ระบบคลังพัสดุ

Inventory Management System

เสนอ

รศ.ดร.อนิราช มิ่งขวัญ

จัดทำโดย

นางสาวปณัชชญา หอยจันทร์ รหัส 6806022511013 Sec3

นายภูรีพัชร์ ศรีโชติกราณต์ รหัส 6806022511064 Sec3

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ปีการศึกษา 2568

คำนำ

รายงานฉบับนี้จัดทำขึ้นเพื่อสรุปผลการศึกษาและการพัฒนาระบบ จัดการคลังสินค้า (Inventory Management System) ซึ่งเป็นส่วนหนึ่งของการเรียนในรายวิชา Computer Programming โดยมีวัตถุประสงค์เพื่อให้นักศึกษาได้ฝึกฝนทักษะการเขียนโปรแกรมและการประยุกต์ใช้ความรู้ทางด้านโปรแกรมมิงในการพัฒนาระบบสารสนเทศที่สามารถใช้งานได้จริง

โครงงานระบบคลังสินค้านี้ถูกออกแบบมาเพื่อช่วยในการจัดเก็บและบริหารข้อมูลสินค้าภายในองค์กร โดยระบบสามารถทำงานได้หลายรูปแบบ เช่น การเพิ่มข้อมูล (Add) การแก้ไขข้อมูล (Update) การลบข้อมูล (Delete) การแสดงข้อมูล (View) และการออกรายงานสรุป (Report) เพื่อช่วยให้การจัดการข้อมูลมีความถูกต้องและสะดวกมากยิ่งขึ้น

รายงานฉบับนี้ประกอบด้วยเนื้อหาเกี่ยวกับขั้นตอนการวิเคราะห์และออกแบบระบบ การพัฒนาโปรแกรมด้วยภาษา Python รวมถึงการทดสอบและสรุปผลการทำงานของระบบ ซึ่งมีเป้าหมายให้นักศึกษาได้เข้าใจกระบวนการพัฒนาโปรแกรมเชิงโครงสร้างและการจัดการข้อมูลในรูปแบบไฟล์ไบนารี

ผู้จัดทำหวังเป็นอย่างยิ่งว่ารายงานฉบับนี้จะเป็นประโยชน์ต่อผู้ที่สนใจในด้านการเขียนโปรแกรมและการพัฒนาระบบ เพื่อเป็นแนวทางในการสร้างและพัฒนาระบบคลังสินค้าหรือระบบบริหารข้อมูลอื่น ๆ ต่อไปในอนาคต

สารบัญ

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

หน้า

สารบัญรูปภาพ(ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 วัตถุประสงค์ของโครงงาน

1.1.1 เพื่อพัฒนาโปรแกรมระบบจัดการคลังพัสดุ/ครุภัณฑ์ ที่สามารถบันทึกข้อมูลหมวดหมู่ รายการพัสดุ และการเคลื่อนย้ายได้อย่างเป็นระบบ

1.1.2 เพื่ออำนวยความสะดวกแก่ผู้ใช้งานในการเพิ่ม ลบ แก้ไข และค้นหาข้อมูลพัสดุได้รวดเร็ว

1.1.3 เพื่อสร้างรายงานสรุปจำนวนและมูลค่าพัสดุ รวมถึงประวัติการเคลื่อนย้ายได้อย่างถูกต้องและชัดเจน

1.1.4 เพื่อฝึกทักษะการออกแบบระบบฐานข้อมูลเชิงไฟล์ไบนารี และการเขียนโปรแกรมเชิงโครงสร้าง

1.2 ขอบเขตของโครงงาน

1.2.1 ระบบสามารถจัดเก็บ หมวดหมู่พัสดุ (Category) เช่น คอมพิวเตอร์ เฟอร์นิเจอร์

1.2.2 ระบบสามารถจัดเก็บ รายการพัสดุ (Item) โดยมีข้อมูล ได้แก่ รหัสพัสดุ ชื่อพัสดุ หมวดหมู่ จำนวน ราคา และสถานะการใช้งาน

1.2.3 ระบบสามารถบันทึก ประวัติการเคลื่อนย้าย (Movement) เช่น การเบิก การคืน การโอน และการซ่อม

1.2.4 ระบบรองรับการทำงานหลัก 4 ส่วน คือ เพิ่มข้อมูล (Add) แก้ไข (Update) ลบ (Delete) และดูข้อมูล (View)

1.2.5 ระบบสามารถออกรายงาน (Report) เพื่อสรุปจำนวน มูลค่า และประวัติการเคลื่อนย้าย

1.2.6 โปรแกรมทำงานผ่าน Command Line Interface (CLI) และใช้ไฟล์ไบนารีเป็นตัวเก็บข้อมูล โดยไม่พึ่งพาฐานข้อมูลภายนอก

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ผู้ใช้งานสามารถจัดเก็บและค้นหาข้อมูลพัสดุได้อย่างรวดเร็วและสะดวกขึ้น

1.3.2 ลดความผิดพลาดในการบันทึกและคำนวณจำนวนหรือมูลค่าพัสดุ

1.3.3 ทำให้การติดตามการเคลื่อนย้ายพัสดุมีความถูกต้อง สามารถตรวจสอบย้อนหลังได้

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.4 ได้โปรแกรมที่สามารถนำไปประยุกต์ใช้กับหน่วยงานที่มีการเก็บพัสดุขนาดเล็ก–กลางได้ทันที

1.3.5 ผู้พัฒนาได้เรียนรู้การออกแบบระบบไฟล์ การจัดการข้อมูล และการประยุกต์ใช้ภาษา Python ในโครงงานจริง

1.4 เครื่องมือที่คาดว่าจะใช้

1.4.1 ภาษาที่ใช้ในการพัฒนาระบบ คือ Python สำหรับการพัฒนาโปรแกรมพื้นฐานที่สามารถเข้าใจได้ง่ายและมีไลบรารีที่หลากหลายช่วยให้สามารถจัดการข้อมูลได้ง่ายมากขึ้น

1.4.2 Visual Studio Code เป็นโปรแกรมประเภท Text Editor และ Integrated Development Environment (IDE) รองรับการเขียนโปรแกรมหลายภาษา เช่น Python, JavaScript, C, C++, Java, PHP, HTML, CSS และอื่น ๆ

1.4.3 GitHub เป็น แพลตฟอร์มบน Cloud ที่ใช้สำหรับเก็บและจัดการซอร์สโค้ดด้วยระบบควบคุมเวอร์ชัน (Version Control System) โดยใช้ Git ทำงานร่วมกันระหว่างทีมได้อย่างมีระบบ เช่น การสร้าง branch, pull request, code review

บทที่ 2
ระบบคลังพัสดุ

2.1 ตารางข้อมูลสินค้า items.bin

ตารางข้อมูลสินค้าประกอบด้วย 8 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญดังนี้

ฟิลด์	ชนิด	ขนาด(bytes)	ตัวอย่าง
flag	B	1	สถานะ 1=Active 0=Deleted
Item_id	I	4	รหัสพัสดุ (PK)
Name	30s	30	ชื่อพัสดุ เช่น “Computer Acer”
cat_id	I	4	หมวดที่อ้างถึงใน categories
qty	I	4	จำนวนคงเหลือปัจจุบัน
price_cents	I	4	ราคาต่อชิ้น (หน่วยสตางค์)
status	B	1	0=available 1=damaged 2=disposed

ตาราง 2.1 ตารางข้อมูลสินค้า

2.1.1 item_id รหัสพัสดุ

item_id เป็นรหัสประจำของพัสดุแต่ละชิ้นในระบบคลังพัสดุ ใช้เพื่อระบุและอ้างอิงรายการพัสดุอย่างเฉพาะเจาะจงไม่ซ้ำกันในแต่ละรายการ ข้อมูลนี้อยู่ในรูปแบบจำนวนเต็ม (integer) เช่น 1001, 1002, 1003 เป็นต้น การกำหนดรหัสพัสดุช่วยให้ระบบสามารถจัดการข้อมูลได้เป็นระเบียบ ค้นหา แก้ไข หรือลบข้อมูลของพัสดุแต่ละชิ้นได้รวดเร็วขึ้น รวมทั้งใช้เป็น Primary Key สำหรับเชื่อมโยงกับข้อมูลในตารางการเคลื่อนย้าย (movements.bin)

2.1.2 name ชื่อพัสดุ

name คือชื่อเรียกของพัสดุแต่ละรายการ เช่น “คอมพิวเตอร์ Acer” หรือ “สายไฟเบอร์ออปติก” เพื่อระบุชนิดของพัสดุให้เข้าใจง่าย จัดเก็บในรูปแบบข้อความ (string) ความยาวไม่เกิน 30 ตัวอักษร ฟิลด์นี้มีความสำคัญต่อการระบุพัสดุในรายงานและการค้นหา เนื่องจากชื่อจะใช้แสดงผลในหน้าจอดูข้อมูล, การออกใบเบิก หรือรายงานเคลื่อนไหวพัสดุ

2.1.3 cat_id รหัสหมวดหมู่

cat_id ใช้ระบุว่าพัสดุนั้นอยู่ในหมวดหมู่ใด โดยอ้างอิงถึงรหัสในไฟล์ categories.bin ข้อมูลนี้เป็นชนิดจำนวนเต็ม (integer) และทำหน้าที่เป็น Foreign Key เชื่อมโยงกับตารางหมวดหมู่ ช่วยให้ระบบสามารถจัดกลุ่มพัสดุและสรุปรายงานตามหมวดได้ เช่น “อุปกรณ์ไฟฟ้า”, “วัสดุสำนักงาน”, “เครื่องมือซ่อมบำรุง” เป็นต้น

2.1.4 qty จำนวนคงเหลือ

qty คือจำนวนพัสดุที่เหลืออยู่ในคลังปัจจุบันของแต่ละรายการ จัดเก็บเป็นจำนวนเต็ม (integer) และจะเปลี่ยนแปลงทุกครั้งที่มีการเคลื่อนย้ายพัสดุ เช่น เบิกออก (issue) หรือคืนเข้า (return) ระบบจะนำค่านี้ไปใช้ในการตรวจสอบสต็อกและในรายงานสรุปจำนวนพัสดุคงเหลือ

2.1.5 price_cents ราคาต่อหน่วย (หน่วย: สตางค์)

price_cents เก็บข้อมูลราคาต่อชิ้นของพัสดุในหน่วย “สตางค์” (integer) เพื่อป้องกันความคลาดเคลื่อนของทศนิยม เช่น ราคาจริง 125.50 บาท จะเก็บเป็น 12550 ข้อมูลนี้จะถูกใช้คำนวณมูลค่ารวมของพัสดุ (qty × price_cents) ในรายงานสรุปสินทรัพย์ของคลัง

2.1.6 status สถานะพัสดุ

status ใช้ระบุสถานะของพัสดุในระบบ โดยมีรหัสสถานะดังนี้

- 0 = available (พร้อมใช้งาน)
- 1 = damaged (ชำรุด)
- 2 = disposed (จำหน่ายออก)

2.1.7 flag ตัวบ่งชี้สถานะการใช้งาน

flag เป็นตัวบ่งชี้ว่าข้อมูล record นั้นยังถูกใช้งานอยู่หรือถูกลบไปแล้ว

- 1 = Active (ใช้งานปัจจุบัน)

- 0 = Deleted (ลบแบบ soft delete)

ใช้เพื่อควบคุมการแสดงผลของข้อมูลในระบบ และช่วยให้สามารถกู้คืนข้อมูลเก่าหรือจัดเก็บข้อมูลประวัติได้

2.1.8 next_free ตัวชี้ตำแหน่งว่าง

next_free ใช้เป็นตัวชี้ (pointer) สำหรับจัดการรายการที่ถูกลบในระบบไฟล์แบบ binary เพื่อให้สามารถนำพื้นที่ record เดิมกลับมาใช้ได้อีกครั้งโดยไม่ต้องขยายไฟล์ใหม่ ระบบจะเชื่อมโยงตำแหน่งว่างในรูปแบบ linked list ซึ่งช่วยให้จัดการหน่วยความจำอย่างมีประสิทธิภาพ

2.2 ตารางข้อมูลหมวดหมู่พัสดุ categories.bin

ตารางข้อมูลหมวดหมู่พัสดุประกอบด้วย 4 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญดังนี้

ฟิลด์	ชนิด	ขนาด(bytes)	ตัวอย่าง
flag	B	1	สถานะ 1=Active 0=Deleted
cat_id	I	4	รหัสหมวด (Primary Key)
name	30s	30	ชื่อหมวด เช่น “อุปกรณ์ไฟฟ้า”
desc	80s	80	รายละเอียดหมวด

ตารางที่ 2.2 ตารางข้อมูลหมวดหมู่พัสดุ

2.2.1 cat_id รหัสหมวดหมู่

cat_id เป็นรหัสประจำของหมวดหมู่พัสดุแต่ละประเภท ใช้สำหรับระบุและอ้างอิงหมวดหมู่ในระบบคลังพัสดุ เช่น “วัสดุสำนักงาน”, “อุปกรณ์ไฟฟ้า”, “เครื่องมือซ่อมบำรุง” เป็นต้น ข้อมูลนี้จะถูกเก็บในรูปแบบจำนวนเต็ม (integer) โดยจะไม่ซ้ำกันในแต่ละหมวดหมู่ เพื่อใช้เป็น Primary Key ในการเชื่อมโยงกับตารางพัสดุ (items.bin) การมีรหัสหมวดหมู่ช่วยให้ระบบสามารถจัดกลุ่มพัสดุได้ง่ายขึ้น และทำให้การสืบค้นหรือสรุปข้อมูลเป็นรายหมวดทำได้รวดเร็ว

2.2.2 name ชื่อหมวดหมู่

name คือชื่อของหมวดหมู่พัสดุในระบบ เช่น “วัสดุสำนักงาน”, “เครื่องใช้ไฟฟ้า”, หรือ “อุปกรณ์คอมพิวเตอร์” เป็นต้น จัดเก็บเป็นข้อความ (string) ความยาวไม่เกิน 30 ตัวอักษร ชื่อ

หมวดนี้จะถูกแสดงในหน้าจอหลักของระบบและรายงาน เพื่อช่วยให้ผู้ใช้งานเข้าใจประเภทของพัสดุได้โดยไม่ต้องจำรหัส

2.2.3 desc รายละเอียดหมวดหมู่

desc คือคำอธิบายเพิ่มเติมของหมวดหมู่ เช่น “วัสดุที่ใช้ในการบริหารสำนักงาน เช่น ปากกา กระดาษ ถ่านไฟฉาย” หรือ “เครื่องมือที่ใช้ในการบำรุงรักษาเครื่องจักร” เก็บในรูปแบบข้อความ (string) ความยาวไม่เกิน 80 ตัวอักษร ฟิลด์นี้ช่วยให้ผู้ดูแลระบบเข้าใจบริบทของหมวดหมู่นั้นมากขึ้น โดยเฉพาะในกรณีที่ชื่อหมวดมีลักษณะคล้ายกัน

