ระบบจัดการสต็อกสินค้า
Inventory Management System

นายธนศักดิ์ เขจรลาภ	รหัสนักศึกษา 6706022510077
นางสาวมณฑากาญจน์ โรจนะ	รหัสนักศึกษา 6706022510042
นางสาวพรญาณี สมดี	รหัสนักศึกษา 6706022510140
นางสาวภาสินี เก่งกลาง	รหัสนักศึกษา 6706022510158
นางสาวกัญชลิกา บำรุงผล	รหัสนักศึกษา 6706022510239

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2567

คำนำ

รายงานฉบับนี้จัดทำขึ้นเพื่อสรุปผลการศึกษาและการพัฒนาระบบจัดการสต็อกสินค้า ซึ่งเป็นส่วนหนึ่งของการเรียนในวิชา Computer Programming วิชานี้มุ่งเน้นการพัฒนาทักษะการเขียนโปรแกรมและการประยุกต์ใช้ความรู้ด้าน Programming ในการสร้างระบบที่สามารถนำไปใช้งานได้จริง ระบบจัดการสต็อกสินค้าเป็นตัวอย่างของการนำแนวคิดการเขียนโปรแกรมมาใช้แก้ไขปัญหาในชีวิตประจำวันของธุรกิจ เนื้อหาภายในรายงานจะครอบคลุมขั้นตอนการออกแบบและพัฒนาระบบตั้งแต่การวิเคราะห์ความต้องการ (Requirements Analysis) การเขียนอัลกอริทึม และการใช้ภาษาคอมพิวเตอร์ในการพัฒนาระบบ พร้อมทั้งอธิบายกระบวนการทำงานของแต่ละฟังก์ชันภายในระบบ

รายงานฉบับนี้ได้จัดทำขึ้นเพื่อเป็นการสรุปประสบการณ์และความรู้ที่ได้จากการเรียนและพัฒนาระบบจริง หวังเป็นอย่างยิ่งว่าจะเป็นประโยชน์ทั้งต่อผู้ศึกษาและผู้ที่สนใจในการนำความรู้ทาง Programming ไปประยุกต์ใช้ในการพัฒนาระบบจัดการต่าง ๆ ในอนาคต

สารบัญ

สารบัญภาพ

บทที่ 1
บทนำ

1.1 วัตถุประสงค์ของโครงงาน

1.1.1 เพื่อพัฒนาระบบที่สามารถจัดการสต็อกสินค้าได้อย่างมีประสิทธิภาพ

1.1.2 เพื่อฝึกทักษะการเขียนโปรแกรมภาษา Python

1.1.3 เพื่อฝึกกระบวนการคิดอย่างเป็นระบบ

1.1.4 เพื่อเรียนรู้การจัดการข้อมูลและไฟล์

1.2 ขอตเขตของโครงงาน

1.2.1 ระบบจัดการสต็อกสินค้ามีฟังก์ชันในการทำงาน 7 ฟังก์ชัน ได้แก่

1.2.1.1 Display Product: แสดงข้อมูลสินค้า

1.2.1.2 Add Product: เพิ่มสินค้า

1.2.1.3 Update Product: อัพเดตข้อมูลสินค้า

1.2.1.4 Delete Product: ลบสินค้า

1.2.1.5 Display Category Product: แสดงข้อมูลสินค้าในระบบตามประเภท

1.2.1.6 Export Inventory Report: ส่งออกรายงานข้อมูลสินค้าในระบบ

1.2.1.7 Exit: หยุดการทำงานของโปรแกรม

1.2.2 ในการจัดทำระบบจัดการสต็อกสินค้าประกอบไปด้วย 5 ฟิลด์ ได้แก่

1.2.2.1 Product_ID: รหัสสินค้า

1.2.2.2 Product_Name: ชื่อสินค้า

1.2.2.3 Product_Category: ประเภทของสินค้า

1.2.2.4 Product_Quantity: จำนวนสินค้าในสต็อก

1.2.2.5 Product_Price: ราคาสินค้าต่อหน่วย

1.3 ประโยชน์ที่ได้รับ

1.3.1 พัฒนาทักษะการเขียนโปรแกรม

1.3.2 เพิ่มประสิทธิภาพในการจัดการสต็อกสินค้า

1.3.3 ฝึกการคิดวิเคราะห์และแก้ไขปัญหา

1.3.4 ความเข้าใจในการทำงานร่วมกับฐานข้อมูล

1.4 เครื่องมือที่คาดว่าจะต้องใช้

1.4.1 ภาษาโปรแกรมที่ใช้ในการพัฒนาระบบ คือ Python สำหรับการพัฒนาโปรแกรมพื้นฐานที่เข้าใจง่ายและมีไลบรารีหลากหลายที่ช่วยจัดการข้อมูลได้ดี

1.4.2 Visual Studio Code เป็นโปรแกรม Text Editor ที่ออกแบบมาเพื่อให้เป็นเครื่องมือที่ช่วยในการเขียนโค้ด โดยสามารถรองรับภาษาโปรแกรมต่าง ๆ เช่น Python, JavaScript และอื่น ๆ

1.4.3 GitHub แพลตฟอร์มที่ช่วยในการเก็บโค้ดและทำงานร่วมกับผู้อื่นในการทำโปรเจ็กต์

บทที่ 2
ระบบจัดการสต็อกสินค้า

2.1 ฟิลด์ในระบบจัดการสต็อกสินค้า

การจัดการข้อมูลหนังสือในระบบประกอบไปด้วย 5 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญ ดังนี้

2.1.1 Product_ID: รหัสสินค้า

หลักการทำงานคือ รหัสสินค้าคือรหัสเฉพาะที่ไม่ซ้ำกัน ใช้ระบุและแยกแยะสินค้าแต่ละชิ้นในระบบสต็อกสินค้า รหัสนี้เป็นตัวเลข

2.1.2 Product_Name: ชื่อสินค้า

หลักการทำงานคือ ชื่อสินค้าคือชื่อที่ระบุถึงสินค้านั้น ๆ ช่วยให้ผู้ใช้ระบบสามารถเข้าใจและระบุประเภทหรือรายละเอียดของสินค้าตามชื่อได้อย่างชัดเจน มีความสำคัญคือ ทำให้สามารถตรวจสอบสถานะสินค้าหรือสั่งซื้อได้อย่างรวดเร็ว

2.1.3 Product_Category: ประเภทของสินค้า

หลักการทำงานคือ ประเภทของสินค้าเป็นการจัดกลุ่มสินค้าที่อยู่ในประเภทเดียวกัน การแบ่งประเภทช่วยให้การจัดการและค้นหาสินค้าเป็นไปได้อย่างเป็นระเบียบและง่ายดาย มีความสำคัญคือ ช่วยให้การจัดการสินค้ามีประสิทธิภาพมากขึ้น โดยเฉพาะเมื่อระบบมีสินค้าหลากหลายประเภท การค้นหาสินค้าจะเร็วขึ้น และตรวจสอบหรือการจัดการข้อมูลสินค้าได้ง่ายขึ้น

2.1.4 Product_Quantity: จำนวนสินค้าในสต็อก

มีหลักการทำงานคือ จำนวนสินค้าในสต็อกหมายถึงปริมาณสินค้าที่มีอยู่ ซึ่งจะถูกบันทึกและอัพเดตในระบบเมื่อมีการเพิ่มหรือจำหน่ายสินค้าออกจากสต็อก มีความสำคัญคือ การตรวจสอบจำนวนสินค้าเป็นสิ่งสำคัญเพื่อให้มั่นใจว่าสินค้ามีเพียงพอต่อความต้องการของลูกค้า การติดตามจำนวนสินค้ายังช่วยในการวางแผนการสั่งซื้อหรือผลิตสินค้า

2.1.5 Product_Price: ราคาสินค้าต่อหน่วย

มีหลักการทำงานคือ ราคาสินค้าต่อหน่วยเป็นราคาที่ถูกกำหนดไว้สำหรับสินค้าหนึ่งหน่วย ซึ่งใช้ในการคำนวณมูลค่ารวมของสินค้าที่ถูกจำหน่ายหรือในสต็อกมีความสำคัญคือ ราคาสินค้ามีความสำคัญในการคำนวณรายได้และต้นทุน การเก็บข้อมูลราคาสินค้าต่อหน่วยในระบบช่วยให้สามารถคำนวณยอดขาย คำนวณกำไร

