า 45
รูปที่ 4.18 หน้า “Customer Group” สร้างกลุ่มลูกค้าใหม่ชื่อ “silvers” 45
รูปที่ 4.19 หน้า “Customer Group”กำลังทำรายการ 46
รูปที่ 4.20 หน้า “Customer Group” ทำรายการสำเร็จกลุ่มใหม่ชื่อ “silvers” ถูกเพิ่มขึ้นมา 46
รูปที่ 4.21 หน้าทดสอบ API เพื่อใช้ทดสอบการคำนวณค่าบริการระบบรายเดือน 47
รูปที่ 4.22 ปรับลดโควตา Customer group ให้เหลือเพียง 1 กลุ่ม 49
รูปที่ 4.23 หน้าทดสอบ API คำนวณค่าบริการระบบรายเดือนในสถานการณ์ที่ 2 49
รูปที่ 4.24 Free package ที่ร้านค้าใช้งานอยู่ 51
รูปที่ 4.25 เมื่อกดปุ่ม “UPGRADE” ใน Package card จะนำมายังหน้า “เปลี่ยนแพ็กเกจ” 52
รูปที่ 4.26 แพ็กเกจถูกเปลี่ยนเป็น Starter package 52
รูปที่ ก.1 การพัฒนาระบบ API ของ Pumpkin-CRM (1) 59
รูปที่ ก.2 การพัฒนาระบบ API ของ Pumpkin-CRM (2) 60
รูปที่ ก.3 การพัฒนาระบบ API ของ Pumpkin-CRM (3) 61

รูปที่ ก.4 การพัฒนาระบบ API ของ Pumpkin-CRM (4) 62
รูปที่ ก.5 การทดสอบการทำงานของ API (1) 63
รูปที่ ก.6 การทดสอบการทำงานของ API (2) 63
รูปที่ ก.7 การทดสอบการทำงานของ API (3) 64
รูปที่ ก.8 การทดสอบการทำงานของ API (4) 65
รูปที่ ก.9 การทดสอบการทำงานของ API (5) 65
รูปที่ ก.10 การทดสอบการทำงานของ API (6) 65
รูปที่ ก.11 การทดสอบการทำงานของ API (7) 66
รูปที่ ก.12 การทดสอบการทำงานของ API (8) 66
รูปที่ ก.13 การตั้งค่าสภาพแวดล้อมของ Postman (1) 67
รูปที่ ก.14 การตั้งค่าสภาพแวดล้อมของ Postman (2) 67
รูปที่ ก.15 การตั้งค่าสภาพแวดล้อมของ Postman (3) 68
รูปที่ ข.1 หน้าเว็บไซต์ของ Lodash 70
รูปที่ ข.2 Terminal ของ VSCode 70
รูปที่ ข.3 ข้อความที่แสดงบน Terminal ของ VSCode 70

บทที่ 1
บทนำ

1.1 ข้อมูลสถานประกอบการที่เข้าร่วมปฏิบัติงานสหกิจศึกษา

บริษัท มิวซ์ อินโนเวชั่น จำกัด (Muze Innovation Co., Ltd.) เป็นบริษัทผู้ให้บริการในการสร้างระบบปฏิบัติการของพาณิชย์อิเล็กทรอนิกส์ (e-commerce platform) สำหรับผู้ค้า (Vendor) ที่ต้องการ e-commerce website ของตัวเอง มีผลงานอาทิเช่น SAMSUNG, ascenD, bemynt, Ari, Superdry และ MINOR เป็นต้น ซึ่ง e-commerce website ที่ถูกสร้างจากเทคโนโลยี e-commerce platform ของบริษัทฯ เป็นเทคโนโลยีที่ถูกเรียกว่า inCart-platform

inCart-platform เป็นเทคโนโลยีที่มีระบบ CMS (Content Management System) ทำให้ผู้ใช้สามารถสร้างหน้าเว็บไซต์ของตัวเองได้โดยไม่จำเป็นต้องมีความรู้เรื่องโปรแกรมมิ่ง (ตัวอย่างเช่น https://webflow.com) และมีระบบที่จำเป็นต่อระบบ e-commerce เช่น ระบบจัดการสินค้า ระบบสต๊อกสินค้า ระบบชำระค่าบริการสินค้า เป็นต้น ทำให้ Vendor สามารถสร้าง e-commerce website เป็นของตัวเองได้สะดวก รวดเร็วที่สุด ซึ่ง inCart-platform สร้างขึ้นจากการใช้เทคโนโลยี Next.js ในการจัดการ Front-end และมี Magento เป็น Back-end (ในอนาคตมีแผนใช้เทคโนโลยีของบริษัทเองโดยมีพื้นฐานจาก Magento) โดยบริษัทใช้สัญลักษณ์ดังรูปที่ 1.1

รูปที่ 1.1 สัญลักษณ์ของบริษัท มิวซ์ อินโนเวชั่น จำกัด

ที่มา https://www.muze.co.th/

1.2 ความเป็นมาและความสำคัญ

ระบบ Pumpkin-CRM เป็นหนึ่งใน Startup products ของบริษัท มิวซ์ อินโนเวชั่น จำกัด ระบบนี้มีวัตถุประสงค์คือ เป็นระบบที่ช่วยสร้างความสัมพันธ์ระยะยาวระหว่างร้านค้ากับลูกค้า เพื่อให้ร้านค้าได้รับลูกค้าที่มีความภักดีต่อร้านค้ามากขึ้น จะด้วยการมอบสิทธิพิเศษแก่ลูกค้าที่มาใช้บริการบ่อยครั้ง หรือการส่งข้อความแจ้งเตือนทางสมาร์ทโฟน แจ้งกิจกรรมหรือโปรโมชั่นของร้านค้า เพื่อดึงดูดลูกค้าที่เคยใช้บริการแล้วกลับมาใช้บริการซ้ำ และดึงดูดลูกค้าที่ไม่เคยใช้บริการร้านค้า หันมาใช้บริการมากขึ้น นอกจากนี้ยังรวมถึงการเก็บพฤติกรรมลูกค้าที่มีต่อร้านและสินค้าของร้าน เพื่อนำข้อมูลไปวิเคราะห์และใช้ในการสร้างกิจกรรมดึงดูดลูกค้าใหม่ๆ และพัฒนาสินค้าของร้านให้ตรงความความต้องการของลูกค้ามากขึ้น เพื่อเป้าหมายนั้น ร้านค้าจึงต้องมีระบบหลังบ้านที่ช่วยในการสร้างกิจกรรม รวบรวมข้อมูล และวิเคราะห์พฤติกรรมลูกค้า ทำได้ง่าย สะดวก รวดเร็วขึ้น จากสาเหตุที่กล่าวมาข้างต้นทำให้เกิดระบบ Pumpkin-CRM ขึ้นมา

ระบบ Pumpkin-CRM มี Products ย่อยที่ใช้ในระบบทั้งหมด 2 แพลตฟอร์ม จำนวน 4 Products ประกอบด้วย

1.2.1 แพลตฟอร์มโมบาย (Mobile application) จำนวน 2 Products ได้แก่

1.2.1.1 **พัมคิน-โลยอลตี้ แอป (Pumpkin–Loyalty app)** แอปพลิเคชันที่ถูกพัฒนาขึ้นเพื่อให้บริการลูกค้าของร้านค้า เป็นฟรีแอปพลิเคชันสะสมคะแนนที่ช่วยให้ผู้ใช้งานไม่ต้องพกบัตรสะสมแสตมป์ (บัตรกระดาษ) ของร้านค้าอีกต่อไป รวมถึงแก้ปัญหาต่าง ๆ ที่ผู้ใช้งานประสบปัญหาอยู่ เช่น บัตรหาย หรือลืมวันหมดอายุ ในแอปพลิเคชันนี้สามารถสะสมคะแนนของทุกร้านค้าที่อยู่ในระบบของ Pumpkin-CRM ได้ไม่จำกัดจำนวน มีระบบแลกของรางวัล แจ้งข่าวสารหรือโปรโมชั่นจากร้านค้า ระบบสะสมคูปองเพื่อแลกสิทธิพิเศษจากร้านค้าต่าง ๆ ได้ เป็นต้น

1.2.1.2 **พัมคิน เมอแช็น (Pumpkin Merchant)** แอปพลิเคชันสำหรับร้านค้าที่ใช้งาน Pumpkin-RCM มีระบบมอบแต้มแก่ลูกค้าที่ผ่านเงื่อนไขกิจกรรมของร้านค้า และมอบแต้มสะสมแก่ลูกค้าที่ซื้อสินค้า

1.2.1 แพลตฟอร์มเว็บไซต์ (Website application) จำนวน 2 Products ได้แก่

1.2.2.1 **Pumpkin - Loyalty** (ver. WebApp) ความสามารถเหมือนใน Mobile application ถูกพัฒนาขึ้นเพื่อแก้ปัญหาความล่าช้าที่เกิดจากการดาวน์โหลดแอปพลิเคชันจาก Store (Play Store สำหรับ Android และ App Store สำหรับ IOS) ในช่วงเวลาเร่งด่วน ลูกค้าสามารถแสกน QR code จากพนักงานเพื่อรับและสะสมแต้มได้บนเว็บไซต์เลยทันที สามารถนำ ID ที่ได้จากเว็บไซต์ไป login เข้า Mobile application เพื่อโอนแต้มเข้าแอปพลิเคชันทีหลังได้

1.2.2.2 **พัมคิน-แดชบอร์ด (Pumpkin-Dashboard)** เป็นระบบหลังบ้านของร้านค้าที่ใช้งานระบบ Pumpkin-CRM มีความสามารถคร่าว ๆ ดังนี้

- **การตรวจสอบและติดตามข้อมูลร้าน (Monitoring)** ดูความเคลื่อนไหวของร้านค้าในช่วงเวลาต่าง ๆ ตรวจสอบสินค้าที่ขายดี ข้อมูลรางวัล ดูข้อมูลสาขา การสะสมแต้มและการแลกของรางวัล
- **การจัดการทรัพยากรร้าน (Management)** จัดการระดับกลุ่มลูกค้าที่เข้ามาใช้บริการร้านค้า เช่น กลุ่มระดับ Bronze, กลุ่มระดับ Silver, กลุ่มระดับ Gold เป็นต้น สามารถปรับแต่ง Brand card สามารถสร้าง QR code สำหรับสาขาต่าง ๆ ได้
- **การสร้างกิจกรรม (Create)** สร้าง Content หรือ กิจกรรมต่าง ๆ เพื่อดึงดูดลูกค้ามาใช้บริการ เช่น Notification, Coupon, Promotion, Survey เพื่อเก็บ Feedback ของลูกค้าที่มีต่อร้านค้า

1.3 วัตถุประสงค์ของการปฏิบัติงาน

ในปัจจุบัน Pumpkin มีแดชบอร์ดที่ให้แอดมินของร้านค้า สามารถสร้าง Notification, Campaign, Survey หรือจัดการกลุ่มลูกค้าให้มีหลายระดับได้ ซึ่งเป็นหนึ่งในหลาย ๆ ความสามารถของแดชบอร์ดได้ฟรี ไม่มีการเก็บค่าบริการใด ๆ ซึ่งในอนาคต Pumpkin มีแผนว่าจะเก็บค่าบริการในการใช้งานแดชบอร์ดจึงต้องการระบบจัดการเรื่องการเก็บค่าบริการเหล่านี้ ทำให้เกิดหัวข้อโปรเจคนี้ขึ้น

1.4 วิธีการดำเนินงาน

1.4.1 ศึกษาระบบและเทคโนโลยีของ Software รวมถึงคุณสมบัติการทำงานต่าง ๆ ตามความต้องการของระบบ (Requirement)

1.4.2 วิเคราะห์ โครงสร้างระบบชำระค่าบริการแบบรายเดือนและแบบชำระทันทีเพื่อให้ได้ตรงตาม Requirement

1.4.3 ทดสอบระบบ

1.4.4 วิเคราะห์ปัญหาที่เกิดขึ้น

1.4.5 เสนอแนะแนวทางการเพิ่มประสิทธิภาพ

1.4.6 สรุปผล

1.5 ขอบเขตของงาน

1.5.1 เรียนรู้การทำงานและศึกษาโครงสร้างระบบชำระค่าบริการ

1.5.2 กำหนดรูปแบบสินค้าและรูปแบบการชำระบริการ

1.5.3 สร้างรูปแบบการชำระบริการตามที่ออกแบบไว้

1.5.4 สามารถคำนวณค่าชำระบริการรายเดือนที่ร้านค้าต้องชำระได้ถูกต้อง

1.5.5 เชื่อมระบบชำระเงินกับผู้ให้บริการชำระเงินบุคคลที่สาม (3rd party payment gateway)

1.5.6 สร้างใบสั่งซื้อสินค้า (Order) ที่รวมยอดชำระที่เกิดขึ้นภายในเดือนได้

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1.6.1 นักศึกษาได้รับความรู้และเข้าใจในเรื่องของการพัฒนาเว็บไซต์ฝั่งติดต่อผู้ใช้งาน (Front-end) การเชื่อมกับระบบ 3rd party payment gateway และ การเขียน API เพื่อสร้างระบบที่ตรงตาม Requirement ที่ได้รับ

1.6.2 นักศึกษาได้รับความรู้และความเข้าใจในการพัฒนาระบบโครงสร้างการทำงานของระบบชำระค่าบริการ

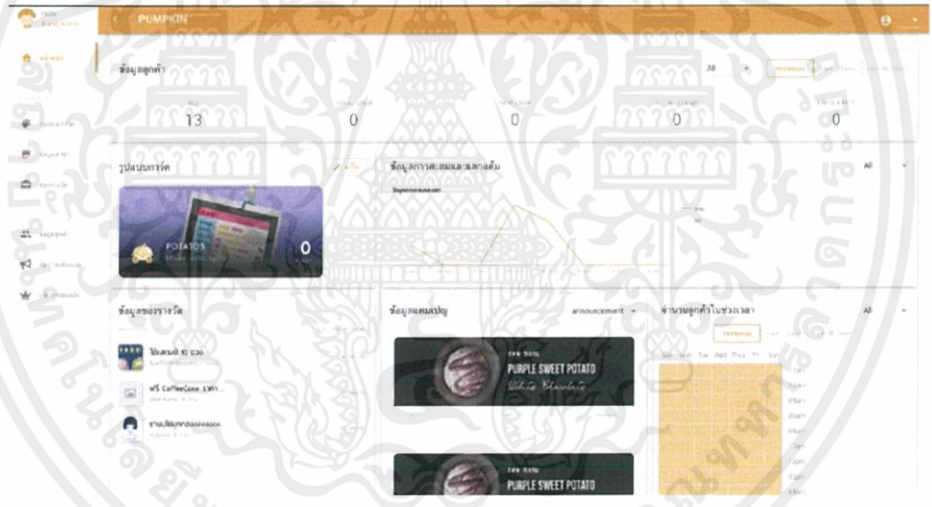
1.6.3 นักศึกษาได้รับความรู้และความเข้าใจในการพัฒนา ระบบการจำกัดจำนวนการใช้งานฟีเจอร์ (feature) ของแดชบอร์ด (Dashboard)

1.6.4 บริษัทฯ ได้รับแนวทางในการพัฒนาระบบชำระค่าบริการ ความต้องการที่วางแผนไว้

บทที่ 2
แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

2.1 ความหมายของแดชบอร์ด (Dashboard)

แดชบอร์ด (Dashboard) ในที่นี้คือ หน้าเว็บไซต์ที่รวมรวบสิ่งต่าง ๆ ที่สำคัญบนระบบ Pumpkin-CRM มารวมไว้ในที่เดียวดังรูปที่ 2.1 โดยที่ผู้สามารถใช้งานได้ คือแอดมินของร้านค้าที่ได้รับสิทธิ มีความสามารถในการตรวจสอบ จัดการทรัพยากรของร้านค้า (คะแนนสะสมของร้าน, จำนวนผู้ดูแลระบบ, Promotion, Notification, Campaign, Survey etc.) ตัวอย่างเช่น สามารถให้แอดมินของร้าน สร้าง Promotion แล้วส่ง Notification ไปหาลูกค้าที่มี Pumpkin-Loyalty application ได้ สามารถตรวจสอบความเคลื่อนไหวของร้านได้(จำนวนแต้มที่มอบให้ลูกค้า, จำนวนแต้มที่ถูกแลก, ข้อมูลที่ได้จากแบบสอบถาม) เป็นต้น



รูปที่ 2.1 Pumpkin-Dashboard

2.2 ความหมายของค่าบริการรายเดือนและค่าบริการแบบชำระทันที

ค่าบริการรายเดือนและค่าบริการแบบชำระทันที แบ่งอธิบายได้ 2 ส่วนคือ ค่าบริการแบบรายเดือน และค่าบริการแบบชำระทันที

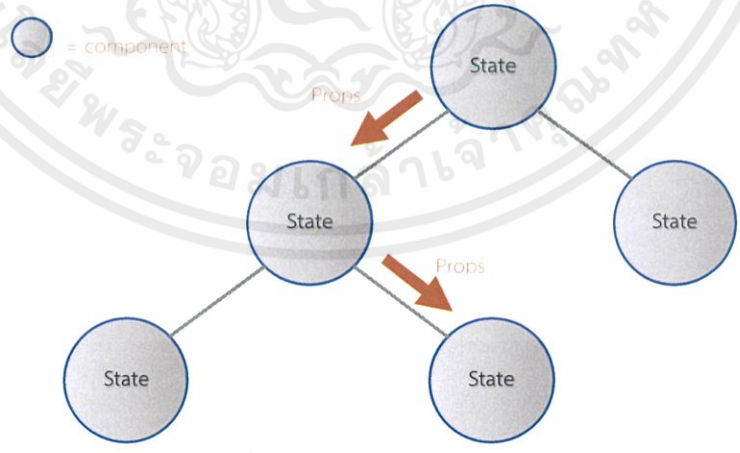
- ค่าบริการรายเดือน คือค่าบริการที่เกิดจากการใช้งานแดชบอร์ดภายใน 1 รอบบิล (1 รอบบิล สามารถกำหนดให้เป็นหลายเดือนได้ แต่ในที่นี้กำหนดให้ 1 รอบบิลเท่ากับ 1 เดือนเพื่อความเข้าใจง่าย) ซึ่งจะเก็บค่าบริการหลังจากจบรอบบิลที่กำหนด
- ค่าบริการแบบชำระทันที คือค่าบริการที่เกิดขึ้นจากการซื้อจำนวนโควตาการใช้งาน feature ของแดชบอร์ดแล้วชำระค่าบริการทันทีที่ซื้อ

2.3 เทคโนโลยีที่เกี่ยวข้อง

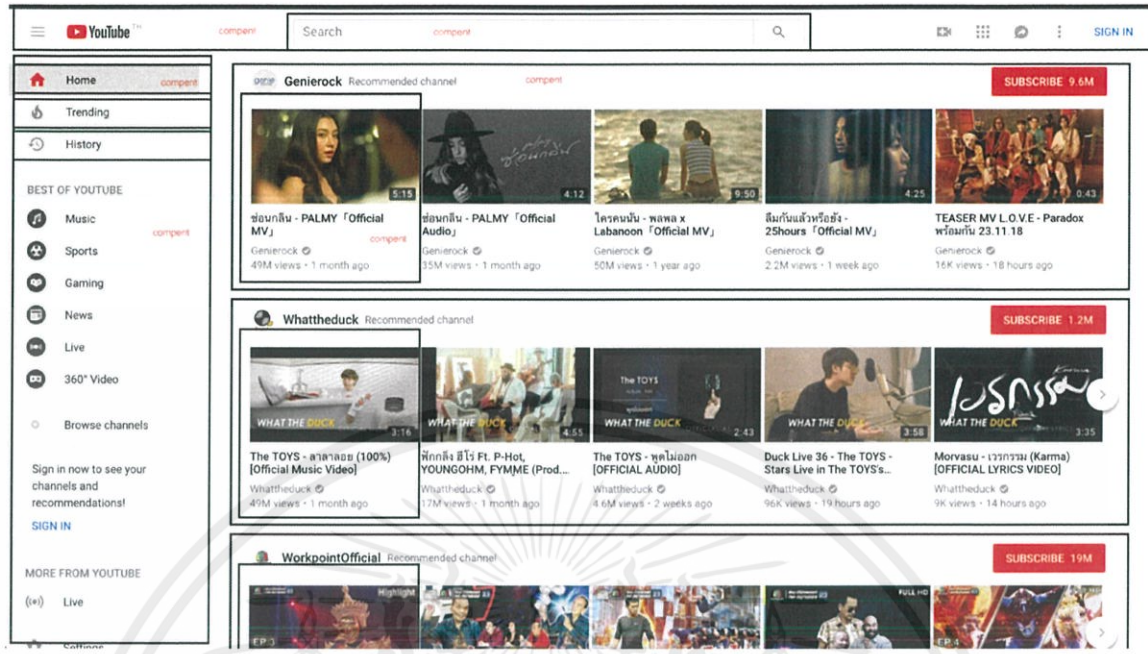
2.3.1 รีแอค (React)

React เป็น JavaScript Library ที่ถูกพัฒนาขึ้นโดย Facebook เพื่อใช้พัฒนา UI สำหรับเว็บไซต์ (UI: User Interface ส่วนหน้าเว็บที่ใช้ติดต่อกับผู้ใช้ มีการแสดงผลที่สวยงาม ง่ายต่อการใช้งาน) โดยจะใช้ JavaScript เพื่ออธิบายหน้าตาและพฤติกรรมของ View แทนที่จะเขียน HTML ตรง ๆ แทน Concept หลักที่ใช้ใน React เบื้องต้นคือ (ดูรูปที่ 2.2 และ 2.3 ประกอบเพื่อทำความเข้าใจ)

- **Component** ส่วนประกอบต่าง ๆ ที่ทำให้เกิดหน้าเว็บไซต์
- **State** ข้อมูลที่ Component ยึดถือเพื่อใช้ในการแสดงผล
- **Props** ข้อมูลที่ถูกส่งต่อจาก Component ชั้นบนลงไปชั้นล่าง



รูปที่ 2.2 การส่งค่าของ React



รูปที่ 2.3 Component คือส่วนประกอบเล็ก ๆ ที่ประกอบกันจนกลายเป็นหน้าเว็บไซต์

จุดเด่นของ React อยู่ที่ การสร้างหน้าด้วย Component ทำให้สามาถนำไป Reuse ได้หลายที่ ที่ ทำให้ไม่ต้องเขียน code ซ้ำซ้อน ทำให้การพัฒนาหน้าเว็บไซต์ทำได้ง่ายและรวดเร็ว ซึ่งแนวคิดของ Component คือการเขียนองค์ประกอบของ View ย่อยมากที่สุดเท่าที่จะทำได้ และเน้นการ Reuse ให้เยอะ อีกจุดเด่นคือ เวลาข้อมูลมีการเปลี่ยนแปลง เฉพาะ View ที่เกี่ยวข้องกับข้อมูลนั้นเท่านั้นถึงจะถูกเปลี่ยนพฤติกรรม อธิบายคือ เปลี่ยน View เฉพาะส่วนที่จำเป็น ทำให้เว็บไซต์เร็วขึ้นเพราะไม่ต้องเปลี่ยนทั้งหน้าเว็บ

ระบบนิเวศ (Ecosystem) ของ **React** ประกอบด้วย

อีเอสซิก (ES6) เป็น JavaScript สมัยใหม่ ที่มี feature ที่สำคัญมากมายเช่น Arrow function (=>), ประกาศตัวแปรใน Block scope ด้วย let, const, Class etc. ทำให้เขียน code ได้ง่ายขึ้น และสวยงามกว่า JavaScript เวอร์ชั่นเก่า

บาเบล (Babel) เนื่องจาก ES6 เป็น JavaScript สมัยใหม่ ทำให้บาง Browser ไม่รองรับ จำเป็นต้องแปลง Syntax ES6 ให้เป็น JavaScript พื้นฐาน เพื่อให้ Browser รองรับได้ ซึ่งเป็นหน้าที่ของ Babel เป็นตัวช่วยแปลง

เว็บแพ็ค (Webpack) เพื่อให้โครงสร้างโปรเจคเป็นระเบียบ จึงมีการแยกฟังก์ชันที่สามารถ Reuse ได้ ออกมาเป็นไฟล์เฉพาะ เก็บไว้เป็น Module เมื่อจะนำมาใช้งานค่อย Include script เข้ามาใช้

ซึ่งเมื่อไฟล์มีจำนวนมาก จะให้การโหลดโปรเจคช้า Webpack จะรวมไฟล์ทั้งหมดในโปรเจคเป็นไฟล์เดียว เมื่อ Compile พร้อมจัดการลำดับ Include ให้เสร็จสรรพ ทำให้โปรเจคเร็วขึ้น

สามารถทดลองเริ่มเขียน React โดยใช้ create-react-app ซึ่งเป็น Command line tools ที่ช่วยให้ผู้ใช้สามารถเริ่มต้นโปรเจค React ได้อย่างรวดเร็วโดยไม่ต้องตั้งค่าอะไรเพิ่มเติมจากที่เป็น

2.3.2 อีโมชัน เจเอส (Emotion JS)

