ระบบการจัดการร้านอาหารฟาสต์ฟู้ด
Fast Food Management System

นายธีรวัฒน์ โพธิ์สาวัง	รหัส 6806022510351 Sec3
นางสาวพีรยา จันแปลง	รหัส 6806022510408 Sec3
นายเดชาวัต เนติชัย	รหัส 6806022511030 Sec3
นายพีรกานต์ ไกรพินิจ	รหัส 6806022511129 Sec3

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ปีการศึกษา 2568

คำนำ

การจัดทำโครงงาน “ระบบการจัดการร้านอาหารฟาสต์ฟู้ด” นี้เป็นส่วนหนึ่งของวิชา Computer Programming ของหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ เพื่อให้นักศึกษาได้นำความรู้ที่เรียนมาทั้งหมดมาประยุกต์ใช้ในการพัฒนาโปรแกรมที่สามารถทำงานได้จริง โดยเน้นการออกแบบและเขียนโปรแกรมด้วยภาษา Python ซึ่งเป็นภาษาที่เรียนมาในวิชา Computer Programming โดยโครงงานนี้จะช่วยการคิดวิเคราะห์และการแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมในการประกอบอาชีพด้านวิศวกรรมสารสนเทศและเครือข่ายในอนาคต

คณะผู้จัดทำหวังว่ารายงานฉบับนี้จะเป็นประโยชน์กับผู้อ่าน หรือนักเรียน นักศึกษา ที่กำลังหาข้อมูลเรื่องนี้อยู่ หากมีข้อแนะนำหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้และขออภัยมา ณ ที่นี้ด้วย

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 วัตถุประสงค์ของโครงงาน

1.1.1 เพื่อพัฒนาระบบการจัดการอาหารฟาสต์ฟู้ดได้อย่างมีประสิทธิภาพ

1.1.2 เพื่อพัฒนาทักษะการเขียนโปรแกรมด้วย Python

1.1.3 เพื่อเรียนรู้วิธีการจัดการข้อมูลและไฟล์

1.1.4 เพื่อเรียนรู้การทำงานเป็นทีม

1.2 ขอบเขตของโครงงาน

1.2.1 ระบบการจัดการร้านอาหารฟาสต์ฟู้ดมีฟังก์ชั่นพื้นฐานในการใช้งานทั้งหมด 17 ฟังก์ชั่น เช่น

1. การจัดการสินค้า
2. เพิ่มสินค้า
3. อัพเดทสินค้า
4. ลบสินค้า
5. ดูรายการสินค้า
6. การจัดการราคา
7. เพิ่มราคาสินค้า
8. อัพเดทราคาสินค้า
9. ลบราคาสินค้า
10. ดูราคาสินค้า
11. การจัดการโปรโมชั่น
12. เพิ่มโปรโมชั่นสินค้า
13. อัพเดทโปรโมชั่นสินค้า
14. ลบโปรโมชั่นสินค้า
15. ดูโปรโมชั่นสินค้า
16. สร้าง report สำหรับแสดงการรายงานสินค้าทั้งหมด
17. เมนูออกจากหน้าการทำงานปัจจุบัน

1.2.2 ระบบการจัดการร้านอาหารฟาสต์ฟู้ดประกอบด้วย 11 ไฟล์ ได้แก่

1. log.txt
2. logger.py
3. main.py
4. price_manager.py
5. prices.dat
6. product_manager.py
7. products.dat
8. promotion_manager.py
9. promotions.dat
10. report.py
11. report.txt

1.2.3 ระบบการจัดการร้านอาหารฟาสต์ฟู้ดมีการจัดเก็บข้อมูลบันทึกรายการอาหาร รหัสสินค้า ชื่อสินค้า ไซส์ของสินค้า โปรโมชั่นสินค้า ราคา จำนวนสินค้า และสถานะของสินค้าอยู่ใน Report.txt

1.2.4 ระบบการจัดการร้านอาหารฟาสต์ฟู้ดจะมีเมนูเพื่อให้ผู้ใช้งานสามารถเลือกดำเนินการได้

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 พัฒนาระบบการจัดการร้านอาหารฟาสต์ฟู้ดได้อย่างมีประสิทธิภาพ

1.3.2 พัฒนาทักษะการเขียนโปรแกรม

1.3.3 เรียนรู้การจัดการข้อมูลและไฟล์

1.3.4 เรียนรู้การทำงานเป็นทีม

1.4 เครื่องมือที่คาดว่าจะต้องใช้

1.4.1 โปรแกรม Visual Studio Code

บทที่ 2

ระบบการจัดการร้านอาหารฟาสต์ฟู้ด

2.1 แฟ้มข้อมูลสินค้า Products.dat

แฟ้มข้อมูลประกอบด้วย 3 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญดังนี้

ฟิลด์	ชนิด	ขนาด(Byte)	คำอธิบาย
Pro_id	String	10	รหัสสินค้า
Pro_name	String	30	ชื่อสินค้า
Promotion_id	Int	4	รหัสโปรโมชั่น

ตารางที่ 2.1 แฟ้มข้อมูลสินค้า

2.1.1 Pro_id รหัสสินค้า

Pro_id เป็นรหัสสินค้าที่ใช้ในการระบุสินค้าได้อย่างชัดเจนและไม่ซ้ำกันฟิลด์นี้ถูกสร้างขึ้นโดยระบบในรูปแบบของตัวอักษร (String) เช่น 1001, 1002, 1003 เป็นต้น การมีรหัสสินค้าที่เป็นเอกลักษณ์นี้เป็นสิ่งจำเป็นในการระบุสินค้า และหลีกเลี่ยงความสับสนระหว่างรายการสินค้าหลายอย่าง และช่วยให้สามารถค้นหาสินค้าได้อย่างรวดเร็ว

2.1.2 Pro_name ชื่อสินค้า

Pro_name ชื่อสินค้า ซึ่งฟิลด์นี้จะแสดงข้อมูลชื่อเมนูอาหารแต่ละรายการของร้านอาหาร ฟิลด์นี้เป็นประเภทข้อมูลข้อความ (string) ตัวอย่างเช่น "เบอร์เกอร์หมูจนาดพิเศษ" หรือ "เบอร์เกอร์ปลารมควัน" การมีชื่ออาหารในระบบมีความสำคัญอย่างยิ่ง เพราะใช้ในการเรียกดูข้อมูล, รับออเดอร์จากลูกค้า, และจัดการข้อมูลต่างๆ ที่เกี่ยวข้อง อาหารแต่ละเมนูจะมีชื่อตามที่ระบุไว้ในระบบ และระบบจะใช้ชื่อดังกล่าวในการค้นหา, รับออเดอร์, และแสดงผลข้อมูลที่เกี่ยวกับเมนูนั้น ๆ

2.1.3 Promotion_id รหัสโปรโมชั่น

รหัสตัวเลขที่ใช้ระบุโปรโมชั่นแต่ละอย่างของร้านอาหาร ซึ่งฟิลด์นี้จะช่วยให้ระบบรู้ว่าลูกค้ากำลังใช้โปรโมชั่นใด หรือเมนูไหนกำลังจัดโปรโมชั่นอยู่ เมื่อกำหนดชนิดข้อมูลเป็น int รหัสโปรโมชั่นจะเป็นตัวเลขจำนวนเต็มเท่านั้น เช่น 101 (ซื้อหนึ่งแถมหนึ่ง), 10 (ลด 10 %) การมีรหัสโปรโมชั่นในระบบยังคงมีความสำคัญอย่างยิ่งเหมือนเดิม คือใช้ในการคำนวณส่วนลด, เชื่อมโยงเมนูอาหาร

2.2 แฟ้มข้อมูลราคา prices.dat

แฟ้มข้อมูลราคาประกอบด้วย 5 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีดังนี้

ฟิลด์	ชนิด	ขนาด(Byte)	คำอธิบาย	ตัวอย่าง
Pro_id	String	10	รหัสสินค้า	P01
Pro_size	String	10	ขนาดสินค้า	เบอร์เกอร์ไซส์ใหญ่
Pro_price	Float	4	ราคาสินค้า	150
Pro_stock	Int	4	จำนวนสินค้าในคลัง	12
Sale_status	Byte	1	สถานะสินค้า 0 : ไม่พร้อมจำหน่าย 1 : พร้อมจำหน่าย	พร้อมจำหน่าย

ตารางที่ 2.2 แฟ้มข้อมูลราคา

2.2.1 Pro_id รหัสสินค้า

Pro_id เป็นรหัสสินค้าที่ใช้ในการระบุสินค้าได้อย่างชัดเจนและไม่ซ้ำกันฟิลด์นี้ถูกสร้างขึ้นโดยระบบในรูปแบบของตัวอักษร (String) เช่น 1001, 1002, 1003 เป็นต้น การมีรหัสสินค้าที่เป็นเอกลักษณ์นี้เป็นสิ่งจำเป็นในการระบุสินค้า และหลีกเลี่ยงความสับสนระหว่างรายการสินค้าหลายอย่าง และช่วยให้สามารถค้นหาสินค้าได้อย่างรวดเร็ว

2.2.2 Pro_size ขนาดสินค้า

Pro_size คือ ฟิลด์ที่ใช้ระบุ ขนาดหรือตัวเลือกต่างๆ ของเมนูอาหารเดียวกัน เพื่อแยกความแตกต่างของสินค้า เช่น เบอร์เกอร์ขนาดเล็ก ขนาดกลาง หรือขนาดใหญ่ ฟิลด์นี้มักเป็นประเภทข้อมูลข้อความ (string) เพื่อให้เข้าใจง่าย ตัวอย่างเช่น "S" (สำหรับขนาดเล็ก), "M" (สำหรับขนาดกลาง), หรือ "L" (สำหรับขนาดใหญ่ การมีฟิลด์ Pro_size มีความสำคัญอย่างยิ่ง เพราะช่วยให้ร้านสามารถตั้งราคาขายที่แตกต่างกันตามขนาดและปริมาณของวัตถุดิบได้

2.2.3 Pro_price ราคาสินค้า

Pro_price คือ ฟิลด์ที่ใช้สำหรับเก็บข้อมูล ราคาขาย ของเมนูอาหารเดียวกัน เพื่อกำหนดราคาที่แตกต่างกันตามขนาด เช่น ราคาเบอร์เกอร์ขนาดเล็ก ขนาดกลาง หรือขนาดใหญ่ ฟิลด์นี้จะเป็นประเภทข้อมูลตัวเลขจำนวนเต็ม (int) เช่น ขนาด "S" อาจมีราคา 89 บาท, ขนาด "M" อาจมีราคา 129 บาท, หรือขนาด "L" อาจมีราคา 159 บาท

2.2.4 Pro_stock จำนวนสินค้าในคลัง

Pro_stock คือ ฟิลด์ที่ใช้สำหรับเก็บข้อมูล จำนวนสินค้าที่เหลืออยู่ในคลัง ของเมนูอาหารนั้น ๆ เพื่อใช้ตรวจสอบว่าสามารถขายต่อได้หรือไม่ ฟิลด์นี้จะเป็นประเภทข้อมูลตัวเลขจำนวนเต็ม (int) เช่น ถ้ามีค่าเป็น 12 หมายถึงยังมีสินค้าเหลืออยู่ 12 ชิ้นในสต็อก หรือถ้ามีค่าเป็น 0 หมายถึงสินค้าหมด ไม่สามารถขายได้

2.2.5 Sale_status สถานะสินค้า

Sale_status คือ ฟิลด์ที่ใช้สำหรับเก็บข้อมูล สถานะการขายของสินค้า โดยใช้ค่าตัวเลขแบบ Byte เพื่อระบุว่าสินค้านั้นพร้อมขายหรือไม่ ฟิลด์นี้จะเป็นข้อมูลตัวเลขขนาดเล็กที่กำหนดค่าเพียง 2 แบบ คือ ค่า 0 หมายถึง สินค้ายังไม่พร้อมขาย ค่า 1 หมายถึง สินค้าพร้อมขาย ตัวอย่างเช่น ถ้า Sale_status มีค่าเป็น 1 แสดงว่าสินค้าพร้อมให้ลูกค้าซื้อได้ทันที แต่ถ้าเป็น 0 แสดงว่ายังไม่สามารถขายได้ในขณะนั้น

2.3 แฟ้มข้อมูลโปรโมชั่น promotions.dat

แฟ้มข้อมูลโปรโมชั่นประกอบด้วย 2 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญดังนี้

ฟิลด์	ชนิด	ขนาด(Byte)	คำอธิบาย	ตัวอย่าง
Promotion_id	Int	4	รหัสโปรโมชั่น	PM10
Promotion_name	sring	30	ชื่อโปรโมชั่น	ลดราคา10%

ตารางที่ 2.3 แฟ้มข้อมูลโปรโมชั่น

2.3.1 Promotion_id รหัสโปรโมชั่น

Promotion_id คือ ฟิลด์ที่ใช้สำหรับเก็บข้อมูล รหัสโปรโมชั่น เพื่อระบุและแยกโปรโมชั่นแต่ละรายการออกจากกัน ฟิลด์นี้เป็นข้อมูลประเภทจำนวนเต็ม (Int) ขนาด 4 ไบต์ โดยค่าที่เก็บจะถูกใช้เป็นรหัสอ้างอิง เช่น "PM10" อาจหมายถึงโปรโมชั่นลดราคา 10%

