ระบบคลังพัสดุ/ครุภัณฑ์

Inventory Management

นายอิศรานุกิต อุ่นใจ รหัส 6806022510441 Sec2

นางสาวศิริรัตน์ กองจินดา รหัส 6806022510483 Sec2

นางสาวเพ็ญนภา เฉลิมลาภ รหัส 6806022510467 Sec2

นายธนเทพ เปรมมณี รหัส 6806022510343 Sec2

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยพระจอมเกล้าพระนครเหนือ

ปีการศึกษา 2568

คำนำ

การจัดทำโปรเจค “คลังพัสดุ/ครุภัณฑ์” นี้เป็นส่วนหนึ่งของรายวิชา Computer Programming ของหลักสูตรวิศวกรรมบัณฑิต สาขาวิศวกรรมสารสนเทศและเครือข่ายภาควิชาเทคโนโลยีคณะเทคโนโลยีและการจัดการอุตสาหกรรมมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ วิทยาเขตปราจีนบุรี เพื่อให้นักศึกษาได้นำความรู้ที่เรียนมาทั้งหมดมาประยุกต์ใช้ในการพัฒนาโปรแกรมที่สามารถนำไปใช้ได้จริง โดยเน้นการออกแบบและเขียนโปรแกรมในภาษา Python ซึ่งเป็นภาษาที่เรียนมาในวิชา Computer Programming โดยโปรเจคนี้จะช่วยการวิเคราะห์และการแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมในการประกอบอาชีพด้านวิศวกรรมสารสนเทศและเครือข่ายในอนาคต

คณะผู้จัดทำหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้อ่าน หรือนักเรียน นักศึกษา ที่กำลังหาข้อมูลเรื่องนี้อยู่ หากมีข้อแนะนำหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับและขอ อภัยมา ณ ที่นี้ด้วย

สารบัญ

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 วัตถุประสงค์ของโครงงาน

1.1.1 เพื่อพัฒนาระบบคลังพัสดุ/ครุภัณฑ์ ได้อย่างมีประสิทธิภาพ

1.1.2 เพื่อฝึกฝนทักษะการเขียนโปรแกรม Python

1.1.3 เพื่อเรียนรู้วิธีการจัดการข้อมูลและไฟล์

1.1.4 เพื่อเรียนรู้การทำงานร่วมกันเป็นทีม

1.2 ขอบเขตของโครงงาน

1.1.5 ระบบคลังพัสดุ/ครุภัณฑ์ มีฟังก์ชันพื้นฐานทั้งหมด 12 ฟังก์ชัน ได้แก่

1. สร้างรหัส ID อัตโนมัติ
2. ใช้ print_line(char='-', length=90) สำหรับพิมพ์เส้นขั้น
3. คืนค่าเวลาข้อมูลปัจจุบัน
4. เพิ่มข้อมูลสินค้าเข้าไปในคลัง
5. แก้ไขข้อมูลสินค้า
6. ลบสินค้าออกจากระบบ
7. โอนย้ายสินค้าไปยัง Location อื่น
8. กำจัดสินค้าทิ้ง
9. แสดงสินค้าใน Inventory
10. ฟังก์ชันสำหรับบันทึกการเคลื่อนไหวของสินค้า
11. แสดงรายงาน Log + Daily Summary
12. ฟังก์ชันหลัก ทำหน้าที่เป็น เมนูระบบ

1.1.6 ระบบคลังพัสดุ/ครุภัณฑ์ ประกอบไปด้วย 3 หมวดหมู่ด้วยกัน ได้แก่

1. เก็บข้อมูลสินค้า
    - 1.1 รหัสของสินค้า
    - 1.2 ชื่อสินค้าหรือรายการ
    - 1.3 รหัสหมวดหมู่ของสินค้า
    - 1. ยี่ห้อหรือรุ่นของสินค้า
    - 1.5 จำนวนสินค้าทั้งหมดที่มีอยู่
    - 1.6 หน่วยสินค้า เช่น กล่อง ชิ้น

1.7 ราคาต่อหน่วย

1.8 สถานะของสินค้า

1.9 สถานที่เก็บสินค้าปัจจุบัน

2. เก็บข้อมูลหมวดหมู่สินค้า

2.1 ชื่อหมวดหมู่สินค้า เช่น เฟอร์นิเจอร์ เครื่องเขียน เครื่องใช้ไฟฟ้า

3. เก็บ Log การเคลื่อนไหว

3.1 รหัสการเคลื่อนไหว

3.2 รหัสหมวดหมู่สินค้าที่ถูกกระทำ

3.3 รหัสสินค้าที่ถูกกระทำ

3.4 ประเภทการกระทำ เช่น Add Update Delete Transfer Dispose

3.5 จำนวนที่มีการเคลื่อนไหว

3.6 สถานนีต้นทาง

3.7 สถานีปลายทาง

3.8 ผู้ดำเนินการ

3.9 วันและเวลาที่ทำรายการ

1.2.3 ระบบคลังพัสดุ/ครุภัณฑ์ มีการจัดเก็บข้อมูลไว้ใน Text File ซึ่งมี Item_ID Category_ID Move_ID

1.2.4 ระบบคลังพัสดุ/ครุภัณฑ์ จะมีเมนูเพื่อให้ผู้ใช้สามารถเลือกดำเนินการได้

1.3 ประโยชน์ที่ได้รับ

1.3.1 พัฒนาระบบที่สามารถ Update คลังพัสดุ/ครุภัณฑ์ ได้อย่างมีประสิทธิภาพ

1.3.2 พัฒนาทักษะการเขียนโปรแกรม

1.3.3 เรียนรู้การจัดการข้อมูลและไฟล์

1.3.4 เรียนรู้การทำงานร่วมกันเป็นทีม

1.4 เครื่องมือที่คาดว่าจะต้องใช้

1.4.1 VS Code

1.4.2 Microsoft Office

บทที่ 2

ระบบคลังพัสดุ/ครุภัณฑ์

2.1 ตาราง Items (เก็บรายการพัสดุ/ครุภัณฑ์)

ตาราง Items ประกอบไปด้วย 9 ฟิลด์หลัก ซึ่งรายละเอียดและความสำคัญดังนี้

ลำดับ	Field name	Data type	ตัวอย่าง	หมายเหตุ
1	Item_id	Varchar(30)	ITM-0001	Primary_key
2	Item_name	Varchar(100)	โต๊ะทำงาน	ชื่อพัสดุ
3	Category_id	Varchar(20)	CAT-001	FK ⟶ Categories
4	Brand_model	Varchar(100)	Index / WoodDesk A	ยี่ห้อ/รุ่น
5	Quantity	INT	50	จำนวนคงเหลือ
6	unit	Varchar(20)	ตัว	หน่วยนับ
7	Unitprice	Decimal(10,2)	2500.00	ราคาต่อหน่วย
8	Status	ENUM('Available','In Use','Disposed')	Available	สถานะ
9	location	Varchar(100)	อาคาร A ชั้น 2	สถานที่จัดเก็บITM-0001

ตารางที่ 2-1 ตารางเก็บรายการพัสดุ/ครุภัณฑ์

2.1.1 Item_id Primary Key

Item_id เป็นรหัสประจำสินค้าในการอ้างอิงค์สินค้าในทุกฟังก์ชัน เช่น add_item() , update_item() , delete_item() , และ log_movement() รหัสต้องไม่ซ้ำกันใช้เป็นตัวเชื่อมกับตาราง movements เพื่อดูประวัติการเคลื่อไหวสินค้านั้นๆ

2.1.2 Item_name ชื่อพัสดุ

Item_name เป็นชื่อเรียกของพัสดุหรือครุภัณฑ์ ใช้การแสดงหน้ารายการ view_items() และในรายงาน report() ผู้ใช้สามารถกำหนดเองได้ตอนเพิ่มข้อมูล add_item

2.1.3 Category_id Foreign Key

Category_id เป็น Foreign Key เชื่อมโยงกับตาราง categories เพื่อบอกว่าสินค้านี้อยู่ในหมวดหมใด เช่น เฟอร์นิเจอร์ อุปกรณ์คอมพิวเตอร์ ถ้า Category ยังไม่มี ระบบจะสร้างใหม่อัตโนมัติในฟังก์ชัน add_item

2.1.4 Brand_model ยี่ห้อ/รุ่น

Brand_model เอาไว้ระบุยี่ห้อหรือของพัสดุ เช่น Dell , Index , HP ใช้ระบุตัวสิค้าในกรณีที่ชื่อสินค้าซ้ำกันไม่มีผลต่อจำนวน แต่ช่วยให้รายละเอียดงานละเอียดขึ้น

2.1.5 Quantity จำนวนคงเหลือ

Quantity มีหน้าที่แสดงจำนวนสินค้าที่ยังมีอยู่ในสต็อกปัจจุบัน จะถูกปรับอัตโนมัติเมื่อมีการ add , update , delete , transfer หรือ dispose ใช้ในกาคำนวณยอดคงเหลือในรายงาน report

