



Project Term Assignment
โปรแกรม Library Management System
รายวิชา Computer Programming รหัสวิชา 060233115
กลุ่ม FourBytes Library

เสนอ

รศ.ดร. อนิราช มิ่งขวัญ

สมาชิก

นาย พิธินัย ชัยนเรศ รหัสนักศึกษา 6806022510165

นางสาว ศุภนิดา แซ่ซิ้ม รหัสนักศึกษา 6806022510246

นางสาว นุชนาถ สารพงษ์ รหัสนักศึกษา 6806022510327

นางสาว ธิตินญา โพธิ์ศรี รหัสนักศึกษา 6806022510084

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ วิทยาเขตปทุมธานี
คณะเทคโนโลยีและการจัดการอุตสาหกรรม
ภาควิชาเทคโนโลยีสารสนเทศ สาขาวิชา วิศวกรรมสารสนเทศและเครือข่าย

คำนำ

รายงานเล่มนี้จัดทำขึ้นเพื่อเสนอเนื้อหาเกี่ยวกับการพัฒนา โปรแกรมระบบห้องสมุด ซึ่งเป็นส่วนหนึ่งของการเรียนรายวิชา Computer Programming โดยมีจุดประสงค์เพื่อศึกษาการพัฒนาระบบสารสนเทศและการเขียนโปรแกรมที่สามารถนำไปใช้งานได้จริงภายในระบบประกอบด้วยฟังก์ชันที่สำคัญ ได้แก่

การสมัครสมาชิก สำหรับจัดเก็บข้อมูลผู้ใช้ใหม่

การยืม-คืนหนังสือ เพื่ออำนวยความสะดวกแก่สมาชิกและเจ้าหน้าที่

การลบและแก้ไขข้อมูล เพื่อให้ฐานข้อมูลมีความถูกต้องและทันสมัย

การจัดทำรายงานและโปรแกรมนี้มีเป้าหมายเพื่อเพิ่มประสิทธิภาพในการจัดการงานห้องสมุด ลดความซ้ำซ้อนของขั้นตอนการทำงาน และส่งเสริมทักษะการประยุกต์ใช้ความรู้ด้านการเขียนโปรแกรมและฐานข้อมูลของผู้จัดทำ

ผู้จัดทำหวังเป็นอย่างยิ่งว่ารายงานเล่มนี้จะเป็นประโยชน์ต่อผู้อ่าน ไม่ว่าจะเป็นนักศึกษา ครูอาจารย์ หรือผู้ที่สนใจในการพัฒนาระบบสารสนเทศ หากมีข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้เพื่อปรับปรุงแก้ไขในโอกาสต่อไป

ด้วยความเคารพ

ผู้จัดทำ

สารบัญเนื้อหา

	หน้า
คำนำ	ก
สารบัญ	ข
สารบัญ(ต่อ)	ค
สารบัญภาพ	ง
สารบัญภาพ(ต่อ)	จ
สารบัญภาพ(ต่อ)	ฉ
สารบัญตาราง	ช
บทที่ 1 บทนำ	1
1.1 วัตถุประสงค์ของโครงการ	1
1.2 ขอบเขตของโครงการ	1
1.3 ประโยชน์ที่คาดว่าจะได้รับ	2
บทที่ 2 ระบบยืม-คืนหนังสือห้องสมุด	3
2.1 ไฟล์ books.dat	3
2.2 ไฟล์ member.dat	4
2.3 ไฟล์ borrows.dat	6
2.4 ไฟล์ report.txt	8
บทที่ 3 การใช้งานระบบยืม-คืนหนังสือห้องสมุด	10
3.1 การใช้งานโปรแกรมระบบยืม – คืนห้องสมุด	10
3.2 การใช้งานโปรแกรมหนังสือเพิ่มข้อมูล	13
3.3 การใช้งานโปรแกรมแสดงข้อมูล	14
3.4 การใช้งานโปรแกรมแก้ไขข้อมูล	15
3.5 การใช้งานโปรแกรมลบข้อมูล	16
3.6 การใช้งานโปรแกรมยืม - คืนหนังสือ	17
บทที่ 4 ผลการดำเนินงาน	20
4.1 ผลลัพธ์ที่ได้จากการทำงานจริง	20
4.2 ฟังก์ชันเมนูจัดการหนังสือ	20
4.3 ฟังก์ชันเมนูจัดการสมาชิก	40
4.4 ฟังก์ชันเมนูยืม – คืนหนังสือ	42

4.5 view_borrow(filename="storage/borrow.dat")	51
4.6 ฟังก์ชัน load	52
4.7 ฟังก์ชัน generate_report()	55
4.8 ฟังก์ชัน fix_str	56
4.9 ฟังก์ชัน from_str	57
บทที่ 5 สรุปผล อภิปราย และข้อเสนอแนะ	58
5.1 สรุปผล	58
5.2 ปัญหาและอุปสรรค	58
5.3 ข้อเสนอแนะ	58
5.4 สิ่งที่ได้จัดทำได้รับในการพัฒนาโครงการ	58

สารบัญภาพ

	หน้า
ภาพที่ 2-1 แฟ้มรายการงานการยืม-คืน report.txt	8
ภาพที่ 3-1 ใช้งานฟังก์ชัน Boo	10
ภาพที่ 3-2 เมนู Book	10
ภาพที่ 3-3 การเลือกใช้งานฟังก์ชันของ Members	11
ภาพที่ 3-4 เมนูของ Members	11
ภาพที่ 3-5 การเลือกใช้งานฟังก์ชันของ Borrows	11
ภาพที่ 3-6 เมนูของ borrows	12
ภาพที่ 3-7 การเลือกใช้งานฟังก์ชันของ report	12
ภาพที่ 3-8 เมนูของ report	12
ภาพที่ 3-9 ออกจากโปรแกรมของ Exit	13
ภาพที่ 3-10 การเลือกใช้งานฟังก์ชัน Add Book	13
ภาพที่ 3.11 การเพิ่มหนังสือ	14
ภาพที่ 3-12 การเพิ่มข้อมูลสมาชิก	14
ภาพที่ 3-13 การเลือกใช้งานฟังก์ชัน View All Book	14
ภาพที่ 3.14 การเลือกใช้งานฟังก์ชัน View All Members	15
ภาพที่ 3.15 การเลือกใช้งานฟังก์ชัน Edit All Book	15
ภาพที่ 3.16 การเลือกใช้งานฟังก์ชัน Edit Manage Borrows	16
ภาพที่ 3.17 การเลือกใช้ฟังก์ชัน Delete Book	16
ภาพที่ 3.18 การเลือกใช้ฟังก์ชัน Delete Member	17
ภาพที่ 3.19 การเลือกใช้งานฟังก์ชัน Borrow Book	17
ภาพที่ 3.20 เลือกใช้ฟังก์ชัน Return Book	18
ภาพที่ 3.21 แสดงหน้ายืมหนังสือ	18
ภาพที่ 3.22 แสดงหน้า Back to Menu	19
ภาพที่ 4-1 ฟังก์ชัน menu_books สำหรับการจัดการข้อมูลหนังสือทั้งหมดในระบบ	21
ภาพที่ 4-2 ฟังก์ชัน _find_record_pos_by_id ใช้ค้นหาตำแหน่งของเรคคอร์ดภายในไฟล์	23
ภาพที่ 4-3 ฟังก์ชัน add_book เพิ่มข้อมูลหนังสือใหม่เข้าสู่ระบบ	24
ภาพที่ 4-4 การเปิดไฟล์ books.dat ด้วยโหมด "rb"	27

สารบัญภาพ(ต่อ)