2.2 ฟังก์ชันการใช้งานในโปรแกรมระบบจัดการสต็อกสินค้า

ระบบจัดการสต็อกสินค้าทำงานผ่านเมนูหลักจากไฟล์ main.py โดยให้ผู้ใช้สามารถจัดการสินค้าต่าง ๆ ในสต็อกได้ โดยอาศัยฟังก์ชันจากไฟล์ function_inventory.py

2.2.1 ฟังก์ชัน main()

เป็นฟังก์ชันหลักที่ทำหน้าที่ควบคุม การทำงานของระบบ โดยจะแสดงเมนูให้ผู้ใช้ได้เลือก และดำเนินการตามที่ผู้ใช้เลือก เช่น แสดงข้อมูลสินค้า เพิ่มสินค้า ลบสินค้า อัปเดตสินค้าในสต็อก และหยุดการทำงานของโปรแกรม

2.2.2 โครงสร้างและการทำงานฟังก์ชัน main()

2.2.2.1 การกำหนดตัวแปร

มีการกำหนดตัวแปร running ให้เป็นตัวแปรที่ใช้ควบคุมการทำงานของลูป และ inventory สร้างออบเจ็กต์จากคลาส Inventory โดยใช้ไฟล์ inventory.bin เพื่อเก็บข้อมูลสินค้า

```
from function_inventory import *

def main():
    running = True
    inventory = Inventory('inventory.bin')
```

ภาพที่ 2-1 ภาพการกำหนดตัวแปร

2.2.2.2 ลูปการทำงานหลัก

ลูปการทำงานนี้จะทำงานต่อไป จนกว่าตัวแปร running จะถูกตั้งค่าเป็น False เมื่อโปรแกรมเริ่มต้นการทำงาน ฟังก์ชัน main() จะทำการแสดงเมนูหลักที่มีตัวเลือกต่างๆ ให้กับผู้ใช้ เช่น แสดงข้อมูลสินค้า เพิ่มสินค้า ลบสินค้า อัปเดตสินค้าในสต็อก และหยุดการทำงานของโปรแกรม

```
    while running:
        print('\nMenu')
        print('1. Display Product')
        print('2. Add Product')
        print('3. Update Product')
        print('4. Delete Product')
        print('5. Display Category Product')
        print('6. Export Inventory Report')
        print('7. Exit')
```

ภาพที่ 2-2 แสดงเมนูในลูปการทำงานหลัก

2.2.2.3 การรับค่าจากผู้ใช้

รับค่าจากผู้ใช้เพื่อระบุเมนูที่ต้องการเลือก โดยมีการแปลงค่าที่ได้ให้เป็น int

```
try:
    menu = int(input('Enter your menu choice: '))
```

ภาพที่ 2-3 คำสั่งการรับค่าจากผู้ใช้

2.2.2.4 การจัดการเมนู

1) เมนู 1 Display Product คือการแสดงสินค้าทั้งหมดในระบบ มีการเรียกใช้ฟังก์ชัน display_products() เพื่อนำเสนอสินค้าทั้งหมดในระบบ

```
if menu == 1:
    inventory.display_products()
```

ภาพที่ 2-4 คำสั่งการทำงานเมนู 1

2) เมนู 2 Add Product คือเมนูการเพิ่มสินค้าลงในระบบ โดยให้ผู้ใช้กรอกรหัสสินค้า (Product ID) และตรวจสอบว่าซ้ำหรือไม่ ถ้าซ้ำจะแจ้งเตือนให้กรอกใหม่ จากนั้นรับข้อมูลสินค้า เรียกใช้ฟังก์ชัน add_product เพื่อเพิ่มข้อมูลสินค้าลงในระบบ

```
elif menu == 2:
    print('Add Product')
    while True:
        try:
            product_id = int(input('Enter Product ID: '))
            # ตรวจสอบว่ารหัสสินค้าซ้ำหรือไม่
            if inventory._check_product_exists(product_id):
                print(f"Product ID {product_id} already exists. Please enter a different ID.")
                continue  # ถ้าซ้ำ ให้เริ่มลูปใหม่
            break  # ออกจากลูปหากไม่มีรหัสซ้ำ
        except ValueError:
            print("Invalid input. Please enter a valid integer for Product ID.")

    # รับข้อมูลสินค้าอื่น ๆ
    while True:
        try:
            product_name = input('Enter Product Name: ')
            product_category = input('Enter Product Category: ')
            product_quantity = int(input('Enter Product Quantity: '))
            product_price = float(input('Enter Product Price: '))
            inventory.add_product(product_id, product_name, product_category, product_quantity, product_price)
            break  # ออกจากลูปถ้าข้อมูลถูกต้อง
        except ValueError:
            print("Invalid input. Please enter a valid number for Quantity and Price.")
```

ภาพที่ 2-5 คำสั่งการเพิ่มสินค้า

3) เมนู 3 Update Product คือเมนูการอัปเดตสินค้าลงในระบบ โดยจะแสดงสินค้า และให้ผู้ใช้กรอกรหัสสินค้าที่ต้องการอัปเดต หากไม่มีรหัสอยู่ในระบบ จะให้กรอกรหัสสินค้าใหม่ จากนั้นจะรับข้อมูลใหม่และเรียกใช้ฟังก์ชัน update_productมาอัปเดตข้อมูล โดยหากมีการกรอกข้อมูลที่ไม่ถูกต้อง (เช่น การกรอกจำนวนหรือราคาเป็นข้อความ) จะจับข้อผิดพลาดและแสดงข้อความแจ้งให้ผู้ใช้ทราบ

```python
elif menu == 3:
    print('Update Product')
    inventory.display_products()  # แสดงรายการสินค้าก่อนเพื่อให้ผู้ใช้เลือก

    while True:
        try:
            product_id = int(input('Enter Product ID to update: '))
            # เช็คว่ามี Product ID อยู่ในระบบหรือไม่
            if not inventory._check_product_exists(product_id):
                print(f"Product ID {product_id} does not exist. Please enter a valid ID.")
                continue  # ถ้าไม่พบ ให้เริ่มลูปใหม่

            # รับข้อมูลใหม่
            product_name = input('Enter new Product Name (leave blank to keep current): ')
            product_category = input('Enter new Product Category (leave blank to keep current): ')
            product_quantity = input('Enter new Product Quantity (leave blank to keep current): ')
            product_price = input('Enter new Product Price (leave blank to keep current): ')

            # แปลงข้อมูลที่กรอกใหม่ให้เป็นชนิดที่ถูกต้อง
            product_quantity = int(product_quantity) if product_quantity else None
            product_price = float(product_price) if product_price else None

            # เรียกใช้ฟังก์ชันอัปเดต
            inventory.update_product(product_id,
                                     product_name if product_name else None,
                                     product_category if product_category else None,
                                     product_quantity,
                                     product_price)

            break  # ออกจากลูปถ้าลบสำเร็จ
        except ValueError:
            print("Invalid input. Please enter valid numbers for Quantity and Price.")
```

ภาพที่ 2-6 คำสั่งการอัปเดตสินค้า

4) เมนู 4 Delete Product คือเมนูการลบสินค้าออกจากระบบ แสดงสินค้าทั้งหมดและให้ผู้ใช้กรอกรหัสของสินค้าที่ต้องการลบ จากนั้นตรวจสอบว่ามีรหัสสินค้านั้นอยู่ในระบบหรือไม่ หากไม่พบจะแจ้งเตือน และเรียกใช้ฟังก์ชัน delete_product เพื่อลบข้อมูลสินค้า

```python
elif menu == 4:
    print('Delete Product')
    inventory.display_products()
    while True:
        try:
            product_id = int(input('Enter Product ID to delete: ')) # เช็คว่ามี Product ID อยู่ในระบบหรือไม่

            if not inventory._check_product_exists(product_id):
                print(f"Product ID {product_id} does not exist. Please enter a valid ID.")
                continue  # ถ้าไม่พบ ให้เริ่มลูปใหม่

            # ถ้าพบให้ลบ
            inventory.delete_product(product_id)
            break  # ออกจากลูปถ้าลบสำเร็จ
        except ValueError:
            print("Invalid input. Please enter a valid integer for Product ID.")
```