2.3.2 Promotion_name ชื่อโปรโมชั่น

Promotion_name คือ ฟิลด์ที่ใช้สำหรับเก็บข้อมูล ชื่อโปรโมชั่น ซึ่งเป็นข้อความอธิบายรายละเอียดของโปรโมชั่นนั้น ๆ ฟิลด์นี้เป็นข้อมูลประเภทข้อความ (String) ขนาด 30 ไบต์ เช่น "ลดราคา 10%" หรือ "ซื้อ 1 แถม 1" เพื่อให้ผู้ใช้งานระบบสามารถเข้าใจและเลือกใช้งานโปรโมชั่นได้ง่าย

2.4 ไฟล์ report.txt

ไฟล์ report.txt ในระบบการจัดการร้านอาหารฟาสต์ฟู้ดประกอบไปด้วย

```
Burger Shop Report
Generated: 2025-10-01 14:22:02

Total Products: 2

Total Price Records: 6

+------------+---------------------------+------------+---------------+---------+-------+-----------+
| Product ID | Product Name              | Size       | Promotion     | Price   | Stock | Status    |
+------------+---------------------------+------------+---------------+---------+-------+-----------+
| F001       | fish burger               | S          | -             | 50.00   | 10    | Ready     |
| F001       | fish burger               | M          | -             | 90.00   | 10    | Ready     |
| F001       | fish burger               | XL         | -             | 150.00  | 5     | Ready     |
| P001       | pork burger               | S          | ลด10%         | 49.00   | 20    | Ready     |
| P001       | pork burger               | M          | ลด10%         | 69.00   | 10    | Sold out  |
| P001       | pork burger               | XL         | ลด10%         | 99.00   | 2     | Ready     |
+------------+---------------------------+------------+---------------+---------+-------+-----------+

Last 10 Log Events:
[2025-10-01 14:13:25] USER - Update Price Product ID F001 Size M:  -> OK
[2025-10-01 14:13:55] USER - Update Price Product ID P001 Size M:  -> OK
[2025-10-01 14:14:13] USER - Update Price Product ID P001 Size M:  -> OK
[2025-10-01 14:14:53] USER - Update Price Product ID P001 Size M:  -> OK
[2025-10-01 14:15:02] SYSTEM - Generate Report:  -> OK
[2025-10-01 14:17:11] SYSTEM - Generate Report:  -> OK
[2025-10-01 14:19:00] SYSTEM - Generate Report:  -> OK
[2025-10-01 14:20:49] SYSTEM - Generate Report:  -> OK
[2025-10-01 14:21:17] SYSTEM - Generate Report:  -> OK
[2025-10-01 14:21:46] SYSTEM - Generate Report:  -> OK
```

ภาพที่ 2.1 ไฟล์ report.txt

2.4.1 header_text ส่วนหัวรายงาน

เป็นฟิลด์ข้อความชื่อรายงาน (string 100 bytes) เพื่อระบุประเภทของรายงานให้ชัดเจน เช่น รายงานนี้เกี่ยวกับ “ระบบจัดการสินค้าร้านเบอร์เกอร์”

2.4.2 generated_at วันและเวลาที่สร้างรายงาน

เป็นข้อมูลวันที่และเวลาที่ไฟล์รายงานถูกสร้างขึ้น (รูปแบบ YYYY-MM-DD HH:MM:SS) เพื่อให้ทราบช่วงเวลาการออกรายงาน

2.4.3 summary_section สรุปข้อมูลเบื้องต้น

แสดงจำนวนสินค้าทั้งหมดในระบบ (2 รายการ ได้แก่ fish burger และ pork burger) และจำนวนเรกคอร์ดข้อมูลราคาทั้งหมด (6 รายการ คือ 3 ขนาดต่อสินค้า)

2.4.4 product_table_header หัวตารางสินค้า

เป็นส่วนหัวของตาราง แสดงชื่อคอลัมน์ที่ใช้สำหรับข้อมูลสินค้าแต่ละรายการ เช่น รหัสสินค้า, ชื่อ, ขนาด, โปรโมชั่น, ราคา, จำนวนคงเหลือ และสถานะ

2.4.5 product_records ข้อมูลสินค้าแต่ละรายการ

เป็นข้อมูลสินค้าของร้านในรูปแบบตาราง โดยแต่ละแถวเป็นหนึ่งเรคคอร์ดของสินค้า เช่น fish burger และ pork burger ที่มีหลายขนาด (S, M, XL) พร้อมราคาจำหน่าย โปรโมชั่น จำนวนคงเหลือ และสถานะ เช่น “Ready” หรือ “Sold out” ตัวอย่างเช่น fish burger ไม่มีโปรโมชั่น ราคาปกติ ส่วน pork burger มีโปรโมชั่นลด 10% และขนาด M ขายหมดแล้ว

2.4.6 log_section บันทึกเหตุการณ์ล่าสุด

เป็นข้อมูลประวัติการทำงานของผู้ใช้และระบบ โดยแต่ละบรรทัดขึ้นต้นด้วยวันที่และเวลา เช่น “[2025-10-01 14:13:25] USER - Update Price Product ID F001 Size M: -> OK” แสดงว่าผู้ใช้ได้ทำการอัปเดตราคาสินค้าเรียบร้อย และมีบันทึกการสร้างรายงานโดยระบบ เช่น “SYSTEM - Generate Report: -> OK” รวมทั้งหมด 10 รายการล่าสุด เพื่อใช้ตรวจสอบการเปลี่ยนแปลงย้อนหลัง

บทที่ 3
การใช้งานระบบการจัดการร้านอาหารฟาสต์ฟู้ด

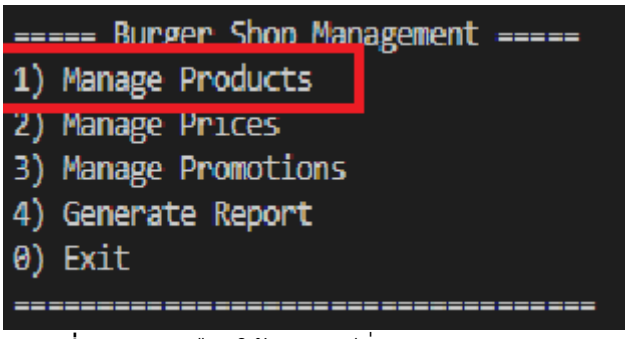
ระบบจัดการร้านอาหารฟาสต์ฟู้ดนี้ช่วยให้ผู้ใช้สามารถจัดการข้อมูลสินค้า ราคาสินค้า และโปรโมชั่นได้ง่าย ๆ ผ่านเมนูโปรแกรม โดยสามารถเพิ่ม แก้ไข หรือลบข้อมูลได้อย่างรวดเร็ว ระบบจะเก็บข้อมูลทั้งหมดลงในไฟล์ เพื่อให้เรียกดูและแก้ไขได้ตลอดเวลา

เมนูหลักของโปรแกรมแบ่งเป็นส่วนจัดการสินค้า การตั้งราคาสินค้า การสร้างโปรโมชั่น และการดูรายงาน โดยแต่ละส่วนช่วยให้การทำงานของร้านเป็นระเบียบและสะดวกมากขึ้นนอกจากนี้ ระบบยังมีฟังก์ชันบันทึกการทำงาน (log) เพื่อเก็บประวัติการใช้งาน เช่น การเพิ่มข้อมูลหรือการสร้างรายงาน ทำให้สามารถตรวจสอบย้อนหลังได้ง่ายด้วยระบบนี้ ร้านอาหารฟาสต์ฟู้ดจะบริหารจัดการข้อมูลได้อย่างรวดเร็ว ถูกต้อง และลดความผิดพลาดในการทำงานประจำวัน

สำหรับผู้ใช้งานโปรแกรม

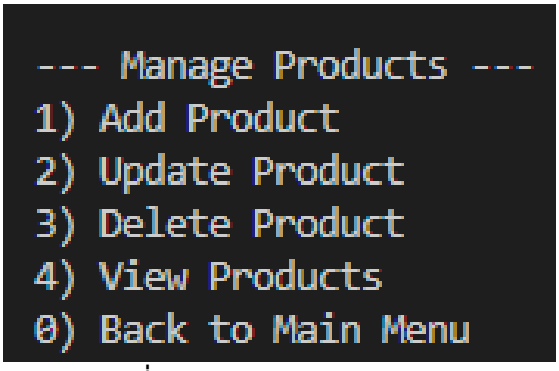
3.1 การใช้งานโปรแกรมระบบจัดการร้านอาหารฟาสต์ฟู้ด

3.1.1 กรอกหมายเลข 1 ภายในกรอบสีแดงเพื่อเรียกฟังก์ชัน Manage Products เพิ่มข้อมูลที่ประกอบไปด้วย Manage Produccts Manage Pricces Manage promotions Generate Report และ Exit



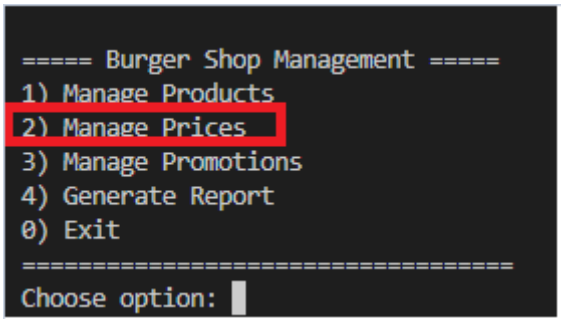
ภาพที่ 3.1 การเลือกใช้งานฟังก์ชั่น Manage Products

3.1.2 เมื่อเมนูฟังก์ชั่น Manage Products ขึ้นมาแล้วจากนั้นก็สามารถระบุเมนูที่ต้องการเลือกได้



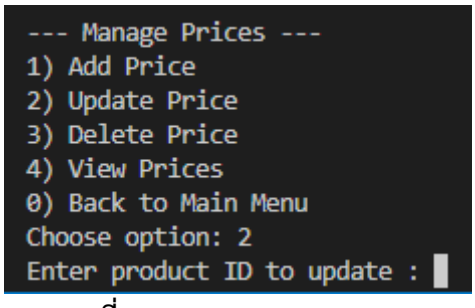
ภาพที่ 3.2 เมนูของ Add Products

3.1.3 กรอกหมายเลข 2 ภายในกรอบสีแดงเพื่อเรียกฟังก์ชัน Manage Pricces เพิ่มข้อมูลที่ประกอบไปด้วย Add price Update Price Delete Price View Price Back to Main menu



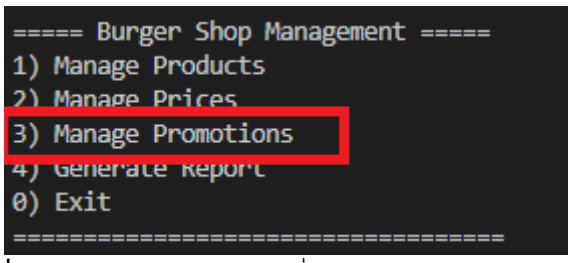
ภาพที่ 3.3 การเลือกใช้งานฟังก์ชั่นของ Manage Price

3.1.4 เมื่อเมนูฟังก์ชั่น Manage Price ขึ้นมาแล้วก็สามารถระบุเมนูที่ต้องการเลือกได้



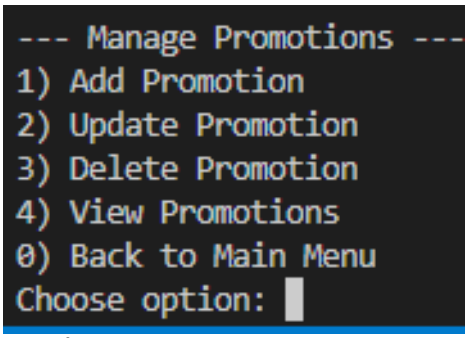
ภาพที่ 3.4 เมนูขอ Manage Price

3.1.5 กรอกหมายเลขที่ 3 ภายในกรอบสีแดงเพื่อเรียกฟังก์ชั่น Mange Promotions เพิ่มข้อมูลที่ประกอยไปด้วย Add Promotion Update Promotion Delete Promotion View Promotions Bavk to Main Menu



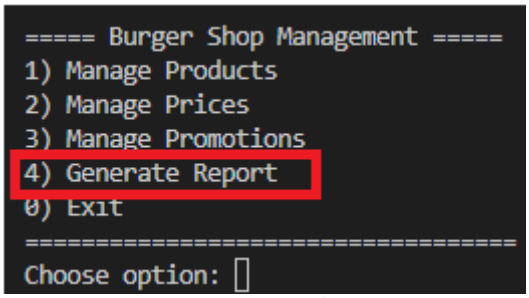
ภาพที่ 3.5 การเลือกใช้งานฟังก์ชั่นของ Manage Promotions

3.1.6 เมื่อเมนูฟังก์ชั่น Manage Promotions ขึ้นมาแล้วก็สามารถระบุเมนูที่ต้องการเลือกได้



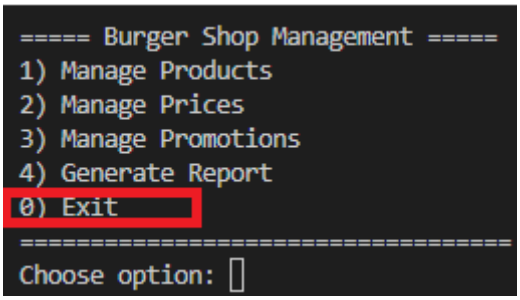
ภาพที่ 3.6 เมนูของ Mange Promotions

3.1.7 กรอกหมายเลขที่ 4 ภายในกรอบสีแดงเพื่อเรียกฟังก์ชั่น Generate report เพื่อเพิ่มไฟล์ report.txt ที่สามารถเขียน report รายละเอียดรายการอาหารทั้งหมด



