

ระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง
Inventory Management System for SMEs

วิชชุดา	ทองก้อน	6504101385
ศุภเดช	เดชคำ	6504101398

วิทยาศาสตร์บัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ มหาวิทยาลัยแม่โจ้
ปีการศึกษา 2567

ระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง
Inventory Management System for SMEs

ระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง
Inventory Management System for SMEs

วิชชุดา	ทองก้อน	6504101385
ศุภเดช	เดชคำ	6504101398

โครงงานนี้นำเสนอต่อสาขาวิชาวิทยาการคอมพิวเตอร์เพื่อเป็นส่วนหนึ่ง
ของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ มหาวิทยาลัยแม่โจ้
ปีการศึกษา 2567
ระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง
Inventory Management System for SMEs

ระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง
Inventory Management System for SMEs

วิชาชุดา	ทองก้อน	6504101385
ศุภเดช	เดชคำ	6504101398

โครงการนี้ได้รับการพิจารณาอนุมัติให้เป็นส่วนหนึ่งของการศึกษา
ตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์

คณะกรรมการสอบโครงการ

.....ประธาน

กรรมการ

ผู้ช่วยศาสตราจารย์ ดร.สมนึก สินธุพาน

กรรมการ

ผู้ช่วยศาสตราจารย์ ดร.สนิท สิทธิ

กรรมการ

ผู้ช่วยศาสตราจารย์ ภาณุวัฒน์ เมฆะ

ปีการศึกษา 2568

บทคัดย่อ

โครงการนี้มีวัตถุประสงค์เพื่อพัฒนา ระบบคลังสินค้าในรูปแบบเว็บแอปพลิเคชัน (Web Application) สำหรับธุรกิจขนาดกลางและขนาดย่อม (SMEs) เพื่อเพิ่มประสิทธิภาพในการบริหารจัดการสินค้าคงคลัง ลดข้อผิดพลาดจากการบันทึกข้อมูลด้วยวิธีแมนนวล และสนับสนุนการเข้าถึงข้อมูลแบบเรียลไทม์ผ่านอุปกรณ์คอมพิวเตอร์ ระบบถูกออกแบบให้ใช้งานง่าย มีความประหยัดต้นทุน และเหมาะสำหรับการใช้งานผ่านเว็บเบราว์เซอร์บนคอมพิวเตอร์

ระบบประกอบด้วยฟังก์ชันสำคัญ ได้แก่ การบันทึกข้อมูลการนำเข้าและเบิกออกสินค้า การติดตามสถานะสินค้าคงคลัง การแจ้งเตือนเมื่อสินค้าต่ำกว่าระดับขั้นต่ำ การจัดการหมวดหมู่สินค้าและบัญชีผู้ใช้ รวมถึงการออกรายงานในรูปแบบ Dashboard ที่เข้าใจง่าย พร้อมรองรับการส่งออกข้อมูลเป็น PDF และ Excel

ในการพัฒนาระบบ ใช้เทคโนโลยี React.js ร่วมกับ Inertia.js สำหรับส่วนติดต่อผู้ใช้ (Frontend) และ Laravel ซึ่งเป็น PHP Framework สำหรับการจัดการฐานข้อมูลและตรรกะทางธุรกิจ (Backend) โดยใช้ฐานข้อมูล MySQL การทดสอบระบบดำเนินไปอย่างครบถ้วนทั้งในระดับฟังก์ชันย่อยและระบบโดยรวม เพื่อปรับปรุงประสบการณ์ผู้ใช้และเสถียรภาพของระบบ

ผลการพัฒนาพบว่า ระบบช่วยให้ผู้ใช้งานสามารถจัดเก็บและบริหารข้อมูลสินค้าได้อย่างถูกต้อง แม่นยำ และรวดเร็ว ลดข้อผิดพลาดที่เกิดจากการจัดการแบบเดิมได้อย่างชัดเจน นอกจากนี้ ระบบยังสนับสนุนการวิเคราะห์ข้อมูลเพื่อการตัดสินใจทางธุรกิจเบื้องต้นได้อย่างมีประสิทธิภาพ และถูกออกแบบให้สามารถขยายต่อ ยอดเชื่อมต่อกับระบบ POS และระบบบัญชีในอนาคตได้อย่างยืดหยุ่น เหมาะสำหรับธุรกิจ SMEs ที่ต้องการระบบคลังสินค้าทันสมัยสำหรับการใช้งานผ่านเครื่องคอมพิวเตอร์

Abstract

This project aims to develop a warehouse management system in the form of a web application for small and medium-sized enterprises (SMEs). The purpose is to enhance efficiency in inventory management, reduce errors from manual record-keeping, and support real-time data access via computer devices. The system is designed to be user-friendly, cost-effective, and suitable for access through web browsers on computers.

The system includes key functions such as recording product imports and withdrawals, tracking inventory status, sending alerts when stock falls below the minimum threshold, managing product categories and user accounts, and generating easy-to-understand dashboard reports. It also supports data export in PDF and Excel formats. For system development, React.js combined with Inertia.js was used for the user interface (Frontend), while Laravel, a PHP framework, was employed for business logic and database management (Backend) with MySQL as the database. System testing was conducted comprehensively at both the unit and overall levels to improve user experience and ensure system stability.

The results of the development show that the system enables users to store and manage product data accurately, efficiently, and quickly, significantly reducing errors compared to traditional methods. Furthermore, the system supports preliminary business decision-making through data analysis and is designed to be scalable, allowing future integration with POS and accounting systems. It is particularly suitable for SMEs that require a modern warehouse management system accessible via compute

สารบัญ

เรื่อง	หน้า
บทคัดย่อภาษาไทย	ง
บทคัดย่อภาษาอังกฤษ	จ
สารบัญตาราง	ญ
สารบัญภาพ	ฎ
บทที่ 1 บทนำ	1
1.1 ที่มาและความสำคัญของปัญหา	1
1.2 วัตถุประสงค์	2
1.3 ฮาร์ดแวร์ ซอฟต์แวร์ เครื่องมือ และเทคโนโลยีที่ใช้ในการศึกษา	2
1.4 วิธีการศึกษา	3
1.5 ขอบเขตของโครงการ	3
1.6 ระยะเวลาดำเนินงาน	5
1.7 ผลที่คาดว่าจะได้รับ	6
บทที่ 2 การศึกษาค้นคว้าและงานที่เกี่ยวข้อง	7
2.1 ระบบสินค้าคงคลัง (Inventory System)	7
2.1.1 วัตถุประสงค์ระบบสินค้าคงคลัง (Inventory System)	7
2.2 แนวคิดหลักของแบบจำลอง EOQ (Economic Order Quantity)	9
2.2.1 วัตถุประสงค์ของแบบจำลอง EOQ (Economic Order Quantity)	9
2.2.2 ต้นทุนที่เกี่ยวข้อง	10
2.2.3 การทำงานของแบบจำลอง	11
2.2.4 สูตรคำนวณ	12
2.2.5 สมมติฐานของแบบจำลอง EOQ (Economic Order Quantity)	13
2.3 การจัดการสินค้าคงคลังแบบ FIFO (First-In, First-Out)	14
2.3.1 ลักษณะการทำงาน	14

	หน้า
2.3.2 ข้อดีของ FIFO (First-In, First-Out)	15
2.3.3 ข้อจำกัดของ FIFO (First-In, First-Out)	15
2.3.4 เทคโนโลยีที่สนับสนุนการทำ FIFO	15
2.3.5 หลักการสำคัญของ FIFO (First-In, First-Out)	15
2.4 ต้นทุนขาย (Cost of Goods Sold: COGS)	16
2.5 ต้นทุนถัวเฉลี่ย (Average Cost)	17
 บทที่ 3 เทคโนโลยีที่ใช้ในการศึกษา	18
3.1 เทคโนโลยีด้านฮาร์ดแวร์ (Hardware Technology)	18
3.2 เทคโนโลยีด้านซอฟต์แวร์ (Software Technology)	20
3.2.1 ส่วนติดต่อผู้ใช้ (Frontend)	20
3.2.2 ฝั่งเซิร์ฟเวอร์ (Backend)	30
3.2.3 Visual Studio Code (VS Code)	43
3.2.4 JavaScript	44
3.2.6 MySQL	44
3.3 เทคโนโลยีด้านฐานข้อมูล (Database Technology)	44
3.4 การรักษาความปลอดภัยและการจัดการผู้ใช้ (Security & User Management)	53
3.4.1 การยืนยันตัวตน (Authentication)	53
3.4.2 การกำหนดสิทธิ์การใช้งาน (Authorization & Role Management)	54
3.4.3 การเข้ารหัสรหัสผ่าน (Password Hashing)	54
3.4.4 การป้องกัน CSRF (Cross-Site Request Forgery Protection)	54

สารบัญ (ต่อ)

	หน้า
3.4.5 การจัดการ Session และ Token	54
บทที่ 4 การวิเคราะห์ระบบและออกแบบระบบ	56
4.1 การวิเคราะห์ระบบ	56
4.1.1 การวิเคราะห์ระบบปัจจุบัน	56
4.1.2 ปัญหาและข้อจำกัด	57
4.1.3 ความต้องการของผู้ใช้งาน	59
4.2 การออกแบบระบบ	61
4.2.1 การออกแบบสถาปัตยกรรมระบบ (System Architecture Design)	61
4.2.2 การออกแบบกระบวนการทำงานของระบบ (Process Design)	63
4.2.3 การออกแบบส่วนติดต่อผู้ใช้งาน (User Interface Design)	65
4.3 การออกแบบฐานข้อมูล (Database Design)	67
4.3.1 การกำหนดตารางข้อมูล (Database Tables)	67
4.3.2 การออกแบบความสัมพันธ์ (ER-Diagram)	70

สารบัญตาราง

ตาราง		หน้า
1.6	ตารางรายละเอียดกิจกรรมและระยะเวลาการดำเนินการ	5
3.3 (1)	ตาราง users	44
3.3 (2)	ตาราง categories	45
3.3 (3)	ตาราง products	46
3.3 (4)	ตาราง sales	47
3.3 (5)	ตาราง sale_items	48
3.3 (6)	ตาราง returns	49
3.3 (7)	ตาราง return_items	49
3.3 (8)	ตาราง receipts	50
3.3 (9)	ตาราง receipt_items	51
3.3 (10)	ตาราง stock_movements	53
3.4	ตารางเปรียบเทียบสิทธิการใช้งานของผู้ใช้ระบบ	55

สารบัญภาพ

รูป		หน้า
2.2.3	การทำงานของแบบจำลอง EOQ	11
2.3.5	แผนภาพการทำงานของระบบ FIFO	16
3.1(1)	เครื่องสแกนบาร์โค้ด	18
3.1(2)	แสดงโครงสร้างการเชื่อมต่อเครือข่ายของระบบคลังสินค้า	19
3.2.1(1)	XAMPP Control Panel v3.3.0	20
3.2.2(1)	แสดงหน้าต้อนรับ (Welcome Page) ของระบบคลังสินค้า	22
3.2.2(2)	แสดงหน้าล็อกอิน (Login Page) ของระบบคลังสินค้า	23
3.2.2(3)	แสดงหน้าขายสินค้า (POS)	24
3.2.2(4)	แสดงหน้าชำระเงิน (Checkout)	25
3.2.2(5)	แสดงใบเสร็จอิเล็กทรอนิกส์	25
3.2.2(6)	แสดงหน้าหลักของระบบคีนสินค้า	26
3.2.2(7)	แสดงรายละเอียดใบเสร็จและรายการสินค้าที่สามารถเลือกคืนได้	26
3.2.2(8)	แสดง Dashboard ของผู้ดูแลระบบ (Admin)	27
3.2.2(9)	แสดง Dashboard ของพนักงาน (Staff)	28
3.2.2(10)	แสดงหน้ารายงาน (Reports Page) ในมุมมองแบบรายการ (Table View)	29
3.2.2(11)	แสดงหน้ารายงาน (Reports Page) ในมุมมองแบบกราฟ (Graph View)	29
3.2.2(1)	แสดงไฟล์เตอร์ Controllers ของ Laravel ที่ใช้ควบคุมการทำงานของระบบ	31
3.2.2(2)	แสดงไฟล์เตอร์ Models ของ Laravel ที่แทนตารางในฐานข้อมูล	32
3.2.2(3)	แสดงไฟล์เตอร์ routes	33
3.2.2(4)	แสดงไฟล์เตอร์ Components	34
3.2.2(5)	แสดงไฟล์เตอร์ Layouts	35
3.2.2(6)	แสดงโครงสร้างไฟล์เตอร์ Pages/Admin	36
3.2.2(7)	แสดงโครงสร้างไฟล์เตอร์ Pages/Auth, POS และ Profile	37
3.2.2(8)	แสดงโครงสร้างไฟล์เตอร์ Pages/Staff, StockMovements และไฟล์ Welcome.jsx	38

3.2.2(9)	แสดงไฟล์เตอร์ database ของ Laravel	38
3.2.2(10)	แสดงไฟล์เตอร์ public ของ Laravel	39
3.2.2(11)	แสดงโครงสร้างภายในไฟล์เตอร์ public	40
3.2.2(12)	แสดงไฟล์เตอร์ config ของ Laravel	41
3.2.2(13)	แสดงการใช้ Composer เพื่อติดตั้งแพ็คเกจที่จำเป็นสำหรับ Laravel	42
3.2.2(14)	แสดงไฟล์ composer.json ของโครงการ	43
3.4.1	แสดงหน้า Login	53
3.4.5	แสดงหน้าจัดการผู้ใช้ (User Management)	54
4.1.1	Use-Case Diagram ของระบบปัจจุบัน (Manual System)	57
4.1.3	Use-Case Diagram ของระบบที่พัฒนา (Proposed System)	61
4.2.1	แผนภาพสถาปัตยกรรมระบบ (System Architecture Design)	62
4.2.2(1)	Context Diagram ของระบบคลั่งสินค้า (Level 0)	63
4.2.2(2)	Data Flow Diagram ระดับ 1 ของระบบคลั่งสินค้า แสดงกระบวนการย่อยหลัก 5 กระบวนการ	65
4.2.3	ตัวอย่างการออกแบบส่วนติดต่อผู้ใช้งาน (User Interface Design)	66
4.3.2	แสดง ER-Diagram ของระบบคลั่งสินค้า	71
4.4.5	ระบบคลั่งสินค้าสำหรับ SMEs: การจัดการผู้ใช้และสิทธิ์	74

บทที่ 1

บทนำ

ระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง (Inventory Management System for SMEs) มีรายละเอียดดังนี้

1.1 ที่มาและความสำคัญของโครงการ

ในปัจจุบัน การบริหารจัดการคลังสินค้าเป็นองค์ประกอบสำคัญที่ส่งผลโดยตรงต่อประสิทธิภาพการดำเนินงานของธุรกิจเกือบทุกประเภท โดยเฉพาะอย่างยิ่งในกลุ่มธุรกิจขนาดกลางและขนาดย่อม (SMEs) ที่ประสบข้อจำกัดด้านทรัพยากร บุคลากร และความพร้อมทางเทคโนโลยี ส่งผลให้ร้านค้าหรือธุรกิจเหล่านี้ ยังคงพึ่งพาวิธีการบันทึกข้อมูลสินค้าแบบแมนนวลหรือใช้ซอฟต์แวร์พื้นฐาน เช่น Microsoft Excel ซึ่งมีความเสี่ยงต่อข้อผิดพลาด ข้อมูลซ้ำซ้อน และความล่าช้าในการตรวจสอบย้อนหลัง

แม้ว่าจะมีระบบบริหารคลังสินค้าหลายรูปแบบในท้องตลาด แต่ระบบเหล่านั้นมักมีความซับซ้อนในการใช้งาน ต้นทุนสูง และไม่ตอบโจทย์ความต้องการเฉพาะของธุรกิจ SMEs เช่น ความสามารถในการเข้าถึงข้อมูลจากอุปกรณ์พกพา การแจ้งเตือนสินค้าใกล้หมด หรือการแสดงผลข้อมูลในรูปแบบที่เข้าใจง่ายและสนับสนุนการตัดสินใจทางธุรกิจได้อย่างรวดเร็ว

จากปัญหาดังกล่าว ผู้พัฒนาจึงเห็นความจำเป็นในการจัดทำโครงการนี้ขึ้น โดยมีเป้าหมายเพื่อพัฒนา “ระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง” ในรูปแบบเว็บแอปพลิเคชัน (Web Application) ที่ใช้งานง่าย ประหยัดต้นทุน และสามารถเข้าถึงได้ทุกที่ทุกเวลา รองรับฟีเจอร์ที่ตอบโจทย์ SMEs โดยเฉพาะ เช่น การสแกนบาร์โค้ด การแนะนำการเติมสินค้าอัตโนมัติ และระบบ Dashboard วิเคราะห์ต้นทุน-กำไรเบื้องต้น ทั้งยังสามารถต่อยอดการพัฒนาในอนาคตเพื่อรองรับการเชื่อมต่อกับระบบบัญชี ระบบ POS หรือระบบวิเคราะห์ข้อมูลได้อีกด้วย

1.2 วัตถุประสงค์

- 1.2.1 เพื่อพัฒนาระบบคลังสินค้าที่สามารถจัดเก็บและบริหารข้อมูลสินค้าได้อย่างเป็นระบบและมีการกำหนดสิทธิในการเข้าถึงคลังสินค้าผู้ใช้(User)
- 1.2.2 เพื่ออำนวยความสะดวกแก่ผู้ใช้งานในการบันทึก ตรวจสอบ และติดตามสถานะสินค้าแบบเรียลไทม์ผ่านเว็บเบราว์เซอร์
- 1.2.3 เพื่อเพิ่มความแม่นยำในการควบคุมจำนวนสินค้าคงคลัง ลดข้อผิดพลาดจากการจัดการด้วยQR Code
- 1.2.4 เพื่อออกรายงานสินค้าคงเหลือ รายงานการเคลื่อนไหวของสินค้า และรองรับการวิเคราะห์ต้นทุนและข้อมูลเบื้องต้นในรูปแบบ Dashboard

1.3 ฮาร์ดแวร์ ซอฟต์แวร์ เครื่องมือ และเทคโนโลยีที่ใช้ในการศึกษา

1.3.1 ฮาร์ดแวร์

1. โน้ตบุ๊กคอมพิวเตอร์ (Notebook Computer)
 - รุ่น Asus tuf gaming A15 1 เครื่อง
 - รุ่น Asus TUF Gaming A16 Advantage Edition FA617NSR-N3016W 1 เครื่อง

1.3.2 ซอฟต์แวร์

1. ซอฟต์แวร์ระบบปฏิบัติการ
 - ระบบปฏิบัติการ Windows 11 ใช้สำหรับแสดงผล
2. ซอฟต์แวร์ที่ใช้ในการพัฒนา
 - โปรแกรม Visual Studio Code ใช้สำหรับพัฒนาโปรแกรม
 - โปรแกรมออกแบบเว็บไซต์ (Figma) ใช้สำหรับออกแบบ UI
3. ภาษาที่ใช้ในการพัฒนา
 - HTML ใช้สำหรับเขียนโครงสร้างหน้าเว็บ
 - CSS ใช้สำหรับตกแต่งหน้าเว็บ
 - JavaScript ใช้สำหรับทำงานบนหน้าเว็บเซิร์ฟเวอร์
 - Node.js ใช้สำหรับเป็นเว็บเซิร์ฟเวอร์
 - Python ใช้สำหรับการคำนวณ
 - PHP คำนวณต้นทุนสินค้า
 - Laravel ใช้เป็น Framework
 - SQL (MySQL หรือ MariaDB) ใช้สำหรับจัดเก็บข้อมูล

1.4 วิธีการศึกษา

1.4.1 ศึกษาข้อมูลพื้นฐานเกี่ยวกับระบบบริหารจัดการคลังสินค้า (Warehouse Management System: WMS) และความต้องการของธุรกิจขนาดกลางและขนาดย่อม (SMEs) เพื่อกำหนด คุณสมบัติที่เหมาะสม

1.4.2 วิเคราะห์ปัญหาและข้อจำกัดจากระบบที่ใช้งานอยู่ในปัจจุบันของกลุ่มเป้าหมาย เพื่อหา แนวทางในการออกแบบระบบใหม่ให้ตอบโจทย์มากยิ่งขึ้น

1.4.3 ออกแบบระบบ โดยกำหนดโครงสร้างฐานข้อมูล (Database Design) และ ออกแบบหน้าตาเว็บไซต์ (User Interface) ให้ใช้งานง่ายและรองรับอุปกรณ์หลากหลาย

1.4.4 พัฒนาระบบต้นแบบในรูปแบบเว็บแอปพลิเคชัน โดยใช้ Laravel Framework ร่วมกับ HTML, CSS, JavaScript และฐานข้อมูล MySQL พร้อมฟังก์ชันสแกนบาร์โค้ด ผ่านกล้องมือถือ ด้วยไลบรารี JavaScript

1.4.5 ทดสอบระบบกับผู้ใช้งานกลุ่มเป้าหมาย เช่น เจ้าของร้านหรือพนักงานคลังสินค้า เพื่อ ประเมินความสะดวก ความแม่นยำ และประสิทธิภาพ แล้วนำข้อเสนอแนะมา ปรับปรุงระบบ

1.4.6 จัดทำเอกสารรายงาน และเตรียมนำเสนอผลงานโครงงานฉบับสมบูรณ์

1.5 ขอบเขตของโครงการ

1.5.1 ลักษณะของระบบ

- ระบบได้รับการพัฒนาในรูปแบบ เว็บแอปพลิเคชัน (Web Application) ซึ่ง สามารถใช้งานผ่านเว็บเบราว์เซอร์ได้จากทุกอุปกรณ์ ไม่ว่าจะเป็น คอมพิวเตอร์ แท็บเล็ต หรือสมาร์ทโฟน
- ระบบสามารถทำงานในรูปแบบ ออนไลน์ (Online) โดยมีการเชื่อมต่อกับ ฐานข้อมูลแบบเรียลไทม์ เพื่อให้สามารถตรวจสอบข้อมูลได้ทันที

1.5.2 กลุ่มผู้ใช้งานของระบบ

ระบบรองรับผู้ใช้งาน 2 ระดับ ได้แก่:

1. ผู้ดูแลระบบ (Admin): มีสิทธิ์ในการจัดการข้อมูลสินค้า บัญชีผู้ใช้งาน และสามารถออกรายงานวิเคราะห์ข้อมูลต่าง ๆ ได้
2. พนักงานทั่วไป (Employees): สามารถบันทึกข้อมูลการนำสินค้าเข้า-ออกคลัง และตรวจสอบข้อมูลสินค้าตามสิทธิ์ที่ได้รับ

1.5.3 ฟังก์ชันการทำงานหลักของระบบ

- การบันทึกข้อมูลการเข้า-ออกของสินค้าในคลัง
- การแสดงข้อมูลสินค้าคงเหลือแบบเรียลไทม์
- ระบบค้นหาและกรองข้อมูลสินค้า
- ระบบแจ้งเตือนอัตโนมัติเมื่อจำนวนสินค้าต่ำกว่าระดับขั้นต่ำ
- การออกรายงานในรูปแบบ PDF หรือ Excel เช่น รายงานสินค้าคงเหลือ รายงานการเคลื่อนไหวสินค้า ฯลฯ

1.5.4 การคำนวณต้นทุนสินค้า

- รองรับการคำนวณต้นทุนสินค้าโดยใช้วิธีทางบัญชี เช่น FIFO (First In First Out) และ Average Cost (ต้นทุนถัวเฉลี่ย)
- สามารถออกรายงานต้นทุนขาย (COGS) และสินค้าคงเหลือในแต่ละงวดได้อย่างแม่นยำ
- รองรับการวิเคราะห์ปริมาณการสั่งซื้อที่เหมาะสม (Economic Order Quantity: EOQ) เพื่อช่วยลดต้นทุนรวมของการจัดเก็บและการสั่งซื้อ โดยคำนึงถึงปัจจัย เช่น ต้นทุนการสั่งซื้อ ต้นทุนการเก็บรักษา และความต้องการใช้สินค้า
- ช่วยให้ธุรกิจสามารถตัดสินใจสั่งซื้อสินค้าได้ในปริมาณที่ประหยัดที่สุด และลดโอกาสการขาดแคลนหรือคงเหลือเกินความจำเป็น

1.5.5 ความสามารถในการขยายระบบในอนาคต

- ระบบถูกออกแบบให้รองรับการเชื่อมต่อกับ ระบบขายหน้าร้าน (POS) เพื่อรวมข้อมูลการขายและคลังสินค้า
- สามารถใช้งานร่วมกับ เครื่องสแกนบาร์โค้ด สำหรับการนำเข้าข้อมูลสินค้าอย่างรวดเร็ว
- รองรับการพัฒนา Dashboard แสดงผลแบบเรียลไทม์ เพื่อการวิเคราะห์ข้อมูลในภาพรวมของระบบคลังสินค้า

1.6 ระยะเวลาดำเนินงาน

ระยะเวลาดำเนินงาน ระหว่างวันที่ 9 มิถุนายน – 8 ตุลาคม พ.ศ.2568
โดยรวมมีรายละเอียดกิจกรรม ดังนี้

ปรับปรุง UI/UX																
8. พัฒนา ฟีเจอร์ รายงานและ ระบบแจ้ง เตือน																
9. จัดทำ เอกสาร ประกอบการ ใช้งานและ คู่มือผู้ใช้																
10. สรุปผลและ เตรียมการ นำเสนอ โครงการ																

1.7 ผลที่คาดว่าจะได้รับ

- 1.7.1 ช่วยให้ธุรกิจสามารถบริหารจัดการคลังสินค้าได้อย่างมีประสิทธิภาพ ลดความผิดพลาดที่เกิดจากการทำงานแบบแมนนวล
- 1.7.2 เพิ่มความแม่นยำในการควบคุมสินค้าคงคลัง โดยสามารถตรวจสอบข้อมูลได้แบบเรียลไทม์ ลดความเสี่ยงจากการขาดหรือเกินของสินค้าโดยไม่รู้ตัว
- 1.7.3 ผู้ดูแลระบบสามารถตรวจสอบข้อมูลย้อนหลัง รวมถึงออกรายงานได้อย่างสะดวก รวดเร็ว และอยู่ในรูปแบบที่เข้าใจง่าย
- 1.7.4 ลดระยะเวลาในการปฏิบัติงานของพนักงาน พร้อมทั้งสนับสนุนการตัดสินใจเชิงธุรกิจด้วยข้อมูลที่เป็นระบบและน่าเชื่อถือ
- 1.7.5 ระบบถูกออกแบบให้มีความยืดหยุ่น รองรับการขยายเพิ่มเติมในอนาคต เช่น การเชื่อมต่อกับระบบขายหน้าร้าน (POS) หรือระบบบัญชีต้นทุน
- 1.7.6 รองรับการใช้งานผ่านเว็บเบราว์เซอร์บนคอมพิวเตอร์ เพื่อความสะดวกและความถูกต้องในการจัดการคลังสินค้า

บทที่ 2

การศึกษาค้นคว้าและงานที่เกี่ยวข้อง

ในการพัฒนาระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง คณะผู้จัดทำได้ศึกษาแนวคิด ทฤษฎี และเทคโนโลยีที่เกี่ยวข้อง เพื่อเป็นรากฐานในการออกแบบและพัฒนาระบบให้ ตอบสนองต่อวัตถุประสงค์ที่ตั้งไว้ได้อย่างมีประสิทธิภาพ โดยมีรายละเอียดดังนี้

2.1 ระบบสินค้าคงคลังและความสำคัญ

2.2 แนวคิดหลักของแบบจำลอง EOQ (Economic Order Quantity)

2.3 การจัดการสินค้าคงคลังแบบ FIFO (First-In, First-Out)

2.4 ต้นทุนขาย (Cost of Goods Sold: COGS)

2.5 ต้นทุนถัวเฉลี่ย (Average Cost)

2.1 ระบบสินค้าคงคลัง (Inventory System)

ระบบสินค้าคงคลังหมายถึง กระบวนการ นโยบาย เครื่องมือ และระบบสารสนเทศที่ องค์กรใช้ในการติดตาม ควบคุม และบริหารการไหลเวียนของสินค้าและวัสดุ ตั้งแต่จัดหาและ รับเข้า เก็บรักษา โยกย้าย เบิกใช้หรือจำหน่าย จนถึงจุดขายหรือการส่งมอบสินค้า เป้าหมาย หลักของระบบนี้ คือการมีสินค้าในปริมาณที่เหมาะสม ณ สถานที่และเวลาที่เหมาะสม เพื่อให้ การดำเนินงานของห่วงโซ่อุปทานเป็นไปอย่างมีประสิทธิภาพ พร้อมข้อมูลที่เชื่อถือได้สำหรับ การตัดสินใจทั้งในระดับปฏิบัติการและยุทธศาสตร์ ระบบสินค้าคงคลังสมัยใหม่ยังเน้นการ ควบคุมสต็อกที่เคลื่อนย้ายเข้าออกคลังหรือข้ามสถานที่จัดเก็บหลายแห่งตลอดซัพพลายเชน เพื่อให้สินค้าถูกต้องและพร้อมใช้งานในเวลาที่ต้องการ

2.1.1 วัตถุประสงค์ระบบสินค้าคงคลัง (Inventory System)

1. ด้านระดับบริการและความพร้อมจำหน่าย (Service Level & Availability)

มุ่งเน้นการบรรลุระดับการให้บริการลูกค้าที่น่าพึงพอใจ เช่น ลดโอกาสการขาด สต็อก (Stockout) หรือการสั่งสินค้าซ้ำซ้อน (Backorder) รวมทั้งคงความพร้อม ในการจำหน่ายสินค้าอย่างต่อเนื่อง เพื่อสร้างสมดุลระหว่างคุณภาพการบริการ

กับต้นทุนการเก็บรักษาสต็อก ซึ่งเป็นหัวใจสำคัญของการบริหารสินค้าคงคลังที่มีประสิทธิภาพ

2. ด้านต้นทุนและเงินทุนหมุนเวียน (Cost & Working Capital)

ช่วยลดต้นทุนรวมหรือค่าใช้จ่ายที่เกิดจากการมีสินค้าคงคลัง เช่น ต้นทุนการเก็บรักษา (รวมถึงที่เก็บ ดูแล ประกัน และการเสื่อมสภาพของสินค้า) ต้นทุนเงินจมในสต็อก ต้นทุนการสั่งซื้อหรือนำเข้าสินค้า และต้นทุนจากการขาดสต็อก (loss of sales, ค่าปรับ หรือผลกระทบต่อภาพลักษณ์) อีกทั้งยังช่วยเพิ่มประสิทธิภาพในการใช้เงินทุนหมุนเวียน ด้วยการลดปริมาณสินค้าคงคลังเกินจำเป็น

3. ด้านความถูกต้องและประสิทธิภาพปฏิบัติการ (Accuracy & Operational Excellence)