2.2.4 flag ตัวบ่งชี้สถานะการใช้งาน

flag ใช้บอกว่าสถานะของข้อมูลหมวดหมู่นั้นยังถูกใช้งานอยู่หรือถูกลบไปแล้ว โดยระบบจะใช้แนวคิด soft delete คือไม่ลบข้อมูลจริง แต่เปลี่ยนค่าสถานะไว้แทน

- 1 = Active (ใช้งานอยู่)
- 0 = Deleted (ลบแล้ว)

ข้อมูลนี้ช่วยให้ระบบสามารถจัดการพื้นที่จัดเก็บ (free list) ได้โดยไม่ต้องสร้างไฟล์ใหม่

2.2.5 next_free ตัวชี้ตำแหน่งว่าง

next_free เป็นตัวแปรสำหรับเชื่อมโยงรายการที่ถูกลบในระบบไฟล์แบบ binary ให้กลายเป็น free-list (โครงสร้างข้อมูลแบบลิงก์รายการ) ระบบจะใช้ฟิลด์นี้เพื่อติดตามตำแหน่งของ record ที่สามารถนำกลับมาใช้งานใหม่ได้ ช่วยลดขนาดไฟล์และเพิ่มประสิทธิภาพการบันทึกข้อมูล

2.3 ตารางข้อมูลการเคลื่อนย้ายพัสดุmovements.bin

ตารางข้อมูลการเคลื่อนย้ายพัสดุประกอบด้วย 7 ฟิลด์หลักซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญดังนี้

ฟิลด์	ชนิด	ขนาด(bytes)	ตัวอย่าง
flag	B	1	สถานะ 1=Active 0=Deleted
move_id	I	4	รหัสการเคลื่อนย้าย (PK)
item_id	I	4	อ้างถึง item ที่เกี่ยวข้อง

qty	I	4	วันที่ YYYYMMDD เก็บเป็นจำนวนเต็ม
type	I	4	จำนวนที่ย้าย (บวก/ลบขึ้นอยู่กับประเภท)
operator	30s	30	0=issue 1=transfer 2=return 3=repair

ตารางที่ 2.3 ตารางข้อมูลการเคลื่อนย้ายพัสดุ

2.3.1 move_id รหัสการเคลื่อนย้าย

move_id เป็นรหัสประจำของรายการการเคลื่อนย้ายพัสดุแต่ละครั้งในระบบ เช่น การเบิกออก การโอนย้าย หรือการคืนเข้าเก็บข้อมูลนี้อยู่ในรูปแบบจำนวนเต็ม (integer) โดยไม่ซ้ำกัน เพื่อใช้เป็น Primary Key สำหรับอ้างอิงในระบบการมีรหัสเฉพาะช่วยให้สามารถตรวจสอบ ประวัติย้อนหลัง หรือสร้างรายงานการเคลื่อนไหวได้อย่างถูกต้องและไม่สับสนระหว่างแต่ละรายการ

2.3.2 item_id รหัสพัสดุ

item_id คือรหัสของพัสดุที่ถูกเคลื่อนย้ายในการทำรายการนั้น ๆ ซึ่งจะอ้างอิงกับรหัสในไฟล์ items.binฟิลด์นี้อยู่ในรูปแบบจำนวนเต็ม (integer) และทำหน้าที่เป็น Foreign Key ที่เชื่อมโยงไปยังรายการพัสดุ

ช่วยให้สามารถทราบได้ว่าการเคลื่อนไหวนี้เกี่ยวข้องกับพัสดุใด เช่น “คอมพิวเตอร์ Dell” หรือ “โต๊ะทำงานไม้”

2.3.3 ymd วันที่ทำรายการ

ymd คือวันที่ที่มีการเคลื่อนย้ายพัสดุ โดยจัดเก็บในรูปแบบจำนวนเต็ม 8 หลัก (YYYYMMDD) เช่น วันที่ 2025-10-05 จะถูกเก็บเป็นค่า 20251005 ฟิลด์นี้ช่วยให้สามารถเรียงลำดับเหตุการณ์ย้อนหลังได้อย่างถูกต้อง และใช้กรองข้อมูลรายวัน รายเดือน หรือรายปีในรายงานสรุป

2.3.4 qty ปริมาณการเคลื่อนย้าย

qty คือจำนวนพัสดุที่เกี่ยวข้องกับรายการนั้น ๆ (หน่วยเป็นชิ้น) เก็บในรูปแบบจำนวนเต็ม (integer) โดยจะตีความตามประเภทของการเคลื่อนย้าย เช่น

- ถ้าเป็นการเบิกออก (issue) ลดจำนวนคงเหลือในคลัง

- ถ้าเป็นการคืนเข้า (return) เพิ่มจำนวนคงเหลือในคลัง

- ถ้าเป็นการโอน (transfer) หรือซ่อม (repair) บันทึกไว้เฉย ๆ โดยไม่กระทบจำนวนในคลังหลัก

2.3.5 type ประเภทของการเคลื่อนย้าย

type ใช้ระบุว่ารายการนั้นเป็นการเคลื่อนย้ายประเภทใด โดยระบบใช้รหัสตัวเลขแทนข้อความ เพื่อให้ประมวลผลได้รวดเร็ว ฟิลด์นี้ใช้ร่วมกับ qty เพื่อคำนวณการเปลี่ยนแปลงของจำนวนพัสดุในแต่ละวัน

ค่า	ความหมาย	ตัวอย่างการใช้งาน
0	issue	เบิกพัสดุออกจากคลัง
1	transfer	โอนย้ายระหว่างแผนก
2	return	คืนพัสดุกลับเข้าคลัง
3	repair	ส่งซ่อม/อยู่ระหว่างบำรุงรักษา

ตารางที่ 2.4 ตารางข้อมูล type ประเภทของการเคลื่อนย้าย

2.3.6 operator ผู้ดำเนินการ

operator คือชื่อผู้ที่ทำรายการเคลื่อนย้ายพัสดุ เช่น “นายสมชาย เทียนทอง” หรือ “น.ส.นวลจันทร์ พนักงานคลัง” จัดเก็บเป็นข้อความ (string) ความยาวไม่เกิน 30 ตัวอักษร ฟิลด์นี้ช่วยให้สามารถตรวจสอบความรับผิดชอบของผู้ปฏิบัติงานได้ เช่น ใครเป็นผู้เบิก ใครเป็นผู้คืน หรือใครเป็นผู้อนุมัติรายการนั้น

2.3.7 flag สถานะการใช้งานของรายการ

flag เป็นตัวบ่งชี้ว่าสถานะของ record นั้นยังใช้งานอยู่หรือถูกลบไปแล้ว ข้อมูลนี้ใช้สำหรับการจัดการพื้นที่จัดเก็บในไฟล์ไบนารี (free-list) เพื่อให้สามารถนำช่องว่างกลับมาใช้ได้โดยไม่ต้องขยายไฟล์

1 = Active (ยังใช้งานอยู่)

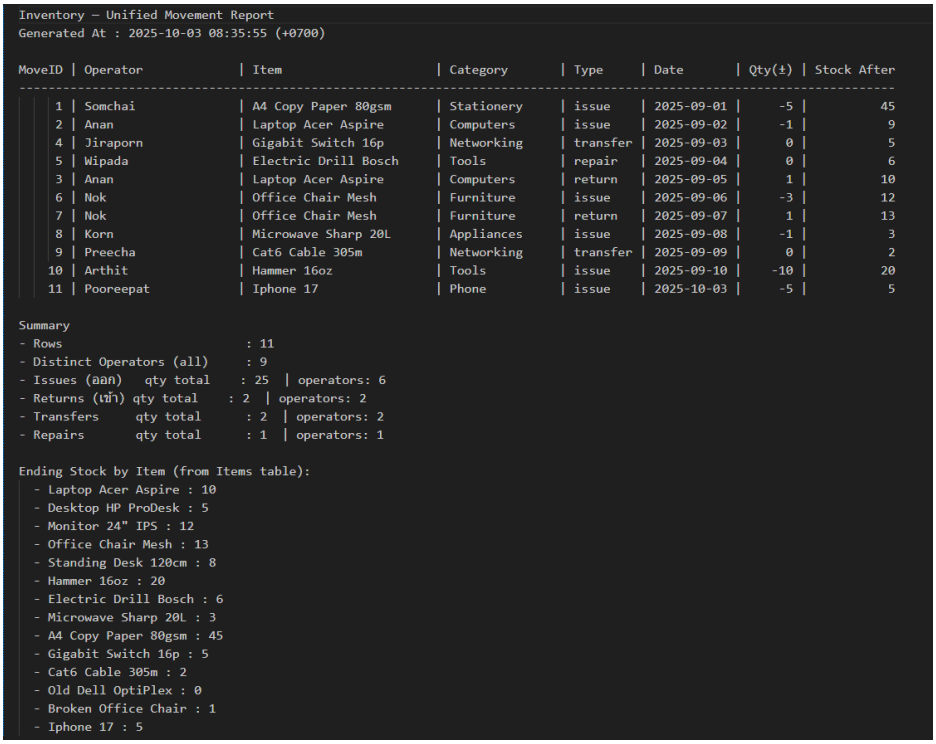
0 = Deleted (ลบออกจากระบบแบบ soft delete)

2.3.8 next_free ตัวชี้ตำแหน่งว่าง (Free Pointer)

next_free ใช้เก็บตำแหน่ง record ถัดไปที่ถูกลบ เพื่อให้ระบบสามารถจัดการช่องว่างในไฟล์อย่างมีประสิทธิภาพ ข้อมูลนี้จะไม่แสดงผลต่อผู้ใช้โดยตรง แต่ใช้ในระดับโครงสร้างของระบบเพื่อเพิ่มความเร็วในการเขียนข้อมูลใหม่

2.4 ไฟล์ inventory_report.txt

ไฟล์ inventory_report.txt ของระบบจัดการคลังสินค้าประกอบด้วย 7 ฟิลด์แต่ละฟิลด์มีรายละเอียดและความสำคัญดังนี้



```
Inventory — Unified Movement Report
Generated At : 2025-10-03 08:35:55 (+0700)

MoveID | Operator            | Item                  | Category      | Type     | Date       | Qty(±) | Stock After
-------------------------------------------------------------------------------------------------------------------
     1 | Somchai             | A4 Copy Paper 80gsm   | Stationery    | issue    | 2025-09-01 |     -5 |          45
     2 | Anan                | Laptop Acer Aspire    | Computers     | issue    | 2025-09-02 |     -1 |           9
     4 | Jiraporn            | Gigabit Switch 16p    | Networking    | transfer | 2025-09-03 |      0 |           5
     5 | Wipada              | Electric Drill Bosch  | Tools         | repair   | 2025-09-04 |      0 |           6
     3 | Anan                | Laptop Acer Aspire    | Computers     | return   | 2025-09-05 |      1 |          10
     6 | Nok                 | Office Chair Mesh     | Furniture     | issue    | 2025-09-06 |     -3 |          12
     7 | Nok                 | Office Chair Mesh     | Furniture     | return   | 2025-09-07 |      1 |          13
     8 | Korn                | Microwave Sharp 20L   | Appliances    | issue    | 2025-09-08 |     -1 |           3
     9 | Preecha             | Cat6 Cable 305m       | Networking    | transfer | 2025-09-09 |      0 |           2
    10 | Arthit              | Hammer 16oz           | Tools         | issue    | 2025-09-10 |    -10 |          20
    11 | Pooreepat           | Iphone 17             | Phone         | issue    | 2025-10-03 |     -5 |           5

Summary
- Rows                       : 11
- Distinct Operators (all)   : 9
- Issues (ออก)   qty total   : 25  | operators: 6
- Returns (เข้า) qty total   : 2  | operators: 2
- Transfers      qty total   : 2  | operators: 2
- Repairs        qty total   : 1  | operators: 1

Ending Stock by Item (from Items table):
  - Laptop Acer Aspire : 10
  - Desktop HP ProDesk : 5
  - Monitor 24" IPS : 12
  - Office Chair Mesh : 13
  - Standing Desk 120cm : 8
  - Hammer 16oz : 20
  - Electric Drill Bosch : 6
  - Microwave Sharp 20L : 3
  - A4 Copy Paper 80gsm : 45
  - Gigabit Switch 16p : 5
  - Cat6 Cable 305m : 2
  - Old Dell OptiPlex : 0
  - Broken Office Chair : 1
  - Iphone 17 : 5
```

รูปที่ 2-1 ไฟล์ inventory_report.txt

2.4.1 header_text ส่วนหัวรายงาน

เป็นฟิลด์ข้อความ (string ขนาดไม่เกิน 80 bytes) ใช้เก็บข้อความชื่อรายงาน เช่น

Inventory – Unified Movement Report

วัตถุประสงค์คือแสดงชื่อและประเภทของรายงาน เพื่อให้ผู้อ่านเข้าใจได้ทันทีว่าเป็นรายงานรวมข้อมูลการเคลื่อนไหวพัสดุของระบบ

2.4.2 generated_at วันที่และเวลาที่สร้างรายงาน

เป็นฟิลด์ข้อความ (string ขนาด 30 bytes) ใช้ระบุวันที่และเวลาที่สร้างรายงานในรูปแบบ YYYY-MM-DD HH:MM:SS (+0700) เช่น Generated at : 2025-10-03 08:35:55 (+0700) ซึ่งช่วยบอกเวลาอ้างอิงในการออกรายงานแต่ละครั้ง เพื่อความถูกต้องของข้อมูล

2.4.3 table_header หัวตารางการเคลื่อนไหว

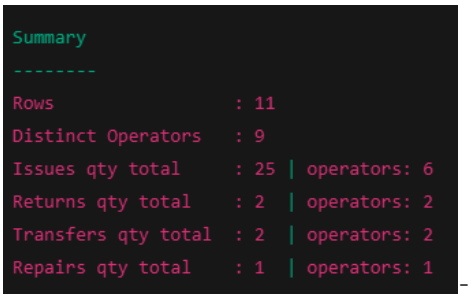
เป็นฟิลด์ข้อความ (string ขนาด 100 bytes) ใช้ระบุหัวข้อของแต่ละคอลัมน์ในตาราง เช่น MoveID | Operator | Item | Category | Type | Date | Qty(±) | Stock After เพื่อกำหนดโครงสร้างการจัดวางข้อมูลภายในรายงานให้อ่านง่ายและเป็นระเบียบ

2.4.4 movement_records ข้อมูลการเคลื่อนไหว

เป็นฟิลด์ข้อความแบบหลายบรรทัด (string N bytes; N ขึ้นอยู่กับจำนวนแถวข้อมูล) ใช้เก็บรายละเอียดการเคลื่อนไหวของพัสดุแต่ละรายการ เช่น

2.4.5 summary_section ส่วนสรุปข้อมูล

เป็นฟิลด์ข้อความ (string ประมาณ 200 bytes) ใช้สำหรับการสรุปข้อมูลว่ามีจำนวนคนที่ใช้สินค้าทั้งหมดกี่คนและทำอะไรกับสินค้าบ้าง