ภาพที่ 3.7 การเลือกใช้งานฟังก์ชั่น Generate Report

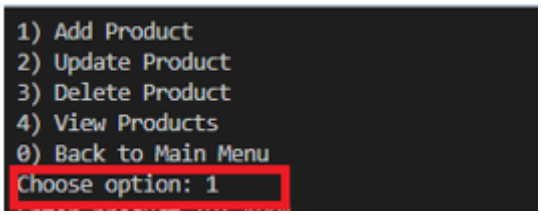
3.1.8 กรอกหมายเลข 5 ภายในกรอบสีแดงดพื่อเรียกฟังก์ชั่น Exit เพื่อออกจากโปรแกรม



ภาพที่ 3.8 การเลือกใช้ฟังก์ชั่นของ Exit

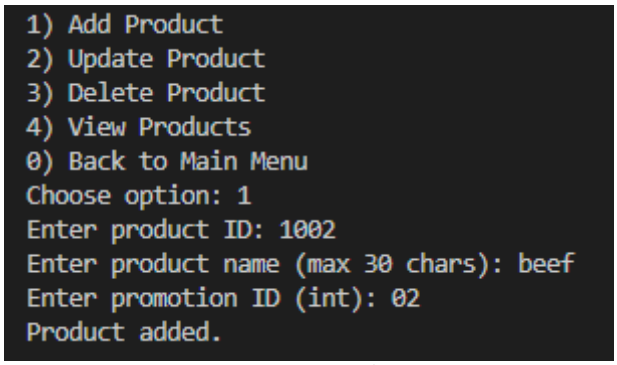
3.2 การใช้งานโปรแกรมเพิ่มข้อมูล

3.2.1 กรอกหมายเลข 1 เพื่อเพิ่มข้อมูลทั้งหมดของเมนูอาหารที่มีในโปรแกรม



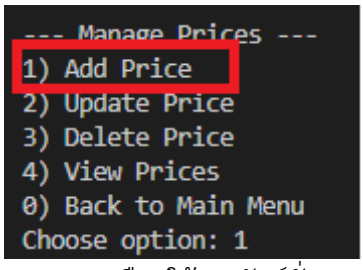
ภาพที่ 3.9 การเลือกใช้งานฟังก์ชั่น Add Products

3.2.2 เมื่อกดเลือกหมายเลข 1 จะปรากฏหัวข้อการใส่ข้อมูลรหัสสินค้า ชื่อสินค้า รหัสโปรโมชั่น



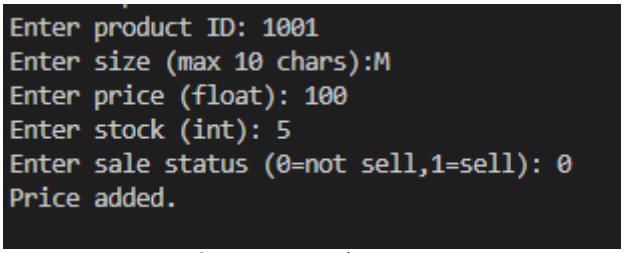
ภาพที่ 3.10 การเพิ่มสินค้า

3.2.3 กรอกหมายเลข 1 เพื่อเพิ่มข้อมูลราคาสิ้นค้าของเมนูอาหารที่มีในโปรแกรม



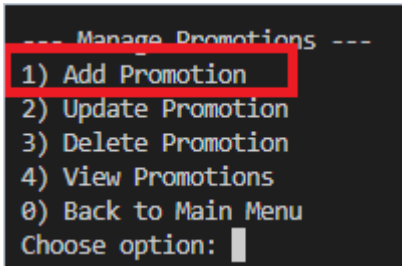
ภาพที่ 3.11 การเลือกใช้งานฟังก์ชั่น Add price

3.2.4 เมื่อกดเลือกหมายเลข 1 จะปรากฏหัวข้อการใส่ข้อมูลรหัสสินค้า



ภาพที่ 3.12 การเพิ่มราคาสินค้า

3.2.5 กรอกหมายเลข 1 เพื่อเพิ่มโปรโมชั่นสิ้นค้า



ภาพที่ 3.13 การเรียกใช้ฟังก์ชั่น Add Promotion

3.2.6 เมื่อกดเลือกหมายเลข 1 จะปรากฏหัวข้อการใส่ข้อมูลโปรโมชั่น

```
--- Manage Promotions ---
1) Add Promotion
2) Update Promotion
3) Delete Promotion
4) View Promotions
0) Back to Main Menu
Choose option: 1
Enter promotion ID (int): 011
Enter promotion name (max 30 chars): ลด30%
Promotion added.
```

ภาพที่ 3.14 การเพิ่มข้อมูลโปรโมชั่น

3.3 การใช้งานโปรแกรมอัพเดตสินค้า

3.3.1 กรอกหมายเลข 2 เพื่ออัพเดตข้อมูลสินค้าที่มีในโปรแกรม

```
--- Manage Products ---
1) Add Product
2) Update Product
3) Delete Product
4) View Products
0) Back to Main Menu
Choose option: 2
```

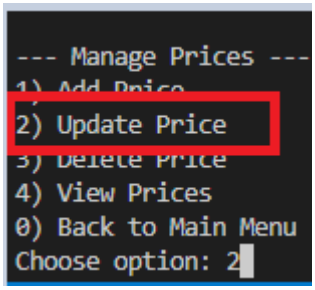
ภาพที่ 3.15 การเลือกใช้งานฟังชั่น Update Product

3.3.2 เมื่อกดเลือกหมายเลข 1 จะปรากฏหัวข้อการการอัพเดตสินค้า

```
Choose option: 2
Enter product ID to update: 1001
Current name: fish, promotion ID: 12
Enter new product name (leave blank to keep): beef
Enter new promotion ID (leave blank to keep): 011
Product updated.
```

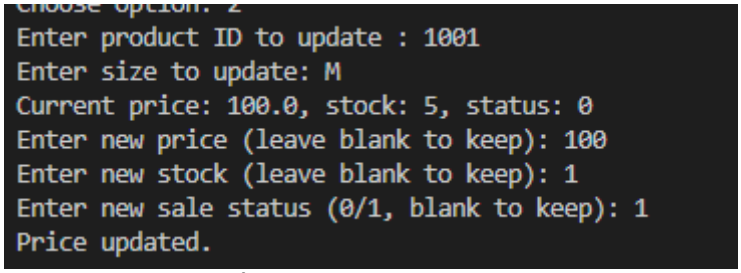
ภาพที่ 3.16 การอัพเดตข้อมูลสินค้า

3.3.3 กรอกหมายเลข 2 เพื่ออัพเดทราคาสินค้า



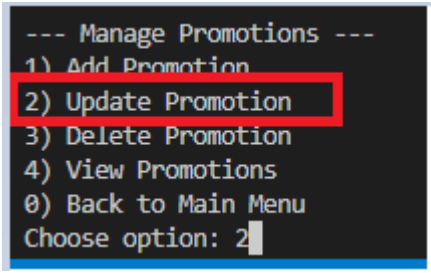
ภาพที่ 3.17 การเลือกใช้งานฟังก์ชั่น Update Price

3.3.4 เมื่อกดเลือกหมายเลข 2 จะปรากฏหัวข้อการการอัพเดตราคาสินค้า



ภาพที่ 3.18 การอัพเดทราคาสินค้า

3.3.5 กรอกหมายเลข 2 เพื่ออัพเดทโปรโมชั่นสินค้า



ภาพที่ 3.19 การใช้งานฟังก์ชั่น Update Promotion

3.3.6 เมื่อกดเลือกหมายเลข 2 จะปรากฏหัวข้อการการอัพเดตราคาสินค้า

```
1) Add Promotion
2) Update Promotion
3) Delete Promotion
4) View Promotions
0) Back to Main Menu
Choose option: 2
Enter promotion ID to update: 002
Current name: fish
Enter new promotion name (leave blank to keep): ลด5%
Promotion updated.
```

ภาพที่ 3.20 การอัพเดตโปรโมชั่นสินค้า

3.4 การใช้งานโปรแกรม ลบสินค้า

3.4.1 กรอกหมายเลข 3 เพื่อลบสินค้า เมื่อกดเลือกหมายเลข 3 จะปรากฏหัวข้อในการลบสินค้า

```
--- Manage Products ---
1) Add Product
2) Update Product
3) Delete Product
4) View Products
0) Back to Main Menu
Choose option: 3
Enter product ID to delete: F001
Product deleted.
```

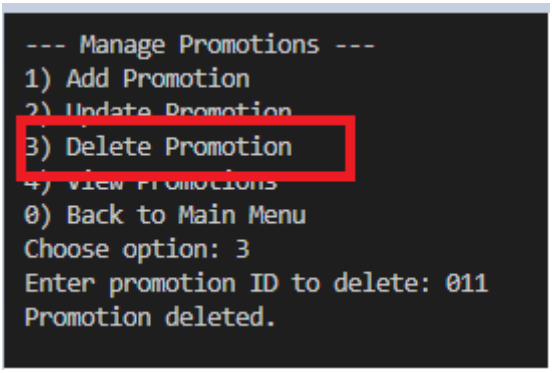
ภาพที่ 3.21 การใช้งานฟังก์ชั่น Delete Product

3.4.2 กรอกหมายเลข 3 เพื่อลบราคาสินค้า เมื่อกดเลือกหมายเลข 3 จะปรากฏหัวข้อในการลบราคาสินค้า

```
--- Manage Prices ---
1) Add Price
2) Update Price
3) Delete Price
4) View Prices
0) Back to Main Menu
Choose option: 3
Enter product ID to delete price: 1001
Enter size to delete: M
Price deleted.
```

ภาพที่ 3.22 การใช้งานฟังก์ชั่น Delete Price

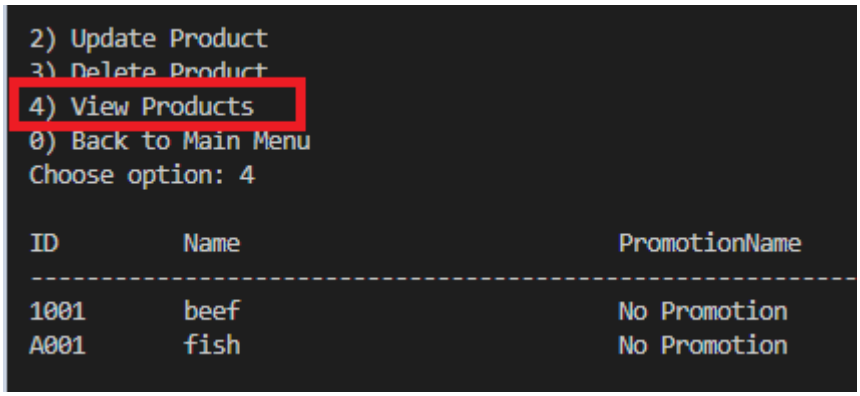
กรอกหมายเลข 3 เพื่อลบโปรโมชั่นสินค้า เมื่อกดเลือกหมายเลข 3 จะปรากฏหัวข้อในการลบโปรโมชั่นสินค้า



ภาพที่ 3.23 การใช้งานฟังก์ชั่น Delete Promotion

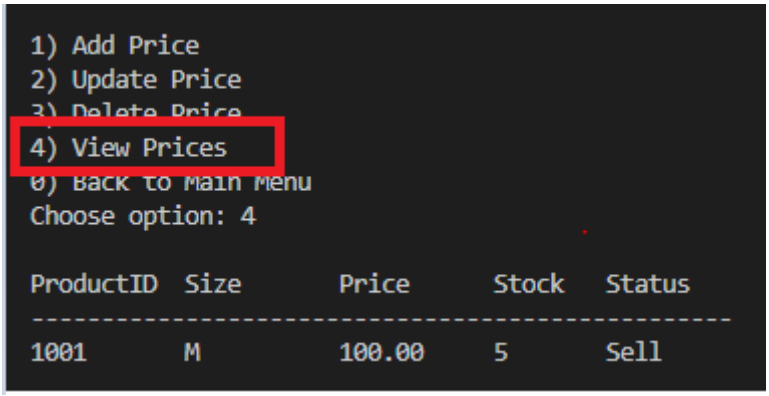
3.5 การใช้งานโปรแกรม แสดงสินค้า

3.5.1 กรอกหมายเลข 4 เพื่อแสดงชื่อสินค้า เมื่อกดเลือกหมายเลข 4 จะปรากฏชื่อสินค้า



ภาพที่ 3.24 การใช้งานฟังก์ชั่น View Product

3.5.2 กรอกหมายเลข 4 เพื่อแสดงชื่อสินค้า เมื่อกดเลือกหมายเลข 4 จะปรากฏราคาสินค้า



ภาพที่ 3.25 การใช้งานฟังก์ชั่น View Price

3.5.1 กรอกหมายเลข 4 เพื่อแสดงชื่อสินค้า เมื่อกดเลือกหมายเลข 4 จะปรากฏราคาสินค้า

ภาพที่ 3.26 การใช้งานฟังก์ชั่น View Promotions

3.5.2 เมื่อกดเลือกหมายเลข 2 จะปรากฏหัวข้อการใส่ข้อมูลรหัสสินค้าใหม่ ชื่อสินค้าใหม่ รหัสโปรโมชั่นใหม่ดังรูปที่ 3-12

```
--- Manage Products ---
1) Add Product
2) Update Product
3) Delete Product
4) View Products
0) Back to Main Menu
Choose option: 2
Enter product ID to update: 1002
Current name: beef, promotion ID: 2
Enter new product name (leave blank to keep): rice
Enter new promotion ID (leave blank to keep): 03
```