Emotion เป็น CSS-in-JS Library ที่มีการพัฒนาต่อยอดจาก CSS-in-JS Library อื่น ๆ เช่น glam, glamor, styled-components และ glamorous มีประสิทธิภาพและมีความยืดหยุ่นสูง ช่วยในการจัดการปัญหาเฉพาะทางของ CSS ที่มีใน Browser หลาย ๆ รุ่น ถือเป็น Library หลักในการพัฒนาเว็บไซต์ที่ใช้ React

2.3.3 สตอรี่บุ๊ค (Storybook)

Storybook เป็นเครื่องมือที่ช่วยให้ Developer สามารถ Test UI component ที่เขียนด้วย framework สมัยใหม่ เช่น Angular, React, Vue ได้ง่ายขึ้น ปกติแล้วกว่าจะสร้าง Component ที่มีรูปร่างตรงกับ Design สวยงามน่าใช้ ต้องมีการแก้แล้ว Save เรื่อย ๆ ใน Codebase ยิ่งโปรเจคที่มีขนาดใหญ่มากยิ่งทำให้ Build ช้า ทำให้การทำงานยิ่งช้าลงไปอีก

Storybook คือสมุดภาพ ที่ Developer สามารถใส่ Component ที่กำลังสร้างอยู่ เพื่อ Build ดูผลลัพธ์จาก code ที่เขียนได้ได้ทันที สามารถ Test component เฉพาะโดยไม่กระทบกับโปรเจคโดยรวมได้ อีกทั้งทำให้ Designer เห็นภาพด้วยทันทีว่าเมื่อ Component แสดงผลบนเว็บจริงจะออกมาเห็นหน้าตาแบบใด

2.3.4 แมททีเรียล-ยูไอ ไลบรารี่ (MATERIAL-UI Library)

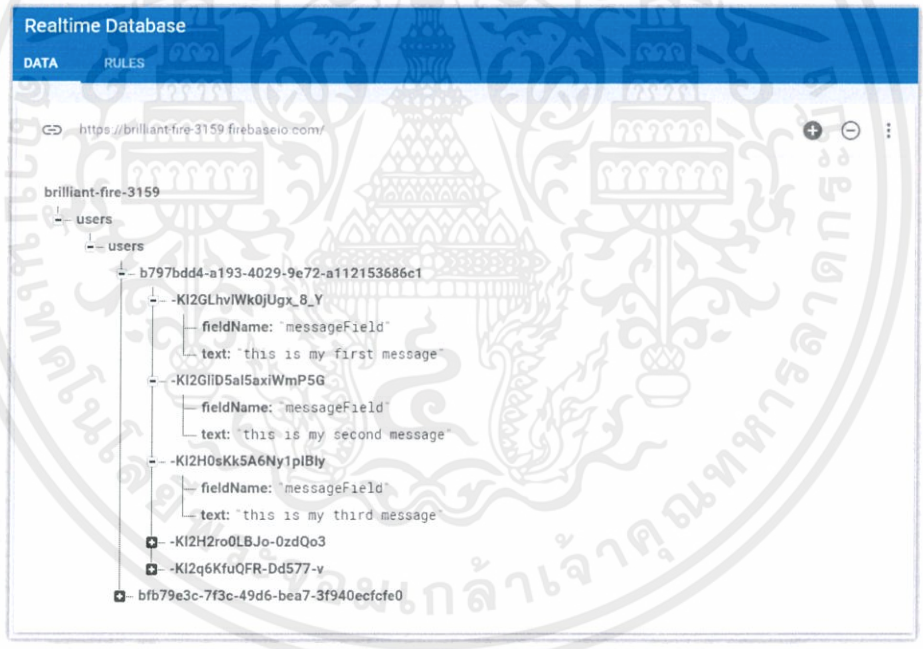
เป็น React Component Library (Open source) ที่อนุญาตให้บุคคลทั่วไปสามาถนำไปใช้งานได้ฟรี ซึ่ง Component ของ Library ออกแบบตามหลักการ Material Design ของ Google สามารถนำ Component พื้นฐานเหล่านี้ไปปรับแต่งให้เป็น Component เฉพาะทางได้

2.3.5 คลาวด์ไฟร์สโตร (Cloud Firestore)

Cloud Firestore คือบริการฐานข้อมูลแบบ NoSQL ที่จัดเก็บในรูปแบบ Document คล้ายกับ NoSQL (คล้าย ๆ MongoDB) แต่ Cloud Firestore ถูกพัฒนาอยู่บน Google Cloud Platform ทำให้สามารถเชื่อมต่อบริการต่าง ๆ ของ Google และ Firebase ได้อย่างแยบยล ในปัจจุบันยังคงเป็นเวอร์ชั่นทดสอบ (Beta version)

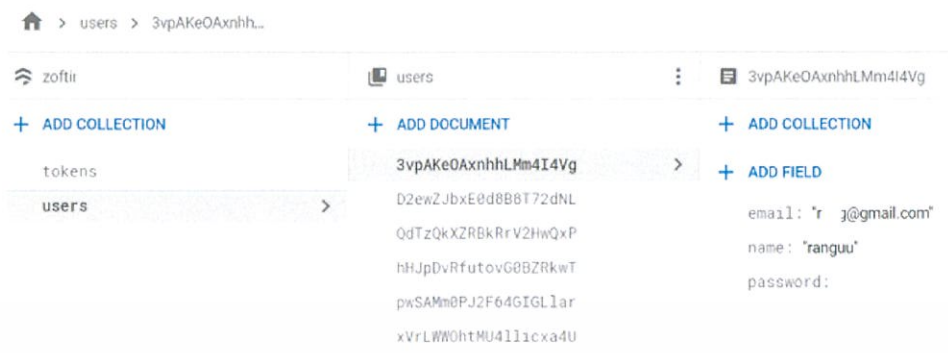
จุดเด่นของ Cloud Firestore นั้นถูกพัฒนาขึ้นเพื่อแก้ปัญหาหลาย ๆ อย่าง ของ Firebase Realtime Database ซึ่งเป็นบริการ Database ของ Google เช่นเดียวกัน ซึ่งข้อแตกต่างของ CFD (Cloud Firestore Database) สามารถเปรียบเทียบกับ RTD (Realtime Database) ได้ดังนี้

- **โครงสร้างและการค้นข้อมูลเพื่อการใช้งาน (Structure and Query)**
  - โครงสร้างของ RTD จัดเก็บเป็น Tree ขนาดมหึมาดังรูปที่ 2.4 ในขณะที่ CFD จัดเก็บข้อมูลในรูปแบบ Collection และ Document ดังรูปที่ 2.5



รูปที่ 2.4 การเก็บข้อมูลของ Realtime Database

ที่มา http://eitguide.net/firebase-qa-1/



รูปที่ 2.5 การเก็บข้อมูลของ Cloud Firestore Database

ที่มา https://firebase.googleblog.com/2018/06/sort-and-filter-in-firestore-console.html

จะเห็นว่า RTD เก็บข้อมูลเป็น Tree ซึ่งมีความซับซ้อนหากมีข้อมูลอยู่เป็นจำนวนมาก ทำให้การ Query ข้อมูลยากขึ้นตามไปด้วย แตกต่างจาก CFD ที่เก็บข้อมูลเป็น Document ภายใน Collection การ Query ข้อมูลและทำความเข้าใจโครงสร้างทำได้ง่ายกว่ามาก

- **ความสามารถในการปรับขนาดการรองรับข้อมูล (Scalable)**
  - Firebase Team ให้การรับรองว่า CFD สามารถ Scale ได้ดีกว่า RTD ไม่ว่า Data จะมีขนาดใหญ่ซักแค่ไหน
- **ง่ายต่อการดึงข้อมูลด้วยเอง (Manual fetching of data)**
  - CFD ยังคงมีความสามารถในการทำ Listener (เมื่อข้อมูลมีการเปลี่ยนแปลง จะมี Event เกิดขึ้นและถูกส่งไปยังอุปกรณ์ หน่วยทำหน้าที่คอยรับ Event เหล่านั้นเรียกว่า Listener) เพื่อ stream ข้อมูลแบบ Realtime เหมือนที่ RTD ทำได้ (ทำ Listener เพื่อ Fetch ข้อมูลอัตโนมัติเมื่อข้อมูลใน Database มีการเปลี่ยนแปลง) แต่ถ้าต้องการ Fetch ข้อมูลเอง CFD ก็ยังสามารถทำได้ดีกว่า RTD เช่นเดียวกัน
- **มีการสำรองข้อมูลไว้หลายตำแหน่ง (Multi region support)**
  - CFD จะเก็บข้อมูลไว้ในฐานข้อมูลหลายๆ Region นั่นคือ CFD อัพเดทข้อมูลเข้าทุก ๆ Region ให้เอง ถ้าหากข้อมูลที่ Region ใดพัง ก็ยังมีข้อมูลที่ Region อื่น ๆ อยู่
- **ค่าบริการ**
  - CFD จะคิดเงินจากการจำนวนการ Read/Write ที่ใช้งานไป
  - RTD จากจำนวนข้อมูลที่รับส่ง

- ข้อได้เปรียบ RTD
  - RTD เร็วกว่า CFD (ไม่ต่ำกว่า 100 Millisecond)
  - RTD มี Native Support for Presence (การแจ้งเตือนเมื่อ User online/offline เมื่อใด) ขณะที่ CFD ต้องทำเองบน Cloud Functions
  - CFD ไม่เหมาะกับการ Read/Write จำนวนมาก เพราะเสียค่าบริการมากกว่า RTD
  - RTD เป็น Production ขณะที่ CFD ยังคงเป็น Beta อยู่

2.3.6 โหนด เจเอส (Node.js)

Node.js เป็น Open Source Platform ที่เขียนด้วย JavaScript ช่วยให้สามารถเขียนโปรแกรมด้วยภาษา JavaScript ที่ฝั่ง Server (โดยปกติแล้ว JavaScript จะถูกประมวลผลที่ฝั่ง Client) เพื่อเป็น Web server ได้ (Cross Platform Runtime Environment)

2.3.7 เอ็กซ์เพรส (Express)

Express เป็น Web application framework บน Node.js มี feature ต่าง ๆ ที่ช่วยให้การทำเว็บสะดวกมากขึ้น เช่น การทำ Routing, Middleware, การจัดการ Request และ Response เป็นต้น ทำให้สามารถพัฒนาเว็บโดยใช้ Node.js ได้สะดวกขึ้น โปรเจคนี้ใช้ Express ในการสร้าง RESTful API

2.3.8 โลแดช (Lodash)

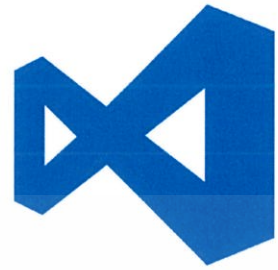
Lodash เป็น JS library ที่ช่วยให้ dev ทำงานง่ายขึ้นมาก ตัว lib (library) มีฟังก์ชันมากมายที่ช่วยจัดการข้อมูลให้ออกมาเป็นรูปแบบต่าง ๆ ที่ dev ต้องการ ตัวอย่างฟังก์ชันที่ใช้งานบ่อยเช่น reduce, groupBy, filter, every, some, keyBy etc.

2.4 เครื่องมือที่ใช้ในการพัฒนาระบบ

2.4.1 วิชวลสตูดิโอโค้ด (Visual Studio Code)

Visual Studio Code (VSCode) เป็นโปรแกรม code editor ที่ถูกริเริ่มพัฒนาจาก Microsoft จากนั้นปล่อยเป็น Open-source ให้นักพัฒนาทั่วโลกสามารถร่วมพัฒนา Software ได้ สามารถใช้งานได้ฟรี ไม่คิดค่าใช้จ่าย มีส่วนเสริม (Extension) แยกติดตั้งความสามารถเฉพาะได้ ทำให้สามารถปรับแต่งคุณสมบัติโปรแกรมเฉพาะส่วน ให้มีความสามารถตามที่ต้องการ ลบคุณสมบัติที่ไม่จำเป็นและไม่ต้องการออกได้ รองรับการใช้งานข้ามแพลตฟอร์ม ทั้ง Windows, OSX และ Linux สนับสนุนภาษา JavaScript,

TypeScript, NodeJs etc. เชื่อมต่อกับ Git และมี Terminal ภายในตัวโปรแกรมเอง โดยใช้สัญลักษณ์ของโปรแกรมดังรูปที่ 2.6



รูปที่ 2.6 สัญลักษณ์ VSCode

2.4.2 ซอร์สทรี (Source Tree)

Source Tree เป็น Software ที่ช่วยจัดการ Git repository (เหมือน Folder เก็บข้อมูลของ Project ที่อยู่บน Git system) ได้เหมือน command line เพียงแต่ Source Tree มี GUI (Graphic User Interface) ที่สวยงามและเป็นมิตรกับผู้ใช้งานมากกว่า command line มาก สามารถตรวจสอบการกระทำ (Activity), ผู้กระทำ, Time-line ที่เกิด Activity ต่าง ๆ ที่เกิดขึ้นกับ Git repository ผ่าน GUI ของ Software ได้ ทำให้ Project development เป็นไปอย่างลื่นไหลและปลอดภัย โดยใช้สัญลักษณ์ของโปรแกรมดังรูปที่ 2.7

รูปที่ 2.7 สัญลักษณ์ Source Tree

2.4.3 กูเกิลโครม (Google Chrome)

Google Chrome คือ โปรแกรมเว็บบราวเซอร์ (Web browser) ที่ใช้สำหรับเปิดเว็บไซต์ โดยมี Google เป็นผู้พัฒนา ปัจจุบันเป็นที่นิยมของผู้ใช้งาน Internet เป็นอย่างมากเพราะมีความภัยสูง สามารถ

ปรับแต่งธีม หรือเสริมความสามารถอื่นได้จากการติดตั้ง โปรแกรมเสริม (Extension) ที่มีอยู่อย่างมากมาย ใน Store

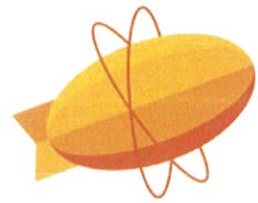
อีกหนึ่งความน่าใช้ของ Google Chrome คือความเป็นมิตรต่อนักพัฒนา (Developer) อย่าง Inspect tool ที่ถือเป็นหัวใจหลักสำหรับนักพัฒนาเว็บไซต์ที่หันมาใช้งาน Google Chrome ซึ่งมีความสามารถในการ Debug code, ตรวจสอบ Network traffic และความสามารถน่าใช้อื่น ๆ ซ่อนอยู่อีกมากมาย โดยใช้สัญลักษณ์ของโปรแกรมดังรูปที่ 2.8

รูปที่ 2.8 สัญลักษณ์ Google Chrome

2.4.4 เซปลิ้น (Zeplin)

ปกติแล้วการสร้างหน้าเว็บไซต์ มีการทำงานร่วมกันระหว่าง Dev (Developer) กับ Designer โดย Designer ออกแบบหน้าเว็บไซต์ที่สมบูรณ์แล้ว Share งาน Design ให้ Dev ไปเขียนหน้าเว็บไซต์ให้ตรงตาม Design ปัญหาที่เกิดคือ เวลา Dev เปิดดูงาน Design แล้วเผลอไปแก้งานโดยไม่ได้ตั้งใจ ทำให้งาน Design เพี้ยนจากที่เคยเป็น (ตัวอย่างเช่น แชร์ไฟล์ Sketch ให้ Dev ไปแกะเอง)

Zeplin เป็นเครื่องมือสำหรับให้ Designer ได้ Share งาน Design ไปยัง Dev ด้วยวิธีง่ายๆ และ Update ได้ตลอดเวลา เพราะมี Plugin เอาไว้ให้ไปติดตั้งใส่ Sketch ส่วน Dev ก็เปิดงานผ่านโปรแกรม Zeplin ได้เลย ซึ่ง Dev จะแก้ไขงานไม่ได้ ดูได้อย่างเดียวเท่านั้น แต่ตัว Zeplin ก็ support การทำงานของ Dev เช่นกัน เพราะสามารถ Generate ภาพ , ตัวอักษร, ขนาด , สี ออกมาเป็น code เลย เพื่อให้ทาง Front-End และ Dev นำ code ไปพัฒนาต่อได้ โดยไม่ต้องมานั่งเขียนเองใหม่ทั้งหมด รวมถึงรูปภาพต่าง ๆ ที่เซฟออกมาได้เลย หรือรูปไอคอนต่าง ๆ ที่ระบบจะแปลงออกมาให้หลายๆขนาด เพื่อนำไปเลือกใช้ได้ (มี Extension เสริมมากมายในเลือกใช้ง่ายต่อ Dev และ Designer) โดยใช้สัญลักษณ์ของโปรแกรมดังรูปที่ 2.9



รูปที่ 2.9 สัญลักษณ์ Zeplin

2.4.5 โพสแมน (Postman)

Postman เป็นเครื่องมือที่ใช้ในการพัฒนา RESTful API และการทำงานของ Service รวมถึงการ Mock Service มี GUI ที่สวยงาม และงานง่าย สามารถจำลอง Request เพื่อทดสอบผลการทำงานของ API มีสภาพแวดล้อมที่ช่วยสนับสนุนการทดสอบ API flow สามารถใช้ Test script เพื่อทดสอบ API อัตโนมัติและรายงานผลการทดสอบเมื่อจบกระบวนการ โดยใช้สัญลักษณ์ของโปรแกรมดังรูปที่ 2.10

รูปที่ 2.10 สัญลักษณ์ Postman

บทที่ 3
ขั้นตอนการดำเนินงาน

วิธีการดำเนินสามารถแบ่งออกได้เป็น 4 ส่วนด้วยกันคือ

- จัดเตรียมเครื่องมือและอุปกรณ์ที่ใช้
- การออกแบบและวิเคราะห์ระบบ
- การดำเนินการพัฒนาระบบ
- ทดสอบระบบ

3.1 จัดเตรียมเครื่องมือและอุปกรณ์ที่ใช้

เครื่องมือที่ใช้เพื่อพัฒนาระบบมีดังนี้

1. Node.js ใช้เพื่อการจัดการสภาพแวดล้อม (Environment) ในการพัฒนาระบบ API ทำหน้าที่ในการจำลองสภาวะ Web server ในเครื่องของนักพัฒนาโดยไม่ต้องใช้เซิร์ฟเวอร์จริงทดสอบ
2. Google Chrome เครื่องมือหลักสำหรับนักพัฒนาเว็บไซต์ (Website developers) ใช้ในการแสดงผลลัพธ์ที่ได้จากการพัฒนา มีเครื่องมือและ Environment ที่ช่วยสนับสนุน Developer อย่าง Inspect tools ทำให้กระบวนการพัฒนาระบบเป็นไปได้อย่างรวดเร็ว
3. Source Tree เครื่องมือจัดการ Git repository เพื่อการสำรองและอัพเดทข้อมูลของโครงงาน (Project) สามารถตรวจสอบและติดตามการเปลี่ยนแปลงที่เกิดจาก Developer คนอื่น ๆ ทำให้กระบวนการพัฒนาระบบเป็นไปได้อย่างราบลื่น รวดเร็ว และปลอดภัย
4. Zeplin เครื่องมือที่ใช้ประสานงานของ Developer กับ Designer มีความสามารถในการแสดงรายละเอียดการตกแต่งหน้าเว็บ (CSS code) ทำให้ Developer ไม่ต้องเสียเวลาในการตรวจสอบสีหรือขนาดของวัตถุใหม่จากงาน Design
5. Postman เครื่องมือทดสอบการทำงานของ RESTful API ที่ใช้ในระบบของ Pumpkin-CRM สามารถจัดเก็บ Script ที่ใช้ทดสอบเป็น Collection ทำให้กระบวนการทำงานเป็นระบบ และสามารถตั้งค่า Environment ในตัวโปรแกรม ให้เหมาะสมกับสถาการณ์ทดสอบ ช่วยลดระยะเวลาและขั้นตอนการทดสอบลงได้
6. VSCode เครื่องมือหลักในการพัฒนา เป็น code editor ที่มี Extension มากมายที่ช่วยในการทำงานกับ code เป็นเรื่องสนุกและง่ายยิ่งขึ้น

3.2 การออกแบบและวิเคราะห์ระบบ

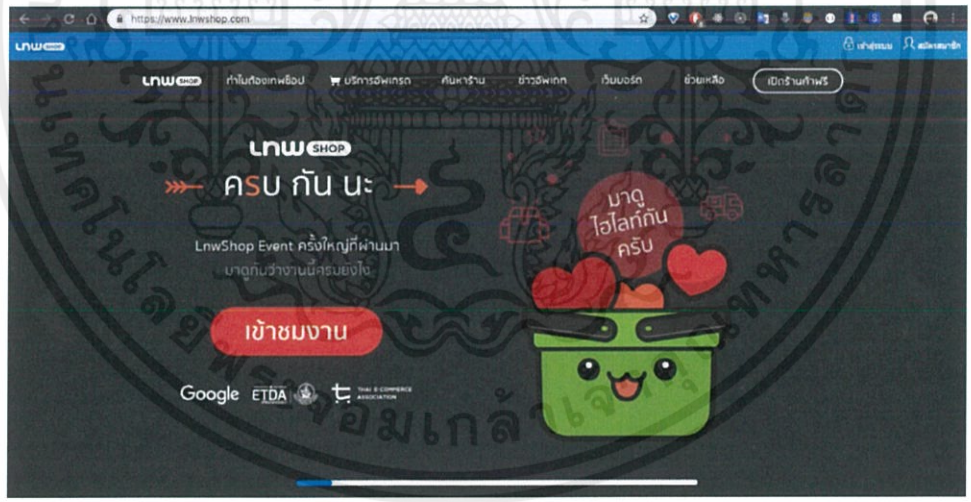
3.2.1 สาเหตุของปัญหา

โดยปกติแล้ว Pumpkin-Dashboard อนุญาตให้แอดมินของร้านค้าที่เข้าร่วมระบบ Pumpkin-CRM สามารถใช้งาน feature ทั้งหมดของแดชบอร์ด ได้โดยไม่มีการจำกัดจำนวนครั้ง เช่น สามารถสร้าง Campaign, Survey แล้วส่ง Notification ไปหาลูกค้าของร้านได้ไม่จำกัดจำนวน เป็นต้น ซึ่งการเรียกใช้ API แต่ละครั้งนั้น มีค่าใช้จ่ายจากการเรียกใช้บริการ API ของ Google Cloud Firestore ทำให้บริษัทขาดทุนจากรายจ่ายส่วนนั้นไป ทางบริษัทเล็งเห็นว่าการควรมีการเก็บค่าบริการในการใช้งาน feature ของ Pumpkin-Dashboard เหล่านั้น เพื่อสามารถนำมาเป็นค่าใช้จ่ายไม่ให้ระบบขาดทุนนั่นเอง

3.2.2 สืบค้นข้อมูล (Research)

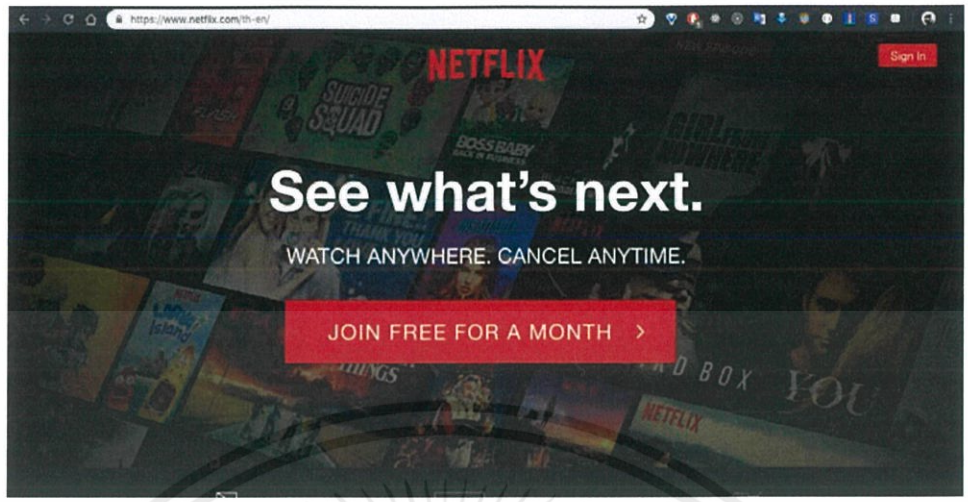
จากสาเหตุที่กล่าวมาเบื้องต้น ผู้จัดทำจึงเริ่มสืบค้นข้อมูล “ระบบชำระค่าบริการ” ในตลาดผ่านเว็บไซต์เพื่อนำมาเป็นต้นแบบในการพัฒนา “ระบบชำระค่าบริการ” สำหรับ Pumpkin-Dashboard สุดท้ายแล้วสามารถสรุปรายชื่อเว็บไซต์ที่ใช้เป็นระบบต้นแบบได้ดังนี้

- **LnwSHOP** ผู้ให้บริการ e-commerce platform แบบสำเร็จรูป ผู้ใช้สามารถปรับแต่งหน้าตาของเว็บไซต์ได้ในระดับหนึ่ง ดังรูปที่ 3.1 แสดงหน้าเว็บเริ่มต้นก่อนเข้าใช้บริการ