รูปที่ 2-2 summary_section ส่วนสรุปข้อมูล

2.4.6 ending_stock_section ส่วนสรุปจำนวนคงเหลือ

เป็นฟิลด์ข้อความ (string แบบหลายบรรทัด; N bytes) ใช้เก็บข้อมูลพัสดุคงเหลือทั้งหมดจากไฟล์ items.bin เช่น

```
Ending Stock by Item (from Items table):
- Laptop Acer Aspire : 10
- A4 Copy Paper 80gsm : 45
- Microwave Sharp 20L : 3
- iPhone 17 : 5
```

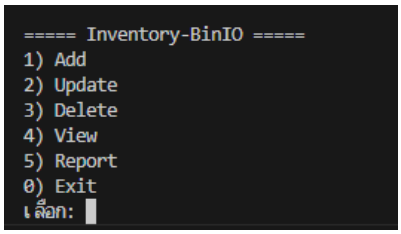
รูปที่ 2-3 ending_stock_section ส่วนสรุปจำนวนคงเหลือ

บทที่ 3
การใช้งานระบบคลังพัสดุ

โปรแกรมระบบคลังพัสดุ (Inventory-BinIO) เป็นระบบที่ใช้สำหรับจัดการข้อมูลพัสดุภายในหน่วยงานหรือองค์กร ให้การจัดเก็บและติดตามข้อมูลของหมวดหมู่ รายการพัสดุ และประวัติการเคลื่อนย้ายเป็นไปอย่างเป็นระบบระบบถูกออกแบบให้ใช้งานง่าย มีเมนูหลักและเมนูย่อยสำหรับเพิ่ม แก้ไข ลบ ค้นหา และสร้างรายงานสรุปข้อมูลได้อย่างสะดวก โดยข้อมูลทั้งหมดจะถูกบันทึกในรูปแบบไฟล์ Binary เพื่อให้การเข้าถึงและประมวลผลมีความรวดเร็ว

3.1 การใช้งานโปรแกรมระบบคลังพัสดุ

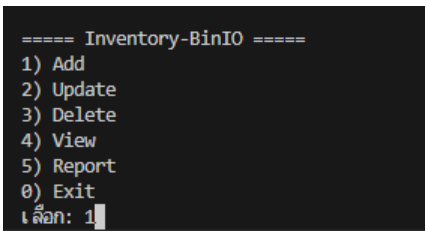
3.1.1 เมื่อเปิดโปรแกรมขึ้นมาผู้ใช้จะพบกับเมนูหลักดังภาพ ซึ่งเป็นศูนย์รวมคำสั่งทั้งหมดของระบบคลังพัสดุผู้ใช้สามารถเลือกทำงานในแต่ละส่วนได้ เช่น เพิ่มข้อมูลพัสดุ แก้ไขข้อมูล ลบข้อมูล หรือสร้างรายงานสรุปพัสดุได้จากเมนูนี้



รูปที่ 3-1 เปิดโปรแกรม

3.1.2 เมนู Add (เพิ่มข้อมูล)

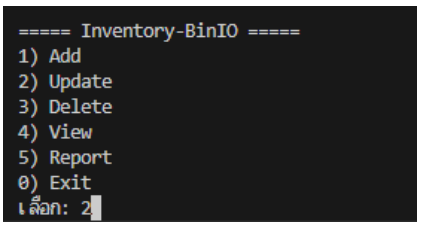
เมื่อเลือกเมนู Add ระบบจะแสดงเมนูย่อยสำหรับการเพิ่มข้อมูลใหม่ในระบบประกอบด้วยเพิ่มหมวดหมู่พัสดุ (Category) เพิ่มรายการพัสดุ (Item) เพิ่มการเคลื่อนย้ายพัสดุ (Movement)



รูปที่ 3-2 เมนู Add (เพิ่มข้อมูล)

3.1.3 เมนู Update (แก้ไขข้อมูล)

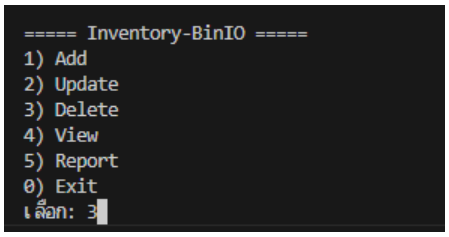
เมนูนี้ใช้สำหรับปรับปรุงหรือแก้ไขข้อมูลที่มีอยู่ในระบบ เช่น แก้ชื่อหมวดหมู่ แก้ปริมาณพัสดุ หรืออัปเดตราคาผู้ใช้สามารถเลือกได้ว่าต้องการแก้ไขหมวดหมู่หรือรายการพัสดุ



รูปที่ 3-3 เมนู Update (แก้ไขข้อมูล)

3.1.4 เมนู Delete (ลบข้อมูล)

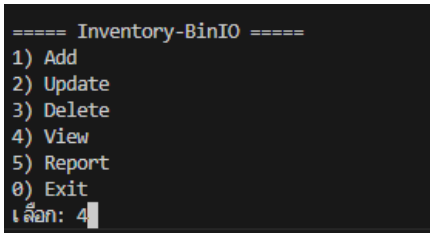
เมนูนี้ใช้สำหรับลบข้อมูลที่ไม่ต้องการใช้งาน โดยระบบจะไม่ลบข้อมูลจริงออกจากไฟล์ แต่จะเปลี่ยนสถานะเป็น “Deleted” เพื่อความปลอดภัยของข้อมูล สามารถเลือกได้ว่าจะลบหมวดหมู่ รายการพัสดุ หรือประวัติการเคลื่อนไหว



รูปที่ 3-4 เมนู Delete (ลบข้อมูล)

3.1.5 เมนู View (ดูข้อมูล)

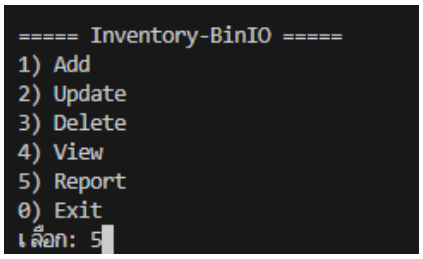
เมนูนี้ใช้สำหรับเรียกดูข้อมูลในระบบ มีให้เลือกดูแบบเดี่ยว ดูทั้งหมด ดูแบบกรองเงื่อนไข หรือดูสถิติพัสดุรวม



รูปที่ 3-5 เมนู View (ดูข้อมูล)

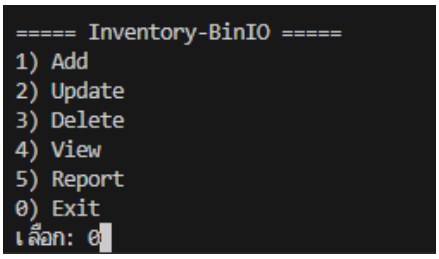
3.1.6 เมนู Report (สร้างรายงาน)

เมนูนี้ใช้สำหรับสร้างรายงานสรุปในรูปแบบข้อความ (inventory_report.txt หรือ inventory_unified_report.txt) รายงานจะแสดงข้อมูลพัสดุทั้งหมดในระบบ เช่น รหัสพัสดุ ชื่อรายการหมวดหมู่ ราคา ปริมาณคงเหลือ และประวัติการเคลื่อนไหวล่าสุด



รูปที่ 3-6 เมนู Report (สร้างรายงาน)

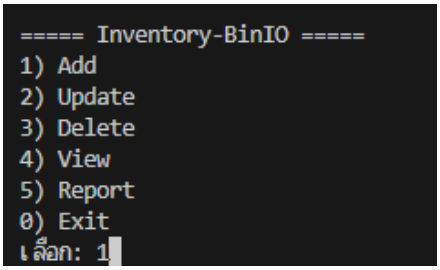
3.1.7 เมนู Exit เพื่อออกจากโปรแกรม



รูปที่ 3-7 เมนู Exit เพื่อออกจากโปรแกรม

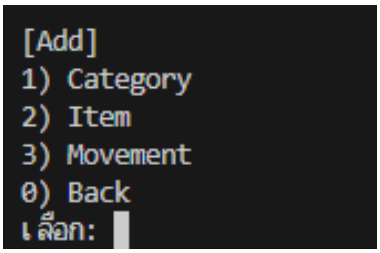
3.2 การใช้งานฟังก์ชันเพิ่มข้อมูล

3.2.1 กรอกหมายเลขที่ 1 เพื่อทำการเข้าใช้งานฟังก์ชัน เพิ่มสินค้า



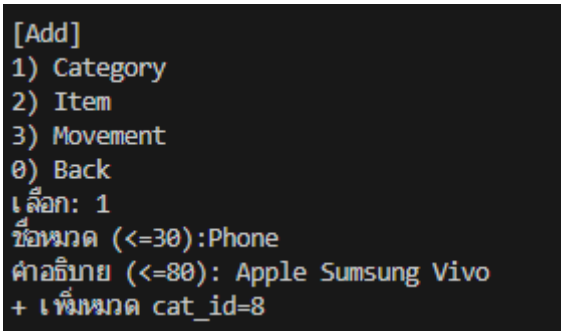
รูปที่ 3-8 การเลือกใช้งานฟังก์ชัน Add

3.2.2 เมื่อกรอกหมายเลข 1 จะปรากฏฟังก์ชัน Add ขึ้นมา โดยจะมีทั้ง 4 หัวข้อ คือ เพิ่มหมวดหมู่พัสดุ (Category) เพิ่มรายการพัสดุ (Item) เพิ่มการเคลื่อนย้ายพัสดุ (Movement)



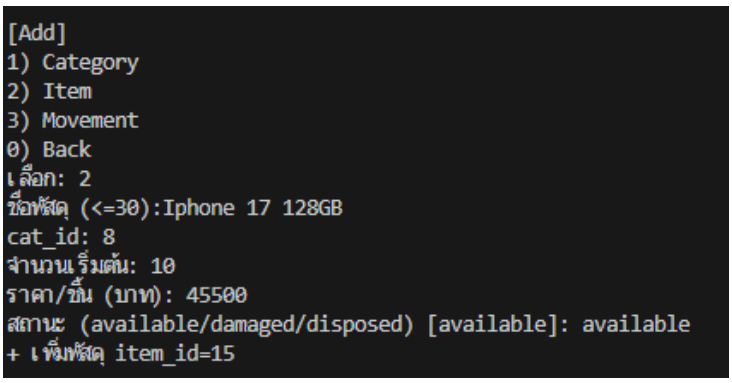
รูปที่ 3-9 ฟังก์ชัน Add

3.2.3 เมื่อกรอกหมายเลข 1 เข้าไปที่ฟังก์ชัน add จะทำการเข้าสู่การเพิ่มหมวดหมู่พัสดุ โดยจะให้กรอกข้อมูลตามภาพ



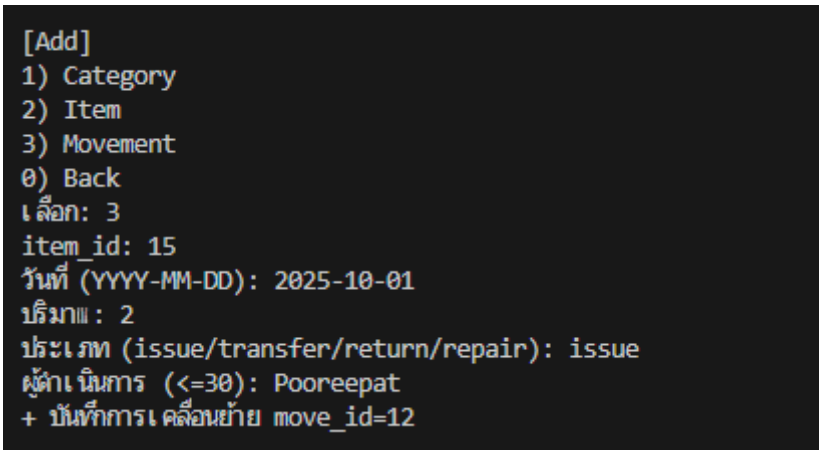
รูปที่ 3-10 Add Category

3.2.4 เมื่อกรอกหมายเลข 2 เข้าไปที่ฟังก์ชัน add จะทำการเข้าสู่การเพิ่มรายการพัสดุ โดยจะให้กรอกข้อมูลตามภาพ



รูปที่ 3-11 Add Item

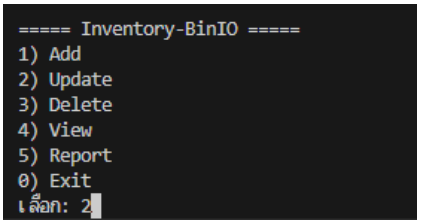
3.2.3 เมื่อกรอกหมายเลข 2 เข้าไปที่ฟังก์ชัน add จะทำการเข้าสู่การเพิ่มการเคลื่อนย้ายพัสดุ โดยจะให้กรอกข้อมูลตามภาพ



รูปที่ 3-12 Add Movement

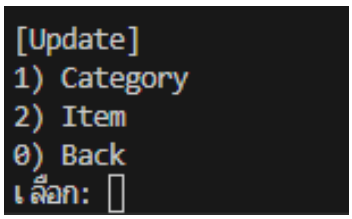
3.3 การใช้งานฟังก์ชันอัพเดตข้อมูล

3.3.1 เมื่อกรอกหมายเลข 2 เพื่อทำการเข้าใช้งานฟังก์ชัน อัพเดตสินค้า



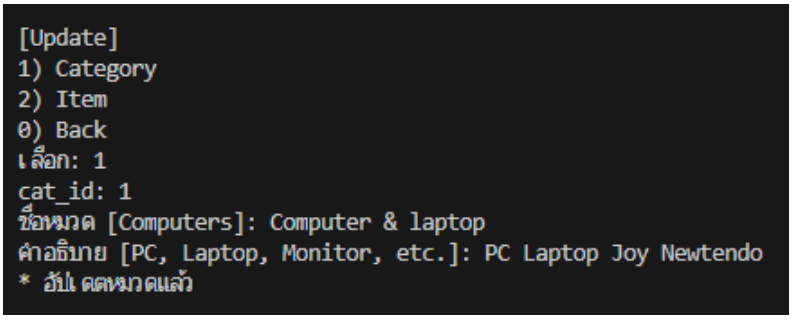
รูปที่ 3-13 การเลือกใช้งานฟังก์ชัน อัพเดตสินค้า

3.3.2 เมื่อกรอกหมายเลข 2 จะปรากฏฟังก์ชัน Update ขึ้นมา โดยจะมีทั้ง 2 หัวข้อ คือ อัพเดตหมวดหมู่พัสดุ (Category) อัพเดตรายการพัสดุ (Item)



รูปที่ 3-14 ฟังก์ชัน Update

3.3.3 เมื่อกรอกหมายเลข 1 เข้าไปที่ฟังก์ชัน Update จะทำการเข้าสู่การอัพเดตหมวดหมู่พัสดุ โดยจะให้กรอกข้อมูลตามภาพ



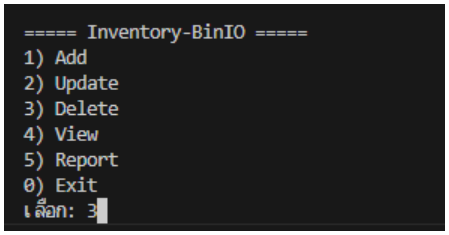
รูปที่ 3-15 Update category

3.3.4 เมื่อกรอกหมายเลข 2 เข้าไปที่ฟังก์ชัน Update จะทำการเข้าสู่การอัพเดตรายการพัสดุ โดยจะให้กรอกข้อมูลตามภาพ