ภาพที่ 3.27 การเลือกใช้งานฟังก์ชั่น Update Product

3.6 การเลือกใช้งานโปรแกรม ลบสินค้า

3.6.1 กรอกหมายเลข 3 จะปรากฏหัวข้อให้ใส่ Promotion ID ดังรูปที่ 3-13

```
--- Manage Products ---
1) Add Product
2) Update Product
3) Delete Product
4) View Products
0) Back to Main Menu
Choose option: 3
Enter product ID to delete: 1002
Product deleted.
```

ภาพที่ 3.28 การเลือกใช้ฟังก์ชั่น Delete Product

3.7 การเลือกใช้งานโปรแกรม ดูรายละเอียดสินค้าทั้งหมด

3.7.1 กรอกหมายเลข 3 จะปรากฏข้อมูลสินค้าทั้งหมด ดังรูปที่ 3-14

```
4) View Products
0) Back to Main Menu
Choose option: 4

ID          Name                          PromotionName
-------------------------------------------------------------------
F001        fish burger                   -
P001        pork burger                   ลด10%
1001        20                            No Promotion
```

ภาพที่ 3.29 การเลือกใช้งานฟังก์ชั่น View Products

บทที่ 4

อธิบายการทำงานของ Code

4.1 ฟังก์ชั่นไบนารีพื้นฐานในระบบการจัดการร้านอาหารฟาสต์ฟู้ด

4.1.1 Module struct ใน Python เป็นเครื่องมือที่ช่วยให้การทำงานกับข้อมูลแบบไบนารีทำได้ง่ายและสะดวก สามารถใช้บีบอัดข้อมูลให้มีขนาดเล็ก ประหยัดพื้นที่ และส่งผ่านเครือข่ายได้รวดเร็ว เหมาะสำหรับระบบที่ต้องการ ความเร็วและประสิทธิภาพสูง เช่น ระบบจัดการร้านอาหาร, ระบบ IoT, หรือ การประมวลผลไฟล์เฉพาะทาง

```
import struct
```

ภาพที่ 4-1 Code Module Struct

4.1.2 Module os เป็นโมดูลมาตรฐานของภาษา Python ที่ใช้สำหรับ ติดต่อกับระบบปฏิบัติการ (Operating System) โดยตรง เช่น การจัดการไฟล์ โฟลเดอร์ และข้อมูลของระบบ โมดูลนี้ช่วยให้โปรแกรมสามารถทำงานเกี่ยวกับไฟล์และไดเรกทอรีได้โดยไม่ต้องพึ่งพาคำสั่งภายนอก

```
import os
```

ภาพที่ 4-2 Code Module OS

4.1.3 Module datetime เป็นโมดูลในภาษา Python ที่ใช้สำหรับการทำงานกับ วัน เวลา และวันที่ โดยเฉพาะ เช่น แสดงวันที่ปัจจุบัน บันทึกเวลาทำรายการ หรือคำนวณจำนวนวันระหว่างวันที่สองวัน การนำเข้าแบบ from datetime import datetime จะเป็นการนำคลาส datetime มาใช้งานโดยตรง ทำให้เรียกใช้สะดวก เช่น datetime.now()

```
from datetime import datetime
```

ภาพที่ 4-3 Code Module Datetime

4.1.4 Class Datetime เป็น คลาสหลัก ในโมดูล datetime ของ Python ซึ่งรวม วัน (date) และ เวลา (time) เข้าด้วยกัน ต่างจาก date ที่มีเฉพาะวัน ต่างจาก time ที่มีเฉพาะเวลา

4.1.5 Class Timezone ใช้สำหรับสร้าง เขตเวลา (time zone) แบบ offset จาก UTC (Coordinated Universal Time) มีประโยชน์เมื่อเราต้องทำงานกับระบบที่มีผู้ใช้งานหลายประเทศ เช่น ร้านของเราตั้งอยู่ประเทศไทย UTC+7

4.1.6 Class Timedelta ใช้แทน ช่วงเวลา (days, hours, minutes, seconds ฯลฯ) สามารถนำไปบวก/ลบกับ datetime ได้ ใช้คำนวณ ระยะเวลา เช่น บันทึกวันที่เพิ่มหรือปรับปรุงสินค้า

4.1.7 PRODUCT_STRUCT เป็นตัวกำหนด layout ของ record สินค้าแต่ละรายการ struct.Struct('<10s30si') หมายถึง: < คือ ใช้ little-endian (byte ต่ำก่อน byte สูง) ,10s คือ รหัสสินค้า (pro_id) ขนาด 10 bytes ,30s คือ ชื่อสินค้า (pro_name) ขนาด 30 bytes ,i คือ รหัสโปรโมชั่น (promotion_id) แบบ integer

```
PRODUCT_STRUCT = struct.Struct('<10s30si')
```

ภาพที่ 4-4 แสดงตัวอย่าง layout ของ PRODUCT_STRUCT

4.1.8 ฟังก์ชัน add_product ฟังก์ชัน add_product() มีหน้าที่ สร้างข้อมูลสินค้า 1 รายการและบันทึกลงไฟล์ไบนารี (products.dat) โดยทำงานตามขั้นตอนดังนี้ : รับข้อมูลสินค้า จากผู้ใช้ โดย รหัสสินค้า (pro_id), ชื่อสินค้า (pro_name) (จำกัดไม่เกิน 30 ตัวอักษร), รหัสโปรโมชั่น (promotion_id) ตรวจสอบรหัสสินค้า โดย อ่านสินค้าทั้งหมดจากไฟล์ ถ้าพบ pro_id ซ้ำกับสินค้าที่มีอยู่แล้วจะทำการแจ้งเตือนและไม่เพิ่ม เตรียมข้อมูลให้เข้ากับ Struct โดย ใช้ฟังก์ชัน pad_string() เพื่อเติม \x00 ให้ข้อความเต็มตามขนาดที่กำหนด, Pack ข้อมูลเป็น binary record ด้วย PRODUCT_STRUCT.pack() บันทึกข้อมูลต่อท้ายไฟล์ (ab) โดย เพิ่มสินค้าใหม่โดย ไม่ทับข้อมูลเดิม บันทึกเหตุการณ์ (Log) โดย ใช้ log_event() สำหรับบันทึกผู้ใช้งานและกิจกรรมที่ทำ จัดการข้อผิดพลาด โดย ใช้ try-except แจ้งผู้ใช้เมื่อเกิดข้อผิดพลาด

```python
def add_product():
    try:
        pro_id = input("Enter product ID: ").strip()
        pro_name = input("Enter product name (max 30 chars): ")[:30]
        promotion_id = int(input("Enter promotion ID (int): "))

        # Check if pro_id already exists
        products = read_all_records(PRODUCT_FILE, PRODUCT_STRUCT.size)
        for rec in products:
            pid_bytes, _, _ = PRODUCT_STRUCT.unpack(rec)
            pid = pid_bytes.decode('utf-8').rstrip('\x00')
            if pid == pro_id:
                print(f"Product ID {pro_id} already exists.")
                return

        packed = PRODUCT_STRUCT.pack(pad_string(pro_id, 10), pad_string(pro_name, 30), promotion_id)
        with open(PRODUCT_FILE, 'ab') as f:
            f.write(packed)

        log_event("USER", f"Add Product ID {pro_id}")
        print("Product added.")
    except Exception as e:
        print("Error adding product:", e)
```

ภาพที่ 4-5 แสดงตัวอย่างการทำงานของฟังก์ชัน add_product

4.1.9 ฟังก์ชัน update_product มีหน้าที่แก้ไขข้อมูลสินค้าที่มีอยู่ในไฟล์ไบนารี (products.dat) โดยทำงานโดยเริ่มจากการรับรหัสสินค้าที่ต้องการแก้ไขจากผู้ใช้ จากนั้นอ่านสินค้าทั้งหมดจากไฟล์ด้วยฟังก์ชัน read_all_records() แล้ววนลูปตรวจสอบรหัสสินค้าว่าตรงกับที่ผู้ใช้กรอกหรือไม่ ถ้าพบสินค้าที่ตรงกันจะแสดงข้อมูลปัจจุบันของสินค้า ได้แก่ ชื่อสินค้าและรหัสโปรโมชั่น พร้อมให้ผู้ใช้กรอกข้อมูลใหม่ของชื่อสินค้าและรหัสโปรโมชั่นโดยสามารถเว้นว่างเพื่อคงค่าปัจจุบันไว้ ข้อมูลที่ได้รับจะถูกเตรียมให้เข้ากับ Struct ด้วยการใช้ฟังก์ชัน pad_string() เพื่อเติม \x00 ให้ข้อความมีความยาวเต็มตามที่กำหนด และแปลงเป็น binary record ด้วย PRODUCT_STRUCT.pack() รายการสินค้าที่อัปเดตแล้วจะถูกเก็บในลิสต์ใหม่ ส่วนสินค้าที่ไม่ถูกแก้ไขจะถูกเก็บไว้ตามเดิม หลังจากตรวจสอบสินค้าทั้งหมดแล้ว หากพบสินค้าที่ต้องแก้ไข ฟังก์ชันจะเขียนข้อมูลทั้งหมดกลับไปยังไฟล์ด้วย write_all_records() โดยทับข้อมูลเดิม พร้อมบันทึกเหตุการณ์ของผู้ใช้ด้วย log_event() เพื่อระบุการแก้ไขสินค้า หากไม่พบสินค้าตามรหัสที่กรอกจะแจ้งผู้ใช้ว่า "Product ID not found." และฟังก์ชันยังมีการจัดการข้อผิดพลาดด้วย try-except เพื่อแจ้งผู้ใช้หากเกิดปัญหาใดๆ ระหว่างการทำงาน ทำให้การแก้ไขข้อมูลสินค้าสามารถทำได้อย่างปลอดภัยโดยไม่ทำลายข้อมูลเดิม

```python
def update_product():
    try:
        pro_id = input("Enter product ID to update: ").strip()
        products = read_all_records(PRODUCT_FILE, PRODUCT_STRUCT.size)
        updated = False
        new_products = []

        for rec in products:
            pid_bytes, pname_b, promo_id = PRODUCT_STRUCT.unpack(rec)
            pid = pid_bytes.decode('utf-8').rstrip('\x00')
            if pid == pro_id:
                print(f"Current name: {pname_b.decode('utf-8').rstrip(chr(0))}, promotion ID: {promo_id}")
                new_name = input("Enter new product name (leave blank to keep): ")
                new_promo = input("Enter new promotion ID (leave blank to keep): ")
                if new_name.strip() == '':
                    new_name = pname_b.decode('utf-8').rstrip(chr(0))
                if new_promo.strip() == '':
                    new_promo = promo_id
                else:
                    new_promo = int(new_promo)
                new_rec = PRODUCT_STRUCT.pack(pad_string(pro_id,10), pad_string(new_name,30), new_promo)
                new_products.append(new_rec)
                updated = True
            else:
                new_products.append(rec)

        if updated:
            write_all_records(PRODUCT_FILE, PRODUCT_STRUCT.size, new_products)
            log_event("USER", f"Update Product ID {pro_id}")
            print("Product updated.")
        else:
            print("Product ID not found.")
    except Exception as e:
        print("Error updating product:", e)
```

ภาพที่ 4-6 แสดงตัวอย่างการทำงานของฟังก์ชัน update_product

4.1.10 ฟังก์ชัน delete_product มีหน้าที่ลบข้อมูลสินค้าที่มีอยู่ในไฟล์ไบนารี (products.dat) โดยทำงานโดยเริ่มจากการรับรหัสสินค้าที่ต้องการลบจากผู้ใช้ จากนั้นอ่านสินค้าทั้งหมดจากไฟล์ด้วยฟังก์ชัน read_all_records() แล้ววนลูปตรวจสอบรหัสสินค้าว่าตรงกับที่ผู้ใช้กรอกหรือไม่ หากพบสินค้าที่ตรงกัน ฟังก์ชันจะไม่เก็บรายการนั้นไว้ในลิสต์ใหม่ (new_products) ส่วนรายการสินค้าที่ไม่ตรงกับรหัสจะถูกเก็บไว้เหมือนเดิม หลังจากตรวจสอบสินค้าทั้งหมดแล้ว หากพบสินค้าที่ต้องลบ ฟังก์ชันจะเขียนข้อมูลสินค้าที่เหลือกลับไปยังไฟล์ด้วย write_all_records() พร้อมบันทึกเหตุการณ์ของผู้ใช้ด้วย log_event() เพื่อระบุผู้ใช้งานและกิจกรรมที่ทำ นอกจากนี้ยังเรียกฟังก์ชัน delete_price_by_product() เพื่อลบข้อมูลราคาที่เกี่ยวข้องกับสินค้าที่ถูกลบ หากไม่พบสินค้าตามรหัสที่กรอกจะแจ้งผู้ใช้ว่า "Product ID not found." และฟังก์ชันมีการจัดการข้อผิดพลาดด้วย try-except เพื่อแจ้งผู้ใช้หากเกิดปัญหาใดๆ ระหว่างการทำงาน ทำให้การลบข้อมูลสินค้าสามารถทำได้อย่างปลอดภัยโดยไม่ทำลายข้อมูลอื่นในไฟล์

```
def view_products():
    products = read_all_records(PRODUCT_FILE, PRODUCT_STRUCT.size)
    promotions = read_all_records(PROMOTION_FILE, PROMOTION_STRUCT.size)

    promo_dict = {}
    for rec in promotions:
        pid, pname_b = PROMOTION_STRUCT.unpack(rec)
        pname = pname_b.decode('utf-8').rstrip('\x00')
        promo_dict[pid] = pname

    if not products:
        print("No products found.")
        return

    print(f"\n{'ID':<10} {'Name':<30} {'PromotionName':<30}")
    print("-" * 70)
    for rec in products:
        pid_bytes, pname_b, promo_id = PRODUCT_STRUCT.unpack(rec)
        pid = pid_bytes.decode('utf-8').rstrip('\x00')
        pname = pname_b.decode('utf-8').rstrip('\x00')
        promo_name = promo_dict.get(promo_id, "No Promotion")
        print(f"{pid:<10} {pname:<30} {promo_name:<30}")
```