2.1.6 Unit หน่วยนับ

Unit เป็นหน่วยนับของพัสดุ เช่น ตัว , เครื่อง , กล่อง หรือ ชุด ใช้แสดงผลหน้ารายงานหรือในระบบแสดงข้อมูล ไม่ได้ใช้ในการคำนวณ แต่มีผลต่อการเข้าใจในปริมาณ

2.1.7 Unitprice ราคาต่อหน่วย

Unitprice คือราคาต่อหน่วยของพัสดุหรือสินค้านั้นใช้คำนวณมูลค่าทั้งหมดของสินค้ารวม (Quantity * Unitprice) ได้ถ้าต้องการทำรายงานมูลค่า ใช้ในฟังก์ชัน update_item() เพื่อแก้ไขราคา

2.1.8 Status สถานะ

Status คือ บอกสถานะของสินค้าปัจจุบัน เช่น Available = พร้อมใช้งานในคลัง , In Use = ถูกนไปใช้งานหรืออาจจะย้ายสถานที่ , Dispose = ถูกจำหน่วยหรือทิ้งแล้ว ฟังก์ชัน dispose_item จะเปลี่ยนค่านี้เป็น Dispose อัตโนมัติเมื่อของหมด

2.1.9 Location สถานที่จัดเก็บ

Location เอาไว้ระบุสถานที่จัดเก็บสินค้าปัจจุบัน เช่น อาคาร ห้อง ชั้น หรือคลัง ใช้ในการฟังก์ชัน transfer_item เพื่อย้ายพัสดุจากที่หนึ่งไปอีกที่หนึ่ง เมื่อมีการย้ายจะเก็บ From_Location และ To_Location ไว้ในตาราง movements

2.2 ตาราง Categories หมวดหมู่พัสดุ/ครุภัณฑ์

ตาราง Categories หมวดหมู่พัสดุ/ครุภัณฑ์ ประกอบไปด้วย 2 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายเอียดและความสำคัญต่อไปนี้

ลำดับ	Field name	Data type	ตัวอย่าง	หมายเหตุ
1	Category_ID	VARCHAR(20)	CAT-001	Primary Key
2	Category_Name	VARCHAR(100)	เฟอร์นิเจอร์สำนักงาน	ชื่อหมวดหมู่

ตารางที่ 2-2 ตารางหมวดหมู่พัสดุ/ครุภัณฑ์

2.2.1 Category_ID Primary Key

Category_ID คือเป็น Primary Key ของตารางนี้ใช้ระบุรหัสหมวดหมู่สินค้าไม่ให้ซ้ำกัน ใช้เป็นตัวเชื่อมกับฟิลด์ Category_id ในตาราง Items ให้รู้ว่าสินค้าแต่ละรายการอยู่ในหมวดหมู่ใด ตัวอย่างสินค้าเช่น สินค้าที่มี Category_ID = CAT-001 จะหมายถึงสินค้านั้นอยู่ในหมวดหมู่ เฟอร์นิเจอร์สำนักงาน ในโค้ด Python จะถูกใช้ตอนเพิ่มสินค้า add_item ถ้าผู้ใช้กรอก catgory_id ที่ยังไม่มีในระบบจะสามารถเพิ่มหมวดหมู่ใหม่ได้โดย add_category

2.2.2 Category_Name ชื่อหมวดหมู่

Category_Name เป็นชื่อของหมวดหมู่ที่อ่านเข้าใจง่าย เช่น คอมพิวเตอร์ , เครื่องใช้สำนักงาน เครื่องมือช่าง เป็นต้น ใช้เพื่ออธิบายว่า Catgory_ID นั้นหมายถึงอะไร ในโค้ด Python ใช้สำหรับการอ้างอิงเมื่อผู้ใช้ต้องการดูรายงานหรือเพิ่มสินค้าใหม่ในหมวดหมู่เดิม ในระบบแสดงผล เช่นรายงาน report จะแสดงชื่อหมวดหมู่แทนรหัสเพื่อให้ผู้ใช้เข้าใจง่าย

2.3 ตาราง Movements บันทึกการเคลื่อนย้าย/การทำรายการ

ตาราง Movements ประกอบด้วย 8 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญดังต่อไปนี้

ลำดับ	Field name	Data type	ตัวอย่าง	หมายเหตุ
1	Move_ID	VARCHAR(30)	MV-2025-001	Primary Key
2	Item_ID	VARCHAR(30)	ITM-0001	FK ⟶ Items
3	Action_Type	ENUM('Add','Update','Delete','View','Transfer','Dispose')	Add	ประเภทการทำงาน/เคลื่อนย้าย
4	Quantity	INT	10	จำนวนที่ทำรายการ
5	From_Location	VARCHAR(100)	คลังกลาง	ต้นทาง (ถ้ามี)
6	To_Location	VARCHAR(100)	ห้องประชุม 1	ปลายทาง (ถ้ามี)
7	Action_By	VARCHAR(100)	Admin	ผู้ทำรายการ
8	Action_Date	DATETIME	2025-09-23 10:15:00	วันเวลาที่ทำ

ตารางที่ 2-3 ตารางบันทึกการเคลื่อย้าย/การทำรายการ

2.3.1 Move_ID Primary Key

Move_ID เป็น Primary Key ของตารางนี้ ใช้รหัสประจำแต่ละ การเคลื่อนไหว รหัสนี้จะไม่ซ้ำกัน เช่น MV-2025-001 , MV-2025-002 ใช้ระบุแต่ละ Log ว่ามาจากการทำงานใด เช่น เพิ่ม , ลบ , ย้ายหรืออัปเดตสินค้า ในโค้ด Python จะถูกสร้างโดยอัตโนมัติด้วยฟังก์ชัน generate_id(“MV”) ก่อนบันทึกลงใน log list ชื่อ movements

2.3.2 Item_ID

Item_ID เป็น Foreign Key เชื่อมโยงกับฟิลด์ Item_ID ในตาราง Items ใช้ระบุว่าสินค้าชิ้นใดเป็นเป้าหมายของการเคลื่อนไหว เช่น ITM-0001 อาจจะเป็น โต๊ะทำงานไม้ แสดงว่าการกระทำนี้เกิดขึ้นกับโต๊ะตัวนั้น โค้ด Python จะอ้างถึงสินค้าที่มีการเพิ่ม add_item ย้าย transfer_item หรือทิ้ง dispose_item

2.3.3 Action_Type ประเภทการทำงาน/เคลื่อนย้าย

Action_Type เก็บประเภทของการกระทำ ที่เกิดขึ้นในระบบ เช่น add = เพิ่มสินค้าใหม่เข้าคลัง update = ปรับปรุงข้อมูล เช่น จำนวน/สถานที่ delete = ลบข้อมูลสินค้าออกจากระบบ view = การดูรายละเอียดสินค้า transfer = โอนย้ายพัสดุจากที่หนึ่งไปอีกที่หนึ่ง dispose = จำหน่าย/ทิ้งพัสดุ ในโค้ด Python จะบันทึกค่า Action_Type นี้ลงไปใน Log ทุกครั้งที่มีการเรียกใช้ฟังก์ชันใดๆที่เกี่ยวกับพัสดุ

2.3.4 Quantity จำนวณที่ทำรายการ

Quantity เก็บจำนวนพัสดุที่เกี่ยวข้องกับการกระทำครั้งนั้น เช่น ถ้า add = หมายถึงจำนวนที่เพิ่งเข้ามา ถ้า transfer = หมายถึงจำนวนที่ย้ายออกจากต้นทาง ถ้า dispose = จำนวนที่จำหน่ายทิ้ง ในโค้ด Python ฟิลด์นี้จะรับค่าจาก input ของผู้ใช้ในแต่ละคำสั่ง

2.3.5 From_Location ต้นทาง

From_Location แสดงสถานที่ต้นทางก่อนการเคลื่อนไหวใช้เมื่อเกิดการย้าย transfer หรือจำหน่าย dispose ถ้าเป็นการ add ครั้งแรก มักจะเป็นเครื่องหมาย “-” หรือเรียกว่า ว่าง ในโค้ด Python จะถูกกำหนดในฟังก์ชัน transfer_item หรือ dispose_item

2.3.6 To_Location ปลายทาง

To_Location แสดงสถานที่ปลายทางหลังการเคลื่อนไหวใช้ในกรณีการโอนย้ายพัสดุหรือเพิ่มสินค้า ถ้าเป็น dispose จะว่าง เพราะของถูกทิ้งไปแล้วโค้ด Python จะอัปเดตฟิลด์นี้เมื่อมีการย้ายพัสดุในระบบ