ภาพที่ 2-7 คำสั่งการลบสินค้า

5) เมนู 5 Display Category Product คือเมนูแสดงสินค้าตามหมวดหมู่ โดยเรียกใช้ฟังก์ชัน generate_inventory_report เพื่อนำเสนอสินค้า

```
elif menu == 5:    # เมนูสำหรับแสดงสินค้าตามหมวดหมู่
    inventory.generate_inventory_report()
```

ภาพที่ 2-8 คำสั่งการแสดงสินค้าตามหมวดหมู่

6) เมนู 6 Export Inventory Report คือเมนูสร้างรายงานสินค้าตามหมวดหมู่ในรูปแบบไฟล์ข้อความ โดยการเรียกใช้ฟังก์ชัน export_inventory_report เพื่อสร้างรายงานสินค้าในระบบให้ไฟล์ข้อความชื่อ inventory_report.txt

```
elif menu == 6:    # เมนูสำหรับสร้างรายงาน
    inventory.export_inventory_report('inventory_report.txt')
```

ภาพที่ 2-9 คำสั่งสร้างรายงานสินค้า

7) เมนู 7 Exit คือเมนที่หยุดการการทำงานของโปรแกรม โดยมีการให้หยุดลูปการทำงาน จากการตั้งค่าตัวแปร running เป็น False

```
elif menu == 7:    # เมนูหยุดทำงาน
    print('Exiting the program.')
    running = False
```

ภาพที่ 2-10 คำสั่งการหยุดการทำงาน

2.2.2.5 กรณีการใส่เมนูที่ไม่ถูกต้อง

จะมีการแจ้งเตือนผู้ใช้ว่าตัวเลือกไม่ถูกต้อง จากนั้นจะวนกลับไปที่จุดเริ่มต้นของลูป while running เพื่อให้ผู้ใช้สามารถเลือกเมนูใหม่ได้

```
else:
    print('Invalid Menu choice. Please try again.')
```

ภาพที่ 2-11 คำสั่งแจ้งเตือนการเลือกเมนูไม่ถูกต้อง

2.2.2.6 การจัดการข้อผิดพลาด

ส่วนนี้จะดักจับ ข้อผิดพลาดทุกประเภท ที่เกิดขึ้นในคำสั่งรับค่าเมนูจากผู้ใช้ เช่นการพิมพ์ตัวอักษรแทนตัวเลข

```
except Exception as e:
    print(f"An error occurred: {e}")
```

ภาพที่ 2-12 คำสั่งจัดการข้อผิดพลาด

2.3 ฟังก์ชันการทำงานในระบบจัดการสต็อกสินค้า

ระบบจัดการสต็อกสินค้า มีการแยกฟังก์ชันออกมาไว้ในไฟล์ function_inventory.py โดยไฟล์ function_inventory.py จะเก็บฟังก์ชันต่าง ๆ ที่ใช้ในระบบจัดการสต็อกสินค้า ส่วนโปรแกรมหลักในไฟล์ main.py จะทำการเรียกใช้ฟังก์ชัน

2.3.1 การใช้งานโมดูล

นำเข้าโมดูล struct ซึ่งใช้สำหรับการบีบอัดและขยายข้อมูลไบนารี (pack และ unpack) เพื่อจัดการกับข้อมูลที่ถูกจัดเก็บในรูปแบบไบนารี และโมดูล os ซึ่งใช้จัดการไฟล์และไดเร็กทอรี โดยสามารถสร้าง ลบ เปลี่ยนชื่อ และตรวจสอบการมีอยู่ของไฟล์และไดเร็กทอรีในระบบได้

```
import struct
import os
```

ภาพที่ 2-13 การใช้งานโมดูล

2.3.2 โครงสร้างของข้อมูล

สร้างตัวแปร PRODUCT_FORMAT เป็นตัวแปรที่เก็บรูปแบบของข้อมูลที่ใช้ในการบีบอัดและขยายข้อมูล มีการกำหนดรูปแบบของข้อมูลคือ 'I30s30sIf' ซึ่งในแต่ละฟิลด์ในผลิตภัณฑ์คือ

2.3.2.1 I แทน int สำหรับ Product_ID ซึ่งจะเก็บค่าตัวเลขที่ไม่ติดลบที่

2.3.2.2 30s แทน string ขนาด 30 ไบต์ สำหรับ Product_Name ซึ่งจะเก็บชื่อของสินค้าได้สูงสุด 30 ตัวอักษร

2.3.2.3 30s แทน string ขนาด 30 ไบต์ สำหรับ Product_Category ซึ่งจะเก็บชื่อหมวดหมู่ได้สูงสุด 30 ตัวอักษร

2.3.2.4 I แทน int สำหรับ Product_Quantity ซึ่งจะเก็บจำนวนสินค้าที่มี

2.3.2.5 f แทน float สำหรับ Product_Price ซึ่งจะเก็บราคาในรูปแบบทศนิยม

```
# กำหนดโครงสร้างของข้อมูล
PRODUCT_FORMAT = 'I30s30sIf'  # (Product_ID, Product_Name, Product_Category, Product_Quantity, Product_Price)
PRODUCT_SIZE = struct.calcsize(PRODUCT_FORMAT)
```

ภาพที่ 2-14 โครงสร้างของข้อมูล

2.3.3 การสร้างคลาส Inventory

สร้างคลาส Inventory โดยมีตัวแปร file_path เก็บตำแหน่งของไฟล์ที่ใช้จัดเก็บข้อมูล และมีเมธอด __init__ เป็น constructor ของคลาสเพื่อกำหนดตำแหน่งของไฟล์สินค้าที่จะใช้ในการอ่านหรือเขียนข้อมูล

```
class Inventory:
    def __init__(self, file_path):
        self.file_path = file_path
```

ภาพที่ 2-15 การสร้างคลาส Inventory

2.3.4 ฟังก์ชัน _check_product_exists

เป็นฟังก์ชันเช็คการมีอยู่ของสินค้าโดยการรับ product_id จากนั้นเปิดไฟล์ในโหมดอ่าน (read binary) วนลูปเพื่ออ่านข้อมูลจนกว่าจะหมด ถ้าตรวจสอบว่า product_id ตรงกับสินค้าที่มีในระบบ จะคืนค่า True แล้วทำงานในขั้นตอนต่อไป หากไม่พบสินค้า คืนค่า False ใช้การ except FileNotFoundError จัดการข้อผิดพลาดถ้าไฟล์ไม่พบ

```
def _check_product_exists(self, product_id):
    """เช็คว่า product_id มีอยู่ในไฟล์หรือไม่"""
    try:
        with open(self.file_path, 'rb') as file:
            while True:
                data = file.read(PRODUCT_SIZE)
                if not data:
                    break
                product = struct.unpack(PRODUCT_FORMAT, data)
                if product[0] == product_id:
                    return True
    except FileNotFoundError:
        return False
    return False
```

ภาพที่ 2-16 คำสั่งฟังก์ชันเช็คการมีอยู่ของสินค้า

2.3.5 ฟังก์ชัน add_product

2.3.5.1 สร้างฟังก์ชันเพิ่มข้อมูลสินค้าลงในไฟล์ โดยมีพารามิเตอร์ที่ได้รับคือ ชื่อไฟล์เก็บข้อมูล รหัสสินค้า ชื่อสินค้า หมวดหมู่สินค้า จำนวน และราคา

```
def add_product(self, product_id, product_name, product_category, product_quantity, product_price):
```

ภาพที่ 2-17 สร้างฟังก์ชันเพิ่มสินค้า

2.3.5.2 การตรวจสอบความยาวของชื่อและหมวดหมู่สินค้าว่าต้องไม่เกิน 30 ตัวอักษร

```
if len(product_name) > 30 or len(product_category) > 30:
    print("Product name and category must not exceed 30 characters!")
    return
```