รูปที่ 3-16 Update item

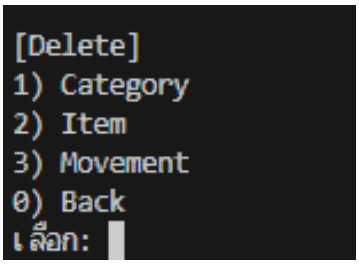
3.4 การใช้งานฟังก์ชันลบข้อมูล

3.4.1 เมื่อกรอกหมายเลข 3 เพื่อทำการเข้าใช้งานฟังก์ชัน ลบข้อมูล



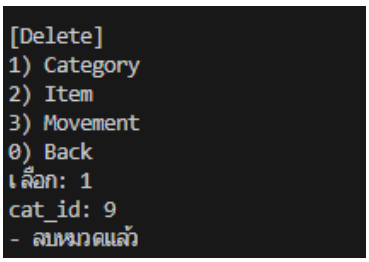
รูปที่ 3-17 การเลือกใช้งานฟังก์ชัน ลบข้อมูล

3.4.2 เมื่อกรอกหมายเลข 3 จะปรากฏฟังก์ชัน Update ขึ้นมา โดยจะมีทั้ง 3 หัวข้อ คือ ลบหมวดหมู่พัสดุ (Category) ลบรายการพัสดุ (Item) ลบความเคลื่อนไหวของสินค้า (movement)



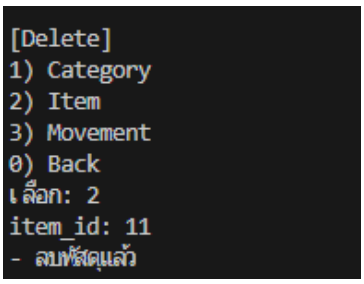
รูปที่ 3-18 ฟังก์ชัน Delete

3.4.3 เมื่อกรอกหมายเลข 1 เข้าไปที่ฟังก์ชัน Delete จะทำการเข้าสู่การลบหมวดหมู่พัสดุ โดยจะให้กรอกข้อมูลตามภาพ



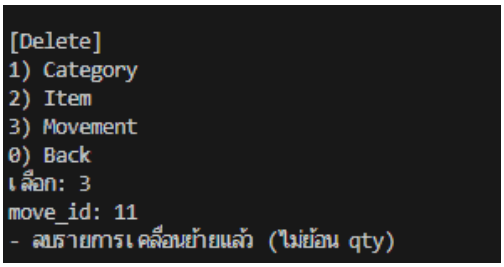
รูปที่ 3-19 Delete category

3.4.4 เมื่อกรอกหมายเลข 2 เข้าไปที่ฟังก์ชัน Delete จะทำการเข้าสู่การลบรายการพัสดุ โดยจะให้กรอกข้อมูลตามภาพ



รูปที่ 3-20 Delete item

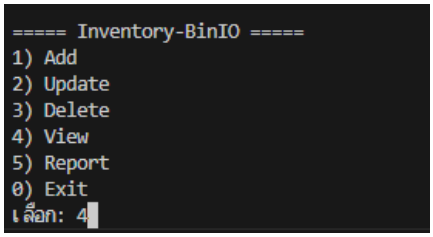
3.4.5 เมื่อกรอกหมายเลข 2 เข้าไปที่ฟังก์ชัน Delete จะทำการเข้าสู่การลบความเคลื่อนไหวของพัสดุ โดยจะให้กรอกข้อมูลตามภาพ



รูปที่ 3-21 Delete movement

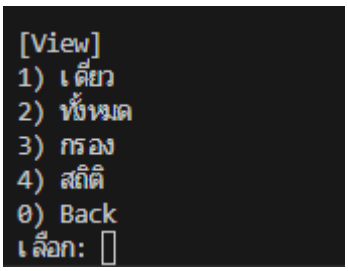
3.5 การใช้งานฟังก์ชันอัพเดตข้อมูล

3.5.1 เมื่อกรอกหมายเลข 4 เพื่อทำการเข้าใช้งานฟังก์ชัน ลบสินค้า



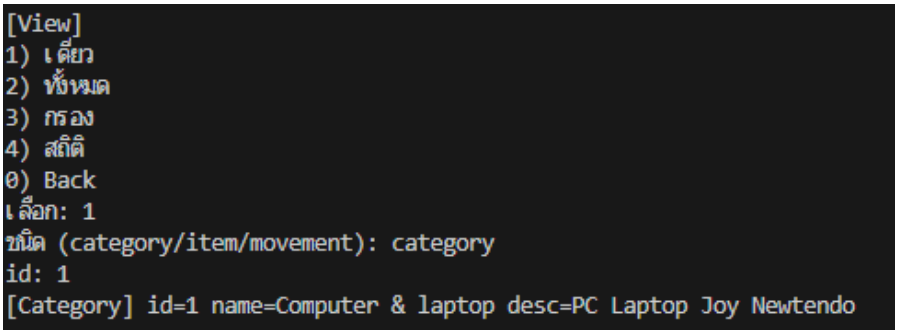
รูปที่ 3-22 การเลือกใช้งานฟังก์ชัน ลบสินค้า

3.5.2 เมื่อกรอกหมายเลข 3 จะปรากฏฟังก์ชัน View ขึ้นมา โดยจะมีทั้ง 4 หัวข้อ คือ เดี่ยว ทั้งหมด กรอง สถิติ



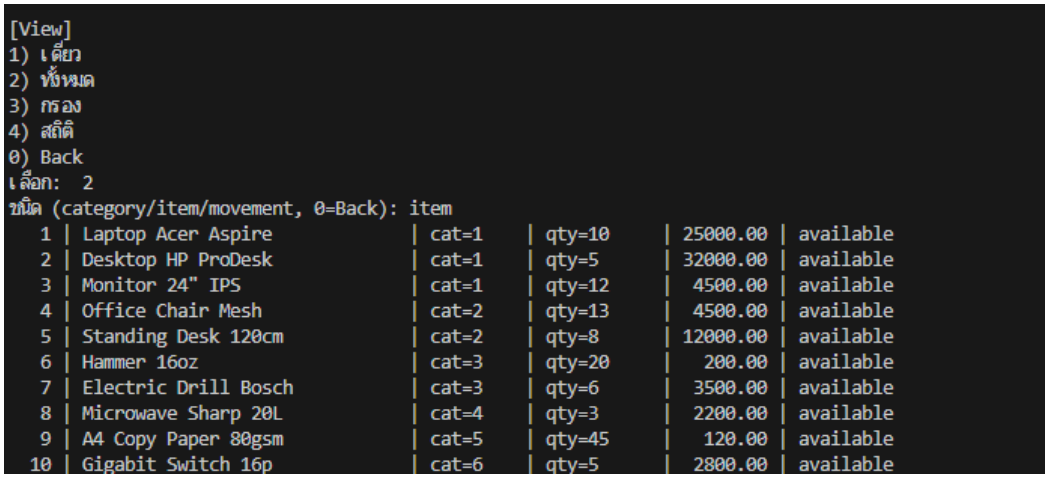
รูปที่ 3-23 ฟังก์ชัน View

3.5.3 เมื่อกรอกหมายเลข 1 เข้าไปที่ฟังก์ชัน View จะทำการเข้าสู่การ ดูข้อมูลแบบเดี่ยว โดยจะมีให้เลือกดูทั้งเมนู category item movement



รูปที่ 3-24 View (เดี่ยว)

3.5.4 เมื่อกรอกหมายเลข 2 เข้าไปที่ฟังก์ชัน View จะทำการเข้าสู่การ ดูข้อมูลทั้งหมด โดยจะมีให้เลือกดูทั้งเมนู category item movement



รูปที่ 3-25 View (ทั้งหมด)

3.5.5 เมื่อกรอกหมายเลข 3 เข้าไปที่ฟังก์ชัน View จะทำการเข้าสู่การ ดูข้อมูลแบบกรอง โดยจะมีให้เลือกดูทั้งเมนู category item movement

```
[View]
1) เดี่ยว
เลือก: 3
ชนิด (category/item/movement, 0=Back): item
สถานะ (available/damaged/disposed หรือเว้นว่าง):
กรอง cat_id (เว้นว่าง=ทั้งหมด): 1
ค้นหาชื่อหรัสดุ (เว้นว่าง=ไม่กรอง):
   1 | Laptop Acer Aspire          | cat=1    | qty=10     | 25000.00 | available
   2 | Desktop HP ProDesk          | cat=1    | qty=5      | 32000.00 | available
   3 | Monitor 24" IPS             | cat=1    | qty=12     |  4500.00 | available
  12 | Old Dell OptiPlex           | cat=1    | qty=0      |  1000.00 | disposed
```

รูปที่ 3-26 View (กรอง)

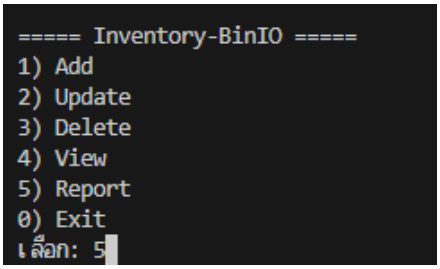
3.5.6 เมื่อกรอกหมายเลข 4 เข้าไปที่ฟังก์ชัน View จะทำการเข้าสู่การ ดูข้อมูลแบบสถิติ โดยจะแสดงข้อมูลอกมาเป็นสถิติทั้งหมด

```
[View]
1) เดี่ยว
2) ทั้งหมด
3) กรอง
4) สถิติ
0) Back
เลือก:  4
Items by status:
  available = 12
  damaged   = 1
  disposed = 1
Total Qty = 146
Total Value = 1,456,500.00 THB
```

รูปที่ 3-27 View (สถิติ)

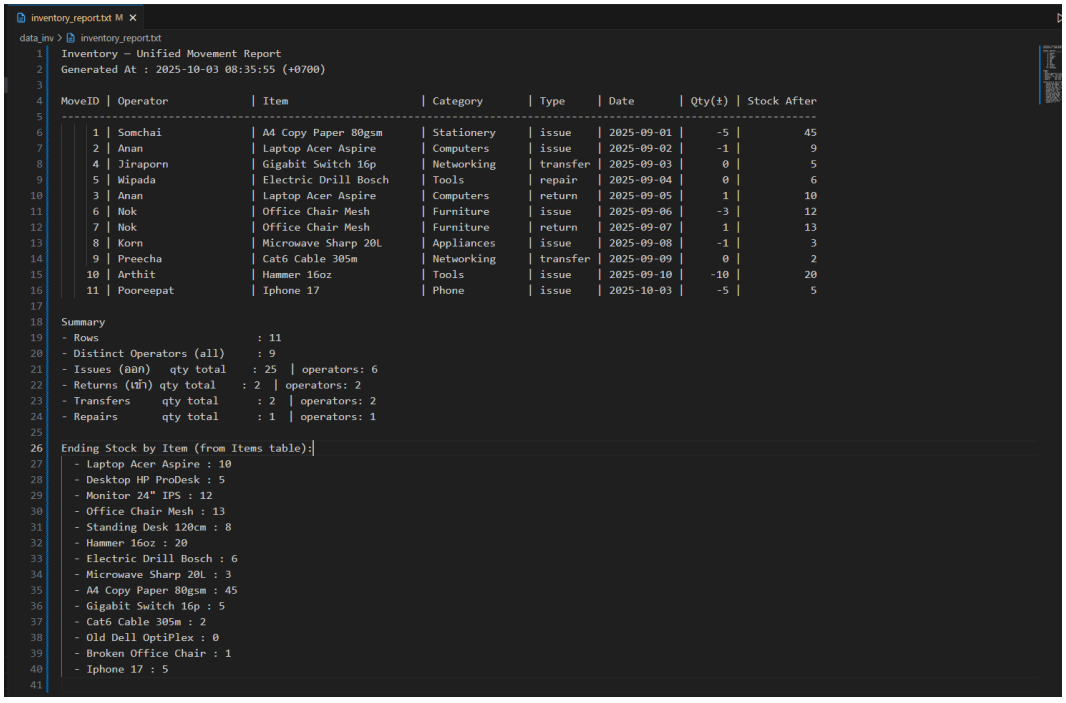
3.6 การใช้งานฟังก์ชัน Report

3.6.1 เมื่อกรอกหมายเลข 5 เพื่อทำการเข้าใช้งานฟังก์ชัน Report



รูปที่ 3-28 การเลือกใช้งานฟังก์ชัน Report

3.6.1 เมื่อกรอกหมายเลข 5 เพื่อทำการเข้าใช้งานฟังก์ชัน Report แล้วจะแสดงไฟล์ inventory_report.txt ออกมา และข้างในจะมีข้อมูลสรุปจากตารางต่างๆ



รูปที่ 3-29 หน้า Report

บทที่ 4

อธิบายการทำงานของ ฉode

4.1 ไลบรารีพื้นฐานในระบบคลังสินค้า

4.1.1 from __future__ import annotations คำสั่งนี้ใช้เพื่อเปิดโหมด “เลื่อนการตีความ type hints” (postponed evaluation of annotations) ทำให้ตัว annotation (เช่นชื่อคลาสหรือ type ของตัวแปร) ที่อ้างถึงในโค้ดสามารถใช้ได้แม้จะถูกประกาศภายหลังซึ่งมีประโยชน์มากเมื่อทำงานกับ dataclass หรือ class ที่อ้างอิงถึงกันเอง

```
from __future__ import annotations
```

รูปที่ 4-1 module annotations

4.1.2 import os โมดูล **os** เป็นเครื่องมือสำหรับติดต่อกับระบบปฏิบัติการ (Operating System Interface) ใช้สำหรับตรวจสอบหรือจัดการไฟล์และโฟลเดอร์ เช่น ตรวจสอบว่าโฟลเดอร์มีอยู่จริงหรือไม่ (os.path.isdir) สร้างโฟลเดอร์ใหม่ (os.makedirs) รวมพาธของไฟล์ (os.path.join) บังคับให้เขียนข้อมูลลงดิสก์จริง (os.fsync)

```
import os
```

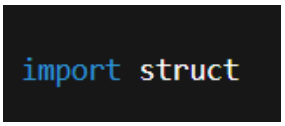
รูปที่ 4-2 module os

4.1.3 import sys โมดูล sys ใช้สำหรับเข้าถึงระดับระบบของ Python โดยเฉพาะการจัดการกับการทำงานของโปรแกรม เช่น การออกจากโปรแกรม (sys.exit()), การรับค่า argument จากบรรทัดคำสั่ง (sys.argv) และการเขียนข้อความแสดงผลข้อผิดพลาดไปยัง stderr

```
import sys
```