2.3.7 Action_By ผู้ทำรายการ

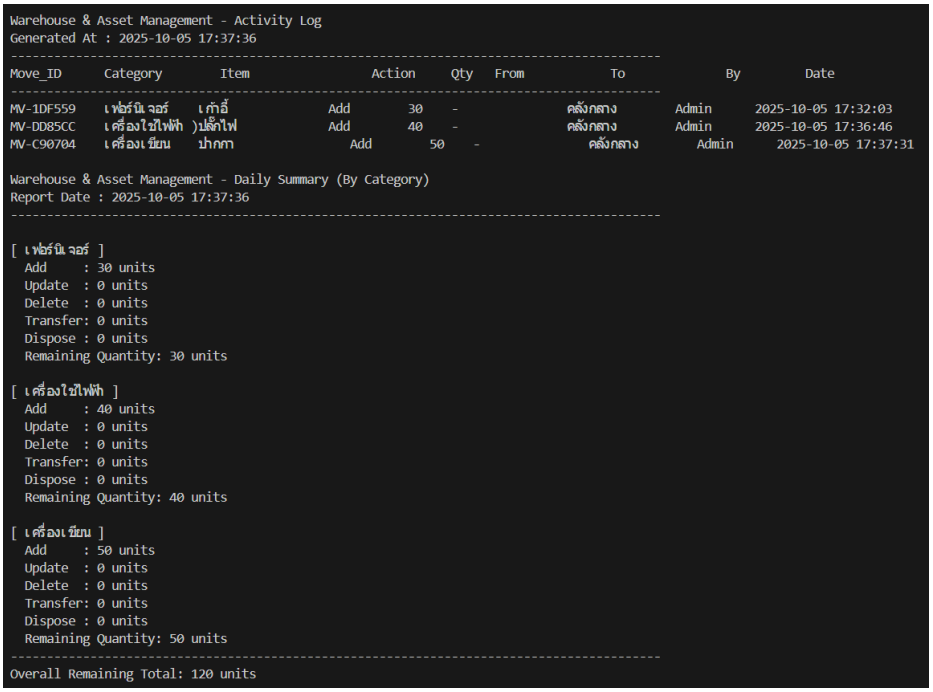
Action_By เก็บชื่อผู้ที่ทำรายการเช่น admin , เจ้าหน้าที่คลัง , ผู้ดูแลระบบ ใช้สำหรับตรวจสอบย้อยหลังว่าใครเป็นผู้ทำการเพิ่ม/ลบ/ย้าย/ทิ้งสินค้า โค้ด Python ตั้งค่าเริ่มต้นเป็น admin โดยอัตโนมัติ แต่สามารถเพิ่มระบบล็อกอินเพื่อเปลี่ยนชื่อผู้ใช้ใดในอนาคต

2.3.8 Action_Date วันเวลาที่ทำ

Action_Date บันทึกวันที่และเวลาที่เกิดการกระทำ ใช้เพื่อตรวจสอบเวลาย้อนหลัง เช่น สินค้าถูกย้ายเมื่อไหร่ หรือ ใครเพิ่มเข้ามาเมื่อวันไหน โค้ด Python จะกำหนดค่าอัตโนมัติทุกครั้งที่บันทึก log ใหม่

2.4 ไฟล์ report.txt

ไฟล์ report.txt ในระบบคลังพัสดุ/ครุภัณฑ์ ประกอบไปด้วย 9 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญดังนี้



```
Warehouse & Asset Management - Activity Log
Generated At : 2025-10-05 17:37:36
-------------------------------------------------------------------------------------------
Move_ID      Category        Item              Action     Qty   From          To            By        Date
-------------------------------------------------------------------------------------------
MV-1DF559    เฟอร์นิเจอร์    เก้าอี้           Add        30    -             คลังกลาง      Admin     2025-10-05 17:32:03
MV-DD85CC    เครื่องใช้ไฟฟ้า )ปลั๊กไฟ          Add        40    -             คลังกลาง      Admin     2025-10-05 17:36:46
MV-C90704    เครื่องเขียน    ปากกา             Add        50    -             คลังกลาง      Admin     2025-10-05 17:37:31

Warehouse & Asset Management - Daily Summary (By Category)
Report Date : 2025-10-05 17:37:36
-------------------------------------------------------------------------------------------

[ เฟอร์นิเจอร์ ]
  Add      : 30 units
  Update   : 0 units
  Delete   : 0 units
  Transfer: 0 units
  Dispose : 0 units
  Remaining Quantity: 30 units

[ เครื่องใช้ไฟฟ้า ]
  Add      : 40 units
  Update   : 0 units
  Delete   : 0 units
  Transfer: 0 units
  Dispose : 0 units
  Remaining Quantity: 40 units

[ เครื่องเขียน ]
  Add      : 50 units
  Update   : 0 units
  Delete   : 0 units
  Transfer: 0 units
  Dispose : 0 units
  Remaining Quantity: 50 units
-------------------------------------------------------------------------------------------
Overall Remaining Total: 120 units
```

รูปภาพที่ 2-4 ไฟล์ report

2.4.1 Move_ID

Data Type VARCHAR/STRING (ตัวอย่าง MV-1DF559) สร้างจาก generate_id(“MV”) ใน log_movement() รหัสเฉพาะของแต่ละรายการ log เป็น primary key ของแถว log ใช้อ้างอิงรายการได้และต้องไม่ซ้ำกันในระบบ

2.4.2 Category

Data type VARCHAR (ชื่อหมวดหมู่) และ/หรือเก็บ Category_ID แยกคอลัมน์ ตัวอย่างเช่น เฟอร์นิเจอร์ , เครื่องใช้ไฟฟ้า , เครื่องเขียน เอาไว้แสดงหมวดหมู่ของสินค้าที่เกิด Action เพื่อให้ดูภาพรวมตามประเภทได้ง่าย

2.4.3 Item

Data type VARCHAR ชื่อสินค้าตัวอย่าง เช่น เก้าอี้ , ปลั๊กไฟ , ปากกา คือสินค้าที่ถูกกระทำ ทำให้ผู้ใช้รู้ว่า action นั้นเกี่ยวกับสินค้าอะไร

2.4.4 Action

Data type ENUM หรือ CARCHAR (ค่าเช่น add , update , delete , transfer , dispose , view) คือประเภทของการกระทำ สำคัญสำหรับสรุป summary และ audit trail

2.4.5 Quantity

Data type INT หรือ DECIMAL ถ้ามีเศษ ตัวอย่าง 30 , 40 , 50 คือจำนวนที่เกี่ยวข้องกับ action ครั้งนั้น เช่น add 30 , transfer 3 ใช้รวมเป็นสถิติใน summary

2.4.6 From_Location

Data type VARCHAR(100) ตัวอย่าง add คลังกลาง คือ สถานที่ต้นทางก่อนการเคลื่อนไหว จำเป็นในการ track เส้นทางของพัสดุ ถ้าไม่ระบุให้ชัดเจนรายงานจะไม่บอกต้นทางที่แท้จริง

2.4.7 To_Location

Data type VARCHAR(100) คือ สถานที่ปลายทางหลังการเคลื่อนไหว ใช้ดูว่าพัสดุไปอยู่ที่ไหนตอนนี้/เมื่อไหร่ สำหรับบาง action เช่น dispose อาจจะไม่มี To ให้สอดคล้องกับ convention เดียวกับ From

2.4.8 Action_By

Data type VARCHDR(100) คือเอาไว้ระบุผู้ทำรายการ สำคัญมากต่อความรับผิดชอบและตรวจสอบย้อนหลัง หากระบบขยายเป็น multi-user ต้องรับชื่อจาก user แทน hard-code

2.4.9 Action_Date

Data type DATETIME ตัวอย่างเช่น 2025-10-05 17:32:03 คือ เวลาที่เหตุการณ์เกิด สำคัญสำหรับการเรียงลำดับ log , audit และกรอกตามช่วงเวลา

บทที่ 3

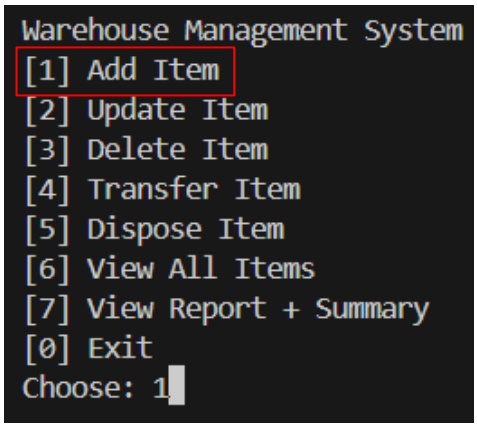
การใช้งานระบบคลังพัสดุ/ครุภัณฑ์

ระบบ คลังพัสดุ/ครุภัณฑ์ คือ ระบบ log การเคลื่อนไหวพัสดุ เป็นระบบที่ใช้สำหรับ บันทึก ติดตามและเคลื่อนไหวของรายการพัสดุ/ครุภัณฑ์ ตั้งแต่การ เพิ่ม , แก้ไข , ลบ , โอนย้าย ไปจนถึง จำหน่าย โดยทุกครั้งที่เกดิการกระทำ ระบบจะสร้าง บันทึกเหตุการณ์เก็บไว้พร้อมรายละเอียด เช่น ผู้ใด เป็นผู้ทำ ทำอะไร เมื่อไหร่ จำนวนเท่าไหร่ ย้ายจากที่ไหนไปที่ไหน เป็นต้น