ภาพที่ 2-18 การตรวจสอบความยาวชื่อและหมวดหมู่

2.3.5.3 มีกำหนดหมวดหมู่เป็น Other ถ้าไม่มีการระบุ และเช็คว่ามี product_id นี้อยู่แล้วหรือไม่

```
# ตั้งค่าหมวดหมู่เป็น "Other" ถ้าไม่มีการกำหนด
if product_category is None or product_category.strip() == "":
    product_category = "Other"

# เช็คว่ามี product_id ซ้ำหรือไม่
if self._check_product_exists(product_id):
    print(f"Unable to add product: Product_ID {product_id} already exists!")
    return
```

ภาพที่ 2-19 การกำหนดและตรวจสอบรหัสสินค้า

2.3.5.4 มีการเริ่มบล็อกเพื่อตรวจจับข้อผิดพลาด และใช้คำสั่ง strip().lower().title() เพื่อให้ตัวอักษรที่เป็นตัวแรกของแต่ละคำเป็นตัวพิมพ์ใหญ่ จากนั้น เขียนข้อมูลที่บีบอัดลงในไฟล์ เมื่อเสร็จสิ้นจะแสดงข้อความ และรอให้ผู้ใช้กด Enter ก่อนกลับไปที่เมนูหลัก

```
try:
    product_category = product_category.strip().lower().title()  # ทำให้ตัวแรกของแต่ละคำเป็นตัวพิมพ์ใหญ่
    product_name = product_name.strip().lower().title()  # ทำให้ตัวแรกของแต่ละคำเป็นตัวพิมพ์ใหญ่

    with open(self.file_path, 'ab') as file:
        data = struct.pack(PRODUCT_FORMAT,
                           product_id,
                           product_name.encode('utf-8'),
                           product_category.encode('utf-8'),
                           product_quantity,
                           product_price
                           )
        file.write(data)
        print("Product added successfully!")

    input("Press Enter to return to the main menu...")  # รอให้ผู้ใช้กด Enter ก่อนที่จะกลับไปที่เมนู

except Exception as e:
    print(f"An error occurred: {e}")
```

ภาพที่ 2-20 การเพิ่มสินค้าลงในระบบ

2.3.6 ฟังก์ชัน display_products

2.3.6.1 การเปิดไฟล์และการอ่านข้อมูลสินค้าในโหมดอ่านแบบไบนารีโดยใช้ with และมีการกำหนด products = [ ] สร้างรายการว่างเพื่อเก็บสินค้าที่อ่านได้ มีการวนลูปอ่านข้อมูล ถ้าไม่มีข้อมูลให้หยุดการวนลูป จากนั้นแปลงข้อมูลไบนารีเป็นทูเพิลตามรูปแบบที่กำหนด เพิ่มสินค้าที่อ่านได้ลงในรายการ มีเรียงรายการด้วยตาม ID ของสินค้า function lambda

```
def display_products(self):
    try:
        with open(self.file_path, 'rb') as file:
            products = []
            # อ่านข้อมูลสินค้าทั้งหมดและเก็บไว้ในรายการ
            while True:
                data = file.read(PRODUCT_SIZE)
                if not data:
                    break
                product = struct.unpack(PRODUCT_FORMAT, data)
                products.append(product)# เพิ่มสินค้าที่อ่านได้ลงในรายการ
            # เรียงสินค้าตาม ID (ตัวแรกในแต่ละทูเพล)
            products.sort(key=lambda x: x[0])
```

ภาพที่ 2-21 การเปิดไฟล์และอ่านข้อมูล

2.3.6.2 การแสดงข้อมูลสินค้าตามรูปแบบที่กำหนด ใช้การวนลูปเพื่อแสดงข้อมูลสินค้าทั้งหมดในระบบ และทำการแปลงข้อมูลด้วยการ decode('utf-8').strip('\x00') ที่แปลงชื่อสินค้าและหมวดหมู่จากไบนารีเป็นสตริง และลบค่า null ถ้ามี

```
    print("Display product list:")
    headers = ['ID', 'Name', 'Category', 'Quantity', 'Price(THB)']
    print(f"{headers[0]:<5} {headers[1]:<30} {headers[2]:<30} {headers[3]:<10} {headers[4]:<10}")

    for product in products:
        # แปลงข้อมูลให้ถูกต้อง
        id_value = product[0]
        name_value = product[1].decode('utf-8').strip('\x00')
        category_value = product[2].decode('utf-8').strip('\x00')
        quantity_value = product[3]
        price_value = product[4]
        # แสดงผล
        print(f"{id_value:<5} {name_value:<30} {category_value:<30} {quantity_value:<10} {price_value:<10,.2f}")
input("Press Enter to return to the main menu...")  # รอให้ผู้ใช้กด Enter ก่อนที่จะกลับไปที่เมนู
```

ภาพที่ 2-22 แสดงสินค้าตามรูปแบบที่กำหนด

2.3.6.3 มีการตรวจจับและจัดการข้อผิดพลาดถ้าไฟล์ไม่พบหรือมีข้อผิดพลาดอื่น ๆ ในระหว่างการทำงานของฟังก์ชัน

```
except FileNotFoundError:
    print("File not found!")
except Exception as e:
    print(f"An error occurred while displaying the product: {e}")
```

ภาพที่ 2-23 การตรวจจับข้อผิดพลาดในฟังก์ชันแสดงสินค้า

2.3.7 ฟังก์ชัน generate_inventory_report

2.3.7.1 การเปิดไฟล์และการอ่านข้อมูลสินค้า มีการสร้างพจนานุกรม categories = {} เพื่อเก็บข้อมูลสินค้าแบ่งตามประเภท ทำการวนลูปเพื่ออ่านข้อมูลในไฟล์ หยุดการอ่านเมื่อไม่มีข้อมูล

```
def generate_inventory_report(self):
    """สร้างรายงานสินค้าที่จัดกลุ่มตามประเภท"""
    try:
        with open(self.file_path, 'rb') as file:
            categories = {}

            while True:
                data = file.read(PRODUCT_SIZE)
                if not data:
                    break
                product = struct.unpack(PRODUCT_FORMAT, data)
```

ภาพที่ 2-24 การอ่านข้อมูลสินค้าตามประเภท

2.3.7.2 การจัดกลุ่มข้อมูลตามประเภท โดยแปลงข้อมูลจากทูเพิลเป็นค่าคือ ID, ชื่อ, หมวดหมู่, จำนวน และราคา ถ้าหมวดหมู่ยังไม่อยู่ในพจนานุกรม categories ให้เพิ่มเข้าไป จากนั้นเพิ่มข้อมูลสินค้าลงในรายการของหมวดหมู่นั้น

```
# แปลงข้อมูลให้ถูกต้อง
id_value = product[0]
name_value = product[1].decode('utf-8').strip('\x00')
category_value = product[2].decode('utf-8').strip('\x00')
quantity_value = product[3]
price_value = product[4]

# เพิ่มข้อมูลสินค้าไปยังหมวดหมู่ที่ถูกต้อง
if category_value not in categories:
    categories[category_value] = []
categories[category_value].append((id_value, name_value, quantity_value, price_value))
```

ภาพที่ 2-25 การจัดกลุ่มประเภทสินค้า

2.3.7.3 แสดงข้อมูลสินค้าในแต่หมวดหมู่ตามที่ได้กำหนดรูปแบบการแสดงผล และรอให้ผู้ใช้กดปุ่ม Enter ก่อนที่จะกลับไปที่เมนูหลัก

```
# แสดงรายงาน
print(f"\nInventory Report:\nNumber of categories: {len(categories)}\n")

for category, products in categories.items():
    total_price = sum(product[2] * product[3] for product in products)  # คำนวณราคารวมของหมวดหมู่

    print(f"Category : {category}")
    print(f"Number of Products: {len(products)}")
    headers = ['ID', 'Name', 'Quantity', 'Price(THB)']
    print('-'*70)
    print(f"{headers[0]:<7} {headers[1]:<30} {headers[2]:<10} {headers[3]:<10} ")
    print('-'*70)

    for product in products:
        print(f"{product[0]:03}{'':<4} {product[1]:<30} {product[2]:<10} {product[3]:<10,.2f}")

    print('-'*70)
    print(f"Total Price : {total_price:,.2f} THB")  # แสดงราคารวมของหมวดหมู่
    print('-'*70)
    print()  # เพื่อให้มีการเว้นบรรทัดระหว่างหมวดหมู่

input("Press Enter to return to the main menu...")  # รอให้ผู้ใช้กด Enter ก่อนที่จะกลับไปที่เมนู
```