ภาพที่ 4-7 แสดงตัวอย่างการทำงานของฟังก์ชัน delete_product

4.1.1 ฟังก์ชัน view_products มีหน้าที่แสดงรายการสินค้าที่บันทึกอยู่ในไฟล์ไบนารี (products.dat) พร้อมชื่อโปรโมชั่นที่เกี่ยวข้อง โดยทำงานโดยเริ่มจากการอ่านสินค้าทั้งหมดจากไฟล์ด้วยฟังก์ชัน read_all_records() และอ่านโปรโมชั่นทั้งหมดจากไฟล์ PROMOTION_FILE เพื่อนำข้อมูลมาสร้างพจนานุกรม (promo_dict) สำหรับจับคู่รหัสโปรโมชั่นกับชื่อโปรโมชั่น หลังจากนั้นจะตรวจสอบว่ามีสินค้าหรือไม่ หากไม่พบสินค้าจะแสดงข้อความแจ้งว่า "No products found." แต่ถ้ามีสินค้าจะทำการพิมพ์หัวตาราง โดยกำหนดช่องสำหรับ ID ของสินค้า ชื่อสินค้า และชื่อโปรโมชั่น จากนั้นวนลูปแต่ละรายการสินค้าเพื่อดึงข้อมูลรหัสสินค้า ชื่อสินค้า และรหัสโปรโมชั่นออกมาจาก binary record ด้วย PRODUCT_STRUCT.unpack() แปลงค่าเป็นสตริงโดยตัดตัว \x00 ออก แล้วหาชื่อโปรโมชั่นจากพจนานุกรม ถ้าไม่มีโปรโมชั่นจะแสดงเป็น "No Promotion" ข้อมูลสินค้าพร้อมชื่อโปรโมชั่นจะแสดงในรูปแบบตารางที่จัดช่องให้สวยงาม ทำให้ผู้ใช้สามารถดูรายการสินค้าทั้งหมดพร้อมรายละเอียดโปรโมชั่นได้อย่างชัดเจน

```python
def view_products():
    products = read_all_records(PRODUCT_FILE, PRODUCT_STRUCT.size)
    promotions = read_all_records(PROMOTION_FILE, PROMOTION_STRUCT.size)

    promo_dict = {}
    for rec in promotions:
        pid, pname_b = PROMOTION_STRUCT.unpack(rec)
        pname = pname_b.decode('utf-8').rstrip('\x00')
        promo_dict[pid] = pname

    if not products:
        print("No products found.")
        return

    print(f"\n{'ID':<10} {'Name':<30} {'PromotionName':<30}")
    print("-" * 70)
    for rec in products:
        pid_bytes, pname_b, promo_id = PRODUCT_STRUCT.unpack(rec)
        pid = pid_bytes.decode('utf-8').rstrip('\x00')
        pname = pname_b.decode('utf-8').rstrip('\x00')
        promo_name = promo_dict.get(promo_id, "No Promotion")
        print(f"{pid:<10} {pname:<30} {promo_name:<30}")
```

ภาพที่ 4-8 แสดงตัวอย่างการทำงานของฟังก์ชัน view_product

4.1.1 PRICE_STRUCT เป็นตัวกำหนด layout ของ record ราคาสินค้าแต่ละรายการ struct.Struct('<10s10sfiB') หมายถึง < คือ ใช้ little-endian (byte ต่ำก่อน byte สูง), 10s คือ รหัสสินค้า (product_id) ขนาด 10 bytes, 10s คือ ประเภทราคาสินค้า (price_type) ขนาด 10 bytes, f คือ ราคาสินค้า (price) แบบ float, i คือ จำนวนสินค้า (quantity) แบบ integer และ B คือ สถานะ (status) แบบ unsigned char

```python
PRICE_STRUCT = struct.Struct('<10s10sfiB')
```

ภาพที่ 4-9 แสดงตัวอย่าง layout ของ PRICE_STRUCT

4.1.2 ฟังก์ชัน add_price มีหน้าที่เพิ่มข้อมูลราคาของสินค้าแต่ละขนาดลงในไฟล์ไบนารี (PRICE_FILE) โดยทำงานโดยเริ่มจากการรับข้อมูลจากผู้ใช้ ได้แก่ รหัสสินค้า (product_id), ขนาดสินค้า (size) ไม่เกิน 10 ตัวอักษร, ราคาสินค้า (price) เป็น float, จำนวนสินค้า (stock) เป็น integer และสถานะการขาย (sale_status) โดยต้องระบุเป็น 0 สำหรับไม่ขาย หรือ 1 สำหรับขาย หลังจากรับข้อมูล ฟังก์ชันจะตรวจสอบว่ารหัสสินค้าที่กรอกมีอยู่ในไฟล์สินค้า (PRODUCT_FILE) หรือไม่ หากไม่พบจะแจ้งผู้ใช้ให้เพิ่มสินค้าก่อน จากนั้นจะอ่านไฟล์ PRICE_FILE เพื่อตรวจสอบว่ามีข้อมูลราคาของสินค้ารหัสเดียวกันและขนาดเดียวกันอยู่แล้วหรือไม่ หากพบจะไม่เพิ่มข้อมูลซ้ำ ข้อมูลใหม่จะถูกเตรียมให้เข้ากับ Struct ด้วยการใช้ฟังก์ชัน pad_string() เติม \x00 ให้ข้อความมีความยาวเต็มตามที่กำหนด และแปลงเป็น binary record ด้วย PRICE_STRUCT.pack() จากนั้นจะเขียนข้อมูลต่อท้ายไฟล์ด้วยโหมด append (ab) เพื่อไม่ให้ทับข้อมูลเดิม พร้อมบันทึกเหตุการณ์ของผู้ใช้ด้วย log_event() เพื่อระบุรหัสสินค้าและขนาดสินค้าที่เพิ่มราคาลงไป ฟังก์ชันยังมีการจัดการข้อผิดพลาดด้วย try-except เพื่อแจ้งผู้ใช้หากเกิดปัญหาใดๆ ระหว่างการทำงาน ทำให้สามารถเพิ่มราคาสินค้าได้อย่างปลอดภัยและถูกต้อง

```python
def add_price():
    try:
        pro_id = input("Enter product ID: ").strip()
        pro_size = input("Enter size (max 10 chars): ")[:10]
        pro_price = float(input("Enter price (float): "))
        pro_stock = int(input("Enter stock (int): "))
        sale_status = int(input("Enter sale status (0=not sell,1=sell): "))
        if sale_status not in (0,1):
            print("Sale status must be 0 or 1")
            return

        products = read_all_records(PRODUCT_FILE, PRODUCT_STRUCT.size)
        if not any(PRODUCT_STRUCT.unpack(p)[0].decode('utf-8').rstrip('\x00') == pro_id for p in products):
            print("Product ID does not exist. Please add product first.")
            return

        prices = read_all_records(PRICE_FILE, PRICE_STRUCT.size)
        for p in prices:
            pid, size_b, _, _, _ = PRICE_STRUCT.unpack(p)
            pid_str = pid.decode('utf-8').rstrip('\x00')
            size = size_b.decode('utf-8').rstrip('\x00')
            if pid_str == pro_id and size == pro_size:
                print("Price record for this product and size already exists.")
                return

        packed = PRICE_STRUCT.pack(pad_string(pro_id, 10), pad_string(pro_size,10), pro_price, pro_stock, sale_status)
        with open(PRICE_FILE, 'ab') as f:
            f.write(packed)

        log_event("USER", f"Add Price for Product ID {pro_id}, size {pro_size}")
        print("Price added.")
    except Exception as e:
        print("Error adding price:", e)
```

ภาพที่ 4.10 แสดงตัวอย่างการทำงานของฟังก์ชัน add_price

4.1.1 ฟังก์ชัน update_price มีหน้าที่แก้ไขข้อมูลราคาของสินค้าที่มีอยู่ในไฟล์ไบนารี (PRICE_FILE) โดยทำงานโดยเริ่มจากการรับรหัสสินค้า (product_id) และขนาดสินค้า (size) ที่ต้องการแก้ไขจากผู้ใช้ จากนั้นอ่านข้อมูลราคาสินค้าทั้งหมดจากไฟล์ด้วยฟังก์ชัน read_all_records() แล้ววนลูปตรวจสอบว่ารหัสสินค้าและขนาดตรงกับที่ผู้ใช้กรอกหรือไม่ หากพบรายการที่ตรงกัน จะแสดงข้อมูลปัจจุบันของราคา (price), จำนวนสินค้า (stock) และสถานะการขาย (status) พร้อมให้ผู้ใช้กรอกข้อมูลใหม่โดยสามารถเว้นว่างเพื่อคงค่าปัจจุบันไว้ ข้อมูลที่ได้รับจะถูกเตรียมให้เข้ากับ Struct ด้วยการใช้ฟังก์ชัน pad_string() เติม \x00 ให้ข้อความมีความยาวเต็มตามที่กำหนด และแปลงเป็น binary record ด้วย PRICE_STRUCT.pack() จากนั้นจะสร้างลิสต์ใหม่ของราคาสินค้าทั้งหมดรวมรายการที่แก้ไขแล้ว หลังจากตรวจสอบรายการทั้งหมดแล้ว หากมีรายการที่ถูกแก้ไข ฟังก์ชันจะเขียนข้อมูลทั้งหมดกลับไปยังไฟล์ด้วย write_all_records() พร้อมบันทึกเหตุการณ์ของผู้ใช้ด้วย log_event() เพื่อระบุรหัสสินค้าและขนาดสินค้าที่แก้ไข หากไม่พบรายการตามรหัสและขนาดที่กรอกจะแจ้งผู้ใช้ว่า "Price record not found." ฟังก์ชันมีการจัดการข้อผิดพลาดด้วย try-except เพื่อแจ้งผู้ใช้หากเกิดปัญหาใดๆ ระหว่างการทำงาน ทำให้การแก้ไขราคาสินค้าสามารถทำได้อย่างปลอดภัยและถูกต้อง

```python
def update_price():
    try:
        pro_id = input("Enter product ID to update : ").strip()
        pro_size = input("Enter size to update: ")[:10]
        prices = read_all_records(PRICE_FILE, PRICE_STRUCT.size)
        updated = False
        new_prices = []

        for rec in prices:
            pid, size_b, price, stock, status = PRICE_STRUCT.unpack(rec)
            pid_str = pid.decode('utf-8').rstrip('\x00')
            size = size_b.decode('utf-8').rstrip('\x00')
            if pid_str == pro_id and size == pro_size:
                print(f"Current price: {price}, stock: {stock}, status: {status}")
                new_price = input("Enter new price (leave blank to keep): ")
                new_stock = input("Enter new stock (leave blank to keep): ")
                new_status = input("Enter new sale status (0/1, blank to keep): ")

                if new_price.strip() == '':
                    new_price = price
                else:
                    new_price = float(new_price)

                if new_stock.strip() == '':
                    new_stock = stock
                else:
                    new_stock = int(new_stock)

                if new_status.strip() == '':
                    new_status = status
                else:
                    new_status = int(new_status)
                    if new_status not in (0,1):
                        print("Sale status must be 0 or 1")
                        return

                new_rec = PRICE_STRUCT.pack(pad_string(pro_id,10), pad_string(pro_size,10), new_price, new_stock, new_status)
                new_prices.append(new_rec)
                updated = True
            else:
                new_prices.append(rec)

        if updated:
            write_all_records(PRICE_FILE, PRICE_STRUCT.size, new_prices)
            log_event("USER", f"Update Price Product ID {pro_id} Size {pro_size}")
            print("Price updated.")
        else:
            print("Price record not found.")
    except Exception as e:
        print("Error updating price:", e)
```