สำหรับผู้ใช้งานโปรแกรม

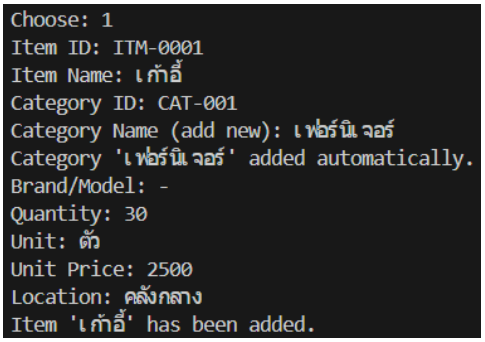
3.1 การใช้งานระบบคลังพัสดุ/ครุภัณฑ์

3.1.1 กรอกหมายเลข 1 ภายในกรอบแดงเพื่อเรียกฟังก์ชัน Add Item เพื่อเพิ่มข้อมูลที่ประกอบไปด้วย Item id , Item Name , Category ID , Category Name , Brand/Model , Quantity Unit , Unit Price , Location



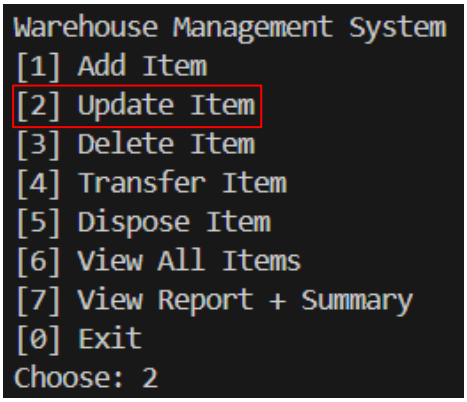
รูปภาพที่ 3-1 การเลือกใช้ฟังก์ชัน ระบบคลังพัสดุ/ครุภัณฑ์

3.1.2 เมื่อฟังก์ชัน Add Item ขึ้นมาแล้วจากนั้นก็สามารถระบุสิ่งที่ต้องการเพิ่มลงไปได้ไม่ว่าจะเป็น โต๊ะ เก้าอี้ หรือกปากกา กรอกจำนวนตามที่เรา add ได้



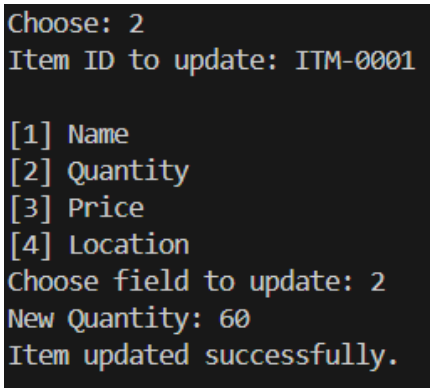
รูปภาพที่ 3-2 การใส่ข้อมูลในฟังก์ชัน ระบบคลังพัสดุ/ครุภัณฑ์

3.1.3 กรอกหมายเลข 2 ภายในกรอบสีแดงเพื่อเรียกใช้ฟังก์ชัน Update Item เพื่อ update ข้อมูลของพัสดุต่างๆในระบบ



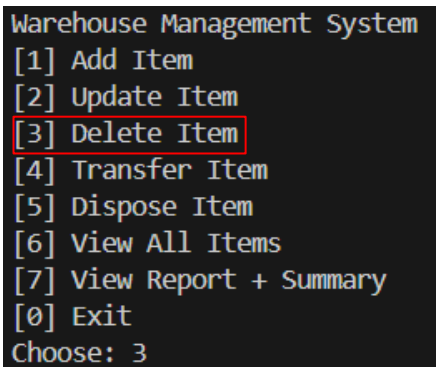
รูปภาพที่ 3-3 การเลือก update ข้อมูลในระบบ

3.1.4 เมื่อกดฟังก์ชัน Update Item ขึ้นมาแล้วจากนั้นสามารถ Update พัสดุในระบบคลังได้ตามที่ต้องการในรูปภาพจะเป็นการเลือก update ว่าจะ update อะไร จะมีให้เลือกว่าจะ update ชื่อ , จำนวน , ราคา และ ที่อยู่ ในรูปภาพจะเป็นการเลือก update จำนวนใหม่



รูปภาพที่ 3-4 การ update ข้อมูลในระบบ

3.1.5 กรอกมหายเลข 3 ในกรอบสีแดงเพื่อเรียกใช้ฟังก์ชัน Delete Item เพื่อ Delete จำนวนสิ่งของในระบบ



รูปภาพที่ 3-5 ฟังก์ชันลบจำนวนสิ่งของ

3.1.6 เมื่อกดฟังก์ชัน Delete Item มาแล้ว เราสามารถเลือก item ที่เราจะลดจำนวนลงได้ในภาพจะเป็นการลดจำนวนของลงจาก 60 เหลือ 50

```
Choose: 3
Item ID to delete: ITM-0001
Current quantity: 60
Quantity to delete: 10
10 units deleted. Remaining quantity: 50 units.
```

รูปภาพที่ 3-6 การ Delete จำนวนออกจากระบบ

3.1.7 กรอกหมายเลข 4 ในกรอบสีแดงเพื่อเรียกใช้ฟังก์ชัน Transfer Item เพื่อย้ายสินค้าในระบบไปที่ต่างๆ

```
Warehouse Management System
[1] Add Item
[2] Update Item
[3] Delete Item
[4] Transfer Item
[5] Dispose Item
[6] View All Items
[7] View Report + Summary
[0] Exit
Choose: 4
```

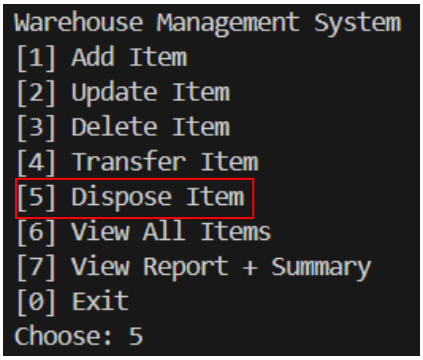
รูปภาพที่ 3-7 ฟังก์ชันย้ายของในระบบ

3.1.8 เมื่อกดฟังก์ชัน Transfer Item จะให้ใส่ Item id ลงไป ใส่จำนวนที่เราจะย้าย และ location ที่เราจะไปได้

```
Choose: 4
Item ID to transfer: ITM-0001
Quantity to transfer: 5
Destination Location: ห้องสมุด
Item 'ITM-0001' transferred to ห้องสมุด.
```

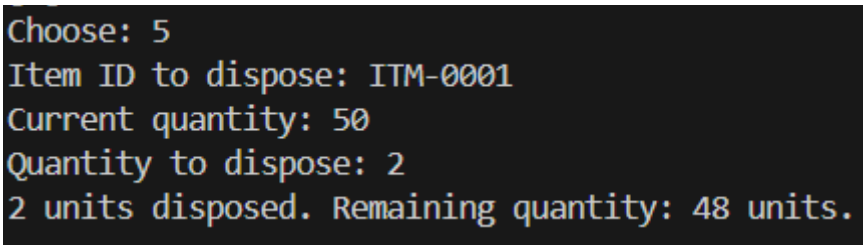
รูปภาพที่ 3-8 การกรอกข้อมูลเพื่อ Transfer ไปยังที่ต่างๆ

3.1.9 กรอกหมายเลข 5 ในกรอบสีแดงเพื่อเรียกใช้ฟังก์ชัน Dispose Item เพื่อทำการลบทิ้งออกจากระบบคลังพัสดุ/ครุภัณฑ์



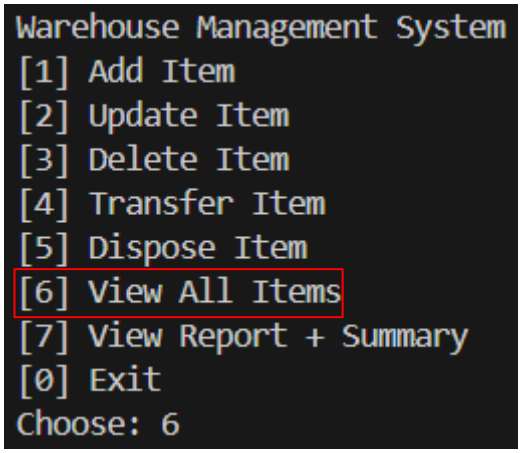
รูปภาพที่ 3-9 ฟังก์ชันลบสินค้าออกจากระบบ

3.1.10 เมื่อกดฟังก์ชัน Dispose Item จะให้ใส่ Item ID ลงไป และจำนวนที่เราจะทำการลบทิ้งออกจากระบบก่อนจะลบจะแสดงโชว์ว่ายังมีจำนวนของเหลือเท่าไหร่ก่อนที่เราจะลบออกจากระบบ