	หน้า
ภาพที่ 4-5 การแสดงหัวตาราง เพื่อให้ข้อมูลแสดงผลอย่างเป็นระเบียบ	27
ภาพที่ 4-6 การอ่านข้อมูลจากไฟล์ เพื่ออ่านข้อมูลที่ละ record	28
ภาพที่ 4-7 การตรวจสอบสถานะการลบ เพื่อใช้ระบุสถานะ ถ้าค่าที่อ่านได้ (deleted)	28
ภาพที่ 4-8 การแปลงข้อมูลกลับจาก binary	29
ภาพที่ 4-9 การแสดงผลแต่ละ record	29
ภาพที่ 4-10 การแสดงผลลัพท์ เพื่อยืนยันว่าหนังสือถูกลบออกจากระบบแล้ว	32
ภาพที่ 4-11 ฟังก์ชัน book_exists_and_active	32
ภาพที่ 4-12 สิ้นสุดฟังก์ชันการทำงานต่างๆของการจัดการรายการหนังสือ	33
ภาพที่ 4-13 เริ่มฟังก์ชันการทำงานต่างๆของการจัดการรายการสมาชิก	33
ภาพที่ 4-14 สร้างรหัสสมาชิกใหม่โดยอัตโนมัติ (Auto Increment)	34
ภาพที่ 4-15 ค้นหาตำแหน่ง record ของสมาชิกในไฟล์	35
ภาพที่ 4-16 เพิ่มข้อมูลสมาชิกใหม่ลงในไฟล์ members.dat	36
ภาพที่ 4-17 แสดงรายการสมาชิกที่บันทึกอยู่ในไฟล์ members.dat	37
ภาพที่ 4-18 สำหรับแก้ไข (Update) ข้อมูลสมาชิกที่บันทึกอยู่ในไฟล์ members.dat	38
ภาพที่ 4-19 สำหรับลบข้อมูลสมาชิกจากไฟล์ members.dat	40
ภาพที่ 4-20 ฟังก์ชัน menu_members จัดการข้อมูลสมาชิกภายในระบบ	41
ภาพที่ 4-21 เมนูย่อยที่ใช้ควบคุมการทำงานเกี่ยวกับการ ยืมและคืนหนังสือ	43
ภาพที่ 4-22 การ import ฟังก์ชันให้ในไฟล์เตอร์ module เข้ามาใช้งานในไฟล์ปัจจุบัน	44
ภาพที่ 4-23 ตรวจสอบว่าสมาชิกที่มีรหัส (member_id)	44
ภาพที่ 4-24 ตรวจสอบว่าสมาชิกที่มีรหัส (member_id)	45
ภาพที่ 4-25 ค้นหารหัสสูงสุด (max ID) ของรายการยืม-คืน (borrow_id)	46
ภาพที่ 4-26 ค้นหารายการยืม-คืนที่มีรหัส (borrow_id)	47
ภาพที่ 4-27 ตรวจสอบว่าสมาชิกที่กำหนด (member_id) กำลังยืมหนังสือเล่มเดิม (book_id)	47
ภาพที่ 4-28 “ยืมหนังสือ” ใหม่ลงในไฟล์ borrows.dat	49
ภาพที่ 4-29 จัดการกระบวนการ “คืนหนังสือ” ในระบบ	50
ภาพที่ 4-30 แสดงข้อมูลการยืม-คืนหนังสือทั้งหมดจากไฟล์ borrows.dat	51

สารบัญภาพ(ต่อ)

	หน้า
ภาพที่ 4-31 สำหรับเรียกใช้ฟังก์ชันเกี่ยวกับระบบปฏิบัติการ	52
ภาพที่ 4-32 อ่านไฟล์ books.dat เก็บข้อมูลหนังสือ	53
ภาพที่ 4-33 อ่านไฟล์ members.dat เก็บข้อมูลสมาชิก	54
ภาพที่ 4-34 อ่านข้อมูลจากไฟล์ borrows.dat ซึ่งเก็บรายการการยืม-คืนหนังสือ	55
ภาพที่ 4-35 ฟังก์ชันส่วนเสริม Utility	56
ภาพที่ 4-36 ฟังก์ชัน fix_str	57
ภาพที่ 4-37 ฟังก์ชัน from_str	57

สารบัญตาราง

	หน้า
ตารางที่ 2-1 แฟ้มข้อมูลหนังสือ	3
ตารางที่ 2-2 แฟ้มข้อมูลสมาชิก	4
ตารางที่ 2-3 แฟ้มข้อมูลการยืม-คืน	6

บทที่ 1

บทนำ

1.1 วัตถุประสงค์ของโครงการ

1.1.1 เพื่อพัฒนาโปรแกรมระบบจัดการห้องสมุดที่สามารถยืม-คืนหนังสือได้อย่างมีประสิทธิภาพ

1.1.2 เพื่อฝึกทักษะการเขียนโปรแกรมด้วยภาษา Python และ การจัดการไฟล์ไบนารี

1.1.3 เพื่อศึกษาการใช้โมดูล struct ในการจัดเก็บข้อมูลแบบ Fixed - Length Record

1.1.4 เพื่อให้ผู้จัดทำเรียนรู้การออกแบบและพัฒนาระบบสารสนเทศตั้งแต่การวิเคราะห์ออกแบบ จนถึงการทดสอบระบบ

1.1.5 เพื่อฝึกการทำงานร่วมกันเป็นทีม และการแบ่งงานตามหน้าที่ CRUD (Create, Read, Update, Delete)

1.2 ขอบเขตของโครงการ

1.2.1 ระบบยืม-คืนหนังสือห้องสมุดจะมีฟังก์ชันพื้นฐานทั้งหมด 15 ฟังก์ชัน ได้แก่

1.2.1.1 เพิ่มหนังสือ

1.2.1.2 แก้ไขข้อมูลหนังสือ

1.2.1.3 ดูข้อมูลหนังสือ

1.2.1.4 ลบหนังสือ

1.2.1.5 กลับไปที่เมนูหลัก

1.2.1.6 เพิ่มสมาชิก

1.2.1.7 ลบสมาชิก

1.2.1.8 แก้ไขข้อมูลสมาชิก

1.2.1.9 แสดงรายชื่อสมาชิกทั้งหมด

1.2.1.10 ยืมหนังสือ

1.2.1.11 คืนหนังสือ

1.2.1.12 แสดงข้อมูลการยืม

1.2.1.13 แสดงข้อมูลการยืมที่ยังไม่คืน

1.2.1.14 เมนูการสร้างรายงานสรุปการยืม-คืนหนังสือ

1.2.1.15 เมนูออกจากการทำงานของระบบ

1.2.2 ระบบจะใช้ไฟล์ไบนารีจำนวน 3 ไฟล์ ได้แก่

1.2.2.1 *books.dat* เก็บข้อมูลหนังสือ

1.2.2.2 *members.dat* เก็บข้อมูลสมาชิก

1.2.2.3 *borrow.dat* เก็บข้อมูลการยืม-คืน

1.2.3 ระบบจะมีการสร้างไฟล์ข้อความ *report.txt* สำหรับสรุปการยืม-คืนและสถิติต่าง ๆ

ของระบบ

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ได้ระบบต้นแบบในการจัดการข้อมูลห้องสมุดที่สามารถใช้งานได้จริง

1.3.2 ช่วยให้ผู้ใช้งานสะดวกในการค้นหาและจัดการข้อมูลหนังสือและสมาชิก

1.3.3 ช่วยลดความผิดพลาดในการบันทึกข้อมูล และเพิ่มความถูกต้องในการยืม-คืน

1.3.4 ผู้จัดทำได้รับความรู้ในการทำงานกับไฟล์ไบนารีและการเขียนโปรแกรมที่ใช้

โครงสร้างข้อมูลแบบ Fixed-Length

1.3.5 ผู้จัดทำได้รับประสบการณ์การทำงานเป็นทีม และการแบ่งงานตามโมดูล (Add, Update, Delete, View, Report)

บทที่ 2

ระบบยืม-คืนหนังสือห้องสมุด

Group: FourBytes Library

Library System

2.1 ไฟล์ books.dat

ตารางที่ 2-1 แฟ้มข้อมูลหนังสือ

	ฟิลด์	ชนิด	ขนาด	ตัวอย่าง
1	book_id	int	4	1001
2	title	string	20	"Clean Architecture"
3	author	string	20	"Robert C. Martin"
4	publisher	string	20	"Pearson"
5	year_published	int	4	2018
6	total_copies	int	4	12
7	available_copies	int	4	8
8	price_thb	float	4	1200.00

2.1.1 book_id

เป็นฟิลด์ชนิดจำนวนเต็ม (int 4 bytes) ใช้ในการระบุรหัสเฉพาะของหนังสือแต่ละเล่ม เช่น 1001, 1002, 1003 เพื่อให้ระบบสามารถจัดการหนังสือจำนวนมากได้โดยไม่สับสน รหัสนี้ถือเป็น Primary Key ของแฟ้มและถูกใช้ในการอ้างอิงข้ามแฟ้ม เช่น ใน borrows.dat จะมีการใช้ book_id เพื่อเชื่อมโยงว่ามี การยืมหนังสือเล่มใด

2.1.2 title ชื่อหนังสือ

title เป็นฟิลด์ข้อความชนิด string ความยาวคงที่ 20 ตัวอักษร ใช้บันทึกชื่อเต็มของหนังสือ เช่น “Clean Architecture” หรือ “Python for Beginners” การจัดเก็บเป็นแบบ fixed-length ทำให้ โครงสร้างของแฟ้มมีความสม่ำเสมอ หากชื่อสั้นกว่าความยาวที่กำหนดจะถูกเติมด้วยช่องว่าง หรือถ้ายาวเกิน

จะถูกตัดทิ้ง การมีชื่อหนังสือในระบบเป็นสิ่งสำคัญเพราะเป็นข้อมูลที่ใช้จะมองหาเป็นอันดับแรกเมื่อต้องการยืมหรือค้นหาหนังสือ

2.1.3 author ชื่อผู้แต่ง

author เป็นฟิลด์ข้อความ string ความยาวคงที่ 20 ตัวอักษร ใช้บันทึกชื่อผู้แต่งของหนังสือ เช่น “Robert C. Martin” หรือ “Anan K.” การเก็บชื่อผู้แต่งมีความสำคัญเนื่องจากช่วยให้สามารถค้นหาหนังสือตามชื่อผู้เขียนได้ รวมถึงใช้เป็นข้อมูลประกอบการอ้างอิงทางวิชาการและในรายงานต่าง ๆ