ภาพที่ 2-26 การแสดงสินค้าตามหมวดหมู่ในรูปแบบที่กำหนด

2.3.7.4 จัดการข้อผิดพลาดถ้าไฟล์ไม่พบหรือมีข้อผิดพลาดอื่น ๆ ในระหว่างการทำงานของฟังก์ชันการแสดงผลตามหมวดหมู่

```
except FileNotFoundError:
    print("File not found!")
except Exception as e:
    print(f"An error occurred while generating the report: {e}")
```

ภาพที่ 2-27 การจัดข้อผิดพลาดในการแสดงผลตามหมวดหมู่

2.3.8 ฟังก์ชัน delete_product

2.3.8.1 การสร้างตัวแปร temp_file_path กำหนดชื่อไฟล์ชั่วคราวที่จะใช้ในการเก็บข้อมูลที่ไม่ถูกลบ และ product_found ตัวแปรที่ใช้ติดตามว่าสินค้าที่ต้องการลบเจอหรือไม่

```
def delete_product(self, product_id):
    """ลบสินค้าจากไฟล์ตาม product_id"""
    temp_file_path = 'temp_inventory.bin'
    product_found = False
```

ภาพที่ 2-28 การกำหนดตัวแปรในฟังก์ชันการลบสินค้า

2.3.8.2 การอ่านข้อมูลสินค้าและการตรวจสอบ วนลูปเพื่ออ่านข้อมูลสินค้าจากไฟล์ อ่านข้อมูลขนาด PRODUCT_SIZE และทำการหยุดการอ่านเมื่อไม่มีข้อมูล จากนั้นตรวจสอบว่ารหัสสินค้าที่ได้ตรงกับ product_id หรือไม่ ถ้าใช่ จะตั้งค่าตัวแปร product_found เป็น True และไม่เขียนสินค้านี้ลงในไฟล์ชั่วคราว ถ้าไม่ใช่ ให้เขียนข้อมูลสินค้าลงในไฟล์ชั่วคราว

```
try:
    with open(self.file_path, 'rb') as file, open(temp_file_path, 'wb') as temp_file:
        while True:
            data = file.read(PRODUCT_SIZE)
            if not data:
                break
            product = struct.unpack(PRODUCT_FORMAT, data)

            if product[0] == product_id:
                product_found = True  # เจอสินค้าที่จะลบ
                continue  # ไม่เขียนสินค้านี้ลงในไฟล์ชั่วคราว

            temp_file.write(data)   # เขียนสินค้าที่ไม่ถูกลบไปยังไฟล์ชั่วคราว
```

ภาพที่ 2-29 การอ่านข้อมูลสินค้าและการตรวจสอบ

2.3.8.3 การจัดการไฟล์หลังจากการลบ ใช้โมดูล os.replace เพื่อเปลี่ยนชื่อไฟล์ชั่วคราวให้เป็นไฟล์หลัก และลบไฟล์หลักเก่า แสดงข้อความยืนยันการลบ ถ้าไม่พบสินค้าให้ลบไฟล์ชั่วคราวด้วย os.remove และแสดงข้อความว่าหาสินค้าไม่พบ

```
if product_found:
    # เปลี่ยนชื่อไฟล์ชั่วคราวเป็นไฟล์หลัก
    import os
    name_value = product[1].decode('utf-8').strip('\x00')
    os.replace(temp_file_path, self.file_path)
    print(f"Product ID {product_id} {name_value} successfully deleted!")
else:
    os.remove(temp_file_path)  # ลบไฟล์ชั่วคราวถ้าไม่พบสินค้า
    print(f"Product ID {product_id} not found!")
```

ภาพที่ 2-30 การจัดการไฟล์หลังจากการลบ

2.3.8.4 จัดการข้อผิดพลาดถ้าไฟล์ไม่พบหรือมีข้อผิดพลาดอื่น ๆ ในระหว่างการทำงานของฟังก์ชันการลบสินค้า

```
except FileNotFoundError:
    print("File not found!")
except Exception as e:
    print(f"An error occurred while deleting the product: {e}")
```

ภาพที่ 2-31 การจัดการข้อผิดพลาดในฟังก์ชั่นการลบ

2.3.9 ฟังก์ชัน update_product

2.3.9.1 การตั้งค่าเริ่มต้น กำหนดพารามิเตอร์บางตัวเป็น None โดยหากไม่ได้ส่งค่ามาฟังก์ชัน จะไม่อัปเดตข้อมูลนั้น สร้างตัวแปรกำหนดชื่อไฟล์ชั่วคราว temp_inventory.bin สำหรับเก็บข้อมูลสินค้าใหม่หลังจากการอัปเดต และกำหนดตัวแปร product_found เพื่อใช้ตรวจสอบว่าสินค้าที่ต้องการอัปเดตมีในระบบหรือไม่

```
def update_product(self, product_id, product_name=None, product_category=None, product_quantity=None, product_price=None):
    """อัปเดตข้อมูลสินค้าตาม product_id"""
    temp_file_path = 'temp_inventory.bin'
    product_found = False
```

ภาพที่ 2-32 การตั้งค่าเริ่มต้น

2.3.9.2 การอ่านข้อมูลสินค้าและตรวจสอบสินค้าที่จะอัปเดต มีการเริ่มบล็อกจับข้อผิดพลาด โดยเปิดไฟล์หลักที่เก็บข้อมูลเดิมในโหมดอ่านไบนารี ('rb') และไฟล์ชั่วคราวที่จะเก็บข้อมูลหลังการอัปเดตในโหมดเขียนแบบไบนารี ('wb') วนลูปเพื่ออ่านข้อมูลในไฟล์หลัก มีการใช้ struct.unpack เพื่อแปลงข้อมูลจากไบนารีเป็นทูเพิล ตรวจสอบว่ารหัสสินค้าที่อ่านได้ตรงกับ product_id หรือไม่ ถ้าตรง จะตั้งค่า product_found เป็น True

```
try:
    with open(self.file_path, 'rb') as file, open(temp_file_path, 'wb') as temp_file:
        while True:
            data = file.read(PRODUCT_SIZE)
            if not data:
                break
            product = struct.unpack(PRODUCT_FORMAT, data)

            if product[0] == product_id:
                product_found = True  # เจอสินค้าที่จะอัปเดต
```

ภาพที่ 2-33 การอ่านและตรวจสอบข้อมูลสินค้าก่อนการอัปเดต

2.3.9.3 การอัปเดตข้อมูลของสินค้าตาม ถ้ามีการส่งข้อมูลใหม่เข้ามา product_name, product_category, product_quantity, product_price ข้อมูลนั้นจะถูกเข้ารหัส จากนั้นจะทำการอัปเดต ถ้าไม่มีข้อมูลใหม่เข้ามา ฟิลด์นั้นจะไม่ถูกเปลี่ยนค่า

```
# หากข้อมูลใหม่ถูกระบุ ให้ทำการอัปเดต
if product_name is not None:
    name_encoded = product_name.encode('utf-8')
else:
    name_encoded = product[1]

if product_category is not None:
    category_encoded = product_category.encode('utf-8')
else:
    category_encoded = product[2]

quantity = product_quantity if product_quantity is not None else product[3]
price = product_price if product_price is not None else product[4]
```

ภาพที่ 2-34 การอัปเดตข้อมูลตามที่ระบุ

2.3.9.4 การเขียนข้อมูลสินค้าใหม่ลงในไฟล์ชั่วคราว ใช้ struct.pack เพื่อแปลงข้อมูลสินค้าใหม่เป็นรูปแบบไบนารีและเขียนลงในไฟล์ชั่วคราว ถ้าสินค้าไม่ใช่สินค้าที่จะอัปเดต ให้เขียนข้อมูลสินค้านั้นลงไฟล์ชั่วคราว

```
    # เขียนข้อมูลสินค้าที่อัปเดตลงในไฟล์ชั่วคราว
    data = struct.pack(PRODUCT_FORMAT,
                       product_id,
                       name_encoded,
                       category_encoded,
                       quantity,
                       price)
    temp_file.write(data)
else:
    temp_file.write(data)   # เขียนสินค้าที่ไม่ถูกอัปเดตไปยังไฟล์ชั่วคราว
```