เน้นให้ข้อมูลสินค้าคงคลังมีความถูกต้อง ทันเวลา และตรวจสอบได้ในทุกกิจกรรม ตั้งแต่การรับเข้า (receiving) เก็บเข้าที่ (putaway) โยกย้าย (transfer) จนถึงการเบิกจ่ายหรือส่งมอบสินค้า รวมถึงการตรวจนับสินค้าเป็นรอบ (cycle counting) เพื่อช่วยลดความคลาดเคลื่อนของข้อมูล ลดการสูญเสีย เช่น การสูญหายหรือการโจรกรรม และยกระดับประสิทธิภาพของคลังสินค้าทั้งหมด

4. ด้านการวางแผนและการตัดสินใจ (Planning & Decision Support)

สนับสนุนการคาดการณ์ความต้องการสินค้า (forecasting) เพื่อกำหนดระดับสินค้าคงคลังเป้าหมาย จุดสั่งซื้อใหม่ (reorder point) และสต็อกนิรภัย (safety stock) รวมถึงการติดตามตัวชี้วัดประสิทธิภาพ เช่น รอบหมุนเวียนของสต็อก (inventory turnover) เพื่อใช้พัฒนาผลงาน และเชื่อมโยงกับแผนการจัดหาหรือการผลิต

5. ด้านธรรมาภิบาลข้อมูลและการบูรณาการ (Data Governance & Integration)

ระบบสินค้าคงคลังต้องมีข้อมูลที่สอดคล้องเป็นมาตรฐาน และสามารถบูรณาการกับระบบธุรกิจที่เกี่ยวข้อง เช่น ระบบ POS, WMS, ERP และบัญชี ซึ่งช่วยลดงานซ้ำซ้อน ลดความผิดพลาดจากการป้อนข้อมูลซ้ำ และทำให้การไหลของสินค้าสอดคล้องกับการเงินอย่างโปร่งใส ทั้งนี้เป็นแนวปฏิบัติที่ได้รับการยอมรับในวงการซัพพลายเชนสากล

2.2 แนวคิดหลักของแบบจำลอง EOQ (Economic Order Quantity)

ความหมายของแบบจำลอง EOQ (Economic Order Quantity) แบบจำลอง EOQ (Economic Order Quantity) หรือ ปริมาณการสั่งซื้อที่ประหยัดที่สุด เป็นแนวคิดเชิงคณิตศาสตร์ที่ถูกพัฒนาขึ้นเพื่อใช้ในการจัดการสินค้าคงคลังอย่างมีประสิทธิภาพ โดยมีเป้าหมายสำคัญในการคำนวณหาปริมาณการสั่งซื้อที่เหมาะสมที่สุด (Optimal Order Quantity) เพื่อให้ ต้นทุนรวมในการบริหารสินค้าคงคลังอยู่ในระดับต่ำที่สุด

หลักการของแบบจำลอง EOQ มุ่งเน้นการสร้างสมดุลระหว่าง ต้นทุนการสั่งซื้อ (Ordering Cost) และ ต้นทุนการเก็บรักษา (Holding Cost) ซึ่งมีความสัมพันธ์ในลักษณะสวนทางกัน กล่าวคือ

- หากสั่งซื้อสินค้าในปริมาณน้อยแต่บ่อยครั้ง จะทำให้ต้นทุนการสั่งซื้อสูงขึ้น
- ในทางกลับกัน หากสั่งซื้อสินค้าในปริมาณมากเพื่อลดความถี่ในการสั่งซื้อ จะส่งผลให้ต้นทุนการเก็บรักษาสินค้าเพิ่มขึ้น

ดังนั้น แบบจำลอง EOQ จึงเป็นเครื่องมือในการวิเคราะห์เชิงปริมาณที่ช่วยให้องค์กรหรือธุรกิจสามารถกำหนดปริมาณการสั่งซื้อที่เหมาะสมที่สุดได้ โดยใช้ข้อมูลสำคัญ เช่น ความต้องการใช้สินค้าประจำช่วงเวลา (Demand), ต้นทุนการสั่งซื้อในแต่ละครั้ง (Ordering Cost per Order) และ ต้นทุนการเก็บรักษาต่อหน่วยสินค้า (Holding Cost per Unit) มาเป็นปัจจัยในการคำนวณ ผลลัพธ์จากแบบจำลอง EOQ จะช่วยให้การบริหารสินค้าคงคลังมีประสิทธิภาพมากขึ้น ลดความเสี่ยงจากการสต็อกสินค้าไม่จำเป็น และยังสามารถรักษาระดับสินค้าคงคลังให้เพียงพอต่อการตอบสนองความต้องการของตลาดได้อย่างเหมาะสมและต่อเนื่อง

2.2.1 วัตถุประสงค์ของแบบจำลอง EOQ (Economic Order Quantity)

แบบจำลอง EOQ มีวัตถุประสงค์หลักในการช่วยให้องค์กรหรือธุรกิจสามารถบริหารสินค้าคงคลังได้อย่างมีประสิทธิภาพ โดยมุ่งเน้นไปที่การลดต้นทุนและเพิ่มความสามารถในการวางแผนการสั่งซื้อสินค้า วัตถุประสงค์สำคัญประกอบด้วย

1. เพื่อลดต้นทุนรวมของการจัดการสินค้าคงคลัง โดยเฉพาะการหาสมดุลระหว่าง ต้นทุนการสั่งซื้อและต้นทุนการเก็บรักษา เพื่อไม่ให้ต้นทุนส่วนใดส่วนหนึ่งสูงจนเกินไป

2. เพื่อกำหนดปริมาณการสั่งซื้อที่เหมาะสมที่สุด (Optimal Order Quantity) ทำให้ธุรกิจสามารถวางแผนการสั่งซื้อสินค้าได้ในปริมาณที่พอดี ไม่มากหรือน้อยจนเกินไป
3. เพื่อป้องกันปัญหาการขาดแคลนสินค้า (Stockout) ช่วยให้สินค้าคงคลังเพียงพอต่อความต้องการของลูกค้าและการดำเนินธุรกิจอย่างต่อเนื่อง
4. เพื่อลดปัญหาสินค้าคงคลังส่วนเกิน (Overstock) หลีกเลี่ยงการเก็บสินค้ามากเกินไป ซึ่งอาจทำให้เกิดต้นทุนจม ต้นทุนการเก็บรักษาที่สูงขึ้น หรือความเสี่ยงจากการเสื่อมสภาพของสินค้า
5. เพื่อเพิ่มประสิทธิภาพในการวางแผนและบริหารจัดการ ทำให้การตัดสินใจเกี่ยวกับการสั่งซื้อสินค้าเป็นระบบมากขึ้น ช่วยให้องค์กรสามารถใช้ทรัพยากรได้อย่างคุ้มค่าและเกิดประโยชน์สูงสุด

2.2.2 ต้นทุนที่เกี่ยวข้อง

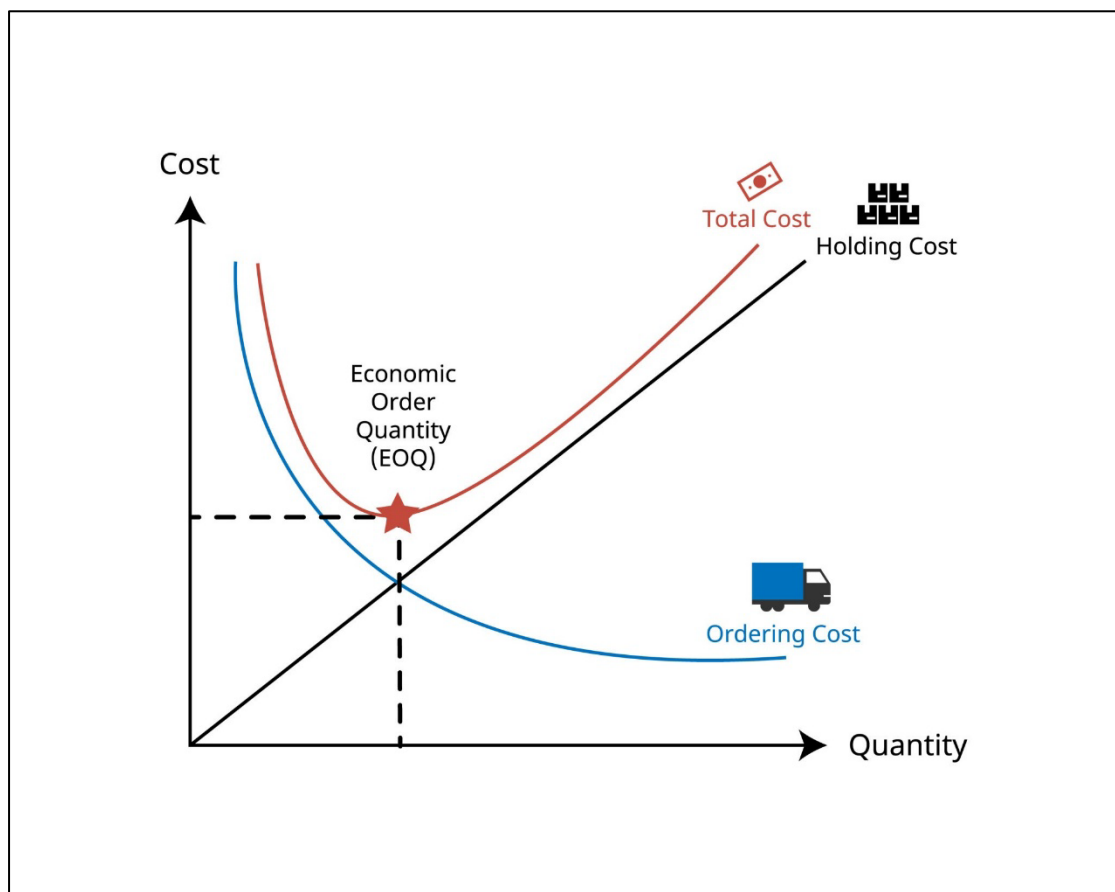
แบบจำลอง EOQ พิจารณาต้นทุนหลักสองส่วนที่เกี่ยวข้องกันและแปรผกผันต่อกัน

- ต้นทุนในการสั่งซื้อ (Ordering Cost หรือ S): เป็นค่าใช้จ่ายที่เกิดขึ้นในแต่ละครั้งที่สั่งซื้อสินค้า ไม่ว่าปริมาณสินค้าที่สั่งจะมากหรือน้อยก็ตาม เช่น ค่าจัดทำเอกสารการสั่งซื้อ, ค่าโทรศัพท์, ค่าขนส่ง, และค่าแรงงานในการจัดซื้อ. ต้นทุนนี้จะลดลงเมื่อมีการสั่งซื้อในปริมาณที่มากขึ้น (สั่งน้อยครั้งลง)
- ต้นทุนในการเก็บรักษา (Holding Cost หรือ H): เป็นค่าใช้จ่ายที่เกิดขึ้นจากการมีสินค้าคงคลังในคลังสินค้า ซึ่งมักจะคิดเป็นอัตราต่อหน่วยต่อปี เช่น ค่าเช่าพื้นที่, ค่าไฟฟ้า, ค่าประกันภัย, ค่าเสื่อมราคาของสินค้า หรือต้นทุนค่าเสียโอกาสของเงินทุนที่จมอยู่. ต้นทุนนี้จะสูงขึ้นเมื่อมีการสั่งซื้อในปริมาณที่มากขึ้น (เก็บสต็อกจำนวนมาก)

2.2.3 การทำงานของแบบจำลอง

แบบจำลอง EOQ ทำงานโดยการหาจุดที่ต้นทุนในการสั่งซื้อรวมและต้นทุนในการเก็บรักษารวมมีค่าเท่ากัน ซึ่งเป็นจุดที่ทำให้ต้นทุนรวมทั้งหมดต่ำที่สุด กราฟนี้แสดงให้เห็นความสัมพันธ์ระหว่างปริมาณการสั่งซื้อ (Q) กับต้นทุนที่เกี่ยวข้อง

- เส้นโค้งสีน้ำเงิน (Holding Cost): แสดงให้เห็นว่าเมื่อปริมาณการสั่งซื้อ (Q) เพิ่มขึ้น ต้นทุนในการเก็บรักษาก็จะสูงขึ้นตามไปด้วย
- เส้นโค้งสีแดง (Ordering Cost): แสดงให้เห็นว่าเมื่อปริมาณการสั่งซื้อ (Q) เพิ่มขึ้น ต้นทุนในการสั่งซื้อรวมก็จะลดลง
- เส้นโค้งสีเขียว (Total Cost): แสดงต้นทุนรวมซึ่งเป็นผลรวมของทั้งสองเส้น จุดที่เส้นนี้มีค่าต่ำสุดคือจุด EOQ ซึ่งเป็นปริมาณการสั่งซื้อที่ประหยัดที่สุด



รูปที่ 2.2.3 การทำงานของแบบจำลอง EOQ

2.2.4 สูตรคำนวณ

1. สูตรที่ใช้ในการคำนวณหาปริมาณการสั่งซื้อที่ประหยัดที่สุด (EOQ)

- ความหมาย ใช้คำนวณหาปริมาณการสั่งซื้อต่อครั้งที่เหมาะสมที่สุด เพื่อให้ต้นทุนรวมในการสั่งซื้อและเก็บรักษาต่ำที่สุด
- สูตร:

$$EOQ = \sqrt{\frac{2DS}{H}}$$

โดยที่

D = ปริมาณความต้องการสินค้าทั้งหมดต่อปี (Annual Demand)

S = ต้นทุนในการสั่งซื้อแต่ละครั้ง (Ordering Cost per order)

H = ต้นทุนในการเก็บรักษาสินค้าต่อหน่วยต่อปี (Holding Cost per unit per year)

2. สูตรจุดสั่งซื้อซ้ำ (Reorder Point: ROP)

- ความหมาย ใช้คำนวณหาจุดที่ควรสั่งซื้อสินค้าใหม่ เพื่อป้องกันการขาดสต็อกในระหว่างที่กำลังรอสินค้า
- สูตร:

$$ROP = d \times LT$$

โดยที่

d = ความต้องการสินค้าเฉลี่ยต่อวัน

LT = ระยะเวลารอสินค้า (จำนวนวัน)

3. สูตรต้นทุนรวมที่ต่ำที่สุด (Total Cost: TC)

- ความหมาย ใช้คำนวณต้นทุนรวมทั้งหมดที่เกิดขึ้นจากการบริหารสินค้าคงคลัง เมื่อสั่งซื้อสินค้าในปริมาณที่เท่ากับ EOQ
- สูตร:

$$TC = (D / EOQ)S + (EOQ / 2)H$$

โดยที่

D = ปริมาณความต้องการใช้สินค้าทั้งหมดต่อปี

EOQ = ปริมาณการสั่งซื้อที่ประหยัดที่สุด

S = ต้นทุนในการสั่งซื้อแต่ละครั้ง

H = ต้นทุนในการเก็บรักษาสินค้าต่อหน่วยต่อปี

2.2.5 สมมติฐานของแบบจำลอง EOQ (Economic Order Quantity)

แบบจำลอง EOQ แม้จะเป็นเครื่องมือที่มีประสิทธิภาพในการคำนวณหาปริมาณการสั่งซื้อที่ประหยัดที่สุด แต่ก็ถูกพัฒนาขึ้นภายใต้สมมติฐานที่ค่อนข้างเข้มงวดหลายประการ สมมติฐานเหล่านี้เป็นเงื่อนไขที่ใช้เพื่อความสะดวกในการคำนวณและการวิเคราะห์ อย่างไรก็ตาม ในการประยุกต์ใช้จริง อาจพบข้อจำกัดที่แตกต่างไปจากทฤษฎี ดังนี้

1. ความต้องการคงที่ (Constant Demand)

- สมมติให้ความต้องการใช้สินค้าในแต่ละช่วงเวลามีค่า คงที่และทราบล่วงหน้า ตลอดทั้งปี โดยไม่มีความผันผวนตามฤดูกาลหรือปัจจัยภายนอกอื่น ๆ
- ข้อสังเกต: ในความเป็นจริง ความต้องการของสินค้ามักมีความไม่แน่นอนและผันผวน จึงมีการพัฒนาแบบจำลองที่ซับซ้อนขึ้นเพื่อรองรับความต้องการที่เปลี่ยนแปลงไปตามเวลา

2. ระยะเวลาออสินค้าคงที่ (Constant Lead Time)

- สมมติให้ ระยะเวลาคอยสินค้า (Lead Time) หลังจากทำการสั่งซื้อมีค่า คงที่และทราบแน่นอน
- ข้อสังเกต: ในทางปฏิบัติ ระยะเวลาออสินค้าอาจเปลี่ยนแปลงได้จากหลายปัจจัย เช่น กระบวนการผลิต การขนส่ง หรือความไม่แน่นอนจากผู้จัดจำหน่าย

3. ราคาต่อหน่วยคงที่ (Constant Unit Price)

- สมมติให้ราคาสินค้าต่อหน่วยมีค่า คงที่เสมอ ไม่ว่าจะสั่งซื้อในปริมาณมากหรือน้อย และ ไม่มีส่วนลดปริมาณ (Quantity Discount)

- ข้อสังเกต: สมมติฐานนี้ไม่สอดคล้องกับแนวทางการค้าจริง ซึ่งมักมีนโยบายส่วนลดสำหรับการสั่งซื้อในปริมาณมาก

4. ต้นทุนคงที่ (Constant Costs)

- สมมติให้ ต้นทุนในการสั่งซื้อ (Ordering Cost) และ ต้นทุนในการเก็บรักษาต่อหน่วยต่อปี (Holding Cost) มีค่า คงที่ ไม่แปรผันตามปริมาณการสั่งซื้อหรือปริมาณสินค้าที่เก็บรักษา
- ข้อสังเกต: ในความเป็นจริง ต้นทุนเหล่านี้สามารถเปลี่ยนแปลงได้ เช่น ค่าเช่าพื้นที่คลังสินค้าที่อาจเพิ่มขึ้นตามปริมาณการจัดเก็บ

5. ไม่มีสินค้าขาดสต็อก (No Stockouts)

- สมมติให้สินค้าที่สั่งซื้อจะมาถึงตามกำหนดเวลาเสมอ และจะไม่เกิดปัญหาการขาดแคลนสินค้าในระบบ
- ข้อสังเกต: สมมติฐานนี้ไม่สะท้อนความเป็นจริง เพราะในทางธุรกิจ อาจเกิดภาวะสินค้าขาดสต็อกได้ ซึ่งจะนำไปสู่การเสียโอกาสในการขายและการสูญเสียความพึงพอใจของลูกค้า

2.3 การจัดการสินค้าคงคลังแบบ FIFO (First-In, First-Out)

ความหมายของระบบการจัดการแบบ FIFO (First-In, First-Out) คือ หลักการจัดการสินค้าคงคลังที่กำหนดให้ “สินค้าที่เข้ามาก่อน จะถูกขายหรือเบิกออกไปก่อน” ซึ่งช่วยลดความเสี่ยงจากการที่สินค้าค้างสต็อกจนเสื่อมคุณภาพหรือหมดอายุ

2.3.1 ลักษณะการทำงาน

1. เมื่อมีการรับสินค้าใหม่เข้าสู่คลัง ระบบจะจัดเก็บข้อมูลวันเวลาที่เข้ามา
2. เมื่อมีการเบิกหรือนำสินค้าออกไป ระบบจะเลือกสินค้าที่เข้ามา ก่อนตามลำดับ
3. สินค้าที่เข้ามาภายหลังจะถูกเก็บไว้รอจนกว่าสินค้านั้นเก่าจะ ถูกเบิกออกหมด

2.3.2 ข้อดีของ FIFO (First-In, First-Out)

- ลดความเสี่ยงสินค้าหมดอายุหรือเสื่อมสภาพ (เหมาะกับสินค้าอาหาร ยา หรือเครื่องสำอาง)
- สะท้อนต้นทุนสินค้าที่ใกล้เคียงกับราคาตลาดปัจจุบัน
- เป็นวิธีที่เข้าใจง่ายและเหมาะกับธุรกิจขนาดกลางและขนาดเล็ก

2.3.3 ข้อจำกัดของ FIFO (First-In, First-Out)

- ต้องมีระบบติดตามที่แม่นยำ เช่น Barcode, QR Code หรือระบบ ERP
- อาจไม่เหมาะกับสินค้าที่ราคามีความผันผวนสูง เพราะต้นทุนการขายจะอิงจากล็อตเก่าเสมอ

2.3.4 เทคโนโลยีที่สนับสนุนการทำ FIFO

- Barcode/QR Code Scanner: สำหรับระบุรหัสสินค้าและวันที่รับเข้า
- ฐานข้อมูล (Database Management System): ใช้จัดเก็บข้อมูลสินค้า วันที่รับเข้า และปริมาณคงเหลือ
- ระบบ ERP/WMS (Warehouse Management System): ซอฟต์แวร์สำหรับจัดการคลังสินค้าแบบอัตโนมัติ
- Dashboard และ Reporting Tools: แสดงรายงานสินค้าคงเหลือและแจ้งเตือนเมื่อสินค้าจะหมดอายุ

2.3.5 หลักการสำคัญของ FIFO (First-In, First-Out)

- การจัดการตามลำดับเวลา: สินค้าที่ถูกรับเข้ามาในคลังก่อนจะต้องถูกนำออกไปใช้หรือจำหน่ายก่อนเสมอ
- การลดความเสี่ยงของสินค้า: ช่วยลดความเสี่ยงที่สินค้าจะหมดอายุ เสื่อมสภาพ หรือล้าสมัย

- การรักษามูลค่า: สินค้าที่ขายออกไปก่อนมักเป็นสินค้าที่ซื้อมาในราคาเก่า ทำให้สินค้าคงคลังที่เหลืออยู่สะท้อนมูลค่าที่แท้จริงตามราคาปัจจุบัน
- การลดการสูญเสีย: ป้องกันการสูญเสียจากการที่สินค้าหมดอายุ ต้องทิ้ง หรือขายลดราคา ช่วยประหยัดต้นทุนได้มากขึ้น
- การเพิ่มความพึงพอใจของลูกค้า: ลูกค้าได้รับสินค้าที่สดใหม่และมีคุณภาพเสมอ ส่งผลต่อความน่าเชื่อถือและความภักดีต่อธุรกิจในระยะยาว



รูปที่ 1.3.5 แผนภาพการทำงานของระบบ FIFO

2.4 ต้นทุนขาย (Cost of Goods Sold: COGS)

ต้นทุนขาย (Cost of Goods Sold: COGS) หมายถึง มูลค่าต้นทุนของสินค้าที่องค์กรจำหน่ายออกไปในรอบระยะเวลาหนึ่ง ซึ่งนับเป็นค่าใช้จ่ายโดยตรงที่เกี่ยวข้องกับการจัดหาสินค้ามาจำหน่าย ไม่ว่าจะอยู่ในรูปแบบการซื้อสินค้าสำเร็จรูปจากซัพพลายเออร์หรือการผลิตขึ้นเอง ต้นทุนขายถือเป็นองค์ประกอบสำคัญในการคำนวณกำไรขั้นต้น (Gross Profit) และใช้ในการวิเคราะห์ประสิทธิภาพการดำเนินงานของธุรกิจ

สูตรการคำนวณต้นทุนขายพื้นฐาน คือ

$$COGS = \text{สินค้าคงเหลือต้นงวด} + \text{สินค้าที่ซื้อหรือผลิตเพิ่มระหว่างงวด} - \text{สินค้าคงเหลือปลายงวด}$$

โดยที่

- สินค้าคงเหลือต้นงวด คือ มูลค่าสินค้าที่มีอยู่ในคลัง ณ จุดเริ่มต้นของรอบบัญชี
- สินค้าที่ซื้อหรือผลิตระหว่างงวด คือ มูลค่าสินค้าที่นำเข้ามาเพิ่มในช่วงงวด
- สินค้าคงเหลือปลายงวด คือ มูลค่าสินค้าที่เหลืออยู่ในคลัง ณ สิ้นรอบบัญชี

2.5 ต้นทุนถัวเฉลี่ย (Average Cost)

ต้นทุนถัวเฉลี่ย (Average Cost Method) คือ วิธีการคำนวณมูลค่าสินค้าคงเหลือและต้นทุนขาย (Cost of Goods Sold: COGS) โดยการนำมูลค่าต้นทุนรวมของสินค้าที่มีอยู่ทั้งหมดในคลัง มาหารด้วยจำนวนหน่วยสินค้าทั้งหมด เพื่อหาต้นทุนต่อหน่วยเฉลี่ย จากนั้นจึงนำต้นทุนต่อหน่วยดังกล่าวไปใช้ในการคำนวณสินค้าคงเหลือปลายงวดและต้นทุนขาย วิธีนี้ทำให้ต้นทุนของสินค้าที่ขายออกไปและสินค้าที่เหลืออยู่ในคลังมีมูลค่าต่อหน่วยเท่ากัน

สูตรการคำนวณต้นทุนถัวเฉลี่ยต่อหน่วย คือ

$$\text{ต้นทุนต่อหน่วยเฉลี่ย} = \frac{\text{มูลค่าสินค้าคงเหลือต้นงวด} + \text{มูลค่าสินค้าที่ซื้อเพิ่มระหว่างงวด}}{\text{จำนวนสินค้าคงเหลือต้นงวด} + \text{จำนวนหน่วยที่ซื้อเพิ่มระหว่างงวด}}$$

เมื่อได้ต้นทุนต่อหน่วยเฉลี่ยแล้ว สามารถนำไปคำนวณเป็นมูลค่าต้นทุนขาย (COGS) และสินค้าคงเหลือปลายงวด ดังนี้

$$COGS = \text{จำนวนสินค้าที่ขาย} \times \text{ต้นทุนต่อหน่วยเฉลี่ย}$$

$$\text{สินค้าคงเหลือปลายงวด} = \text{จำนวนสินค้าที่เหลืออยู่} \times \text{ต้นทุนต่อหน่วยเฉลี่ย}$$

บทที่ 3

เทคโนโลยีที่ใช้ในการศึกษา

ในการพัฒนาระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง (Inventory Management System for SMEs) คณะผู้จัดทำได้เลือกใช้เทคโนโลยีและเครื่องมือที่เหมาะสม เพื่อให้ระบบสามารถทำงานได้อย่างมีประสิทธิภาพ ครอบคลุมตั้งแต่ส่วนติดต่อผู้ใช้ การประมวลผลหลังบ้าน การจัดเก็บข้อมูล ความปลอดภัย การเชื่อมต่อเครือข่าย รวมถึงการทดสอบระบบ โดยรายละเอียดดังนี้

3.1 เทคโนโลยีด้านฮาร์ดแวร์ (Hardware Technology)

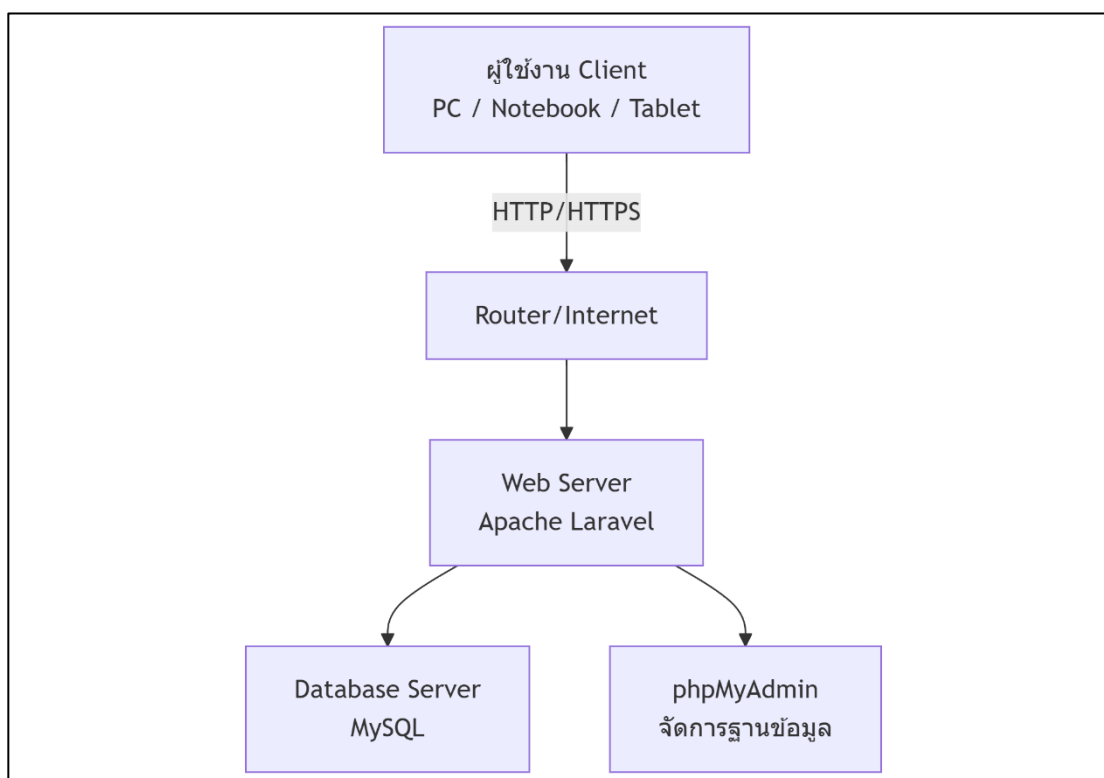
ฮาร์ดแวร์เป็นส่วนสำคัญในการสนับสนุนการทำงานของซอฟต์แวร์ โดยประกอบไปด้วยอุปกรณ์ต่าง ๆ เช่น คอมพิวเตอร์ เครื่องเซิร์ฟเวอร์ อุปกรณ์เครือข่าย และเครื่องสแกนบาร์โค้ด ซึ่งมีบทบาทในการรองรับการประมวลผล การจัดเก็บ และการสื่อสารข้อมูล เพื่อให้ระบบคลังสินค้าสามารถทำงานได้อย่างต่อเนื่องและมีเสถียรภาพ โดยมีองค์ประกอบดังนี้

1. คอมพิวเตอร์/Notebook ที่ใช้พัฒนาและทดสอบระบบ จำนวน 2 เครื่อง
2. เครื่องสแกนบาร์โค้ด (Barcode Scanner) สำหรับบันทึกสินค้าเข้า-ออก 1 เครื่อง



รูปที่ 3.1 (1) เครื่องสแกนบาร์โค้ด

3. อุปกรณ์เครือข่าย (Router, Switch, Internet) สำหรับเชื่อมต่อระบบแสดงโครงสร้างการเชื่อมต่อเครือข่ายของระบบคลังสินค้า โดยผู้ใช้งานสามารถเข้าถึงระบบผ่านอินเทอร์เน็ตหรือเครือข่ายภายใน (LAN) โดยเชื่อมต่อผ่าน Router ไปยัง Web Server (Laravel) ซึ่งทำงานร่วมกับฐานข้อมูล MySQL และ phpMyAdmin