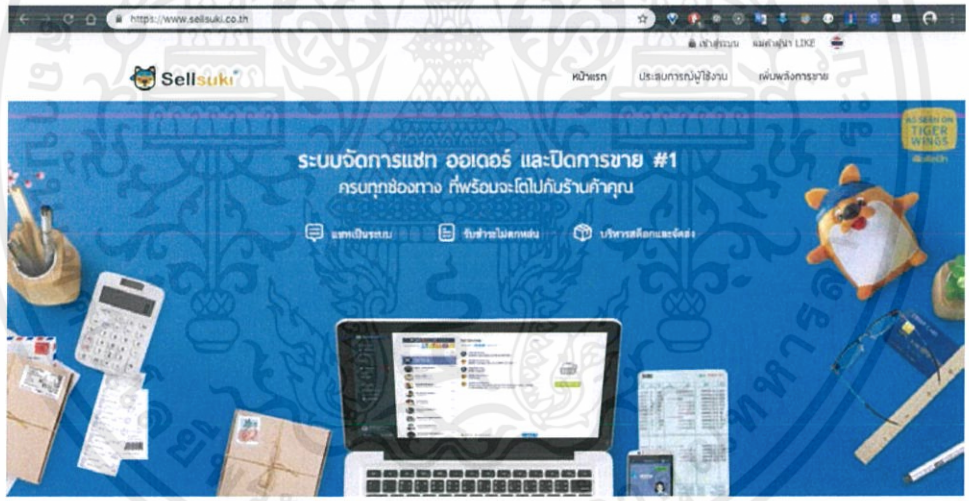
รูปที่ 3.1 เว็บไซต์ของ LnwSHOP

- **Netflix** บริการสตรีมมิ่งที่ให้ลูกค้าสามารถรับชมเนื้อหาความบันเทิงหลากหลายสำหรับ รายการทีวี ภาพยนตร์ สารคดีที่ชนะรางวัลและอื่น ๆ อีกมากมาย ในอุปกรณ์ต่าง ๆ ที่เชื่อมต่ออินเทอร์เน็ต ดังรูปที่ 3.2 แสดงหน้าเว็บเริ่มต้นก่อนเข้าใช้งาน



รูปที่ 3.2 เว็บไซต์ของ Netflix

- **SellSUKI** ผู้ให้บริการ e-commerce platform แบบสำเร็จรูป มีแดชบอร์ดสำหรับตรวจสอบรายงาน จัดการรายละเอียดการซื้อขาย มีระบบคลังสินค้า ระบบสต๊อกสินค้า ระบบติดต่อลูกค้า ฯลฯ ดังรูปที่ 3.3 แสดงหน้าเว็บเริ่มต้นก่อนใช้เข้าบริการ



รูปที่ 3.3 เว็บไซต์ของ SellSUKI

3.2.3 วิเคราะห์ข้อมูลที่สืบค้น

จากการวิเคราะห์ระบบของเว็บไซต์ทั้ง 3 พบว่า การเก็บค่าบริการต่าง ๆ ของเว็บไซต์ต้นแบบจะต้องมีสินค้า (Products) ในระบบก่อน ตัวอย่างเช่น สินค้าค้าของ SellSUKI และ LnwSHOP (สาเหตุที่เลือกใช้ทั้ง 2 เว็บไซต์ เนื่องจากมีแนวทางคล้ายระบบของ Pumpkin-CRM) คือ feature ของ platform

การเก็บค่าบริการต่าง ๆ จะนับจากจำนวนโควตาการใช้งาน feature ที่มีอยู่ในระบบ ซึ่งจำนวนโควตาการใช้งานจะถูกเพิ่มตามระดับราคาของ Package ที่ถูกขาย ดังรูปที่ 3.4

ระบบ Package คือ การกำหนดราคาการใช้งานหลาย ๆ feature แบบเหมาจ่าย ซึ่งถ้าเทียบราคาโควตาต่อหน่วยของเฉพาะ feature การซื้อแบบ Package จะราคาถูกกว่า แลกกับการที่โควตาของ Package มีจำกัด ถ้ามีการใช้งาน feature ใดบ่อย ๆ จนหมดโควตา จะไม่สามารถซื้อเพิ่มเฉพาะได้ ต้องเปลี่ยนเป็น Package ที่มีราคาสูงขึ้น เพื่อจะได้โควตาของ feature มากขึ้น

รูปที่ 3.4 ระบบ Package ของ SellSUKI

3.2.4 ออกแบบระบบสินค้า (Pumpkin-Dashboard's products)

เมื่อได้ผลวิเคราะห์จากหัวข้อก่อนหน้านี้มา ทีมผู้พัฒนา ("ทีมผู้พัฒนา" ในที่นี้หมายถึงทีมผู้พัฒนา Pumpkin-CRM)จึงนำแนวคิดที่ได้ มาออกแบบระบบชำระค่าบริการของ Pumpkin-Dashboard (จากนี้ไปคำว่า แดชบอร์ด สื่อถึง Pumpkin-Dashboard) ดังนี้

ทีมผู้พัฒนาปรับปรุงระบบต้นแบบที่ได้จากการวิเคราะห์หัวข้อก่อนหน้า โดยกำหนดให้ในระบบของแดชบอร์ดจะมี 2 ทางเลือกในการเพิ่มจำนวนโควตาการใช้งาน feature ทำให้แดชบอร์ดมีระบบสินค้า (Pumpkin-Dashboard's products) แบ่งออกเป็น 2 ประเภท คือระบบ Products และ ระบบ Package

ระบบ Package (Package of Pumpkin-Dashboard's products) ทีมผู้พัฒนายังคงใช้แนวทางที่ได้จากการวิเคราะห์ระบบของ SellSUKI และ LnwSHOP นั่นคือ การเพิ่มจำนวนโควตาการใช้งาน feature ของแดชบอร์ด จะสามารถเพิ่มได้โดยการเลื่อนระดับ Package เป็นระดับสูงขึ้น ซึ่งจะมีผลให้ได้รับโควตาการใช้งาน feature ของแดชบอร์ด โดยรวมเพิ่มขึ้นด้วย แต่ด้วยความที่ว่าระบบ Pumpkin ไม่ต้องการจำกัดทางเลือกให้แก่ผู้ใช้งานมากเกินไป ("ผู้ใช้" ที่สามารถใช้งานแดชบอร์ด ได้ คือแอดมินของร้านค้าที่ลงทะเบียนใช้งาน Pumpkin-CRM อธิบายคือ แอดมินของร้านค้าที่นำระบบ Pumpkin-CRM ไปติดตั้ง) ทีมผู้พัฒนาจึงออกแบบอีกระบบหนึ่งขึ้นมา นั่นคือ

ระบบ Products (Products of Pumpkin-Dashboard's products) เป็นอีกทางเลือกหนึ่งในการเพิ่มจำนวนโควตาการใช้งานแดชบอร์ด สามารถเพิ่มโควตาเฉพาะ feature ที่ต้องการได้ โดยไม่จำเป็นต้องเปลี่ยนระดับของ Package ทำให้ผู้ใช้รู้สึกไม่โดนเอาเปรียบเกินไป (Fair to users)

3.2.5 ออกแบบระบบชำระค่าบริการ

การออกแบบการชำระค่าการนั้น แบ่งรูปแบบการชำระค่าบริการตามระบบสินค้าของแดชบอร์ด สามารถอธิบายรายละเอียดได้ดังนี้

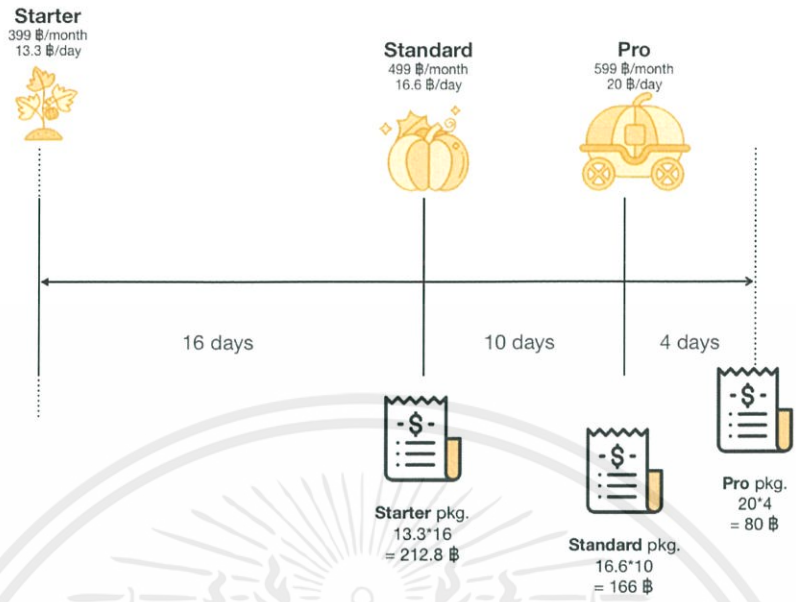
3.2.5.1 ออกแบบระบบชำระค่าบริการสินค้าระบบ Package

ในแรกเริ่มระบบกำหนดให้ร้านค้าทุกร้านที่ใช้งาน Pumpkin-CRM มี Package พื้นฐานเป็น Free package การคิดค่าบริการของสินค้าระบบ Package จะคิดค่าบริการเป็นรายเดือน เมื่อชำระค่าบริการเสร็จสิ้น จะรีเซทจำนวนโควตา feature ที่เคยใช้ไปก่อนหน้านี้ใหม่ จำนวนค่าบริการคิดแบบ "ยุติธรรมต่อผู้ใช้ (Fair to users)" นั่นคือ คิดค่าบริการตามจำนวนที่ใช้งานไป โดยจำนวนการใช้งานของ Package คิดตามจำนวนวันที่ใช้งาน Package นั้นอยู่ ยกตัวอย่างเช่น

สมมุติเหตุการณ์ให้ ในรอบบิลใหม่ กำหนดให้ 1 รอบบิลมีระยะเวลา 1 เดือน หรือ 30 วัน ร้านค้าใด ๆ ในระบบใช้งาน Starter package ที่มีค่าบริการ 399 บาทต่อเดือน เมื่อเวลาผ่านไป 16 วัน ร้านค้าเปลี่ยนไปใช้งาน Standard package ที่มีค่าบริการ 499 บาทต่อเดือน ระบบจะสร้าง Quotation (ข้อมูลค่าบริการที่ใช้ไปก่อนหน้าที่จะเปลี่ยน Package ซึ่งจะถูกบันทึกไว้ในฐานข้อมูลหรือ Database) ซึ่งคิดค่าบริการตามจำนวนวันที่ใช้ไป ต่อมาเมื่อเวลาผ่านไปอีก 10 วัน ร้านค้าเปลี่ยนเป็น Professional package ค่าบริการ 599 บาทต่อเดือน ระบบสร้าง Quotation อันที่ 2 เก็บไว้ (ซึ่งเป็นค่าบริการที่ใช้งาน Standard package เป็นระยะเวลา 10 วัน) เมื่อสิ้นรอบบิล ระบบสร้าง Quotation สุดท้ายที่เป็นค่าบริการจาก Package ล่าสุดออกมา ซึ่งสุดท้ายแล้วค่าบริการที่ต้องชำระในรอบบิลนี้สามารถคำนวณได้ตามตารางที่ 3.1 และสามารถอธิบายเหตุการณ์ได้ตามรูปที่ 3.5

ตารางที่ 3.1 ตารางคำนวณตัวอย่างการคิดค่าบริการรายเดือนระบบ Package

Package	ราคาต่อเดือน	ราคาต่อวัน	จำนวนวันที่ใช้	ค่าบริการ(บาท)
Starter	399	13.3	16	212.8
Standard	499	16.6	10	166
Professional	599	20	4	80
รวม				**458.8**



รูปที่ 3.5 แผนภาพแสดงเหตุการณ์ตัวอย่างการคิดค่าบริการรายเดือนระบบ Package

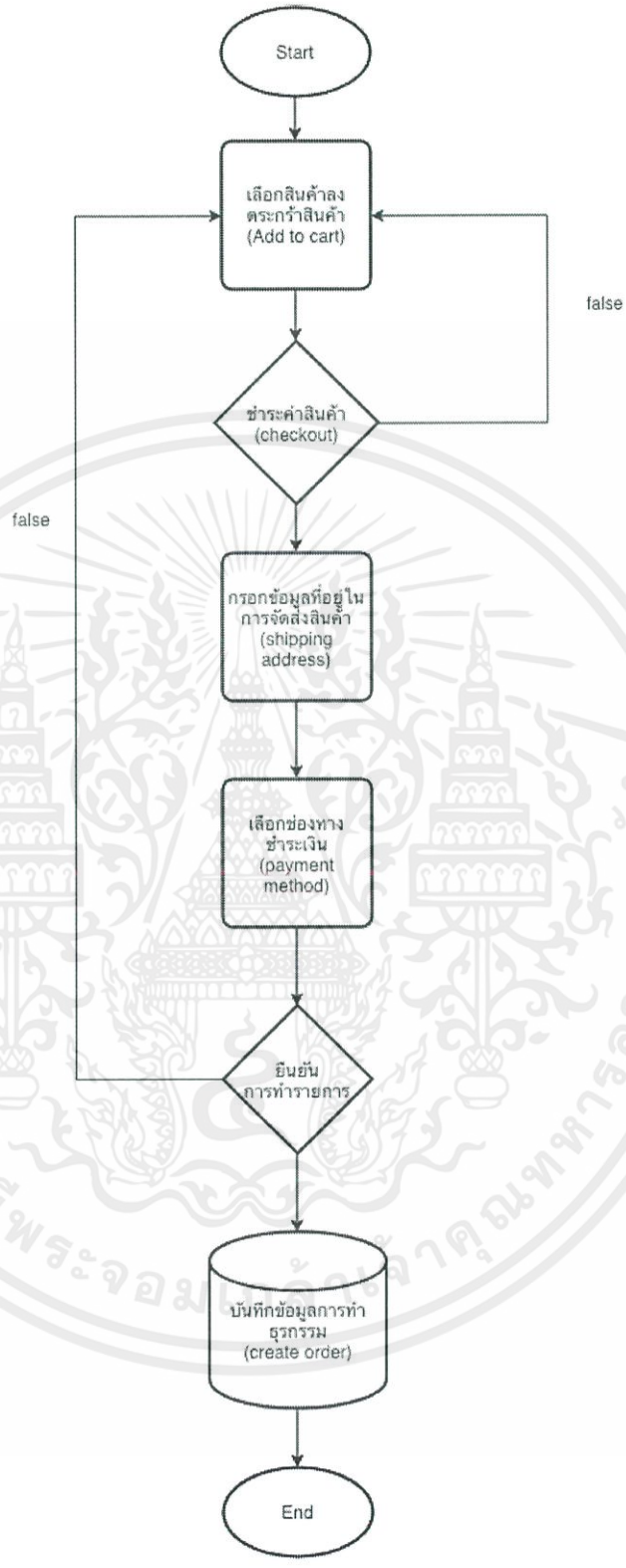
เมื่อได้ค่าบริการทั้งหมดของรอบบิลนี้แล้ว ระบบจะสร้าง order ขึ้น เพื่อออกเอกสารเก็บค่าบริการตามรูปแบบของบริษัทต่อไป เมื่อชำระค่าบริการเสร็จสิ้น ระบบจึงจะรีเซทโควตาการใช้งาน feature ที่เคยใช้ไปก่อนหน้านี้ใหม่ ตามประเภทของ feature ตัวอย่างเช่น “Notification feature” ที่ใน Starter package ได้โควตาส่ง Notification ให้ลูกค้าของร้านได้ 500 คน เมื่อใช้โควตาครบแล้วก็ไม่สามารถส่งได้อีก อย่างไรก็ตาม feature นี้จะถูกรีเซทโควตาเมื่อชำระค่าบริการเสร็จสิ้น แต่บางประเภทจะก็ไม่ถูกรีเซทโควตา ตัวอย่างเช่น “Customer group feature” ที่ใช้ในการแบ่งกลุ่มลูกค้าที่อยู่ในระบบของร้านค้าเพื่อจัดระดับกลุ่ม เช่น กลุ่มระดับธรรมดา กลุ่มระดับ VIP เป็นต้น ซึ่งจำนวนกลุ่มลูกค้าที่ทางร้านจัดไว้จะเพิ่มจำนวนโควตาขึ้นก็ต่อเมื่อแอดมินลบ “กลุ่มลูกค้า” ออกจากระบบด้วยตัวเองเท่านั้น จะสังเกตเหตุว่าในบาง feature ที่เกิดจากการกระทำของแอดมินต่อระบบโดยตรงไม่สามารถบังคับลดจำนวนได้ เช่น หากรีเซทโควตาของ “Customer group feature” ก็จำเป็นต้องบังคับลบกลุ่มลูกค้าที่แอดมินเคยสร้างไว้ก่อนหน้านี้ออกไปด้วย ซึ่งไม่สมเหตุสมผลที่จะทำให้เกิดเหตุการณ์นี้ขึ้น

3.2.5.2 ออกแบบระบบชำระค่าบริการสินค้าระบบ Products

ด้วยความที่ว่าแนวทางของบริษัทมีความคุ้นชินกับระบบ e-commerce เป็นอย่างมาก ดังนั้นในระบบ Products นี้ จึงใช้ Checkout flow ของระบบ e-commerce

ทีมผู้พัฒนามองว่า feature แต่ละอย่างของแดชบอร์ดคือสินค้าชนิดหนึ่ง ซึ่งการเลือกเพิ่มจำนวนโควตาก็เหมือนการเลือกซื้อสินค้าในจำนวนต่าง ๆ ขั้นตอนการชำระค่าบริการสามารถใช้แนวทางการชำระเงินของ e-commerce website ทั่วไปได้ ส่วนการเลือกช่องทางการชำระเงิน (Payment methods) เพื่อลดความยุ่งยากของรูปแบบของช่องทางการชำระเงินลง ในโปรเจคสหกิจนี้จึงกำหนดให้ใช้การชำระค่าบริการผ่านช่องทางบัตรเครดิต/เดบิตเพียงอย่างเดียวแทน และกำหนดให้ต้องเชื่อมต่อกับผู้ให้บริการชำระเงินบุคคลที่สาม (3rd-party payment gateway) เพื่อเก็บค่าบริการจากบัตรเครดิต/เดบิต ได้ทันที (ชำระค่าบริการแบบชำระทันที) ซึ่งทีมผู้พัฒนาเลือกใช้บริการของ OMISE เป็นผู้ให้บริการ

สามารถอธิบายขั้นตอนกระบวนการชำระสินค้าของระบบ e-commerce คร่าว ๆ ได้จากรูปที่ 3.6



รูปที่ 3.6 Flowchart แสดง Checkout flow ของระบบ e-commerce

การซื้อโควตา feature เฉพาะเหล่านี้ จะถูกใช้แล้วหมดไป ไม่มีการรีเซทโควตาเหมือนระบบของ Package ซึ่งหากไม่สามารถใช้หมดภายในเดือนที่ซื้อ ก็ยังสามารถเก็บไว้ใช้ต่อไป (ซื้อขาดในครั้งเดียว) ถือเป็นโควตาสำรองที่เอาไว้ใช้ฉุกเฉินตอนโควตา feature ของ Package หมดไปแล้วได้ ซึ่ง feature เหล่านี้ถูกเรียกว่า Top up products

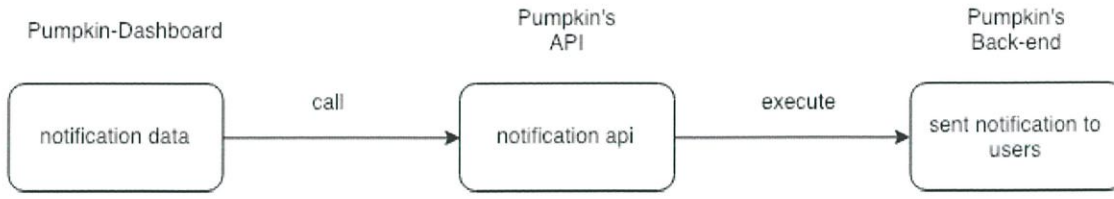
อย่างไรก็ตาม จากที่ทราบจากหัวข้อก่อนหน้านี้ว่า ในระบบ Package มี feature บางประเภทจะก็ไม่ถูกรีเซทโควตา ซึ่ง feature เหล่านี้ก็สามารถซื้อโควตาเฉพาะได้เช่นเดียวกัน ทำให้เกิดการสะสมเช่นเดียวกับ Top up products ซึ่งหากคิดค่าบริการแบบเดียวกันก็จะเกิดเหตุการณ์ที่ผู้ใช้ เลือกใช้งาน Package ระดับต่ำ แต่เลือกซื้อโควตาของ feature ประเภทนี้เอาไว้จำนวนมาก ด้วยความที่ว่าเป็นการซื้อขาดในครั้งเดียว ทำให้เกิดเหตุการณ์ที่ว่าจ่ายครั้งเดียวใช้ตลอดชีพ ซึ่งจะทำให้ระบบนี้ไม่สอดคล้องกับระบบ Package (ในระบบ Package คิดค่าบริการโควตาของ feature ประเภทนี้ทุกเดือนอยู่แล้ว) จึงแยกคิดค่าบริการของ feature ประเภทนี้ ให้เป็นไปตามระบบรูปแบบการคิดค่าบริการของระบบ Package เรียก feature ประเภทนี้ว่า Recurring products หรือ Subscription products นั่นเอง (feature เหล่านี้คิดค่าบริการเป็นต่อหน่วยต่อเดือน)

การคิดค่าบริการของ Recurring products จะยอมให้ผู้ใช้สามารถเพิ่มจำนวนโควตาได้ไม่จำกัด (คิดค่าบริการเมื่อสิ้นรอบบิล) ซึ่งคิดค่าบริการตามจำนวนวันที่มีโควตาประเภทนี้ในระบบของร้านแม้จะไม่ได้ใช้งานโควตาส่วนนี้เลยก็ตาม (ไม่คิดโควตาที่ได้มาจาก Package) แอดมินสามารถลดโควตาลงได้หากต้องการ ระบบจะสร้าง Quotation แบบเดียวกับการคิดค่าบริการของ Package ตามสมการที่ 1

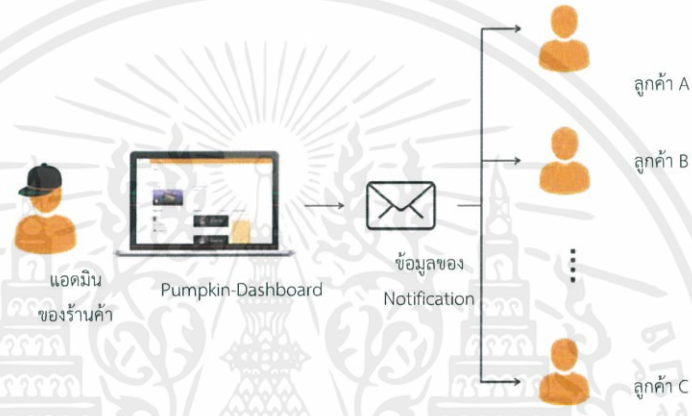
สมการที่ 1 ค่าบริการ Recurring product ใด ๆ = ราคาต่อหน่วย x จำนวนโควตา
x จำนวนวันที่มีโควตาอยู่ในระบบ

3.2.6 การรวมกับระบบเดิม

ขอยกกรณีการส่ง Notification เพื่อให้เข้าใจ Workflow คร่าวๆของแดชบอร์ด ดังนี้ หลังจากข้อมูล Notification ที่ต้องการส่งหาลูกค้าถูกฝังกับ Request body เมื่อเรียกใช้งาน Notification API (RESTful API) แล้ว API จะนำข้อมูลที่ได้จาก Request ไปจัดการตาม Logic ของ API ที่ Pumpkin's Back-end สุดท้ายแล้ว ลูกค้าทุกคนที่มีชื่ออยู่ในเป้าหมายของ Notification จะได้รับข้อความในที่สุด อธิบายด้วยรูปที่ 3.7 และ 3.8



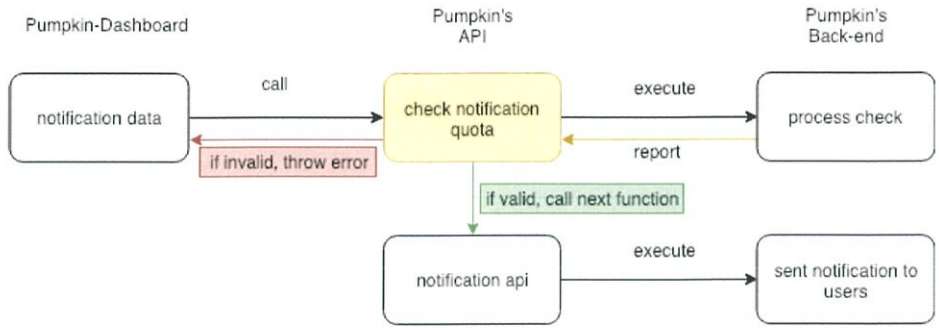
รูปที่ 3.7 Block diagram การทำงานของ Notification feature