ภาพที่ 2-35 การเขียนข้อมูลสินค้าที่อัปเดตลงไฟล์ชั่วคราว

2.3.9.5 การจัดการไฟล์หลังจากการอัปเดต ถ้าพบสินค้าที่จะอัปเดตจะใช้ os.replace เพื่อเปลี่ยนชื่อไฟล์ชั่วคราวให้เป็นไฟล์หลัก แสดงข้อความยืนยันการอัปเดตสำเร็จ ถ้าไม่พบสินค้าลบไฟล์ชั่วคราวด้วย os.remove และแสดงข้อความว่าหาสินค้าไม่พบ

```
if product_found:
    os.replace(temp_file_path, self.file_path)
    print(f"Product ID {product_id} successfully updated!")
else:
    os.remove(temp_file_path)  # ลบไฟล์ชั่วคราวถ้าไม่พบสินค้า
    print(f"Product ID {product_id} not found!")
```

ภาพที่ 2-36 การจัดการไฟล์หลังการอัปเดต

2.3.9.6 มีการจัดการข้อผิดพลาดกรณีที่ไฟล์ไม่พบหรือเกิดปัญหาอื่น ๆ

```
except FileNotFoundError:
    print("File not found!")
except Exception as e:
    print(f"An error occurred while updating the product: {e}")
```

ภาพที่ 2-37 การจัดการข้อผิดพลาดในฟังก์ชันอัปเดตสินค้า

2.3.10 ฟังก์ชัน export_inventory_report

ฟังก์ชันการสร้างรายทำหน้าที่ส่งออกข้อมูลสินค้าจากไฟล์ไบนารีไปยังไฟล์ข้อความ .txt โดยจัดข้อมูลเป็นหมวดหมู่ เพื่อให้ผู้ใช้สามารถดูข้อมูลทั้งหมดได้ในรูปแบบที่อ่านง่ายขึ้น

2.3.10.1 การเปิดไฟล์และจัดกลุ่มสินค้าตามหมวดหมู่ อ่านข้อมูลจากไฟล์ไบนารีที่มีรายการสินค้าทั้งหมด อ่านสินค้าจากไฟล์ไบนารีทีละ PRODUCT_SIZE คือขนาดของสินค้าแต่ละรายการในไฟล์ แล้วใช้ struct.unpack เพื่อแปลงเป็นทูเพิลโดยแยก ID, ชื่อ, หมวดหมู่, จำนวน, และราคา จากนั้นนำสินค้านั้นใส่ลงในดิกชันนารี categories โดยจัดกลุ่มสินค้าตาม category_value

```
def export_inventory_report(self, output_file_path):
    """ส่งออกข้อมูลไปยังไฟล์ .txt ตามหมวดหมู่"""
    try:
        with open(self.file_path, 'rb') as file:
            categories = {}

            # อ่านข้อมูลสินค้าทั้งหมดและจัดกลุ่มตามหมวดหมู่
            while True:
                data = file.read(PRODUCT_SIZE)
                if not data:
                    break
                product = struct.unpack(PRODUCT_FORMAT, data)

                # แปลงข้อมูลให้ถูกต้อง
                id_value = product[0]
                name_value = product[1].decode('utf-8').strip('\x00')
                category_value = product[2].decode('utf-8').strip('\x00')
                quantity_value = product[3]
                price_value = product[4]

                # จัดกลุ่มสินค้าเข้าตามหมวดหมู่
                if category_value not in categories:
                    categories[category_value] = []
                categories[category_value].append((id_value, name_value, quantity_value, price_value))
```

ภาพที่ 2-38 การจัดหมวดหมู่สินค้าในการสร้างรายงาน

2.3.10.1 สร้างรายงานข้อมูลสินค้า โดยเขียนข้อมูลลงในไฟล์ข้อความ เปิดไฟล์สำหรับเขียนข้อมูลที่พาธ output_file_path คือ inventory_report.txt จากนั้นจะใช้คำสั่งในการเขียนคือ output_file.write() เขียนข้อมูลสินค้าที่จัดกลุ่มแล้วลงในไฟล์ โดยแสดงหมวดหมู่ จำนวนสินค้าในแต่ละหมวด และรายละเอียดสินค้าแต่ละรายการ ตามรูปแบบที่มีการกำหนด และมีการแสดงราคารวมของสินค้าทั้งหมดในแต่ละหมวดหมู่

```python
# เขียนข้อมูลลงไฟล์
with open(output_file_path, 'w') as output_file:
    output_file.write(f"Inventory Report:\n")
    output_file.write(f"Number of categories: {len(categories)}\n")
    output_file.write('-'*70 + '\n\n')

    for category, products in categories.items():
        total_price = sum(product[2] * product[3] for product in products)  # คำนวณราคารวมของหมวดหมู่
        output_file.write(f"Category: {category}\n")
        output_file.write(f"Number of Products: {len(products)}\n")
        headers = ['ID', 'Name', 'Quantity', 'Price(THB)']
        output_file.write('-'*70 + '\n')
        output_file.write(f"{headers[0]:<7} {headers[1]:<30} {headers[2]:<10} {headers[3]:<10} \n")
        output_file.write('-'*70 + '\n')

        for id_value, name_value, quantity_value, price_value in products:
            output_file.write(f"{id_value:03}{'':<3}  {name_value:<30} {quantity_value:<10} {price_value:<10,.2f}\n")
        output_file.write("-"*70 +"\n")
        output_file.write(f"Total Price : {total_price:,.2f} THB\n")  # แสดงราคารวมของหมวดหมู่
        output_file.write("-"*70 +"\n\n") # เพื่อเว้นบรรทัดระหว่างหมวดหมู่
```

ภาพที่ 2-39 การเขียนไฟล์เพื่อสร้างรายงาน

2.3.10.2 ผลลัพธ์การสร้างรายงาน จะมีการแจ้งเตือนว่าสร้างรายงานข้อมูลสินค้าที่ชื่อ inventory_report.txt ได้สำเร็จ และรอให้ผู้ใช้กด Enter เพื่อกลับสู่เมนูหลัก

```python
print(f"\nData successfully exported to '{output_file_path}' ! \n")

input("Press Enter to return to the main menu...")# รอให้ผู้ใช้กด Enter ก่อนที่จะกลับไปที่เมนู
```

ภาพที่ 2-40 ผลลัพธ์การสร้างรายงาน

2.3.10.3 มีการจัดการข้อผิดพลาดกรณีที่ไฟล์ไม่พบหรือเกิดปัญหาอื่น ๆ

```python
except FileNotFoundError:
    print("File not found!")
except Exception as e:
    print(f"An error occurred during data export: {e}")
```

ภาพที่ 2-41 การจัดการข้อผิดพลาดในฟังก์ชั่นสร้างรายงาน

2.4 ตัวอย่างการทำงานของโปรแกรมระบบจัดการสต็อกสินค้า

2.4.1 เริ่มต้นโปรแกรม

เมื่อผู้ใช้รันโปรแกรม Python ผ่านเทอร์มินอล หรือ IDE อย่าง Visual Studio Code โปรแกรมจะแสดงเมนูการทำงานหลักให้ผู้ใช้เลือกดังนี้

```
Menu
1. Display Product
2. Add Product
3. Update Product
4. Delete Product
5. Display Category Product
6. Export Inventory Report
7. Exit
Enter your menu choice:
```

ภาพที่ 2-42 แสดงเมนูเริ่มต้นโปรแกรม

2.4.2 เมนูที่ 1 Display Product

เมื่อมีการเลือกเมนูที่ 1 และกด Enter โปรแกรมจะแสดงรายการข้อมูลสินค้าที่มีอยู่ในระบบทั้งหมด

```
Menu
1. Display Product
2. Add Product
3. Update Product
4. Delete Product
5. Display Category Product
6. Export Inventory Report
7. Exit
Enter your menu choice: 1
Display product list:
ID    Name                          Category                  Quantity    Price(THB)
1     Mouse                         Electronics               20          120.00
2     Keyboard                      Electronics               10          500.00
3     Lego                          Toys                      10          200.00
4     Bear                          Other                     20          100.00
5     Borad Game                    Toys                      30          250.00
6     Camera                        Electronics               10          1,500.00
20    Tt                            Other                     20          100.00
Press Enter to return to the main menu...
```