2.1.4 publisher สำนักพิมพ์

publisher เป็นฟิลด์ข้อความ string ความยาวคงที่ 20 ตัวอักษร ใช้เก็บชื่อสำนักพิมพ์ เช่น “Pearson” หรือ “O’Reilly” ซึ่งบ่งบอกถึงแหล่งที่มาของหนังสือและสามารถใช้แยกหมวดหมู่หรือวิเคราะห์ข้อมูลตามผู้จัดพิมพ์ได้

2.1.5 year_published ปีที่พิมพ์

year_published เป็นฟิลด์ชนิดจำนวนเต็ม (int 4 bytes) ใช้เก็บปีที่หนังสือถูกตีพิมพ์ เช่น 2018 หรือ 2021 ข้อมูลนี้ช่วยให้ผู้ใช้สามารถตรวจสอบอายุของหนังสือและตัดสินใจ เลือกอ่านตามความใหม่หรือความเก่าได้

2.1.6 total_copies จำนวนเล่มทั้งหมด

total_copies เป็นฟิลด์ชนิดจำนวนเต็ม (int 4 bytes) ใช้เก็บจำนวนเล่มรวมของหนังสือเล่มนั้นในห้องสมุด เช่น 12 เล่ม ข้อมูลนี้สะท้อนจำนวนทรัพยากรจริงที่ห้องสมุดมีในคลัง และเป็นตัวชี้วัดความเพียงพอของหนังสือสำหรับผู้ใช้อย่างหลายคน

2.1.7 available_copies จำนวนเล่มที่สามารถให้ยืมได้

available_copies เป็นฟิลด์ชนิดจำนวนเต็ม (int 4 bytes) ใช้เก็บจำนวนเล่มที่ยังคงสามารถให้ยืมได้ในปัจจุบัน เช่น 8 เล่ม ค่าในฟิลด์นี้จะถูกปรับลดลงทุกครั้งที่มีการยืม และจะถูกปรับเพิ่มขึ้นเมื่อมีการคืน ช่วยให้ระบบสามารถบอกสถานะของหนังสือได้ทันทีว่าเหลือกี่เล่มที่ยังยืมได้

2.1.8 price_thb ราคาหนังสือ

price_thb เป็นฟิลด์ชนิดทศนิยม (float 4 bytes) ใช้เก็บราคาของหนังสือแต่ละเล่มเช่น 1200.00 บาท การบันทึกข้อมูลราคามีประโยชน์ในด้านการประเมินมูลค่าทรัพยากร รวมของห้องสมุด และสามารถประกอบการทำบัญชีหรือรายงานการเงินของห้องสมุดได้

2.2 ไฟล์ members.dat

ตารางที่ 2-2 เพิ่มข้อมูลสมาชิก

	ฟิลด์	ชนิด	ขนาด	ตัวอย่าง
1	member_id	int	4	5001
2	full_name	string	20	"Somchai Dee"
3	address	string	20	"99 Moo 2, Bangkok"
4	phone	string	10	"0812345678"
5	email	string	30	" somchai@example.com "
6	join_date	int	4	2020

2.2.1 member_id รหัสสมาชิก

member_id เป็นฟิลด์ชนิดจำนวนเต็ม (int 4 bytes) ใช้ระบุรหัสเฉพาะของสมาชิกแต่ละคน เช่น 5001, 5002 เพื่อไม่ให้ข้อมูลสมาชิกซ้ำกัน รหัสนี้ถือเป็น Primary Key ของแฟ้ม และถูกใช้ในการอ้างอิงร่วมกับแฟ้มการยืม-คืนเพื่อบันทึกว่าการยืมแต่ละครั้งเป็นของสมาชิกคนใด

2.2.2 full_name ชื่อ-นามสกุล

full_name เป็นฟิลด์ข้อความ string ความยาวคงที่ 20 ตัวอักษร ใช้เก็บชื่อจริงและนามสกุลของสมาชิก เช่น "Somchai Dee" หรือ "Mali Intira" การเก็บชื่อสมาชิกมีความสำคัญในการระบุตัวตนและแสดงผลในรายงานหรือหน้าจอโปรแกรม

2.2.3 address ที่อยู่

address เป็นฟิลด์ข้อความ string ความยาวคงที่ 20 ตัวอักษร ใช้เก็บที่อยู่ของสมาชิก เช่น "99 Moo 2, Bangkok" ฟิลด์นี้มีความสำคัญในด้านการติดต่อสมาชิกเพื่อแจ้งเตือนหรือ ใช้ยืนยันข้อมูลหากเกิดปัญหาเกี่ยวกับการยืม-คืน

2.2.4 phone เบอร์โทรศัพท์

phone เป็นฟิลด์ข้อความ string ความยาวคงที่ 10 ตัวอักษร ใช้เก็บหมายเลขโทรศัพท์ของสมาชิก เช่น "0812345678" การมีเบอร์โทรศัพท์ทำให้เจ้าหน้าที่ห้องสมุดสามารถติดต่อสมาชิกได้โดยตรง เช่น แจ้งเตือนเรื่องกำหนดวันคืนหนังสือ

2.2.5 email อีเมล

email เป็นฟิลด์ข้อความ string ความยาวคงที่ 30 ตัวอักษร ใช้เก็บอีเมลของสมาชิก เช่น "somchai@example.com" ฟิลด์นี้ช่วยเพิ่มช่องทางในการสื่อสาร โดยอาจใช้สำหรับ ส่งข้อมูลแจ้งเตือนการยืม-คืนหรือข้อมูลข่าวสารของห้องสมุด

2.2.6 join_date วันที่สมัคร

join_date เป็นฟิลด์จำนวนเต็ม (int 4 bytes) ใช้เก็บปีที่สมาชิกทำการสมัครเข้าร่วมใช้งานระบบ เช่น 2020, 2022 การเก็บข้อมูลนี้ช่วยให้สามารถตรวจสอบสถานการณ์เป็นสมาชิกได้ และอาจใช้วิเคราะห์พฤติกรรมของสมาชิกตามช่วงปีที่สมัคร

2.3 ไฟล์ borrows.dat

ตารางที่ 2-3 เพิ่มข้อมูลการยืม-คืน

#	ฟิลด์	ชนิด	ขนาด	ตัวอย่าง
1	borrow_id	int	4	90001
2	member_id	int	4	5001
3	book_id	int	4	1001
4	borrow_date	int	4	20250901
5	due_date	int	4	20250910
6	return_date	int	4	0
7	is_returned	int (1/0)	4	0

2.3.1 borrow_id รหัสการยืม

borrow_id เป็นฟิลด์ชนิดจำนวนเต็ม (int 4 bytes) ใช้ระบุรหัสเฉพาะของธุรกรรมการยืมแต่ละครั้ง เช่น 90001, 90002 รหัสนี้ช่วยให้ติดตามประวัติการยืมได้ว่ามีใครยืมเมื่อไหร่

2.3.2 member_id รหัสสมาชิก

member_id เป็นจำนวนเต็ม (int 4 bytes) ใช้อ้างอิงไปยังแฟ้ม members.dat เพื่อระบุว่าสมาชิกคนใดเป็นผู้ทำการยืม การเชื่อมโยงนี้ทำให้ระบบสามารถแสดงข้อมูลของผู้ยืมได้

ครบถ้วน เช่น ชื่อ ที่อยู่ เบอร์โทรศัพท์

2.3.3 book_id รหัสหนังสือ

book_id เป็นจำนวนเต็ม (int 4 bytes) ใช้อ้างอิงไปยังแฟ้ม books.dat เพื่อระบุว่าหนังสือเล่มใดถูกยืมออกไป การเชื่อมโยงนี้ทำให้ระบบรู้ว่าหนังสือใดอยู่ในสถานะถูกยืมและต้องลดค่า available_copies ลง

2.3.4 borrow_date วันที่ยืม

borrow_date เป็นจำนวนเต็ม (int 4 bytes) ใช้เก็บวันที่ทำการยืมจริงในรูปแบบ YYYYMMDD เช่น 20250901 (1 กันยายน 2025) ข้อมูลนี้ช่วยให้สามารถตรวจสอบประวัติการยืมย้อนหลังได้

2.3.5 due_date กำหนดวันคืน

due_date เป็นจำนวนเต็ม (int 4 bytes) ใช้เก็บวันที่ครบกำหนดคืนหนังสือ เช่น 20250910 (10 กันยายน 2025) 필드นี้มีประโยชน์ในการตรวจสอบว่าหนังสือเล่มนั้นเลยกำหนดคืนแล้วหรือยัง

2.3.6 return_date วันที่คืนจริง

return_date เป็นจำนวนเต็ม (int 4 bytes) ใช้เก็บวันที่คืนหนังสือจริงในรูปแบบ YYYYMMDD เช่น 20250908 หากยังไม่คืนจะเก็บค่าเป็น 0 ข้อมูลนี้ช่วยให้ติดตามได้ว่า สมาชิกมีการคืนหนังสือตรงเวลาหรือไม่

2.3.7 is_returned สถานะการคืน

is_returned เป็นจำนวนเต็ม (int 4 bytes) ที่มีค่าเพียง 1 หรือ 0 โดย 1 หมายถึงคืนแล้ว และ 0 หมายถึงยังไม่คืน การมีฟิลด์นี้ช่วยให้ระบบตรวจสอบสถานะปัจจุบันของหนังสือได้อย่างรวดเร็วโดยไม่ต้องเทียบกับวันที่คืน