รูปที่ 4-3 module sys

4.1.4 import struct โมดูล struct เป็นหัวใจสำคัญของระบบนี้ ใช้สำหรับแปลงข้อมูล Python ให้เป็น “ไฟล์แบบบันทึกคงที่” (Fixed-Length Binary Record) โดยใช้ฟังก์ชัน pack() และ unpack() เพื่อจัดการข้อมูลชนิดต่าง ๆ เช่น int, float, string ให้เป็น bytes และกำหนดรูปแบบการจัดเรียงข้อมูลแบบ Little-endian (<) เพื่อให้ไฟล์มีขนาดและโครงสร้างคงที่ทุกครั้ง

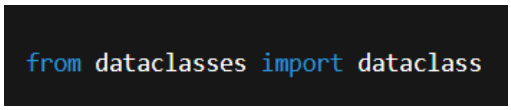


รูปที่ 4-4 module struct

4.1.5 import argparse โมดูล argparse ใช้สำหรับจัดการพารามิเตอร์จากบรรทัดคำสั่ง (Command-Line Arguments) เช่น การกำหนดค่า --data-dir เพื่อระบุโฟลเดอร์เก็บข้อมูล .bin ช่วยให้ผู้ใช้สามารถรันโปรแกรมในหลายโฟลเดอร์ข้อมูลได้โดยไม่ต้องแก้ไขโค้ด

รูปที่ 4-5 module argparse

4.1.6 from dataclasses import dataclass โมดูล dataclasses ใช้สำหรับประกาศคลาสที่เก็บข้อมูล (Data Container Class) ได้อย่างกระชับ โดยไม่ต้องเขียน __init__() เอง เหมาะสำหรับข้อมูลที่มีโครงสร้างตายตัว เช่น Header และ IndexSlot ของไฟล์ไบนารี



รูปที่ 4-6 module dataclasses

4.1.7 from datetime import datetime, date โมดูล datetime ใช้ในการจัดการวันเวลา เช่น บันทึกวันที่และเวลาที่สร้างไฟล์ (created_at) บันทึกเวลาที่อัปเดตข้อมูลล่าสุด (updated_at) แปลงวันที่จากรูปแบบ YYYY-MM-DD เป็นตัวเลข (ymd_to_int) และกลับคืน (int_to_ymd) ในระบบคลังพัสดุจะใช้ datetime เพื่อกำหนดวันที่ของการเคลื่อนไหวพัสดุ (Movement) และเวลาที่สร้างรายงาน

```
from datetime import datetime, date
```

รูปที่ 4-7 module datetime

4.1.8 from typing import Optional, Iterable, Tuple, Dict, Any โมดูล typing ใช้สำหรับประกาศชนิดของข้อมูล (Type Hint) เพื่อเพิ่มความชัดเจนให้กับโค้ดและช่วยให้ตรวจสอบข้อผิดพลาดได้ง่าย ในโปรแกรมนี้ใช้กับการนิยามชนิดของตัวแปรในฟังก์ชันต่าง ๆ เช่น

```
from typing import Optional, Iterable, Tuple, Dict, Any
```

รูปที่ 4-8 module typing

4.2 ฟังก์ชันเมนูระบบคลังสินค้า

ฟังก์ชัน run() เป็นหัวใจหลักของโปรแกรม Inventory-BinIO ทำหน้าที่ควบคุมและแสดง “เมนูหลัก” ของระบบคลังพัสดุ ซึ่งเป็นส่วนที่ผู้ใช้โต้ตอบกับโปรแกรมโดยตรง เมื่อโปรแกรมเริ่มทำงาน ฟังก์ชันนี้จะเข้าสู่ลูปหลัก (while True) เพื่อวนแสดงเมนูให้เลือกการทำงานที่ต้องการ โดยมี 6 ตัวเลือกคือ

4.2.1 Add – เพิ่มข้อมูลใหม่ เช่น หมวดหมู่สินค้า (Category) รายการพัสดุ (Item) หรือข้อมูลการเคลื่อนไหว (Movement)

4.2.2 Update – แก้ไขข้อมูลที่มีอยู่ เช่น เปลี่ยนชื่อหมวดหมู่หรือปรับปรุงรายละเอียดของพัสดุ

4.2.3 Delete – ลบข้อมูลออกจากระบบ เช่น หมวดหมู่ พัสดุ หรือรายการเคลื่อนไหว

4.2.4 View – เรียกดูข้อมูลจากไฟล์ทั้งหมด สามารถเลือกดูรายเดียว ดูทั้งหมด กรองข้อมูล หรือดูสถิติ

4.2.5 Report – สร้างรายงานสรุปข้อมูลจากทั้งสามไฟล์ (categories, items, movements) แล้วบันทึกเป็นไฟล์ inventory_report.txt

4.2.6 Exit – บันทึกข้อมูลล่าสุดทั้งหมด ปิดไฟล์ และออกจากโปรแกรม

```python
# ---------- Menu ----------
def run(self):
    while True:
        print("\n===== Inventory-BinIO =====")
        print("1) Add  \n2) Update  \n3) Delete  \n4) View  \n5) Report  \n0) Exit")
        c = (input('เลือก: ') or '0').strip().lower()
        try:
            if c == '1':
                while True:
                    print("\n[Add] \n1) Category \n2) Item \n3) Movement  \n0) Back")
                    ch = input('เลือก: ').strip().lower()
                    if ch in ('0', 'b', 'back'): break
                    {'1': self.add_category,
                     '2': self.add_item,
                     '3': self.add_movement}.get(ch, lambda: print('ตัวเลือกไม่ถูกต้อง'))()
            elif c == '2':
                while True:
                    print("\n[Update] \n1) Category \n2) Item  \n0) Back")
                    ch = input('เลือก: ').strip().lower()
                    if ch in ('0', 'b', 'back'): break
                    {'1': self.update_category,
                     '2': self.update_item}.get(ch, lambda: print('ตัวเลือกไม่ถูกต้อง'))()
            elif c == '3':
                while True:
                    print("\n[Delete] \n1) Category \n2) Item \n3) Movement  \n0) Back")
                    ch = input('เลือก: ').strip().lower()
                    if ch in ('0', 'b', 'back'): break
                    {'1': self.delete_category,
                     '2': self.delete_item,
                     '3': self.delete_movement}.get(ch, lambda: print('ตัวเลือกไม่ถูกต้อง'))()
            elif c == '4':
                while True:
                    print("\n[View] \n1) เดียว \n2) ทั้งหมด \n3) กรอง \n4) สถิติ  \n0) Back")
                    ch = input('เลือก: ').strip().lower()
                    if ch in ('0', 'b', 'back'): break
                    {'1': self.view_single,
                     '2': self.view_all,
                     '3': self.view_filter,
                     '4': self.view_stats}.get(ch, lambda: print('ตัวเลือกไม่ถูกต้อง'))()
            elif c == '5':
                out = os.path.join(os.path.dirname(self.cats.path), 'inventory_report.txt')
                self.generate_report(out)
            elif c == '0':
                out = os.path.join(os.path.dirname(self.cats.path), 'inventory_report.txt')
                self.generate_report(out)
                print('บันทึกและออก...')
                self.close()
                break
            else:
                print('ตัวเลือกไม่ถูกต้อง')
        except Exception as e:
            print('! error:', e)
```

รูปที่ 4-9 code menu

4.3 ฟังก์ชันเพิ่มข้อมูล

4.3.1 ฟังก์ชัน add_category

ฟังก์ชัน add_category(self) ทำหน้าที่เพิ่มข้อมูลหมวดหมู่พัสดุใหม่เข้าสู่ระบบ โดยเริ่มจากการรับค่าจากผู้ใช้ผ่านทางคีย์บอร์ด ซึ่งบรรทัดแรก name = input('ชื่อหมวด (<=30): ').strip() จะให้ผู้ใช้พิมพ์ชื่อหมวดหมู่ที่ต้องการเพิ่ม และใช้ .strip() เพื่อตัดช่องว่างส่วนเกินออก บรรทัดต่อมา desc = input('คำอธิบาย (<=80): ').strip() รับคำอธิบายเพิ่มเติมของหมวดหมู่นั้น ถ้าผู้ใช้ไม่ได้กรอกชื่อหมวด (ตรวจสอบด้วย if not name:) โปรแกรมจะแสดงข้อความเตือนว่า “ข้อมูลไม่ถูกต้อง” และหยุดการทำงานของฟังก์ชันทันทีด้วย return เพื่อไม่ให้สร้างหมวดหมู่ที่ไม่มีชื่อ

เมื่อผู้ใช้กรอกข้อมูลถูกต้อง ระบบจะเรียกใช้ self.cats.next_id() เพื่อสร้างรหัสหมวด (cat_id) ใหม่โดยอัตโนมัติ ซึ่งเป็นตัวเลขที่ไม่ซ้ำกับของเดิม จากนั้นบรรทัด self.cats.add_record(cid, self.cats.pack(1, cid, name, desc)) จะทำการบันทึกข้อมูลหมวดหมู่ลงในไฟล์ไบนารี categories.bin โดยเรียกใช้เมธอด pack() เพื่อจัดรูปแบบข้อมูลให้ตรงตามโครงสร้างที่กำหนดในระบบ (แปลงเป็น bytes ก่อนเก็บ) แล้วส่งต่อให้ add_record() เขียนข้อมูลนั้นลงไฟล์จริง สุดท้ายโปรแกรมจะพิมพ์ข้อความยืนยันการเพิ่มหมวดหมู่ใหม่ เช่น “+ เพิ่มหมวด cat_id=1” เพื่อแจ้งให้ผู้ใช้ทราบว่าการเพิ่มข้อมูลสำเร็จเรียบร้อยแล้ว

```
def add_category(self):
    name = input('ชื่อหมวด (<=30): ').strip()
    desc = input('คำอธิบาย (<=80): ').strip()
    if not name: print('! ข้อมูลไม่ถูกต้อง'); return
    cid = self.cats.next_id()
    self.cats.add_record(cid, self.cats.pack(1, cid, name, desc))
    print(f'+ เพิ่มหมวด cat_id={cid}')
```

รูปที่ 4-10 add_category

4.3.2 ฟังก์ชัน add_item

ใช้สำหรับเพิ่ม “รายการพัสดุ” ลงในระบบ โดยขั้นตอนภายในฟังก์ชันนี้ออกแบบมาให้มีการตรวจสอบความถูกต้องของข้อมูลและเชื่อมโยงกับหมวดหมู่ (category) ที่มีอยู่ในไฟล์ categories.bin อย่างเป็นระบบ เริ่มจากบรรทัดแรก name = input('ชื่อพัสดุ (<=30): ').strip() ใช้รับชื่อของพัสดุจากผู้ใช้ และตัดช่องว่างส่วนเกินออก ต่อมาภายในคำสั่ง try: โปรแกรมจะพยายามรับข้อมูลเพิ่มเติมที่เป็นตัวเลข ได้แก่

4.3.2.1 cat_id คือรหัสหมวดหมู่ที่พัสดุนั้นสังกัด

4.3.2.2 qty คือจำนวนเริ่มต้นของพัสดุในสต็อก

4.3.2.3 priceb คือราคาต่อชิ้นของพัสดุในหน่วยบาท

หากผู้ใช้กรอกค่าที่ไม่สามารถแปลงเป็นตัวเลขได้ เช่น ตัวอักษรหรือค่าว่าง คำสั่ง except Exception: จะทำงานทันที โดยแสดงข้อความว่า “! อินพุตไม่ถูกต้อง” แล้วออกจากฟังก์ชัน เพื่อป้องกันข้อผิดพลาดเมื่อรับค่ามาครบ โปรแกรมจะตรวจสอบเงื่อนไขต่อว่า if not name or qty < 0 or priceb < 0: เพื่อไม่ให้ชื่อว่างหรือค่าจำนวนและราคาติดลบ หากไม่ผ่านเงื่อนไขนี้ก็จะเตือนว่า “! ข้อมูลไม่ถูกต้อง” และหยุดทำงาน จากนั้นมีการตรวจสอบอีกขั้นด้วย if not

self.cats.read_record(cat_id): เพื่อดูว่ามีหมวดหมู่ (cat_id) นั้นอยู่จริงหรือไม่ ถ้าไม่พบ ระบบจะแสดงข้อความ “! ไม่พบหมวด” และหยุดการทำงาน

```
def add_item(self):
    name = input('ชื่อพัสดุ (<=30): ').strip()
    try:
        cat_id = int(input('cat_id: ').strip())
        qty    = int(input('จำนวนเริ่มต้น: ').strip())
        priceb = float(input('ราคา/ชิ้น (บาท): ').strip())
    except Exception:
        print('! อินพุตไม่ถูกต้อง'); return
    if not name or qty < 0 or priceb < 0: print('! ข้อมูลไม่ถูกต้อง'); return
    if not self.cats.read_record(cat_id): print('! ไม่พบหมวด'); return
    st_in = (input('สถานะ (available/damaged/disposed) [available]: ').strip().lower() or 'available')
    status = ITEM_STATUS_REV.get(st_in, 0)
    iid = self.items.next_id()
    self.items.add_record(iid, self.items.pack(1, iid, name, cat_id, qty, int(round(priceb*100)), status))
    print(f'+ เพิ่มพัสดุ item_id={iid}')
```

รูปที่ 4-11 add_item

4.3.3 ฟังก์ชัน add_movement

ฟังก์ชัน add_movement(self) มีหน้าที่ในการบันทึก “การเคลื่อนไหวของพัสดุ” (Movement) เช่น การเบิกจ่าย (issue), การโอน (transfer), การคืน (return) หรือการซ่อม (repair) โดยจะเชื่อมโยงข้อมูลระหว่างไฟล์ items.bin (ข้อมูลพัสดุ) และ movements.bin (ประวัติการเคลื่อนไหว)

เมื่อโปรแกรมทำงาน ฟังก์ชันจะเริ่มต้นด้วยการรับค่าจากผู้ใช้

4.3.3.1 iid (รหัสพัสดุ) เพื่อระบุว่ากำลังทำรายการกับพัสดุชิ้นใด

4.3.3.2 วันที่ (ymd) ในรูปแบบ YYYY-MM-DD โดยแปลงเป็นจำนวนเต็มผ่านฟังก์ชัน ymd_to_int()

4.3.3.3 qty คือจำนวนพัสดุที่มีการเคลื่อนไหว

ถ้าผู้ใช้กรอกข้อมูลผิดพลาด เช่น พิมพ์ตัวอักษรแทนตัวเลข จะเข้าสู่ส่วน except และแสดงข้อความ “! อินพุตไม่ถูกต้อง” จากนั้นออกจากฟังก์ชัน เพื่อป้องกันการคำนวณผิดพลาด เมื่อรับข้อมูลสำเร็จ ระบบจะตรวจสอบว่าจำนวน (qty) ต้องมากกว่า 0 เท่านั้น หากไม่ผ่านจะเตือนว่า “! qty ต้อง > 0” ต่อมาโปรแกรมจะอ่านข้อมูลพัสดุจากไฟล์ items.bin ผ่านคำสั่ง self.items.read_record(iid) ถ้าไม่พบข้อมูลของพัสดุนั้น จะแสดงข้อความ “! ไม่พบพัสดุ” แล้วหยุดทำงาน ถ้าพัสดุมีอยู่จริง โปรแกรมจะดึงข้อมูลออกมา (unpack) แล้วให้ผู้ใช้เลือก ประเภทของการเคลื่อนไหว (issue/transfer/return/repair) จากนั้นโปรแกรมจะตรวจสอบว่าค่าที่กรอกตรงกับประเภทที่กำหนดไว้ใน MOVE_TYPE_REV หรือไม่ ถ้าไม่ตรง จะขึ้นข้อความ “! ประเภทไม่ถูกต้อง”