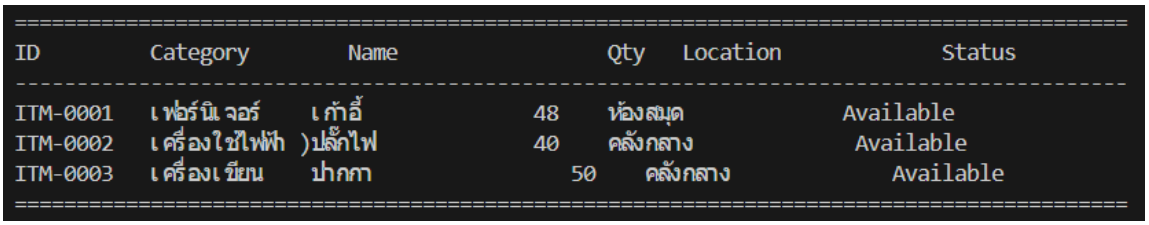
รูปภาพที่ 3-10 กรอกจำนวนเพื่อลบจำนวนของออกจากระบบ

3.1.11 กรอกหมายเลข 6 ในกรอบสีแดงเพื่อเรียกใช้ฟังก์ชัน View All Items เพื่อเรียกดูจำนวนของและสินค้าที่ยังอยู่ในระบบ สามารถรู้ชื่อสินค้าและหมวดหมู่ของสินค้าได้ รวมถึง status ได้



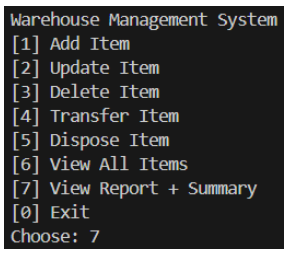
รูปภาพที่ 3-11 ฟังก์ชัน View All Items

3.1.12 เมื่อกดฟังก์ชัน View All Items เราสารถรู้จำนวนของและสินค้าที่ยังอยู่ในระบบ สามารถรู้ชื่อสินค้าและหมวดหมู่ของสินค้าได้ รวมถึง status ได้



รูปภาพที่ 3-12 View All Items

3.1.13 กรอกหมายเลข 7 ในกรอบสีแดงเพื่อเรียกใช้ฟังก์ชัน View Report + Summary เพื่อเรียกดูสินค้าและจำนวนคงเหลือทั้งหมด



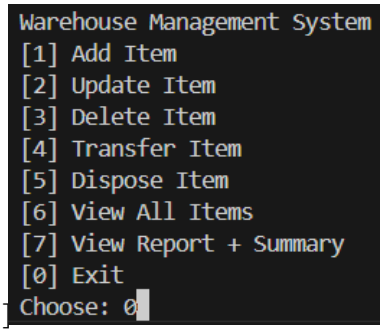
รูปภาพที่ 3-13 ฟังก์ชันดู report ทั้งหมด

3.1.14 เมื่อกดฟังก์ชัน View Report + Summary เราสารถรู้จำนวนและสินค้าทั้งหมดและสามารถรู้ใครทำการเพิ่ม ลบ ย้ายสินค้า ลบสินค้าออกจากระบบ และ อัปเดต รู้จำนวนคงเหลือแต่ละหมวดหมู่ได้ทุกอย่าง

```
Warehouse & Asset Management - Activity Log
Generated At : 2025-10-06 00:15:07
-------------------------------------------------------------------------------------------
Move_ID      Category        Item                Action     Qty    From          To            By         Date
-------------------------------------------------------------------------------------------
MV-1DF559    เฟอร์นิเจอร์    เก้าอี้             Add        30     -             คลังกลาง      Admin      2025-10-05 17:32:03
MV-DD85CC    เครื่องใช้ไฟฟ้า )ปลั๊กไฟ            Add        40     -             คลังกลาง      Admin      2025-10-05 17:36:46
MV-C90704    เครื่องเขียน    ปากกา               Add        50     -             คลังกลาง      Admin      2025-10-05 17:37:31
MV-F96613    เฟอร์นิเจอร์    เก้าอี้             Update     30     -             คลังกลาง      Admin      2025-10-05 19:44:49
MV-537867    เฟอร์นิเจอร์    เก้าอี้             Delete     10     คลังกลาง      -             Admin      2025-10-05 19:57:07
MV-84598F    เฟอร์นิเจอร์    เก้าอี้             Transfer   5      คลังกลาง      ห้องสมุด      Admin      2025-10-05 23:48:36
MV-42849E    เฟอร์นิเจอร์    เก้าอี้             Dispose    2      ห้องสมุด      -             Admin      2025-10-05 23:56:34

Warehouse & Asset Management - Daily Summary (By Category)
Report Date : 2025-10-06 00:15:07
-------------------------------------------------------------------------------------------

[ เฟอร์นิเจอร์ ]
  Add      : 30 units
  Update   : 30 units
  Delete   : 10 units
  Transfer: 5 units
  Dispose  : 2 units
  Remaining Quantity: 48 units

[ เครื่องใช้ไฟฟ้า ]
  Add      : 40 units
  Update   : 0 units
  Delete   : 0 units
  Transfer: 0 units
  Dispose  : 0 units
  Remaining Quantity: 40 units

[ เครื่องเขียน ]
  Add      : 50 units
  Update   : 0 units
  Delete   : 0 units
  Transfer: 0 units
  Dispose  : 0 units
  Remaining Quantity: 50 units
```

รูปภาพที่ 3-14 ภาพ report ทั้งหมด

3.1.15 กรอกหมายเลข 0 เพื่อเรียกใช้ฟังก์ชัน Exit เพื่อทำการออกจากระบบที่เราได้ทำการเข้ามาใช้ระบบคลังพัสดุ/ครุภัณฑ์



รูปภาพที่ 3-15 ฟังก์ชันออกจากระบบ

บทที่ 4
อธิบายการทำงานของ Code

4.1 ฟังก์ชันไบนารีพื้นฐานในระบบคลังพัสดุ/ครุภัณฑ์

4.1.1 นำเข้าโมดูล datetime เพื่อใช้การจัดการวันที่/เวลา (เช่น datetime.datetime.now()), ใช้แสดง Action_Date ใน log และหัวรายงาน

```
import datetime
```

รูปภาพที่ 4-1 import datetime

4.1.2 นำเข้าโมดูล uuid เพื่อสร้างรหัสสุ่มไม่ซ้ำ (unique id) สำหรับ Move_ID (ผ่าน uuid.uuid4())

```
import uuid
```

รูปภาพที่ 4-2 import uuid

4.1.3 นำเข้า sys เพื่อใช้ sys.exit(0) ในการออกจากโปรแกรมอย่างสะอาดเมื่อผู้ใช้เลือก Exit

```
import sys
```

รูปภาพที่ 4-3 import sys

4.1.4 คอมเมนต์แบ่งโซน items = {} dictionary เก็บสินค้าทั้งหมด โดยคีย์คือ item_id (string) และ value เป็น dict ของฟิลด์สินค้า (Item_Name, Quantity, Location, ...) categories = {}dictionary เก็บหมวด เช่น categories["CAT-001"] = {"Category_Name": "เฟอร์นิเจอร์"}movements = [] list เก็บ log แต่ละรายการเป็น dict (movement records)

```
items = {}
categories = {}
movements = []
```

รูปภาพที่ 4-4 คอมเมนต์แบ่งโซน

4.2 ฟังก์ชันช่วย utility helpers

4.2.1 def generate_id(prefix): ประกาศฟังก์ชันที่รับ prefix (เช่น "MV") uuid.uuid4() สร้าง UUID แบบสุ่ม .hex เอารูปแบบ hex string ของ UUID [:6] ตัด 6 ตัวแรก (ลดความยาวให้สั้น) .upper() แปลงเป็นตัวพิมพ์ใหญ่ f-string รวม prefix กับรหัสตัวอักษร เช่น "MV-1A2B3C" ข้อสังเกต: 6 ตัวอาจชนกันได้ในระบบใหญ่ ถ้าต้องการความปลอดภัยสูง ให้เพิ่มจำนวนตัวอักษร

```
def generate_id(prefix):
    return f"{prefix}-{uuid.uuid4().hex[:6].upper()}"
```

รูปภาพที่ 4-5 ฟังก์ชันสร้าง UUID แบบสุ่ม

4.2.2 ฟังก์ชันเล็ก ๆ สำหรับพิมพ์เส้นคั่น ('-' * 90) เพื่อทำให้ output ดูเป็นตารางใน terminal พารามิเตอร์ char และ length สามารถเปลี่ยนได้

```
def print_line(char='-', length=90):
    print(char * length)
```

รูปภาพที่ 4-6 ฟังก์ชันสำหรับพิมพ์เส้นคั่น

4.2.3 คืนค่าวันที่และเวลา ณ ปัจจุบันในรูปแบบ YYYY-MM-DD HH:MM:SS ใช้เมื่อบันทึก Action_Date ใน movement และแสดงใน report

```
def get_datetime_now():
    return datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

รูปภาพที่ 4-7 ฟังก์ชันคืนค่าวันที่และวันเวลา

4.3 ฟังก์ชันเพิ่มสินค้าใหม่

4.3.1 ประกาศฟังก์ชัน add_item รับ Item ID จากผู้ใช้ผ่าน input() และ .strip() เพื่อลบช่องว่างหัว/ท้าย