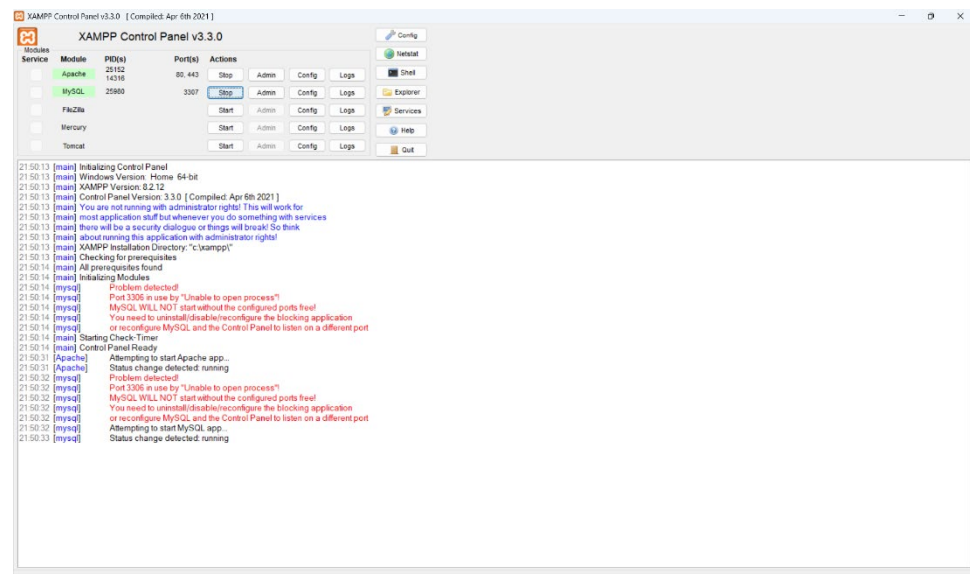


รูปที่ 3.1 (2) แสดงโครงสร้างการเชื่อมต่อเครือข่ายของระบบคลังสินค้า

3.2 เทคโนโลยีด้านซอฟต์แวร์ (Software Technology)

ซอฟต์แวร์เป็นหัวใจหลักที่ทำให้ระบบสามารถดำเนินการได้อย่างถูกต้องและมีประสิทธิภาพ ครอบคลุมตั้งแต่ส่วนติดต่อผู้ใช้ (Frontend) ที่ช่วยให้ใช้งานง่ายและสะดวก ส่วนประมวลผลและตรรกะธุรกิจ (Backend) ที่จัดการกับการทำงานเบื้องหลัง ระบบฐานข้อมูลที่เก็บข้อมูลอย่างเป็นระบบ ตลอดจนเครื่องมือที่ช่วยออกรายงานและนำเสนอข้อมูลในรูปแบบที่เข้าใจง่ายและเป็นประโยชน์ต่อการตัดสินใจ โดยมีองค์ประกอบดังนี้

3.2.1 ซอฟต์แวร์ที่ใช้ในการพัฒนาและทดสอบระบบคลังสินค้าบนเครื่องคอมพิวเตอร์ สำหรับติดตั้ง Backend และฐานข้อมูล ในการติดตั้งและทดสอบระบบ คณะผู้จัดทำได้เลือกใช้ XAMPP ซึ่งเป็นซอฟต์แวร์จำลอง Web Server ที่รวม Apache, MySQL, และ PHP ไว้ในแพ็คเกจเดียว เพื่อความสะดวกในการพัฒนาและทดสอบระบบคลังสินค้า โดย Apache ทำหน้าที่เป็น Web Server สำหรับรัน Laravel Framework ส่วน MySQL ใช้ในการจัดการฐานข้อมูล และ phpMyAdmin ใช้ในการตรวจสอบและจัดการข้อมูลผ่าน Web Interface



รูปที่ 3.2.1 (1) XAMPP Control Panel v3.3.0

3.2.2 ส่วนติดต่อผู้ใช้ (Frontend)

ส่วนติดต่อผู้ใช้ (Frontend) เป็นองค์ประกอบสำคัญที่ทำให้ผู้ใช้งานสามารถโต้ตอบกับระบบได้อย่างสะดวกและมีประสิทธิภาพ โดยโครงงานนี้เลือกใช้ React.js และ Inertia.js เป็นเทคโนโลยีหลักในการพัฒนา

React.js เป็นไลบรารีสำหรับสร้างส่วนติดต่อผู้ใช้ในรูปแบบ Single Page Application (SPA) ซึ่งช่วยให้การทำงานลื่นไหล ผู้ใช้ไม่จำเป็นต้องโหลดหน้าเว็บใหม่ทั้งหมดเมื่อเปลี่ยนไปยังหน้าต่าง ๆ ของระบบ แต่เพียงอัปเดตเฉพาะส่วนที่เปลี่ยนแปลง ทำให้ผู้ใช้ได้รับประสบการณ์ที่รวดเร็วและทันสมัย อีกทั้ง React.js ยังสนับสนุนการพัฒนาแบบ Component-Based ที่สามารถนำส่วนประกอบของหน้าจอมาใช้ซ้ำได้ เช่น ปุ่ม ฟอรั่ม ตารางสินค้า และแถบเมนู ช่วยลดเวลาในการพัฒนาและง่ายต่อการบำรุงรักษาในอนาคต

Inertia.js ทำหน้าที่เป็นตัวกลางเชื่อมต่อระหว่าง React.js และ Laravel Backend โดยไม่จำเป็นต้องสร้าง REST API แยก ทำให้โครงสร้างระบบเรียบง่ายและลดความซับซ้อนในการพัฒนา Inertia.js จะส่งข้อมูลจากฝั่งผู้ใช้ไปยัง Laravel เพื่อประมวลผล และส่งผลลัพธ์กลับมาแสดงผลบน React.js แบบเรียลไทม์ ตัวอย่างการใช้งาน เช่น หน้าขายสินค้า (POS) ที่ผู้ใช้เลือกสินค้าและทำการขาย หรือหน้าคืนสินค้า ที่มีการอัปเดตสต็อกสินค้าโดยอัตโนมัติ

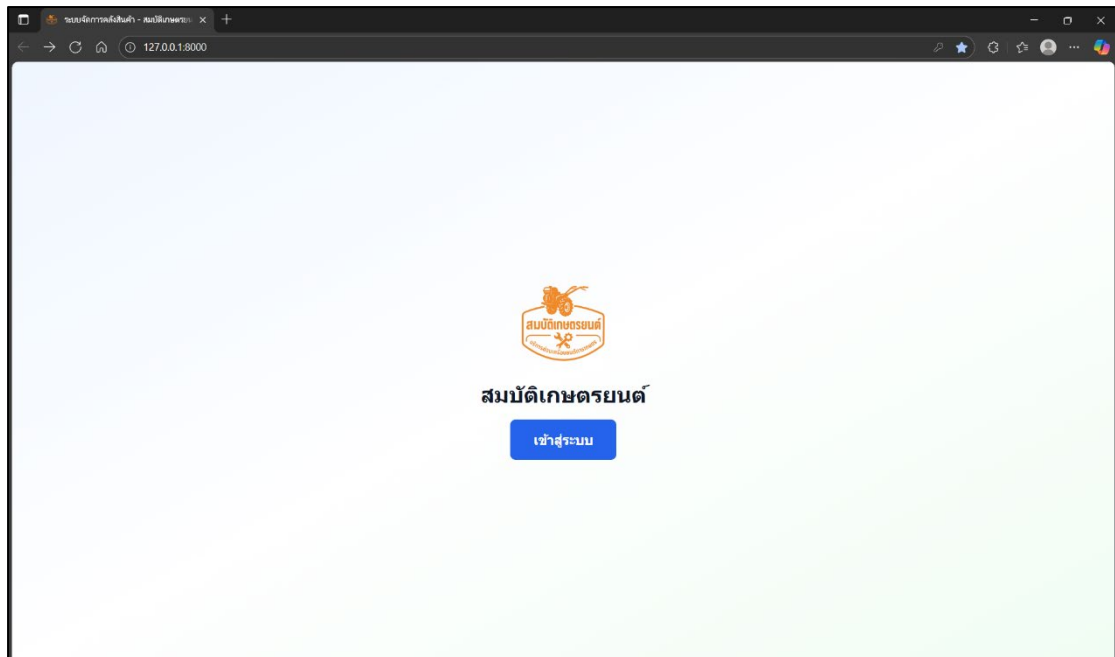
การใช้ React.js และ Inertia.js ร่วมกัน ทำให้ส่วนติดต่อผู้ใช้ของระบบ คลังสินค้ามีความทันสมัย ใช้งานง่าย และตอบสนองได้อย่างรวดเร็ว ช่วยให้พนักงาน และผู้ใช้งานระบบสามารถทำงานได้อย่างมีประสิทธิภาพ

3.2.2.1 React.js เป็นไลบรารีสำหรับการพัฒนา ส่วนติดต่อผู้ใช้ (User Interface – UI) ที่มีประสิทธิภาพสูงและทันสมัย โดยระบบคลังสินค้านี้เลือกใช้ React.js เพื่อพัฒนาในรูปแบบ Single Page Application (SPA) ซึ่งช่วยให้การใช้งานลื่นไหล ผู้ใช้ไม่จำเป็นต้องโหลดหน้าเว็บใหม่ทั้งหมดเมื่อมีการเปลี่ยนหน้า แต่เพียงอัปเดตเฉพาะส่วนที่เปลี่ยนแปลง ทำให้ประสบการณ์การใช้งานสะดวกและรวดเร็วขึ้น

React.js ยังสนับสนุนการพัฒนาแบบ Component-Based คือสามารถแบ่งหน้าจอออกเป็นส่วนย่อย ๆ (Component) เช่น แถบเมนู, ตารางสินค้า, ตะกร้าสินค้า ทำให้โค้ดมีความเป็นระเบียบ สามารถนำกลับมาใช้ซ้ำได้ และง่ายต่อการบำรุงรักษาในอนาคต ตัวอย่างการใช้งาน React.js ในระบบ ได้แก่

1. Welcome Page (หน้าต้อนรับเข้าสู่ระบบ)

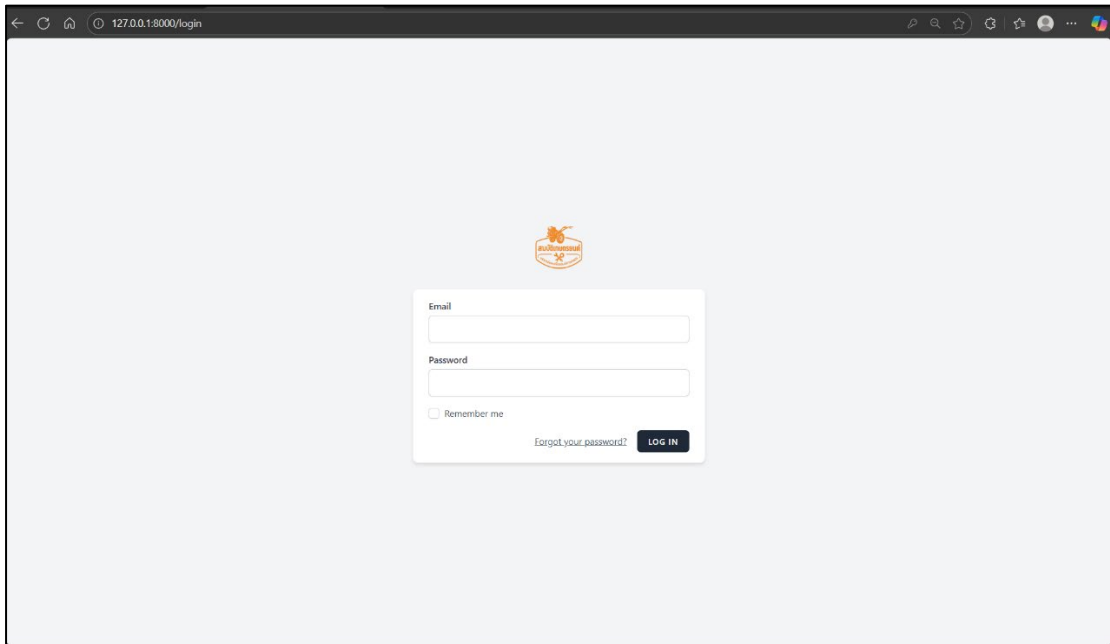
หน้าต้อนรับเป็นหน้าจอแรกที่ผู้ใช้งานเห็นเมื่อเข้าสู่เว็บไซต์ระบบคลังสินค้า ถูกพัฒนาด้วย React.js โดยมีการออกแบบให้เรียบง่ายและชัดเจน แสดงชื่อระบบและปุ่ม “เข้าสู่ระบบ” เพื่อให้ผู้ใช้สามารถเข้าสู่หน้าล็อกอินได้ทันที การออกแบบในลักษณะนี้ช่วยให้ผู้ใช้เข้าใจได้ง่าย ลดความซับซ้อน และสร้างความน่าเชื่อถือของระบบ



รูปที่ 3.2.2 (1) แสดงหน้าต้อนรับ (Welcome Page) ของระบบคลังสินค้า

2. Login Page (หน้าล็อกอินเข้าสู่ระบบ)

หน้าล็อกอินเป็นส่วนที่ผู้ใช้งานต้องกรอก อีเมลและรหัสผ่าน เพื่อเข้าสู่ระบบ โดยการพัฒนาด้วย React.js เช่นกัน เพื่อแสดงฟอร์มกรอกข้อมูลที่ทันสมัย รองรับการตรวจสอบความถูกต้องของข้อมูลเบื้องต้น (Validation) และส่งข้อมูลเข้าสู่ Backend ผ่าน Inertia.js และ Laravel เพื่อทำการตรวจสอบสิทธิ์ (Authentication) การออกแบบหน้านี้ถือเป็นส่วนสำคัญของระบบ เพราะเป็นจุดที่เชื่อมโยงการเข้าถึงระหว่างผู้ใช้งานทั่วไปกับระบบหลังบ้าน (Backend)



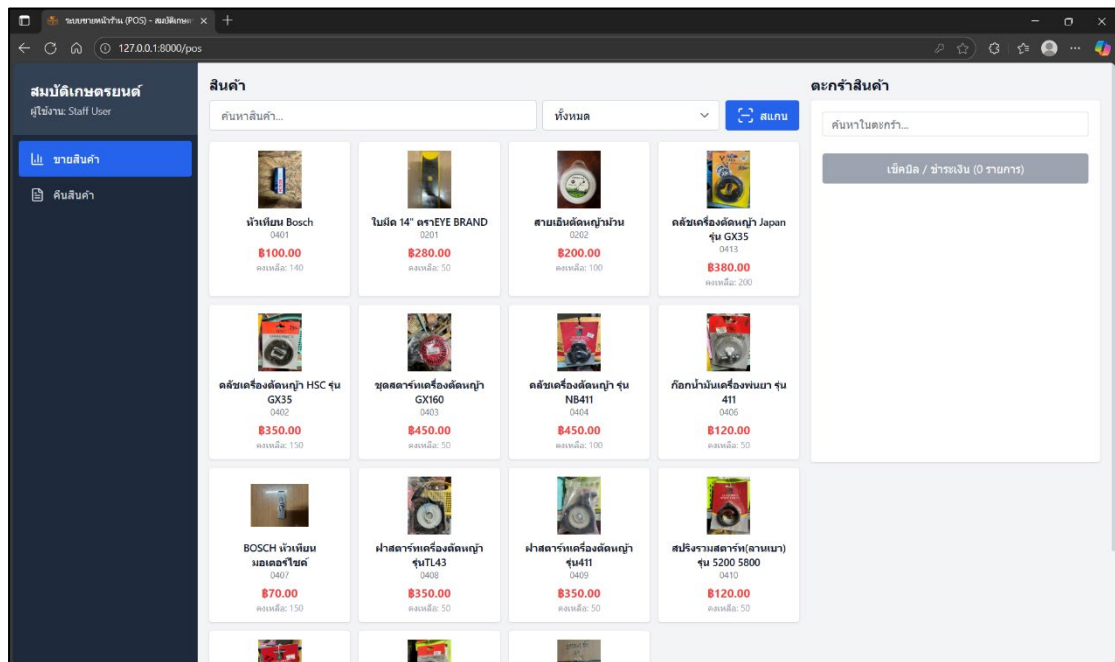
รูปที่ 3.2.2 (2) แสดงหน้าล็อกอิน (Login Page) ของระบบคลังสินค้า

3.2.2.2 Inertia.js ทำหน้าที่เป็นตัวกลางในการเชื่อมต่อระหว่าง React.js (Frontend) และ Laravel (Backend) โดยไม่จำเป็นต้องสร้าง REST API แยกสำหรับการสื่อสารข้อมูล ทำให้โครงสร้างระบบเรียบง่ายขึ้นและลดความซับซ้อนในการพัฒนา ผู้ใช้สามารถโต้ตอบกับข้อมูลได้แบบเรียลไทม์ เช่น การขายสินค้า (POS) การคืนสินค้า หรือการจัดการสต็อก โดย Inertia.js จะทำงานร่วมกับ Laravel เพื่อดึงข้อมูลจากฐานข้อมูล MySQL และส่งกลับไปยัง React.js เพื่อแสดงผลในรูปแบบที่ทันสมัยและใช้งานง่าย

ตัวอย่างการใช้งาน Inertia.js ในระบบ ได้แก่ Inertia.js ทำหน้าที่เป็นตัวกลางในการเชื่อมต่อระหว่าง React.js (Frontend) และ Laravel (Backend) โดยไม่จำเป็นต้องสร้าง REST API แยกสำหรับการสื่อสารข้อมูล ทำให้โครงสร้างระบบเรียบง่ายขึ้นและลดความซับซ้อนในการพัฒนา ผู้ใช้สามารถโต้ตอบกับข้อมูลได้แบบเรียลไทม์ เช่น การขายสินค้า (POS) การคืนสินค้า หรือการจัดการสต็อก โดย Inertia.js จะทำงานร่วมกับ Laravel เพื่อดึงข้อมูลจากฐานข้อมูล MySQL และส่งกลับไปยัง React.js เพื่อแสดงผลในรูปแบบที่ทันสมัยและใช้งานง่าย

1. หน้าขายสินค้า (Point of Sale – POS)

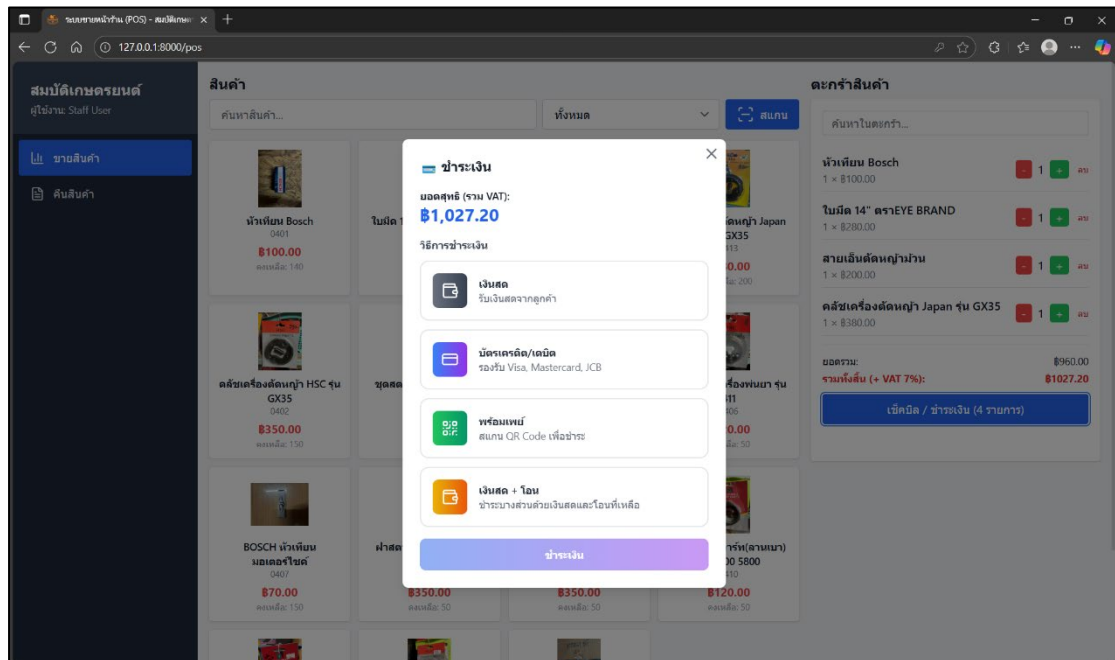
หน้าขายสินค้าเป็นฟังก์ชันหลักที่ช่วยให้พนักงานสามารถค้นหาและเลือกสินค้าที่ลูกค้าต้องการซื้อได้อย่างรวดเร็ว โดยแสดงข้อมูลสินค้า เช่น ชื่อสินค้า รหัสสินค้า ราคา และจำนวนคงเหลือ พร้อมทั้งมี ตะกร้าสินค้า ด้านขวา สำหรับรวมรายการที่เลือกไว้ก่อนการชำระเงิน หน้านี้ถูกพัฒนาด้วย React.js เพื่อแสดงผลที่ทันสมัยและตอบสนองอย่างรวดเร็ว และใช้ Inertia.js เชื่อมต่อกับ Laravel Backend เพื่อบันทึกการขายและอัปเดตสต็อกสินค้าแบบอัตโนมัติ



รูปที่ 3.2.2 (3) แสดงหน้าขายสินค้า (POS)

2. หน้าชำระเงิน (Checkout/Payment Page)

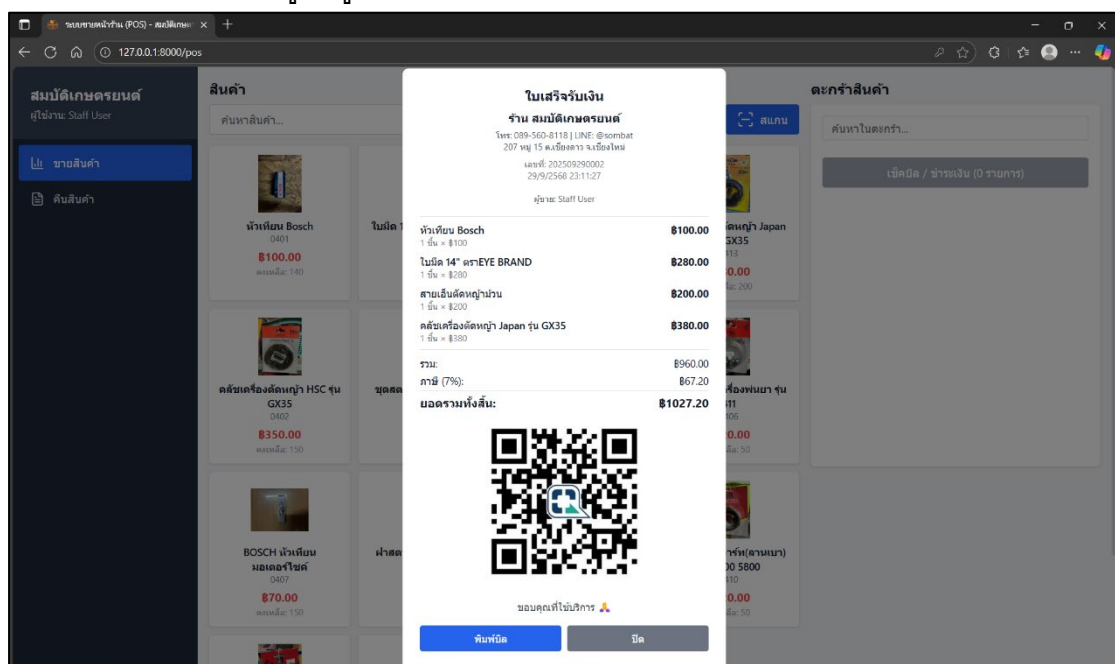
หน้าคืนสินค้าเป็นส่วนที่ออกแบบมาเพื่อรองรับการจัดการสินค้าที่ถูกส่งคืน โดยมีการแสดงยอดการคืนสินค้าในแต่ละวันและสรุปยอดรวมของเดือน ผู้ใช้งานสามารถค้นหาใบเสร็จย้อนหลังหรือเลือกสินค้าที่ต้องการคืนได้ ระบบจะประมวลผลและอัปเดตสต็อกให้อัตโนมัติ การพัฒนาหน้านี้ใช้ React.js สำหรับส่วนแสดงผล และ Inertia.js ทำหน้าที่เป็นตัวกลางในการติดต่อกับ Laravel Backend เพื่อบันทึกและปรับปรุงข้อมูลในฐานข้อมูล MySQL



รูปที่ 3.2.2 (4) แสดงหน้าชำระเงิน (Checkout)

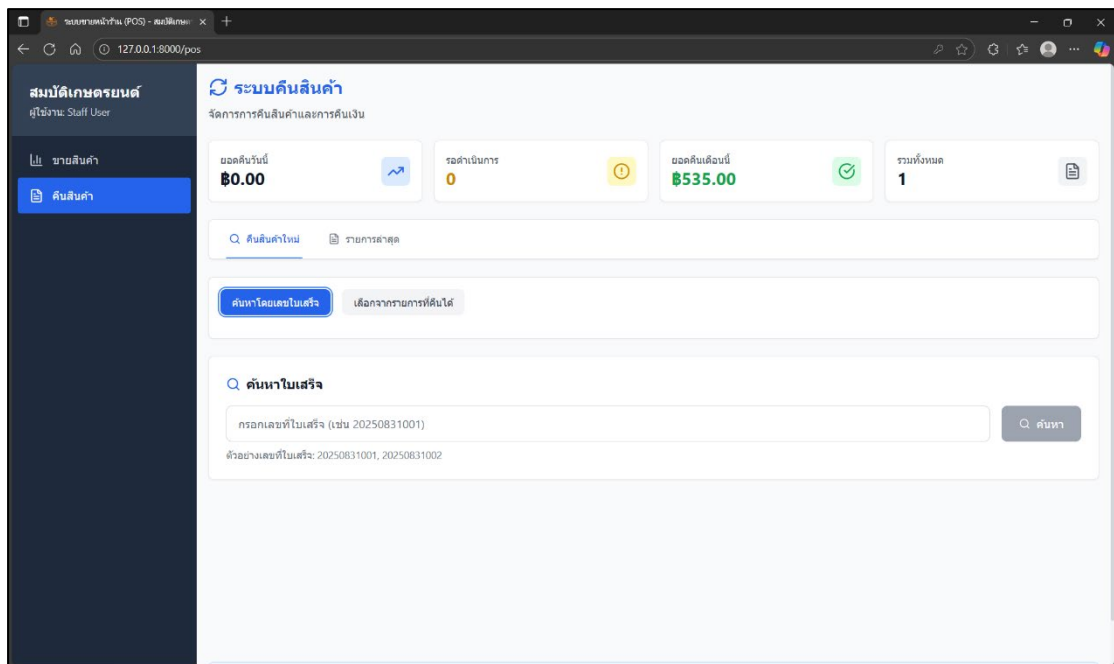
3. หน้าใบเสร็จ (Receipt Page)

หลังจากทำการชำระเงินเสร็จ ระบบจะแสดง ใบเสร็จอิเล็กทรอนิกส์ โดยมีรายละเอียดสินค้า จำนวนเงิน รวมภาษี และ QR Code สำหรับการตรวจสอบหรือชำระเงินผ่านพร้อมเพย์ ผู้ใช้งานสามารถกดพิมพ์ใบเสร็จเพื่อส่งให้ลูกค้าได้ทันที หน้านี้เป็นตัวอย่างของการที่ React.js แสดงผลข้อมูลที่ได้จาก Laravel Backend ผ่าน Inertia.js เพื่อให้ผู้ใช้งานได้รับข้อมูลที่ถูกต้องและครบถ้วน

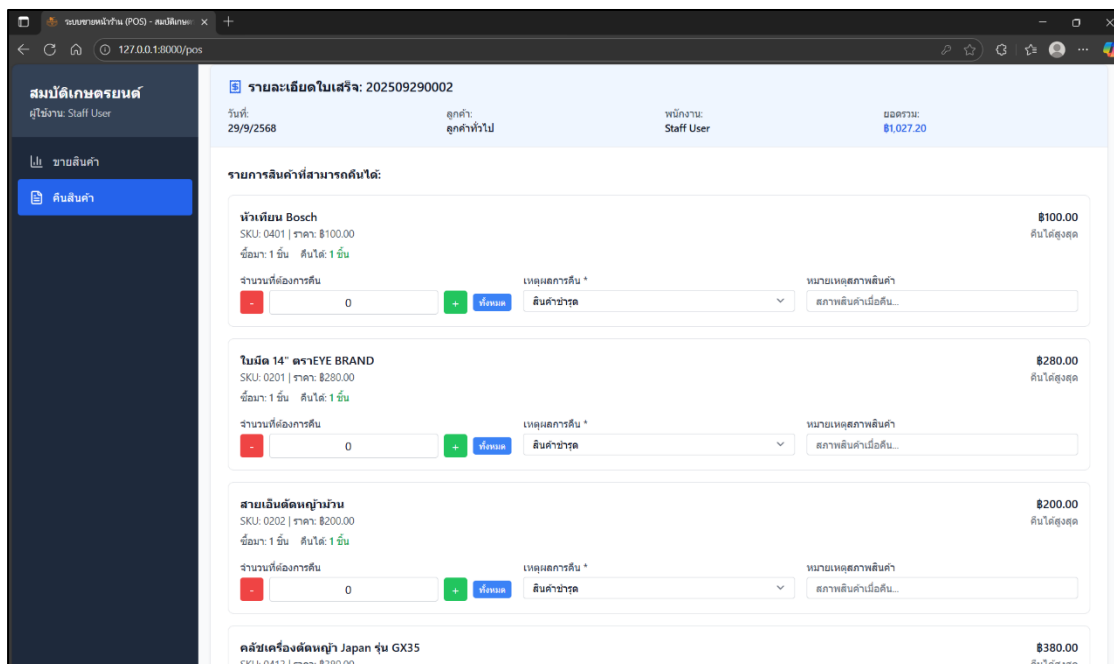


รูปที่ 3.2.2 (5) แสดงใบเสร็จอิเล็กทรอนิกส์

หน้าคีนสินค้าออกแบบมาเพื่อจัดการกับสินค้าที่ลูกค้าส่งคืน โดยผู้ใช้งานสามารถค้นหาใบเสร็จเดิม เลือกรายการสินค้าที่ต้องการคืน ระบุจำนวนและเหตุผลของการคืนสินค้า ระบบจะทำการตรวจสอบความถูกต้อง และอัปเดตสต็อกสินค้าอัตโนมัติ หน้านี้พัฒนาด้วย React.js สำหรับการสร้างฟอร์มการคืน และใช้ Inertia.js เชื่อมต่อกับ Laravel Backend เพื่อปรับปรุงฐานข้อมูลให้สอดคล้องกับการคืนสินค้า