ภาพที่ 4.11 แสดงตัวอย่างการทำงานของฟังก์ชัน update_price

4.1.2 ฟังก์ชัน delete_price และ delete_price_by_product มีหน้าที่ลบข้อมูลราคาของสินค้าตามรหัสสินค้า (product_id) และขนาดสินค้า (size) ที่ผู้ใช้ระบุ โดยทำงานโดยเริ่มจากการรับรหัสสินค้าและขนาดสินค้าที่ต้องการลบจากผู้ใช้ จากนั้นอ่านข้อมูลราคาสินค้าทั้งหมดจากไฟล์ PRICE_FILE ด้วยฟังก์ชัน read_all_records() วนลูปตรวจสอบแต่ละรายการ หากพบรายการที่ตรงกับรหัสสินค้าและขนาด ฟังก์ชันจะไม่เก็บรายการนั้นในลิสต์ใหม่ (new_prices) ส่วนรายการที่ไม่ตรงกับเงื่อนไขจะถูกเก็บไว้เหมือนเดิม หลังจากตรวจสอบทั้งหมดแล้ว หากมีรายการถูกลบ ฟังก์ชันจะเขียนข้อมูลที่เหลือกลับไปยังไฟล์ด้วย write_all_records() พร้อมบันทึกเหตุการณ์ของผู้ใช้ด้วย log_event() เพื่อระบุรหัสสินค้าและขนาดสินค้าที่ลบ หากไม่พบรายการตามเงื่อนไขจะแจ้งผู้ใช้ว่า "Price record not found." ฟังก์ชันมีการจัดการข้อผิดพลาดด้วย try-except เพื่อแจ้งผู้ใช้หากเกิดปัญหาใดๆ ระหว่างการทำงาน ฟังก์ชัน delete_price_by_product(pro_id) ทำงานในลักษณะอัตโนมัติสำหรับการลบราคาทั้งหมดของสินค้าที่ถูกลบจากไฟล์สินค้า โดยจะอ่านข้อมูลราคาทั้งหมดจาก PRICE_FILE วนลูปตรวจสอบรหัสสินค้า หากรหัสไม่ตรงกับสินค้าที่ถูกลบ จะเก็บรายการไว้ในลิสต์ใหม่ และเขียนข้อมูลกลับไปยังไฟล์พร้อมบันทึกเหตุการณ์ระบบด้วย log_event() ระบุว่าลบราคาทั้งหมดของสินค้ารหัสนั้น ทำให้การจัดการราคาสินค้าเป็นไปอย่างปลอดภัยและสอดคล้องกับการลบสินค้าที่เกี่ยวข้อง

```python
def delete_price():
    try:
        pro_id = input("Enter product ID to delete price: ").strip()
        pro_size = input("Enter size to delete: ")[:10]
        prices = read_all_records(PRICE_FILE, PRICE_STRUCT.size)
        new_prices = []
        deleted = False

        for rec in prices:
            pid, size_b, _, _, _ = PRICE_STRUCT.unpack(rec)
            pid_str = pid.decode('utf-8').rstrip('\x00')
            size = size_b.decode('utf-8').rstrip('\x00')
            if pid_str == pro_id and size == pro_size:
                deleted = True
                continue
            new_prices.append(rec)

        if deleted:
            write_all_records(PRICE_FILE, PRICE_STRUCT.size, new_prices)
            log_event("USER", f"Delete Price Product ID {pro_id} Size {pro_size}")
            print("Price deleted.")
        else:
            print("Price record not found.")
    except Exception as e:
        print("Error deleting price:", e)

def delete_price_by_product(pro_id):
    prices = read_all_records(PRICE_FILE, PRICE_STRUCT.size)
    new_prices = []
    for p in prices:
        pid_bytes = PRICE_STRUCT.unpack(p)[0]
        pid = pid_bytes.decode('utf-8').rstrip('\x00')
        if pid != pro_id:
            new_prices.append(p)
    write_all_records(PRICE_FILE, PRICE_STRUCT.size, new_prices)
    log_event("SYSTEM", f"Delete all prices of deleted product ID {pro_id}")
```

ภาพที่ 4.12 แสดงตัวอย่างการทำงานของฟังก์ชัน delete_price()

4.1.3 ฟังก์ชัน view_prices มีหน้าที่แสดงรายการราคาของสินค้าทั้งหมดที่บันทึกอยู่ในไฟล์ไบนารี (PRICE_FILE) โดยทำงานโดยเริ่มจากการอ่านข้อมูลราคาทั้งหมดจากไฟล์ด้วยฟังก์ชัน read_all_records() หากไม่พบข้อมูลราคาจะพิมพ์ข้อความแจ้งว่า "No price records found." แต่ถ้ามีข้อมูล ราคาสินค้าทั้งหมดจะถูกแสดงในรูปแบบตาราง โดยมีคอลัมน์สำหรับรหัสสินค้า (ProductID), ขนาดสินค้า (Size), ราคาสินค้า (Price), จำนวนสินค้า (Stock) และสถานะการขาย (Status) ฟังก์ชันจะวนลูปแต่ละรายการราคาสินค้า ใช้ PRICE_STRUCT.unpack() เพื่อดึงข้อมูลรหัสสินค้า ขนาด ราคา จำนวน และสถานะ จากนั้นแปลงข้อมูลรหัสสินค้าและขนาดเป็นสตริงโดยตัดตัว \x00 ออก และแปลงสถานะเป็นข้อความ "Sell" หรือ "Not Sell" ข้อมูลราคาสินค้าแต่ละรายการจะถูกพิมพ์ออกมาเรียงตามตาราง ทำให้ผู้ใช้สามารถดูราคาของสินค้าทั้งหมดพร้อมรายละเอียดได้อย่างชัดเจน

```
def view_prices():
    prices = read_all_records(PRICE_FILE, PRICE_STRUCT.size)
    if not prices:
        print("No price records found.")
        return
    print(f"\n{'ProductID':<10} {'Size':<10} {'Price':<10} {'Stock':<7} {'Status':<6}")
    print("-"*50)
    for rec in prices:
        pid, size_b, price, stock, status = PRICE_STRUCT.unpack(rec)
        pid_str = pid.decode('utf-8').rstrip('\x00')
        size = size_b.decode('utf-8').rstrip('\x00')
        status_str = "Sell" if status == 1 else "Not Sell"
        print(f"{pid_str:<10} {size:<10} {price:<10.2f} {stock:<7} {status_str:<6}")
```

ภาพที่ 4.13 แสดงตัวอย่างการทำงานของฟังก์ชัน view_prices

4.1.4 PROMOTION_STRUCT เป็นตัวกำหนด layout ของ record โปรโมชั่นแต่ละรายการ struct.Struct('<i30s') หมายถึง < คือ ใช้ little-endian (byte ต่ำก่อน byte สูง), i คือ รหัสโปรโมชั่น (promotion_id) แบบ integer, และ 30s คือ ชื่อโปรโมชั่น (promotion_name) ขนาด 30 bytes ซึ่งใช้สำหรับจัดเก็บข้อมูลโปรโมชั่นสินค้า ทำให้สามารถ pack และ unpack ข้อมูลไบนารีได้อย่างถูกต้อง

```
PROMOTION_STRUCT = struct.Struct('<i30s')
```

ภาพที่ 4.14 แสดงตัวอย่าง layout ของ PROMOTION_STRUCT

4.1.5 ฟังก์ชัน add_promotion มีหน้าที่เพิ่มข้อมูลโปรโมชั่นสินค้าใหม่ลงในไฟล์ไบนารี (PROMOTION_FILE) โดยทำงานโดยเริ่มจากการรับรหัสโปรโมชั่น (promotion_id) เป็นจำนวนเต็ม และชื่อโปรโมชั่น (promotion_name) ไม่เกิน 30 ตัวอักษร จากนั้นอ่านข้อมูลโปรโมชั่นทั้งหมดจากไฟล์ด้วยฟังก์ชัน read_all_records() เพื่อตรวจสอบว่ารหัสโปรโมชั่นที่กรอกมีอยู่แล้วหรือไม่ หากพบว่าซ้ำจะแจ้งผู้ใช้และไม่เพิ่มข้อมูลใหม่ ข้อมูลโปรโมชั่นใหม่จะถูกเตรียมให้เข้ากับ Struct ด้วยการใช้ฟังก์ชัน pad_string() เติม \x00 ให้ข้อความมีความยาวเต็มตามที่กำหนด และแปลงเป็น binary record ด้วย PROMOTION_STRUCT.pack() จากนั้นจะเขียนข้อมูลต่อท้ายไฟล์ด้วยโหมด append (ab) เพื่อไม่ให้ทับข้อมูลเดิม พร้อมบันทึกเหตุการณ์ของผู้ใช้ด้วย log_event() เพื่อระบุรหัสโปรโมชั่นที่เพิ่ม ฟังก์ชันมีการจัดการข้อผิดพลาดด้วย try-except เพื่อแจ้งผู้ใช้หากเกิดปัญหาใดๆ ระหว่างการทำงาน ทำให้สามารถเพิ่มโปรโมชั่นสินค้าได้อย่างปลอดภัยและถูกต้อง

```
def add_promotion():
    try:
        promotion_id = int(input("Enter promotion ID (int): "))
        promotion_name = input("Enter promotion name (max 30 chars): ")[:30]
        promotions = read_all_records(PROMOTION_FILE, PROMOTION_STRUCT.size)
        for rec in promotions:
            pid, _ = PROMOTION_STRUCT.unpack(rec)
            if pid == promotion_id:
                print("Promotion ID already exists.")
                return
        packed = PROMOTION_STRUCT.pack(promotion_id, pad_string(promotion_name, 30))
        with open(PROMOTION_FILE, 'ab') as f:
            f.write(packed)
        log_event("USER", f"Add Promotion ID {promotion_id}")
        print("Promotion added.")
    except Exception as e:
        print("Error adding promotion:", e)
```

ภาพที่ 4.15 แสดงตัวอย่างการทำงานของฟังก์ชัน add_promotion

4.1.6 ฟังก์ชัน update_promotion มีหน้าที่แก้ไขข้อมูลโปรโมชั่นที่มีอยู่ในไฟล์ไบนารี (PROMOTION_FILE) โดยทำงานโดยเริ่มจากการรับรหัสโปรโมชั่น (promotion_id) ที่ต้องการแก้ไขจากผู้ใช้ จากนั้นอ่านข้อมูลโปรโมชั่นทั้งหมดจากไฟล์ด้วยฟังก์ชัน read_all_records() แล้ววนลูปตรวจสอบว่ารหัสโปรโมชั่นตรงกับที่ผู้ใช้กรอกหรือไม่ หากพบรายการที่ตรงกัน จะแสดงชื่อโปรโมชั่นปัจจุบันและให้ผู้ใช้กรอกชื่อโปรโมชั่นใหม่ โดยสามารถเว้นว่างเพื่อคงค่าปัจจุบันไว้ ข้อมูลใหม่จะถูกเตรียมให้เข้ากับ Struct ด้วยการใช้ฟังก์ชัน pad_string() เติม \x00 ให้ข้อความมีความยาวเต็มตามที่กำหนด และแปลงเป็น binary record ด้วย PROMOTION_STRUCT.pack() หลังจากตรวจสอบรายการทั้งหมดแล้ว หากมีรายการถูกแก้ไข ฟังก์ชันจะเขียนข้อมูลทั้งหมดกลับไปยังไฟล์ด้วย write_all_records() พร้อมบันทึกเหตุการณ์ของผู้ใช้ด้วย log_event() เพื่อระบุรหัสโปรโมชั่นที่แก้ไข หากไม่พบรายการตามรหัสที่กรอกจะแจ้งผู้ใช้ว่า "Promotion ID not found." ฟังก์ชันมีการจัดการข้อผิดพลาดด้วย try-except เพื่อแจ้งผู้ใช้หากเกิดปัญหาใดๆ ระหว่างการทำงาน ทำให้การแก้ไขโปรโมชั่นสินค้าเป็นไปอย่างปลอดภัยและถูกต้อง

```
def update_promotion():
    try:
        promotion_id = int(input("Enter promotion ID to update: "))
        promotions = read_all_records(PROMOTION_FILE, PROMOTION_STRUCT.size)
        updated = False
        new_promos = []
        for rec in promotions:
            pid, pname_b = PROMOTION_STRUCT.unpack(rec)
            pname = pname_b.decode('utf-8').rstrip('\x00')
            if pid == promotion_id:
                print(f"Current name: {pname}")
                new_name = input("Enter new promotion name (leave blank to keep): ")
                if new_name.strip() == '':
                    new_name = pname
                new_rec = PROMOTION_STRUCT.pack(promotion_id, pad_string(new_name,30))
                new_promos.append(new_rec)
                updated = True
            else:
                new_promos.append(rec)
        if updated:
            write_all_records(PROMOTION_FILE, PROMOTION_STRUCT.size, new_promos)
            log_event("USER", f"Update Promotion ID {promotion_id}")
            print("Promotion updated.")
        else:
            print("Promotion ID not found.")
    except Exception as e:
        print("Error updating promotion:", e)
```