ภาพที่ 2-43 เมนูที่ 1: Display Product

2.4.3 เมนูที่ 2 Add Product

เมื่อมีการเลือกเมนูที่ 2 และกด Enter โปรแกรมจะให้ผู้ใช้กรอกรายละเอียดสินค้าใหม่ เช่น รหัสสินค้า, ชื่อสินค้า, ประเภท, จำนวน และราคา

2.4.3.1 เมื่อเพิ่มสินค้าใหม่ลงในระบบ โดยโปรแกรมจะตรวจสอบรหัสสินค้าว่าซ้ำหรือไม่ ถ้ารหัสซ้ำจะให้กรอกใหม่

```
Menu
1. Display Product
2. Add Product
3. Update Product
4. Delete Product
5. Display Category Product
6. Export Inventory Report
7. Exit
Enter your menu choice: 2
Add Product
Enter Product ID: 14
Product ID 14 already exists. Please enter a different ID.
Enter Product ID:
```

ภาพที่ 2-44 ตรวจสอบรหัสสินค้า

2.4.3.2 เมื่อโปรแกรมตรวจสอบว่ารหัสสินค้าไม่ซ้ำจะให้ผู้ใช้กรอกรายละเอียดสินค้าใหม่ คือ ชื่อสินค้า, ประเภท, จำนวน และราคา

```
Menu
1. Display Product
2. Add Product
3. Update Product
4. Delete Product
5. Display Category Product
6. Export Inventory Report
7. Exit
Enter your menu choice: 2
Add Product
Enter Product ID: 14
Enter Product Name: lego
Enter Product Category: toys
Enter Product Quantity: 10
Enter Product Price: 8
Product added successfully!
Press Enter to return to the main menu...
```

ภาพที่ 2-45 การเพิ่มสินค้าลงในระบบ

2.4.3.3 หากผู้ใช้มีการพิมพ์ข้อมูลจำนวนสินค้าหรือราคาผิด เช่น ต้องใส่ตัวเลขแต่ใส่เป็นตัวอักษร ระบบจะแจ้งให้ทำการกรอกข้อมูลใหม่ให้ถูกต้อง

```
Enter your menu choice: 2
Add Product
Enter Product ID: 99
Enter Product Name: IPhone XR
Enter Product Category: electronics
Enter Product Quantity: 2
Enter Product Price: tree
Invalid input. Please enter a valid number for Quantity and Price.
Enter Product Name:
```

ภาพที่ 2-46 การตรวจสอบการใส่ข้อมูลจำนวนสินค้าผิดพลาด

2.4.4 เมนูที่ 3 Update Product

เมื่อมีการเลือกเมนูที่ 3 และกด Enter เป็นเมนูที่จะอัปเดตสินค้าในระบบ จากนั้นให้ผู้ใช้ระบุรหัสสินค้าที่ต้องการแก้ไข และให้กรอกข้อมูลใหม่ที่ต้องการอัปเดต เช่น ชื่อสินค้า หมวดหมู่ จำนวน หรือราคา

2.4.4.1 โปรแกรมจะแสดงสินค้าทั้งหมดเพื่อให้ผู้ใช้เห็นรายการปัจจุบัน

```
Enter your menu choice: 3
Update Product
Display product list:
ID    Name                    Category                Quantity    Price(THB)
1     Mouse                   Electronics             20          120.00
2     Keyboard                Electronics             10          500.00
3     Lego                    Toys                    10          200.00
4     Bear                    Other                   20          100.00
5     Borad Game              Toys                    30          250.00
6     Camera                  Electronics             10          1,500.00
14    Lego                    Toys                    10          8.00
19    Bol                     Toys                    20          500.00
20    Tt                      Other                   20          100.00
99    Itim                    Toys                    10          50.00
Press Enter to return to the main menu...
```

ภาพที่ 2-47 แสดงรายการสินค้าก่อนอัปเดตสินค้า

2.4.4.2 ตรวจสอบว่ามีรหัสสินค้าที่ผู้ใช้ต้องการอัปเดตนั้นมีหรือไม่

```
Enter Product ID to update: 100
Product ID 100 does not exist. Please enter a valid ID.
Enter Product ID to update:
```

ภาพที่ 2-48 ตรวจสอบรหัสสินค้า

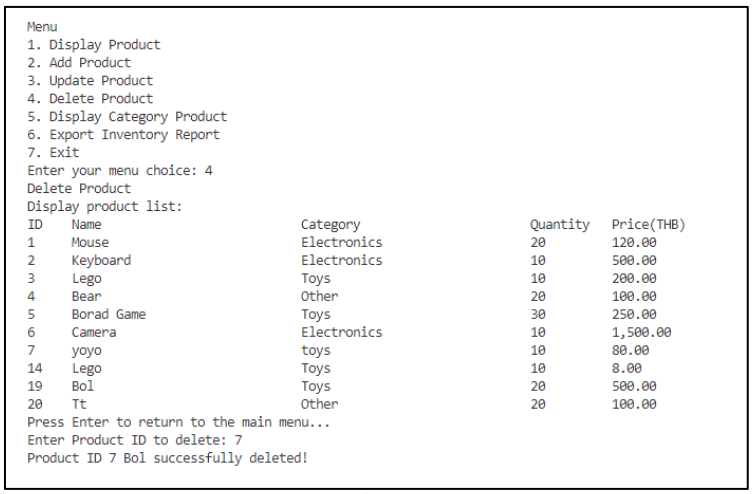
2.4.4.3 การอัปเดตสินค้าในระบบ ผู้ใช้ระบุรหัสสินค้าในระบบที่ต้องการแก้ไข จากนั้นให้กรอกข้อมูลใหม่ที่ต้องการอัปเดต เช่น ชื่อสินค้า หมวดหมู่ จำนวน และราคา ถ้าหากไม่ต้องการแก้ไขฟิลด์นั้น ให้กด Enter แล้วไปแก้ไขฟิลด์ถัดไป

```
14    Lego                    Toys                    10          8.00
19    Bol                     Toys                    20          500.00
20    Tt                      Other                   20          100.00
99    Itim                    Toys                    10          50.00
Press Enter to return to the main menu...
Enter Product ID to update: 100
Product ID 100 does not exist. Please enter a valid ID.
Enter Product ID to update: 20
Enter new Product Name (leave blank to keep current):YoYo
Enter new Product Category (leave blank to keep current):
Enter new Product Quantity (leave blank to keep current):
Enter new Product Price (leave blank to keep current): 80
Product ID 20 successfully updated!
```