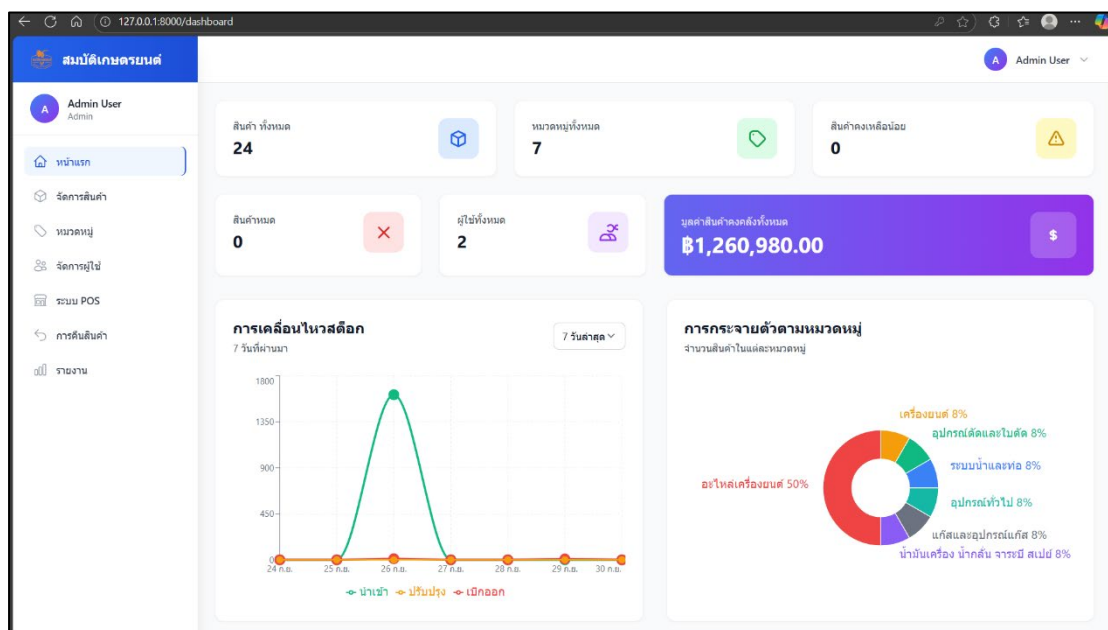
รูปที่ 3.2.2 (6) แสดงหน้าหลักของระบบคืนสินค้า



รูปที่ 3.2.2 (7) แสดงรายละเอียดใบเสร็จและรายการสินค้าที่สามารถเลือกคืนได้

5. หน้าแดชบอร์ดผู้ดูแลระบบ (Admin Dashboard Page)

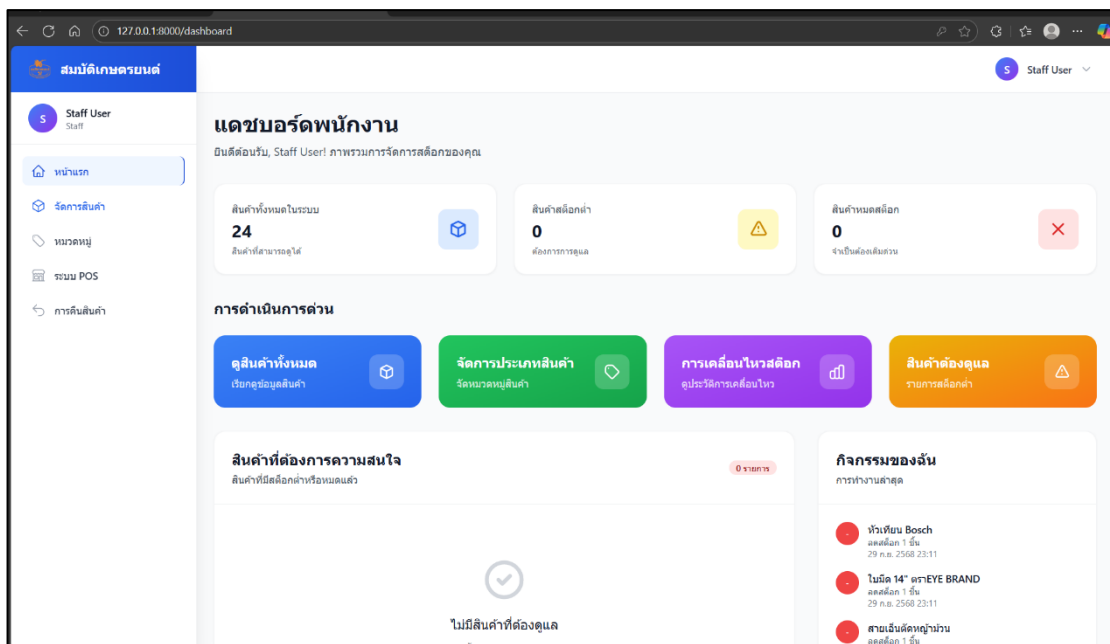
Admin Dashboard แสดงข้อมูลภาพรวมทั้งหมดของระบบ เช่น จำนวนสินค้า, จำนวนหมวดหมู่, จำนวนผู้ใช้งาน, มูลค่ารวมของสินค้าคงคลัง และรายงานการเคลื่อนไหวสต็อก (นำเข้า, ปรับปรุง, เบิกออก) รวมถึงการกระจายสินค้าตามหมวดหมู่ในรูปแบบกราฟ เพื่อให้ผู้ดูแลสามารถวิเคราะห์และตัดสินใจได้อย่างมีประสิทธิภาพ



รูปที่ 3.2.2 (8) แสดง Dashboard ของผู้ดูแลระบบ (Admin)

6. หน้าแดชบอร์ดพนักงาน (Staff Dashboard Page)

Staff Dashboard แสดงข้อมูลที่เกี่ยวข้องกับการทำงานประจำวันของพนักงาน เช่น จำนวนสินค้าที่เหลือในสต็อก, สินค้าที่ต้องดูแล, หมวดหมู่สินค้า, ประวัติการเคลื่อนไหว และกิจกรรมการทำงานล่าสุด โดยจำกัดสิทธิ์ให้สอดคล้องกับหน้าที่ของพนักงาน



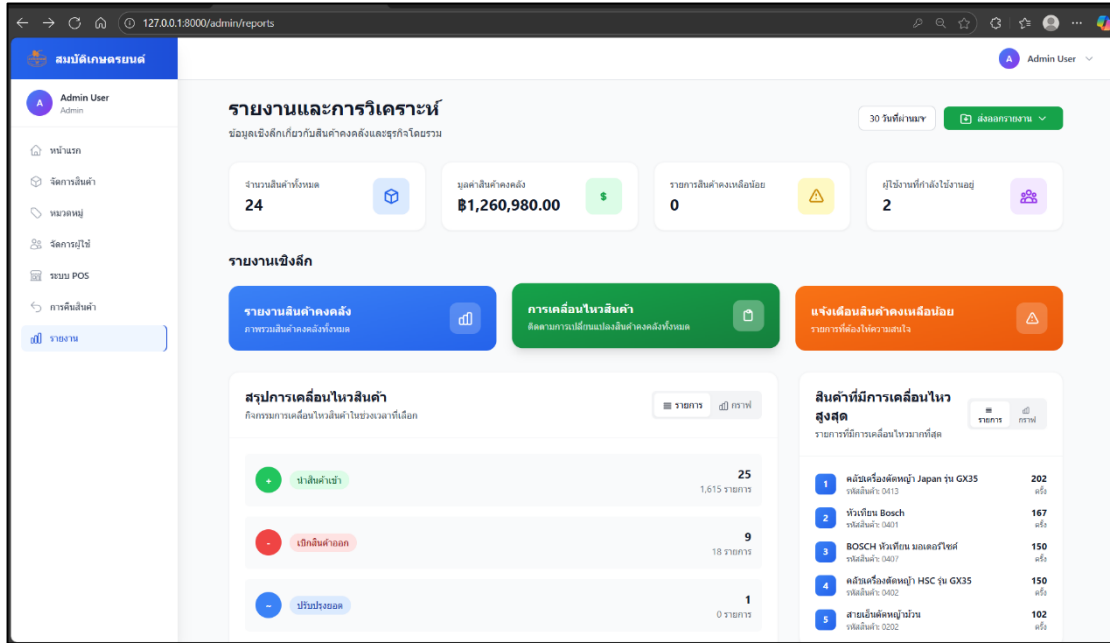
รูปที่ 3.2.2 (9) แสดง Dashboard ของพนักงาน (Staff)

7. หน้าแสดงรายงาน (Reports Page)

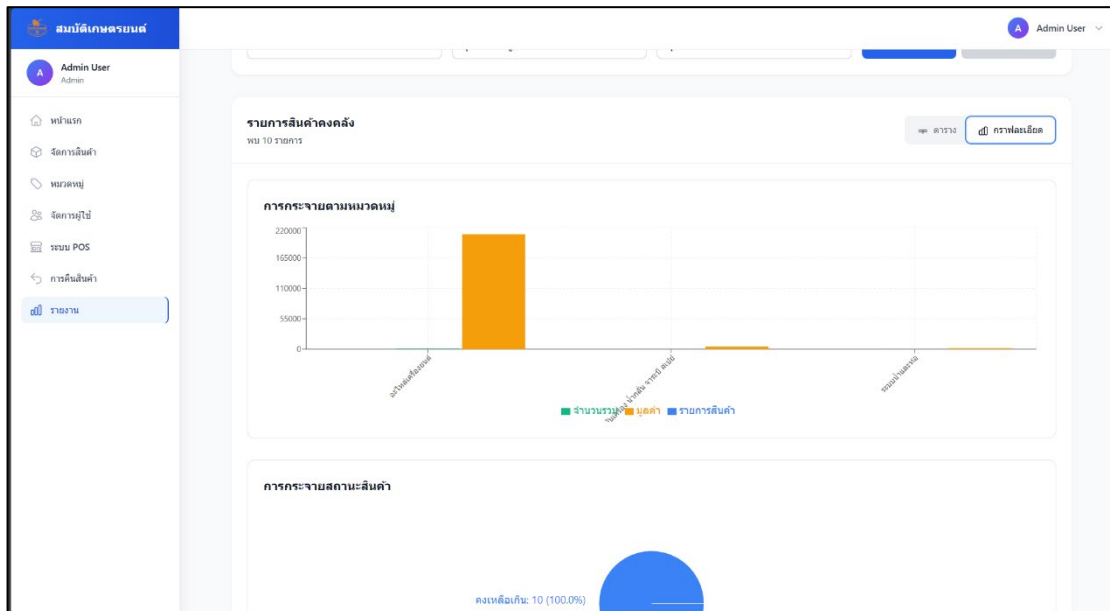
หน้ารายงานเป็นส่วนสำคัญของระบบที่พัฒนาด้วย React.js เพื่อให้ผู้ใช้งานสามารถติดตามและวิเคราะห์ข้อมูลสินค้าคงคลังได้อย่างละเอียด โดยผู้ดูแลระบบ (Admin) สามารถเลือกดูข้อมูลได้ทั้งในรูปแบบ รายการ (Table View) และ กราฟ (Graph View) ซึ่งช่วยให้การนำเสนอข้อมูลมีความยืดหยุ่นและเข้าใจง่ายขึ้น คุณสมบัติของหน้ารายงานได้แก่

- สรุปข้อมูลภาพรวม : จำนวนสินค้าทั้งหมด, มูลค่าคงคลังรวม, สินค้าใกล้หมด และจำนวนผู้ใช้งาน
- รายงานเชิงลึก : สามารถเลือกดูได้ทั้งแบบ “รายการ” (List/Table) และ “กราฟ” (Chart/Graph)
- การกระจายสินค้า : แสดงข้อมูลในรูปแบบกราฟแท่งและแผนภูมิวงกลมเพื่อแสดงหมวดหมู่สินค้าและสถานะสินค้า
- ระบบกรองข้อมูล : ค้นหาและเลือกช่วงเวลาหรือหมวดหมู่สินค้าเฉพาะเพื่อแสดงผลเฉพาะที่ต้องการ
- ส่งออกข้อมูล : รองรับการ Export เป็นไฟล์ CSV สำหรับใช้งานต่อในการวิเคราะห์เชิงธุรกิจ

การใช้ React.js ทำให้ระบบสามารถอัปเดตผลลัพธ์การค้นหาและการแสดงผลกราฟได้ทันทีแบบ Real-time Rendering โดยไม่ต้องโหลดหน้าใหม่ทั้งหมด



รูปที่ 3.2.2 (10) แสดงหน้ารายงาน (Reports Page) ในมุมมองแบบรายการ (Table View)



รูปที่ 3.2.2 (11) แสดงหน้ารายงาน (Reports Page) ในมุมมองแบบกราฟ (Graph View)

3.2.3 ผังเซิร์ฟเวอร์ (Backend)

การพัฒนาระบบคลังสินค้าจำเป็นต้องมี ส่วนประมวลผลหลังบ้าน (Backend) เพื่อทำหน้าที่จัดการข้อมูลและกำหนดตรรกะทางธุรกิจ (Business Logic) โดยในโครงงานนี้ได้เลือกใช้ Laravel Framework (PHP) เป็น Framework หลัก เนื่องจากมีความยืดหยุ่น มีโครงสร้างที่ชัดเจน และมีฟังก์ชันพร้อมใช้งานครบถ้วน เช่น ระบบ Routing การจัดการสิทธิ์ผู้ใช้งาน (Authentication & Authorization) และ Middleware ที่ช่วยเพิ่มความปลอดภัยในการเข้าถึงระบบ

Laravel ยังรองรับการทำงานร่วมกับ ฐานข้อมูล MySQL ได้อย่างมีประสิทธิภาพ โดยใช้ Eloquent ORM (Object Relational Mapping) ในการเชื่อมโยงข้อมูล ทำให้การ Query ข้อมูลและการจัดการตารางต่าง ๆ เป็นไปอย่างสะดวกและเข้าใจง่าย นอกจากนี้ Laravel ยังสนับสนุนการสร้าง Migration เพื่อช่วยให้การจัดการโครงสร้างฐานข้อมูลมีความเป็นระบบและสามารถปรับปรุงหรือย้อนกลับได้เมื่อมีการเปลี่ยนแปลง

อีกหนึ่งเครื่องมือที่มีความสำคัญคือ Composer ซึ่งเป็นตัวจัดการแพ็คเกจของ PHP ที่ช่วยให้นักพัฒนาสามารถติดตั้ง Library หรือโมดูลเสริมต่าง ๆ ได้ง่ายขึ้น โดยไม่ต้องดาวน์โหลดไฟล์ด้วยตนเอง เพียงใช้คำสั่ง `composer require` ระบบก็จะติดตั้งและอัปเดตไลบรารีให้ทันที

3.2.3.1 Laravel Framework (PHP) เป็น Framework ของภาษา

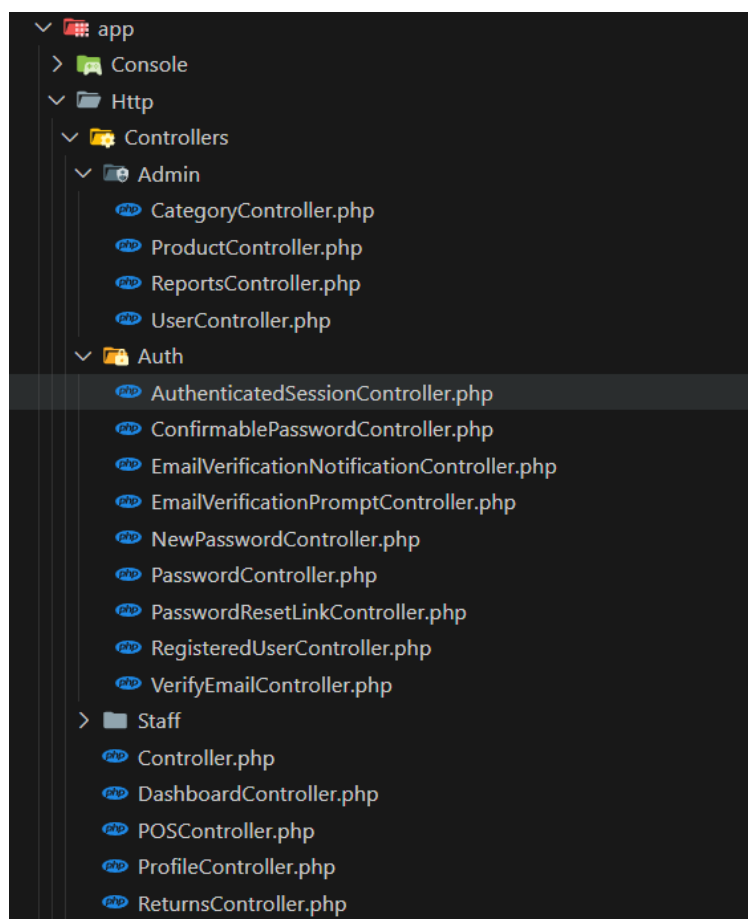
PHP ที่ได้รับความนิยมอย่างกว้างขวางสำหรับการพัฒนาเว็บแอปพลิเคชันและระบบหลังบ้าน (Backend) เนื่องจากมีโครงสร้างที่เป็นระบบระเบียบ ใช้งานง่าย และมีเครื่องมือสนับสนุนการพัฒนาอย่างครบถ้วน Laravel มาพร้อมกับฟังก์ชันพื้นฐานที่สำคัญ เช่น Routing สำหรับกำหนดเส้นทางการทำงานของเว็บแอปพลิเคชัน, Middleware สำหรับจัดการกระบวนการตรวจสอบก่อนและหลังการร้องขอ, Authentication & Authorization สำหรับการเข้าสู่ระบบและการกำหนดสิทธิ์ของผู้ใช้งาน ตลอดจน Eloquent ORM ที่ช่วยให้การเชื่อมต่อและจัดการฐานข้อมูล MySQL เป็นเรื่องง่ายและมีความยืดหยุ่นสูง

ในโครงงานระบบคลังสินค้านี้ Laravel ถูกใช้เป็นแกนหลักของผังเซิร์ฟเวอร์ (Backend) เพื่อทำหน้าที่ควบคุมการทำงานของระบบ ตั้งแต่การจัดการข้อมูลสินค้า ผู้ใช้งาน การเคลื่อนไหวของสต็อก ไปจนถึงการออกรายงาน โดย Laravel ช่วยให้การพัฒนาเป็นไปอย่างมีประสิทธิภาพ รวดเร็ว และปลอดภัย อีกทั้งยัง

รองรับการทำงานร่วมกับ React.js และ Inertia.js ได้อย่างราบรื่น ทำให้การสื่อสารระหว่างฝั่งผู้ใช้และเซิร์ฟเวอร์มีความเสถียร ตัวอย่างการใช้งาน Laravel Framework (PHP) ในระบบ ได้แก่

1. โฟลเดอร์ app/Http/Controllers

โฟลเดอร์นี้ทำหน้าที่เก็บไฟล์ Controller ซึ่งเป็นกลางที่เชื่อมระหว่างส่วนติดต่อผู้ใช้ (Frontend) และตรรกะของระบบ (Business Logic) โดย Controller จะรับคำสั่งจากผู้ใช้ผ่านทาง (Routing) แล้วส่งต่อไปยัง Model หรือฐานข้อมูลเพื่อประมวลผล และส่งผลลัพธ์กลับไปแสดงผลยัง View หรือ React.js ผ่าน Inertia.js ในโครงงานนี้มีการสร้าง Controller หลายตัว เช่น DashboardController สำหรับแสดงแดชบอร์ดสรุปข้อมูล, POSController สำหรับการขายสินค้า, ReturnsController สำหรับจัดการคืนสินค้า และ UserController สำหรับการจัดการข้อมูลผู้ใช้งาน

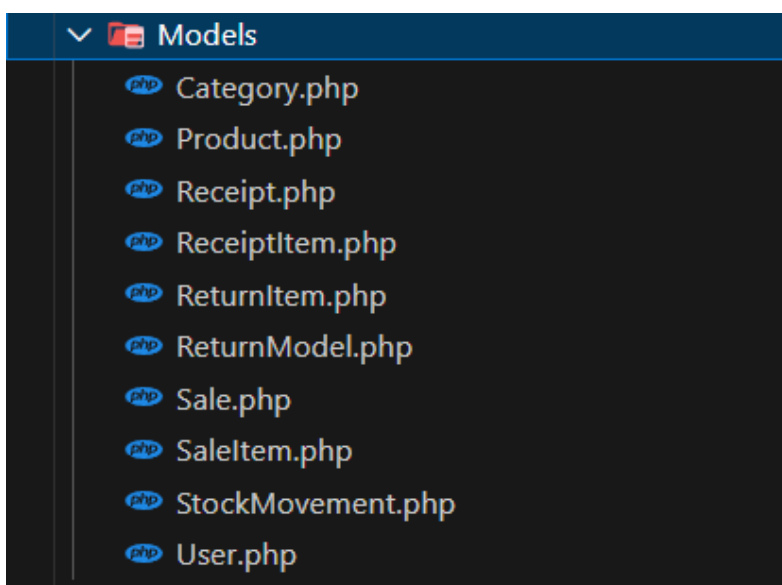


รูปที่ 3.2.3 (1) แสดงโฟลเดอร์ Controllers ของ Laravel ที่ใช้

ควบคุมการทำงานของระบบ

2. โฟลเดอร์ app/Models

โฟลเดอร์นี้เก็บไฟล์ Model ซึ่งเป็นตัวแทนของตารางในฐานข้อมูล โดยใช้คุณสมบัติของ Eloquent ORM ที่ Laravel จัดเตรียมไว้ ทำให้นักพัฒนาสามารถเขียนโค้ดเชื่อมต่อและจัดการข้อมูลในฐานข้อมูลได้ง่ายขึ้น โดยไม่ต้องเขียน SQL ซ้ำซ้อน ตัวอย่างเช่น Product.php ใช้แทนตารางสินค้า, Sale.php ใช้แทนการขายสินค้า, Receipt.php ใช้แทนใบเสร็จ, และ StockMovement.php ใช้แทนการเคลื่อนไหวของสต็อกสินค้า การใช้ Model ทำให้โค้ดมีความเป็นระเบียบ เข้าใจง่าย และช่วยลดความผิดพลาดในการเชื่อมต่อฐานข้อมูล



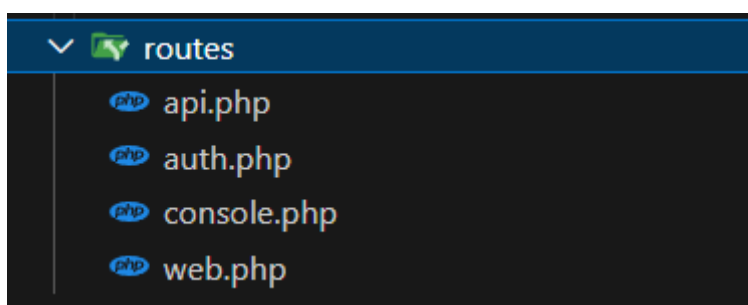
รูปที่ 3.2.3 (2) แสดงโฟลเดอร์ Models ของ Laravel ที่แทนตารางในฐานข้อมูล

3. โฟลเดอร์ routes

โฟลเดอร์ routes/ ใช้เก็บไฟล์กำหนดเส้นทางการทำงาน (Routing) ของแอปพลิเคชัน โดยใน Laravel เส้นทางการเหล่านี้จะเป็นตัวกำหนดว่าคำร้องขอ (Request) จากผู้ใช้จะถูกส่งไปยัง Controller หรือฟังก์ชันใด ตัวอย่างไฟล์ที่สำคัญ ได้แก่

- web.php ใช้กำหนดเส้นทางสำหรับเว็บแอปพลิเคชันทั่วไปที่ทำงานผ่านโปรโตคอล HTTP และรองรับการใช้ Middleware เช่น การตรวจสอบสิทธิ์ผู้ใช้

- api.php ใช้สำหรับกำหนดเส้นทางของ API ซึ่งมักจะส่งข้อมูลในรูปแบบ JSON และเหมาะสำหรับการเชื่อมต่อกับบริการภายนอก หรือ Mobile Application
- auth.php ใช้กำหนดเส้นทางสำหรับการยืนยันตัวตน (Authentication) เช่น การล็อกอิน การสมัครสมาชิก และการรีเซตรหัสผ่าน
- console.php ใช้กำหนดเส้นทางของคำสั่งที่รันผ่าน Artisan Console ทำให้นักพัฒนาสามารถสร้างคำสั่งเฉพาะของตนเองได้



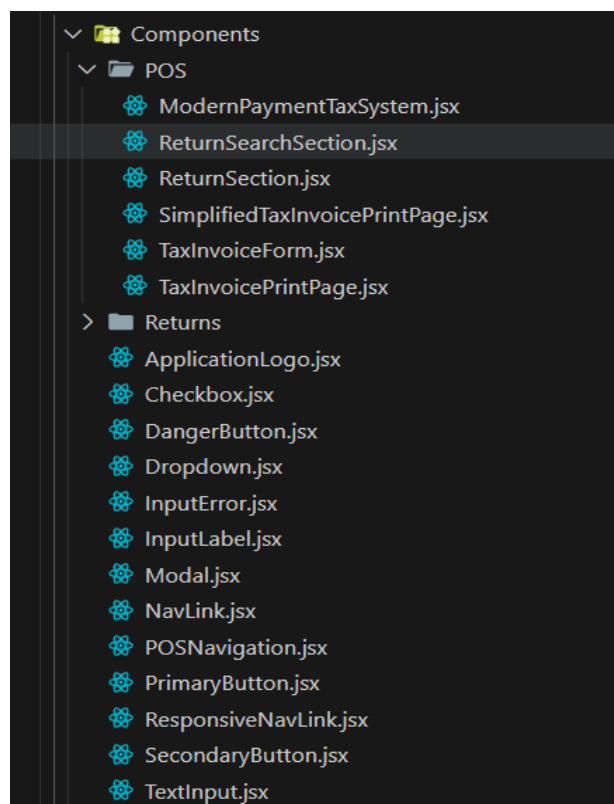
รูปที่ 3.2.3 (3) แสดงโฟลเดอร์ routes

4. โฟลเดอร์ Components

โฟลเดอร์ resources/js/Components/ ใช้สำหรับเก็บ React Component ที่สามารถนำกลับมาใช้ซ้ำได้ในหลาย ๆ หน้า (Reusable Components) ซึ่งช่วยลดการเขียนโค้ดซ้ำซ้อน และทำให้โครงสร้างโค้ดมีความเป็นระเบียบ ตัวอย่างที่สำคัญ ได้แก่

- POS เก็บคอมโพเนนต์ที่เกี่ยวข้องกับระบบขายสินค้า (Point of Sale) เช่น
 - ModernPaymentTaxSystem.jsx ส่วนจัดการภาษีและการชำระเงิน
 - ReturnSection.jsx, ReturnSearchSection.jsx ส่วนของการคืนสินค้า
 - TaxInvoiceForm.jsx, TaxInvoicePrintPage.jsx, SimplifiedTaxInvoicePrintPage.jsx ส่วนของการจัดทำและพิมพ์ใบกำกับภาษี

- Returns เก็บคอมโพเนนต์ที่เกี่ยวข้องกับการคืนสินค้า เช่น ฟอรั่มค้นหาหรือจัดการการคืน
- คอมโพเนนต์ทั่วไป (Generic Components) → ใช้ซ้ำได้หลายหน้า เช่น
 - ApplicationLogo.jsx โลโก้ของระบบ
 - Button หลายแบบ เช่น PrimaryButton.jsx, SecondaryButton.jsx, DangerButton.jsx ปุ่มต่าง ๆ สำหรับการทำงานที่แตกต่างกัน
 - TextInput.jsx, InputError.jsx, InputLabel.jsx ส่วนประกอบของฟอรั่ม
 - Modal.jsx กล่องโต้ตอบ (Dialog)
 - Dropdown.jsx เมนูแบบเลื่อนลง
 - NavLink.jsx, ResponsiveNavLink.jsx ลิงก์นำทางที่ใช้ในเมนู

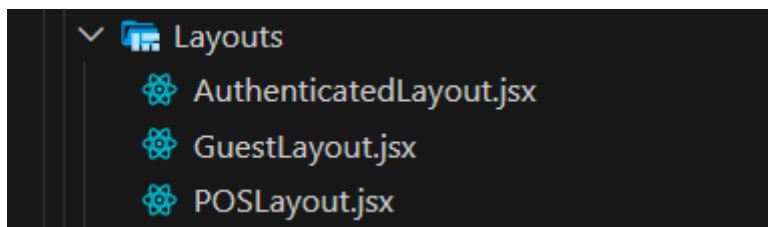


รูปที่ 3.2.3 (4) แสดงไฟล์เตอร์ Components

5. โฟลเดอร์ Layouts

โฟลเดอร์ `resources/js/Layouts/` ใช้สำหรับเก็บโครงร่างหลักของหน้าเว็บ (Page Layouts) ที่สามารถนำไปใช้ซ้ำได้ในหลาย ๆ หน้า เช่น

- `AuthenticatedLayout.jsx` โครงร่างสำหรับผู้ใช้ที่เข้าสู่ระบบแล้ว (เช่น Admin, Staff)
- `GuestLayout.jsx` โครงร่างสำหรับผู้ใช้ที่ยังไม่ได้เข้าสู่ระบบ เช่น หน้า Login, Register
- `POSLayout.jsx` โครงร่างเฉพาะสำหรับหน้าขายสินค้า (Point of Sale)

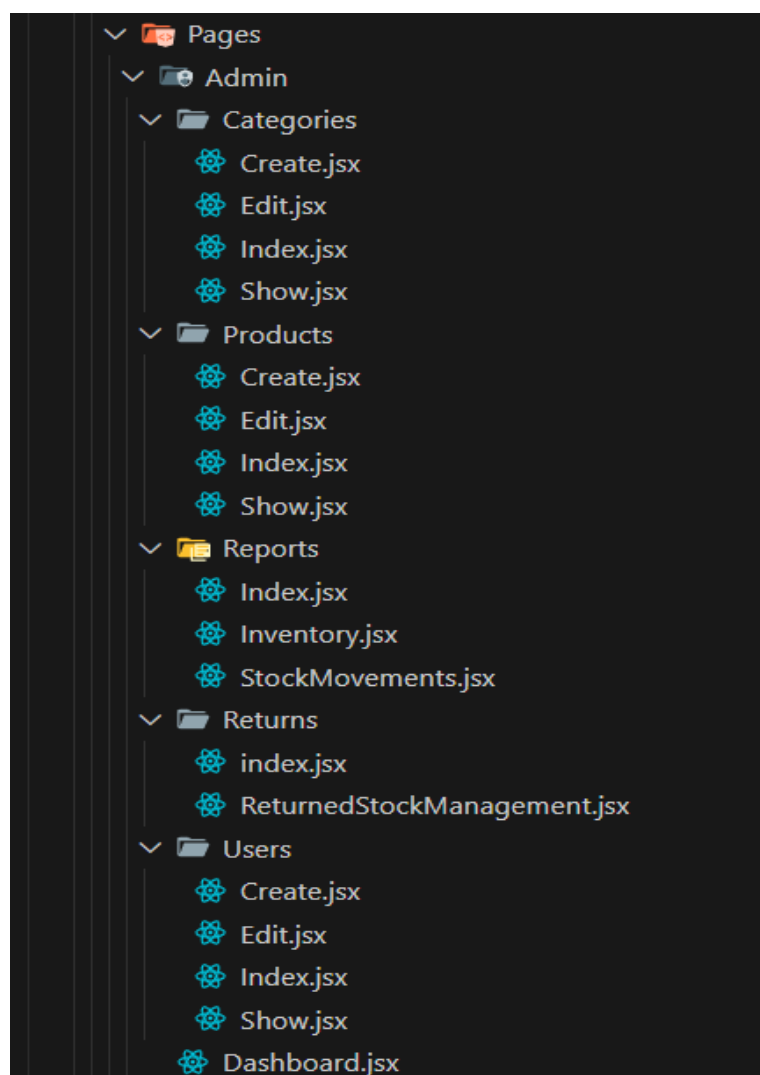


รูปที่ 3.2.3 (5) แสดงโฟลเดอร์ Layouts

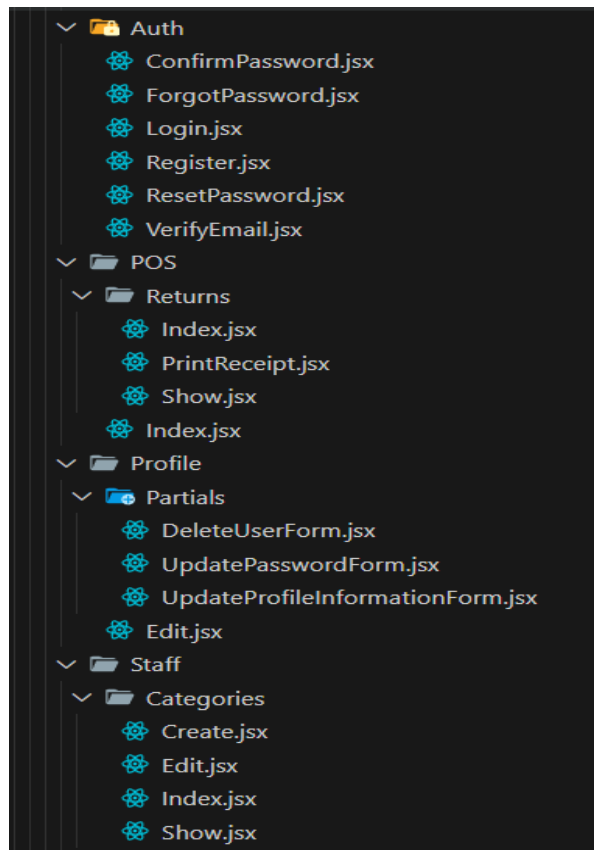
6. โฟลเดอร์ Pages

โฟลเดอร์ `resources/js/Pages/` ใช้เก็บไฟล์หน้าจอหลัก (Page Components) ของระบบ โดยเชื่อมโยงกับ Laravel ผ่าน Inertia.js ซึ่งทำให้การ Render หน้าเว็บแต่ละส่วนสอดคล้องกับ Routing ของ Laravel และสามารถติดต่อกับ Controller/Model ได้โดยตรง ภายในโฟลเดอร์มีการจัดแบ่งหมวดหมู่ เช่น