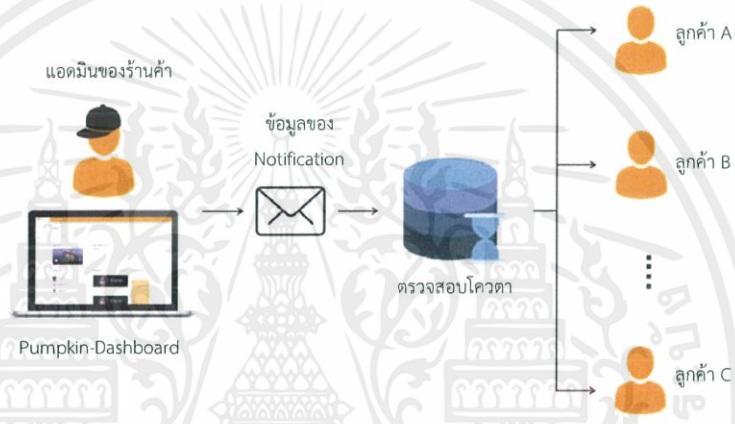
รูปที่ 3.8 แผนภาพแสดงการส่ง Notification ไปหาลูกค้า

ซึ่งเมื่อมีการนำระบบ Package และ Products เข้าไปรวมกับระบบเดิม ทำให้ต้องสนใจเรื่องโควตาการใช้งาน feature ต่าง ๆ ของแดชบอร์ด จึงต้องมีการปรับ Workflow เดิม ให้เข้ากับระบบใหม่ด้วย

ทีมผู้พัฒนาจึงปรับให้ทุกครั้งที่เรียกใช้งาน Pumpkin's API ที่เกี่ยวข้องกับจำนวนโควตาการใช้งาน feature ต้องมีการตรวจสอบ "โควตาที่เหลือ" ของร้านค้าก่อน ถ้ามีโควตาเพียงพอก็สามารถเรียกใช้งานได้ปกติ แต่ถ้ามีโควตาไม่เพียงพอ จะต้องแจ้งให้ผู้ใช้ซื้อโควตาเพิ่มจากระบบของ Pumpkin-Dashboard สามารถอธิบายรูปแบบที่ถูกปรับใหม่ได้จากภาพที่ 3.9 และ 3.10



รูปที่ 3.9 Block diagram การทำงานของ Notification feature แบบใหม่



รูปที่ 3.10 แผนภาพแสดงการส่ง Notification ไปหาลูกค้าแบบใหม่

3.3 การดำเนินการพัฒนาระบบ

3.3.1 ศึกษาและทำความเข้าใจความต้องการของระบบ

สรุปความต้องการของระบบชำระค่าบริการ (Requirement of Billing system)

ระบบสินค้าแบบ Products

- ระบบแยกประเภทของ Products เป็น Top up และ Recurring
- สามารถซื้อโควตาเฉพาะ feature ที่ต้องการได้
- สามารถ เพิ่ม/ลด โควตาของ Recurring products ได้
- ใช้รูปแบบชำระค่าบริการแบบชำระทันที ตามรูปแบบ e-commerce's checkout flow
- ใช้ช่องทางชำระค่าบริการ (payment method) เป็นบัตรเครดิต/เดบิต เพียงอย่างเดียวเท่านั้น
- ใช้ OMISE payment gateway เป็น 3rd-party payment gateway
- คิดค่าบริการ Recurring products ได้ถูกต้อง

ระบบสินค้าแบบ Package

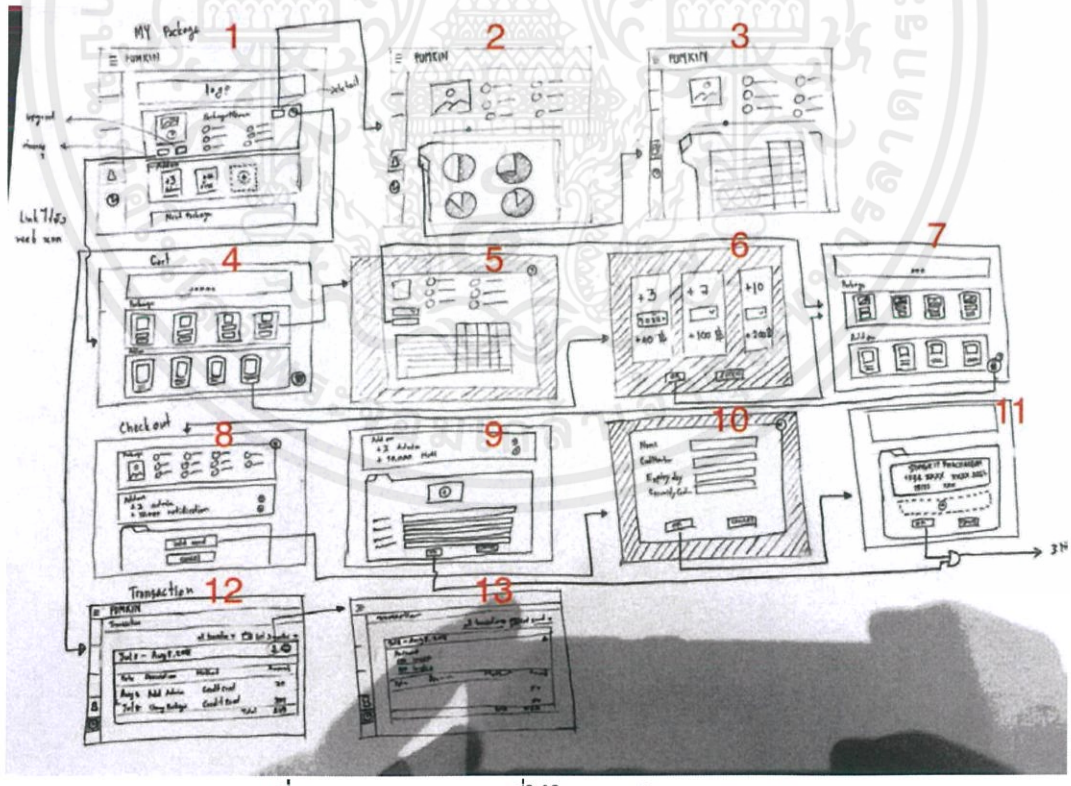
- สามารถเปลี่ยนระดับ Package ได้
- ใช้รูปแบบชำระค่าบริการแบบรายเดือน มีระบบ Quotation ในการคิดค่าบริการแบบยุติธรรมต่อผู้ใช้ (Fair to users)
- สามารถหักค่าบริการอัตโนมัติ(auto payment) จากบัตรเครดิต/เดบิตที่ลงทะเบียนเอาไว้

การรวมกับระบบเดิม

- ผู้ใช้ต้องสามารถตรวจสอบจำนวนโควตาที่คงเหลืออยู่ของตัวเองได้ (มีหน้าแสดงสถานะปัจจุบัน)
- ต้องมีการตรวจสอบโควตาคงเหลือของ feature ใด ๆ ก่อนที่ถูกเรียกใช้งาน (Block API)

3.3.2 ออกแบบ UI/UX

- เพื่อให้ Designer เข้าใจสิ่งที่ทีมผู้พัฒนาต้องการ จึงใช้ Wireframe (การ Design หน้า UI ที่ควรมีใน Workflow และแสดงเส้นเชื่อมต่อหน้าถัดไปเมื่อเกิดการกดปุ่มภายในหน้านั้น ๆ) ในการอธิบาย Workflow ของระบบ ดังรูปที่ 3.11



รูปที่ 3.11 Wireframe ที่ใช้ในการอธิบาย Workflow

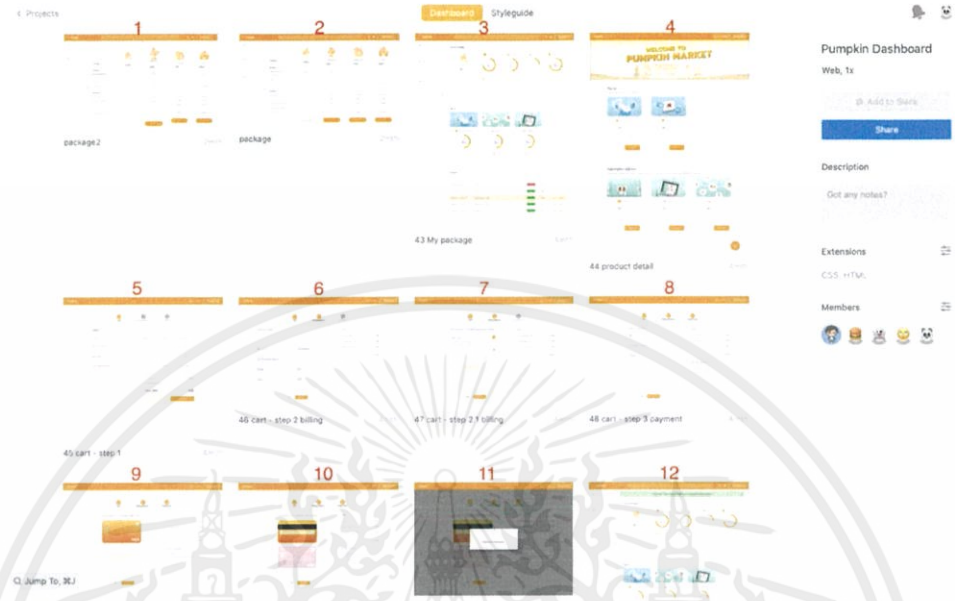
หมายเลขที่ 1-3 หน้า “แพ็กเกจของฉัน” หรือ “My Package”
หมายเลขที่ 1 ส่วนแสดงข้อมูล Package ปัจจุบัน และ Products ที่ซื้อโควตาเพิ่ม
หมายเลขที่ 2 หน้าแสดงข้อมูล Package แบบละเอียด แสดงข้อมูลเป็นพายกราฟ (Pie graph)
หมายเลขที่ 3 หน้าแสดงข้อมูล Package แบบละเอียด แสดงข้อมูลเป็นตาราง

หมายเลขที่ 4-7 หน้า “สินค้า” หรือ “Cart”
หมายเลขที่ 4 แสดง Products ที่สามารถซื้อโควตาจากระบบเพิ่มได้
หมายเลขที่ 5 Modal แสดงข้อมูลรายละเอียดของ Product ชิ้นที่สนใจ
หมายเลขที่ 6 Modal แสดงข้อมูลรายละเอียดของ Package ที่สนใจ
หมายเลขที่ 7 เมื่อกดซื้อ Package/Product จากปุ่ม “Add to cart” ใน Modal จะแสดง Floating button ที่มี icon บอกจำนวนสินค้าในตะกร้าสินค้า

หมายเลขที่ 8-11 หน้า “ชำระค่าบริการ” หรือ “Checkout”
หมายเลขที่ 8 แสดงข้อมูลรายละเอียดตะกร้าสินค้าว่ามีสินค้าอะไรบ้าง
หมายเลขที่ 9 หน้าเลือกวิธีชำระเงิน (Payment methods)
หมายเลขที่ 10 Modal แบบฟอร์มเก็บข้อมูลที่จำเป็นกับวิธีที่เลือกชำระ
หมายเลขที่ 11 หน้ายืนยันการทำรายการธุรกรรม

หมายเลขที่ 12-13 หน้า “ตรวจสอบข้อมูล” หรือ “Transaction history”
หมายเลขที่ 12 ส่วนแสดงข้อมูลธุรกรรมที่เคยเกิดขึ้นสามารถ Download เป็น PDF ได้
หมายเลขที่ 13 ส่วนแสดงหน้าตาของเอกสารที่จะถูก Download เป็น PDF

- เมื่อ Designer เข้าใจ Workflow ที่ต้องการแล้ว ก็จะ Design งาน และส่งมอบให้ dev ผ่านโปรแกรม Zeplin ดังรูปที่ 3.12



รูปที่ 3.12 งาน Design ที่ออกแบบจาก Wireframe

หมายเลขที่ 1-2 หน้า “Package”

หมายเลขที่ 1 ส่วนแสดงข้อมูล Package ที่สามารถซื้อได้ และ Animation เมื่อมีการ Hover (การเอา Cursor ชี้ที่วัตถุ)

หมายเลขที่ 2 ส่วนแสดงข้อมูล Package ที่สามารถซื้อได้ และ Animation เมื่อมีการไม่ Hover

หมายเลขที่ 3 หน้า “แพ็กเกจของฉัน” หรือ “My Package” ปรับ Design ให้แสดงข้อมูล Package ปัจจุบัน, Products ที่ซื้อโควตาเพิ่ม และหน้า Payment history ถูกรวมไว้ที่หน้านี้หน้าเดียว

หมายเลขที่ 4 หน้า “Market” แสดงข้อมูล Products ทั้งหมดที่สามารถซื้อโควตาเพิ่มได้จากระบบ

หมายเลขที่ 5-11 หน้า “Checkout”

หมายเลขที่ 5 ขั้นตอนที่ 1 Cart detail (หลังจากกดปุ่ม “Buy Now” ในหน้า Market) แสดงข้อมูลรายละเอียดตะกร้าสินค้าว่ามีสินค้าอะไรบ้าง

หมายเลขที่ 6 ขั้นตอนที่ 2 Billing address ฟอร์มเก็บข้อมูลเพื่อออกในสั่งซื้อสินค้า

หมายเลขที่ 7 ขั้นตอนที่ 3 Credit card เลือกชำระด้วยบัตรเครดิต/เดบิทที่เคยลงทะเบียนไว้หรือใช้ข้อมูลใหม่

หมายเลขที่ 8 หากเลือกใช้ข้อมูลใหม่กรอกข้อมูลในฟอร์ม

หมายเลขที่ 9 เมื่อกรอกข้อมูลเสร็จสิ้นและบัตรได้รับการ Verify แสดงรูปบัตรตัวแทน

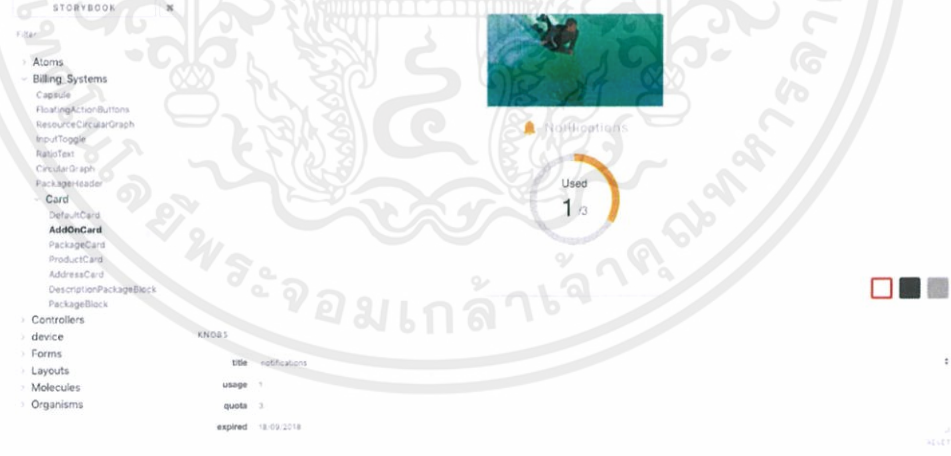
หมายเลขที่ 10 Animation เมื่อกดที่บัตรตัวแทน

หมายเลขที่ 11 เมื่อกดยืนยันธุรกรรม แสดง Modal เพื่อให้รอกระบวนการ

หมายเลขที่ 12 หน้า “แพ็กเกจของฉัน” หรือ “My Package” เมื่อการทำธุรกรรมเสร็จสิ้นกระบวนการ แสดง Notification เพื่อบอกผลลัพธ์การทำธุรกรรม

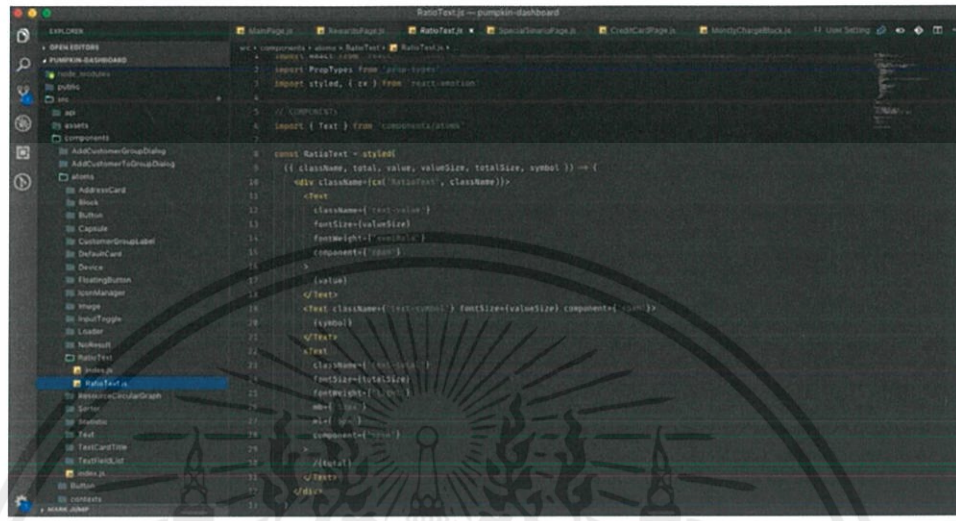
3.3.3 การพัฒนาระบบหน้าบ้าน (Front-end)

เมื่อได้ต้นแบบจากงาน Design แล้ว ขั้นตอนถัดไปก็คือการพัฒนา Front-end ให้ตรงตามแบบที่ได้ ในส่วนนี้ใช้เทคโนโลยี React ซึ่งเป็น JavaScript Library และเพื่อความรวดเร็วในการพัฒนาระบบทางทีมผู้พัฒนาใช้ Storybook ที่ช่วยให้ Dev สามารถสร้าง Component ได้รวดเร็วขึ้น ดังรูปที่ 3.13



รูปที่ 3.13 Component ที่ถูกเขียนด้วย React แสดงผลใน Storybook

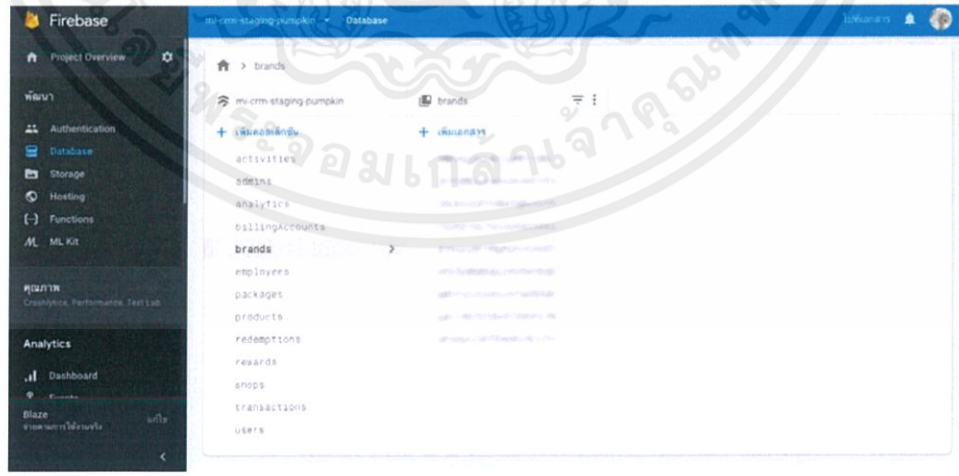
ในโปรเจคนี้ใช้การวาง Project structure โดยอิงแนวคิด Atomic มาปรับใช้ ทำให้สามารถจำแนก Component ได้อย่างมีประสิทธิภาพมากขึ้น ดังรูปที่ 3.14



รูปที่ 3.14 การพัฒนา Front-end โดยปรับใช้ Atomic structure ผ่านเครื่องมือ VSCode

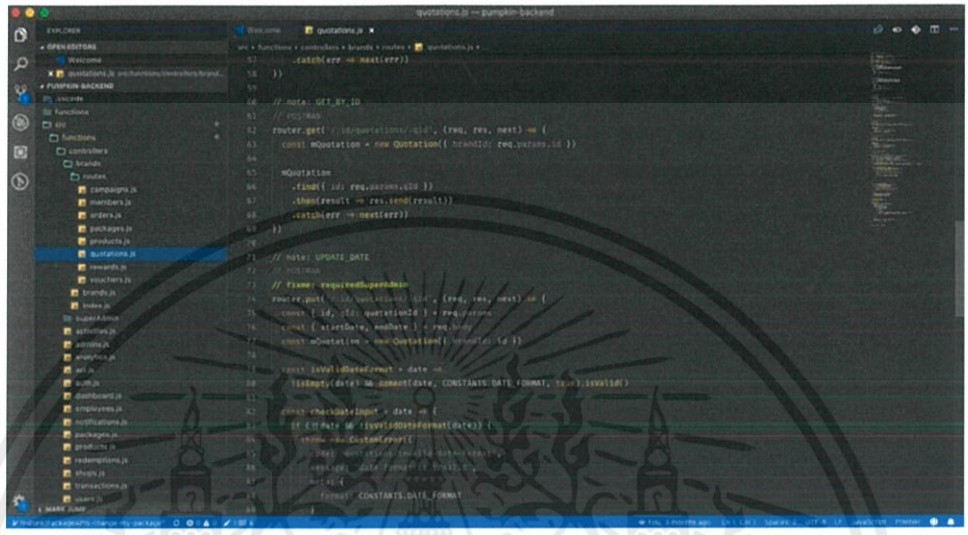
3.3.3 การพัฒนาระบบหลังบ้าน (Back-end)

ในฝั่ง Back-end ทีมผู้พัฒนาใช้ Cloud Firestore ซึ่งเป็นบริการของ Database ของ Google ซึ่งช่วยจัดการเรื่องจิปาถะของ Back-end ต่าง ๆ ทีมผู้พัฒนาเพียงต้องจัดการเรื่อง API เท่านั้น ทำให้การสร้าง Project เป็นไปได้อย่างรวดเร็ว ดังรูปที่ 3.15 คือหน้าแดชบอร์ดของ Cloud Firestore สามารถตรวจสอบและแก้ไขข้อมูลที่เก็บใน Database และตั้งค่าระบบส่วนต่าง ๆ ของ Cloud Firestore ได้



รูปที่ 3.15 Cloud Firestore Dashboard

การพัฒนาของฝั่งนี้จึงเป็นการพัฒนา API เป็นหลัก เพราะเป็นส่วนที่ทำหน้าที่ติดต่อระหว่าง Front-end กับ Back-end โดยใช้เทคโนโลยี Express ซึ่งเป็น Web application framework บน Node.js ในการสร้าง RESTful API สามารถดูตัวอย่างการพัฒนา API จากรูปที่ 3.16



รูปที่ 3.16 การพัฒนา API บนเครื่องมือ VSCode

3.4 ทดสอบระบบ

ในการทดสอบระบบจะถูกทดสอบในเซิร์ฟเวอร์ (Server) ทดสอบ (Pumpkin-Staging server และ OMISE-Dev mode) ทำให้สามารถจำลองสถานการณ์ต่าง ๆ ได้โดยไม่เกิดผลกระทบกับระบบหลัก (Production server)

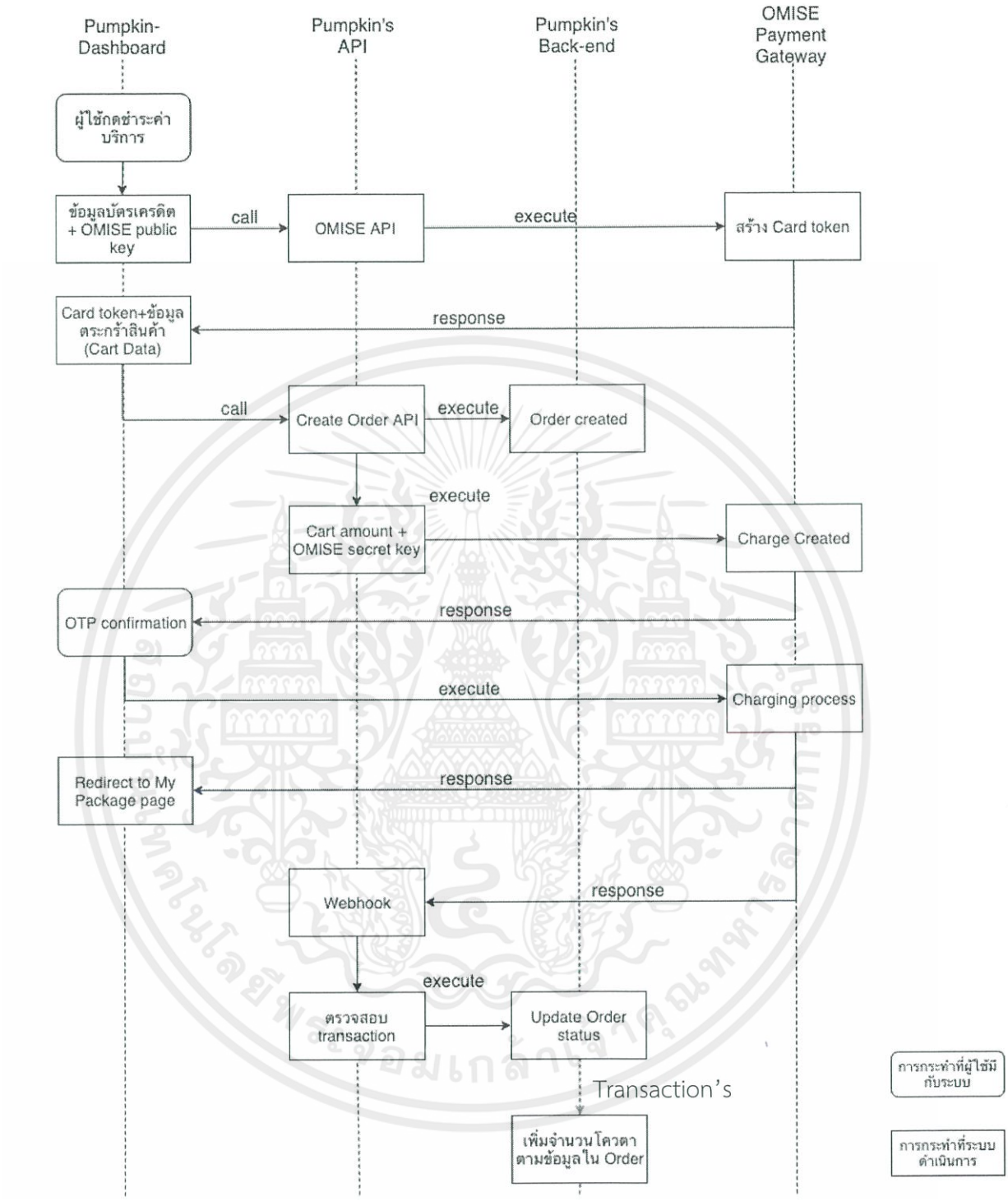
3.4.1 การทดสอบระบบชำระค่าแบบชำระทันที (Instant Billing System)