2.4 ไฟล์ report.txt

```

1 =====
2 LIBRARY MANAGEMENT SYSTEM - BORROW/RETURN SUMMARY REPORT
3 =====
4 Generated At : 2025-10-02 20:47:09
5 Encoding : UTF-8
6 Content : Borrow/Return summary only
7 =====
8 BORROW/RETURN RECORDS
9 =====
10 BorrowID MemberID Member Name Phone BookID Book Title Borrow Due Return Status
11 -----
12 1 5 pihinal 075085134 6 Computer Engineering 2568-10-02 2568-10-10 2568-10-02 Returned
13 2 7 kittitach 046851368 7 Sql Sever 2568-09-09 2568-10-13 - Active
14 3 4 yuthapop 0136587426 7 Sql Sever 2568-09-11 2568-11-15 2568-10-02 Returned
15 4 3 chadchal 0987456321 10 Next.js 2568-10-01 2568-10-10 - Active
16 5 4 yuthapop 0136587426 8 Oracle 2568-09-09 2568-10-02 2568-10-02 Returned
17 -----
18 SUMMARY
19 -----
20 - Total Borrow Records : 5
21 - Currently Borrowed : 2
22 - Released : 3
23 =====
  
```

ภาพที่ 2-1แฟ้มรายงานการยืม-คืน report.txt

แฟ้มรายงานนี้เป็นไฟล์ข้อความที่ระบบสร้างขึ้นเพื่อแสดงสรุปข้อมูลการยืม-คืนของห้องสมุดในช่วงเวลาที่กำหนด รูปแบบจะประกอบด้วยส่วนหัวรายงาน (header), ข้อมูลการยืม-คืน (records) และส่วนสรุป (summary) เพื่อให้ผู้ใช้หรือผู้ดูแลระบบสามารถตรวจสอบภาพรวมได้ง่าย

2.4.1 header_text ส่วนหัวรายงาน

header_text เป็นฟิลด์ข้อความ (string 100 bytes) ใช้เก็บข้อความส่วนหัวของรายงาน เช่น “LIBRARY MANAGEMENT SYSTEM – BORROW/RETURN SUMMARY REPORT” ซึ่งแสดงอยู่ด้านบนสุดของรายงานในภาพ ตัวข้อความนี้บอกอย่างชัดเจนว่าไฟล์นี้คือรายงานประเภทใด และช่วยให้ผู้ใช้งานทราบทันทีว่าเป็นข้อมูลสรุปเกี่ยวกับการยืม-คืนหนังสือ ไม่ใช่ไฟล์ฐานข้อมูลหลัก

2.4.2 generated_at วันที่และเวลาที่สร้างรายงาน

generated_at เป็นฟิลด์ข้อความ (string 25 bytes) ใช้เก็บวันและเวลาที่รายงานถูกสร้างขึ้นจริงในรูปแบบ YYYY-MM-DD HH:MM เช่น “2025-10-02 20:47:09” ตามตัวอย่างในภาพ ฟิลด์นี้ช่วยให้

สามารถติดตามและอ้างอิงได้ว่าไฟล์รายงานนี้สร้างเมื่อใด เหมาะสำหรับเก็บเป็นหลักฐานหรือดูความเปลี่ยนแปลงตามช่วงเวลา

2.4.3 encoding การเข้ารหัสไฟล์

encoding เป็นฟิลด์ข้อความ (string 20 bytes) ใช้ระบุรูปแบบการเข้ารหัสไฟล์ เช่น “UTF-8” ซึ่งปรากฏในรายงานที่คุณส่งมา การกำหนด encoding ชัดเจนทำให้ไฟล์รายงานสามารถเปิดได้ถูกต้องในหลายระบบและรองรับภาษาไทย/อังกฤษได้

2.4.4 content_type เนื้อหารายงาน

ในภาพจะเห็นบรรทัด “Content : Borrow/Return summary only” ซึ่งถือเป็นฟิลด์ข้อความ (string 30 bytes) ที่ใช้บอกว่ารายงานนี้เป็นประเภทใด เช่น “Borrow/Return summary only” เพื่อให้ผู้ใช้งานเข้าใจได้ทันทีว่าข้อมูลที่แสดงเป็นสรุปการยืม-คืน ไม่ใช่สรุปหนังสือหรือสมาชิก

2.4.5 borrow_table_header หัวตาราง

borrow_table_header เป็นฟิลด์ข้อความ (string 80 bytes) ใช้เก็บหัวตารางที่ใช้แสดงข้อมูลการยืม-คืน เช่น “BorrowID | MemberID | Member Name | Phone | BookID | Book Title | Borrow | Due | Return | Status” ดังที่เห็นในรูป หัวตารางนี้ทำให้ผู้ใช้รู้ว่าแต่ละคอลัมน์หมายถึงข้อมูลอะไร และช่วยให้อ่านตารางได้ง่าย

2.4.6 borrow_records ข้อมูลการยืม-คืน

borrow_records เป็นฟิลด์ข้อความ (string 120*N bytes, N=จำนวนระเบียน) ใช้เก็บข้อมูลธุรกรรมการยืม-คืนแต่ละรายการในรูปแบบความยาวคงที่ (fixed-length) เช่นในภาพ 1 | 5 | pihinai | 0756985134 | 6 | Computer Engineering | 2568-10-02 | 2568-10-10 | 2568-10-02 | Returned ซึ่งบอก BorrowID, MemberID, ชื่อสมาชิก, เบอร์โทร, BookID, ชื่อหนังสือ, วันที่ยืม, วันที่กำหนดคืน, วันที่คืนจริง และสถานะปัจจุบัน (Returned/Active) ฟิลด์นี้ช่วยให้สามารถแสดงผลรายการทั้งหมดของการยืม-คืนได้อย่างเป็นระบบ

2.4.7 summary_section ส่วนสรุป

summary_section เป็นฟิลด์ข้อความ (string 150 bytes) ใช้เก็บข้อมูลสรุปของระบบ เช่น “Total Borrow Records : 5”, “Currently Borrowed : 2”, “Returned : 3” ดังตัวอย่าง ในภาพ ข้อมูลนี้ช่วยให้ผู้ใช้งานเห็นภาพรวมได้ทันทีว่ามีธุรกรรมทั้งหมดกี่รายการ กำลังถูกยืมอยู่กี่เล่ม และคืนแล้วกี่เล่ม

บทที่ 3

การใช้งานระบบ-คินหนังสือห้องสมุด

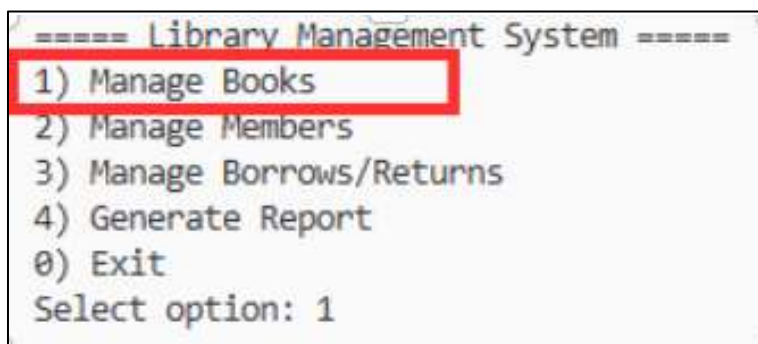
โปรแกรมการยืม - คินหนังสือห้องสมุดคือการช่วยการยืม - คินหนังสือให้สะดวกและช่วยจัดประเภทของหนังสือให้ดูง่ายมากยิ่งขึ้น

โปรแกรมการยืม - คินหนังสือห้องสมุดประกอบไปด้วย

สำหรับผู้ใช้งานโปรแกรม

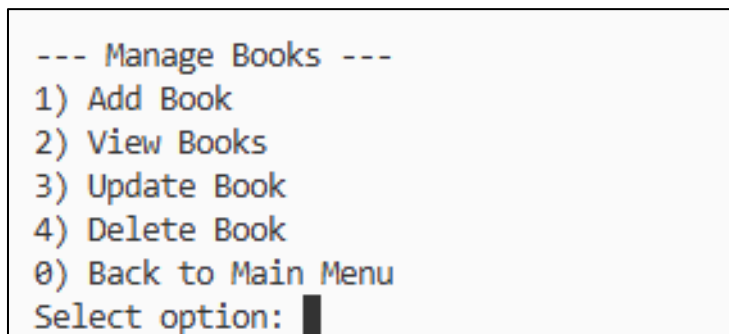
3.1 การใช้งานโปรแกรมระบบยืม - คินห้องสมุด

3.1.1 กรอกรหัสเลข 1 เพื่อเรียกใช้ฟังก์ชัน Manage Books สำหรับเข้าสู่เมนูจัดการหนังสือ เช่น เพิ่ม แก้ไข ลบ หรือดูข้อมูลหนังสือ