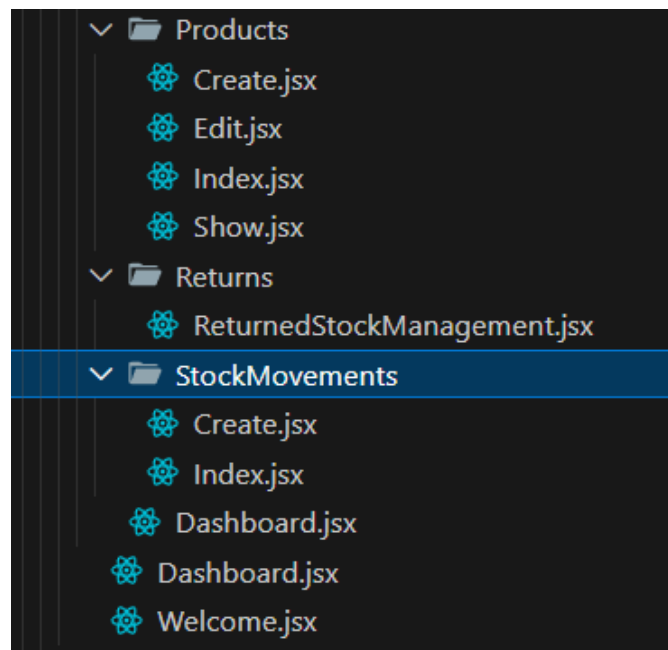
- Admin/ หน้าสำหรับผู้ดูแลระบบ เช่น Dashboard, การจัดการหมวดหมู่สินค้า, สินค้า, ผู้ใช้, รายงาน, การคืนสินค้า
- Auth/ หน้าสำหรับการยืนยันตัวตน เช่น Login, Register, Reset Password, Verify Email
- POS/ หน้าการขายสินค้าและการออกไปเสร็จ
- Profile/ หน้าจัดการโปรไฟล์ผู้ใช้
- Staff/ หน้าสำหรับพนักงาน เช่น การจัดการหมวดหมู่สินค้า
- StockMovements/ หน้าการติดตามการเคลื่อนไหวของสินค้า
- `Welcome.jsx` หน้าแรก (Landing Page) ของระบบ



รูปที่ 3.2.3 (6) แสดงโครงสร้างโฟลเดอร์ Pages/Admin



รูปที่ 3.2.3 (7) แสดงโครงสร้างโฟลเดอร์
Pages/Auth, POS และ Profile

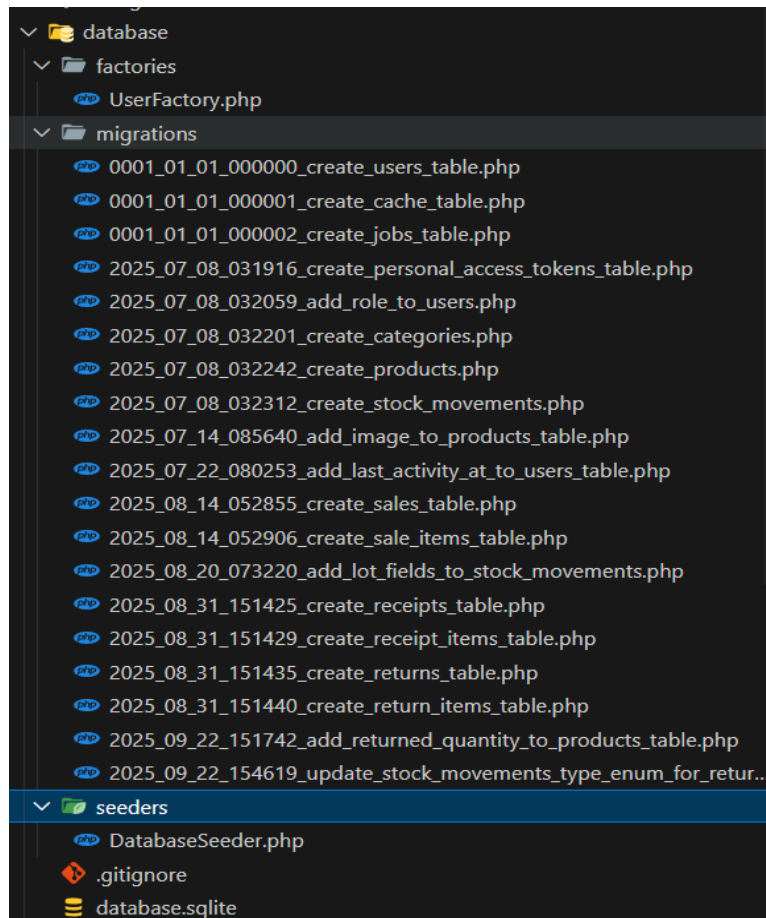


รูปที่ 3.2.3 (8) แสดงโครงสร้างโฟลเดอร์ Pages/Staff,
StockMovements และไฟล์ Welcome.jsx

7. โฟลเดอร์ database

โฟลเดอร์ database/ ของ Laravel ใช้สำหรับจัดการฐานข้อมูลและการสร้างข้อมูลตัวอย่าง โดยแบ่งเป็น 3 ส่วนหลัก ได้แก่

- migrations/ ใช้เก็บไฟล์ Migration ซึ่งเป็นสคริปต์ที่กำหนดโครงสร้างตาราง เช่น ตารางผู้ใช้ (users), ตารางสินค้า (products), ตารางการขาย (sales) เพื่อให้การพัฒนาและการแก้ไขโครงสร้างฐานข้อมูลสามารถทำได้อย่างเป็นระบบและควบคุมเวอร์ชันได้
- seeders/ ใช้สำหรับใส่ข้อมูลเริ่มต้นเข้าสู่ฐานข้อมูล เช่น ข้อมูลผู้ดูแลระบบ (Admin), ข้อมูลสินค้าตัวอย่าง เพื่อให้ นักพัฒนาสามารถทดสอบระบบได้ทันทีโดยไม่ต้องกรอกข้อมูลเอง
- factories/ ใช้สร้างข้อมูลจำลองจำนวนมากเพื่อการทดสอบระบบ เช่น การสร้างข้อมูลผู้ใช้ 100 คนโดยอัตโนมัติ
- database.sqlite ไฟล์ฐานข้อมูล SQLite ที่ระบบผูกใช้งานในระหว่างการพัฒนาและทดสอบ

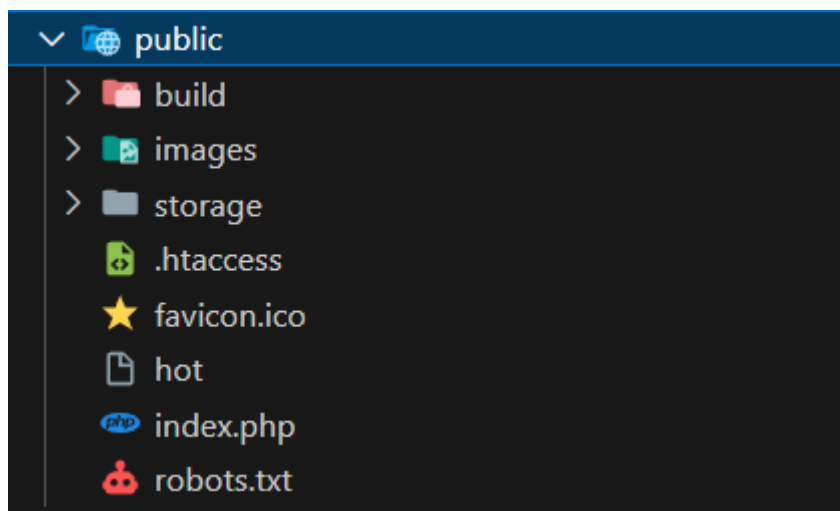


รูปที่ 3.2.3 (9) แสดงโฟลเดอร์ database ของ Laravel

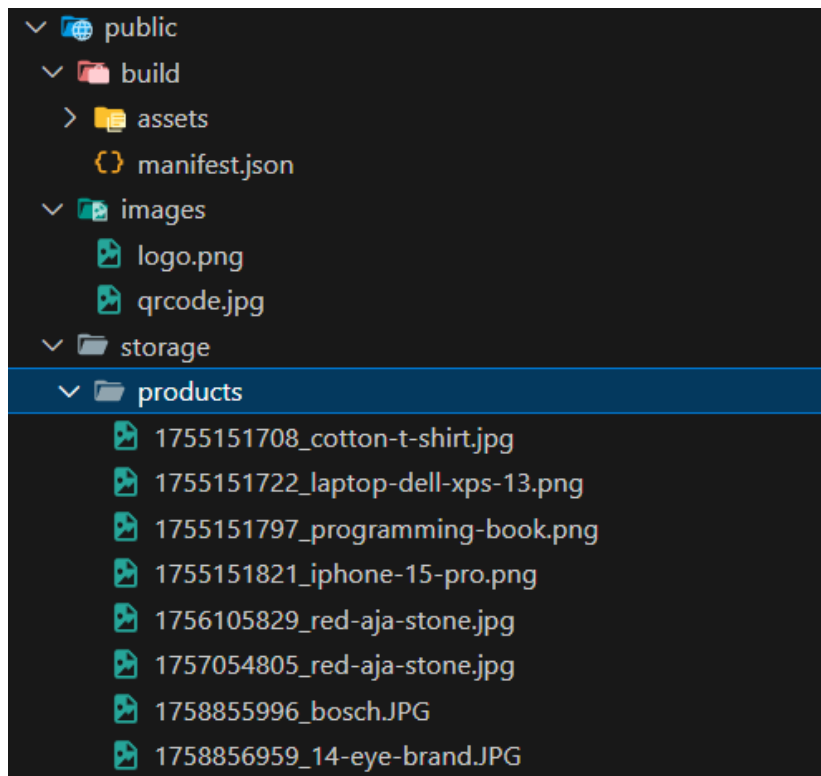
8. โฟลเดอร์ public

โฟลเดอร์ public/ ใน Laravel ทำหน้าที่เป็น Root Directory ของระบบ โดยเป็นจุดเริ่มต้นที่ผู้ใช้งานเข้าถึงเมื่อเปิดเว็บไซต์ ผ่านทาง Web Server ในโครงการนี้โฟลเดอร์ public/ ได้ถูกใช้งานเพื่อเก็บไฟล์ที่สำคัญต่อการทำงานของระบบ เช่น

- index.php ไฟล์เริ่มต้นของระบบ Laravel ที่ทำหน้าที่เป็นตัวกลางในการประมวลผล Request
- build/ เก็บไฟล์ที่ถูก Build มาจาก React เช่น manifest.json เพื่อช่วยในการโหลดหน้าเว็บ
- images/ เก็บรูปภาพที่ใช้ภายในระบบ เช่น โลโก้ (logo.png) และ QR Code สำหรับการชำระเงิน (qrcode.jpg)
- storage/products/ ใช้เก็บรูปภาพสินค้าที่ผู้ดูแลระบบหรือพนักงานอัปโหลดเข้าสู่ระบบ เพื่อให้สามารถเรียกดูสินค้าได้จากหน้า Frontend



รูปที่ 3.2.3 (10) แสดงโฟลเดอร์ public ของ Laravel



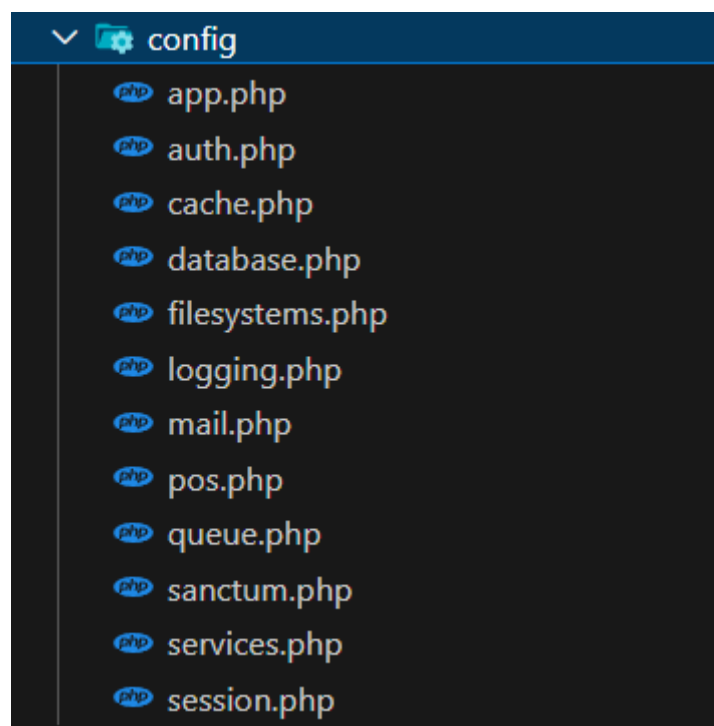
รูปที่ 3.2.3 (11) แสดงโครงสร้างภายในโฟลเดอร์ public

9. โฟลเดอร์ config

โฟลเดอร์ config/ เป็นพื้นที่จัดเก็บไฟล์การตั้งค่าต่าง ๆ ของ Laravel Framework ซึ่งช่วยให้ผู้พัฒนาสามารถปรับแต่งค่าการทำงานของระบบได้ตามความต้องการ โดยไม่จำเป็นต้องแก้ไขโค้ดหลักของเฟรมเวิร์ก ในโครงงานนี้มีไฟล์การตั้งค่าที่สำคัญ เช่น

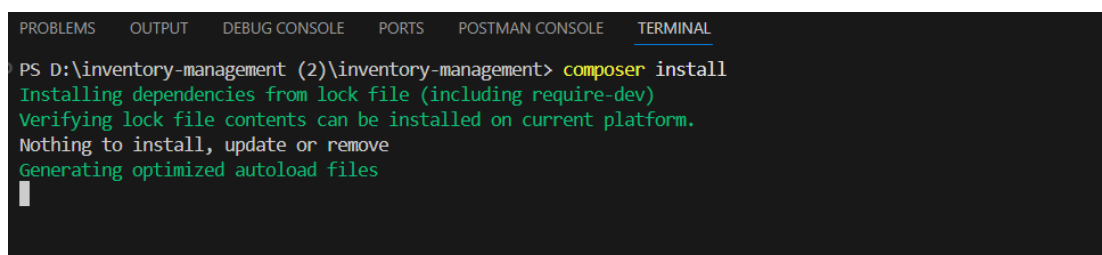
- app.php ใช้กำหนดค่าทั่วไปของแอปพลิเคชัน เช่น timezone, locale และ service provider
- auth.php เกี่ยวข้องกับการตรวจสอบสิทธิ์ผู้ใช้ (Authentication) เช่น การตั้งค่า session guard และ provider ของผู้ใช้
- cache.php ใช้กำหนดค่าการทำงานของระบบแคช เช่น file, database, redis
- database.php ตั้งค่าการเชื่อมต่อฐานข้อมูล (MySQL, SQLite, PostgreSQL ฯลฯ)
- filesystems.php ใช้กำหนดวิธีการจัดเก็บไฟล์ เช่น local, public, หรือ cloud storage

- logging.php จัดการระบบบันทึก Log ของระบบ เช่น channel, driver (single, daily, stack)
- mail.php ใช้ตั้งค่าการส่งอีเมล เช่น SMTP, Mailgun หรือ Sendmail
- pos.php ไฟล์ที่ผู้พัฒนาเพิ่มขึ้นเพื่อรองรับการทำงานของระบบขายหน้าร้าน (Point of Sale)
- queue.php ใช้จัดการการทำงานแบบ Queue สำหรับงานเบื้องหลัง (background jobs)
- sanctum.php ใช้ตั้งค่า API Authentication ด้วย Laravel Sanctum
- services.php เก็บค่า configuration สำหรับบริการภายนอก เช่น Mailgun, AWS, Google API
- session.php ใช้จัดการการทำงานของ session เช่น driver (file, cookie, database) และ lifetime



รูปที่ 3.2.3 (12) แสดงโฟลเดอร์ config ของ Laravel

3.2.3.2 Composer เป็นเครื่องมือจัดการแพ็คเกจ (Package Manager) ของภาษา PHP ที่มีความสำคัญต่อการพัฒนา Backend ด้วย Laravel เนื่องจากช่วยให้นักพัฒนาสามารถติดตั้งและจัดการ Library หรือ โมดูลเสริม (Dependencies) ที่จำเป็นต่อระบบได้อย่างสะดวกและรวดเร็ว เช่น การติดตั้งแพ็คเกจสำหรับการจัดการไฟล์ Excel, การออกรายงาน PDF หรือการทำงานร่วมกับ API ภายนอก โดยเพียงใช้คำสั่ง เช่น `composer require` ระบบจะทำการดาวน์โหลดและติดตั้งไลบรารีให้อัตโนมัติ รวมถึงจัดเก็บข้อมูลการติดตั้งไว้ในไฟล์ `composer.json` เพื่อให้การพัฒนาในทีมเป็นไปในทิศทางเดียวกัน ทำให้โครงการนี้สามารถปรับปรุงและต่อยอดฟังก์ชันได้ง่ายขึ้นในอนาคต



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  POSTMAN CONSOLE  TERMINAL
PS D:\inventory-management (2)\inventory-management> composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Nothing to install, update or remove
Generating optimized autoload files

```

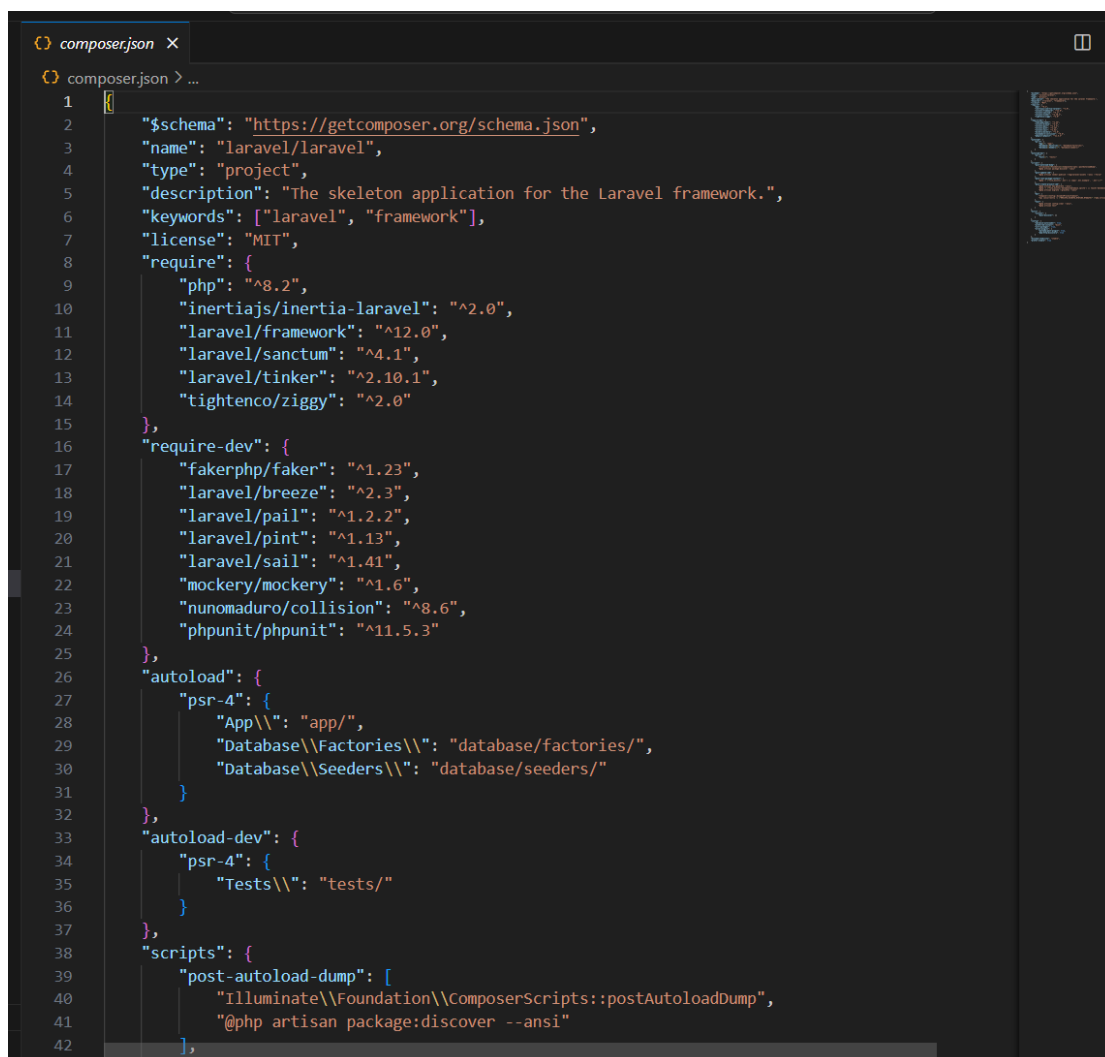
รูปที่ 3.2.3 (13) แสดงการใช้ Composer เพื่อติดตั้งแพ็คเกจที่จำเป็นสำหรับ Laravel

ไฟล์ `composer.json` เป็นไฟล์กำหนดค่า (Configuration File) ที่ใช้สำหรับจัดการ Dependencies ของ PHP และ Laravel ผ่าน Composer โดยจะเก็บข้อมูลสำคัญ เช่น รายชื่อแพ็คเกจที่ต้องใช้, เวอร์ชัน, autoload, และ script ที่รันอัตโนมัติ ในโครงการนี้ไฟล์ `composer.json` มีการติดตั้งแพ็คเกจหลัก เช่น

- `laravel/framework` แกนหลักของ Laravel
- `inertiajs/inertia-laravel` ใช้เชื่อม React.js เข้ากับ Laravel
- `laravel/sanctum` ใช้สำหรับ Authentication ของ API
- `tightenco/ziggy` ช่วยจัดการ route ของ Laravel ในฝั่ง JavaScript
- `laravel/tinker` เครื่องมือทดสอบโค้ด Laravel ผ่าน CLI

และในส่วนของ `require-dev` (ใช้ตอนพัฒนา) เช่น

- `fakerphp/faker` สำหรับสร้างข้อมูลจำลอง
- `laravel/breeze` สำหรับ Authentication template