ในระบบชำระค่าบริการแบบชำระทันที (Instant Billing System) ถูกใช้ในการชำระค่าบริการของ Top up products อธิบาย API Workflow ได้ว่า

หลังจากผู้ใช้กรอกข้อมูลตามฟอร์มครบถ้วนและยืนยันการทำธุรกรรม (Transaction confirmation) ระบบสร้าง Card token ซึ่งเป็นข้อมูลที่ถูกสร้างจากการนำข้อมูลบัตรเครดิต/เดบิทของผู้ใช้ส่งไปให้เซิร์ฟเวอร์ของ OMISE (เป็น Hash ค่าหนึ่งมีอายุการใช้งาน 1 ครั้ง ระยะเวลาจำกัด) ซึ่ง OMISE จะทำหน้าที่เก็บข้อมูลส่วนนี้ไว้ให้เพื่อความปลอดภัย (การทำ Transaction กับ OMISE จะไม่ใช้ข้อมูลบัตรโดยตรง แต่จะใช้ Token ถูกสร้างจาก OMISE เป็นตัวแทนข้อมูลบัตร) ผ่าน OMISE API (ฝัง OMISE's public key ที่ Front-end ก่อนเรียกใช้งาน keyนี้มีหน้าที่เพียงเพื่อสร้าง Token เพียงอย่างเดียวเท่านั้น) เมื่อได้ Card token จาก OMISE แล้ว ระบบจะเรียกใช้ API สำหรับสร้างข้อมูลใบสั่งซื้อ

สินค้า (Order) ซึ่ง API ตัวนี้ จะนำข้อมูลที่ได้จาก Request body ไปสร้าง Order ใน Pumpkin's Database หลังจากนั้นนำจำนวนเงินที่ต้องชำระ (Order amount) มาสร้างข้อมูลการชำระค่าบริการ (Charge data) กับ OMISE เพื่อหักค่าบริการจากบัตร (ฝัง OMISE's secret key ที่ Back-end ก่อนเรียกใช้งาน ควรเข้ารหัส key ประเภทนี้ก่อนเก็บบนเซิร์ฟเวอร์) การทำธุรกรรมนี้จำเป็นต้องยืนยัน OTP (OTP confirmation) ซึ่งเป็นหน้าที่ของ OMISE จัดการ เมื่อการยืนยัน OTP เสร็จสิ้น จะทำการ Redirect ไปยังหน้าที่ตั้งค่าไว้กับ OMISE API (ในที่นี้ กำหนดให้ใช้หน้า My Package เป็นหน้าที่บอกสถานะโควตาคงเหลือของ feature ทั้งหมดที่ร้านค้าสามารถงานใช้ได้และสามารถตรวจสอบข้อมูลการทำธุรกรรมย้อนหลังในอดีตได้)

อย่างไรก็ตาม ทุกความเคลื่อนไหวที่เกี่ยวข้องกับบัญชี OMISE ข้อมูลแจ้งเตือน (Events) จะถูกส่งจาก OMISE server มายัง URL ที่ได้ตั้งไว้ (Webhook) ซึ่ง Webhook นี้จะทำหน้าที่อ่านข้อมูลแจ้งเตือนที่ได้จากการสร้างข้อมูลการชำระค่าบริการ (Charge event) และคอยอัพเดทสถานะข้อมูลใบสั่งซื้อสินค้า (Order status) ไม่ว่า Transaction จะสำเร็จหรือไม่ก็ตาม แต่ถ้าสำเร็จระบบจะอัพเดทจำนวนโควตาของ feature ที่อยู่ในใบสั่งซื้อสินค้าที่เกี่ยวข้องกับ Charge event นั้น ๆ สามารถดูรูปที่ 3.17 ประกอบ



รูปที่ 3.17 Instant billing system's workflow

ซึ่งผลทดสอบความเป็นไปตามกระบวนการทำงานของรูปที่ 3.17

3.4.2 การทดสอบระบบชำระค่าบริการแบบรายเดือน (Monthly Billing System)

ในระบบชำระค่าบริการแบบรายเดือน (Monthly Billing System) ถูกใช้ในการชำระค่าบริการของ Recurring products และ Package เนื่องจากมีข้อจำกัดหลายประการเกิดขึ้นทำให้ไม่สามารถทำขั้นตอนนี้ได้แล้วเสร็จ

3.4.2.1 ปัญหาที่เกิดขึ้น

3.4.2.1.1 ข้อจำกัดด้านเทคนิค

ในการพัฒนาระบบชำระค่าบริการรายเดือนนี้ ติดปัญหาเรื่องการไม่มี OMISE API ที่สามารถนำมาใช้งานกับรูปแบบการเก็บค่าบริการที่เปลี่ยนไปทุกเดือนได้ (การคิดค่าบริการแบบยุติธรรมต่อผู้ใช้ (Fair to users) ทำให้ค่าบริการที่ต้องเก็บเปลี่ยนไปตามพฤติกรรมของผู้ใช้ในแต่ละเดือน)

ปกติแล้วในการเก็บค่าบริการรายเดือนนั้นมี OMISE API ทำหน้าที่ในการจัดการเรื่องนี้อยู่ เรียกว่า Charge Schedule API ซึ่งมีความสามารถในการช่วยสร้าง, เรียกข้อมูล, หรือลบรายการการรับชำระเงิน (Charge) ที่ได้มีการตั้งล่วงหน้าไว้ เช่น สร้างรายการรับชำระเงินทุก ๆ วันที่ 15 ของเดือน เป็นระยะเวลา 12 เดือน เป็นต้น แต่จำนวนเงินที่กำหนดให้ API นี้ จะไม่สามารถเปลี่ยนแปลงจำนวนได้ ทำให้ไม่สามารถนำมาใช้งานร่วมกับระบบคิดเงินของ Pumpkin-Dashboard ได้

3.4.2.1.1 ข้อจำกัดด้านเวลา

เนื่องจากหน้าที่ในการสร้างระบบทั้งหมด ผู้จัดทำรับผิดชอบด้านเทคนิคแต่เพียงผู้เดียว ทำให้มีข้อจำกัดหลาย ๆ อย่างเกิดขึ้นและปัญหาด้านเวลานับเป็นข้อจำกัดใหญ่ที่ไม่สามารถแก้ไขได้ ทำให้จำเป็นต้องปรับลดสโคปงานลง เพื่อให้พนักงานในทีมรับช่วงไปพัฒนาระบบต่อไป

3.4.2.2 การแก้ปัญหา

3.4.2.2.1 การแก้ปัญหาด้านเทคนิค

เนื่องจากข้อจำกัด Charge Schedule API ของ OMISE จึงมีการปรึกษากับทีมและเสนอให้ใช้ API ที่มีความสามารถทดแทนกันได้อย่างการเปลี่ยนมาใช้ Customer API ที่สามารถเก็บ Token บัตรเครดิตให้สามารถใช้งานได้เรื่อย ๆ (ปกติ Token จะมีอายุการใช้งาน 1 ครั้งในช่วงระยะเวลาหนึ่ง) และต้องใช้ API พิเศษที่สามารถสร้างรายการการรับชำระเงินอัตโนมัติโดยไม่ต้องยืนยัน OTP (การสร้าง Charge โดยปกติแล้วต้องมีการยืนยัน OTP จากผู้ใช้ทุกครั้งที่เกิดการทำธุรกรรมขึ้น) แต่การจะข้ามขั้นตอนยืนยัน OTP นั้น จำเป็นต้องประสานงานกับทีมงานของ

OMISE โดยตรงเพื่อขอ API พิเศษนี้ ซึ่งขั้นตอนการประสานงาน พนักงานในทีม Pumpkin จะเป็นผู้รับช่วงต่อแทน ส่วนการตั้งเวลาให้เก็บค่าบริการตามวันที่กำหนดนั้นจะใช้ Crontab (Service ที่ตั้งค่าเวลาให้ทำงานอัตโนมัติ) ยิงมาสร้างรายการการรับชำระเงินจากระบบของ Pumpkin แทนการใช้งาน Schedule API

3.4.2.2.2 การแก้ปัญหาด้านเวลา

ด้วยการต้องลดสโคปลงจึงเปลี่ยนจากพัฒนาทั้งระบบ กลายเป็นเพียงสร้าง API ที่ใช้คำนวณค่าบริการรายเดือนให้ถูกต้องก็พอแล้ว ทำให้สโคปของระบบชำระค่าบริการแบบรายเดือนจบลงที่การคิดค่าบริการได้ถูกต้องโดยไม่ต้องเชื่อมต่อกับ OMISE payment gateway

บทที่ 4

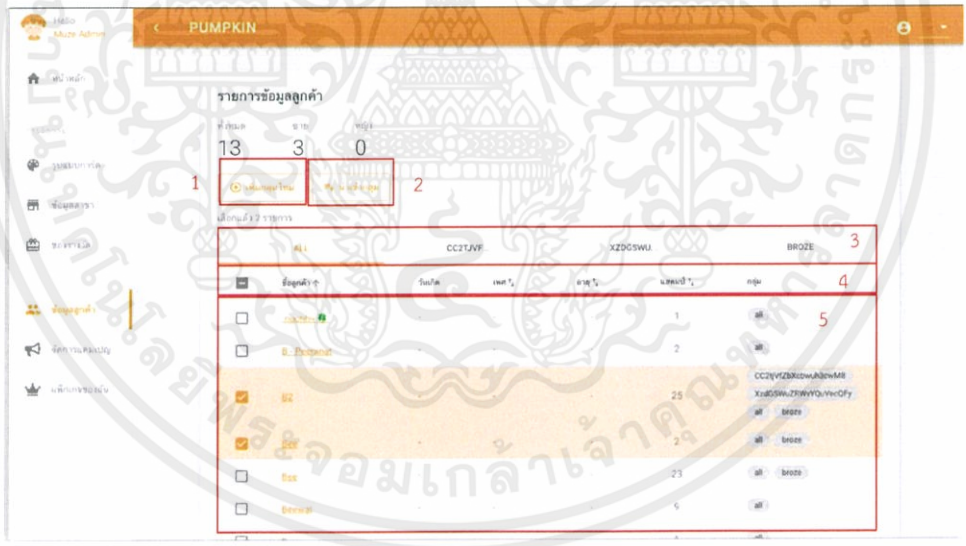
ผลการทำงาน

ผลลัพธ์ที่นำมาแสดงในรายงานเล่มนี้เป็นผลลัพธ์ของระบบที่ทำงานบนเซิร์ฟเวอร์ทดสอบ (Staging server ของ Pumpkin-CRM และ OMISE-DEV mode) ทำให้สภาพแวดล้อมในการทดสอบระบบมีจำกัด ยังคงไม่อาจนำไปใช้งานจริงได้อย่างเต็มประสิทธิภาพ เพียงผ่านเกณฑ์ที่ทีมกำหนดไว้เท่านั้น

แบ่งแสดงผลการทำงานออกเป็น 2 ส่วน คือระบบชำระค่าบริการแบบชำระทันทีและระบบชำระค่าบริการแบบรายเดือน

4.1 ระบบชำระค่าบริการแบบชำระทันที (Instant billing system)

- ในระบบนี้จำลองสถานการณ์ทดสอบดังนี้ กำหนดให้ร้านค้าใช้งาน Free package ซึ่งมีโควตาการสร้าง Customer group อยู่ที่ 3 กลุ่ม และปัจจุบันแอดมินใช้งานโควตาเต็มจำนวนแล้ว ดังรูปที่ 4.1



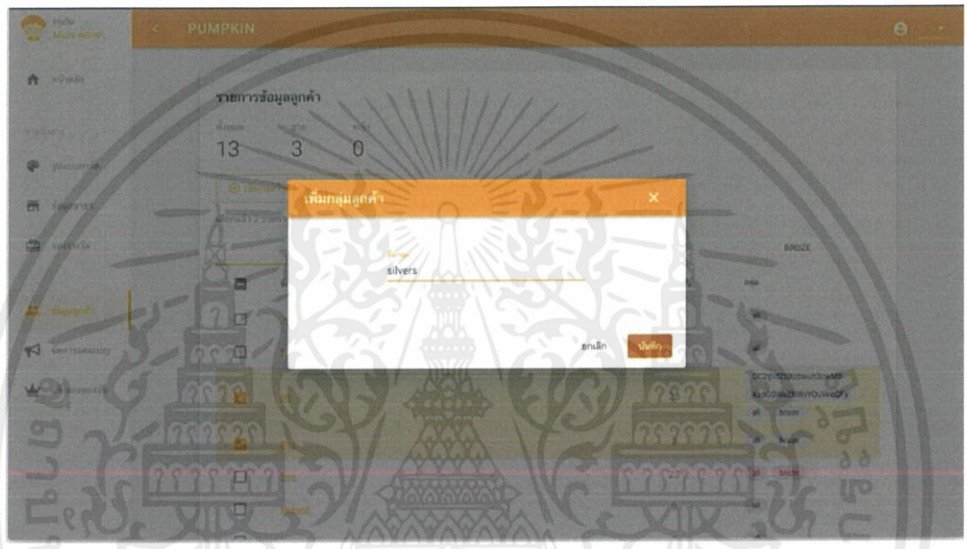
รูปที่ 4.1 หน้าสร้าง Customer group ที่มีการสร้างกลุ่มไปแล้ว 3 กลุ่ม

หมายเลข 1 ปุ่มเพิ่มกลุ่มลูกค้าใหม่
หมายเลข 2 ปุ่มเพิ่มลูกค้าเข้ากลุ่มที่มีอยู่แล้ว
หมายเลข 3 แท็บแสดงรายชื่อกลุ่มลูกค้าที่สร้างไว้

หมายเลข 4 หัวข้อรายละเอียดข้อมูลลูกค้า

หมายเลข 5 รายชื่อลูกค้าในระบบของร้านค้า

- เมื่อแอดมินต้องการสร้างกลุ่มเพิ่มดังรูปที่ 4.2 ระบบจะตรวจสอบโควตาคงเหลือก่อนสร้างดังรูปที่ 4.3 และสุดท้ายไม่สามารถสร้างกลุ่มได้เนื่องจากครบจำนวนโควตาที่มีอยู่แล้วพร้อมแจ้งให้ซื้อจำนวนโควตาเพิ่มดังรูปที่ 4.4

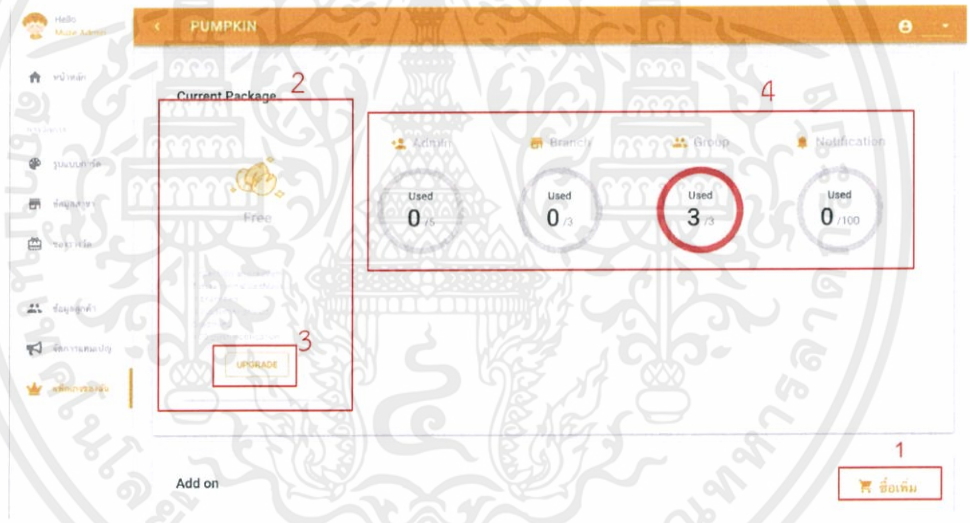


รูปที่ 4.2 สร้างกลุ่มลูกค้าใหม่ชื่อกลุ่มว่า “silvers”

รูปที่ 4.3 รอกระบวนการสร้างกลุ่มใหม่

รูปที่ 4.4 แจ้งเตือนว่าไม่สามารถสร้างกลุ่มใหม่ได้ ต้องซื้อโควตาเพิ่ม

- แอดมินสามารถตรวจสอบโควตาคงเหลือได้ที่หน้า “แพ็กเกจของฉัน” หรือ “My Package” ดังรูปที่ 4.5 และสามารถซื้อโควตาได้จากหน้านี้



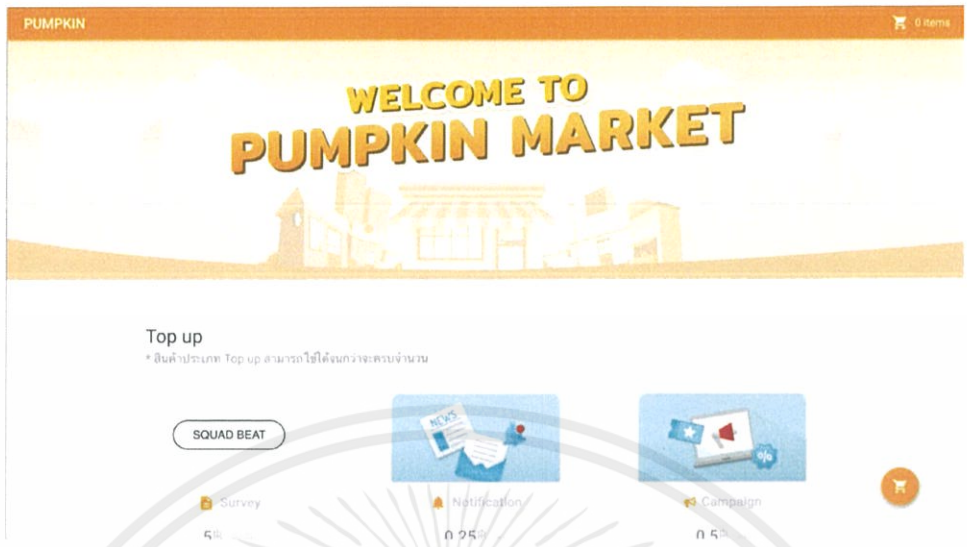
รูปที่ 4.5 หน้า “แพ็กเกจของฉัน”

หมายเลข 1 ปุ่ม “ซื้อเพิ่ม” จะนำไปยังหน้า “Market” ดังรูปที่ 4.6

หมายเลข 2 Package cad บอกสถานะว่าใช้งานแพ็กเกจไหนอยู่

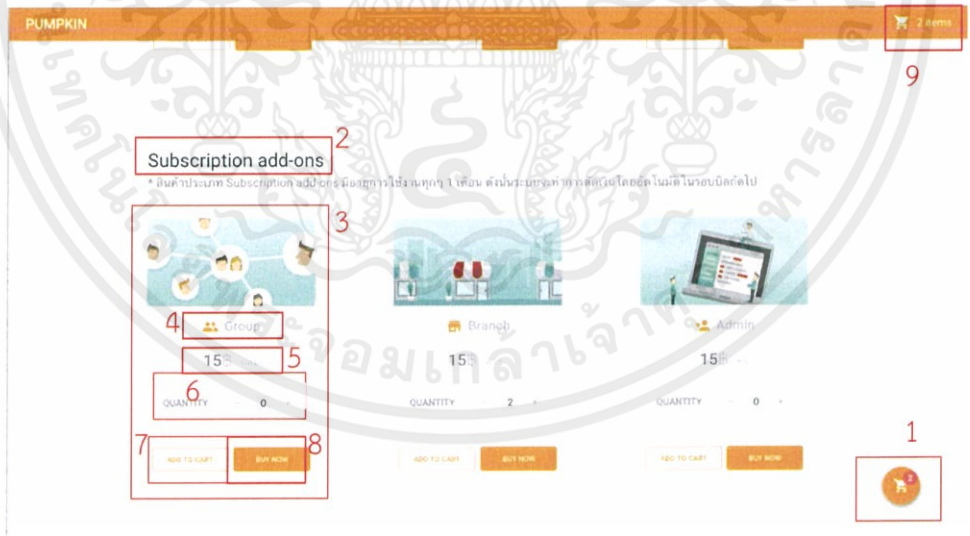
หมายเลข 3 ปุ่ม “อัพเกรด” จะนำไปยังหน้า “เปลี่ยนแพ็กเกจ”

หมายเลข 4 วงแหวนบอกสถานะโควตาของ feature ที่สามารถใช้งานได้



รูปที่ 4.6 หน้า “Market” ที่รวบรวม feature ที่สามารถซื้อโควตาเพิ่มได้

- ที่หน้า “Market” สามาถเลือกซื้อโควตาเฉพาะ feature ที่ต้องการได้ โดยมี Products card ที่บอกรายละเอียดการซื้อเฉพาะ feature นั้น ๆ สามารถกดปุ่ม “Add to cart” จะเพิ่มสินค้าลงตะกร้าพร้อมเลือกซื้อสินค้าชิ้นต่อ ๆ ไป ค่อยทำการชำระสินค้าในภายหลัง ดังรูปที่ 4.7 หรือสามารถกดปุ่ม “Buy now” เพื่อเพิ่มสินค้าลงตะกร้าพร้อมเข้าสู่ที่กระบวนการชำระสินค้าทันที



รูปที่ 4.7 เมื่อเลือกกด Add to cart ที่ Product card จะเพิ่มจำนวนสินค้าที่อยู่ในตะกร้า

หมายเลข 1,9 ปุ่ม checkout จะนำไปสู่หน้า “ชำระตะกร้าสินค้า”

หมายเลข 2 บอกประเภทของ Products ว่าเป็นแบบ Top up หรือ Subscription (Subscription ชื่อทางการค้า Recurring ชื่อในระบบ)

หมายเลข 3 Product card บอกรายละเอียด feature ที่สามารถซื้อโควาตาได้

หมายเลข 4 ชื่อ feature ของ Product card ใบนี้

หมายเลข 5 ราคาต่อหน่วยต่อเดือน

หมายเลข 6 จำนวนโควตาที่ต้องการซื้อเพิ่ม

หมายเลข 7 ปุ่ม “Add to cart” เพิ่มสินค้าที่เลือกลงในตะกร้าสินค้า

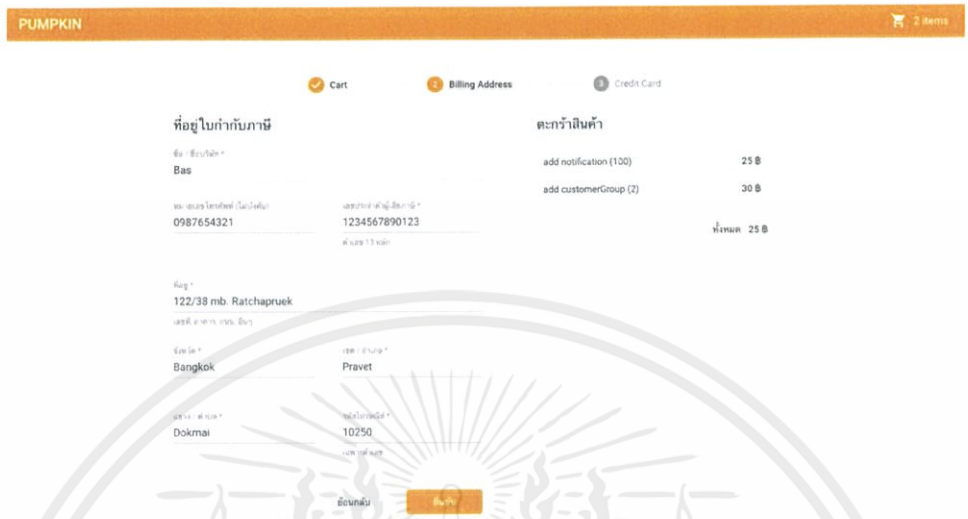
หมายเลข 8 ปุ่ม “Buy now” เพิ่มสินค้าที่เลือกลงตะกร้าสินค้า แล้วไปยังหน้า “ชำระตะกร้าสินค้า” ทันที