ภาพที่ 4.16 แสดงตัวอย่างการทำงานของฟังก์ชัน update_promotion

4.1.7 ฟังก์ชัน delete_promotion มีหน้าที่ลบข้อมูลโปรโมชั่นสินค้าที่มีอยู่ในไฟล์ไบนารี (PROMOTION_FILE) โดยทำงานโดยเริ่มจากการรับรหัสโปรโมชั่น (promotion_id) ที่ผู้ใช้ต้องการลบ จากนั้นอ่านข้อมูลโปรโมชั่นทั้งหมดจากไฟล์ด้วยฟังก์ชัน read_all_records() และวนลูปตรวจสอบแต่ละรายการ หากพบรายการที่ตรงกับรหัสโปรโมชั่น ฟังก์ชันจะไม่เก็บรายการนั้นในลิสต์ใหม่ (new_promos) ส่วนรายการที่ไม่ตรงจะถูกเก็บไว้เหมือนเดิม หลังจากตรวจสอบทั้งหมดแล้ว หากมีรายการถูกลบ ฟังก์ชันจะเขียนข้อมูลที่เหลือกลับไปยังไฟล์ด้วย write_all_records() พร้อมบันทึกเหตุการณ์ของผู้ใช้ด้วย log_event() เพื่อระบุรหัสโปรโมชั่นที่ถูกลบ หากไม่พบรายการตามรหัสที่กรอกจะแจ้งผู้ใช้ว่า "Promotion ID not found." ฟังก์ชันมีการจัดการข้อผิดพลาดด้วย try-except เพื่อแจ้งผู้ใช้หากเกิดปัญหาใดๆ ระหว่างการทำงาน ทำให้การลบโปรโมชั่นสินค้าเป็นไปอย่างปลอดภัยและถูกต้อง

```python
def delete_promotion():
    try:
        promotion_id = int(input("Enter promotion ID to delete: "))
        promotions = read_all_records(PROMOTION_FILE, PROMOTION_STRUCT.size)
        new_promos = []
        deleted = False
        for rec in promotions:
            pid, _ = PROMOTION_STRUCT.unpack(rec)
            if pid == promotion_id:
                deleted = True
                continue
            new_promos.append(rec)
        if deleted:
            write_all_records(PROMOTION_FILE, PROMOTION_STRUCT.size, new_promos)
            log_event("USER", f"Delete Promotion ID {promotion_id}")
            print("Promotion deleted.")
        else:
            print("Promotion ID not found.")
    except Exception as e:
        print("Error deleting promotion:", e)
```

ภาพที่ 4.17 แสดงตัวอย่างการทำงานของฟังก์ชัน delete_promotion

4.1.1 ฟังก์ชัน view_promotions มีหน้าที่แสดงรายการโปรโมชั่นสินค้าทั้งหมดที่บันทึกอยู่ในไฟล์ไบนารี (PROMOTION_FILE) โดยทำงานโดยเริ่มจากการอ่านข้อมูลโปรโมชั่นทั้งหมดจากไฟล์ด้วยฟังก์ชัน read_all_records() หากไม่พบข้อมูลโปรโมชั่นจะแสดงข้อความแจ้งว่า "No promotions found." แต่ถ้ามีข้อมูลโปรโมชั่น ฟังก์ชันจะพิมพ์หัวตาราง โดยมีคอลัมน์สำหรับรหัสโปรโมชั่น (PromotionID) และชื่อโปรโมชั่น (Name) จากนั้นวนลูปแต่ละรายการโปรโมชั่น ใช้ PROMOTION_STRUCT.unpack() เพื่อดึงข้อมูลรหัสโปรโมชั่นและชื่อโปรโมชั่น แปลงชื่อโปรโมชั่นเป็นสตริงโดยตัดตัว \x00 ออก ข้อมูลโปรโมชั่นแต่ละรายการจะถูกพิมพ์ออกมาเรียงตามตาราง ทำให้ผู้ใช้สามารถดูโปรโมชั่นทั้งหมดพร้อมรายละเอียดได้อย่างชัดเจน

```python
def view_promotions():
    promotions = read_all_records(PROMOTION_FILE, PROMOTION_STRUCT.size)
    if not promotions:
        print("No promotions found.")
        return
    print(f"\n{'PromotionID':<12} {'Name':<30}")
    print("-" * 45)
    for rec in promotions:
        pid, pname_b = PROMOTION_STRUCT.unpack(rec)
        pname = pname_b.decode('utf-8').rstrip('\x00')
        print(f"{pid:<12} {pname:<30}")
```

ภาพที่ 4.18 แสดงตัวอย่างการทำงานของฟังก์ชัน view_promotions

4.1.2 ฟังก์ชัน manage_products_menu มีหน้าที่สร้างเมนูสำหรับจัดการสินค้าภายในระบบ ให้ผู้ใช้สามารถเข้าถึงฟังก์ชันจัดการสินค้า ได้แก่ การเพิ่มสินค้า (add_product()), แก้ไขสินค้า (update_product()), ลบสินค้า (delete_product()), และดูรายการสินค้าทั้งหมด (view_products()) โดยทำงานในลูป while True เพื่อให้เมนูวนซ้ำจนกว่าผู้ใช้จะเลือกกลับไปยังเมนูหลัก (0) เมนูจะแสดงตัวเลือกพร้อมข้อความคำสั่ง และรอรับตัวเลือกจากผู้ใช้ (choice) หากผู้ใช้กรอกตัวเลือกที่ถูกต้อง ฟังก์ชันจะเรียกฟังก์ชันที่สอดคล้องกับตัวเลือกนั้น หากกรอกตัวเลือกไม่ถูกต้อง จะแสดงข้อความแจ้งว่า "Invalid option" ทำให้ผู้ใช้สามารถจัดการสินค้าได้อย่างง่ายดายและสะดวกผ่านเมนูแบบ text-based

```python
def manage_products_menu():
    while True:
        print("\n--- Manage Products ---")
        print("1) Add Product")
        print("2) Update Product")
        print("3) Delete Product")
        print("4) View Products")
        print("0) Back to Main Menu")
        choice = input("Choose option: ")
        if choice == '1':
            add_product()
        elif choice == '2':
            update_product()
        elif choice == '3':
            delete_product()
        elif choice == '4':
            view_products()
        elif choice == '0':
            break
        else:
            print("Invalid option")
```

ภาพที่ 4.19 แสดงตัวอย่างเมนูจัดการสินค้าใน console

4.1.1 ฟังก์ชัน manage_prices_menu มีหน้าที่สร้างเมนูสำหรับจัดการราคาสินค้า ให้ผู้ใช้สามารถเข้าถึงฟังก์ชันจัดการราคาสินค้า ได้แก่ การเพิ่มราคา (add_price()), แก้ไขราคา (update_price()), ลบราคา (delete_price()), และดูราคาสินค้าทั้งหมด (view_prices()) โดยทำงานในลูป while True เพื่อให้เมนูวนซ้ำจนกว่าผู้ใช้จะเลือกกลับไปยังเมนูหลัก (0) เมนูจะแสดงตัวเลือกพร้อมข้อความคำสั่ง และรอรับตัวเลือกจากผู้ใช้ (choice) หากผู้ใช้กรอกตัวเลือกที่ถูกต้อง ฟังก์ชันจะเรียกฟังก์ชันที่สอดคล้องกับตัวเลือกนั้น หากกรอกตัวเลือกไม่ถูกต้อง จะแสดงข้อความแจ้งว่า "Invalid option" ทำให้ผู้ใช้สามารถจัดการราคาสินค้าได้อย่างสะดวกและง่ายดายผ่านเมนูแบบ text-based

```python
def manage_prices_menu():
    while True:
        print("\n--- Manage Prices ---")
        print("1) Add Price")
        print("2) Update Price")
        print("3) Delete Price")
        print("4) View Prices")
        print("0) Back to Main Menu")
        choice = input("Choose option: ")
        if choice == '1':
            add_price()
        elif choice == '2':
            update_price()
        elif choice == '3':
            delete_price()
        elif choice == '4':
            view_prices()
        elif choice == '0':
            break
        else:
            print("Invalid option")
```

ภาพที่ 4.20 แสดงตัวอย่างเมนูจัดการราคาสินค้าใน console

4.1.1 ฟังก์ชัน manage_promotions_menu มีหน้าที่สร้างเมนูสำหรับ จัดการโปรโมชัน ให้ผู้ใช้สามารถเข้าถึงฟังก์ชันที่เกี่ยวข้องกับการจัดการโปรโมชัน ได้แก่ การ เพิ่มโปรโมชัน (add_promotion()), การ แก้ไขโปรโมชัน (update_promotion()), การ ลบโปรโมชัน (delete_promotion()), และการ ดูโปรโมชันทั้งหมด (view_promotions()) โดยฟังก์ชันนี้จะทำงานใน ลูป while True เพื่อให้เมนูวนซ้ำและพร้อมรับคำสั่งตลอดเวลา จนกว่าผู้ใช้จะเลือกกลับไปยังเมนูหลัก (0) ซึ่งจะทำให้ลูปหยุดทำงาน ภายในลูป เมนูจะแสดงตัวเลือกพร้อมข้อความคำสั่งที่ชัดเจน จากนั้นจะรอรับตัวเลือกจากผู้ใช้ (choice) หากผู้ใช้กรอกตัวเลือกที่ถูกต้อง ฟังก์ชันจะเรียกใช้ฟังก์ชันย่อยที่สอดคล้องกับตัวเลือกนั้นทันที แต่หากผู้ใช้กรอกตัวเลือกที่ไม่ถูกต้อง ฟังก์ชันจะแสดงข้อความแจ้งว่า "Invalid option" การออกแบบนี้ช่วยให้ผู้ใช้สามารถจัดการโปรโมชันได้อย่างสะดวกและง่ายดายผ่านเมนูแบบ text-based ที่ใช้งานง่าย

```python
def manage_promotions_menu():
    while True:
        print("\n--- Manage Promotions ---")
        print("1) Add Promotion")
        print("2) Update Promotion")
        print("3) Delete Promotion")
        print("4) View Promotions")
        print("0) Back to Main Menu")
        choice = input("Choose option: ")
        if choice == '1':
            add_promotion()
        elif choice == '2':
            update_promotion()
        elif choice == '3':
            delete_promotion()
        elif choice == '4':
            view_promotions()
        elif choice == '0':
            break
        else:
            print("Invalid option")
```

ภาพที่ 4.21 แสดงตัวอย่างเมนูจัดการโปรโมชั่นใน console

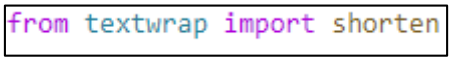
4.1.1 ฟังก์ชัน main_menu มีหน้าที่สำคัญในการทำหน้าที่เป็น เมนูหลัก ของระบบ Burger Shop Management โดยทำหน้าที่เป็นศูนย์กลางในการนำทางให้ผู้ใช้เข้าถึงส่วนงานย่อยต่างๆ ของระบบได้อย่างเป็นระเบียบ ฟังก์ชันนี้ทำงานอยู่ภายใน ลูป while True เพื่อให้เมนูแสดงตัวเลือกและพร้อมรับคำสั่งจากผู้ใช้ตลอดเวลา จนกว่าผู้ใช้จะเลือกออกจากระบบ (0) ภายในลูป ฟังก์ชันจะแสดงหัวข้อ "===== Burger Shop Management =====" พร้อมด้วยตัวเลือกการทำงานหลักที่ชัดเจน ได้แก่ การจัดการสินค้า (1), การจัดการราคา (2), การจัดการโปรโมชัน (3), การสร้างรายงาน (4), และการออกจากระบบ (0) จากนั้นจึงรอรับตัวเลือกจากผู้ใช้ (choice) เมื่อผู้ใช้กรอกตัวเลือก: หากเลือกตัวเลือก 1 ถึง 4 ฟังก์ชันจะเรียกใช้ฟังก์ชันย่อยที่เกี่ยวข้องทันที เช่น manage_products_menu() หรือ generate_report() เพื่อให้ผู้ใช้เข้าสู่ระบบงานย่อยนั้น ๆ หากผู้ใช้เลือกตัวเลือก 0 ฟังก์ชันจะแสดงข้อความ "Goodbye!" และใช้คำสั่ง break เพื่อหยุดการทำงานของลูปและสิ้นสุดโปรแกรม แต่หากกรอกตัวเลือกอื่นที่ไม่ตรงตามที่กำหนด ฟังก์ชันจะแสดงข้อความแจ้งเตือน "Invalid option" การออกแบบนี้ช่วยให้ผู้ใช้สามารถจัดการและเข้าถึงฟังก์ชันการทำงานหลักของระบบได้อย่างเป็นระเบียบและมีประสิทธิภาพผ่านเมนูแบบ text-based

```python
def main_menu():
    while True:
        print("\n===== Burger Shop Management =====")
        print("1) Manage Products")
        print("2) Manage Prices")
        print("3) Manage Promotions")
        print("4) Generate Report")
        print("0) Exit")
        print("==================================")
        choice = input("Choose option: ")
        if choice == '1':
            manage_products_menu()
        elif choice == '2':
            manage_prices_menu()
        elif choice == '3':
            manage_promotions_menu()
        elif choice == '4':
            generate_report()
        elif choice == '0':
            print("Goodbye!")
            break
        else:
            print("Invalid option")
```