```

1  {
2      "$schema": "https://getcomposer.org/schema.json",
3      "name": "laravel/laravel",
4      "type": "project",
5      "description": "The skeleton application for the Laravel framework.",
6      "keywords": ["laravel", "framework"],
7      "license": "MIT",
8      "require": {
9          "php": "^8.2",
10         "inertiajs/inertia-laravel": "^2.0",
11         "laravel/framework": "^12.0",
12         "laravel/sanctum": "^4.1",
13         "laravel/tinker": "^2.10.1",
14         "tightenco/ziggy": "^2.0"
15     },
16     "require-dev": {
17         "fakerphp/faker": "^1.23",
18         "laravel/breeze": "^2.3",
19         "laravel/pail": "^1.2.2",
20         "laravel/pint": "^1.13",
21         "laravel/sail": "^1.41",
22         "mockery/mockery": "^1.6",
23         "nunomaduro/collision": "^8.6",
24         "phpunit/phpunit": "^11.5.3"
25     },
26     "autoload": {
27         "psr-4": {
28             "App\\": "app/",
29             "Database\\Factories\\": "database/factories/",
30             "Database\\Seeders\\": "database/seeders/"
31         }
32     },
33     "autoload-dev": {
34         "psr-4": {
35             "Tests\\": "tests/"
36         }
37     },
38     "scripts": {
39         "post-autoload-dump": [
40             "Illuminate\\Foundation\\ComposerScripts::postAutoloadDump",
41             "@php artisan package:discover --ansi"
42         ],

```

รูปที่ 3.2.3 (14) แสดงไฟล์ composer.json ของโครงการ

3.2.4 Visual Studio Code (VS Code)

เป็นโปรแกรมแก้ไขโค้ด (Source Code Editor) ที่ใช้ในการพัฒนาโครงการ โดย VS Code รองรับหลายภาษาโปรแกรม มีระบบส่วนขยาย (Extension) ที่ช่วยให้การพัฒนา Frontend และ Backend เป็นไปอย่างสะดวก เช่น การตรวจสอบความถูกต้องของโค้ด การจัดการ Git และการทำงานร่วมกับ Laravel และ React.js

3.2.5 JavaScript

เป็นภาษาที่ใช้ในฝั่ง Frontend โดยทำงานร่วมกับ React.js เพื่อสร้างหน้าจอที่โต้ตอบได้อย่างมีประสิทธิภาพ และช่วยให้ระบบสามารถทำงานในรูปแบบ Single Page Application (SPA) ได้อย่างราบรื่น

3.2.6 MySQL

เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System: RDBMS) ที่ใช้จัดเก็บข้อมูลทั้งหมดของระบบ เช่น ข้อมูลสินค้า ผู้ใช้ หมวดยี่ห้อ และการเคลื่อนไหวของสต็อก โดยทำงานร่วมกับ Laravel ผ่าน Eloquent ORM เพื่อให้การจัดการข้อมูลเป็นไปอย่างเป็นระบบและมีประสิทธิภาพ

3.3 เทคโนโลยีด้านฐานข้อมูล (Database Technology)

ระบบนี้เลือกใช้ MySQL Database เป็นระบบจัดการฐานข้อมูล (RDBMS) เนื่องจากมีความเสถียร ทำงานได้รวดเร็ว และรองรับการจัดเก็บข้อมูลเชิงสัมพันธ์ เช่น ข้อมูลสินค้า ผู้ใช้ การขาย และการคืนสินค้า ฐานข้อมูลถูกออกแบบให้สัมพันธ์กันผ่าน Primary Key และ Foreign Key เพื่อป้องกันความซ้ำซ้อนของข้อมูล

- ใช้ phpMyAdmin เป็นเครื่องมือช่วยจัดการฐานข้อมูล เช่น การสร้างตาราง การแก้ไข และการ Query
- ระบบรองรับการทำงานร่วมกับ Laravel ผ่าน Eloquent ORM ซึ่งช่วยให้ติดต่อกับฐานข้อมูลได้อย่างสะดวก โดยไม่ต้องเขียน SQL ตรง ๆ ข้อมูลที่จัดเก็บ เช่น

1. ตาราง users → เก็บข้อมูลผู้ใช้งานระบบ (Admin/Staff)

ฟิลด์ที่สำคัญ

- id → รหัสผู้ใช้ (Primary Key)
- name → ชื่อผู้ใช้
- email → อีเมลสำหรับล็อกอิน
- role → บทบาทผู้ใช้ (admin, staff)
- is_active → สถานะการใช้งาน
- password → รหัสผ่าน (เข้ารหัส)
- created_at, updated_at → วันและเวลาที่สร้าง/แก้ไข

ตารางที่ 3.3 (1) ตาราง users

	id	name	email	role	is_active	email_verified_at	last_activity_at	password	remember_token	created_at	updated_at
<input type="checkbox"/>	1	Admin User	admin@example.com	admin	1	NULL	2025-09-30 08:34:54	\$2y\$12\$PdYeKl9tYVvAsAlkH5PeOuj6LOWztlEQnXGsaBZSso...	NULL	2025-09-25 07:43:38	2025-09-30 08:34:54
<input type="checkbox"/>	2	Staff User	staff@example.com	staff	1	NULL	2025-09-30 08:26:10	\$2y\$12\$WCNG3nYmcDuwe4n3gMgqeD82nAjH3pNFdf0x8Qti/...	NULL	2025-09-25 07:43:38	2025-09-30 08:26:10

- name → ชื่อหมวดหมู่สินค้า (เช่น ระบบน้ำและท่อ, เครื่องยนต์)
- description → คำอธิบายรายละเอียดหมวดหมู่
- is_active → สถานะการใช้งาน (1 = ใช้งาน, 0 = ไม่ใช้งาน)
- created_at → วันที่สร้างข้อมูล
- updated_at → วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 3.3 (2) ตาราง categories

		id	name	description	is_active	created_at	updated_at
<input type="checkbox"/>	Edit	1	ระบบน้ำและท่อ	ระบบน้ำและท่อ	1	2025-09-25 07:43:38	2025-09-25 07:43:38
<input type="checkbox"/>	Edit	2	อุปกรณ์ตัดและใบตัด	มีด ใบมีดเครื่องตัดหญ้า สายเอ็น เลื่อย	1	2025-09-25 07:43:38	2025-09-25 07:43:38
<input type="checkbox"/>	Edit	3	เครื่องยนต์	เครื่องสูบน้ำ เครื่องพ่นยา ปั๊มน้ำ เครื่องตัดหญ้า	1	2025-09-25 07:43:38	2025-09-25 07:43:38
<input type="checkbox"/>	Edit	4	อะไหล่เครื่องยนต์	อะไหล่เครื่องตัดหญ้า อะไหล่ปั๊มน้ำ อะไหล่รถไถ อะไหล่...	1	2025-09-25 07:43:38	2025-09-25 07:43:38
<input type="checkbox"/>	Edit	5	น้ำมันเครื่อง น้ำมัน จาระบี สเปรย์	น้ำมันเครื่อง น้ำมัน จาระบี สเปรย์ต่างๆ ดีดฟัน	1	2025-09-26 09:57:46	2025-09-26 06:17:52
<input type="checkbox"/>	Edit	6	แก๊สและอุปกรณ์แก๊ส	แก๊สและอุปกรณ์แก๊ส เคาน์เตอร์ แก๊สแก๊ส อะไหล่แก๊ส	1	2025-09-26 11:34:34	2025-09-26 11:34:34
<input type="checkbox"/>	Edit	7	อุปกรณ์ทั่วไป	อุปกรณ์ทั่วไป ใช้งานทั่วไป	1	2025-09-26 11:40:42	2025-09-26 11:40:42

3. ตาราง products → เก็บข้อมูลสินค้า ตารางนี้ใช้สำหรับจัดเก็บรายละเอียดของสินค้าที่อยู่ในระบบ (ชื่อ, ราคา, สต็อกสินค้า, รายละเอียดสินค้า, หมวดหมู่สินค้า, จำนวนคงเหลือ, จำนวนที่ถูกคืน, ระดับขั้นต่ำของสต็อก, สถานะการใช้งาน, รูปภาพสินค้า, วันที่สร้างและวันที่แก้ไข)

ฟิลด์ที่สำคัญ

- id → รหัสสินค้า (Primary Key)
- name → ชื่อสินค้า
- sku → รหัสสินค้า (Stock Keeping Unit) ใช้สำหรับอ้างอิงสินค้า
- description → รายละเอียดสินค้า
- image → เก็บชื่อไฟล์รูปภาพสินค้า
- category_id → หมวดหมู่สินค้าที่เชื่อมโยงกับตาราง categories (Foreign Key)
- price → ราคาสินค้า
- quantity → จำนวนคงเหลือในสต็อก
- returned_quantity → จำนวนสินค้าที่ถูกคืน

- min_stock → ระดับขั้นต่ำของสต็อก (ใช้แจ้งเตือนเมื่อสินค้าใกล้หมด)
- is_active → สถานะการใช้งาน (1 = ใช้งาน, 0 = ไม่ใช้งาน)
- created_at → วันที่สร้างข้อมูลสินค้า
- updated_at → วันที่แก้ไขข้อมูลล่าสุด
-

ตารางที่ 3.3 (3) ตาราง products

	id	name	sku	description	image	category_id	price	quantity	returned_quantity	min_stock	is_active	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	5	หัวเทียน Bosch	0401	หัวเทียน Bosch แท้ ไม่ถลอก จักรยานยนต์ เครื่องยนต์	1758855996_bosch.JPG	4	100.00	138		5	10	2025-09-26 03:05:41	2025-09-29 16:11:27
<input type="checkbox"/> Edit Copy Delete	6	ไม้มัด 14" ตาEYE BRAND	0201	ไม้มัดติดอยู่กับหัวรถเครื่อง ตัดหญ้า 14 นิ้ว	1758856959_14-eye-brand.JPG	2	280.00	48		0	10	2025-09-26 03:22:39	2025-09-29 16:11:27
<input type="checkbox"/> Edit Copy Delete	7	สายเอ็นดัดหญ้าหมุน	0202	สายเอ็นดัดหญ้าสีขาว เป็น ม้วน หน้า 4.0 มิล ยาว 12มด...	1758857297_.JPG	2	200.00	98		0	5	2025-09-26 03:28:17	2025-09-29 16:11:27
<input type="checkbox"/> Edit Copy Delete	8	คัตเตอร์ตัดหญ้า Japan รุ่น GX35	0413	คัตเตอร์ตัดหญ้า ยี่ห้อ Y- do Japan แบบ 1สปริง จ...	1758857582_.JPG	4	380.00	198		0	20	2025-09-26 03:33:02	2025-09-29 16:11:27
<input type="checkbox"/> Edit Copy Delete	9	คัตเตอร์ตัดหญ้า HSC รุ่น GX35	0402	คัตเตอร์ตัดหญ้า ยี่ห้อ HSC 1สปริง รุ่น GX35	1758857934_hsc.JPG	4	350.00	150		0	10	2025-09-26 03:38:54	2025-09-26 03:55:45
<input type="checkbox"/> Edit Copy Delete	10	ชุดสกรูเครื่องตัดหญ้า GX160	0403	ชุดสกรูเครื่องตัดหญ้า GX160	1758858658_gx160.JPG	4	450.00	50		0	5	2025-09-26 03:50:58	2025-09-26 03:50:58
<input type="checkbox"/> Edit Copy Delete	11	คัตเตอร์ตัดหญ้า รุ่น NB411	0404	คัตเตอร์ตัดหญ้า รุ่น NB411	1758858889_nb411.JPG	4	450.00	100		0	10	2025-09-26 03:54:49	2025-09-26 03:54:49
<input type="checkbox"/> Edit Copy Delete	12	ก๊อมน้ำมันเครื่องพ่นยา รุ่น 411	0406	ก๊อมน้ำมันเครื่องพ่นยา รุ่น 411	1758859372_411.JPG	4	120.00	50		0	5	2025-09-26 04:02:52	2025-09-26 04:45:27
<input type="checkbox"/> Edit Copy Delete	13	BOSCH หัวเทียน มอเตอร์ไซด์	0407	BOSCH หัวเทียน สำหรับ มอเตอร์ไซด์	1758859591_bosch.JPG	4	70.00	150		0	10	2025-09-26 04:06:31	2025-09-26 04:06:31
<input type="checkbox"/> Edit Copy Delete	14	ผ่าสกรูเครื่องตัดหญ้า รุ่น TL43	0408	ผ่าสกรูเครื่องตัดหญ้า รุ่น TL43	1758859790_tl43.JPG	4	350.00	50		0	5	2025-09-26 04:09:50	2025-09-26 04:09:50
<input type="checkbox"/> Edit Copy Delete	15	ผ่าสกรูเครื่องตัดหญ้า รุ่น 411	0409	ผ่าสกรูเครื่องตัดหญ้า รุ่น 411	1758859956_411.JPG	4	350.00	50		0	5	2025-09-26 04:12:36	2025-09-26 04:12:36
<input type="checkbox"/> Edit Copy Delete	16	สปริงรวมสกรู(ลานเบ) รุ่น 5200 5800	0410	สปริงรวมสกรู(ลานเบ) ชุด สปริงลาน 5200 5800 เครื่อง...	1758860139_5200-5800.JPG	4	120.00	50		0	5	2025-09-26 04:15:39	2025-09-26 04:15:39
<input type="checkbox"/> Edit Copy Delete	17	คัตเตอร์ตัดหญ้า รุ่น 767	0411	คัตเตอร์ตัดหญ้า รุ่น 767 ยี่ห้อ HSC	1758860279_767.JPG	4	380.00	50		0	5	2025-09-26 04:18:00	2025-09-26 04:18:00
<input type="checkbox"/> Edit Copy Delete	18	สปริงลานสกรู รุ่น 411 (ลานเบ)	0412	สปริงลานสกรู รุ่น 411 (ลานเบ)	1758860435_411.JPG	4	120.00	50		0	5	2025-09-26 04:20:35	2025-09-26 04:20:35
<input type="checkbox"/> Edit Copy Delete	19	ฟองหนา 2 นิ้ว ตราช้าง	0101	ฟองหนา 2 นิ้ว ตราช้าง	1758860572_2.JPG	1	40.00	30		0	5	2025-09-26 04:22:52	2025-09-26 04:26:34
<input type="checkbox"/> Edit Copy Delete	20	ข้อต่อกรรกล-หนา 2"1นิ้ว ยี่ห้อ SCG	0102	ข้อต่อกรรกล-หนา 2"1นิ้ว ยี่ห้อ SCG (55*25 มม)	1758860769_21-scg.JPG	1	10.00	20		0	5	2025-09-26 04:26:09	2025-09-26 04:26:09

4. ตาราง sales → เก็บข้อมูลการขาย ตารางนี้ใช้สำหรับบันทึก รายละเอียดการขายสินค้า (รหัสการขาย, รหัสผู้ใช้, ยอดรวมราคาสินค้า, มูลค่าภาษี, ยอดรวมสุทธิ, วิธีการชำระเงิน, จำนวนเงินที่ได้รับ, เงินทอน, วันที่ขาย, หมายเลข, วันที่สร้าง และวันที่แก้ไข)

ฟิลด์ที่สำคัญ

- id → รหัสการขาย (Primary Key) ใช้กำหนดรายการขายแต่ละครั้ง
- user_id → รหัสผู้ใช้ที่ทำการขาย (Foreign Key อ้างอิงจาก users)
- total_amount → ยอดรวมราคาสินค้า (ยังไม่รวมภาษี)
- tax_amount → มูลค่าภาษีที่คิดจากยอดขาย
- grand_total → ยอดรวมสุทธิหลังจากรวมภาษีแล้ว

- payment_method → วิธีการชำระเงิน (เช่น cash, creditcard, promptpay)
- received_amount → จำนวนเงินที่ลูกค้าชำระเข้ามา
- change_amount → จำนวนเงินทอนที่ต้องคืนลูกค้า
- sale_date → วันที่และเวลาทำการขาย
- notes → หมายเหตุ เช่น "POS Sale"
- created_at → วันที่สร้างข้อมูลการขาย
- updated_at → วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 3.3 (4) ตาราง sales

	id	user_id	total_amount	tax_amount	grand_total	payment_method	received_amount	change_amount	sale_date	notes	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	1	1000.00	70.00	1070.00	creditcard	NULL	0.00	2025-09-26 06:19:10	POS Sale	2025-09-26 06:19:10	2025-09-26 06:19:10
<input type="checkbox"/> Edit Copy Delete	2	2	960.00	67.20	1027.20	cash	1200.00	172.80	2025-09-29 16:10:16	POS Sale	2025-09-29 16:10:16	2025-09-29 16:10:16
<input type="checkbox"/> Edit Copy Delete	3	2	960.00	67.20	1027.20	promptpay	NULL	0.00	2025-09-29 16:11:27	POS Sale	2025-09-29 16:11:27	2025-09-29 16:11:27

5. ตาราง sale_items → เก็บรายละเอียดสินค้าในแต่ละการขาย (ใช้คู่กับ sales) ตารางนี้ทำหน้าที่บันทึกรายละเอียดของสินค้าที่อยู่ในรายการขาย โดยเชื่อมโยงกับตาราง sales และ products เพื่อระบุว่าสินค้าใดถูกขายออกไปในแต่ละธุรกรรม (รหัสสินค้า, จำนวนที่ขาย, ราคาต่อหน่วย, ราคารวม, วันที่สร้าง และวันที่แก้ไข)

ฟิลด์ที่สำคัญ

- id → รหัสรายการขายสินค้า (Primary Key)
- sale_id → รหัสการขาย อ้างอิงไปยังตาราง sales
- product_id → รหัสสินค้า อ้างอิงไปยังตาราง products
- quantity → จำนวนสินค้าที่ถูกขาย
- unit_price → ราคาต่อหน่วยของสินค้า
- total_price → ราคารวมของสินค้าในรายการนั้น (quantity × unit_price)
- created_at → วันที่สร้างข้อมูล
- updated_at → วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 3.3 (5) ตาราง sale_items

				id	sale_id	product_id	quantity	unit_price	total_price	created_at	updated_at
<input type="checkbox"/>				1	1	5	10	100.00	1000.00	2025-09-26 06:19:10	2025-09-26 06:19:10
<input type="checkbox"/>				2	2	5	1	100.00	100.00	2025-09-29 16:10:16	2025-09-29 16:10:16
<input type="checkbox"/>				3	2	6	1	280.00	280.00	2025-09-29 16:10:16	2025-09-29 16:10:16
<input type="checkbox"/>				4	2	7	1	200.00	200.00	2025-09-29 16:10:16	2025-09-29 16:10:16
<input type="checkbox"/>				5	2	8	1	380.00	380.00	2025-09-29 16:10:16	2025-09-29 16:10:16
<input type="checkbox"/>				6	3	5	1	100.00	100.00	2025-09-29 16:11:27	2025-09-29 16:11:27
<input type="checkbox"/>				7	3	6	1	280.00	280.00	2025-09-29 16:11:27	2025-09-29 16:11:27
<input type="checkbox"/>				8	3	7	1	200.00	200.00	2025-09-29 16:11:27	2025-09-29 16:11:27
<input type="checkbox"/>				9	3	8	1	380.00	380.00	2025-09-29 16:11:27	2025-09-29 16:11:27

6. ตาราง returns → เก็บข้อมูลการคืนสินค้า ตารางนี้ทำหน้าที่เก็บรายละเอียดการคืนสินค้าในระบบ โดยเชื่อมโยงกับใบเสร็จเดิม (original_receipt_id) และผู้ใช้ที่ทำการคืน (user_id) เพื่อให้ระบบสามารถติดตามและตรวจสอบการคืนสินค้าได้อย่างชัดเจน มีการบันทึกยอดเงินคืน ภาษีคืน ประเภทการคืนสินค้า สถานะการคืน และวันที่ทำการรายการ

ฟิลด์ที่สำคัญ

- id → รหัสการคืนสินค้า (Primary Key)
- return_number → เลขที่เอกสารการคืนสินค้า
- original_receipt_id → อ้างอิงถึงใบเสร็จการขายเดิม
- user_id → ผู้ใช้หรือพนักงานที่ทำการคืนสินค้า (Foreign Key ไปยัง users)
- total_return_amount → ยอดรวมราคาสินค้าที่คืน (ไม่รวมภาษี)
- tax_return_amount → มูลค่าภาษีที่คืน
- grand_return_total → ยอดรวมสุทธิของการคืน (รวมภาษี)
- return_type → ประเภทการคืน เช่น full (คืนทั้งหมด) หรือ partial (คืนบางส่วน)
- reason → เหตุผลในการคืนสินค้า
- status → สถานะการคืน เช่น pending, completed
- returned_at → วันที่และเวลาที่ทำการคืน
- notes → หมายเหตุเพิ่มเติมเกี่ยวกับการคืน
- created_at → วันที่สร้างข้อมูลการคืน

- updated_at → วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 3.3 (6) ตาราง returns

	id	return_number	original_receipt_id	user_id	total_return_amount	tax_return_amount	grand_return_total	return_type	reason	status	returned_at	notes	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	R202509260001		1	500.00	35.00	535.00	partial	NULL	completed	2025-09-26 13:19:51	Approved: Auto-approved by system Completed: Compl...	2025-09-26 06:19:51	2025-09-26 06:19:51

7. ตาราง return_items → เก็บรายละเอียดสินค้าที่ถูกคืน (ใช้คู่กับ returns) ตารางนี้ใช้เก็บข้อมูลรายละเอียดของสินค้าแต่ละชิ้นที่ถูกคืน แยกตามรายการคืน (return_id) เพื่อให้ระบบสามารถตรวจสอบและติดตามการคืนสินค้าเป็นรายสินค้าได้ชัดเจน โดยข้อมูลจะอ้างอิงไปยัง ตาราง products และ returns

ฟิลด์ที่สำคัญ

- id → รหัสรายละเอียดการคืนสินค้า (Primary Key)
- return_id → อ้างอิงถึงการคืนสินค้าหลัก (Foreign Key ไปยัง ตาราง returns)
- receipt_item_id → อ้างอิงไปยังสินค้าที่อยู่ในใบเสร็จเดิม
- product_id → รหัสสินค้า (Foreign Key ไปยัง products)
- quantity → จำนวนสินค้าที่ถูกคืน
- unit_price → ราคาต่อหน่วยของสินค้าที่คืน
- total_price → ราคารวมของสินค้าที่คืน
- reason → เหตุผลในการคืนสินค้า (เช่น สินค้าชำรุด, ส่งผิดรุ่น)
- condition_note → หมายเหตุเกี่ยวกับสภาพสินค้า (ถ้ามี)
- created_at → วันที่สร้างข้อมูลการคืน
- updated_at → วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 3.3 (7) ตาราง return_items

	id	return_id	receipt_item_id	product_id	quantity	unit_price	total_price	reason	condition_note	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	1	5	5	100.00	500.00	สินค้าชำรุด	NULL	2025-09-26 06:19:51	2025-09-26 06:19:51

8. ตาราง receipts → เก็บข้อมูลใบเสร็จ/การชำระเงิน ตารางนี้ใช้บันทึกข้อมูลใบเสร็จที่เกิดจากการขายสินค้า โดยแต่ละรายการจะเชื่อมโยงกับการขาย (sales) และผู้ใช้ (users) เพื่อแสดงรายละเอียดเกี่ยวกับจำนวนเงิน วิธีการชำระเงิน และสถานะของใบเสร็จ

ฟิลด์ที่สำคัญ

- id → รหัสใบเสร็จ (Primary Key)
- receipt_number → หมายเลขใบเสร็จ (รันตามลำดับอัตโนมัติ เช่น R202509260001)
- sale_id → อ้างอิงไปยังรายการขาย (Foreign Key ไปที่ sales)
- user_id → ผู้ใช้ที่เกี่ยวข้องกับการขาย
- customer_name → ชื่อลูกค้า (ถ้ามี)
- customer_phone → เบอร์โทรลูกค้า (ถ้ามี)
- customer_tax_id → เลขประจำตัวผู้เสียภาษี (ใช้ในกรณีออกใบกำกับภาษี)
- total_amount → ยอดรวมราคาสินค้า
- tax_amount → ภาษีที่คิดเพิ่มจากการขาย