- หน้า “ชำระตะกร้าสินค้า” หรือ “Checkout” ในการทดสอบนี้ เลือกซื้อโควตาเพิ่ม 2 feature นั่นคือ เพิ่มโควตา Customer group 2 กลุ่ม และ Notification 100 คน ดังรูปที่ 4.8

รูปที่ 4.8 หน้า Checkout แสดงรายการสินค้าที่เลือกไว้ก่อนชำระค่าสินค้า

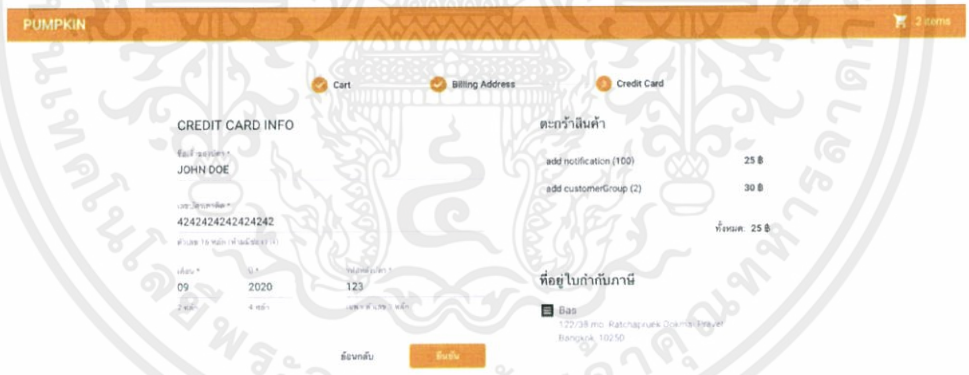
เนื่องจากการมีสินค้าประเภท Recurring products อย่าง Customer group feature อยู่ในตะกร้าสินค้า ระบบจะไม่เก็บค่าบริการทันที การเก็บค่าบริการสินค้าประเภทนี้จะคำนวณค่าบริการเช่นเดียวกับระบบ Package โดยใช้วิธีสร้าง Quotation เก็บไว้ ส่วนสินค้าที่เป็นประเภท Top up ใช้การคิดค่าบริการแบบชำระทันที

- ขั้นตอนถัดมาคือกรอกแบบฟอร์มข้อมูลที่อยู่ใบกำกับภาษี (Billing Address) เพื่อออกใบสั่งซื้อสินค้า (Order) ดังรูปที่ 4.9



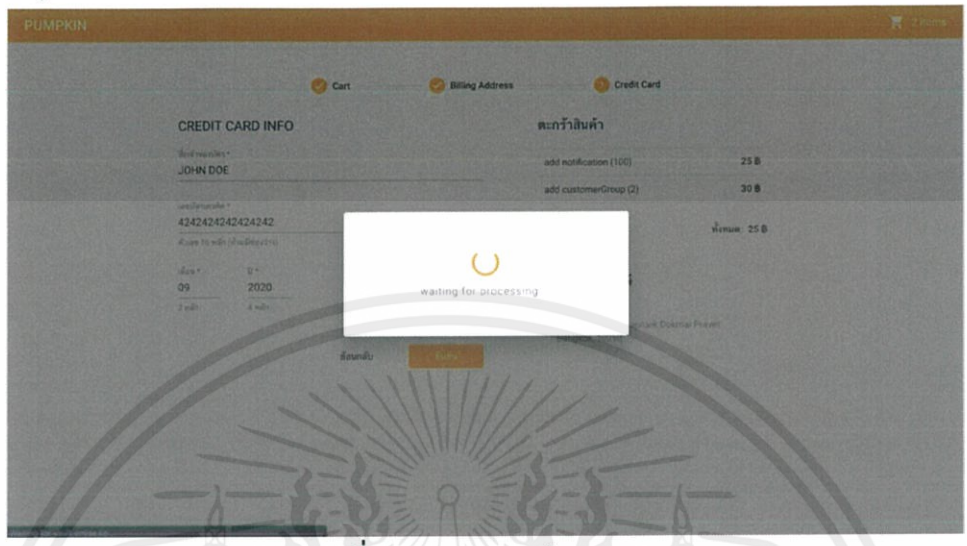
รูปที่ 4.9 แบบฟอร์มสำหรับเก็บข้อมูลใบกำกับภาษี

- ขั้นตอนสุดท้ายคือการข้อมูลบัตรเครดิต/เดบิต ที่ใช้ในการชำระค่าบริการ ดังรูปที่ 4.10 ซึ่งข้อมูลบัตรจะไม่ถูกเก็บในระบบของ Pumpkin โดยตรงเพื่อความปลอดภัยของผู้ใช้งาน



รูปที่ 4.10 แบบฟอร์มสำหรับบัตรเครดิต/เดบิต

- รอระบบชำระค่าบริการ ดังรูปที่ 4.11 ซึ่งกระบวนการทำงานเป็นไปตาม Workflow ในรูปที่ 3.17 ของบทที่ 3

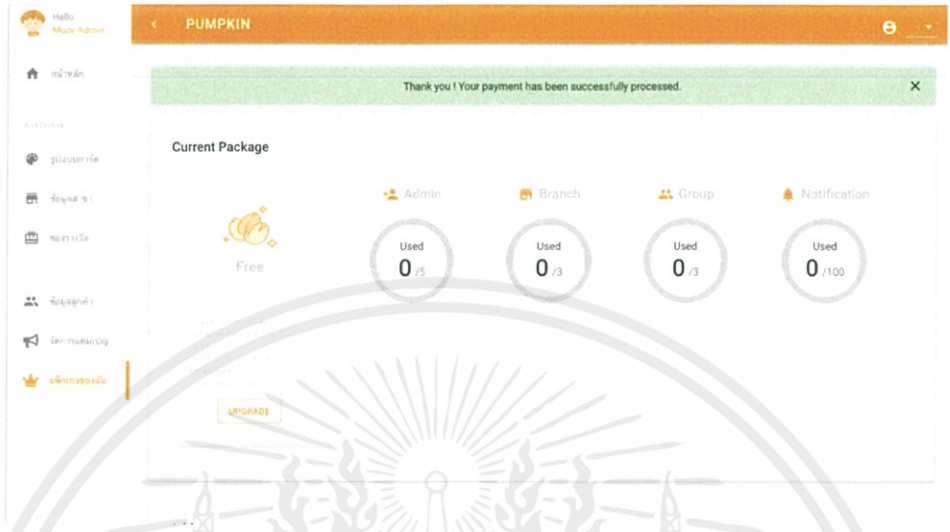


รูปที่ 4.11 รอระบบทำรายการ

- มีการเชื่อมต่อไปยังหน้ากรอก OTP ดังรูปที่ 4.12 เพื่อยืนยัน transaction กับ OMISE (ในส่วนนี้ระบบของ Pumpkin ไม่มีส่วนเกี่ยวข้อง) ซึ่งในเซิร์ฟเวอร์ทดสอบ (OMISE-Dev mode) จะข้ามกระบวนการนี้ไป โดยถือว่าขั้นตอนนี้สำเร็จอย่างแน่นอน

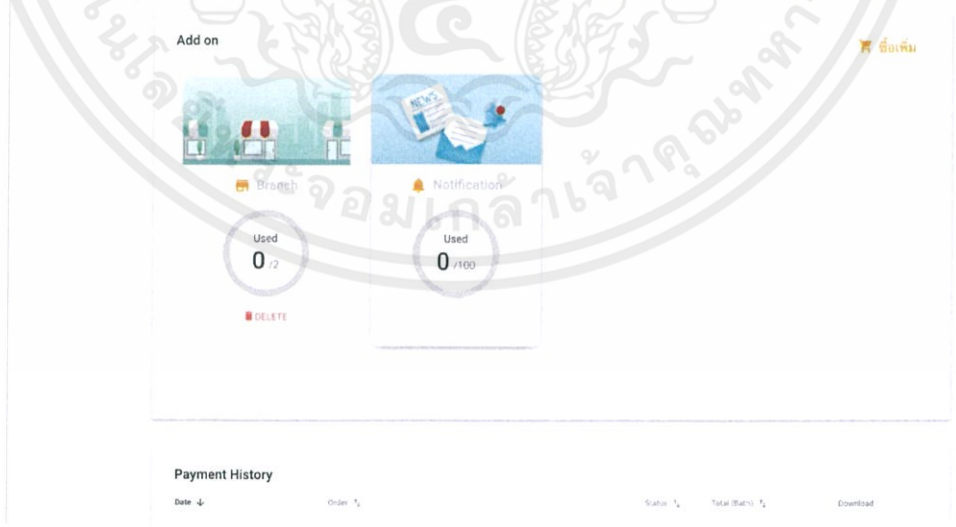
รูปที่ 4.12 หน้ากรอก OTP ของ OMISE ที่ถูกข้ามขั้นตอนกรอกข้อมูลใน OMISE-Dev mode

- สุดท้ายแล้วหลังขั้นตอนกรอก OTP เสร็จสิ้น OMISE จะ Redirect มายังหน้าที่กำหนดไว้ ในที่นี้คือหน้า “แพ็กเกจของฉัน” ซึ่งจะมีแจ้งเตือนบอกผลการทำรายการว่าสำเร็จหรือไม่ ดังรูปที่ 4.13



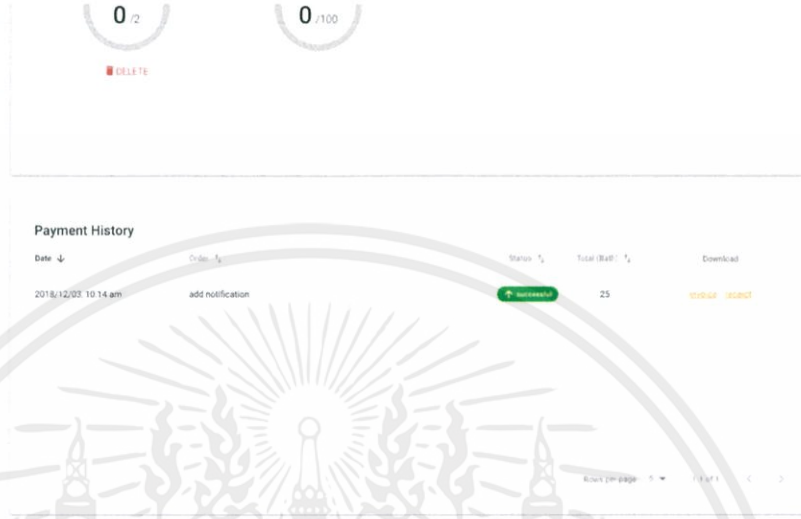
รูปที่ 4.13 หน้า “แพ็กเกจของฉัน” ที่ถูก Redirect มาจาก OMISE

- เมื่อเลื่อนลงในหน้า “แพ็กเกจของฉัน” จะเห็นว่ามีส่วนที่แสดงผลการซื้อโควตาของ feature เฉพาะอยู่ ดังรูปที่ 4.14 และเมื่อ transaction สำเร็จ (สามารถเก็บเงินจากบัตรที่กรอกได้) ผู้ใช้จะมีโควตาของ feature ที่ซื้อแสดงขึ้นมา และสามารถใช้งานนอกเหนือจากโควตาที่ได้รับจาก Package



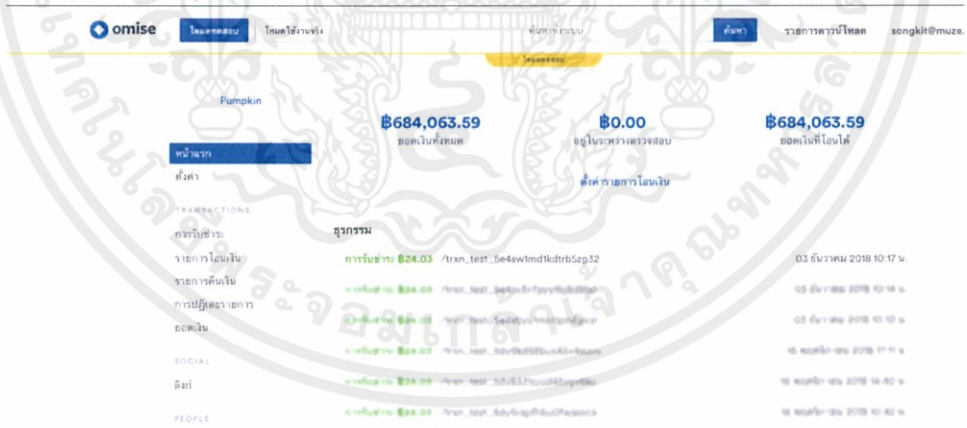
รูปที่ 4.14 หน้า “แพ็กเกจของฉัน” ส่วนแสดง Products ที่ซื้อจากระบบ

- ส่วนสุดท้ายในหน้า “แพ็กเกจของฉัน” แสดงข้อมูลรายการใบสั่งซื้อ (Order) ย้อนหลัง สามารถพิมพ์หรือดาวน์โหลดเป็น PDF ได้ ดังรูปที่ 4.15



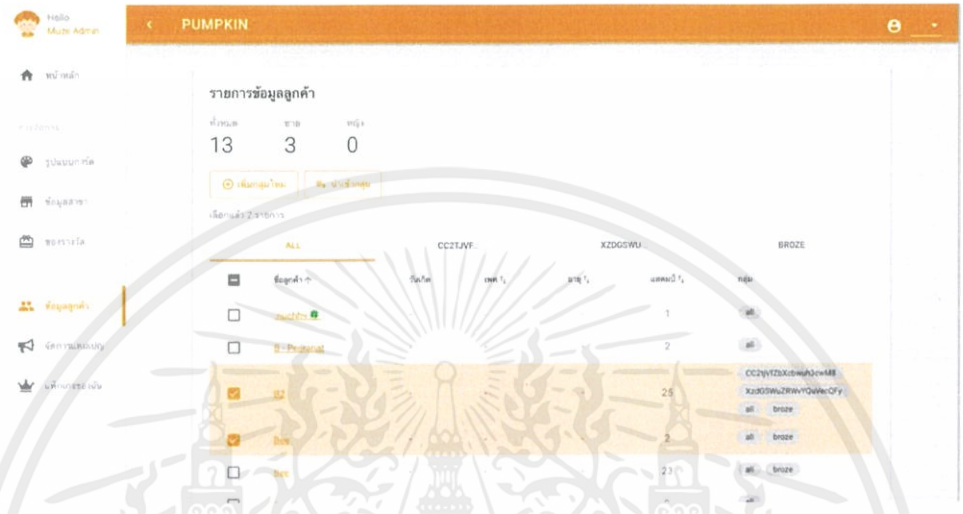
รูปที่ 4.15 หน้า “แพ็กเกจของฉัน” ส่วนแสดง Order list

- ที่ OMISE Dashboard สามารถตรวจสอบ transaction ที่เคยเกิดขึ้นได้ จะเห็นว่ามีค่าบริการที่เกิดขึ้นใหม่ใกล้เคียงกับค่าบริการก่อนหน้านี้ (ถูก OMISE หักค่าดำเนินการ) ดังรูปที่ 4.16

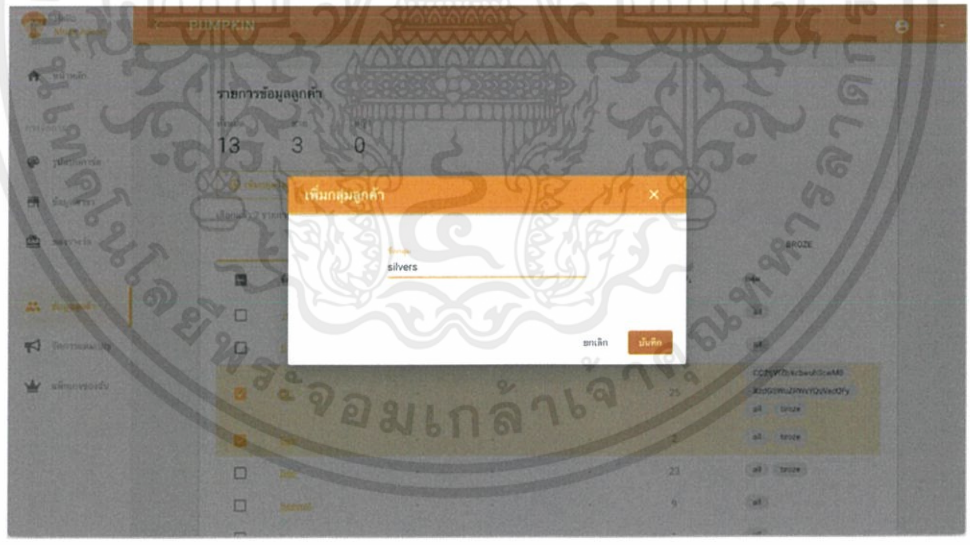


รูปที่ 4.16 หน้า “OMISE Dashboard” แสดงข้อมูลธุรกรรมที่เกิดขึ้น

- เมื่อย้อนกลับมาสร้าง Customer group ใหม่อีกครั้ง โดยเริ่มด้วยการเลือกลูกค้าที่จะเพิ่มเข้ามาในกลุ่มดังรูปที่ 4.17 ตั้งชื่อกลุ่มลูกค้าใหม่ดังรูปที่ 4.18 รอกระบวนการตรวจสอบโควตาดังรูปที่ 4.19 เสร็จสิ้น ก็จะสามารถสร้างกลุ่มใหม่ได้สำเร็จดังรูปที่ 4.20

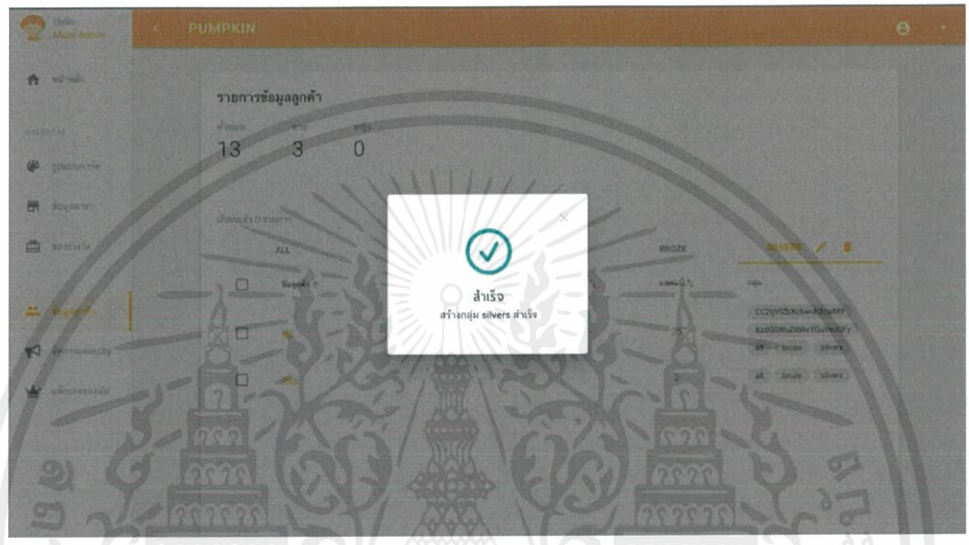


รูปที่ 4.17 หน้า “Customer Group” แสดงข้อมูลลูกค้าทั้งหมดที่มีในระบบของร้านค้า



รูปที่ 4.18 หน้า “Customer Group” สร้างกลุ่มลูกค้าใหม่ชื่อ “silvers”

รูปที่ 4.19 หน้า “Customer Group”กำลังทำรายการ

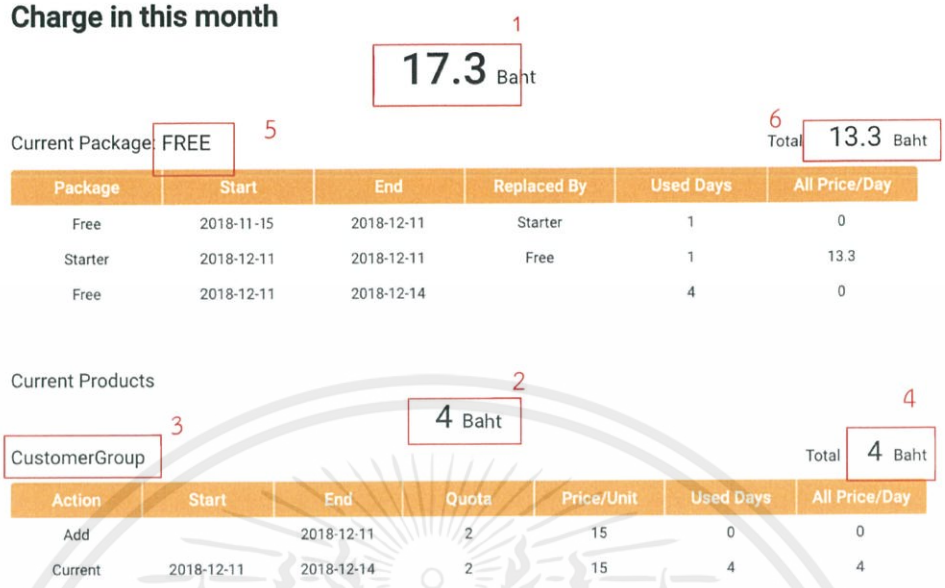


รูปที่ 4.20 หน้า “Customer Group” ทำรายการสำเร็จกลุ่มใหม่ชื่อ “silvers” ถูกเพิ่มขึ้นมา

4.2 ระบบชำระค่าบริการแบบรายเดือน (Monthly billing system)

ในระบบนี้ปรับสโคปงานเป็นเพียงการคำนวณค่าบริการการใช้งาน Package และ Recurring products ดังนั้น จำลองสถานการณ์ดังนี้

- สถานการณ์ที่ 1 กำหนดให้ร้านค้าใด ๆ ในระบบ Pumpkin-CRM ในงาน Pumpkin-Dashboard ในช่วงรอบบิลระหว่างวันที่ 15 พฤศจิกายน – 14 ธันวาคม ร้านค้าใช้งาน Free package ตั้งแต่ต้นรอบบิล เมื่อถึงวันที่ 11 ธันวาคม ร้านค้าเปลี่ยนไปใช้งาน Starter package ราคา 399 บาทต่อเดือน อีกทั้งซื้อโควตาเพิ่มจำนวน Customer group feature 2 กลุ่ม (Recurring products) ในราคา 15 บาทต่อหน่วยต่อเดือน และ Notification feature 100 คน (Top up products) ในวันเดียวกันนั้นเอง ร้านค้าเปลี่ยนกลับไปใช้งาน Free package อีกครั้ง คำนวนค่าบริการสุดท้ายที่ต้องชำระได้จากรูปที่ 4.21 และตารางที่ 4.1



รูปที่ 4.21 หน้าทดสอบ API เพื่อใช้ทดสอบการคำนวณค่าบริการระบบรายเดือน

หมายเลขที่ 1 ค่าบริการสุดท้ายที่ต้องชำระ เกิดจากค่าบริการจากการใช้งาน Package และ Recurring products รวมกัน (Package + Recurring products)

หมายเลขที่ 2 ค่าบริการรวมการใช้งาน Recurring products ทั้งหมด

หมายเลขที่ 3 ชื่อ Recurring product ที่ร้านค้าซื้อเพิ่มนอกเหนือจากโควตาที่ได้จาก Package ในที่นี้คือซื้อ Customer group เพิ่มจำนวน 2 กลุ่ม

หมายเลขที่ 4 ค่าบริการสุดท้ายหากไม่มีการเปลี่ยนแปลงจำนวนโควตาของ Recurring product นี้นั่นคือ หากใช้งาน Customer group 2 กลุ่ม ไปจนจบรอบบิลนี้ต้องเสียค่าบริการจำนวน 4 บาท

หมายเลขที่ 5 ชื่อของ Package ปัจจุบันที่ใช้งานอยู่

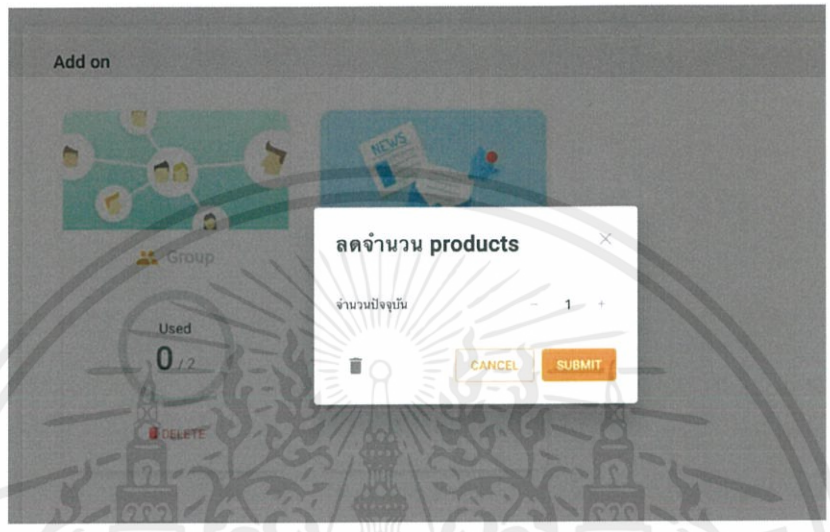
หมายเลขที่ 6 ค่าบริการสุดท้ายที่เกิดจากการในงาน Package ในรอบบิลนี้