```
def add_item():
    item_id = input("Item ID: ").strip()
```

รูปภาพที่ 4-8 ฟังก์ชันรับ item ผ่าน input

4.3.2 ถ้า item_id มีอยู่แล้วใน items — แจ้งผู้ใช้และ return เพื่อหยุดฟังก์ชัน (ไม่อนุญาตเพิ่มซ้ำ)

```
if item_id in items:
    print("Item ID already exists.")
    return
```

รูปภาพที่ 4-9 ฟังก์ชันตรวจสอบการซ้ำกัน

4.3.3 รับชื่อสินค้าและรหัสหมวด (Category ID)

```
item_name = input("Item Name: ").strip()
category_id = input("Category ID: ").strip()
```

รูปภาพที่ 4-10 ฟังก์ชันรับชื่อสินค้าและหมวดหมู่

4.3.4 ตรวจว่าหมวดมีอยู่ไหม ถ้าไม่มี ให้รับ Category Name แล้วสร้าง entry ใหม่ใน categories แบบง่าย ๆ แล้วแจ้งผู้ใช้

```
if category_id not in categories:
    category_name = input("Category Name (add new): ").strip()
    categories[category_id] = {"Category_Name": category_name}
    print(f"Category '{category_name}' added automatically.")
```

รูปภพที่ 4-11 Category

4.3.5 รับข้อมูลเสริม: ยี่ห้อ, จำนวน (int), หน่วย, ราคาต่อหน่วย (float), ตำแหน่งเก็บ ข้อควรระวัง: int() และ float() จะโยน ValueError หากผู้ใช้พิมพ์ไม่ถูก — ถ้าต้องการความทนทานควรใส่ try/except และให้ผู้ใช้กรอกใหม่

```
brand = input("Brand/Model: ").strip()
qty = int(input("Quantity: "))
unit = input("Unit: ").strip()
price = float(input("Unit Price: "))
location = input("Location: ").strip()
```

รูปภาพที่ 4-12 ฟังก์ชันรับข้อมูลเสริม

4.3.6 สร้าง dict ของสินค้าแล้วเก็บใน items โดยใช้ item_id เป็นคีย์ ฟิลด์ Status กำหนดเริ่มต้นเป็น "Available"

```
items[item_id] = {
    "Item_Name": item_name,
    "Category_ID": category_id,
    "Brand_Model": brand,
    "Quantity": qty,
    "Unit": unit,
    "UnitPrice": price,
    "Status": "Available",
    "Location": location
}
```

รูปภาพที่ 4-13 ฟังก์ชันสร้าง dict ของสินค้า

4.3.7 เรียก log_movement() เพื่อบันทึก movement ประเภท "Add" from_loc ใช้ "-" เป็นตัวแทนว่าไม่มีต้นทาง (เพิ่มใหม่) และ to_loc เป็น location แจ้งผู้ใช้ว่าการเพิ่มเสร็จ

```
log_movement(item_id, "Add", qty, "-", location)
print(f"Item '{item_name}' has been added.")
```

รูปภาพที่ 4-14 ฟังก์ชันบันทึกการ movement

4.4 ฟังก์ชัน แก้ไขข้อมูลสินค้า

4.4.1 รับ Item ID ที่จะอัปเดต ตรวจว่ามีหรือไม่ หากไม่มีจะหยุดและแจ้ง

```
def update_item():
    item_id = input("Item ID to update: ").strip()
    if item_id not in items:
        print("Item not found.")
        return
```

รูปภาพที่ 4-15 ฟังก์ชันรับ item id

4.4.2 แสดงเมนูย่อยให้ผู้ใช้เลือกว่าจะอัปเดตอะไร แล้วรับคำตอบ

```
print("\n[1] Name\n[2] Quantity\n[3] Price\n[4] Location")
choice = input("Choose field to update: ").strip()
```

รูปภาพที่ 4-16 ฟังก์ชันแสดงเมนูย่อย

4.4.3 ถ้าเลือก 1 รับชื่อใหม่แล้วเขียนทับ Item_Name ใน items

```
if choice == "1":
    items[item_id]["Item_Name"] = input("New Name: ")
```

รูปภาพที่ 4-17 ฟังก์ชัน choice

4.4.4 ถ้าเลือก 2 รับ New Quantity เป็น int คำนวณ diff = จำนวนใหม่ – จำนวนเก่า (จะเป็นบวกเมื่อเพิ่ม, ลบเมื่อลด) อัปเดต items[item_id]["Quantity"] และบันทึก movement "Update" โดยเก็บ Quantity = diff ข้อสังเกต: การเก็บ diff เป็นวิธีหนึ่ง แต่จะทำให้ summary ต้องตีความว่า Update หมายถึงความเปลี่ยนแปลง (ไม่ใช่ยอดหลังการเปลี่ยน)

```
elif choice == "2":
    new_qty = int(input("New Quantity: "))
    diff = new_qty - items[item_id]["Quantity"]
    items[item_id]["Quantity"] = new_qty
    log_movement(item_id, "Update", diff, "-", items[item_id]["Location"])
```

รูปภาพที่ 4-18 ฟังก์ชัน new quantity

4.4.5 ถ้าเลือก 3 อัปเดตราคาต่อหน่วย (ไม่มี log โดยปริยาย)

```
elif choice == "3":
    items[item_id]["UnitPrice"] = float(input("New Price: "))
```

รูปภาพที่ 4-19 ฟังก์ชันอัปเดตราคาต่อหน่วย

4.4.6 ถ้าเลือก 4 เปลี่ยน Location ของสินค้า (ปัจจุบันไม่มีการบันทึกเป็น Transfer ใน log)

```
elif choice == "4":
    items[item_id]["Location"] = input("New Location: ")
```

รูปภาพที่ 4-20 ฟังก์ชันเปลี่ยน location

4.4.7 หากเลือกตัวเลือกอื่นที่ไม่รองรับ จบด้วยข้อความ error หากอัปเดตสำเร็จ แจ้งผู้ใช้

```
else:
    print("Invalid option.")
    return
print("Item updated successfully.")
```

รูปภาพที่ 4-21 ฟังก์ชันตรวจสอบข้อความที่ผิด

4.5 ฟังก์ชันลบสินค้า (บางส่วนหรือทั้งหมด)

4.5.1 รับ Item ID ตรวจว่ามีหรือไม่

```
def delete_item():
    item_id = input("Item ID to delete: ").strip()
    if item_id not in items:
        print("Item not found.")
        return
```

รูปภาพที่ 4-22 รับ item แล้วตรวจสอบว่ามีหรือไม่

4.5.2 เอา current_qty มาแสดง แล้วรับจำนวนที่จะลบ (int)

```
current_qty = items[item_id]["Quantity"]
print(f"Current quantity: {current_qty}")
qty_to_delete = int(input("Quantity to delete: "))
```

รูปภาพที่ 4-23 ฟังก์ชันรับจำนวนที่จะลบ

4.5.3 ตรวจความถูกต้องของจำนวน (ไม่เป็นลบ/ไม่มากกว่า stock)

```
if qty_to_delete <= 0:
    print("Invalid quantity.")
    return
elif qty_to_delete > current_qty:
    print("Not enough quantity to delete.")
    return
```

รูปภาพที่ 4-24 ตรวจความถูกต้องของจำนวน

4.5.4 บันทึก movement ประเภท Delete โดยระบุ from_loc และ to เป็น "-"

```
from_loc = items[item_id]["Location"]
log_movement(item_id, "Delete", qty_to_delete, from_loc, "-")
```

รูปภาพที่ 4-25 บันทึก movement

4.5.5 ถ้าลบเท่ากับจำนวนทั้งหมด ลบ entry ทั้งหมดออกจาก items

```
if qty_to_delete == current_qty:
    del items[item_id]
    print("Item completely deleted from inventory. Remaining quantity: 0")
```

รูปภาพที่ 4-26 ฟังก์ชัน qty_to_delete

4.5.6 ถ้าลบบางส่วน ลดจำนวนใน items และแสดงยอดคงเหลือ

```
else:
    items[item_id]["Quantity"] -= qty_to_delete
    remaining = items[item_id]["Quantity"]
    print(f"{qty_to_delete} units deleted. Remaining quantity: {remaining} units.")
```

รูปภาพที่ 4-27 ลดจำนวนใน item

4.6 ฟังก์ชัน โอนย้ายสินค้า

4.6.1 รับ Item ID ตรวจสถานะ

```
def transfer_item():
    item_id = input("Item ID to transfer: ").strip()
    if item_id not in items:
        print("Item not found.")
        return
```

รูปภาพที่ 4-28 ตรวจสอบสถานะ item id

4.6.1 รับจำนวนที่จะโอน (int), เก็บ from_loc ปัจจุบัน และรับ to_loc (ปลายทาง)

```
qty = int(input("Quantity to transfer: "))
from_loc = items[item_id]["Location"]
to_loc = input("Destination Location: ").strip()
```

รูปภาพที่ 4-29 รับจำนวนที่จะโอน

4.6.2 ตรวจว่าเพียงพอหรือไม่