ภาพที่ 3-1 ใช้งานฟังก์ชัน Boo

3.1.2 เมื่อเข้าสู่เมนู Manage Book ขึ้นมาแล้วก็สามารถระบุเมนูที่ต้องการเลือกได้



ภาพที่ 3-2 เมนู Book

3.1.3 กรอกหมายเลข 2 เพื่อเรียกใช้เมนู Manage Members สำหรับเข้าสู่เมนูจัดการสมาชิก เช่น เพิ่มสมาชิก แก้ไขข้อมูล หรือลบ สมาชิก

```

===== Library Management System =====
1) Manage Books
2) Manage Members
3) Manage Borrows/Returns
4) Generate Report
0) Exit
Select option: 2

--- Manage Members ---
1) Add Member
2) View Members
3) Update Member
4) Delete Member
0) Back to Main Menu
Select option: █

```

ภาพที่ 3-3 การเลือกใช้งานฟังก์ชันของ Members

3.1.4 เมื่อเมนูฟังก์ชัน Manage Members ขึ้นมาแล้วจากนั้นก็สมารถระบุเมนูที่ต้องการเลือกได้

```

--- Manage Members ---
1) Add Member
2) View Members
3) Update Member
4) Delete Member
0) Back to Main Menu
Select option: █

```

ภาพที่ 3-4 เมนูของ Members

3.1.5 กรอกหมายเลข 3 เพื่อเรียกใช้ฟังก์ชัน Manage Borrows

```

===== Library Management System =====
1) Manage Books
2) Manage Members
3) Manage Borrows/Returns
4) Generate Report
0) Exit
Select option: 3

--- Manage Borrows/Returns ---
1) Add Borrow
2) Return Book
3) View Borrows
0) Back to Main Menu
Select option: █

```

ภาพที่ 3-5 การเลือกใช้งานฟังก์ชันของ Borrows

3.1.6 เมื่อเมนูฟังก์ชัน Manage Borrows จะสามารถระบุเมนูที่ต้องการเลือกได้

```

--- Manage Borrows/Returns ---
1) Add Borrow
2) Return Book
3) View Borrows
0) Back to Main Menu
Select option: █

```

ภาพที่ 3-6 เมนูของ borrows

3.1.7 กรอกหมายเลข 4 เพื่อเรียกใช้ฟังก์ชัน report เพื่อสามารถสร้างไฟล์รายงานได้

```

===== Library Management System =====
1) Manage Books
2) Manage Members
3) Manage Borrows/Returns
4) Generate Report
0) Exit
Select option: 4
Report generated: report/report_2025-10-04_21-30-02.txt

===== Library Management System =====
1) Manage Books
2) Manage Members
3) Manage Borrows/Returns
4) Generate Report
0) Exit
Select option: █

```

ภาพที่ 3-7 การเลือกใช้งานฟังก์ชันของ report

3.1.8 เมื่อเมนูฟังก์ชัน report จะสร้างไฟล์ report.txt



```

1  ===== Library Management System =====
2  1) Manage Books
3  2) Manage Members
4  3) Manage Borrows/Returns
5  4) Generate Report
6  0) Exit
7  Select option: 4
8  Report generated: report/report_2025-10-04_21-30-02.txt
9
10 ===== Library Management System =====
11 1) Manage Books
12 2) Manage Members
13 3) Manage Borrows/Returns
14 4) Generate Report
15 0) Exit
16 Select option: █

```

BorrowID	MemberID	Member Name	Phone	BookID	Book Title	Borrow Date	Due Date	Status
1	0	admin	070000000	0	Computer Engineering	2000-00-00	2000-00-00	Returned
2	0	admin	000000000	0	SQL Server	2000-00-00	2000-00-00	Returned
3	0	admin	000000000	0	SQL Server	2000-00-00	2000-00-00	Returned
4	0	admin	000000000	0	SQL Server	2000-00-00	2000-00-00	Returned
5	0	admin	000000000	0	SQL Server	2000-00-00	2000-00-00	Returned

```

17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

ภาพที่ 3-8 ตัวอย่างไฟล์ report.txt

3.1.9 กรอกหมายเลข 0 โปรแกรมจะแสดงข้อความ Exiting program และใช้ break เพื่อตัดลูปออกจากลูป จบการทำงานของโปรแกรม

```
===== Library Management System =====
1) Manage Books
2) Manage Members
3) Manage Borrows/Returns
4) Generate Report
0) Exit
Select option: 0
Exiting program...
PS D:\FourBytes-Library-main>
```

ภาพที่ 3-9 ออกจากโปรแกรมของ Exit

3.2 การใช้งานโปรแกรมเพิ่มข้อมูล

3.2.1 กรอกหมายเลข 1 เพื่อเพิ่มข้อมูลทั้งหมดของหนังสือที่มีในโปรแกรม เมื่อกดเลือกหมายเลข 1 จะปรากฏข้อการใส่รหัสหนังสือและจากนั้นใส่ข้อมูลในหัวข้อทั้งหมดดัง

```
===== Library Management System =====
1) Manage Books
2) Manage Members
3) Manage Borrows/Returns
4) Generate Report
0) Exit
Select option: 1

--- Manage Books ---
1) Add Book
2) View Books
3) Update Book
4) Delete Book
0) Back to Main Menu
Select option: 1
Assigned Book ID: 11
Enter Year Published: 2016
Enter Total Copies:
```

ภาพที่ 3-10 การเลือกใช้งานฟังก์ชัน Add Book

3.2.2 เมื่อเลือกหมายเลข 1 จะปรากฏหัวข้อการใส่รหัสหนังสือและจากนั้นใส่ข้อมูลในหัวข้อทั้งหมดภาพที่ 3-10

```
--- Manage Books ---
1) Add Book
2) View Books
3) Update Book
4) Delete Book
0) Back to Main Menu
Select option: 0
```

ภาพที่ 3.11 การเพิ่มหนังสือ

3.2.3 กรอกรหัสเลข 3 เพื่ออัปเดตข้อมูลทั้งหมดของผู้ใช้งานมีโปรแกรม

```

--- Manage Members ---
1) Add Member
2) View Members
3) Update Member
4) Delete Member
0) Back to Main Menu
Select option: 1
Assigned Member ID: 8
Enter Join Year:

```

ภาพที่ 3-12 การเพิ่มข้อมูลสมาชิก

3.3 การใช้งานโปรแกรมแสดงข้อมูล

3.3.1 กรอกรหัสเลข 2 เพื่อแสดงข้อมูลทั้งหมดของหนังสือที่มีโปรแกรม

```

--- Manage Books ---
1) Add Book
2) View Books
3) Update Book
4) Delete Book
0) Back to Main Menu
Select option: 2

```

ID	Title	Author	Publisher	Year	Total	Avail	Price
6	Computer Engineering	somchai chuchat	wangauksorn	2013	5	5	170.00
7	Sql Sever	somporn jeeraworn	wangruk	2000	7	6	69.00
8	Oracle	natchaporn tangthong	prayoshdee	2023	4	4	39.00
9	Smart screen	Hong Tung	Angkit	2019	7	7	49.00
10	Next.js	Kittichai	Wangrungthai	2000	8	7	89.00

ภาพที่ 3-13 การเลือกใช้งานฟังก์ชัน View All Book

3.3.2 กรอกหมายเลข 2 เพื่อแสดงข้อมูลทั้งหมดของผู้ใช้งานที่มีในโปรแกรม

Select option: 2						
ID	Full Name	Address	Phone	Email	Year	
3	chadchai	123/123 silkroad ayu	0987456321	chadchai@gmail.com	2525	
4	yuthapop	128/169 prachairoad	0136587426	yutthapop@gmail.com	2000	
5	pihinai	78/98 songroad bangk	0756985134	pithinai@gmail.com	2022	
6	sukchai	14/45 road mhapham r	0423698513	sukchai@gmail.com	2021	
7	kittitach	78/79 praram road ay	0468513684	kittihach@gmail.com	2020	

ภาพที่ 3.14 การเลือกใช้งานฟังก์ชัน View All Members

3.4 การใช้งานโปรแกรมแก้ไขข้อมูล

3.4.1 กรอกหมายเลข 3 เพื่อแก้ไขข้อมูลหนังสือที่มีในโปรแกรม

```

--- Manage Books ---
1) Add Book
2) View Books
3) Update Book
4) Delete Book
0) Back to Main Menu
Select option: 3
Enter Book ID to update: 9
Leave blank to keep old value.
New Title:

```

ภาพที่ 3.15 การเลือกใช้งานฟังก์ชัน Edit All Book

3.4.2 เมื่อกดเลือกหมายเลข 3 จะปรากฏหน้าจอแก้ไขข้อมูลการยืม - คืนหนังสือ

```

--- Manage Borrows/Returns ---
1) Add Borrow
2) Return Book
3) View Borrows
0) Back to Main Menu
Select option: 3
BorrowID MemberID BookID BorrowDate DueDate ReturnDate
-----
1 1 1 20251021 20251021 25551220
2 1 1 20251220 25551222 25561212
3 6 7 20250510 20250615 00000000

--- Manage Borrows/Returns ---
1) Add Borrow
2) Return Book
3) View Borrows
0) Back to Main Menu
Select option: 

```

ภาพที่ 3.16 การเลือกใช้งานฟังก์ชัน Edit Manage Borrows

3.5 การใช้งานโปรแกรมลบข้อมูล

3.5.1 เมื่อกดเลือกหมายเลข 4 ฟังก์ชัน Book เพื่อจะลบข้อมูลหนังสือที่มีอยู่แล้ว