ภาพที่ 2-49 การอัปเดตสินค้า

2.4.5 เมนูที่ 4 Delete Product

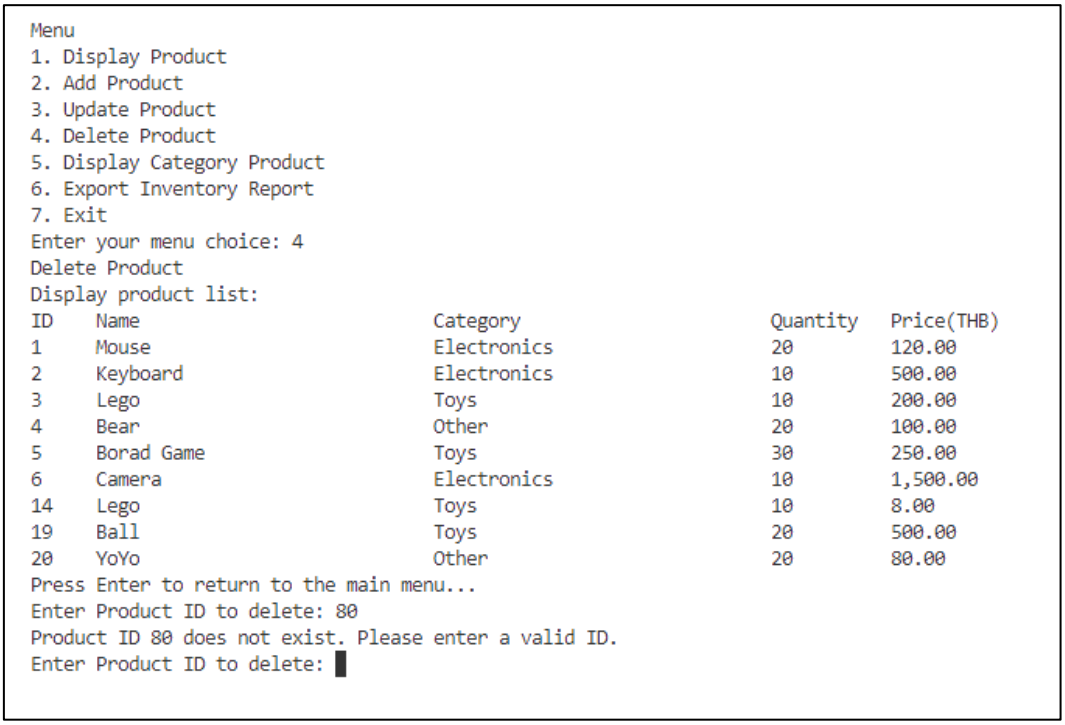
2.4.5.1 เมื่อมีการเลือกเมนูที่ 4 และกด Enter โปรแกรมจะแสดงสินค้าทั้งหมดที่มีในระบบ จากนั้นให้ผู้ใช้ระบุรหัสสินค้าที่ต้องการลบ เมื่อยืนยันรหัสถูกต้อง ระบบจะลบสินค้ารายนั้นออก และแจ้งเตือนว่าลบสำเร็จ



```
Menu
1. Display Product
2. Add Product
3. Update Product
4. Delete Product
5. Display Category Product
6. Export Inventory Report
7. Exit
Enter your menu choice: 4
Delete Product
Display product list:
ID    Name                Category            Quantity   Price(THB)
1     Mouse               Electronics         20         120.00
2     Keyboard            Electronics         10         500.00
3     Lego                Toys                10         200.00
4     Bear                Other               20         100.00
5     Borad Game          Toys                30         250.00
6     Camera              Electronics         10         1,500.00
7     yoyo                toys                10         80.00
14    Lego                Toys                10         8.00
19    Bol                 Toys                20         500.00
20    Tt                  Other               20         100.00
Press Enter to return to the main menu...
Enter Product ID to delete: 7
Product ID 7 Bol successfully deleted!
```

ภาพที่ 2-50 เมนูที่ 4 Delete Product

2.4.5.2 หากรหัสสินค้าที่ผู้ใช้ต้องการจะลบไม่มีในระบบ โปรแกรมจะแจ้งเตือนว่าไม่พบและให้กรอกรหัสสินค้าใหม่อีกครั้ง



```
Menu
1. Display Product
2. Add Product
3. Update Product
4. Delete Product
5. Display Category Product
6. Export Inventory Report
7. Exit
Enter your menu choice: 4
Delete Product
Display product list:
ID    Name                Category            Quantity   Price(THB)
1     Mouse               Electronics         20         120.00
2     Keyboard            Electronics         10         500.00
3     Lego                Toys                10         200.00
4     Bear                Other               20         100.00
5     Borad Game          Toys                30         250.00
6     Camera              Electronics         10         1,500.00
14    Lego                Toys                10         8.00
19    Ball                Toys                20         500.00
20    YoYo                Other               20         80.00
Press Enter to return to the main menu...
Enter Product ID to delete: 80
Product ID 80 does not exist. Please enter a valid ID.
Enter Product ID to delete:
```

ภาพที่ 2-51 การตรวจสอบรหัสสินค้าก่อนลบ

2.4.6 เมนูที่ 5 Display Category Product

เมื่อมีการเลือกเมนูที่ 5 และกด Enter โปรแกรมจะขอให้ผู้ใช้กรอกชื่อหมวดหมู่ที่ต้องการดูสินค้าภายในหมวดหมู่นั้น เช่น Electronics แล้วจะแสดงรายการสินค้าที่ตรงกับหมวดหมู่ที่ระบุถ้าไม่มีสินค้าที่ตรงกับหมวดหมู่ ระบบจะแจ้งว่า "No products found in this category."

```
Menu
1. Display Product
2. Add Product
3. Update Product
4. Delete Product
5. Display Category Product
6. Export Inventory Report
7. Exit
Enter your menu choice: 5

Inventory Report:
Number of categories: 3

Category : Electronics
Number of Products: 3
--------------------------------------------------------------------------
ID      Name                            Quantity   Price(THB)
--------------------------------------------------------------------------
002     Keyboard                        10         500.00
001     Mouse                           20         120.00
006     Camera                          10         1,500.00
--------------------------------------------------------------------------
Total Price : 22,400.00 THB
--------------------------------------------------------------------------

Category : Toys
Number of Products: 4
--------------------------------------------------------------------------
ID      Name                            Quantity   Price(THB)
--------------------------------------------------------------------------
003     Lego                            10         200.00
005     Borad Game                      30         250.00
014     Lego                            10         8.00
019     Bol                             20         500.00
--------------------------------------------------------------------------
Total Price : 19,580.00 THB
--------------------------------------------------------------------------

Category : Other
Number of Products: 2
--------------------------------------------------------------------------
ID      Name                            Quantity   Price(THB)
--------------------------------------------------------------------------
004     Bear                            20         100.00
020     Tt                              20         100.00
--------------------------------------------------------------------------
Total Price : 4,000.00 THB
--------------------------------------------------------------------------
```

ภาพที่ 2-52 เมนูที่ 5 Display Category Product

2.4.7 เมนูที่ 6 Export Inventory Report

2.4.7.1 เมื่อมีการเลือกเมนูที่ 6 และกด Enter โปรแกรมจะสร้างรายงานข้อมูลสินค้าทั้งหมดในรูปแบบไฟล์ข้อความ inventory_report.txt ซึ่งประกอบด้วยรายการสินค้าในระบบ

```
Menu
1. Display Product
2. Add Product
3. Update Product
4. Delete Product
5. Display Category Product
6. Export Inventory Report
7. Exit
Enter your menu choice: 6

Data successfully exported to 'inventory_report.txt' !

Press Enter to return to the main menu...
```

ภาพที่ 2-53 เมนูที่ 6 Export Inventory Report

2.4.7.2 ตัวอย่างรายงานไฟล์ inventory_report.txt แสดงรายงานข้อมูลสินค้าทั้งหมด

inventory_report.txt

```
Inventory Report:
Number of categories: 3
------------------------------------------------------------------------

Category: Electronics
Number of Products: 3
------------------------------------------------------------------------
ID      Name                            Quantity   Price(THB)
------------------------------------------------------------------------
002     Keyboard                        10         500.00
001     Mouse                           20         120.00
006     Camera                          10         1,500.00
------------------------------------------------------------------------
Total Price : 22,400.00 THB
------------------------------------------------------------------------

Category: Toys
Number of Products: 4
------------------------------------------------------------------------
ID      Name                            Quantity   Price(THB)
------------------------------------------------------------------------
003     Lego                            10         200.00
005     Borad Game                      30         250.00
014     Lego                            10         8.00
019     Bol                             20         500.00
------------------------------------------------------------------------
Total Price : 19,580.00 THB
------------------------------------------------------------------------

Category: Other
Number of Products: 2
------------------------------------------------------------------------
ID      Name                            Quantity   Price(THB)
------------------------------------------------------------------------
004     Bear                            20         100.00
020     Tt                              20         100.00
------------------------------------------------------------------------
Total Price : 4,000.00 THB
------------------------------------------------------------------------

```

ภาพที่ 2-54 รายงานไฟล์ inventory_report.txt แสดงรายงานข้อมูลสินค้าทั้งหมด

2.4.8 เมนูที่ 7 Exit

เมื่อมีการเลือกเมนูที่ 7 และกด Enter โปรแกรมจะหยุดการทำงาน

```
Menu
1. Display Product
2. Add Product
3. Update Product
4. Delete Product
5. Display Category Product
6. Export Inventory Report
7. Exit
Enter your menu choice: 7
Exiting the program.
```

ภาพที่ 2-55 การหยุดการทำงานของโปรแกรม