ตารางที่ 4.1 ตารางคำนวณค่าบริการที่เกิดขึ้นจากการจำลองสถานการณ์ที่ 1

ประเภท	ชื่อสินค้า	เริ่ม	สิ้นสุด	ราคา/หน่วย/เดือน	ราคา/วัน	โควตา	จำนวนวันที่ใช้	ค่าบริการ
Package	Free	15-11-2018	11-12-2018	0	0	1	27	0
	Starter	11-12-2018	11-12-2018	399	13.3	1	1	13.3
	Free	11-12-2018	14-12-2018	0	0	1	4	0
Products	Customer group	11-12-2018	14-12-2018	15	0.5	2	4	4
รวม								17.3

จะเห็นว่าค่าบริการสุดท้ายที่ได้จากการคำนวณของ API ในรูปที่ 4.21 และการคำนวณในตารางที่ 4.1 มีค่าเท่ากัน

- สถานการณ์ที่ 2 จากเหตุการณ์ก่อนหน้า หากร้านค้าปรับลงโควตาของ Customer group ลง ดังรูปที่ 4.22 ภายในวันที่ 11 ธันวาคม ให้เหลือโควตาเพียง 1 กลุ่ม จากเดิม 2 กลุ่ม คำนวนค่าบริการสุดท้ายที่ต้องชำระใหม่ได้จากรูปที่ 4.23 และตารางที่ 4.2



รูปที่ 4.22 ปรับลดโควตา Customer group ให้เหลือเพียง 1 กลุ่ม

Charge in this month

16.3 Baht

Current Package: FREE

Total 13.3 Baht

Package	Start	End	Replaced By	Used Days	All Price/Day
Free	2018-11-15	2018-12-11	Starter	1	0
Starter	2018-12-11	2018-12-11	Free	1	13.3
Free	2018-12-11	2018-12-14		4	0

Current Products

3 Baht

CustomerGroup

Total 3 Baht

Action	Start	End	Quota	Price/Unit	Used Days	All Price/Day
Add		2018-12-11	2	15	0	0
Delete	2018-12-11	2018-12-11	2	15	1	1
Current	2018-12-11	2018-12-14	1	15	4	2

รูปที่ 4.23 หน้าทดสอบ API คำนวณค่าบริการระบบรายเดือนในสถานการณ์ที่ 2

ตารางที่ 4.2 ตารางคำนวณค่าบริการที่เกิดขึ้นจากการจำลองสถานการณ์ที่ 2

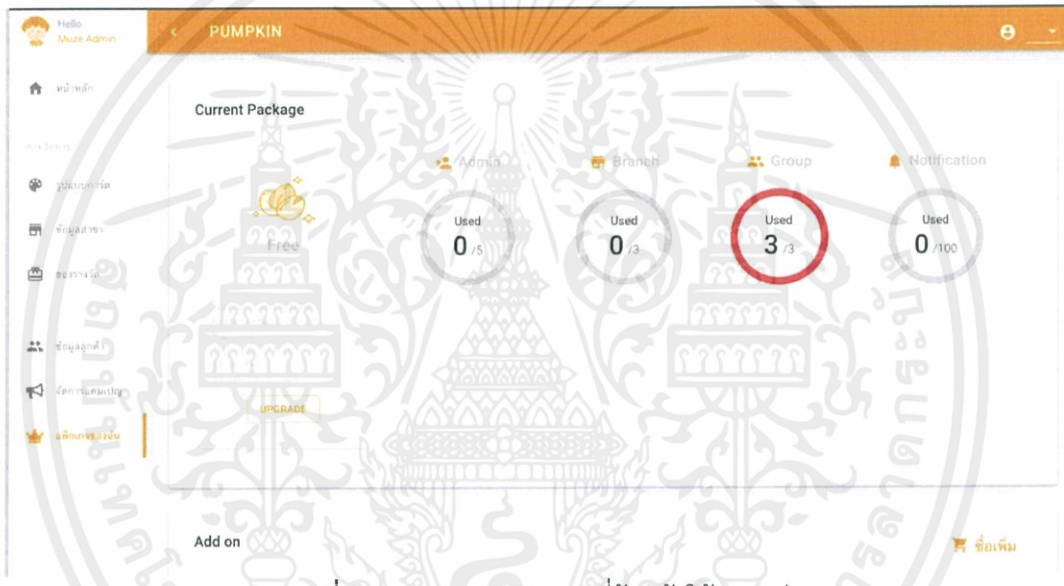
ประเภท	ชื่อสินค้า	เริ่ม	สิ้นสุด	ราคา/หน่วย/เดือน	ราคา/วัน	โควตา	จำนวนวันที่ใช้	ค่าบริการ
Package	Free	15-11-2018	11-12-2018	0	0	1	27	0
	Starter	11-12-2018	11-12-2018	399	13.3	1	1	13.3
	Free	11-12-2018	14-12-2018	0	0	1	4	0
Products	Customer group	11-12-2018	11-12-2018	15	0.5	2	1	1
		11-12-2018	14-12-2018			1	4	2
รวม								16.3

จะเห็นว่าค่าบริการสุดท้ายที่ได้จากการคำนวณของ API ในรูปที่ 4.23 และการคำนวณในตารางที่ 4.2 มีค่าเท่ากัน สรุปว่า API ทำงานได้ตามกำหนดไว้

4.3 ระบบเปลี่ยนแพ็กเกจ (Package)

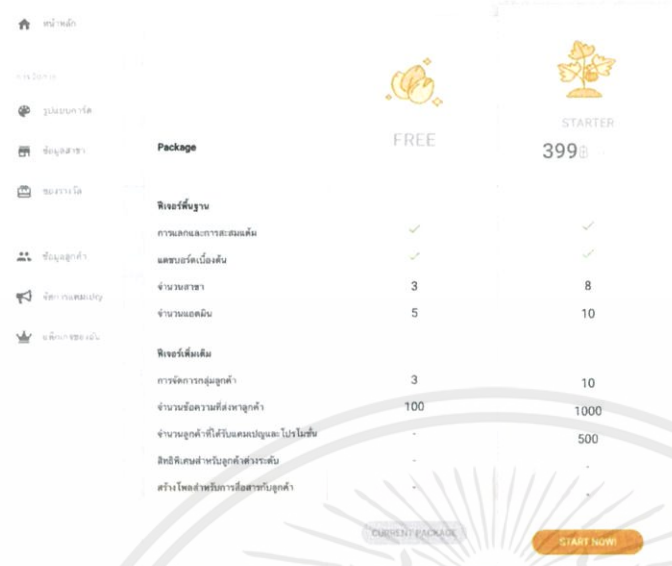
ในการเปลี่ยนระดับแพ็กเกจ เกิดขึ้นเมื่อร้านค้าต้องการโควตาแพคเกจโดยรวมมากกว่าแพ็กเกจปัจจุบัน โควตาเดิมที่เคยใช้ไปก่อนหน้า ก็จะถูกโอนไปในแพ็กเกจใหม่ด้วยเช่นกัน (Upgrade Package) ในทางกลับกัน ในเกิดกรณีที่แพ็กเกจปัจจุบันมีโควตาโดยรวมมากเกินจำเป็น ทำให้ต้องการลดระดับแพ็กเกจลงเพื่อประหยัดค่าใช้จ่าย (Downgrade Package) สามารถอธิบายขั้นตอนการเปลี่ยนแพ็กเกจใหม่ดังนี้

- เริ่มต้นแอดมินต้องมาอยู่ในหน้า “แพ็กเกจของฉัน” โดย block ส่วนบน จะแสดงข้อมูลแพ็กเกจปัจจุบันที่ร้านค้าใช้งานอยู่ดังรูปที่ 4.24



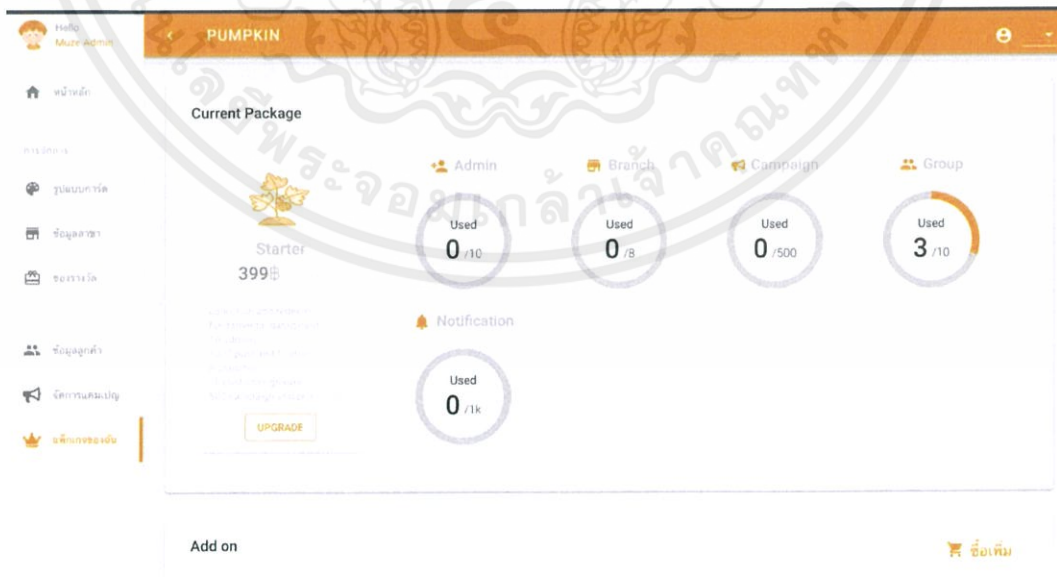
รูปที่ 4.24 Free package ที่ร้านค้าใช้งานอยู่

- เพื่อเปลี่ยนแพ็กเกจ ต้องกดปุ่ม “UPGRADE” ที่อยู่บน Package card จึงจะเข้าสู่หน้า “เปลี่ยนแพ็กเกจ” ซึ่งเป็นหน้าอธิบายรายละเอียดของแพ็กเกจที่มีในระบบและร้านค้าสามารถเปลี่ยนไปใช้งานได้ ดังรูปที่ 4.25



รูปที่ 4.25 เมื่อกดปุ่ม “UPGRADE” ใน Package card จะนำมายังหน้า “เปลี่ยนแพ็กเกจ”

- เมื่อเลือกแพ็กเกจที่ต้องการได้แล้วก็กดที่ปุ่ม “START NOW!” ระบบจะเปลี่ยนระดับของแพ็กเกจปัจจุบันให้ดังรูปที่ 4.26 โดยโควตาใช้งานจากแพ็กเกจเดิมก็ถูกโอนมาด้วยจนกว่าจะครบรอบบิลและชำระค่าบริการรายเดือนเสร็จสิ้นจึงจะรีเซทโควตาใหม่ให้ และทันทีการเปลี่ยนแพ็กเกจเกิดขึ้นระบบจะสร้าง Quotation เพื่อบันทึกค่าบริการในการใช้งานแพ็กเกจเดิมก่อนหน้าและเก็บค่าบริการเมื่อสิ้นรอบบิล



รูปที่ 4.26 แพ็กเกจถูกเปลี่ยนเป็น Starter package

บทที่ 5

สรุปผลการวิจัย และข้อเสนอแนะ

5.1 สรุปผลวิจัย

ระบบชำระค่าบริการ เป็นระบบหนึ่งที่มีความจำเป็นในธุรกิจต่าง ๆ ไม่ว่าจะเป็นบนโลกออนไลน์หรือบนโลกจริง ขั้นตอน กระบวนการ ในการนำไปสู่การชำระค่าบริการเป็นสิ่งสำคัญที่จะทำให้ผู้ใช้งานรู้สึกอยากจะปลดภาระหนี้สิน ยิ่งเป็นระบบที่เข้าใจง่าย สะดวก รวดเร็ว และมีความปลอดภัยก็จะทำให้ผู้คนต้องการใช้งานระบบนั้นเป็นจำนวนมาก ความรู้สึกประทับใจต่อผู้ใช้งานจะนำพาเส้นทางใหม่ ๆ ให้เกิดขึ้นได้ ทั้งโอกาสทางธุรกิจ โอกาสในการพัฒนาตนเอง ผลพลอยได้คือทำให้ธุรกิจนั้น ๆ เจริญเติบโตขึ้นได้อย่างรวดเร็วและมั่นคง เป็นรากฐาน เป็นแรงผลักดัน ให้เกิดการพัฒนาอุปกรณ์หรือเครื่องมือในการเพิ่มความสะดวกสบายสำหรับการดำรงชีวิตประจำวันใหม่ ๆ แก่ทุกผู้คนบนโลกใบนี้

ระบบชำระค่าบริการนี้ถูกพัฒนาขึ้นเพื่อความสะดวก ประหยัดเวลา และลดความซับซ้อนของกระบวนการชำระหนี้ การพัฒนาระบบชำระค่าบริการแบบชำระทันที โดยอิงขั้นตอนการซื้อ-ขายของระบบ e-commerce จะช่วยให้ผู้ใช้งานรู้สึกคุ้นชินได้อย่างรวดเร็วโดยไม่ต้องเสียเวลาเรียนรู้ระบบใหม่ตั้งแต่ต้น ทำให้ผู้ใช้งานรู้สึกพึงพอใจที่จะเข้าสู่ขั้นตอนชำระหนี้สินได้ง่ายขึ้น อีกทั้งยังสามารถตรวจสอบรายการย้อนหลังผ่านระบบของ Pumpkin ได้ ทำให้ลดภาระด้านการจัดเก็บข้อมูลโดยใช้เอกสารที่เลี่ยงต่อการสูญหายหรือถูกทำลายในภายหลัง

5.2 ปัญหาและอุปสรรคในการดำเนินงาน

การพัฒนาระบบชำระค่าบริการรายเดือนติดข้อจำกัดการทำงานบาง Webservice ของ OMISE payment gateway ทำให้ไม่สามารถนำมาใช้งานกับระบบชำระค่าบริการรายเดือนโดยตรงได้ (อธิบายรายละเอียดไว้ใน บทที่ 3 หัวข้อที่ 3.4.2.1)

5.3 วิธีการแก้ปัญหา

เสนอแนวทางการขอใช้ API พิเศษจาก OMISE payment gateway และวางแผนการปรับใช้ API พิเศษที่ได้รับมา (อธิบายรายละเอียดไว้ใน บทที่ 3 หัวข้อที่ 3.4.2.2)

5.4 ข้อเสนอแนะ

ในการทำสหกิจศึกษาใช้ความรู้ความสามารถนอกห้องเรียนเป็นอย่างมาก หากมีหัวข้อที่สนใจควรเริ่มฝึกทำก่อน การรู้ก่อนจะทำให้ลดระยะเวลาศึกษาในการทำงานจริงลงมากและลดความคลาดเคลื่อนต่อตารางเวลาในการทำสหกิจ นักศึกษาต้องมีความใคร่ในการเรียนรู้สิ่งใหม่ ๆ มีความอดทนและไม่ย่อท้อต่ออุปสรรค มีความยืดหยุ่นในกระบวนการคิด และเปิดใจรับสิ่งใหม่ ๆ อย่าหยุดขวยขวายการหาความรู้เพราะการทำงานจริงมีสิ่งที่ต้องเปลี่ยนแปลงอยู่ตลอดเวลา การรู้จักปรับเปลี่ยนตัวเอง และลดความหยิ่งผยองลง จะทำให้นักศึกษาเรียนรู้ไวและเติบโตขึ้น

ภาคผนวก

โดยโปรแกรมที่นำมาใช้ในการปฏิบัติงานครั้งนี้ได้แก่

- Node.js
- Google Chrome
- Source Tree
- Zeplin
- Postman
- VSCode

ภาคผนวก ก

การพัฒนา RESTful API และการทดสอบการทำงานโดยใช้ Postman

ประวัติของ RESTful API

RESTful หรือ REST ย่อมาจาก Representational state transfer คือ การสร้าง Webservice ชนิดหนึ่งที่ใช้ในการสื่อสารบนระบบ Internet โดยเรียกใช้ผ่านทาง HTTP Methods การทำงานจะเป็น Stateless (ไม่มี Session ทำให้แตกต่างจาก Webservice ประเภทอื่น ๆ อย่าง SOAP หรือ WDSL) และอาศัย URI หรือ URL ของ Request เพื่อค้นหาและประมวลผล แล้วตอบกลับเป็นข้อมูลในรูปแบบของ XML, JSON, HTML สามารถพัฒนาด้วย Programming language ได้หลากหลาย อาทิ JavaScript, Python, GO เป็นต้น

HTTP Methods ควรเลือกใช้ตามความหมาย ดังนี้

- GET ใช้เมื่อต้องการดึงข้อมูลใด ๆ จาก URI/URL part ห้ามมีข้อมูลใน Request body
- POST ใช้เมื่อต้องการสร้างข้อมูล สามารถใส่ข้อมูลลงใน Request body ได้
- PUT ใช้เมื่อต้องการอัพเดทข้อมูล สามารถใส่ข้อมูลลงใน Request body ได้
- DELETE ใช้เมื่อต้องการลบข้อมูล สามารถใส่ข้อมูลลงใน Request body ได้

นอกจากนี้ยังมี HTTP Method อื่น ๆ อีก แต่ที่ใช้งานหลัก ๆ ดังที่ยกตัวอย่างก็เพียงพอแล้ว

ในการปฏิบัติงานพัฒนาระบบ API ของ Pumpkin-CRM เลือกใช้งาน RESTful API โดยใช้ภาษา JavaScript ในการพัฒนา ใช้เทคโนโลยีของ Node.js ในการจำลองสภาพแวดล้อม (Environment) ของเซิร์ฟเวอร์และเพื่อทำให้สามารถสร้างโปรแกรมจาก JavaScript ได้ และเลือกใช้ Express ที่สนับสนุนการทำ Routing, Middleware เพื่อจัดการ Request และ Response ทำให้การพัฒนา API สามารถทำได้ง่ายขึ้น

เมื่อมีการสร้างย่อมตามมาด้วยการทดสอบการทำงานเพื่อป้องกันข้อผิดพลาดที่อาจเกิดขึ้นจากการใช้งานจริงและปัญหาที่ตามมาทีหลัง ในบทนี้จะอธิบายถึงกระบวนการพัฒนา API ตลอดถึงการทดสอบการทำงานที่เกิดขึ้นในการพัฒนาระบบ โดยเลือกใช้เครื่องมือ Postman ในการทดสอบ API ที่ถูกสร้าง

การพัฒนา API ของ Pumpkin-CRM

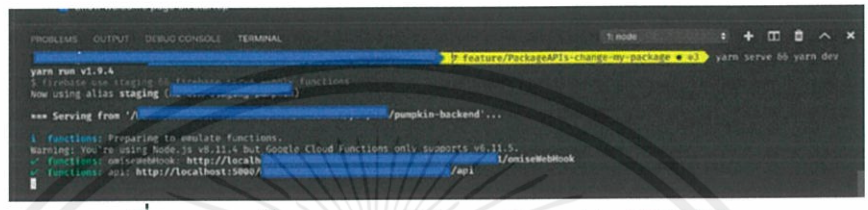
ในขั้นแรกสุด เปิดโครงงานของชื่อ “pumpkin-backend” ซึ่งเป็นโครงงานที่ใช้ในการพัฒนาระบบ API ของ Pumpkin-CRM ด้วยโปรแกรม VSCode ดังรูปที่ ก.1 เปิด Terminal ของโปรแกรม รันคำสั่ง

> yarn serve && yarn dev

yarn serve เป็นคำสั่งให้ระบบ API ของ Pumpkin-CRM เริ่มทำงานโดยใช้เครื่องของผู้พัฒนาเป็นเครื่องเซิร์ฟเวอร์จำลอง โดยกำหนด Port ตำแหน่งที่ 5000 เป็น Port รับ Request (localhost:5000)

yarn dev เป็นคำสั่งให้ระบบ API ของ Pumpkin-CRM เข้าสู่ Dev mode (ประมวลผล Functions โดยไม่ต้องให้ Babel แปลง JavaScript เป็นเวอร์ชั่นดั้งเดิม)

หมายเหตุ Babel ทำหน้าที่แปลง JavaScript เป็นเวอร์ชั่นดั้งเดิมเพื่อ Support ระบบเก่าที่ไม่สามารถประมวล JavaScript สมัยใหม่ได้ การปิดไม่ให้ทำงานเป็นการทำให้ Functions ทำงานเร็วเพราะไม่ต้องแปลงข้อมูล



รูปที่ ก.1 การพัฒนาระบบ API ของ Pumpkin-CRM (1)

เมื่อโครงงานพร้อมทำงาน ก็สามารถเริ่มพัฒนาระบบ API ของ Pumpkin-CRM ได้ทันที

ทดลองสร้าง RESTful API

ในหัวข้อนี้อธิบายถึงกระบวนการสร้าง API ที่ใช้ในโครงงาน โดยเป็นการสร้าง API ตัวใหม่ที่ทำหน้าที่ตอบกลับเป็น Mock data รูปแบบ Object ไปให้ผู้เรียกใช้

ขั้นตอนที่ 1 สร้าง File ขึ้นมาในโครงงานชื่อ “testApi.js” ตำแหน่ง src/functions/controller

ขั้นตอนที่ 2 ใน testApi.js สร้าง Mock data โดยเป็นข้อมูลพื้นฐานของ user คนหนึ่ง

```
const user = {
    name: "John Doe",
    age: 15,
    address: "534 Sugar Street Southgate, MI 48195",
    country: "US",
    phone: "068-065-8148 x65253"
}
```

ขั้นตอนที่ 3 ติดตั้ง API part ใหม่ ใน app.js ตำแหน่ง src/functions/controller/api.js โดยใช้ Part เป็น /test-api ดังรูปที่ ก.2

```
26 import testApi from 'controllers/testApi'
27
28 const app = express()
29 app.use(cors({ origin: true }))
30
31 app.use('/test-api', logRequestPath('test-api'), testApi)
```

รูปที่ ก.2 การพัฒนาระบบ API ของ Pumpkin-CRM (2)

บรรทัดที่ 26 Import file ชื่อ testApi จากตำหน่ง controllers/testApi (ในโครงงานนี้ถูกตั้งค่าให้สามารถใช้ Absolute import ได้ หากไม่มีการตั้งค่าต้องใช้ตำแหน่งเป็น ‘./testApi’)

บรรทัดที่ 28-29 สร้าง Instant จาก Express ชื่อ “app” ซึ่ง File นี้เป็น Root file ที่จะถูกนำไปประมวลผลเพื่อเอา Functions ทั้งหมดนำมาใส่ใน File นี้ไปสร้างระบบ API ของ Pumpkin-CRM

บรรทัดที่ 31 สร้าง API part โดย

> /testApi คือ part ที่จะต่อท้ายจาก API address หลัก

> logRequestPath(‘testApi) คือ Middleware คอยดัก Request ที่เรียก API part นี้เพื่อรายงานระบบว่า Request นี้ถูกส่งมาจากที่ใด

> testApi คือ สุดท้ายแล้วเมื่อ Request ผ่าน Middleware จะถูกส่งไปให้ฟังก์ชันใน testApi ทำงาน

ขั้นตอนที่ 4 กลับมาที่ testApi.js เขียน code ตามรูปที่ ก.3

```
1   import express from 'express'
2
3   const app = express()
4
5   const user = {
6     name: 'John Doe',
7     age: 15,
8     address: '534 Sugar Street Southgate, MI 48195',
9     country: 'US',
10    phone: '068-065-8148 x65253'
11  }
12
13
14
15  export default app
16
```

รูปที่ ก.3 การพัฒนาระบบ API ของ Pumpkin-CRM (3)

บรรทัดที่ 1 เรียกใช้ Express framework

บรรทัดที่ 3 สร้าง Instant ของ Express ชื่อ “app”

บรรทัดที่ 5-11 Mock data ที่เตรียมสำหรับเป็นข้อมูลตอบกลับไปหา Request

บรรทัดที่ 15 คำสั่งให้ File ใด ๆ ที่ต้องการเรียก API ใน File ไปใช้งาน สามารถกำหนดชื่อได้เองตอน Import ดังขั้นตอนที่ 3 บรรทัดที่ 26

ขั้นตอนที่ 5 ใน testApi.js บรรทัดที่ 13 สร้าง API ขึ้นมาตาม code ด้านล่าง

```
app.post('/user', (req, res, next) => {
  const { token } = req.body
  if (token) return res.send(user)
  next(new Error(“please login”))
})
```