ต่อมาจะให้ผู้ใช้กรอกชื่อผู้ดำเนินการ (operator) เพื่อเก็บไว้เป็นหลักฐานว่าใครเป็นผู้ทำรายการนั้น หลังจากนั้นจะเข้าสู่ส่วนสำคัญของฟังก์ชัน คือการคำนวณจำนวนพัสดุใหม่ (new_qty) ตามประเภทของการเคลื่อนไหว

4.3.3.4 ถ้าเป็น issue (เบิกออกจากสต็อก) โปรแกรมจะตรวจสอบก่อนว่ามีจำนวนพัสดุเพียงพอหรือไม่ ถ้าไม่พอจะขึ้นข้อความ “! สต็อกไม่พอ”
หากเพียงพอ ระบบจะหักจำนวนออก (new_qty -= qty)

4.3.3.5 ถ้าเป็น return (นำของคืนเข้าสต็อก) จะบวกจำนวนกลับ (new_qty += qty)

4.3.3.6 ส่วนประเภท transfer และ repair จะไม่กระทบกับจำนวนในสต็อก เพราะเป็นเพียงการบันทึกการเคลื่อนไหวเท่านั้น

เมื่อคำนวณเสร็จ ระบบจะสร้างรหัสประจำรายการเคลื่อนไหว (move_id) ใหม่อัตโนมัติด้วย self.moves.next_id() แล้วบันทึกข้อมูลลงไฟล์ movements.bin ผ่านเมธอด

```
def add_movement(self):
    try:
        iid = int(input('item_id: '))
        ymd = ymd_to_int(input('วันที่ (YYYY-MM-DD): ').strip())
        qty = int(input('ปริมาณ: '))
    except Exception:
        print('! อินพุตไม่ถูกต้อง'); return
    if qty <= 0: print('! qty ต้อง > 0'); return
    it_raw = self.items.read_record(iid)
    if not it_raw: print('! ไม่พบพัสดุ'); return
    it = self.items.unpack(it_raw)
    typ_str = (input('ประเภท (issue/transfer/return/repair): ').strip().lower())
    typ = MOVE_TYPE_REV.get(typ_str, None)
    if typ is None: print('! ประเภทไม่ถูกต้อง'); return
    operator = input('ผู้ดำเนินการ (<=30): ').strip()
    # ปรับ qty ตามประเภท
    new_qty = it['qty']
    if typ == 0:  # issue
        if qty > it['qty']:
            print('! สต็อกไม่พอ'); return
        new_qty -= qty
    elif typ == 2:  # return
        new_qty += qty
    # transfer/repair: ไม่แตะ qty
    mid = self.moves.next_id()
    self.moves.add_record(mid, self.moves.pack(1, mid, iid, ymd, qty, typ, operator))
    if new_qty != it['qty']:
        self.items.update_record(iid, self.items.pack(1, it['item_id'], it['name'], it['cat_id'], new_qty, it['price_cents'], it['status']))
    print(f'+ บันทึกการเคลื่อนย้าย move_id={mid}')
```

รูปที่ 4-12 add_movement

4.4 ฟังก์ชันอัพเดตข้อมูล

4.4.1 ฟังก์ชัน update_category

ฟังก์ชัน update_category(self) ทำหน้าที่แก้ไขหรืออัปเดตข้อมูลหมวดหมู่พัสดุที่มีอยู่แล้วในไฟล์ categories.bin โดยมีขั้นตอนการทำงานอย่างเป็นระบบ ดังนี้

เริ่มต้นโปรแกรมจะขอให้ผู้ใช้กรอกรหัสหมวด (cat_id) ผ่าน input() แล้วพยายามแปลงค่าที่กรอกให้เป็นตัวเลข (int) หากผู้ใช้กรอกค่าที่ไม่ใช่ตัวเลข โปรแกรมจะเกิดข้อผิดพลาด (Exception) และเข้าสู่คำสั่ง except เพื่อแสดงข้อความว่า “! อินพุตไม่ถูกต้อง” จากนั้นจะหยุดการทำงาน (return) เพื่อป้องกันการค้นหาหมวดหมู่ด้วยข้อมูลผิดประเภท

หากรหัสหมวดถูกต้อง โปรแกรมจะอ่านข้อมูลหมวดหมู่จากไฟล์ไบนารี (categories.bin) โดยใช้ self.cats.read_record(cid) ซึ่งจะดึงข้อมูลดิบ (raw bytes) ของหมวดหมู่นั้นขึ้นมา หากไม่พบหมวดที่มีรหัสตรงกัน จะแสดงข้อความ “! ไม่พบหมวด” แล้วจบการทำงานทันที

ในกรณีที่พบข้อมูลหมวด โปรแกรมจะถอดรหัสข้อมูลไบนารีด้วยคำสั่ง self.cats.unpack(raw) เพื่อแปลงให้อยู่ในรูปแบบที่อ่านได้ (dict) และเก็บในตัวแปร r จากนั้นจะแสดงค่าปัจจุบันของชื่อหมวด (r['name']) และคำอธิบาย (r['desc']) ออกมาในวงเล็บ เพื่อให้ผู้ใช้สามารถกรอกข้อมูลใหม่แทนของเดิม หรือกด Enter เพื่อคงค่าเดิมไว้

```
def update_category(self):
    try: cid = int(input('cat_id: '))
    except Exception: print('! อินพุตไม่ถูกต้อง'); return
    raw = self.cats.read_record(cid)
    if not raw: print('! ไม่พบหมวด'); return
    r = self.cats.unpack(raw)
    name = input(f"ชื่อหมวด [{r['name']}]: ").strip() or r['name']
    desc = input(f"คำอธิบาย [{r['desc']}]: ").strip() or r['desc']
    self.cats.update_record(cid, self.cats.pack(1, cid, name, desc))
    print('* อัปเดตหมวดแล้ว')
```

รูปที่ 4-13 update_category

4.4.2 ฟังก์ชัน update_item

ฟังก์ชัน update_item(self) ใช้สำหรับแก้ไขข้อมูลของพัสดุที่บันทึกไว้ในไฟล์ items.bin โดยมีขั้นตอนทำงานอย่างเป็นลำดับเพื่อให้มั่นใจว่าข้อมูลใหม่ถูกต้องและไม่ทำให้ระบบคลังเสียหาย

เมื่อเริ่มทำงาน โปรแกรมจะให้ผู้ใช้กรอก รหัสพัสดุ (item_id) เพื่อระบุว่าต้องการแก้ไขรายการใด โดยใช้ input() และพยายามแปลงเป็นตัวเลข (int) หากกรอกไม่ถูกต้อง เช่น พิมพ์ตัวอักษรหรือเว้นว่างไว้ จะเกิด Exception และเข้าสู่ except เพื่อแสดงข้อความ “! อินพุตไม่ถูกต้อง” และหยุดการทำงานทันที

หากรหัสถูกต้อง โปรแกรมจะอ่านข้อมูลพัสดุจากไฟล์ไบนารีผ่าน self.items.read_record(iid) เพื่อดึงข้อมูลดิบ (raw bytes) ของพัสดุนั้นขึ้นมา ถ้าไม่พบข้อมูลจะขึ้นข้อความ “! ไม่พบพัสดุ” แล้วจบการทำงาน หากพบ โปรแกรมจะใช้ self.items.unpack(raw) แปลงข้อมูลจากไบนารีให้อยู่ในรูปแบบพจนานุกรม (dict) เพื่อให้อ่านและแก้ไขได้สะดวก

```
def update_item(self):
    try: iid = int(input('item_id: '))
    except Exception: print('! อินพุตไม่ถูกต้อง'); return
    raw = self.items.read_record(iid)
    if not raw: print('! ไม่พบพัสดุ'); return
    r = self.items.unpack(raw)
    name = input(f"ชื่อพัสดุ [{r['name']}]: ").strip() or r['name']
    try:
        cat_id = int(input(f"cat_id [{r['cat_id']}]: ") or r['cat_id'])
        qty    = int(input(f"จำนวน [{r['qty']}]: ") or r['qty'])
        priceb = float(input(f"ราคา/ชิ้น [{r['price_cents']/100:.2f}]: ") or (r['price_cents']/100))
    except Exception:
        print('! ตัวเลขไม่ถูกต้อง'); return
    st_in = input(f"สถานะ ({'/'.join(ITEM_STATUS.values())}) [{ITEM_STATUS[r['status']]}]: ").strip() or ITEM_STATUS[r['status']]
    status = ITEM_STATUS_REV.get(st_in.lower(), r['status'])
    if qty < 0 or priceb < 0: print('! ข้อมูลไม่ถูกต้อง'); return
    if not self.cats.read_record(cat_id): print('! ไม่พบหมวด'); return
    self.items.update_record(iid, self.items.pack(1, iid, name, cat_id, qty, int(round(priceb*100)), status))
    print('* อัปเดตพัสดุแล้ว')
```

รูปที่ 4-14 update_item

4.5 ฟังก์ชันลบข้อมูล

4.5.1 ฟังก์ชัน delete_category

ฟังก์ชัน delete_category(self) ใช้สำหรับลบหมวดหมู่พัสดุออกจากระบบ (ไฟล์ categories.bin) แต่มีการตรวจสอบและป้องกันข้อผิดพลาดหลายชั้น เพื่อไม่ให้เกิดการลบข้อมูลที่ยังถูกใช้งานอยู่ในระบบจริง

เมื่อผู้ใช้เรียกใช้ฟังก์ชันนี้ โปรแกรมจะเริ่มต้นด้วยการขอให้กรอกรหัสหมวด (cat_id) ที่ต้องการลบ โดยใช้ input() และพยายามแปลงค่าที่กรอกให้เป็นตัวเลข (int) หากผู้ใช้กรอกข้อมูลไม่ถูกต้อง เช่น พิมพ์ตัวอักษรหรือเว้นว่างไว้ จะเกิด Exception และเข้าสู่คำสั่ง except เพื่อแสดงข้อความ “! อินพุตไม่ถูกต้อง” แล้วหยุดการทำงานทันที (return) เพื่อป้องกันความเสียหายจากค่าที่ไม่ถูกต้อง

```
def delete_category(self):
    try: cid = int(input('cat_id: '))
    except Exception: print('! อินพุตไม่ถูกต้อง'); return
    # กันลบถ้ายังมี item อ้างอิง
    for _, raw in self.items.iter_active():
        it = self.items.unpack(raw)
        if it['cat_id'] == cid:
            print('! มีพัสดุอ้างอิงหมวดนี้ ลบไม่ได้')
            return
    try: self.cats.delete_record(cid); print('- ลบหมวดแล้ว')
    except Exception as e: print('!', e)
```

รูปที่ 4-15 delete_category

4.5.2 ฟังก์ชัน delete_item

ฟังก์ชัน delete_item(self) เป็นส่วนหนึ่งของระบบจัดการคลังพัสดุที่ทำหน้าที่ลบข้อมูลพัสดุออกจากไฟล์ items.bin โดยใช้วิธี soft delete ซึ่งหมายถึงข้อมูลจะไม่ถูกลบทิ้งจริง ๆ แต่เพียงเปลี่ยนสถานะของระเบียนให้ไม่ถูกใช้งาน เพื่อป้องกันความเสียหายของไฟล์และเก็บประวัติไว้ตรวจสอบในอนาคต การทำงานของฟังก์ชันเริ่มจากให้ผู้ใช้กรอกรหัสพัสดุ (item_id) ผ่าน input() แล้วแปลงเป็นตัวเลข ถ้าผู้ใช้กรอกผิดหรือไม่ใช่ตัวเลข โปรแกรมจะเข้าสู่ส่วน except และแจ้งเตือน “! อินพุตไม่ถูกต้อง” เพื่อหยุดการทำงานทันที จากนั้นโปรแกรมจะพยายามลบข้อมูลพัสดุด้วยคำสั่ง self.items.delete_record(iid) ซึ่งจะไปเปลี่ยนค่า flag ของระเบียนนั้นในไฟล์ให้เป็น 0 (หมายถึงถูกลบ) และเพิ่มตำแหน่งของระเบียนลงใน free list เพื่อเก็บไว้ใช้ซ้ำในการบันทึกข้อมูลใหม่ เมื่อการลบสำเร็จ โปรแกรมจะแสดงข้อความ “- ลบพัสดุแล้ว”

```
def delete_item(self):
    try: iid = int(input('item_id: '))
    except Exception: print('! อินพุตไม่ถูกต้อง'); return
    try: self.items.delete_record(iid); print('- ลบพัสดุแล้ว')
    except Exception as e: print('!', e)
```

รูปที่ 4-16 delete_item

4.5.3 ฟังก์ชัน delete_movement

ฟังก์ชัน delete_movement(self) ในโปรแกรม Inventory-BinIO เป็นส่วนที่ใช้สำหรับลบข้อมูล “การเคลื่อนย้ายพัสดุ” ออกจากไฟล์ movements.bin โดยมีลักษณะการทำงาน

แบบ Soft Delete เหมือนกับการลบในตารางอื่น คือ ไม่ได้ลบข้อมูลจริงจากไฟล์แต่เปลี่ยนสถานะ (flag) ของระเบียนให้เป็น 0 เพื่อรักษาความสมบูรณ์ของโครงสร้างไฟล์และสามารถเก็บประวัติย้อนหลังไว้ได้

การทำงานเริ่มจากให้ผู้ใช้กรอกหมายเลข move_id ของรายการที่ต้องการลบผ่านคำสั่ง input() แล้วพยายามแปลงเป็นจำนวนเต็ม หากผู้ใช้กรอกข้อมูลผิดหรือไม่ใช่ตัวเลข โปรแกรมจะเข้าสู่ except Exception และแสดงข้อความ “! อินพุตไม่ถูกต้อง” พร้อมยุติการทำงานทันที

หากแปลงค่าได้ถูกต้อง โปรแกรมจะเรียก self.moves.delete_record(mid) เพื่อไปทำการลบระเบียนนั้นในไฟล์ ซึ่งคำสั่งนี้จะเปลี่ยนสถานะของระเบียนให้ไม่ใช้งานและบันทึกตำแหน่งไว้ใน free list สำหรับนำกลับมาใช้ได้ในอนาคต เมื่อการลบสำเร็จ จะมีข้อความ “- ลบรายการเคลื่อนย้ายแล้ว (ไม่ย้อน qty)”

```
def delete_movement(self):
    try: mid = int(input('move_id: '))
    except Exception: print('! อินพุตไม่ถูกต้อง'); return
    try: self.moves.delete_record(mid); print('- ลบรายการเคลื่อนย้ายแล้ว (ไม่ย้อน qty)')
    except Exception as e: print('!', e)
```