```
if qty > items[item_id]["Quantity"]:
    print("Not enough quantity.")
    return
```

รูปภาพที่ 4-30 ตรวจสอบจำนวนก่อนลบ

4.6.3 สำคัญ: โค้ดนี้เปลี่ยน Location ของ item_id ทันทีเป็น to_loc สำหรับทั้งรายการ หมายความว่าถ้า item_id มี 10 ชิ้นและผู้ใช้โอน 3 ชิ้น โค้ดจะเปลี่ยนตำแหน่งทั้ง 10 ชิ้นไปที่ to_loc (model ปัจจุบันไม่รองรับ stock แยกตำแหน่ง)บันทึก Transfer movement พร้อมจำนวนที่ระบุ และแจ้งผล

```
items[item_id]["Location"] = to_loc
log_movement(item_id, "Transfer", qty, from_loc, to_loc)
print(f"Item '{item_id}' transferred to {to_loc}.")
```

รูปภาพที่ 4-31 เปลี่ยนตำแหน่งและบันทึกการย้าย

4.7 ฟังก์ชัน กำจัดสินค้า (Dispose)

4.7.1 รับ Item ID ตรวจว่ามี

```
def dispose_item():
    item_id = input("Item ID to dispose: ").strip()
    if item_id not in items:
        print("Item not found.")
        return
```

รูปภาพที่ 4-32 รับ item id ตรวจว่ามีหรือป่าว

4.7.2 แสดงจำนวนปัจจุบัน รับจำนวนที่จะทิ้ง

```
current_qty = items[item_id]["Quantity"]
print(f"Current quantity: {current_qty}")
qty_to_dispose = int(input("Quantity to dispose: "))
```

รูปภาพที่ 4-33 แสดงจำนวนปัจจุบัน รับจำนวนที่จะทิ้ง

4.7.3 เอาไว้ตรวจจำนวน

```
if qty_to_dispose <= 0:
    print("Invalid quantity.")
    return
elif qty_to_dispose > current_qty:
    print("Not enough quantity to dispose.")
    return
```

รูปภาพที่ 4-34 ตรวจจำนวน

4.7.4 บันทึก movement "Dispose" (to = "-")

```
from_loc = items[item_id]["Location"]
log_movement(item_id, "Dispose", qty_to_dispose, from_loc, "-")
```

รูปภาพที่ 4-35 บันทึก movement dispose

4.7.5 ถ้าทิ้งทั้งหมดตั้ง Status เป็น "Disposed" และ Quantity = 0 ถ้าทิ้งบางส่วน ลด Quantity และแสดงยอดคงเหลือ

```
if qty_to_dispose == current_qty:
    items[item_id]["Status"] = "Disposed"
    items[item_id]["Quantity"] = 0
    print("🗑️ Item completely disposed.")
else:
    items[item_id]["Quantity"] -= qty_to_dispose
    remaining = items[item_id]["Quantity"]
    print(f"🗑️ {qty_to_dispose} units disposed. Remaining quantity: {remaining} units.")
```

รูปภาพที่ 4-36 ลด quantity และแสดงยอดคงเหลือ

4.8 ฟังก์ชันแสดงรายการทั้งหมด

4.8.1 พิมพ์ header ตาราง โดยใช้ f-string กับ alignment (:<n) เพื่อให้คอลัมน์ตรงกันใน terminal

```
def view_items():
    print_line("=")
    print(f"{'ID':<10} {'Category':<15} {'Name':<20} {'Qty':<5} {'Location':<20} {'Status':<10}")
    print_line("-")
```

รูปภาพที่ 4-37 ใช้เพื่อให้คอลัมน์นั้นตรงกัน

4.8.2 วนแต่ละ item ดึง Category_Name จาก categories และพิมพ์แถวข้อมูล ข้อควรระวัง: ถ้าหมวดหมู่ถูกลบหรือ Category_ID ไม่อยู่จะเกิด KeyError

```
for item_id, data in items.items():
        cat_name = categories[data["Category_ID"]]["Category_Name"]
        print(f"{item_id:<10} {cat_name:<15} {data['Item_Name']:<20} {data['Quantity']:<5} {data['Location']:<20} {data['Status']:<10}")
    print_line("=")
```

รูปภาพที่ 4-38 Item id data

4.9 ฟังก์ชัน สร้างบันทึกการเคลื่อนไหว

4.9.1 ประกาศฟังก์ชันสำหรับสร้าง movement record และสร้าง Move_ID ผ่าน generate_id

```
def log_movement(item_id, action, qty, from_loc, to_loc):
    move_id = generate_id("MV")
```

รูปภาพที่ 4-39 ฟังก์ชันสำหรับสร้าง movement record

4.9.2 พยายามดึง Category_ID ของ item ถ้า item_id ยังอยู่ใน items ถ้าไม่อยู่ให้ใช้ "-" (ป้องกันกรณี log เรียกหลัง item ถูกลบไปแล้ว)

```
cat_id = items[item_id]["Category_ID"] if item_id in items else "-"
```

รูปภาพที่ 4-40 การดึง cat id

4.9.3 สร้าง dict ที่ประกอบด้วย field ต่าง ๆ (Move_ID, Category_ID, Item_ID, Action_Type, Quantity, From, To, By, Date) Action_By กำหนดเป็น "Admin" แบบคงที่ (หากต้องการ multi-user ต้องแก้ให้รับ user จาก session)

```
movement = {
    "Move_ID": move_id,
    "Category_ID": cat_id,
    "Item_ID": item_id,
    "Action_Type": action,
    "Quantity": qty,
    "From_Location": from_loc,
    "To_Location": to_loc,
    "Action_By": "Admin",
    "Action_Date": get_datetime_now()
}
```

รูปภาพที่ 4-41 สร้าง dict ที่ปะกอบด้วยฟิลด์ต่างๆ

4.9.4 นำ movement dict เพิ่มเข้า movements list (บันทึกในหน่วยความจำ)

```
movements.append(movement)
```

รูปภาพที่ 4-42 movement dict

4.10 ฟังก์ชันแสดง Activity Log และสรุปตามหมวด

4.10.1 แสดงหัวรายงานและเวลาสร้าง

```
def report():
    print("\nWarehouse & Asset Management - Activity Log")
    print(f"Generated At : {get_datetime_now()}")
    print_line()
```

รูปภาพที่ 4-43 แสดงหัวรายงานและเวลาสร้าง

4.10.2 พิมพ์ header ของ activity log โดยตั้งความกว้างคอลัมน์เพื่อให้ตรง

```
print(f"{'Move_ID':<12} {'Category':<15} {'Item':<20} {'Action':<10} {'Qty':<5} {'From':<15} {'To':<15} {'By':<10} {'Date'}")
    print_line("-")
```

รูปภาพที่ 4-44 พิมพ์ header ของ action log

4.10.3 วนแต่ละ movement: ดึง Category_Name ถ้ามี (else "-") ดึง Item_Name ถ้าสินค้ายังอยู่ใน items (else "-") พิมพ์แถวข้อมูลให้ตรงคอลัมน์

```
for mv in movements:
    cat_name = categories[mv["Category_ID"]]["Category_Name"] if mv["Category_ID"] in categories
    item_name = items[mv["Item_ID"]]["Item_Name"] if mv["Item_ID"] in items else "-"
    print(f"{mv['Move_ID']:<12} {cat_name:<15} {item_name:<20} {mv['Action_Type']:<10} {mv['Quant
```

รูปภาพที่ 4-45 พิมพ์แถวข้อมูลให้ตรงคอลัมน์

4.10.4 แสดงหัวเรื่องสรุปรายวัน

```
print("\nWarehouse & Asset Management - Daily Summary (By Category)")
print(f"Report Date : {get_datetime_now()}")
print_line()
```

รูปภาพที่ 4-46 แสดงหัวเรื่องสรุปรายวัน

4.10.5 เตรียม dict สำหรับเก็บผลรวม action แยกตาม category (summary) และยอดคงเหลือตาม category (remaining_qty_by_cat)

```
summary = {}
remaining_qty_by_cat = {}
```

รูปภาพที่ 4-47 เตรียม dict สำหรับเก็บผลรวม

4.10.6 วน movements เพื่อสะสมค่า per-category: หา cat_name หาก category นี้ยังไม่อยู่ใน summary ให้สร้าง bucket เริ่มต้นสำหรับ action types ที่ระบบใช้ถ้า Action_Type ตรงกับคีย์ใน bucket ให้เพิ่ม Quantity เข้าไป (สะสมยอดของ action แบบ simple)**ข้อสังเกต:** Update ในระบบเก็บ diff ทำให้ summary ของ Update เป็นผลรวมของการเปลี่ยนแปลง (อาจเป็นลบหาก diff negative)