การทำงานของ API นี้คือ Request ที่ Part เป็น ".../test-api/user" ใช้ POST method ถ้ามีข้อมูล token ถูกส่งมาพร้อมกับ Request body ให้ส่งข้อมูล Mock data ของ user ตอบกลับไป แต่ถ้าไม่มีให้ส่ง Error ตอบกลับไป สุดท้ายใน testApi.js จะเป็นดังรูปที่ ก.4

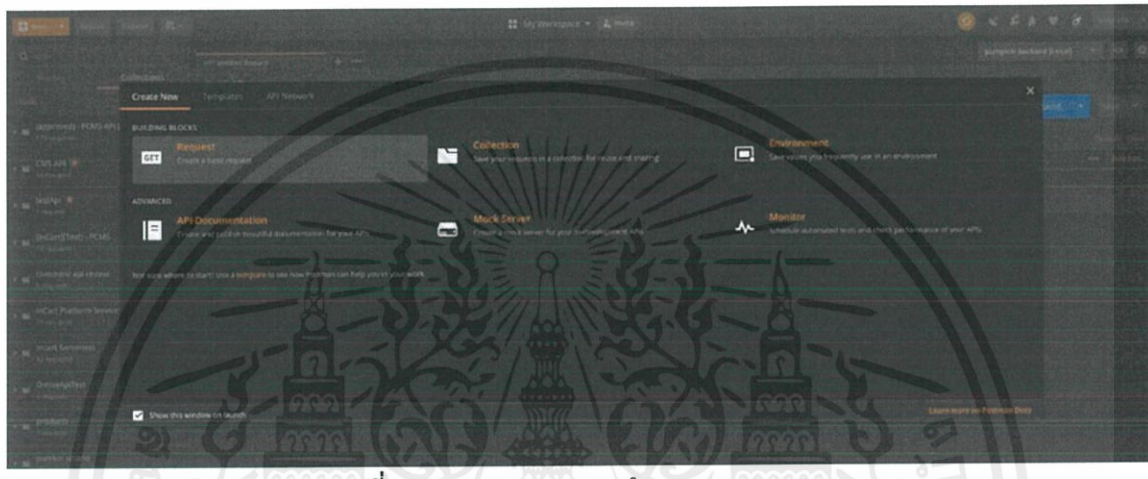
```
import express from 'express'

const app = express()

const user = {
  name: 'John Doe',
  age: 15,
  address: '534 Sugar Street Southgate, MI 48195',
  country: 'US',
  phone: '068-065-8148 x65253'
}

app.post('/user', (req, res, next) => {
  const { token } = req.body
  if (token) return res.send(user)
  next(new Error('please login'))
})

export default app
```

รูปที่ ก.4 การพัฒนาระบบ API ของ Pumpkin-CRM (4)

เมื่อจบขั้นตอนที่ 5 แล้วระบบก็จะมี API เพิ่มขึ้นมาอีก 1 ตัว ขั้นตอนต่อไปคือการทดสอบการทำงานของ API ที่เขียน

การทดสอบการทำงานของ API

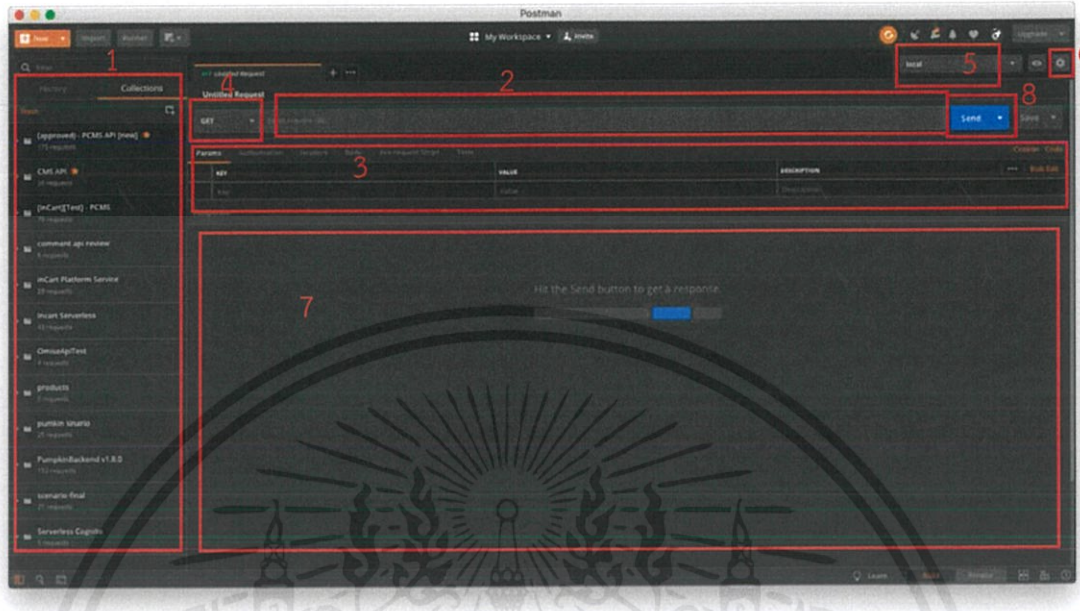
เปิดโปรแกรม Postman ขึ้นมา กด crl+n ใน windows หรือ cmd+n ใน macOS จะปรากฏหน้าต่างสำหรับเลือกสร้างดังรูปที่ ก.5 กดที่การสร้าง Request จะปรากฏหน้าให้กรอกรายละเอียดของ Request ที่ต้องการสร้างดังรูปที่ ก.6 เลือกสร้าง Request ตั้งชื่อเป็น “user” เก็บไว้ใน Collection ชื่อ “testApi” (ควรใช้ชื่อเดียวกับชื่อของ API ที่ใช้ทดสอบเพื่อป้องกันความสับสน)



รูปที่ ก.5 การทดสอบการทำงานของ API (1)

รูปที่ ก.6 การทดสอบการทำงานของ API (2)

เมื่อกด save แล้วจะปรากฏหน้าต่างดังรูปที่ ก.7



รูปที่ ก.7 การทดสอบการทำงานของ API (3)

หมายเลขที่ 1 Postman มี Collection แยก Script ที่เคยสร้างเป็นส่วน ๆ ตามโฟลเดอร์ ทำให้การทดสอบเป็นระเบียบ หากใช้รุ่นที่มีการจ่ายค่าลิขสิทธิ์จะสามารถแชร์ Collection กับผู้อื่นได้

หมายเลขที่ 2 URL part ที่ API นั้น ๆ อยู่

หมายเลขที่ 3 ส่วนตั้งค่า Request ที่ใช้ทดสอบ API

หมายเลขที่ 4 ชนิด Request ที่ส่งไปหา API (GET, PUT, POST, DELETE etc.)

หมายเลขที่ 5 Environment ที่เลือกใช้กับการทดสอบ API

หมายเลขที่ 6 ปุ่มตั้งค่า Environment

หมายเลขที่ 7 ส่วนแสดงผลลัพธ์เมื่อ API นั้นทำงานสำเร็จ

หมายเลขที่ 8 ปุ่มกดส่ง Request เพื่อเรียก API

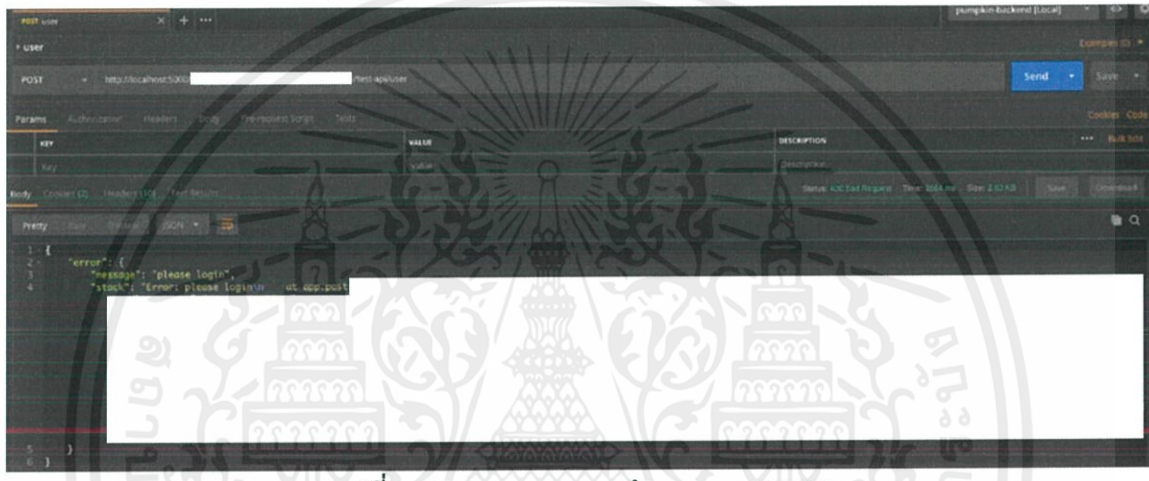
ใส่ URL ของ API address ในตำแหน่งหมายเลขที่ 2 ในที่นี้คือ

> http://localhost:5000/.../test-api/user

เลือก Request method เป็น POST ในตำแหน่งหมายเลขที่ 4 ให้ตรงกับ API ที่ทดสอบดังรูปที่ ก.8

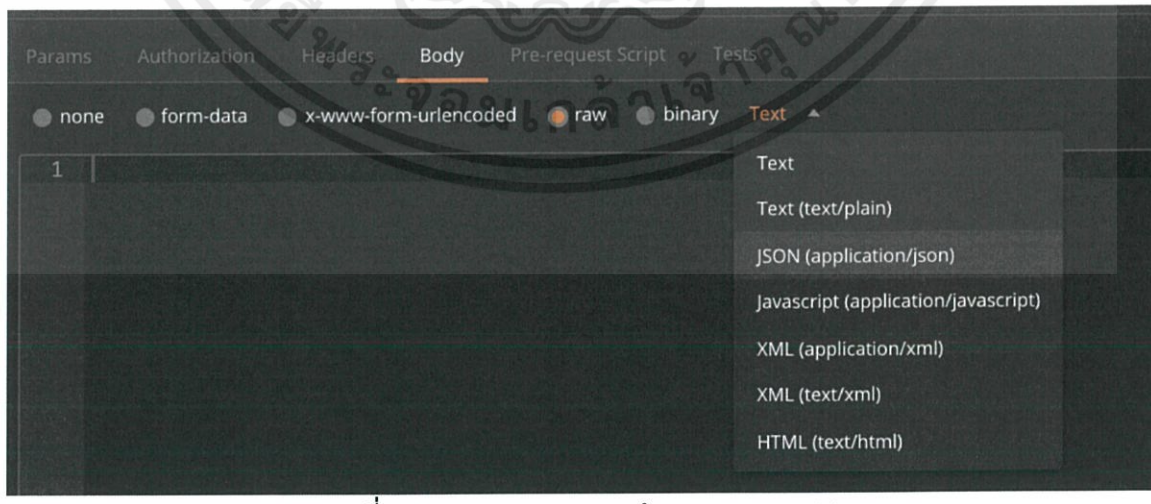
รูปที่ ก.8 การทดสอบการทำงานของ API (4)

เมื่อกด Send จะส่ง Request ไปยัง API address ที่ใส่ไว้ซึ่งผลตอบกลับของ API ปรากฏดังรูปที่ ก.9

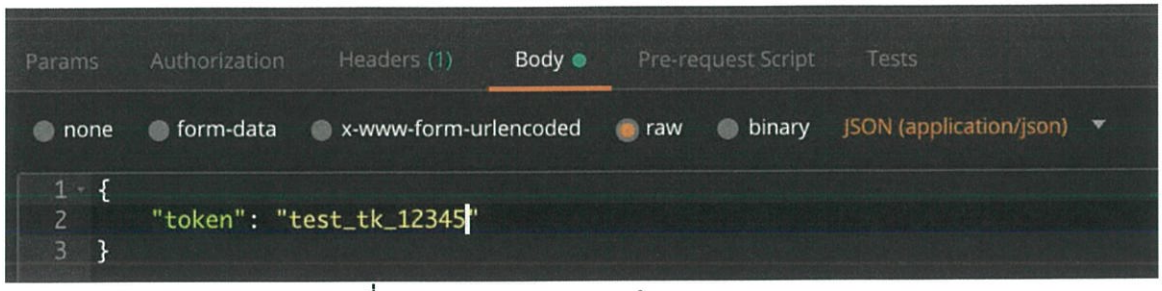


รูปที่ ก.9 การทดสอบการทำงานของ API (5)

สาเหตุที่ API ตอบ Error กลับมาเนื่องจาก API ที่ทดสอบ ตรวจว่า Request ที่เรียก API นี้ มี token มากับ Request body ด้วยหรือไม่ ซึ่งขั้นตอนก่อนหน้าไม่มีการใส่ไว้ ถือว่าการทำงานเป็นไปตามที่เรากำหนดไว้ 1 ข้อ ต่อมาทำการเพิ่ม token เข้าไปใน Request body ดังรูปที่ ก.10 และ ก.11



รูปที่ ก.10 การทดสอบการทำงานของ API (6)



รูปที่ ก.11 การทดสอบการทำงานของ API (7)

เมื่อกด Send จะส่ง Request อีกครั้งผลตอบกลับของ API ปรากฏดังรูปที่ ก.12

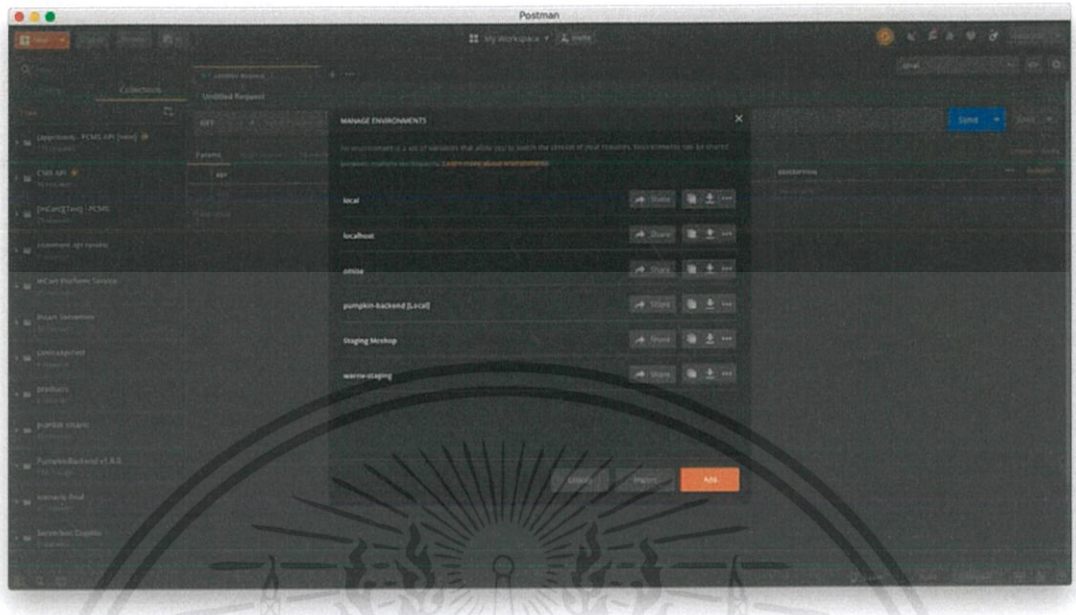
รูปที่ ก.12 การทดสอบการทำงานของ API (8)

ซึ่ง API ส่ง Mock data ของ user กลับมาให้ตรงตามเงื่อนไขที่กำหนดไว้ ดังนั้นจึงสรุปว่า API ที่ทดสอบสามารถใช้งานได้

การตั้งค่าสภาพแวดล้อมของ Postman

การทดสอบหรือพัฒนา API ในหลาย ๆ Environment เป็นสิ่งที่หลีกเลี่ยงไม่ได้ Postman สามารถตั้งค่าบางข้อมูลให้เป็นตัวแปร (Variable) อย่างเช่น URL หรือ key ทำให้ผู้ทดสอบไม่จำเป็นต้องแก้ไข URL หรือ Variable บ่อย ๆ เมื่อเปลี่ยนสถานการณ์ทดสอบก็สามารถเลือกใช้การตั้งค่า Environment ที่เคย set ไว้ ที่เหมาะสมกับสถานการณ์ทดสอบนั้น ๆ ทำให้ลดการผิดพลาดและลดเวลาทดสอบลงได้

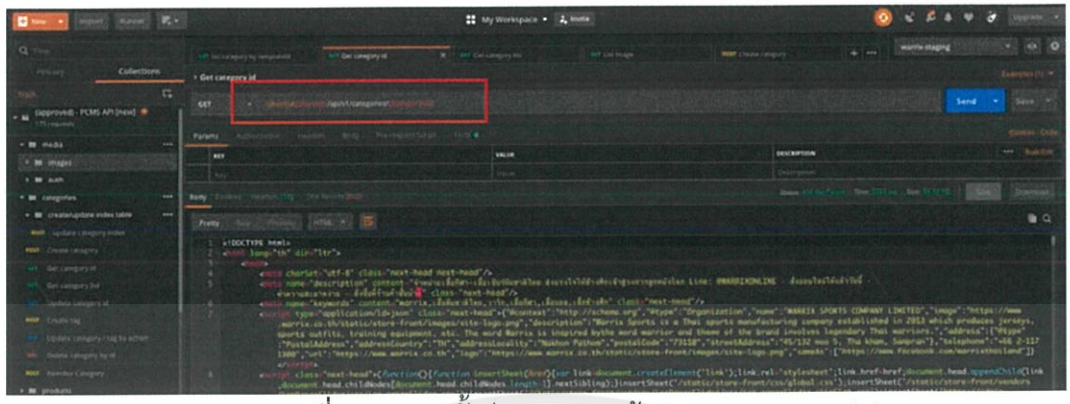
ทำการตั้งค่าตัวแปรที่จำเป็นในการทดสอบโดยการกดที่รูปเฟืองในตำแหน่งหมายเลขที่ 6 ของรูปที่ ก.7 จะปรากฏหน้าต่างแสดงรายการ Environment ที่เคยตั้งค่าไว้ดังรูปที่ ก.13 และสามารถสร้าง Environment ใหม่หรือแก้ไข Environment เดิมจะปรากฏหน้าต่างให้สร้างตัวแปรที่ใช้ใน Environment นั้น ๆ ดังรูปที่ ก.14



รูปที่ ก.13 การตั้งค่าสภาพแวดล้อมของ Postman (1)

รูปที่ ก.14 การตั้งค่าสภาพแวดล้อมของ Postman (2)

สุดท้ายเมื่อตั้งค่าตัวแปรที่จำเป็นสำหรับ Environment นั้น ๆ แล้ว ก็สามารเลือกใช้ ตัวแปรที่กำหนดไว้ใน Environment ได้ ดังรูปที่ ก.15 จะเห็นว่าตำแหน่ง URL part สามารถใช้ตัวแปรแทนการเขียน URL ตรง ๆ ได้ ทำให้เพิ่มความรวดเร็วในการทดสอบ API



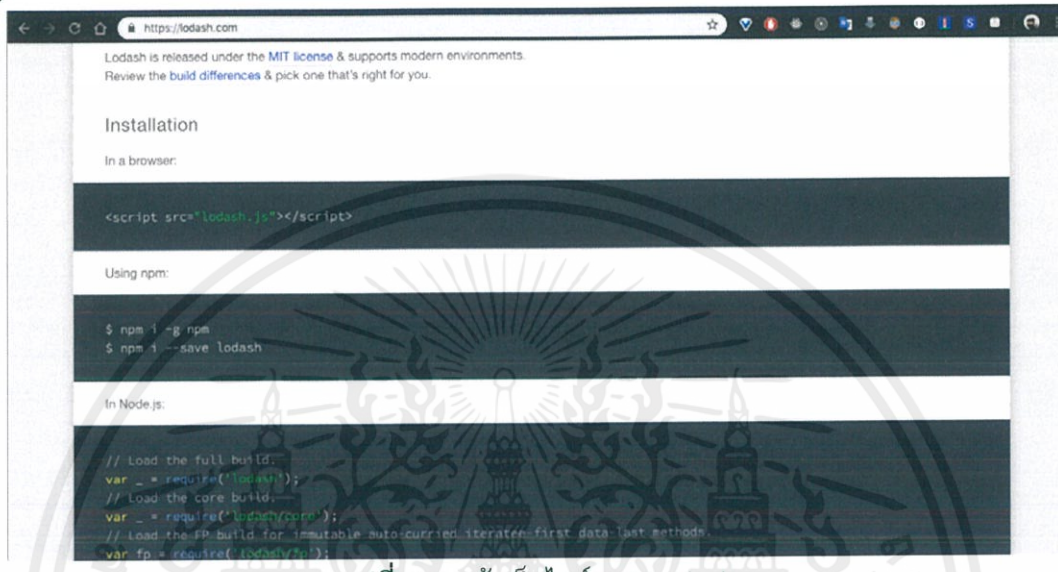
รูปที่ ก.15 การตั้งค่าสภาพแวดล้อมของ Postman (3)

ภาคผนวก ข

วิธีติดตั้ง JS library ลงในโครงงาน

ในบทนี้ขอยกตัวอย่างการติดตั้ง Lodash ซึ่งเป็น JS library ที่ใช้ใน Project นี้

ขั้นตอนแรกให้เข้าไปดูหน้าเว็บไซต์ของ library ที่ต้องการติดตั้ง เลื่อนหาหมวดแสดงวิธีติดตั้ง ดังรูปที่ ข.1



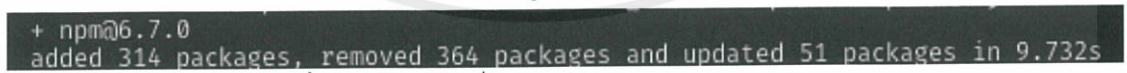
รูปที่ ข.1 หน้าเว็บไซต์ของ Lodash

คัดลอก Script ที่ได้มาจากเว็บไซต์ของ Library ในที่นี้ เลือกใช้วิธีของ npm ไปรันบน Terminal ของ VSCode ดังรูปที่ ข.2

หมายเหตุ ถ้าไม่สามารถติดตั้งได้ให้ใส่ sudo หน้า Script ที่คัดลอกมา เช่น sudo npm i -g npm

รูปที่ ข.2 Terminal ของ VSCode

เมื่อติดตั้งสำเร็จจะมีข้อความแสดงคล้ายรูปที่ ข.3



รูปที่ ข.3 ข้อความที่แสดงบน Terminal ของ VSCode

สามารถนำไปประยุกต์ใช้กับการติดตั้ง JS library อื่น ๆ ได้ ซึ่งขั้นตอนและวิธีการติดตั้งอาจเปลี่ยนแปลงไปตามรายละเอียดของ JS library อื่น ๆ

เอกสารอ้างอิง

[1] Storybook [ระบบออนไลน์] แหล่งที่มา : https://bit.ly/2TKNmFq (15 มกราคม 2562)

[2] Cloud Firestore [ระบบออนไลน์] แหล่งที่มา : https://bit.ly/2MfOCRt (15 มกราคม 2562)

[3] Node.js [ระบบออนไลน์] แหล่งที่มา :
http://www.siamhtml.com/introduction-to-node-js/
https://devahoy.com/posts/getting-started-with-nodejs/
(15 มกราคม 2562)

[4] Express [ระบบออนไลน์] แหล่งที่มา : https://bit.ly/2tO6asG (15 มกราคม 2562)

[5] Visual Studio Code [ระบบออนไลน์] แหล่งที่มา : https://bit.ly/2TGWATR (15 มกราคม 2562)

[6] Source tree [ระบบออนไลน์] แหล่งที่มา : https://bit.ly/2D67QTE (15 มกราคม 2562)

[7] Google chrome [ระบบออนไลน์] แหล่งที่มา : https://bit.ly/2J8RGtZ (15 มกราคม 2562)

[8] Zeplin [ระบบออนไลน์] แหล่งที่มา :
https://bit.ly/2H84XFF
https://bit.ly/2A3UHJj
(15 มกราคม 2562)

[9] Netflix [ระบบออนไลน์] แหล่งที่มา : https://help.netflix.com/th/node/412
(15 มกราคม 2562)

[10] OMISE payment gatway [ระบบออนไลน์] แหล่งที่มา : https://www.omise.co/th/docs
(15 มกราคม 2562)

[11] SellSUKI [ระบบออนไลน์] แหล่งที่มา : https://www.sellsuki.co.th/ (15 มกราคม 2562)

[12] LnwSHOP [ระบบออนไลน์] แหล่งที่มา : https://www.lnwshop.com/ (15 มกราคม 2562)

[13] Atomic structure [ระบบออนไลน์] แหล่งที่มา :
https://github.com/Rulox/react-atomic-structure
https://bit.ly/2DcoL6Q
(15 มกราคม 2562)

[14] Postman [ระบบออนไลน์] แหล่งที่มา : https://bit.ly/2TczMLn (15 มกราคม 2562)

[15] RESTful [ระบบออนไลน์] แหล่งที่มา : https://saixiii.com/what-is-restful/ (15 มกราคม 2562)