รูปที่ 4-17 delete_movement

4.6 ฟังก์ชันลบข้อมูล

4.6.1 ฟังก์ชัน view_single

ฟังก์ชัน view_single() ใช้แสดงข้อมูล เพียง 1 รายการ ตามที่ผู้ใช้เลือกว่าจะดูหมวดหมู่ (Category), พัสดุ (Item), หรือการเคลื่อนย้าย (Movement) โดยให้กรอกชนิดและหมายเลข ID ของรายการนั้น

โปรแกรมจะอ่านข้อมูลจากไฟล์ไบนารีที่เกี่ยวข้อง (categories.bin, items.bin, หรือ movements.bin) แล้วแสดงรายละเอียด เช่น

4.6.1.1 หมวดหมู่ แสดง id, ชื่อ, คำอธิบาย

4.6.1.2 พัสดุ แสดง id, ชื่อ, หมวด, จำนวน, ราคา, สถานะ

4.6.1.3 การเคลื่อนย้าย แสดง id, item_id, วันที่, ประเภท, จำนวน, ผู้ดำเนินการ

```python
def view_single(self):
    t = input('ชนิด (category/item/movement): ').strip().lower()
    try: i = int(input('id: '))
    except Exception: print('! อินพุตไม่ถูกต้อง'); return
    if t.startswith('cat'):
        raw = self.cats.read_record(i)
        if not raw: print('! ไม่พบ'); return
        r = self.cats.unpack(raw)
        print(f"[Category] id={r['cat_id']} name={r['name']} desc={r['desc']}")
    elif t.startswith('item'):
        raw = self.items.read_record(i)
        if not raw: print('! ไม่พบ'); return
        r = self.items.unpack(raw)
        print(f"[Item] id={r['item_id']} name={r['name']} cat_id={r['cat_id']} qty={r['qty']} "
              f"price={r['price_cents']/100:.2f} status={ITEM_STATUS[r['status']]}")
    else:
        raw = self.moves.read_record(i)
        if not raw: print('! ไม่พบ'); return
        r = self.moves.unpack(raw)
        print(f"[Move] id={r['move_id']} item_id={r['item_id']} date={int_to_ymd(r['ymd'])} "
              f"type={MOVE_TYPE[r['type']]} qty={r['qty']} by={r['operator']}")
```

รูปที่ 4-18 view_single

4.6.2 ฟังก์ชัน view_all

ฟังก์ชัน view_all(self) ใช้ “แสดงรายการทั้งหมดที่ยังใช้งานอยู่ (active)” ของตารางที่ผู้ใช้เลือก category, item, หรือ movement

ลำดับทำงานย่อ ๆ:

4.6.2.1 รับชนิดข้อมูลจากผู้ใช้ (t) และรองรับปุ่มย้อนกลับ ('', '0', 'b', 'back') แล้ว return ทันที

4.6.2.2 ถ้าเป็น category วน self.cats.iter_active() เพื่ออ่านเฉพาะระเบียนที่ flag=1 แล้ว unpack() ออกมา พิมพ์ cat_id | name | desc

4.6.2.3 ถ้าเป็น item วน self.items.iter_active() แล้วพิมพ์ item_id | name | cat_id | qty | price | status (ราคาแปลงจากสตางค์เป็นบาทด้วย /100, สถานะแปลงเลขเป็นข้อความผ่าน ITEM_STATUS)

4.6.2.4 ถ้าเป็น movement วน self.moves.iter_active() แล้วพิมพ์ move_id | item_id | date | type | qty | operator (วันที่แปลงด้วย int_to_ymd, ประเภทแปลงเลขเป็นข้อความผ่าน MOVE_TYPE)

4.6.2.5 ถ้าผู้ใช้พิมพ์ชนิดไม่ถูกต้อง จะแจ้ง “เขียนไม่ถูกต้อง”

```python
def view_all(self):
    t = input('ชนิด (category/item/movement, 0=Back): ').strip().lower()
    if t in ('', '0', 'b', 'back'):
        return
    if t.startswith('cat'):
        for _, raw in self.cats.iter_active():
            r = self.cats.unpack(raw)
            print(f"{r['cat_id']:>4} | {r['name']:<30} | {r['desc']}")
    elif t.startswith('item'):
        for _, raw in self.items.iter_active():
            r = self.items.unpack(raw)
            print(f"{r['item_id']:>4} | {r['name']:<30} | cat={r['cat_id']:<4} | "
                  f"qty={r['qty']:<6} | {r['price_cents']/100:>8.2f} | {ITEM_STATUS[r['status']]:<8}")
    elif t.startswith('move'):
        for _, raw in self.moves.iter_active():
            r = self.moves.unpack(raw)
            print(f"{r['move_id']:>5} | item={r['item_id']:<4} | {int_to_ymd(r['ymd'])} | "
                  f"{MOVE_TYPE[r['type']]:<8} | qty={r['qty']:<6} | by={r['operator']}")
    else:
        print("เขียนไม่ถูกต้อง")
```

รูปที่ 4-19 view_all

4.6.3 ฟังก์ชัน view_filter

view_filter() เป็นฟังก์ชันที่ช่วยให้ผู้ใช้ค้นหาข้อมูลในระบบได้อย่างเฉพาะเจาะจง เช่น ดูเฉพาะพัสดุในหมวดที่ต้องการ ดูเฉพาะรายการที่อยู่ในช่วงวันที่กำหนด หรือค้นหาพัสดุที่มีสถานะเสียหาย เป็นต้น เพื่อช่วยให้การตรวจสอบข้อมูลในคลังมีประสิทธิภาพและรวดเร็วมากขึ้น

4.6.3.1 เริ่มต้นโปรแกรมจะให้กรอกชนิดข้อมูลที่ต้องการค้นหา หากพิมพ์ “0”, “b”, “back” หรือเว้นว่าง จะย้อนกลับทันที

4.6.3.2 ถ้าเลือก category จะให้กรอกคำค้นชื่อหมวด แล้ววนอ่านทุกระเบียนใน categories.bin (เฉพาะที่ active) ถ้าชื่อหมวดมีคำค้นอยู่ในนั้น จะพิมพ์ cat_id และ name ออกมา

4.6.3.3 ถ้าเลือก item จะให้กรองตาม “สถานะ (available/damaged/disposed)”, “รหัสหมวด (cat_id)” และ “ชื่อพัสดุ” โปรแกรมจะตรวจสอบอินพุตอย่างละเอียด เช่น กรอกชื่อบางส่วนของสถานะก็จับได้ จากนั้นจะกรองเฉพาะรายการที่ตรงเงื่อนไขทั้งสามข้อ แล้วแสดง item_id | name | cat_id | qty | price | status

4.6.3.4 ถ้าเลือก movement จะให้กรอกช่วงวันที่ “FROM,TO (YYYY-MM-DD,YYYY-MM-DD)” และสามารถกรอก item_id เพิ่มเติมเพื่อจำกัดผลได้ โปรแกรมจะแปลงวันที่เป็นตัวเลข (YYYYMMDD) เพื่อเปรียบเทียบ แล้วพิมพ์เฉพาะรายการที่อยู่ในช่วงนั้นออกมาเป็น move_id | item_id | date | type | qty | operator

4.6.3.5 ถ้าพิมพ์ชนิดผิด โปรแกรมจะแจ้งว่า “เขียนไม่ถูกต้อง”

```
def view_filter(self):
    t = input('ชนิด (category/item/movement, 0=Back): ').strip().lower()
    if t in ('', '0', 'b', 'back'):
        return
    if t.startswith('cat'):
        q = input('ค้นหาชื่อหมวด: ').strip().lower()
        for _, raw in self.cats.iter_active():
            r = self.cats.unpack(raw)
            if q in r['name'].lower():
                print(f"{r['cat_id']:>4} | {r['name']}")
    elif t.startswith('item'):
        raw_in = input('สถานะ (available/damaged/disposed หรือเว้นว่าง): ').strip().lower()
        st_code = None
        if raw_in:
            if raw_in.isdigit() and int(raw_in) in ITEM_STATUS:
                st_code = int(raw_in)
            else:
                exact = [k for k,v in ITEM_STATUS.items() if v == raw_in]
                if exact: st_code = exact[0]
                else:
                    matched = [k for k,v in ITEM_STATUS.items() if v.startswith(raw_in)]
                    if len(matched)==1: st_code = matched[0]
                    elif len(matched)>1:
                        print('คำกวม:', ', '.join(ITEM_STATUS[m] for m in matched)); return
        cat_in = input('กรอง cat_id (เว้นว่าง=ทั้งหมด): ').strip()
        cat_id = int(cat_in) if cat_in.isdigit() else None
        name_q = input('ค้นหาชื่อพัสดุ (เว้นว่าง=ไม่กรอง): ').strip().lower()
        for _, raw in self.items.iter_active():
            r = self.items.unpack(raw)
            if (st_code is None or r['status']==st_code) and \
               (cat_id is None or r['cat_id']==cat_id) and \
               (not name_q or name_q in r['name'].lower()):
                print(f"{r['item_id']:>4} | {r['name']:<30} | cat={r['cat_id']:<4} | "
                      f"qty={r['qty']:<6} | {r['price_cents']/100:>8.2f} | {ITEM_STATUS[r['status']]:<8}")
```

รูปที่ 4-20 view_filter (ส่วนที่ 1)

```
elif t.startswith('move'):
    try:
        a_str,b_str = input('ช่วงวันที่ FROM,TO (YYYY-MM-DD,YYYY-MM-DD): ').split(',')
        a,b = ymd_to_int(a_str.strip()), ymd_to_int(b_str.strip())
    except Exception:
        print('รูปแบบวันที่ไม่ถูกต้อง'); return
    item_in = input('item_id (เว้นว่าง=ทั้งหมด): ').strip()
    item_id = int(item_in) if item_in.isdigit() else None
    for _, raw in self.moves.iter_active():
        r = self.moves.unpack(raw)
        if a <= r['ymd'] <= b and (item_id is None or r['item_id']==item_id):
            print(f"{r['move_id']:>5} | item={r['item_id']:<4} | {int_to_ymd(r['ymd'])} | "
                  f"{MOVE_TYPE[r['type']]:<8} | qty={r['qty']:<6} | by={r['operator']}")
else:
    print("เขียนไม่ถูกต้อง")
```

รูปที่ 4-21 view_filter (ส่วนที่ 2)

4.6.4 ฟังก์ชัน view_stats

ฟังก์ชัน view_stats ทำหน้าที่ แสดงสรุปภาพรวมของคลังพัสดุ ได้แก่ จำนวนพัสดุในแต่ละสถานะ จำนวนทั้งหมด และมูลค่ารวมเป็นเงินบาท เพื่อช่วยให้ผู้ใช้ตรวจสอบข้อมูลเชิงสถิติของสินค้าคงคลังได้อย่างรวดเร็วและเข้าใจง่าย

```python
def view_stats(self):
    # นับสินค้าตามสถานะ
    cnt = {k:0 for k in ITEM_STATUS}
    total_qty = 0
    total_value = 0
    for _, raw in self.items.iter_active():
        r = self.items.unpack(raw)
        cnt[r['status']] += 1
        total_qty += r['qty']
        total_value += r['qty'] * r['price_cents']
    print('Items by status:')
    for k,v in cnt.items(): print(f"  {ITEM_STATUS[k]:<8} = {v}")
    print(f"Total Qty = {total_qty}")
    print(f"Total Value = {total_value/100:,.2f} THB")
```

รูปที่ 4-22 view_stats

4.7 ฟังก์ชัน generate_report

มีหน้าที่ สร้างรายงานสรุปการเคลื่อนไหวของพัสดุ (Unified Movement Report) รวมข้อมูลจากทั้ง 3 ไฟล์ ได้แก่ categories.bin, items.bin และ movements.bin ให้อยู่ในรายงานเดียว ขั้นตอนการทำงานสรุปดังนี้

4.7.1 เตรียม lookup ชื่อหมวดหมู่ (cat_name)

- วนอ่านข้อมูลหมวดหมู่ที่ยังใช้งานอยู่ (iter_active() จาก categories.bin)

- ถอดข้อมูล (unpack()) แล้วเก็บชื่อหมวดใน dictionary โดยใช้ cat_id เป็น key เพื่อให้สามารถอ้างอิงชื่อหมวดจาก cat_id ได้ในภายหลัง

4.7.2 เตรียมข้อมูลพัสดุและจำนวนคงเหลือ (item_info และ current_qty)

- อ่านข้อมูลจาก items.bin เฉพาะรายการที่ active เก็บชื่อพัสดุและชื่อหมวดที่สังกัดใน item_info เก็บจำนวนคงเหลือของแต่ละพัสดุใน current_qty

4.7.3 รวบรวมข้อมูลการเคลื่อนไหว (moves)

- อ่านข้อมูลทั้งหมดจาก movements.bin ที่ยังใช้งานอยู่ (flag=1) แปลงเป็นโครงสร้างข้อมูล dictionary แล้วเก็บไว้ในลิสต์ moves เพื่อใช้ในรายงาน

4.7.4 กรณีไม่มีข้อมูลเคลื่อนไหวเลย

- ถ้าไม่มี movement ใด ๆ (เช่น ยังไม่เคยมีการเบิก–คืนพัสดุ) โปรแกรมจะสร้างไฟล์รายงานและเขียนบรรทัด

```python
def generate_report(self, out_path: str):
    # --- เตรียม lookup ชื่อหมวด ---
    cat_name: Dict[int, str] = {}
    for _, raw in self.cats.iter_active():
        c = self.cats.unpack(raw)
        cat_name[c['cat_id']] = c['name'] or f"cat#{c['cat_id']}"

    # --- เตรียมข้อมูล item และ stock ปัจจุบัน ---
    item_info: Dict[int, Dict[str, Any]] = {}
    current_qty: Dict[int, int] = {}
    for _, raw in self.items.iter_active():
        it = self.items.unpack(raw)
        item_info[it['item_id']] = {
            'name': it['name'],
            'cat': cat_name.get(it['cat_id'], f"cat#{it['cat_id']}")
        }
        current_qty[it['item_id']] = int(it['qty'])

    # --- รวบรวม movements ที่ active ---
    moves = []
    for _, raw in self.moves.iter_active():
        m = self.moves.unpack(raw)
        moves.append(m)

    # ถ้ายังไม่มี movement เลย
    if not moves:
        with open(out_path, 'w', encoding='utf-8') as f:
            ts = datetime.now().astimezone().strftime('%Y-%m-%d %H:%M:%S (%z)')
            f.write('Inventory — Unified Movement Report\n')
            f.write(f'Generated At : {ts}\n\n')
            f.write('(no data)\n')
        print('* เขียนรายงานที่', out_path)
        return
```

รูปที่ 4-23 generate_report

4.7.5 คำนวณ Net Delta ต่อพัสดุ (net_delta)

- วนซ้ำใน moves ซึ่งเก็บรายการเคลื่อนไหวทั้งหมด

- ถ้าเป็น issue (เบิกออก) จะถือเป็นจำนวนติดลบ (-q) ถ้าเป็น return (คืนเข้า) จะเป็นจำนวนบวก (q) ส่วน transfer และ repair จะไม่นับรวม (ค่า delta = 0)