```

--- Manage Books ---
1) Add Book
2) View Books
3) Update Book
4) Delete Book
0) Back to Main Menu
Select option: 4
Enter Book ID to delete: 9
Are you sure you want to delete Book ID 9? (y/n): y
Book deleted successfully.

```

ภาพที่ 3.17 การเลือกใช้ฟังก์ชัน Delete Book

3.5.2 เมื่อกดหมายเลข 4 ฟังก์ชันmember เพื่อลบmember

```

--- Manage Members ---
1) Add Member
2) View Members
3) Update Member
4) Delete Member
0) Back to Main Menu
Select option: 4
Enter Member ID to delete: █

```

ภาพที่ 3.18 การเลือกใช้ฟังก์ชัน Delete Member

3.6 การใช้งานโปรแกรมยืม – คืนหนังสือ

3.6.1 กรอกหมายเลข 1 เพื่อยืมหนังสือที่มีในโปรแกรม

```

--- Manage Borrows/Returns ---
1) Add Borrow
2) Return Book
3) View Borrows
0) Back to Main Menu
Select option: 1
Assigned Borrow ID: 4
Enter Member ID: █

```

ภาพที่ 3.19 การเลือกใช้งานฟังก์ชัน Borrow Book

3.6.2 กรอกหมายเลข 2 เพื่อคืนหนังสือในโปรแกรม

```

--- Manage Borrows/Returns ---
1) Add Borrow
2) Return Book
3) View Borrows
0) Back to Main Menu
Select option: 2
Enter Borrow ID to return: 

```

ภาพที่ 3.20 เลือกใช้ฟังก์ชัน Return Book

3.6.3 เมื่อกรอกหมายเลข 3 จะแสดงหน้าการยืมหนังสือทั้งหมด

```

--- Manage Borrows/Returns ---
1) Add Borrow
2) Return Book
3) View Borrows
0) Back to Main Menu
Select option: 3
BorrowID MemberID BookID BorrowDate DueDate ReturnDate
-----
1          1         1    20251021  20251021  25551220
2          1         1    20251220  25551222  25561212
3          6         7    20250510  20250615  00000000

--- Manage Borrows/Returns ---
1) Add Borrow
2) Return Book
3) View Borrows
0) Back to Main Menu
Select option: 

```

ภาพที่ 3.21 แสดงหน้ายืมหนังสือ

3.6.4 เมื่อกดหมายเลข 0 จะกลับไปหน้าจอเมนูหลัก

```
==== Library Management System ====
1) Manage Books
2) Manage Members
3) Manage Borrows/Returns
4) Generate Report
0) Exit
Select option: █
```

ภาพที่ 3.22 แสดงหน้า Back to Menu

บทที่ 4

ผลการดำเนินงาน

4.1 ผลลัพธ์ที่ได้จากการทำงานจริง

- 4.1.1 โปรแกรมสามารถ เพิ่ม/แก้ไข/ลบข้อมูลหนังสือและสมาชิก ได้ถูกต้อง
- 4.1.2 ระบบสามารถ ทำธุรกรรมการยืม-คืน และอัปเดตจำนวนเล่มคงเหลืออัตโนมัติ
- 4.1.3 มีการสร้างไฟล์ report.txt แสดงสรุปข้อมูลการยืม-คืน พร้อมจำนวนที่คืนแล้ว/ยังไม่คืน

4.2 ฟังก์ชันเมนูจัดการหนังสือ (Books)

4.2.1 ฟังก์ชัน menu_books

ฟังก์ชัน menu_books เป็นเมนูย่อยสำหรับการจัดการข้อมูลหนังสือทั้งหมดในระบบ ซึ่งประกอบด้วย การเพิ่ม ดู แก้ไข และลบข้อมูลหนังสือ ฟังก์ชันนี้ทำงานภายในลูป while True เพื่อให้ผู้ใช้สามารถดำเนินการซ้ำได้หลายครั้ง โดยไม่ต้องกลับบอกไปทีละเมนูหลักทุกครั้ง เมื่อเข้าสู่ฟังก์ชัน จะพิมพ์ข้อความหัวข้อ "--- Manage Books ---" เพื่อแจ้งผู้ใช้งานว่ากำลังอยู่ในเมนูจัดการหนังสือ จากนั้นจะแสดงรายการเมนูย่อยได้แก่

Add Book – สำหรับเพิ่มข้อมูลหนังสือใหม่ โดยเรียกใช้ฟังก์ชัน books.add_book() ซึ่งจะให้ผู้ใช้งานกรอกรายละเอียด เช่น รหัสหนังสือ ชื่อหนังสือ ผู้แต่ง สำนักพิมพ์ ปีที่พิมพ์ จำนวนเล่ม และราคาหนังสือ แล้วทำการบันทึกลงแฟ้มข้อมูล books.dat

View Books – สำหรับเรียกดูข้อมูลหนังสือทั้งหมดในระบบ โดยเรียกใช้ฟังก์ชัน books.view_books() เพื่อนำข้อมูลจากแฟ้ม books.dat มาแสดงผลในรูปแบบตาราง ซึ่งช่วยให้ผู้ใช้ตรวจสอบรายการหนังสือได้อย่างรวดเร็ว

Update Book – สำหรับการแก้ไขข้อมูลหนังสือที่มีอยู่ โดยเรียกใช้ฟังก์ชัน books.update_book() ฟังก์ชันนี้จะให้ผู้ใช้เลือกหนังสือจากรหัส (book_id) แล้วป้อนข้อมูลใหม่แทนค่าที่เดิม จากนั้นจึงเขียนทับข้อมูลเก่าลงในแฟ้ม books.dat

Delete Book – สำหรับการลบข้อมูลหนังสือออกจากระบบ โดยเรียกใช้ฟังก์ชัน books.delete_book() ฟังก์ชันจะถามยืนยันการลบก่อนเสมอเพื่อป้องกันข้อผิดพลาด จากนั้นจึงแก้ไขสถานะหนังสือหรือทำการลบ record ออกจากแฟ้ม books.dat

Back to Main Menu – ใช้สำหรับออกจากเมนูหนังสือ โดยการ break ออกจากลูป while True เพื่อกลับไปยังเมนูหลักของระบบ หากผู้ใช้ป้อนค่าที่ไม่ถูกต้อง โปรแกรมจะแสดงข้อความ "Invalid choice." เพื่อแจ้งเตือนและวนกลับไปยังเมนูย่อยนี้อีกครั้ง

หากผู้ใช้ป้อนค่าที่ไม่ถูกต้อง โปรแกรมจะแสดงข้อความ "Invalid choice." เพื่อแจ้งเตือนและวนกลับไปยังเมนูย่อยนี้อีกครั้ง

กล่าวโดยสรุป ฟังก์ชัน menu_books มีบทบาทเป็น ศูนย์ควบคุมการจัดการข้อมูลหนังสือ ซึ่งเชื่อมโยงผู้ใช้ไปยังฟังก์ชันย่อยต่าง ๆ ได้แก่ การเพิ่ม การดู การแก้ไข และการลบ ทำให้ผู้ใช้สามารถจัดการฐานข้อมูลหนังสือได้ครบวงจรในเมนูเดียว