ภาพที่ 4.22 แสดงตัวอย่างเมนูหลักใน console

4.1.2 ฟังก์ชัน generate_report() มีหน้าที่ในการสร้างรายงานสรุปข้อมูลสินค้า ราคาสินค้า และโปรโมชั่นในระบบ โดยรวมข้อมูลจากไฟล์ต่าง ๆ ได้แก่ ไฟล์สินค้า (PRODUCT_FILE), ไฟล์ราคา (PRICE_FILE) และไฟล์โปรโมชั่น (PROMOTION_FILE) มาประมวลผลและแสดงผลในรูปแบบตาราง ทั้งทางหน้าจอ Console และบันทึกลงในไฟล์รายงาน (REPORT_FILE) โดยเริ่มต้นจากการอ่านข้อมูลทั้งหมดจากแต่ละไฟล์ด้วยฟังก์ชัน read_all_records() ซึ่งจะนำข้อมูลในรูปแบบ binary มาถอดรหัสด้วย struct.unpack() เพื่อแปลงเป็นค่าที่สามารถใช้งานได้ จากนั้นจะสร้างพจนานุกรม promo_dict เพื่อแม็ปค่ารหัสโปรโมชั่น (promotion_id) กับชื่อโปรโมชั่น (promotion_name) เพื่อใช้ในการแสดงชื่อโปรโมชั่นที่ถูกต้องในรายงาน ต่อมา จะสร้างพจนานุกรม product_info โดยเชื่อมโยง product_id เข้ากับชื่อสินค้าและชื่อโปรโมชั่นที่เกี่ยวข้อง ทำให้สามารถนำข้อมูลสินค้าและโปรโมชั่นมาแสดงร่วมกันได้อย่างถูกต้อง หลังจากนั้น ฟังก์ชันจะสร้างส่วนหัวของตาราง (header) และวนลูปผ่านข้อมูลราคาสินค้า เพื่อสร้างแถวของตารางรายงาน โดยจะนำข้อมูลจากทั้ง PRICE_STRUCT และ product_info มาประกอบกันเป็นแถว ๆ พร้อมแสดงข้อมูล เช่น รหัสสินค้า ชื่อสินค้า ขนาด โปรโมชั่น ราคา จำนวนคงเหลือ และสถานะของสินค้า (พร้อมขายหรือไม่) โดยใช้ข้อความจัดรูปแบบให้อยู่ในลักษณะตาราง เพื่อให้อ่านง่าย เมื่อตารางข้อมูลทั้งหมดถูกจัดเตรียมแล้ว ฟังก์ชันจะแสดงรายงานบนหน้าจอ Console พร้อมทั้งเขียนรายงานลงในไฟล์ โดยภายในไฟล์จะประกอบด้วยหัวรายงาน วันที่สร้างรายงาน จำนวนสินค้าที่บันทึกไว้ และตารางข้อมูลที่จัดรูปแบบแล้ว นอกจากนี้ยังแนบข้อมูลบันทึกกิจกรรมล่าสุด 10 รายการจากไฟล์ log (LOG_FILE) หากมีไฟล์ดังกล่าวอยู่ และสุดท้ายจะเรียกใช้ฟังก์ชัน log_event() เพื่อบันทึกการสร้างรายงานไว้ในระบบอีกด้วย ฟังก์ชันนี้ช่วยให้ผู้ใช้สามารถดูภาพรวมของข้อมูลสินค้าทั้งหมดได้ในที่เดียว ทั้งผ่านหน้าจอและไฟล์รายงานอย่างเป็นระบบ สะดวกต่อการตรวจสอบสถานะสินค้า การกำหนดราคาขาย และติดตามการใช้งานย้อนหลัง

4.1.3 from textwrap import shorten คำสั่งนี้เป็นการนำเข้า (import) ฟังก์ชัน shorten จากโมดูล textwrap ซึ่งเป็นโมดูลมาตรฐานของ Python ที่ใช้สำหรับจัดการและจัดรูปแบบข้อความ โดย shorten() มีหน้าที่ในการ ตัดข้อความให้สั้นลงตามจำนวนความยาวที่กำหนด (width) และเติม ... หรือข้อความอื่น ๆ ที่กำหนดไว้หากข้อความถูกตัด ตัวอย่างเช่น ถ้ามีชื่อสินค้ายาวเกินไปสำหรับคอลัมน์ในตาราง ฟังก์ชัน shorten() จะตัดข้อความนั้นและเติม ... เพื่อให้พอดีกับความกว้างของคอลัมน์



ภาพที่ 4.23 ตัวอย่างการใช้ shorten

4.1.4 ฟังก์ชัน log_event(actor, action, status="OK", detail="") มีหน้าที่ในการบันทึกเหตุการณ์ต่าง ๆ ที่เกิดขึ้นในระบบลงในไฟล์บันทึก (LOG_FILE) โดยรับข้อมูลจากพารามิเตอร์ ได้แก่ ผู้กระทำ (actor), กิจกรรมที่ดำเนินการ (action), สถานะของเหตุการณ์ (status) ซึ่งมีค่าเริ่มต้นเป็น "OK" และรายละเอียดเพิ่มเติม (detail) ฟังก์ชันจะดึงวันเวลาปัจจุบันของระบบในรูปแบบ "ปี-เดือน-วัน ชั่วโมง:นาที:วินาที" ด้วยคำสั่ง datetime.now().strftime() แล้วจัดรูปแบบข้อความ log ให้อยู่ในรูปแบบ

```
def log_event(actor, action, status="OK", detail=""):
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    line = f"[{timestamp}] {actor} - {action}: {detail} -> {status}\n"
    with open(LOG_FILE, "a", encoding="utf-8") as f:
        f.write(line)
```

ภาพที่ 4-24 แสดงฟังก์ชัน log_event

4.1.5 ฟังก์ชัน pad_string(s, length) ทำหน้าที่แปลงสตริง s ให้เป็นข้อมูลไบต์ในรูปแบบ UTF-8 แล้วตัดข้อมูลไบต์ให้มีความยาวสูงสุดเท่ากับ length เพื่อป้องกันไม่ให้ข้อมูลยาวเกินกว่าที่กำหนด จากนั้นจะเติมไบต์ \x00 ซึ่งเป็นอักขระ null เข้าไปทางด้านขวาให้ครบความยาว length ด้วยเมธอด ljust() ของไบต์ เพื่อให้ได้ข้อมูลไบต์ที่มีความยาวคงที่ตามที่ต้องการ การทำเช่นนี้ช่วยให้ข้อมูลที่เก็บในรูปแบบไบนารีมีขนาดเท่ากันทุกระเบียน ซึ่งมีประโยชน์ในการอ่านและเขียนข้อมูลแบบโครงสร้าง (struct) ที่ต้องการความแม่นยำเรื่องขนาดข้อมูล

```
def pad_string(s, length):
    b = s.encode('utf-8')[:length]
    return b.ljust(length, b'\x00')
```

ภาพที่ 4.25 แสดงฟังก์ชัน pad_string

4.1.6 ฟังก์ชัน read_all_records(filename, record_size) มีหน้าที่อ่านข้อมูลทั้งหมดจากไฟล์ไบนารีที่ระบุด้วย filename โดยจะแบ่งข้อมูลออกเป็นระเบียนย่อย ๆ ที่มีขนาดเท่ากับ record_size ไบต์แต่ละระเบียน ฟังก์ชันจะตรวจสอบก่อนว่าไฟล์นั้นมีอยู่จริงหรือไม่ หากไม่พบไฟล์จะคืนค่าเป็นลิสต์ว่าง เพื่อป้องกันข้อผิดพลาด จากนั้นจะเปิดไฟล์ในโหมดอ่านไบนารี ('rb') และอ่านข้อมูลทั้งหมดเก็บไว้ในตัวแปร data แล้วทำการแบ่งข้อมูลนี้ออกเป็นส่วน ๆ โดยใช้ list comprehension ซึ่งวนลูปเริ่มจากตำแหน่ง 0 ไปจนถึงความยาวข้อมูลทีละ record_size ไบต์ ผลลัพธ์ที่ได้คือรายชื่อของไบต์ข้อมูลแต่ละระเบียนที่พร้อมนำไปถอดรหัสหรือใช้งานต่อไปในโปรแกรม

```
def read_all_records(filename, record_size):
    if not os.path.exists(filename):
        return []
    with open(filename, 'rb') as f:
        data = f.read()
    return [data[i:i+record_size] for i in range(0, len(data), record_size)]
```

ภาพที่ 4.26 แสดงฟังก์ชัน read_all_records

4.1.7 ฟังก์ชัน write_all_records(filename, record_size, records) มีหน้าที่เขียนข้อมูลระเบียนทั้งหมดที่อยู่ในลิสต์ records ลงไปในไฟล์ไบนารีที่ระบุด้วยชื่อ filename โดยจะเปิดไฟล์ในโหมดเขียนไบนารี ('wb') ซึ่งจะเขียนทับข้อมูลเดิมทั้งหมด จากนั้นจะทำการวนลูปเขียนแต่ละระเบียน rec ในลิสต์ records ลงในไฟล์ทีละรายการ ฟังก์ชันนี้ช่วยให้การบันทึกข้อมูลที่เป็นระเบียนขนาดคงที่ในรูปแบบไบนารีเป็นไปอย่างรวดเร็วและง่ายดาย โดยไม่ต้องจัดการกับการแบ่งไฟล์หรือการเปิดปิดไฟล์หลายครั้ง

```
def write_all_records(filename, record_size, records):
    with open(filename, 'wb') as f:
        for rec in records:
            f.write(rec)
```

ภาพที่ 4.27 แสดงฟังก์ชัน write_all_records

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

ระบบการจัดการร้านอาหารฟาสต์ฟู้ดที่พัฒนาขึ้นสามารถช่วยบริหารจัดการข้อมูลเมนูสินค้า การสั่งซื้อ การชำระเงิน และรายงานยอดขายได้อย่างมีประสิทธิภาพ โดยใช้การจัดเก็บข้อมูลในรูปแบบไฟล์ไบนารี พร้อมเมนูสำหรับเพิ่ม แก้ไข ลบ และแสดงผลข้อมูล ระบบยังรองรับการตรวจสอบสถานะของสินค้า เช่น เมนูที่มีอยู่ในสต็อก หมดสต็อก หรืออยู่ระหว่างโปรโมชั่น ตลอดจนสามารถสร้างรายงานสรุปยอดขาย รายการสินค้าขายดี และสถิติการสั่งซื้อในแต่ละช่วงเวลา ซึ่งช่วยให้การบริหารจัดการร้านสะดวก รวดเร็ว และลดความผิดพลาดจากการบันทึกข้อมูลด้วยมือ

5.2 ปัญหาและอุปสรรคในการทำงาน

ในการพัฒนาระบบการจัดการร้านอาหารฟาสต์ฟู้ด พบปัญหาหลักคือ ความซับซ้อนของการจัดการไฟล์ไบนารีที่ต้องอาศัยโครงสร้างข้อมูลแบบคงที่ (struct) ซึ่งอาจเกิดข้อผิดพลาดได้หากการเข้ารหัสหรือถอดรหัสไม่ถูกต้อง นอกจากนี้ยังพบข้อจำกัดด้านการแสดงผล เช่น ความยาวของชื่อเมนูหรือข้อมูลโปรโมชั่นที่ต้องถูกจำกัดตามขนาดของโครงสร้างข้อมูล อีกทั้งระบบยังไม่มีการเชื่อมต่อฐานข้อมูลจริง ทำให้ไม่สามารถรองรับข้อมูลจำนวนมาก หรือการเข้าถึงพร้อมกันจากหลายผู้ใช้งานได้อย่างเต็มรูปแบบ

5.3 ข้อเสนอแนะ

เพื่อให้ระบบสมบูรณ์และสามารถใช้งานได้จริงในอนาคต ควรมีการพัฒนาและปรับปรุงดังนี้

5.3.1 พัฒนาให้รองรับฐานข้อมูลเชิงสัมพันธ์ (Relational Database) เช่น MySQL หรือ SQLite เพื่อรองรับข้อมูลจำนวนมากและผู้ใช้งานหลายคนพร้อมกัน

5.3.2 เพิ่มฟังก์ชันการค้นหาและกรองข้อมูล เช่น ค้นหาเมนูตามชื่อ ประเภทอาหาร หรือช่วงราคาขาย

5.3.3 ปรับปรุงระบบการยืนยันตัวตนของผู้ใช้งาน แยกสิทธิ์ของผู้ดูแล พนักงานขาย และผู้จัดการร้าน เพื่อความปลอดภัยของข้อมูล

5.3.4 พัฒนาให้ระบบมีส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) หรือในรูปแบบเว็บแอปพลิเคชัน เพื่อให้ใช้งานได้สะดวกและเข้าถึงได้จากหลายอุปกรณ์ เช่น คอมพิวเตอร์ แท็บเล็ต และสมาร์ตโฟน

5.4 สิ่งที่ผู้จัดทำได้รับจากโครงการ

จากการพัฒนาโครงการระบบการจัดการร้านอาหารฟาสต์ฟู้ดครั้งนี้ ผู้จัดทำได้รับความรู้และประสบการณ์ในการออกแบบระบบบริหารจัดการข้อมูลสินค้า การพัฒนาโปรแกรมด้วยภาษา Python และการใช้โครงสร้างข้อมูลแบบ ไบนารี เพื่อจัดเก็บและเรียกใช้ข้อมูลอย่างมีประสิทธิภาพ ได้ฝึกฝนทักษะการคิดวิเคราะห์ การแก้ไขปัญหาเชิงตรรกะ และการออกแบบโครงสร้างข้อมูลให้เหมาะสมกับงานจริง นอกจากนี้ยังได้รับประสบการณ์ในการทำงานร่วมกันเป็นทีม การสื่อสาร การแบ่งหน้าที่รับผิดชอบ และการจัดการเวลาให้สอดคล้องกับแผนงาน ทำให้ผู้จัดทำมีความเข้าใจในกระบวนการพัฒนา ระบบซอฟต์แวร์จัดการร้านอาหาร มากยิ่งขึ้น และสามารถนำความรู้ที่ได้รับไปประยุกต์ใช้ในโครงการหรืองานจริงในอนาคตได้อย่างมีประสิทธิภาพ