- grand_total → ยอดรวมสุทธิ (รวมภาษีแล้ว)
- payment_method → วิธีการชำระเงิน (เช่น cash, creditcard, promptpay)
- received_amount → จำนวนเงินที่ลูกค้าจ่ายจริง
- change_amount → เงินทอน (ถ้ามี)
- receipt_type → ประเภทใบเสร็จ (เช่น sale หรือ return)
- status → สถานะของใบเสร็จ (active, canceled เป็นต้น)
- issued_at → วันที่ออกใบเสร็จ
- notes → หมายเหตุเพิ่มเติม
- created_at → วันที่สร้างข้อมูล
- updated_at → วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 3.3 (8) ตาราง receipts

	id	receipt_number	sale_id	user_id	customer_name	customer_phone	customer_tax_id	total_amount	tax_amount	grand_total	payment_method	received_amount	change_amount	receipt_type	status	issued_at	notes	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	202509260001	1	1	NULL	NULL	NULL	1000.00	70.00	1070.00	creditcard	NULL	0.00	sale	active	2025-09-26 06:19:10	NULL	2025-09-26 06:19:10	2025-09-26 06:19:10
<input type="checkbox"/> Edit Copy Delete	2	202509290001	2	2	NULL	NULL	NULL	960.00	67.20	1027.20	cash	1200	172.00	sale	active	2025-09-29 16:10:16	NULL	2025-09-29 16:10:16	2025-09-29 16:10:16
<input type="checkbox"/> Edit Copy Delete	3	202509290002	3	2	NULL	NULL	NULL	960.00	67.20	1027.20	promptpay	NULL	0.00	sale	active	2025-09-29 16:11:27	NULL	2025-09-29 16:11:27	2025-09-29 16:11:27

9. ตาราง receipt_items → เก็บรายละเอียดใบเสร็จ ตารางนี้ใช้บันทึกรายการสินค้าที่อยู่ภายในใบเสร็จ (receipts) โดยจะแสดงว่าสินค้าใดถูกขายไปในแต่ละใบเสร็จ ปริมาณและราคาของสินค้า รวมถึงการเชื่อมโยงกับตารางสินค้า (products) เพื่อให้ระบบสามารถตรวจสอบย้อนหลังได้ว่าในแต่ละการขายมีสินค้าอะไรบ้าง

ฟิลด์ที่สำคัญ

- id → รหัสรายการใบเสร็จ (Primary Key)
- receipt_id → อ้างอิงไปยังใบเสร็จ (Foreign Key → receipts)
- product_id → อ้างอิงไปยังสินค้า (Foreign Key → products)
- product_name → ชื่อสินค้า
- product_sku → รหัสสินค้า (SKU)
- quantity → จำนวนสินค้าที่ขาย
- unit_price → ราคาต่อหน่วย
- total_price → ราคารวมของสินค้านั้น (quantity × unit_price)
- unit → หน่วยของสินค้า (เช่น ชิ้น, กล่อง, แพ็ค)
- returned_quantity → จำนวนที่ถูกคืน (ถ้ามี)
- created_at → วันที่สร้างข้อมูล
- updated_at → วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 3.3 (9) ตาราง receipt_items

+ - > <													
		id	receipt_id	product_id	product_name	product_sku	quantity	unit_price	total_price	unit	returned_quantity	created_at	updated_at
<input type="checkbox"/>				1	1	5 หัวเทียน Bosch	0401	10	100.00	1000.00	ชิ้น	5	2025-09-26 06:19:10 2025-09-26 06:19:51
<input type="checkbox"/>				2	2	5 หัวเทียน Bosch	0401	1	100.00	100.00	ชิ้น	0	2025-09-29 16:10:16 2025-09-29 16:10:16
<input type="checkbox"/>				3	2	6 โคมไฟ 14" ตราEYE BRAND	0201	1	280.00	280.00	ชิ้น	0	2025-09-29 16:10:16 2025-09-29 16:10:16
<input type="checkbox"/>				4	2	7 สายเอ็นคัตถ์ญี่ปุ่น	0202	1	200.00	200.00	ชิ้น	0	2025-09-29 16:10:16 2025-09-29 16:10:16
<input type="checkbox"/>				5	2	8 คลื่นเครื่องตัดหญ้า Japan รุ่น GX35	0413	1	380.00	380.00	ชิ้น	0	2025-09-29 16:10:16 2025-09-29 16:10:16
<input type="checkbox"/>				6	3	5 หัวเทียน Bosch	0401	1	100.00	100.00	ชิ้น	0	2025-09-29 16:11:27 2025-09-29 16:11:27
<input type="checkbox"/>				7	3	6 โคมไฟ 14" ตราEYE BRAND	0201	1	280.00	280.00	ชิ้น	0	2025-09-29 16:11:27 2025-09-29 16:11:27
<input type="checkbox"/>				8	3	7 สายเอ็นคัตถ์ญี่ปุ่น	0202	1	200.00	200.00	ชิ้น	0	2025-09-29 16:11:27 2025-09-29 16:11:27
<input type="checkbox"/>				9	3	8 คลื่นเครื่องตัดหญ้า Japan รุ่น GX35	0413	1	380.00	380.00	ชิ้น	0	2025-09-29 16:11:27 2025-09-29 16:11:27

10. ตาราง stock_movements → บันทึกการเคลื่อนไหวของสต็อก (เช่น รับเข้า, ขายออก, คืนสินค้า) ตารางนี้ใช้สำหรับเก็บข้อมูลทุกการเปลี่ยนแปลงของจำนวนสินค้าภายในคลัง ไม่ว่าจะเป็นการเพิ่ม (รับเข้า), การลด (ขายออก), การคืนสินค้า หรือการปรับปรุงยอด ทำให้ผู้ดูแลสามารถตรวจสอบประวัติการเคลื่อนไหวของสินค้าได้ละเอียด และช่วยในการควบคุมการจัดการสต็อกให้มีความถูกต้อง

ฟิลด์ที่สำคัญ

- id → รหัสการเคลื่อนไหวของสต็อก (Primary Key)
- product_id → รหัสสินค้า (Foreign Key → products) ที่มีการเปลี่ยนแปลงสต็อก
- user_id → รหัสผู้ใช้งานที่ทำรายการ (เชื่อมกับ users)
- type → ประเภทการเคลื่อนไหว เช่น in (รับเข้า), out (ขายออก), return (คืนสินค้า), adjustment (ปรับปรุงยอด)
- quantity → จำนวนสินค้าที่เคลื่อนไหว
- previous_quantity → จำนวนคงเหลือก่อนการเคลื่อนไหว
- new_quantity → จำนวนคงเหลือหลังการเคลื่อนไหว
- notes → หมายเหตุ เช่น "Initial stock entry" หรือ "Quantity adjusted"
- lot_number → หมายเลขล็อตสินค้า (ถ้ามี)
- expiry_date → วันหมดอายุของสินค้า (ใช้กับสินค้าที่มีวันหมดอายุ)
- supplier → ผู้จัดจำหน่าย (ถ้ามีการบันทึก)
- purchase_price → ราคาซื้อของสินค้า (ใช้สำหรับการวิเคราะห์ต้นทุน)
- created_at → วันที่สร้างข้อมูล
- updated_at → วันที่แก้ไขข้อมูลล่าสุด

ตารางที่ 3.3 (10) ตาราง stock_movements

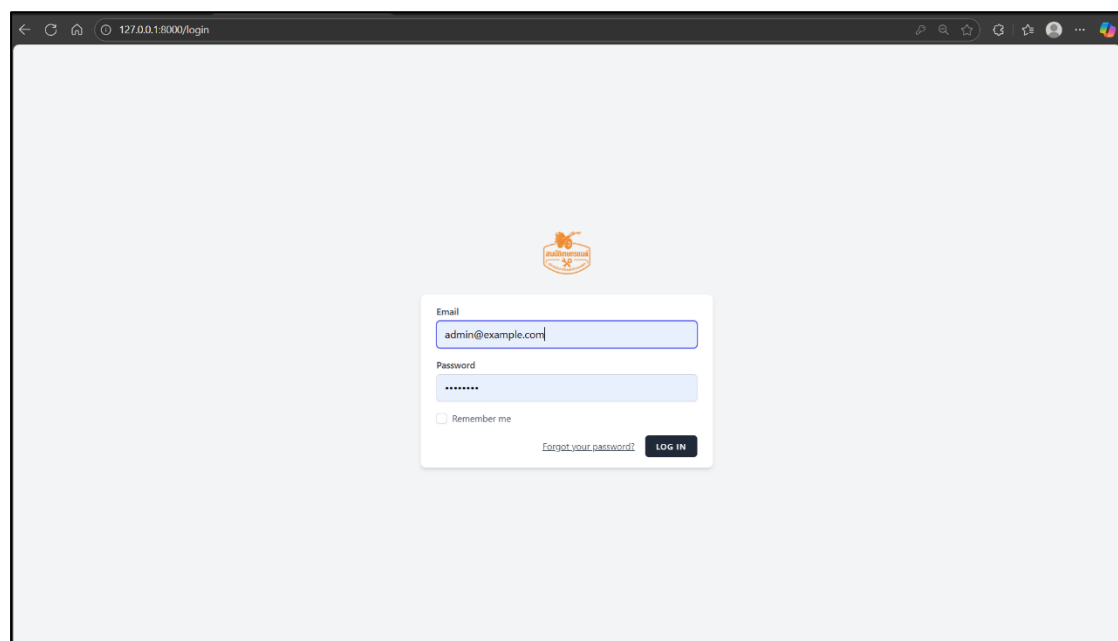
	id	product_id	user_id	type	quantity	previous_quantity	new_quantity	notes	lot_number	expiry_date	supplier	purchase_price	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	5	1	in	150	0	150	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 03:05:41	2025-09-26 03:05:41
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	5	1	adjustment	0	150	150	Quantity adjusted from 150 to 150	NULL	NULL	NULL	NULL	2025-09-26 03:06:36	2025-09-26 03:06:36
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	6	1	in	50	0	50	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 03:22:39	2025-09-26 03:22:39
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	7	1	in	100	0	100	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 03:28:17	2025-09-26 03:28:17
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	8	1	in	200	0	200	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 03:33:02	2025-09-26 03:33:02
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	9	1	in	150	0	150	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 03:38:54	2025-09-26 03:38:54
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	10	1	in	50	0	50	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 03:50:58	2025-09-26 03:50:58
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	11	1	in	100	0	100	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 03:54:49	2025-09-26 03:54:49
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	9	12	1	in	50	0	50	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 04:02:52	2025-09-26 04:02:52
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	10	13	1	in	150	0	150	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 04:06:31	2025-09-26 04:06:31
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	11	14	1	in	50	0	50	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 04:09:50	2025-09-26 04:09:50
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	12	15	1	in	50	0	50	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 04:12:36	2025-09-26 04:12:36
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	13	16	1	in	50	0	50	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 04:15:39	2025-09-26 04:15:39
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	14	17	1	in	50	0	50	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 04:18:00	2025-09-26 04:18:00
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	15	18	1	in	50	0	50	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 04:20:35	2025-09-26 04:20:35
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	16	19	1	in	30	0	30	Initial stock entry	NULL	NULL	NULL	NULL	2025-09-26 04:22:52	2025-09-26 04:22:52

3.4 การรักษาความปลอดภัยและการจัดการผู้ใช้ (Security & User Management)

เพื่อให้ระบบคลังสินค้าและการขายหน้าร้านมีความปลอดภัยต่อข้อมูลและป้องกันการเข้าถึงโดยไม่ได้รับอนุญาต จึงได้ออกแบบและใช้เทคโนโลยีด้านความปลอดภัยดังนี้

3.4.1 การยืนยันตัวตน (Authentication)

ระบบใช้ Laravel Authentication และ Laravel Sanctum สำหรับจัดการการเข้าสู่ระบบ โดยผู้ใช้จะต้องกรอกชื่อผู้ใช้และรหัสผ่านที่ถูกเข้ารหัสก่อนเข้าสู่ระบบ เพื่อป้องกันการเข้าถึงจากผู้ที่ไม่ได้รับอนุญาต



รูปที่ 3.4.1 แสดงหน้า Login

ระบบแบ่งผู้ใช้งานออกเป็น 2 บทบาทหลัก ได้แก่

1. Admin → เข้าถึงและจัดการได้ทุกส่วนของระบบ เช่น การเพิ่ม/แก้ไข/ลบสินค้า การดูรายงาน การจัดการผู้ใช้งาน
2. Staff → เข้าถึงเฉพาะฟังก์ชันที่เกี่ยวข้องกับการทำงาน เช่น การขายสินค้า การคืนสินค้า ไม่สามารถแก้ไขผู้ใช้หรือรายงานระดับสูงได้

3.4.3 การเข้ารหัสรหัสผ่าน (Password Hashing)

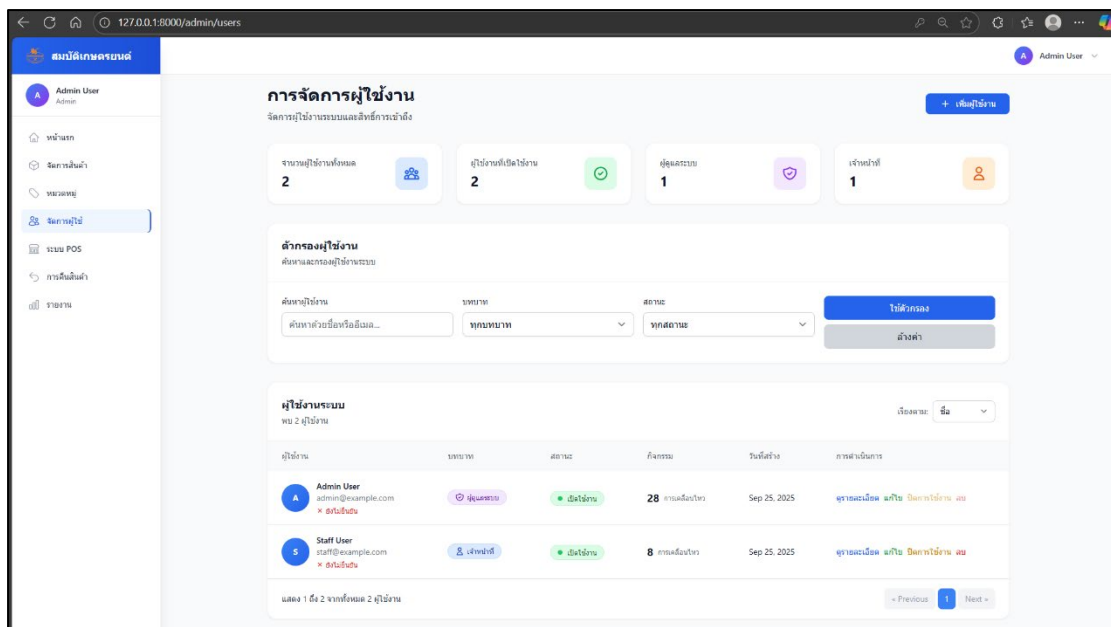
รหัสผ่านทั้งหมดถูกจัดเก็บในรูปแบบเข้ารหัส (Hashing) ด้วยอัลกอริทึม bcrypt/argon2 ซึ่งทำให้ไม่สามารถถอดรหัสกลับเป็นรหัสจริงได้

3.4.4 การป้องกัน CSRF (Cross-Site Request Forgery Protection)

Laravel มีระบบป้องกัน CSRF Token ในทุกการส่งแบบฟอร์ม เพื่อป้องกันการโจมตีโดยใช้ Request ปลอมแปลง

3.4.5 การจัดการ Session และ Token

ระบบใช้ Session และ Token เพื่อตรวจสอบสิทธิ์ของผู้ใช้งานทุกครั้งในการเข้าใช้งาน และเมื่อผู้ใช้ทำการ Logout Session จะถูกลบออกทันที เพื่อความปลอดภัย



รูปที่ 3.4.5 แสดงหน้าจัดการผู้ใช้ (User Management)

ตารางที่ 3.4 ตารางเปรียบเทียบสิทธิ์การใช้งานของผู้ใช้ระบบ

ฟังก์ชันในระบบ	Admin	Staff	หมายเหตุ
เข้าสู่ระบบ (Login)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ทั้งสองบทบาทต้องเข้าสู่ระบบเพื่อใช้งาน
จัดการผู้ใช้งาน (User Management)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Admin เท่านั้นที่สามารถเพิ่ม/แก้ไข/ลบผู้ใช้ได้
จัดการสินค้า (Products)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ทั้งสองทำได้ แต่ Staff ถูกจำกัดสิทธิ์บางส่วน
จัดการหมวดหมู่สินค้า (Categories)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Admin เข้าถึงได้ทั้งหมด, Staff ได้เฉพาะที่อนุญาต
การขายสินค้า (Sales / POS)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ทั้งสองบทบาทใช้งานได้เต็มรูปแบบ
การคืนสินค้า (Returns)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ใช้งานได้ แต่ข้อมูลจะบันทึกว่าใครเป็นผู้ทำการ
การจัดการสต็อก (Stock Movements)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Admin ใช้สำหรับปรับปรุง/ตรวจสอบสต็อก
การดูรายงาน (Reports)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Admin เท่านั้นที่สามารถเข้าถึงรายงานเชิงวิเคราะห์
การตั้งค่าระบบ (System Config)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	จำกัดสิทธิ์เฉพาะ Admin เท่านั้น

บทที่ 4

การวิเคราะห์ระบบและออกแบบระบบ

ในการพัฒนาโครงการระบบคลังสินค้าสำหรับธุรกิจขนาดกลาง (Inventory Management System for SMEs) จำเป็นต้องมีการวิเคราะห์และออกแบบระบบอย่างเป็นระบบ และมีขั้นตอนที่ชัดเจน เพื่อให้การพัฒนาระบบเป็นไปตามวัตถุประสงค์ที่กำหนดไว้ การวิเคราะห์ระบบช่วยให้เข้าใจถึงปัญหาและข้อจำกัดของระบบเดิม รวมถึงการระบุความต้องการของผู้ใช้งาน ในขณะที่การออกแบบระบบเป็นขั้นตอนสำคัญในการกำหนดโครงสร้างของระบบใหม่ ทั้งในด้านสถาปัตยกรรมระบบ การออกแบบฐานข้อมูล และการออกแบบส่วนติดต่อผู้ใช้ (User Interface)

4.1 การวิเคราะห์ระบบ

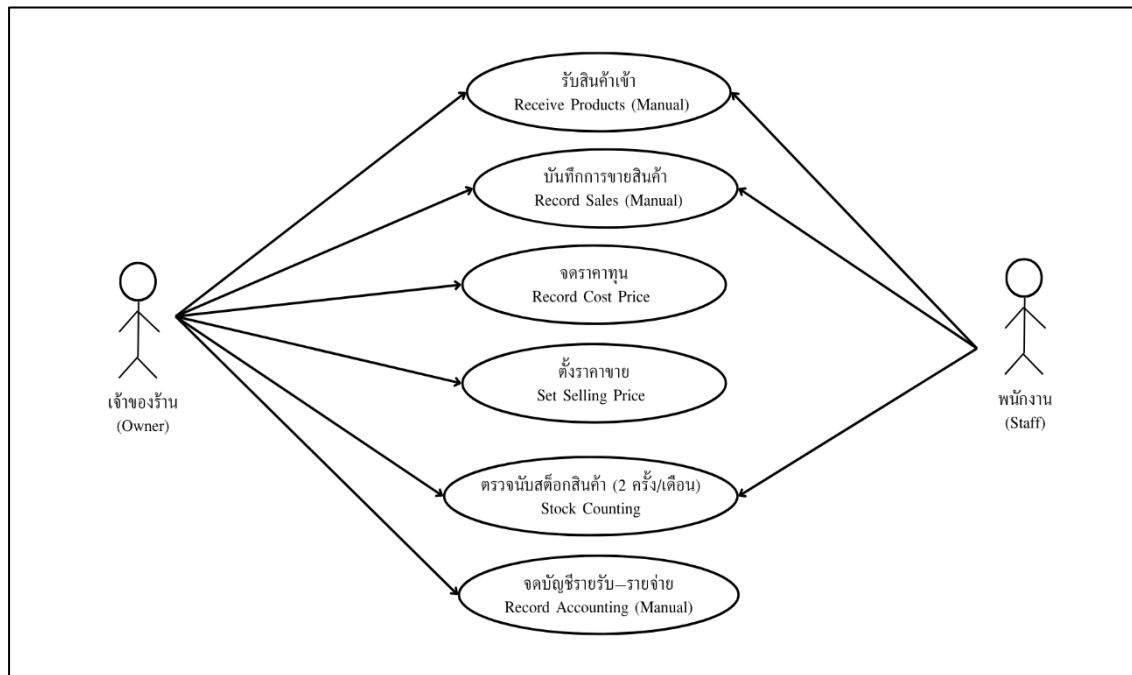
การวิเคราะห์ระบบเป็นขั้นตอนที่ใช้ในการศึกษาปัญหา ข้อจำกัด และความต้องการของผู้ใช้งาน เพื่อนำไปสู่การออกแบบระบบใหม่ที่ตอบโจทย์การทำงานของธุรกิจ โดยในโครงการนี้ ได้ทำการวิเคราะห์ดังนี้

4.1.1 การวิเคราะห์ระบบปัจจุบัน

จากการศึกษาพบว่า ร้านค้ายังคงใช้การจัดการคลังสินค้าในรูปแบบแมนนวล (Manual System) โดยการจดบันทึกข้อมูลลงในสมุดบัญชีหรือเอกสารทั่วไป ซึ่งวิธีการดังกล่าวมีข้อจำกัดหลายประการ อาทิ ความผิดพลาดที่อาจเกิดขึ้นจากการกรอกข้อมูลด้วยมือ ความซ้ำซ้อนของข้อมูล และความล่าช้าในการตรวจสอบย้อนหลัง นอกจากนี้ยังไม่มีฟังก์ชันที่ช่วยแจ้งเตือนอัตโนมัติเมื่อจำนวนสินค้าลดต่ำกว่าระดับที่กำหนด อีกทั้งยังขาดระบบรายงานที่สามารถนำเสนอข้อมูลในเชิงวิเคราะห์เพื่อสนับสนุนการตัดสินใจของผู้บริหารได้อย่างทันทั่วถึง

การดำเนินงานปัจจุบันประกอบด้วยขั้นตอนหลัก เช่น การรับสินค้าเข้าและบันทึกราคาทุน การตั้งราคาขายสินค้า การบันทึกยอดขายในแต่ละวัน การตรวจนับสต็อกสินค้าประมาณเดือนละสองครั้ง และการบันทึกกำไร-รายจ่ายลงในสมุดบัญชี กระบวนการเหล่านี้อาศัยแรงงานคนทั้งหมด ส่งผลให้ข้อมูลที่ได้อาจไม่ถูกต้อง ครบถ้วน หรือไม่สะดวกต่อการนำไปใช้ในการวิเคราะห์เชิงลึก

เพื่อแสดงให้เห็นถึงลักษณะการทำงานของระบบปัจจุบัน จึงได้ทำการจัดทำ Use-Case Diagram ของระบบเดิม โดยแผนภาพดังกล่าวสะท้อนให้เห็นถึงกิจกรรมที่เจ้าของร้านและพนักงานต้องปฏิบัติด้วยตนเอง เช่น การรับสินค้า การบันทึกการขาย การตรวจนับสต็อก และการจดบัญชีรายรับ-รายจ่าย ซึ่งชี้ให้เห็นข้อจำกัดด้านความถูกต้อง ความรวดเร็ว และประสิทธิภาพของระบบการจัดการคลังสินค้าแบบแมนนวล



รูปที่ 4.1.1 Use-Case Diagram ของระบบปัจจุบัน (Manual System)

4.1.2 ปัญหาและข้อจำกัด

จากการวิเคราะห์ระบบปัจจุบันของร้าน ซึ่งยังใช้การบันทึกและการจัดการคลังสินค้าในรูปแบบแมนนวล (Manual System) พบว่ามีข้อจำกัดและความเสี่ยงหลายประการ ดังนี้

- ความถูกต้องของข้อมูล (Human Error)
การบันทึกสินค้าคงคลังด้วยการเขียนมือ มีความเสี่ยงที่จะเกิดความผิดพลาด เช่น การกรอกจำนวนไม่ตรง ความซ้ำซ้อนของข้อมูล หรือการลืมบันทึก ส่งผลให้ข้อมูลไม่ตรงกับความเป็นจริง
- การนับสต็อกและติดตามการเคลื่อนไหวไม่ต่อเนื่อง
ร้านทำการตรวจนับสต็อกเดือนละเพียง 2 ครั้ง ทำให้ไม่สามารถทราบจำนวนสินค้าที่แท้จริงได้ตลอดเวลา อีกทั้งไม่สามารถติดตามได้ว่าสินค้าออกจากคลังไปเมื่อใดและด้วยสาเหตุใด
- การสั่งสินค้าไม่แม่นยำ
เนื่องจากไม่มีการบันทึกจำนวนสินค้าคงเหลือแบบเรียลไทม์ การสั่งซื้อสินค้าจึงไม่สามารถกำหนดเกณฑ์ที่ชัดเจนได้ บางครั้งอาจเกิดการขาดสต็อก (Stockout) หรือสั่งสินค้ามากเกินไปจนเกิดความจำเป็น (Overstock)
- ข้อผิดพลาดด้านการทำบัญชี

การบันทึกรายรับ-รายจ่ายลงสมุดบัญชีไม่สามารถทำได้สม่ำเสมอ ทำให้เกิดการคาดเคลื่อนของข้อมูล ทั้งในด้านจำนวนเงิน รายการสินค้า และยอดคงเหลือ รวมถึงความผิดพลาดจากการคำนวณด้วยมือ

- การตรวจสอบย้อนหลังทำได้ยาก
หากลูกค้าต้องการคืนสินค้าตามเงื่อนไขการรับประกัน ร้านไม่สามารถตรวจสอบบิลย้อนหลังได้ หากบิลเขียนมือสูญหายไป ทั้งจากฝั่งลูกค้าหรือร้านค้าเอง
- การออกบิลและใบกำกับภาษีไม่สมบูรณ์
ร้านยังคงใช้บิลเขียนมือ ซึ่งไม่ใช่ใบกำกับภาษีแบบสมบูรณ์ ทำให้ไม่สามารถใช้เป็นเอกสารทางการแพทย์หรือภาษีได้อย่างถูกต้อง อีกทั้งยังสร้างความไม่มั่นใจให้กับลูกค้าเมื่อต้องการหลักฐานการซื้อขาย
- การจัดการสินค้าเป็นล็อตไม่มีประสิทธิภาพ
ร้านไม่สามารถติดตามได้ว่าสินค้าใดเป็นล็อตเก่าหรือใหม่ บางครั้งมีการขายสินค้าล็อตใหม่ก่อนล็อตเก่า อีกทั้งราคาทุนของแต่ละล็อตไม่เท่ากัน ทำให้การคำนวณต้นทุนและกำไรไม่แม่นยำ
- การจัดการสินค้าคืนเคลมไม่มีระบบ
เมื่อมีสินค้าที่ลูกค้าเคลมคืนเพราะเสียหายภายในระยะเวลาประกัน ร้านต้องนำสินค้าจากคลังไปเปลี่ยนใหม่ให้ลูกค้า ส่วนสินค้าที่เสียหายจะต้องเก็บแยกไว้เพื่อส่งคืนบริษัทผู้จำหน่าย แต่เนื่องจากไม่มีระบบบันทึกวันและเงื่อนไขการเคลมที่ชัดเจน ทำให้บางครั้งร้านลืมวันหมดประกัน ส่งผลให้สินค้าไม่สามารถเคลมกับบริษัทได้และกลายเป็นต้นทุนที่ร้านต้องรับผิดชอบเอง
- ความปลอดภัยและความสะดวกในการเข้าถึงข้อมูลต่ำ
การเก็บข้อมูลลงสมุดหรือเอกสารกระดาษมีความเสี่ยงที่จะสูญหาย เสียหายจากไฟ น้ำ หรือการใช้งานผิดพลาด อีกทั้งการค้นหาข้อมูลย้อนหลังต้องใช้เวลาเปิดเอกสารจำนวนมาก ทำให้สิ้นเปลืองเวลาและไม่สะดวกต่อการใช้งาน

ข้อจำกัดเหล่านี้สะท้อนให้เห็นว่าระบบปัจจุบันของร้านที่ยังใช้วิธีการบันทึกแบบแมนนวล ไม่สามารถรองรับการทำงานได้อย่างมีประสิทธิภาพ ทั้งในด้านการจัดเก็บข้อมูลที่เสี่ยงต่อความผิดพลาดและการสูญหาย การตรวจสอบสต็อกที่ทำได้ไม่ต่อเนื่อง การสั่งซื้อสินค้าที่ขาดความแม่นยำ การทำบัญชีที่มีโอกาสคาดเคลื่อน รวมถึงการออกบิลและใบกำกับภาษีที่ไม่สมบูรณ์ อีกทั้งยังไม่มีระบบติดตามการเคลมสินค้า ทำให้บางครั้งสินค้าหมดระยะประกันก่อนส่งคืนบริษัท ความซับซ้อนเหล่านี้ส่งผลกระทบ

โดยตรงต่อการควบคุมสินค้า ต้นทุน และความน่าเชื่อถือของร้านค้า ดังนั้นจึงมีความจำเป็นต้องพัฒนาระบบใหม่ที่จะช่วยจัดการข้อมูลสินค้าและบัญชีอย่างเป็นระบบ มีความถูกต้อง สามารถตรวจสอบย้อนหลังได้สะดวก รองรับการติดตามสินค้าคืนเคลม และสร้างรายงานสรุปเพื่อสนับสนุนการตัดสินใจของผู้บริหารได้อย่างทันท่วงที

4.1.3 ความต้องการของผู้ใช้งาน

จากการศึกษาข้อจำกัดและปัญหาของระบบปัจจุบัน สามารถสรุปความต้องการของผู้ใช้งานระบบคลังสินค้าใหม่ได้ว่า ผู้ใช้งานต้องการระบบที่สามารถทำงานได้อย่างมีประสิทธิภาพ ถูกต้อง และตอบโจทย์การใช้งานจริง โดยเฉพาะอย่างยิ่งการลดความผิดพลาดจากการทำงานแบบแมนนวล และการสนับสนุนการตัดสินใจเชิงธุรกิจ ความต้องการเหล่านี้สามารถแบ่งออกได้ตามบทบาทของผู้ใช้งานหลัก 2 กลุ่ม คือ ผู้ดูแลระบบ (Admin) และ พนักงาน (Staff) ดังนี้

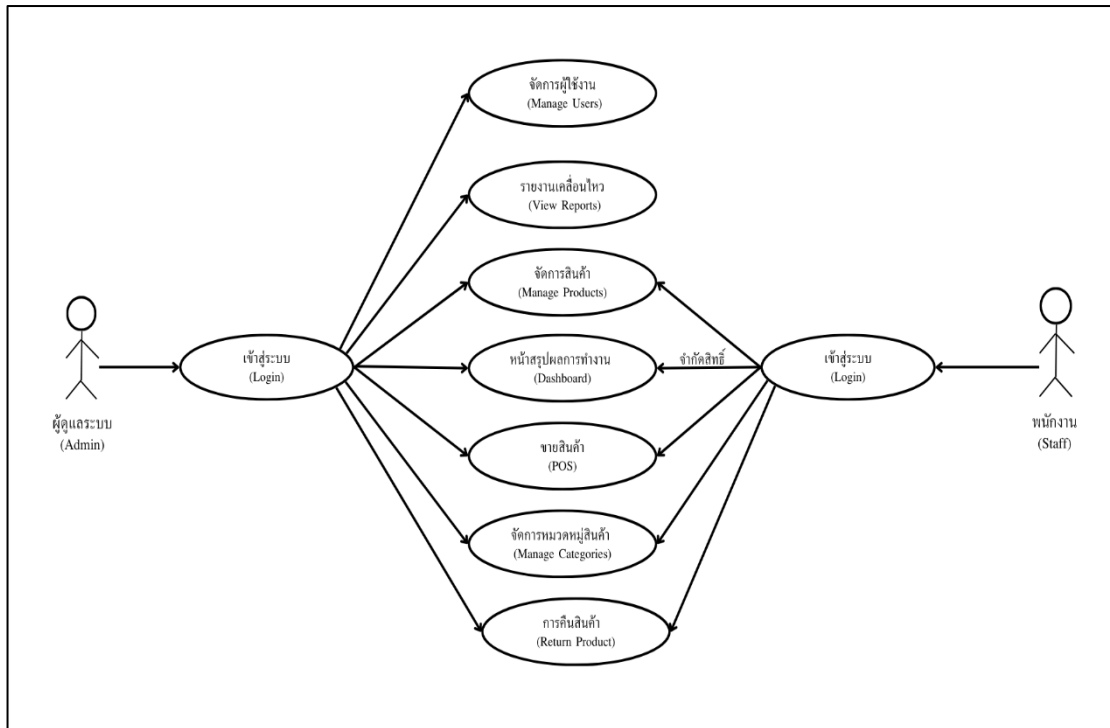