```
for mv in movements:
    cat_name = categories[mv["Category_ID"]]["Category_Name"] if mv["Category_ID"] in categories e
    if cat_name not in summary:
        summary[cat_name] = {"Add": 0, "Update": 0, "Delete": 0, "Transfer": 0, "Dispose": 0}
        remaining_qty_by_cat[cat_name] = 0
    if mv["Action_Type"] in summary[cat_name]:
        summary[cat_name][mv["Action_Type"]] += mv["Quantity"]
```

รูปภาพที่ 4-48 วน movements เพื่อสะสมค่า per-category

4.10.7 วน items ที่ยังอยู่จริง เพื่อรวมยอดคงเหลือตาม category (ใช้สถานะปัจจุบันของ items เป็นแหล่งความจริงของยอดคงเหลือ)

```
for item_id, data in items.items():
    cat_name = categories[data["Category_ID"]]["Category_Name"] if data["Category_ID"] in categorie
    remaining_qty_by_cat[cat_name] = remaining_qty_by_cat.get(cat_name, 0) + data["Quantity"]
```

รูปภาพที่ 4-49 วน items ที่ยังอยู่จริง

4.10.8 สำหรับแต่ละ category ใน summary: พิมพ์ชื่อ category พิมพ์ยอดของแต่ละ action (Add/Update/Delete/Transfer/Dispose) ดึง Remaining Quantity จาก remaining_qty_by_cat (fallback 0 ถ้าไม่มี) รวม total สำหรับทุก category แล้วพิมพ์ Overall Remaining Total ข้อสังเกตสำคัญ: summary มาจาก movements ขณะที่ remaining_qty_by_cat มาจาก items ถ้ามีการเปลี่ยนแปลงใน items ที่ไม่มีการบันทึก movement ทั้งสองฝั่งอาจไม่สอดคล้อง

```
total = 0
for cat, data in summary.items():
    print(f"\n[ {cat} ]")
    for action, qty in data.items():
        print(f"  {action:<8}: {qty} units")
    balance = remaining_qty_by_cat.get(cat, 0)
    total += balance
    print(f"  Remaining Quantity: {balance} units")
print_line()
print(f"Overall Remaining Total: {total} units\n")
```

รูปภาพที่ 4-50 summary มาจาก movements

4.11 ฟังก์ชัน เมนูหลัก และการควบคุม loop

4.11.1 main() เป็น loop เมนูหลัก ใช้ while True เพื่อทำงานต่อเนื่องจนกว่าจะ sys.exit(0) แสดงเมนูตัวเลือก และรับ choice จากผู้ใช้

```
def main():
    while True:
        print("\nWarehouse Management System")
        print("[1] Add Item")
        print("[2] Update Item")
        print("[3] Delete Item")
        print("[4] Transfer Item")
        print("[5] Dispose Item")
        print("[6] View All Items")
        print("[7] View Report + Summary")
        print("[0] Exit")
        choice = input("Choose: ").strip()
```

รูปภาพที่ 4-51 เป็น loop เมนูหลัก

4.11.2 ตรวจค่าที่ผู้ใช้กรอกแล้วเรียกฟังก์ชันที่ตรงกับเมนู ถ้า 0 จะพิมพ์ข้อความแล้วเรียก sys.exit(0) เพื่อจบโปรแกรม ถ้า input ไม่ตรงกับตัวเลือกใด ๆ แจ้งให้ป้อนใหม่

```
if choice == "1":
    add_item()
elif choice == "2":
    update_item()
elif choice == "3":
    delete_item()
elif choice == "4":
    transfer_item()
elif choice == "5":
    dispose_item()
elif choice == "6":
    view_items()
elif choice == "7":
    report()
elif choice == "0":
    print("Exiting program...")
    sys.exit(0)
else:
    print("Please choose a valid number.")
```

รูปภาพที่ 4-52 ตรวจค่าที่ผู้ใช้กรอกแล้วเรียกฟังก์ชันที่ตรงกับเมนู

4.11.3 เมื่อไฟล์นี้ถูก run เป็นสคริปต์หลัก (python script.py) จะเรียก main() เพื่อเริ่มระบบ หากโค้ดนี้ถูก import เป็นโมดูลจากไฟล์อื่น ส่วนนี้จะไม่รัน

```
if __name__ == "__main__":
    main()
```

รูปภาพที่ 4-53 python script.py

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

ระบบนี้ถือเป็น โปรแกรมจัดการคลังพัสดุแบบเบื้องต้น (Inventory Management System)ที่เน้นการ เก็บประวัติการเคลื่อนไหว (Transaction Log) และ รายงานสรุปแบบเรียลไทม์ใน Terminal ซึ่งแม้จะไม่ใช้ฐานข้อมูลจริง (ใช้ list/dict ใน Python) แต่มีโครงสร้างครบถ้วนเทียบเท่าระบบจัดการทรัพย์สินระดับองค์กรในรูปแบบย่อได้อย่างดีเยี่ยม จัดเก็บข้อมูลพัสดุและครุภัณฑ์ อย่างเป็นระบบ ติดตามการเคลื่อนไหวของพัสดุ (Movement Log) เช่น การเพิ่ม, การโอนย้าย, การลบ, การจำหน่าย สรุปยอดคงเหลือ และทำ รายงานสรุปผลประจำวัน/ประเภทหมวดหมู่ ช่วยให้ผู้ดูแลคลังสามารถตรวจสอบรายการพัสดุได้ง่ายและลดข้อผิดพลาดจากการทำงานด้วยมือ

5.2 ปัญหาและอุปสรรคในการดำเนินงาน

ในการพัฒนาระบบคลังพัสดุ/ครุภัณฑ์ ระบบปัจจุบันทำงานได้ดีในระดับ Prototype (ต้นแบบ) แต่ยังมีข้อจำกัดเรื่อง การจัดเก็บข้อมูล, ความปลอดภัย, การใช้งานหลายผู้ใช้, และประสบการณ์ผู้ใช้ (UX) ซึ่งสามารถพัฒนาเพิ่มเติมให้เป็นระบบบริหารจัดการคลังพัสดุแบบสมบูรณ์ได้ในอนาคต

5.3 ข้อเสนอแนะ

เพื่อให้ระบบและพร้อมใช้งานจริงในอนาคต ควรปรับปรุงดังนี้

5.3.1 เพิ่มระบบฐานข้อมูลถาวร (Database System) ปัจจุบันข้อมูลทั้งหมดจัดเก็บในหน่วยความจำ (Memory) ซึ่งจะหายเมื่อปิดโปรแกรม ควรพัฒนาให้เชื่อมต่อฐานข้อมูล เช่น SQLite, MySQL หรือ PostgreSQL เพื่อเก็บข้อมูลถาวรและเรียกใช้งานได้ภายหลัง

5.3.2 เพิ่มระบบสำรองและกู้คืนข้อมูล (Backup & Restore) เพื่อป้องกันการสูญหายของข้อมูล ควรสร้างระบบสำรองอัตโนมัติ เช่น การบันทึกลงไฟล์ .csv หรือ .json ทุกครั้งหลังทำรายการ

5.3.3 ปรับปรุงส่วนแสดงผล (User Interface / GUI) จากการใช้งานผ่าน Terminal ควรพัฒนาเป็นโปรแกรมแบบกราฟิก (GUI) ด้วย Tkinter หรือ PyQt เพื่อให้ผู้ใช้ทั่วไปสามารถใช้งานได้ง่ายขึ้น ไม่ต้องพิมพ์คำสั่งเอง

5.3.4 พัฒนาระบบรายงานอัตโนมัติ (Report System) ควรให้ระบบสามารถสรุปรายงานในรูปแบบกราฟ, ตาราง หรือส่งออกเป็นไฟล์ PDF/Excel ได้ เพื่อความสะดวกในการนำเสนอข้อมูลต่อผู้บริหาร

5.3.5 เพิ่มระบบแจ้งเตือน (Notification System) เมื่อพัสดุเหลือน้อยหรือมีการเคลื่อนไหวจำนวนมาก ควรมีระบบแจ้งเตือน เช่น ส่งอีเมล หรือแสดงข้อความเตือนอัตโนมัติ

5.3.6 เพิ่มการตรวจสอบความถูกต้องของข้อมูล (Data Validation) ก่อนบันทึกข้อมูล เช่น ป้องกันไม่ให้ใส่ตัวอักษรในช่องจำนวน หรือป้อนรหัสสินค้าซ้ำ เพื่อให้ข้อมูลมีความถูกต้องและน่าเชื่อถือ

5.4 สิ่งที่ผู้จัดทำได้รับการพัฒนารางงาน

จากการพัฒนาโครงงานครั้งนี้ ผู้จัดทำได้รับความรู้และประสบการณ์ด้านการออกแบบระบบการเขียนโปรแกรมด้วยภาษา Python การใช้โครงงานสร้างข้อมูลไบนารี รวมถึงการคิดวิเคราะห์และแก้ไขปัญหาเชิงตรรกะ นอกจากนี้ยังได้ฝึกทักษะการทำงานเป็นทีม การแบ่งหน้าที่รับผิดชอบ และการจัดการเวลาให้สอดคล้องกับแผนงาน ทำให้ผู้จัดทำมีความเข้าใจในกระบวนการพัฒนาระบบซอฟต์แวร์มากยิ่งขึ้นและสามารถนำไปประยุกต์ใช้ในโครงการหรืองานจริงในอนาคตต่อไปได้