```

1 def menu_books():
2     while True:
3         print("\n--- Manage Books ---")
4         print("1) Add Book")
5         print("2) View Books")
6         print("3) Update Book")
7         print("4) Delete Book")
8         print("0) Back to Main Menu")
9         c = input("Select option: ").strip()
10        if c == "1":
11            books.add_book()
12        elif c == "2":
13            books.view_books()
14        elif c == "3":
15            books.update_book()
16        elif c == "4":
17            books.delete_book()
18        elif c == "0":
19            break
20        else:
21            print("Invalid choice.")

```

ภาพที่ 4-1 ฟังก์ชัน menu_books สำหรับการจัดการข้อมูลหนังสือทั้งหมดในระบบ

4.2.2 ฟังก์ชัน _scan_max_id(filename)

ฟังก์ชัน _scan_max_id เป็นฟังก์ชันที่ใช้สำหรับค้นหาค่ารหัสสูงสุด (Maximum ID) ของหนังสือจากแฟ้มข้อมูลที่เก็บอยู่ในไฟล์ โดยฟังก์ชันจะเปิดไฟล์ในโหมดการอ่านแบบไบนารี (binary read) และอ่านข้อมูลเป็นช่วงขนาดคงที่ตามค่าที่กำหนดไว้ในตัวแปร BOOK_REC_SIZE ซึ่งแทนขนาดของหนึ่งเรคอร์ด จากนั้นจะทำการอ่าน 4 ไบต์แรกของแต่ละเรคอร์ดเพื่อแปลงกลับมาเป็นตัวเลขจำนวนเต็ม โดยถือว่าเป็นรหัสหนังสือ (book_id) แล้วทำการตรวจสอบว่ามีค่ามากกว่าค่าสูงสุดที่พบก่อนหน้านี้หรือไม่ หากมากกว่าให้แทนที่

ค่าปัจจุบันเป็นค่าที่พบใหม่จนกระทั่งอ่านจบทั้งไฟล์ กรณีที่ไฟล์ยังไม่ถูกสร้างขึ้น ฟังก์ชันจะคืนค่าเป็นศูนย์ เพื่อรองรับการสร้างรหัสเริ่มต้นได้อย่างถูกต้อง

4.2.3 ฟังก์ชัน `get_next_book_id(filename="storage/books.dat")`

ฟังก์ชัน `get_next_book_id` เป็นฟังก์ชันที่ใช้สำหรับสร้างรหัสหนังสือใหม่โดยอัตโนมัติ โดยจะเรียกใช้ฟังก์ชัน `_scan_max_id` เพื่อตรวจสอบค่ารหัสสูงสุดที่มีอยู่ในไฟล์ก่อนหน้า แล้วนำค่าที่ได้มาบวกเพิ่มอีกหนึ่ง เพื่อให้มั่นใจว่ารหัสใหม่ที่ได้จะไม่ซ้ำกับของเดิมที่อยู่ในระบบ ตัวอย่างเช่น หากพบว่ารหัสสูงสุดในไฟล์คือ 1005 ฟังก์ชันจะคืนค่าเป็น 1006 เพื่อใช้เป็นรหัสใหม่ในการเพิ่มหนังสือเล่มถัดไป

4.2.4 ฟังก์ชัน `_find_record_pos_by_id(target_id, filename)`

ฟังก์ชัน `_find_record_pos_by_id` เป็นฟังก์ชันที่ใช้ค้นหาตำแหน่งของเรคอร์ดภายในไฟล์จากรหัสหนังสือที่ต้องการ โดยฟังก์ชันจะเปิดไฟล์ในโหมดการอ่านแบบไบนารีและอ่านข้อมูลที่ละเรคอร์ดตามขนาดที่กำหนด จากนั้นจะดึง 4 ไบต์แรกของเรคอร์ดเพื่อแปลงกลับมาเป็นรหัสหนังสือ หากรหัสตรงกับค่าที่ผู้ใช้กำหนด (`target_id`) ฟังก์ชันจะส่งคืนค่าตำแหน่งไบต์ที่พบเรคอร์ดนั้นและข้อมูลเรคอร์ดเต็มทั้งชุด หากไม่พบข้อมูลหรือไฟล์ยังไม่มีการสร้าง ฟังก์ชันจะคืนค่าเป็น `(-1, None)` เพื่อบ่งบอกว่าไม่มีเรคอร์ดที่ค้นหาอยู่ในระบบ ฟังก์ชันนี้มีความสำคัญต่อการทำงานในส่วนของการแสดงผล แก้ไข หรือการลบข้อมูลหนังสือ เนื่องจากสามารถระบุตำแหน่งที่แน่นอนของเรคอร์ดในไฟล์ได้โดยตรง



```

1  def _scan_max_id(filename):
2      max_id = 0
3      try:
4          with open(filename, "rb") as f:
5              while chunk := f.read(BOOK_REC_SIZE):
6                  bid = from_i32(chunk[0:4])
7                  if bid > max_id:
8                      max_id = bid
9      except FileNotFoundError:
10         pass
11     return max_id
12
13 def get_next_book_id(filename="storage/books.dat"):
14     return _scan_max_id(filename) + 1
15
16 def _find_record_pos_by_id(target_id, filename):
17     try:
18         with open(filename, "rb") as f:
19             pos = 0
20             while chunk := f.read(BOOK_REC_SIZE):
21                 bid = from_i32(chunk[0:4])
22                 if bid == target_id:
23                     return pos, chunk
24                 pos += BOOK_REC_SIZE
25     except FileNotFoundError:
26         return -1, None
27     return -1, None

```

ภาพที่ 4-2 ฟังก์ชัน `_find_record_pos_by_id` ใช้ค้นหาตำแหน่งของเรคอร์ดภายในไฟล์

4.2.5 ฟังก์ชัน `add_book(filename="storage/books.dat")`

ฟังก์ชัน `add_book` เป็นฟังก์ชันสำหรับเพิ่มข้อมูลหนังสือใหม่เข้าสู่ระบบ โดยข้อมูลที่ได้จากผู้ใช้จะถูกตรวจสอบความถูกต้อง (data validation) ก่อนบันทึกลงไฟล์ไบนารี `books.dat` ซึ่งเป็นแฟ้มเก็บข้อมูลหลักของหนังสือในระบบ

```

1 def add_book(filename="storage/books.db"):
2     book_id = get_next_book_id(filename)
3     print(f"Assigned Book ID: {book_id}")
4
5     while True:
6         year_str = input("Enter Year Published: ").strip()
7         if year_str.isdigit() and int(year_str) > 0:
8             year = int(year_str)
9             break
10        print("Invalid year. Please enter a positive number.")
11
12    while True:
13        total_str = input("Enter Total Copies: ").strip()
14        if total_str.isdigit() and int(total_str) >= 0:
15            total = int(total_str)
16            break
17        print("Invalid total copies. Must be a non-negative integer.")
18
19    while True:
20        avail_str = input("Enter Available Copies: ").strip()
21        if avail_str.isdigit() and int(avail_str) >= 0:
22            avail = int(avail_str)
23            if avail <= total:
24                break
25            else:
26                print("Available copies cannot exceed total copies.")
27        else:
28            print("Invalid available copies. Must be a non-negative integer.")
29
30    while True:
31        price_str = input("Enter Price (THB): ").strip()
32        try:
33            price = float(price_str)
34            if price >= 0:
35                break
36            else:
37                print("Price must be non-negative.")
38        except ValueError:
39            print("Invalid price. Please enter a number.")
40
41    while True:
42        title = input("Enter Title: ").strip()
43        if title:
44            break
45        print("Title cannot be empty.")
46    while True:
47        author = input("Enter Author: ").strip()
48        if author:
49            break
50        print("Author cannot be empty.")
51    while True:
52        publisher = input("Enter Publisher: ").strip()
53        if publisher:
54            break
55
56

```

ภาพที่ 4-3 ฟังก์ชัน add_book เพิ่มข้อมูลหนังสือใหม่เข้าสู่ระบบ
การทำงานของฟังก์ชันสามารถอธิบายเป็นลำดับขั้นตอนดังนี้

4.2.5.1 การกำหนดรหัสหนังสือ (book_id)

เมื่อเรียกใช้งาน ฟังก์ชันจะเรียก get_next_book_id(filename) เพื่อตรวจสอบค่ารหัสสูงสุดที่มีอยู่ในไฟล์ก่อนหน้า และสร้างรหัสใหม่โดยการบวกเพิ่มอีกหนึ่ง ตัวอย่างเช่น หากรหัสสูงสุดคือ 1005 ฟังก์ชันจะแสดงผลว่า Assigned Book ID: 1006 และใช้รหัสนั้นเป็นรหัสของหนังสือใหม่โดยอัตโนมัติ

4.2.5.2 การกรอกปีที่พิมพ์ (year published)

ระบบจะวนลูปรอรับค่าปีจากผู้ใช้ โดยต้องเป็นข้อมูลตัวเลขบวก (isdigit()) และค่ามากกว่า 0) หากผู้ใช้กรอกผิด

เช่น กรอกตัวอักษร หรือกรอกเลขศูนย์ระบบจะแสดงข้อความ Invalid year. Please enter a positive number และวนกลับไปให้กรอกใหม่จนกว่าจะถูกต้อง

4.2.5.3 การกรอกจำนวนเล่มทั้งหมด (total copies)

ระบบจะให้กรอกจำนวนเล่มทั้งหมดของหนังสือ โดยค่าที่กรอกต้องเป็นจำนวนเต็มบวกหรือศูนย์ (≥ 0) หากผู้ใช้กรอกผิด เช่น เป็นข้อความหรือเป็นเลขติดลบ ระบบจะแจ้งเตือนว่า Invalid total copies. Must be a non-negative integer. และวนให้กรอกใหม่

4.2.5.4 การกรอกจำนวนเล่มที่สามารถยืมได้ (available copies)

ระบบจะให้กรอกจำนวนเล่มที่เหลืออยู่หรือพร้อมให้ยืม โดยค่าที่กรอกต้องเป็นจำนวนเต็มบวกหรือศูนย์เช่นเดียวกัน (≥ 0) และมีเงื่อนไขเพิ่มเติมคือ ต้องไม่เกินค่าของ total copies หากผู้ใช้กรอกมากกว่า ระบบจะแสดงข้อความเตือนว่า Available copies cannot exceed total copies. และวนกลับไปให้กรอกใหม่

4.2.5.5 การกรอกราคา (price)

ผู้ใช้งานกรอกราคาของหนังสือเป็นตัวเลขทศนิยม (float) โดยค่าที่รับต้องไม่เป็นลบ หากกรอกผิด เช่น ใส่ข้อความหรือใส่ค่าติดลบ ระบบจะแสดงข้อความ Invalid price. Please enter a number. หรือ Price must be non-negative. แล้ววนกลับไปกรอกใหม่

4.2.5.6 การกรอกชื่อหนังสือ (title)

ระบบจะตรวจสอบว่าชื่อหนังสือที่กรอกมาต้องไม่เป็นค่าว่าง หากปล่อยว่าง ระบบจะแสดงข้อความ Title cannot be empty. และวนให้กรอกใหม่จนกว่าจะถูกต้อง

4.2.5.7 การกรอกชื่อผู้แต่ง (author)

ระบบจะตรวจสอบว่าชื่อผู้แต่งต้องไม่เป็นค่าว่าง หากปล่อยว่างจะแสดงข้อความ Author cannot be empty. และวนให้กรอกใหม่

4.2.5.8 การกรอกสำนักพิมพ์ (publisher)

การกรอกสำนักพิมพ์ (publisher) ระบบจะตรวจสอบว่าข้อมูลสำนักพิมพ์ต้องไม่เป็นค่าว่าง หากผู้ใช้เว้นว่าง ระบบจะแสดงข้อความ Publisher cannot be empty. และวนกลับไปกรอกใหม่

4.2.6 ฟังก์ชัน view_books(filename="storage/books.dat")

ฟังก์ชัน view_books ใช้สำหรับแสดงข้อมูลหนังสือที่ถูกบันทึกไว้ในไฟล์ไบนารี โดยอ่านข้อมูลแบบ record-by-record และนำมาแสดงผลในรูปแบบตารางบนหน้าจอ โดยมีการตรวจสอบสถานะของหนังสือเพื่อแสดงเฉพาะรายการที่ยังไม่ถูกลบ (active records) เท่านั้น

การทำงานของฟังก์ชันสามารถอธิบายได้ดังนี้

4.2.6.1 การเปิดไฟล์

ฟังก์ชันพยายามเปิดไฟล์ `books.dat` ด้วยโหมด `"rb"` (read binary) หากไฟล์ไม่พบ (`FileNotFoundError`) ระบบจะแสดงข้อความ `"No book records found."` เพื่อแจ้งผู้ใช้งานว่ามีไฟล์ข้อมูล

4.2.6.2 การแสดงหัวตาราง

ก่อนอ่านข้อมูลแต่ละ record จะพิมพ์หัวตารางเป็นคอลัมน์ เช่น ID, Title, Author, Publisher, Year, Total, Avail, และ Price พร้อมกำหนดความกว้างของแต่ละคอลัมน์ด้วย `<width>` เพื่อให้ข้อมูลแสดงผลอย่างเป็นระเบียบ

4.2.6.3 การอ่านข้อมูลจากไฟล์

ฟังก์ชันใช้คำสั่ง `while chunk := f.read(BOOK_REC_SIZE):` เพื่ออ่านข้อมูลที่ละ record ตามขนาด `BOOK_REC_SIZE` ที่ถูกกำหนดไว้ในระบบ (เช่น 88 bytes) การอ่านจะวนต่อไปเรื่อย ๆ จนกว่าจะหมดไฟล์

4.2.6.4 การตรวจสอบสถานะการลบ (deleted flag)

ข้อมูลหนังสือแต่ละ record มีการเก็บค่าที่ตำแหน่งไบต์ 84–88 เพื่อใช้ระบุสถานะ ถ้าค่าที่อ่านได้ (deleted) ไม่เท่ากับ 0 หมายถึงข้อมูลนี้ถูกลบแล้ว ฟังก์ชันจะ `continue` ข้ามไป record ถัดไป ทำให้ผู้ใช้เห็นเฉพาะข้อมูลที่ยัง active เท่านั้น

4.2.6.5 การแปลงข้อมูลกลับจาก binary

ค่าที่อ่านมาแต่ละ field จะถูกแปลงจาก binary กลับเป็นชนิดข้อมูลที่ใช้งานได้

4.2.6.6 การแสดงผลแต่ละ record

ข้อมูลที่ถูกแปลงแล้วจะแสดงออกมาในรูปแบบตาราง โดยใช้ f-string จัดตำแหน่งคอลัมน์ เช่น `f'{book_id:<5} {title:<20} ... {price:<8}'` ทำให้ข้อมูลเรียงกันเป็นระเบียบเหมือนตาราง

4.2.6.7 การตรวจสอบว่ามีข้อมูลหรือไม่

ฟังก์ชันใช้ `flag found = False` เริ่มต้นไว้ หากพบ record ที่ยังไม่ถูกลบ จะตั้งค่า `found = True` หลังจากแสดงผลเสร็จ หากจบการอ่านไฟล์แล้วยังไม่มีข้อมูลถูกแสดง จะพิมพ์ข้อความ `"No active book records found."`