- เก็บผลรวมสุทธิของแต่ละ item_id ลงใน dictionary net_delta เช่น ถ้าพัสดุรหัส 1 ถูกเบิก 5 ชิ้นและคืน 2 ชิ้น จะได้ net_delta[1] = -3

4.7.6 คำนวณ Opening Stock (สต็อกเปิดต้น)

- ใช้สูตร opening = current - net_delta หมายถึง ปริมาณปัจจุบันลบด้วยการเปลี่ยนแปลงสุทธิ = ปริมาณก่อนเกิด movement ทั้งหมด เช่น ปัจจุบันเหลือ 7 ชิ้น และมีการเบิกสุทธิ 3 ชิ้น opening = 10

4.7.7 จัดเรียงข้อมูล movement

- เรียงลำดับตามวันที่ (ymd) และรหัส move_id เพื่อให้รายงานเรียงตามเวลา

- สร้างสำเนา running ของ opening เพื่อใช้ติดตามจำนวนคงเหลือหลังแต่ละ transaction

4.7.8 สร้างส่วนหัวของรายงาน

- ใส่ชื่อรายงานและเวลาที่สร้าง (Generated At)

- สร้างหัวตารางที่ประกอบด้วย MoveID | Operator | Item | Category | Type | Date | Qty(±) | Stock After

```
# --- คำนวณ net delta ต่อ item (สำหรับหา opening stock) ---
net_delta: Dict[int, int] = {}
for m in moves:
    iid = m['item_id']; q = int(m['qty'])
    if m['type'] == 0:      # issue
        delta = -q
    elif m['type'] == 2:    # return
        delta = q
    else:
        delta = 0
    net_delta[iid] = net_delta.get(iid, 0) + delta

# opening = current - net_delta
opening: Dict[int, int] = {}
for iid, cur in current_qty.items():
    opening[iid] = cur - net_delta.get(iid, 0)

# --- เรียง movement ตามวันที่ แล้วตาม id ---
moves.sort(key=lambda x: (x['ymd'], x['move_id']))
running = dict(opening)

# --- สร้างบรรทัดหัว ---
ts = datetime.now().astimezone().strftime('%Y-%m-%d %H:%M:%S (%z)')
lines = [
    'Inventory — Unified Movement Report',
    f'Generated At : {ts}',
    ''
]
th = (f"{'MoveID':>6} | {'Operator':<20} | {'Item':<24} | "
      f"{'Category':<14} | {'Type':<8} | {'Date':<10} | {'Qty(±)':>6} | {'Stock After':>11}")
lines += [th, '-' * len(th)]
```

รูปที่ 4-24 generate_report (ส่วนที่ 2)

4.7.9 ตัวแปรสรุป (Summary variables)

- totals_by_type = {0: 0, 1: 0, 2: 0, 3: 0} ใช้เก็บ “จำนวนรวมของแต่ละประเภทการเคลื่อนไหว” ได้แก่ 0=issue (เบิก), 1=transfer (โอน), 2=return (คืน), 3=repair (ซ่อม) โดยบันทึกปริมาณจริงที่เกิดขึ้น (ไม่ใส่เครื่องหมายบวกหรือลบ)

- operators_all = set() เก็บชื่อผู้ดำเนินการทั้งหมดที่มีอยู่ในรายงาน (ไม่ซ้ำ)

- operators_by_type = {0:set(),1:set(),2:set(),3:set()} แยกเก็บชื่อผู้ดำเนินการแต่ละประเภท เช่น ใครบ้างที่เคยเบิก, คืน, โอน, ซ่อม

- rows = 0 นับจำนวนแถวในรายงาน (จำนวน movement ทั้งหมด)

4.7.10 เติมข้อมูลแต่ละรายการลงในตารางหลัก วนซ้ำในลิสต์ moves ซึ่งเก็บรายการเคลื่อนไหวทั้งหมด

- ดึง item_id แล้ว lookup จาก item_info เพื่อหาชื่อพัสดุ (item_nm) และหมวด (cat_nm)

- แปลงประเภท (m['type']) เป็นข้อความอ่านง่ายด้วย MOVE_TYPE เช่น 0 "issue"

- แปลงวันที่ ymd ให้เป็นรูปแบบ YYYY-MM-DD ด้วย int_to_ymd()

4.7.11 คำนวณจำนวนเปลี่ยนแปลง (Qty) และจำนวนคงเหลือหลังทำรายการ (Stock After)

- ถ้าเป็น issue แสดงจำนวนติดลบ (-qty)

- ถ้าเป็น return แสดงจำนวนบวก (+qty)

- ถ้าเป็น transfer/repair ไม่เปลี่ยนแปลงสต็อก (0)

- ใช้ตัวแปร before เก็บจำนวนก่อนเคลื่อนไหว, after เก็บจำนวนหลังเคลื่อนไหว แล้วอัปเดต running[iid] = after

4.7.12 เพิ่มบรรทัดลงในรายงาน เขียนบรรทัดในรูปแบบ:

- MoveID | Operator | Item | Category | Type | Date | Qty(±) | Stock After โดยค่าของ Qty(±) จะมีเครื่องหมายบวก/ลบแสดงการเปลี่ยนแปลงของสต็อก

4.7.13 อัปเดตค่ารวมสำหรับสรุปท้ายรายงาน

- เพิ่มตัวนับ rows += 1

- เก็บชื่อผู้ดำเนินการลงใน operators_all

- เก็บชื่อแยกตามประเภทลงใน operators_by_type[m['type']]

- เพิ่มยอดรวมจำนวนต่อประเภทใน totals_by_type[m['type']]

```python
# ตัวแปรสรุป
totals_by_type = {0: 0, 1: 0, 2: 0, 3: 0}     # รวมปริมาณต่อประเภท (ใช้ปริมาณจริง ไม่ใส่เครื่องหมาย)
operators_all = set()
operators_by_type = {0: set(), 1: set(), 2: set(), 3: set()}
rows = 0

# --- เดิมตารางหลัก พร้อมคำนวณ Stock After ---
for m in moves:
    iid = m['item_id']
    it = item_info.get(iid, {'name': f"item#{iid}", 'cat': '(unknown)'})
    item_nm = it['name']; cat_nm = it['cat']
    tname = MOVE_TYPE.get(m['type'], f"type#{m['type']}")
    ymd_str = int_to_ymd(m['ymd'])

    # ปรับเครื่องหมาย qty สำหรับคลังกลาง
    if m['type'] == 0:      # issue
        qty_disp = -int(m['qty'])
    elif m['type'] == 2:    # return
        qty_disp = int(m['qty'])
    else:                   # transfer/repair -> 0
        qty_disp = 0

    before = running.get(iid, 0)
    after = before + qty_disp
    running[iid] = after

    lines.append(
        f"{m['move_id']:>6} | {m['operator']:<20.20} | {item_nm:<24.24} | "
        f"{cat_nm:<14.14} | {tname:<8} | {ymd_str:<10} | {qty_disp:>6} | {after:>11}"
    )

    rows += 1
    operators_all.add(m['operator'])
    operators_by_type[m['type']].add(m['operator'])
    totals_by_type[m['type']] += int(m['qty'])
```

รูปที่ 4-25 generate_report (ส่วนที่ 3)

4.7.14 ส่วนสรุปท้ายตาราง

- โปรแกรมเพิ่มบรรทัด “Summary” ลงในลิสต์ lines เพื่อแสดงหัวข้อสรุปท้ายรายงาน

Summary - Rows : 12 - Distinct Operators (all) : 5

- Rows คือจำนวนรายการเคลื่อนไหวทั้งหมดในรายงาน
- Distinct Operators (all) คือจำนวนผู้ดำเนินการที่ไม่ซ้ำ (รวมทุกประเภท)

4.7.15 แสดงสถิติตามประเภทของการเคลื่อนไหว

ใช้ค่าที่เก็บไว้ในตัวแปรสรุปก่อนหน้า (totals_by_type และ operators_by_type) เพื่อสรุป:

- ปริมาณรวม (qty total) ของแต่ละประเภท เช่น เบิก (issue), คืน (return), โอน (transfer), ซ่อม (repair)
- จำนวนผู้ดำเนินการ (operators) ที่เกี่ยวข้องในแต่ละประเภท

ตัวอย่างบรรทัดในรายงาน

- Issues (ออก) qty total : 50 | operators: 3
- Returns (เข้า) qty total : 30 | operators: 2
- Transfers qty total : 10 | operators: 1
- Repairs qty total : 5 | operators: 1

4.7.16 สรุป “Ending Stock” ปัจจุบันจากตาราง Items

- แสดงข้อมูลจำนวนคงเหลือของพัสดุทุกชิ้นจากไฟล์ items.bin
- ใช้ข้อมูลจาก dictionary current_qty ซึ่งถูกอ่านไว้ตั้งแต่ตอนต้นของฟังก์ชัน
- ถ้าไม่มีพัสดุเลย (ฐานข้อมูลว่าง) จะแสดง (no items)

ถ้ามีข้อมูลจะพิมพ์ชื่อพัสดุและจำนวนคงเหลือ เช่น

Ending Stock by Item (from Items table):

- Computer : 15
- Projector : 8
- Cable : 50

4.7.17 เขียนไฟล์รายงานออกสู่ระบบ

- นำข้อมูลทั้งหมดใน lines มาต่อกันด้วย \n แล้วบันทึกเป็นไฟล์ข้อความ

- (inventory_report.txt) ที่ path ที่กำหนดใน out_pathเมื่อบันทึกเสร็จจะแจ้งข้อความในคอนโซลว่า * เขียนรายงานที่ data_inv/inventory_report.txt

```
# --- สรุปท้ายตาราง ---
lines += [
    '',
    'Summary'
]
lines += [f"- Rows                          : {rows}"]
lines += [f"- Distinct Operators (all)      : {len(operators_all)}"]
lines += [f"- Issues (ออก)   qty total     : {totals_by_type.get(0,0)}   | operators: {len(operators_by_type.get(0,set()))}"]
lines += [f"- Returns (เข้า) qty total      : {totals_by_type.get(2,0)}   | operators: {len(operators_by_type.get(2,set()))}"]
lines += [f"- Transfers     qty total       : {totals_by_type.get(1,0)}   | operators: {len(operators_by_type.get(1,set()))}"]
lines += [f"- Repairs       qty total       : {totals_by_type.get(3,0)}   | operators: {len(operators_by_type.get(3,set()))}"]
lines += ['']

# แสดง Ending Stock ปัจจุบันจาก items table (ภาพรวมตอนนี้)
lines += ['Ending Stock by Item (from Items table):']
if current_qty:
    for iid in sorted(current_qty):
        nm = item_info.get(iid, {'name': f"item#{iid}"})['name']
        lines.append(f"  - {nm} : {current_qty[iid]}")
else:
    lines += ['  (no items)']

with open(out_path, 'w', encoding='utf-8') as f:
    f.write('\n'.join(lines) + '\n')
print('* เขียนรายงานที่', out_path)
```

รูปที่ 4-25 generate_report (ส่วนที่ 4)

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

ระบบบริหารจัดการคลังพัสดุที่พัฒนาขึ้นนี้ ใช้หลักการจัดเก็บข้อมูลแบบไฟล์ไบนารี (Binary File) เพื่อให้สามารถจัดการข้อมูลได้อย่างมีประสิทธิภาพ โดยมีการออกแบบให้สามารถดำเนินการเพิ่ม (Add), แก้ไข (Update), ลบ (Delete) และ แสดงผลข้อมูล (View) ได้ครบวงจร รวมถึงสามารถสร้างรายงาน (Report) เพื่อสรุปความเคลื่อนไหวของพัสดุในคลังได้โปรแกรมนี้ประกอบด้วยฐานข้อมูล 3 ส่วนหลัก ได้แก่ ข้อมูลหมวดหมู่ (categories.bin) ข้อมูลพัสดุ (items.bin) และข้อมูลการเคลื่อนไหว (movements.bin) โดยแต่ละส่วนเชื่อมโยงกันผ่านรหัสอ้างอิง ช่วยให้สามารถติดตามประวัติของพัสดุแต่ละรายการได้อย่างถูกต้องและต่อเนื่อง ผลการทดสอบแสดงให้เห็นว่าโปรแกรมสามารถทำงานได้สมบูรณ์ เก็บและแสดงข้อมูลได้อย่างถูกต้องตามที่ออกแบบไว้

5.2 ปัญหาและอุปสรรคในการดำเนินงาน

ในระหว่างการพัฒนา พบว่าการทำงานกับไฟล์ Binary จำเป็นต้องระมัดระวังเรื่องขนาดของข้อมูลและรูปแบบ Struct เพราะหากกำหนดความยาวผิดจะทำให้ไม่สามารถอ่านไฟล์ได้ นอกจากนี้การอัปเดตข้อมูลที่เชื่อมโยงกัน เช่น การลบหมวดหมู่ที่ยังมีพัสดุอ้างอิงอยู่ ต้องมีการตรวจสอบเงื่อนไขก่อนทุกครั้งเพื่อป้องกันข้อผิดพลาด

5.3 ข้อเสนอแนะ

5.3.1 ควรเพิ่มระบบสำรองและกู้คืนข้อมูล (Backup/Restore) เพื่อป้องกันความเสียหายของไฟล์

5.3.2 ควรเชื่อมต่อกับฐานข้อมูลจริง เช่น SQLite หรือ MySQL เพื่อให้รองรับข้อมูลจำนวนมากได้ดียิ่งขึ้น

5.3.3 เพิ่มระบบกำหนดสิทธิ์ผู้ใช้ (Admin/Operator) เพื่อความปลอดภัยของข้อมูล

5.3 ข้อเสนอแนะ(ต่อ)

5.3.4 พัฒนาเป็นแบบ GUI แทนการพิมพ์คำสั่ง เพื่อให้ผู้ใช้ทั่วไปสามารถใช้งานได้ง่ายขึ้น

5.3.5 เพิ่มฟังก์ชันแจ้งเตือนพัสดุใกล้หมดหรือสถานะเสียหาย เพื่อช่วยให้การบริหารคลังมีประสิทธิภาพยิ่งขึ้น

5.4 สิ่งที่ผู้พัฒนาได้รับจากการพัฒนาโครงการ

จากการพัฒนาระบบนี้ ผู้พัฒนาได้รับความรู้และประสบการณ์ในด้านการออกแบบระบบและการเขียนโปรแกรมด้วยภาษา Python โดยเฉพาะในส่วนของการใช้โมดูล struct เพื่อจัดเก็บและอ่านข้อมูลแบบ Fixed-Length Record ในรูปแบบ Binary File ซึ่งช่วยเพิ่มความเข้าใจในการจัดการข้อมูลในระดับต่ำ (Low-level Data Handling) นอกจากนี้ยังได้ฝึกการวิเคราะห์ระบบ การออกแบบโครงสร้างไฟล์ การจัดการข้อผิดพลาด (Exception Handling) และการจัดรูปแบบรายงานให้อ่านง่ายและเหมาะสมกับผู้ใช้งานจริง ซึ่งเป็นประสบการณ์ที่สามารถนำไปต่อยอดกับการพัฒนาโปรแกรมในระดับที่ซับซ้อนยิ่งขึ้นในอนาคต