ความต้องการของผู้ดูแลระบบ (Admin)

- การจัดการผู้ใช้งาน (User Management) สามารถเพิ่ม ลบ แก้ไข และกำหนดสิทธิ์การเข้าถึงของผู้ใช้งานได้
- การจัดการสินค้าและหมวดหมู่สินค้า (Product & Category Management) สามารถเพิ่ม แก้ไข หรือลบสินค้าและหมวดหมู่ได้ รวมถึงการกำหนดจำนวนขั้นต่ำเพื่อใช้ในการแจ้งเตือน และจัดการสินค้าตามล็อต
- ระบบแจ้งเตือนอัตโนมัติ (Alert System) ต้องการระบบแจ้งเตือนเมื่อสินค้าลดลงต่ำกว่าระดับที่กำหนด เพื่อป้องกันการขาดสต็อก
- การตรวจสอบการเคลื่อนไหวของสินค้า (Stock Movement Monitoring) สามารถดูประวัติการนำเข้า เบิกออก และการคืนสินค้าได้อย่างละเอียด
- การออกรายงานวิเคราะห์ (Analytical Reports) ต้องการรายงานในรูปแบบกราฟ ตาราง และ Dashboard เพื่อช่วยวิเคราะห์ข้อมูลเชิงธุรกิจ เช่น สินค้าขายดี ต้นทุนสินค้าคงคลัง หรือมูลค่าการขาย
- การเชื่อมโยงกับระบบอื่น (System Integration) ต้องการให้ระบบสามารถเชื่อมต่อกับระบบ POS หรือระบบบัญชี เพื่ออัปเดตข้อมูลโดยอัตโนมัติ
- ความปลอดภัยและการติดตาม (Security & Audit Log) ต้องการระบบที่มีการบันทึกประวัติการเข้าใช้งานและการเปลี่ยนแปลงข้อมูล เพื่อความโปร่งใสและปลอดภัย

- การจัดการบิลและภาษี: ออกบิลย้อนหลังและใบกำกับภาษีที่ถูกต้องตามมาตรฐานเพื่อนำไปใช้กับงานบัญชีและภาษี

ความต้องการของพนักงาน (Staff)

- การเข้าสู่ระบบอย่างปลอดภัย (Secure Login) สามารถเข้าสู่ระบบด้วยบัญชีผู้ใช้ของตนเอง และเข้าถึงฟังก์ชันที่ได้รับอนุญาตเท่านั้น
- การบันทึกข้อมูลสินค้าเข้า-ออก (Stock In/Out) สามารถบันทึกการนำเข้าสินค้าและเบิกสินค้าออกจากคลังได้สะดวกและถูกต้อง
- การขายสินค้า (Sales/POS): บันทึกการขายสินค้าได้ โดยระบบจะตัดสต็อกให้อัตโนมัติ และสามารถออกบิลการขายได้
- การคืนสินค้า (Product Return): สามารถบันทึกการคืนสินค้าได้ง่าย พร้อมระบุสาเหตุการคืนเพื่อให้ข้อมูลถูกต้องครบถ้วน
- การตรวจสอบจำนวนสินค้าคงเหลือ (Inventory Checking) สามารถตรวจสอบจำนวนสินค้าปัจจุบันในคลังได้แบบเรียลไทม์
- การค้นหาสินค้า (Search & Filter) ต้องการระบบค้นหาและกรองสินค้าอย่างรวดเร็ว เพื่อให้ง่ายต่อการใช้งาน
- การดูรายงานเบื้องต้น (Basic Reports) สามารถดูรายงานเบื้องต้น เช่น รายงานสินค้าคงเหลือ หรือประวัติการทำรายการของตนเองได้
- การค้นหาสินค้า: ค้นหาสินค้าด้วยชื่อ รหัส หรือหมวดหมู่ได้อย่างรวดเร็ว
- การจัดการบิลและภาษี: ออกบิลย้อนหลังและใบกำกับภาษีที่ถูกต้องตามมาตรฐานได้



รูปภาพที่ 4.1.3 Use-Case Diagram ของระบบที่พัฒนา (Proposed System)

4.2 การออกแบบระบบ

การออกแบบระบบเป็นขั้นตอนสำคัญที่ต่อยอดมาจากการวิเคราะห์ระบบและความต้องการของผู้ใช้งาน โดยมีวัตถุประสงค์เพื่อสร้างโครงสร้างระบบที่สามารถแก้ไขข้อจำกัดของระบบปัจจุบัน และตอบสนองต่อความต้องการของผู้ใช้งานได้อย่างครบถ้วน การออกแบบระบบในโครงงานนี้ประกอบด้วย การออกแบบในหลายมิติ ได้แก่ การออกแบบสถาปัตยกรรมระบบ (System Architecture) การออกแบบโครงสร้างฐานข้อมูล (Database Design) การออกแบบกระบวนการทำงานของระบบ (Process Design) และการออกแบบส่วนติดต่อผู้ใช้งาน (User Interface Design)

4.2.1 การออกแบบสถาปัตยกรรมระบบ (System Architecture Design)

ระบบคลังสินค้าที่พัฒนาขึ้นถูกออกแบบให้ทำงานบนสถาปัตยกรรม Client-Server โดยผู้ใช้งานสามารถเข้าถึงระบบผ่านเว็บแอปพลิเคชัน (Web Application) ซึ่งเชื่อมต่อกับเซิร์ฟเวอร์ที่ประมวลผลและฐานข้อมูลกลาง การทำงานในลักษณะนี้ช่วยให้การจัดการข้อมูลทำได้แบบเรียลไทม์ และสามารถรองรับผู้ใช้งานหลายคนพร้อมกันได้

สถาปัตยกรรมของระบบถูกแบ่งออกเป็น 3 ชั้น (Three-tier Architecture) ได้แก่

1. Presentation Layer (Client Side)

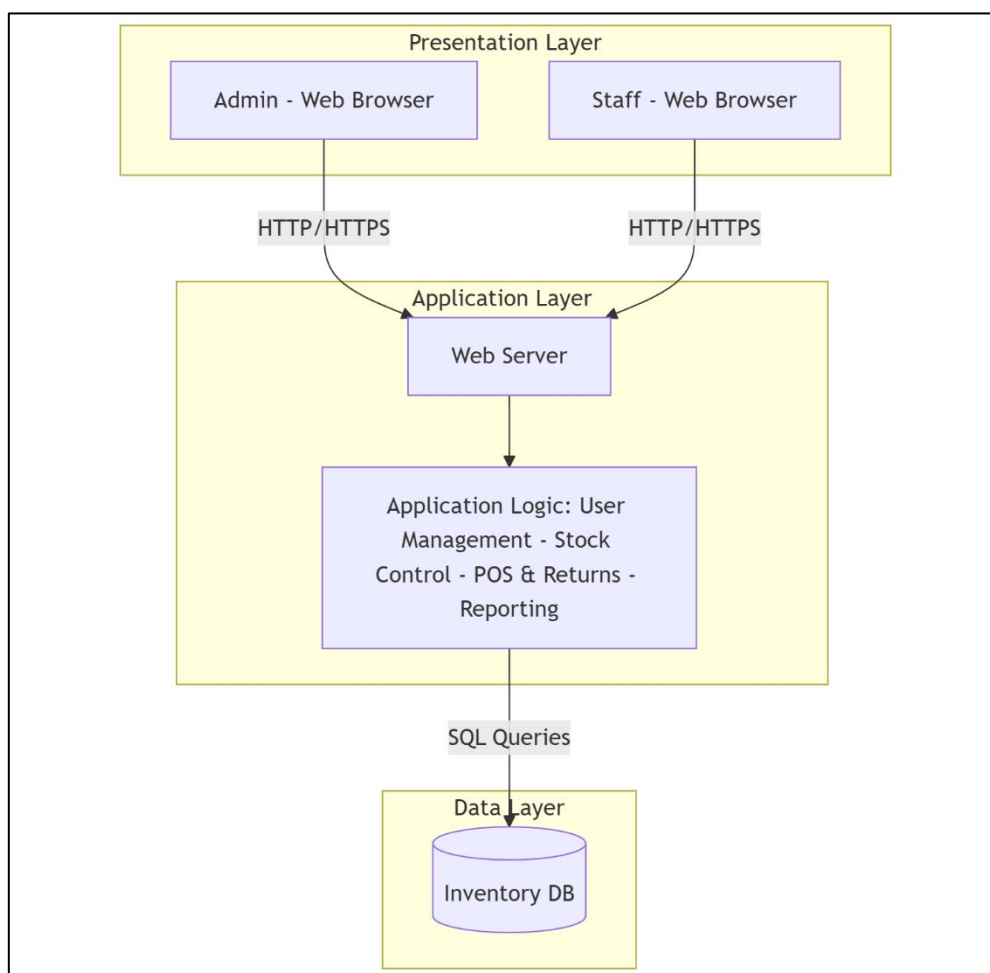
เป็นชั้นที่ผู้ใช้งานติดต่อกับระบบโดยตรง ผ่านเว็บเบราว์เซอร์ โดยมีทั้งผู้ดูแลระบบ (Admin) และพนักงาน (Staff) ซึ่งการเข้าถึงระบบจะผ่านโปรโตคอล HTTP/HTTPS

2. Application Layer (Server Side)

เป็นชั้นที่ทำหน้าที่ประมวลผลและควบคุมการทำงานของระบบ โดยมี Web Server เป็นตัวกลางรับ-ส่งข้อมูลระหว่างผู้ใช้งานกับระบบ และมี Business Logic ที่เกี่ยวข้อง เช่น การจัดการผู้ใช้งาน การควบคุมสต็อกสินค้า การขายสินค้า (POS) การคืนสินค้า และการสร้างรายงาน

3. Data Layer (Database Server)

เป็นชั้นที่ทำหน้าที่จัดเก็บและจัดการข้อมูลทั้งหมด เช่น ข้อมูลสินค้าหมวดหมู่สินค้า ผู้ใช้งาน ข้อมูลการขาย การคืนสินค้า การเคลื่อนไหวสต็อก รวมถึงบิลและใบกำกับภาษี โดยข้อมูลจะถูกจัดเก็บในฐานข้อมูล Inventory DB และมีการสื่อสารผ่าน SQL Queries

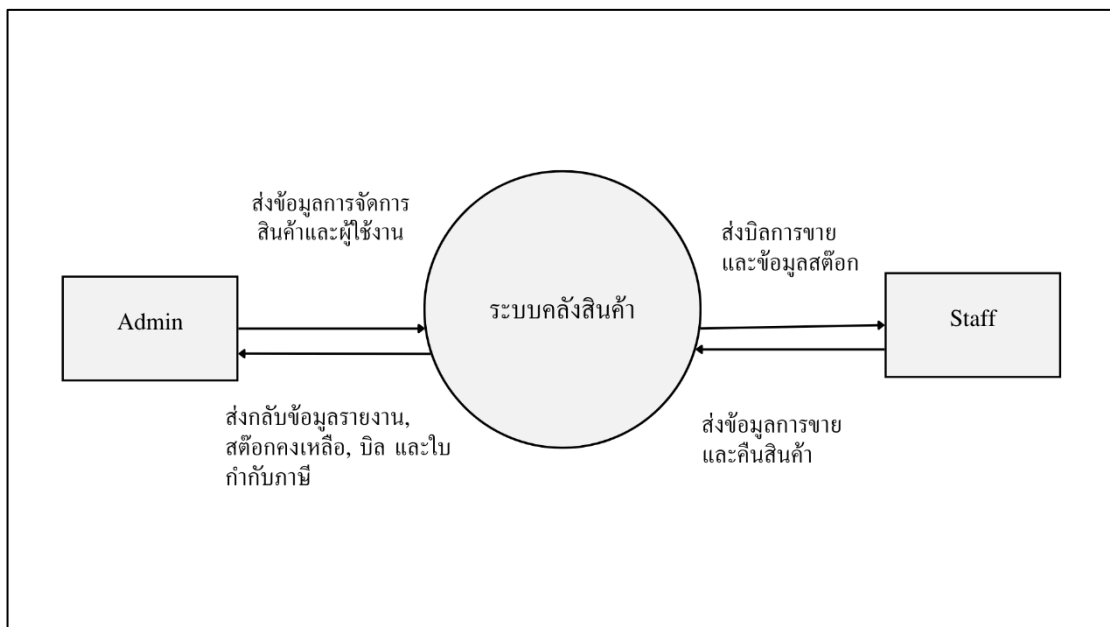


รูปที่ 4.2.1 แผนภาพสถาปัตยกรรมระบบ (System Architecture Design)

4.2.2 การออกแบบกระบวนการทำงานของระบบ (Process Design)

การออกแบบกระบวนการทำงานของระบบมีจุดประสงค์เพื่อแสดงให้เห็นถึงการไหลของข้อมูล (Data Flow) ระหว่างผู้ใช้งานกับระบบ รวมถึงการประมวลผลภายในระบบ เพื่อให้สามารถเข้าใจโครงสร้างการทำงานโดยรวมได้อย่างชัดเจน ในโครงงานนี้ได้ออกแบบกระบวนการทำงานโดยใช้ Data Flow Diagram (DFD) ซึ่งประกอบด้วย 2 ระดับ ได้แก่

1. Context Diagram เป็นการแสดงภาพรวมของระบบในมุมมองแบบ Black Box โดยระบุผู้ใช้งานหลัก (External Entities) ได้แก่ ผู้ดูแลระบบ (Admin) และพนักงาน (Staff) ที่มีการติดต่อกับระบบคลังสินค้า ระบบจะแสดงเพียงการแลกเปลี่ยนข้อมูลระหว่างผู้ใช้งานกับระบบโดยไม่ลงรายละเอียดภายใน เช่น Admin ส่งข้อมูลการจัดการสินค้าและพนักงานเข้าสู่ระบบ และสามารถรับรายงาน สต็อกคงเหลือ บิล และใบกำกับภาษีออกมาได้ ส่วน Staff จะส่งข้อมูลการขายและการคืนสินค้าเข้าสู่ระบบ และรับข้อมูลบิลการขายหรือจำนวนสินค้าคงเหลือกลับมา



รูปที่ 4.2.2 (1) Context Diagram ของระบบคลังสินค้า (Level 0)

2. Data Flow Diagram ระดับ 1 (Level 1 DFD)

Data Flow Diagram ระดับ 1 (Level 1 DFD) ของระบบคลังสินค้า เป็นการขยายรายละเอียดจาก Context Diagram เพื่อแสดง กระบวนการทำงานย่อยภายในระบบ และการเชื่อมโยงกับฐานข้อมูล ต่าง ๆ โดยระบุถึงการไหลของข้อมูลระหว่างผู้ใช้งานหลัก (Admin และ Staff) กับกระบวนการภายในระบบ ซึ่งช่วยให้เข้าใจบทบาท หน้าที่ และการประมวลผลของแต่ละส่วนได้ชัดเจนขึ้น

ระบบคลังสินค้ามีกระบวนการย่อยหลัก 5 กระบวนการ ได้แก่

1. จัดการสินค้า (Product Management)

Admin สามารถเพิ่ม แก้ไข หรือลบข้อมูลสินค้าและหมวดหมู่ สินค้า โดยข้อมูลจะถูกบันทึกหรืออัปเดตลงใน Product DB (D1)

2. ขายสินค้า (Sales / POS)

Staff หรือ Admin สามารถบันทึกการขายสินค้า ระบบจะ จัดเก็บข้อมูลลงใน Sales DB (D3) และทำการตัดสต็อกใน Stock Movement DB (D4) โดยอัตโนมัติ พร้อมทั้งออกบิลการ ขายส่งให้ผู้ใช้งาน

3. คืนสินค้า (Return Management):

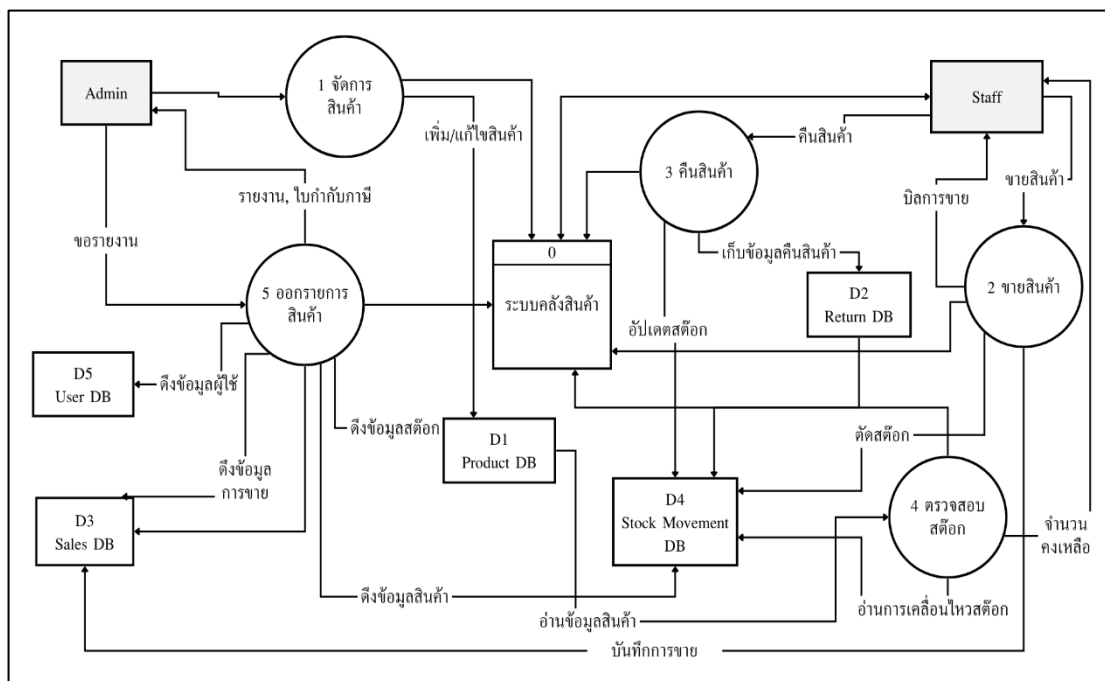
Staff สามารถบันทึกการคืนสินค้า ข้อมูลจะถูกจัดเก็บไว้ใน Return DB (D2) และระบบจะอัปเดตการเคลื่อนไหวสต็อกใน Stock Movement DB (D4)

ตรวจสอบสต็อก (Inventory Control)

4. Admin หรือ Staff สามารถตรวจสอบจำนวนสินค้าคงเหลือใน ระบบได้แบบเรียลไทม์ โดยระบบจะดึงข้อมูลจาก Product DB (D1) และ Stock Movement DB (D4) มาแสดงผล

5. ออกรายงาน (Reporting)

Admin สามารถขอรายงานการทำงานของระบบ เช่น รายงาน ยอดขาย รายงานสินค้าคงเหลือ และรายงานการคืนสินค้า โดย ระบบจะดึงข้อมูลจาก User DB (D5), Sales DB (D3), Return DB (D2), Product DB (D1) และ Stock Movement DB (D4) มาประมวลผลและนำเสนอในรูปแบบรายงานหรือใบกำกับ ภาษี



รูปที่ 4.2.2 (2) Data Flow Diagram ระดับ 1 ของระบบคลังสินค้า แสดงกระบวนการย่อยหลัก

5กระบวนการ

4.2.3 การออกแบบส่วนติดต่อผู้ใช้ (User Interface Design)

การออกแบบส่วนติดต่อผู้ใช้งาน (User Interface Design: UI) ของระบบ คลังสินค้า ได้ถูกออกแบบโดยยึดหลักความง่ายในการใช้งาน (User Friendly) ความชัดเจน (Clarity) และการทำงานได้อย่างมีประสิทธิภาพ (Efficiency) โดยเน้นให้ผู้ใช้งาน ทั้ง ผู้ดูแลระบบ (Admin) และ พนักงาน (Staff) สามารถทำงานได้สะดวก รวดเร็ว และ ถูกต้อง

หน้าจอของระบบถูกออกแบบให้มีโครงสร้างที่เรียบง่ายและสม่ำเสมอ (Consistency) เพื่อให้ผู้ใช้คุ้นเคยและใช้งานได้อย่างต่อเนื่อง โดยรองรับการทำงานผ่านเว็บแอปพลิเคชัน ที่สามารถเข้าถึงได้แบบเรียลไทม์

- ### 1. หน้าล็อกอิน (Login Page)

ใช้สำหรับการเข้าสู่ระบบ โดยผู้ใช้งานต้องกรอกอีเมลและรหัสผ่านเพื่อยืนยันสิทธิ์
การใช้งาน

- ## 2. หน้า Dashboard

แสดงภาพรวมการทำงานของระบบ เช่น การเคลื่อนไหวของสต็อก, กราฟ
ยอดขาย, สินค้าใกล้หมด และสรุปข้อมูลสำคัญ

- ### 3. หน้า POS (ขายสินค้า)

ออกแบบให้ Staff ใช้งานง่าย สามารถค้นหา เลือกสินค้า และบันทึกการขายได้ทันที ระบบจะตัดสต็อกอัตโนมัติ และออกบิลการขายให้ลูกค้า

4. หน้าชำระเงิน (Payment)

รองรับการชำระเงินหลายช่องทาง เช่น เงินสด โอน PromptPay หรือบัตรเครดิต พร้อมคำนวณยอดรวมให้อัตโนมัติ

5. หน้าจัดการสินค้า (Product Management)

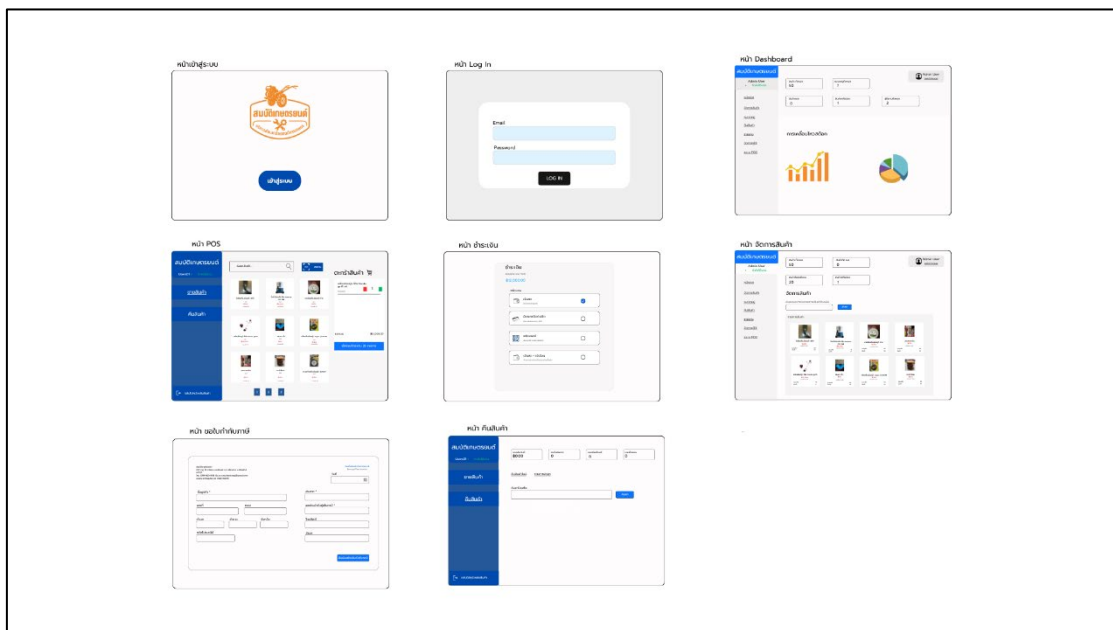
Admin สามารถเพิ่ม แก้ไข หรือลบข้อมูลสินค้า และกำหนดจำนวนขั้นต่ำของสินค้าเพื่อใช้ในการแจ้งเตือนสินค้าใกล้หมด

6. หน้าคืนสินค้า (Return Management)

ใช้สำหรับบันทึกการคืนสินค้า ระบุสาเหตุ และอัปเดตข้อมูลเข้าสู่ฐานข้อมูลคืนสินค้า พร้อมทั้งอัปเดตสต็อกสินค้า

7. หน้าออกใบกำกับภาษี (Invoice / Tax Report)

รองรับการออกบิลหรือใบกำกับภาษีที่สมบูรณ์ เพื่อให้ร้านค้าสามารถใช้งานได้จริงและรองรับงานด้านบัญชี



รูปที่ 4.2.3 ตัวอย่างการออกแบบส่วนติดต่อผู้ใช้งาน (User Interface Design)

4.3 การออกแบบฐานข้อมูล (Database Design)

การออกแบบฐานข้อมูลของระบบคลังสินค้า มีวัตถุประสงค์เพื่อจัดเก็บข้อมูลอย่างเป็นระบบ ระบุความสัมพันธ์ของข้อมูล และสนับสนุนการทำงานของฟังก์ชันต่าง ๆ เช่น การจัดการสินค้า การขาย การคืนสินค้า การตรวจสอบสต็อก และการออกรายงาน ฐานข้อมูลถูกออกแบบให้อยู่ในรูปแบบเชิงสัมพันธ์ (Relational Database) เพื่อความถูกต้อง ความสม่ำเสมอ และความปลอดภัยของข้อมูล

4.3.1 การกำหนดตารางข้อมูล (Database Tables)

ระบบนี้ประกอบด้วยตารางหลักที่สำคัญ ได้แก่

1. ตาราง Users – เก็บข้อมูลผู้ใช้งานระบบ (Admin, Staff)

- user_id (PK) : รหัสผู้ใช้
- name : ชื่อผู้ใช้
- email : อีเมล
- password : รหัสผ่าน (เข้ารหัส)
- role : บทบาท (admin/staff)
- created_at, updated_at : วัน-เวลาที่สร้างและปรับปรุง

2. ตาราง Products – เก็บข้อมูลสินค้าคงคลัง

- product_id (PK) : รหัสสินค้า
- name : ชื่อสินค้า
- sku : รหัสสินค้า (Stock Keeping Unit)
- category_id (FK) : หมวดหมู่สินค้า
- price : ราคาขาย
- quantity : จำนวนคงเหลือ
- min_stock : จำนวนขั้นต่ำที่ควรมี
- is_active : สถานะการใช้งานสินค้า

3. ตาราง Categories – เก็บหมวดหมู่สินค้า

- category_id (PK) : รหัสหมวดหมู่
- category_name : ชื่อหมวดหมู่

4. ตาราง Sales – เก็บข้อมูลการขาย

- sale_id (PK) : รหัสการขาย
- user_id (FK) : ผู้ทำการขาย
- sale_date : วันที่ขาย
- total_amount : ยอดรวมก่อนภาษี
- tax_amount : ภาษี
- grand_total : ยอดรวมสุทธิ
- payment_method : วิธีการชำระเงิน
- received_amount, change_amount : เงินรับ-เงิน
ทอน

5. ตาราง Sale_Items – รายการสินค้าในแต่ละการขาย

- sale_item_id (PK) : รหัสรายการขาย
- sale_id (FK) : รหัสการขาย
- product_id (FK) : รหัสสินค้า
- quantity : จำนวนที่ขาย
- unit_price : ราคาต่อหน่วย
- total_price : ราคารวม

6. ตาราง Returns – เก็บข้อมูลการคืนสินค้า

- return_id (PK) : รหัสการคืน
- user_id (FK) : ผู้ทำการคืน
- return_number : หมายเลขการคืน
- total_return_amount : มูลค่ารวมที่คืน
- reason : เหตุผลในการคืน
- status : สถานะการคืน
(pending/approved/completed)

7. ตาราง Return_Items – รายการสินค้าที่คืน

- return_item_id (PK) : รหัสรายการคืน
- return_id (FK) : รหัสการคืนสินค้า

- product_id (FK) : รหัสสินค้า
- quantity : จำนวนที่คืน
- condition_note : หมายเหตุ/สภาพสินค้า

8. ตาราง Stock_Movements – บันทึกการเคลื่อนไหวสต็อก

- movement_id (PK) : รหัสการเคลื่อนไหว
- product_id (FK) : รหัสสินค้า
- user_id (FK) : ผู้ทำรายการ
- type : ประเภทการเคลื่อนไหว (in, out, return_in, adjustment)
- quantity : จำนวนที่เปลี่ยนแปลง
- new_quantity : จำนวนคงเหลือใหม่
- lot_number, expiry_date : ข้อมูลล็อตสินค้าและวันหมดอายุ

9. ตาราง Receipts – เก็บข้อมูลใบเสร็จ/ใบกำกับภาษี

- receipt_id (PK) : รหัสใบเสร็จ
- receipt_number : เลขที่ใบเสร็จ
- sale_id (FK) : รหัสการขายที่เกี่ยวข้อง
- user_id (FK) : ผู้ทำรายการ
- total_amount, tax_amount, grand_total : ยอดรวมภาษี และยอดสุทธิ

10. ตาราง Receipt_Items – รายการสินค้าในใบเสร็จ

- receipt_item_id (PK) : รหัสรายการในใบเสร็จ
- receipt_id (FK) : รหัสใบเสร็จ
- product_id (FK) : รหัสสินค้า
- quantity : จำนวนสินค้า
- unit_price : ราคาต่อหน่วย
- total_price : ราคารวม

4.3.2 การออกแบบความสัมพันธ์ (ER-Diagram)

จากการออกแบบระบบคลังสินค้า ได้ทำการสร้างแบบจำลองความสัมพันธ์ระหว่างตาราง (Entity–Relationship Diagram: ER-Diagram) เพื่อแสดงให้เห็นโครงสร้างข้อมูลและความสัมพันธ์ของแต่ละตารางที่ใช้ในระบบ

ER-Diagram ของระบบนี้ประกอบด้วยตารางหลักที่สำคัญ ได้แก่

1. Users

ใช้เก็บข้อมูลผู้ใช้งานระบบ ทั้งผู้ดูแลระบบ (Admin) และพนักงาน (Staff) โดยมีการเชื่อมโยงกับตารางการขาย (Sales), ตารางการคืนสินค้า (Returns) และการเคลื่อนไหวของสต็อก (Stock_Movements)

2. Products

เก็บข้อมูลรายละเอียดของสินค้า เช่น ชื่อสินค้า รหัสสินค้า (SKU) ราคาต้นทุน ราคาขาย และจำนวนคงเหลือ เชื่อมโยงกับตาราง Categories, Sale_Items, Receipt_Items, Stock_Movements และ Returns

3. Categories

ใช้เก็บหมวดหมู่สินค้า เพื่อให้ง่ายต่อการจัดการและการค้นหา เชื่อมโยงกับ Products

4. Sales

เก็บข้อมูลการขายแต่ละครั้ง เช่น วันที่ขาย ยอดรวม และผู้ใช้ที่ทำการขาย เชื่อมโยงกับ Sale_Items และ Receipt_Items

5. Sale_Items

รายการสินค้าที่อยู่ในการขายแต่ละครั้ง ระบุจำนวนและราคาสินค้า เชื่อมโยงกับ Sales และ Products

6. Returns

เก็บข้อมูลการคืนสินค้า เช่น วันที่คืน เหตุผล และมูลค่าสินค้าที่คืน เชื่อมโยงกับ Products และ Users

7. Stock_Movements

เก็บข้อมูลการเคลื่อนไหวของสินค้า เช่น การรับเข้า การขายออก การคืนสินค้า หรือการปรับปรุงสต็อก เชื่อมโยงกับ Products และ Users

8. Receipt_Items

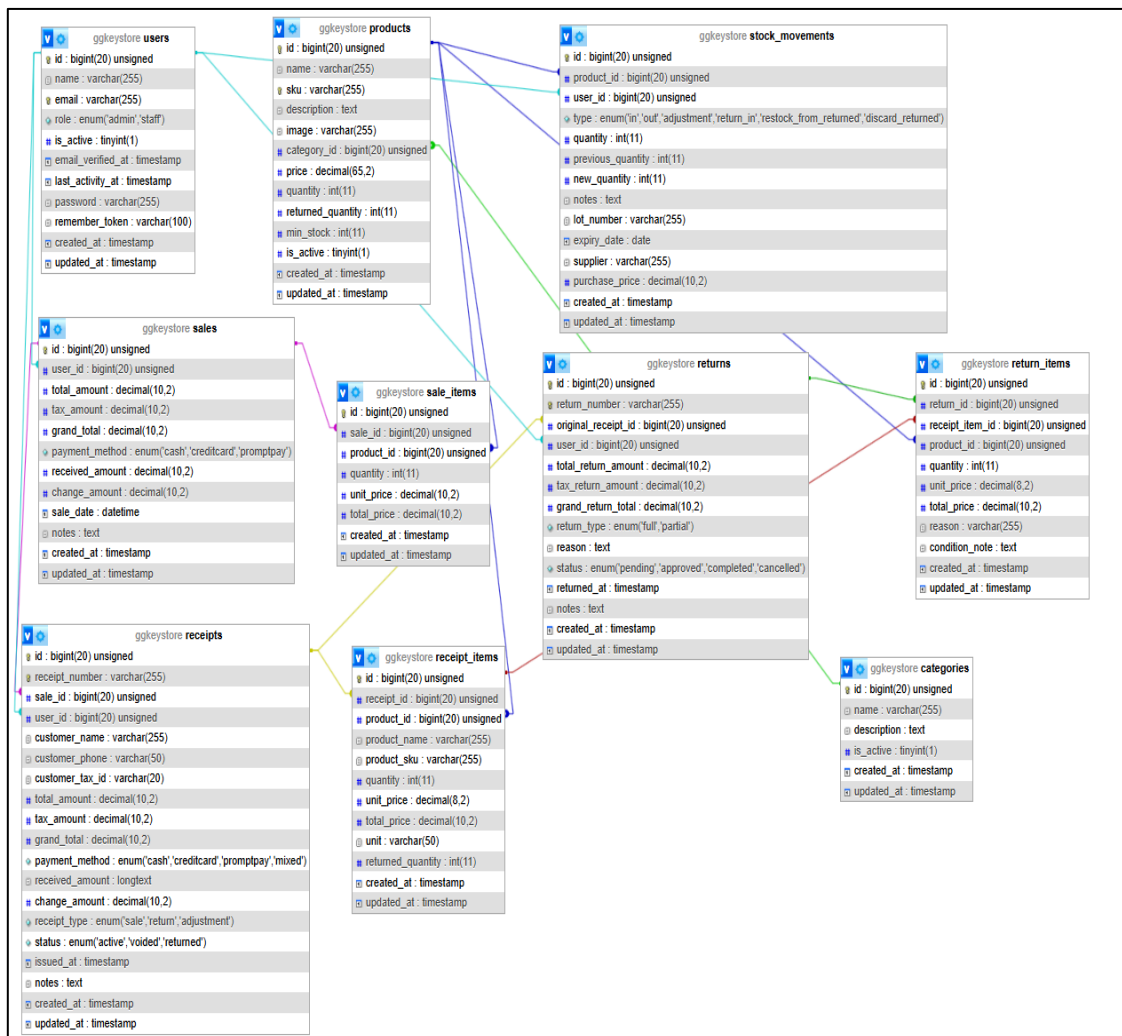
รายการสินค้าที่ปรากฏในใบเสร็จหรือใบกำกับภาษี เชื่อมโยงกับ Products และ Sales

9. Receipts

ใช้เก็บข้อมูลใบเสร็จรับเงินหรือใบกำกับภาษี เช่น หมายเลขใบเสร็จ วันที่ออกใบเสร็จ มูลค่ารวม ภาษี และยอดสุทธิ เชื่อมโยงกับตาราง Sales และ Receipt_Items

10. Return_Items

ใช้เก็บรายละเอียดสินค้าที่ถูกคืนในแต่ละการทำรายการคืนสินค้า ระบุจำนวนสินค้า สาเหตุ และสภาพสินค้า เชื่อมโยงกับตาราง Returns และ Products



รูปที่ 4.3.2 แสดง ER-Diagram ของระบบคลังสินค้า

4.4 การรักษาความปลอดภัยและการจัดการผู้ใช้ในระบบ (Security and User Management Design)

ความปลอดภัยของข้อมูลและการจัดการผู้ใช้งานเป็นองค์ประกอบสำคัญของระบบคลังสินค้า เนื่องจากข้อมูลที่จัดเก็บ เช่น ข้อมูลสินค้า ประวัติการเคลื่อนไหว และบัญชีผู้ใช้ มีความสำคัญต่อธุรกิจ SMEs การออกแบบระบบจึงต้องคำนึงถึงมาตรการรักษาความปลอดภัย (Security) และ การกำหนดสิทธิ์ของผู้ใช้งาน (User Management) อย่างเป็นระบบ โดยเทคโนโลยีที่นำมาใช้ในโครงงานประกอบด้วย

4.4.1 การยืนยันตัวตนและการอนุญาต (Authentication and Authorization)

- ในระบบนี้ใช้ Laravel Authentication System ซึ่งเป็นฟีเจอร์สำเร็จรูปของ Laravel สำหรับการเข้าสู่ระบบและการจัดการ Session ของผู้ใช้ โดยรองรับการเข้ารหัสผ่านอัตโนมัติด้วย bcrypt
 - ใช้แนวทาง Role-Based Access Control (RBAC) โดยกำหนดบทบาทหลัก 2 ระดับ ได้แก่
 1. ผู้ดูแลระบบ (Admin): มีสิทธิ์จัดการสินค้า หยอดหมู่ ผู้ใช้งาน และเข้าถึงรายงานทั้งหมด
 2. เจ้าหน้าที่ (Staff): จำกัดสิทธิ์เฉพาะการบันทึกการนำเข้า-เบิกออกสินค้า และตรวจสอบข้อมูลที่ได้รับอนุญาต
1. ฟังก์ชันการกำหนดสิทธิ์ดำเนินการผ่าน Middleware ของ Laravel ซึ่งจะตรวจสอบสิทธิ์ผู้ใช้งานก่อนเข้าถึงแต่ละฟังก์ชัน

4.4.2 การเข้ารหัสและการปกป้องข้อมูล (Encryption and Data Protection)

- รหัสผ่านผู้ใช้งานถูกเข้ารหัสด้วย Hashing Algorithm (bcrypt) ที่ Laravel จัดเตรียมไว้ ทำให้ไม่สามารถกู้คืนรหัสผ่านจากฐานข้อมูลได้โดยตรง
- การสื่อสารระหว่าง Frontend (React + Inertia.js) และ Backend (Laravel) ใช้ HTTPS/SSL เพื่อเข้ารหัสข้อมูลระหว่างผู้ใช้งานกับเซิร์ฟเวอร์

4.4.3 การบันทึกและติดตามกิจกรรม (Logging and Monitoring)

- Laravel มีระบบ Activity Log และการบันทึก Error Log ผ่านไฟล์ log ซึ่งช่วยตรวจสอบกิจกรรมของผู้ใช้งาน เช่น การเข้าสู่ระบบ การแก้ไขข้อมูลสินค้า หรือการปรับสิทธิ์
- สามารถนำข้อมูล Log มาวิเคราะห์ความปลอดภัย และช่วยระบุสาเหตุเมื่อเกิดปัญหา

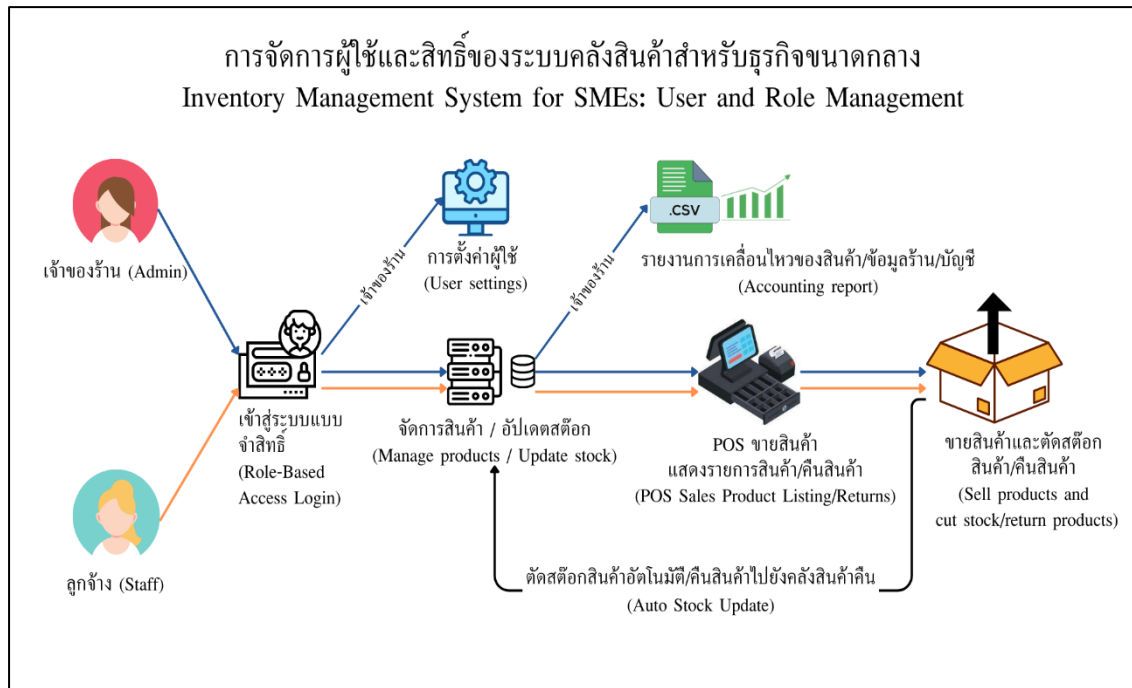
4.4.4 การป้องกันช่องโหว่ระบบ (System Vulnerability Protection)

Laravel มีการป้องกันภัยคุกคามที่พบบ่อยโดยค่าเริ่มต้น เช่น

- CSRF Protection: ใช้ CSRF Token เพื่อป้องกันการโจมตีแบบ Cross-Site Request Forgery
- SQL Injection Protection: ใช้ Query Builder และ Eloquent ORM ที่รองรับการทำงานแบบ Parameterized Query
- XSS Protection: Laravel มีฟังก์ชัน {{ }} สำหรับ Escape Output เพื่อป้องกัน Cross-Site Scripting

4.4.5 การจัดการผู้ใช้และสิทธิ์ (User and Role Management)

- ระบบรองรับการ สร้าง แก้ไข และลบผู้ใช้งาน ผ่าน Dashboard ของผู้ดูแลระบบ
- สามารถเปิด-ปิดการใช้งานบัญชี (Activate/Deactivate) ได้โดยไม่ต้องลบข้อมูล เพื่อความยืดหยุ่นในการจัดการบุคลากร
- การกำหนดบทบาทผู้ใช้งาน (Admin/Staff) ทำให้การเข้าถึงข้อมูล และฟังก์ชันของระบบมีความชัดเจน ป้องกันการเข้าถึงเกินสิทธิ์



รูปที่ 4.4.5 ระบบคลังสินค้าสำหรับ SMEs: การจัดการผู้ใช้และสิทธิ์