```

1 def view_books(filename="storage/books.dat"):
2     try:
3         with open(filename, "rb") as f:
4             print(f"{'ID':<5} {'Title':<20} {'Author':<20} {'Publisher':<20} {'Year':<6} {'Total':<6} {'Avail':<6} {'Price':<8}")
5             print("-"*90)
6             found = False
7             while chunk := f.read(BOOK_REC_SIZE):
8                 deleted = from_i32(chunk[84:88])
9                 if deleted != 0:
10                     continue
11                 book_id = from_i32(chunk[0:4])
12                 title = from_str(chunk[4:24])
13                 author = from_str(chunk[24:44])
14                 publisher = from_str(chunk[44:64])
15                 year = from_i32(chunk[64:68])
16                 total = from_i32(chunk[68:72])
17                 avail = from_i32(chunk[72:76])
18                 price = from_str(chunk[76:84])
19                 print(f"{book_id:<5} {title:<20} {author:<20} {publisher:<20} {year:<6} {total:<6} {avail:<6} {price:<8}")
20                 found = True
21             if not found:
22                 print("No active book records found.")
23     except FileNotFoundError:
24         print("No book records found.")
25

```

ภาพที่ 4-4 การเปิดไฟล์ books.dat ด้วยโหมด "rb"

```

1 def update_book(filename="storage/books.dat"):
2     try:
3         target_id = int(input("Enter Book ID to update: ").strip())
4     except ValueError:
5         print("Invalid Book ID.")
6         return
7
8     pos, chunk = _find_record_pos_by_id(target_id, filename)
9     if pos < 0 or chunk is None or from_i32(chunk[84:88]) != 0:
10         print("Book not found.")
11         return
12     print("Leave blank to keep old value.")
13
14     with open(filename,

```

ภาพที่ 4-5 การแสดงหัวตาราง เพื่อให้ข้อมูลแสดงผลอย่างเป็นระเบียบ



```

1  with open(filename, "r+b") as f:
2      new_title = input("New Title: ").strip()
3      if new_title:
4          f.seek(pos + 4); f.write(fix_str(new_title, 20))
5      new_author = input("New Author: ").strip()
6      if new_author:
7          f.seek(pos + 24); f.write(fix_str(new_author, 20))
8      new_pub = input("New Publisher: ").strip()
9      if new_pub:
10         f.seek(pos + 44); f.write(fix_str(new_pub, 20))
11
12     while True:
13         new_year = input("New Year Published: ").strip()
14         if not new_year:
15             break
16         if new_year.isdigit() and int(new_year) > 0:
17             f.seek(pos + 64); f.write(to_i32(int(new_year)))
18             break
19         print("Invalid year. Please enter a positive number or leave blank.")
20
21     cur_total = from_i32(chunk[68:72])
22     cur_avail = from_i32(chunk[72:76])

```

ภาพที่ 4-6 การอ่านข้อมูลจากไฟล์ เพื่ออ่านข้อมูลทีละ record



```

1  while True:
2      new_total = input("New Total Copies: ").strip()
3      if not new_total:
4          break
5      if new_total.isdigit() and int(new_total) >= 0:
6          t = int(new_total)
7          a = cur_avail
8          if a > t:
9              print("Invalid: available copies > total copies.")
10         else:
11             f.seek(pos + 68); f.write(to_i32(t))
12             break
13         print("Invalid total copies. Please enter non-negative integer or leave blank.")
14

```

ภาพที่ 4-7 การตรวจสอบสถานะการลบ เพื่อใช้ระบุสถานะ ถ้าค่าที่อ่านได้ (deleted) ไม่เท่ากับ 0

```

1  while True:
2      new_avail = input("New Available Copies: ").strip()
3      if not new_avail:
4          break
5      if new_avail.isdigit() and int(new_avail) >= 0:
6          a = int(new_avail)
7          t = int(new_total) if new_total else cur_total
8          if a > t:
9              print("Invalid: available copies > total copies.")
10             else:
11                 f.seek(pos + 72); f.write(to_i32(a))
12                 break
13             print("Invalid available copies. Please enter non-negative integer or leave blank.")
14

```

ภาพที่ 4-8 การแปลงข้อมูลกลับจาก binary

```

1  while True:
2      new_price = input("New Price (THB): ").strip()
3      if not new_price:
4          break
5      try:
6          p = float(new_price)
7          if p >= 0:
8              f.seek(pos + 76); f.write(fix_str(f"{p:.2f}", 8))
9              break
10             else:
11                 print("Price must be non-negative.")
12             except ValueError:
13                 print("Invalid price. Please enter a number or leave blank.")
14
15     print("Book updated successfully.")
16

```

ภาพที่ 4-9 การแสดงผลแต่ละ record

4.2.7 ฟังก์ชัน update_book(filename=e/books.dat)

ฟังก์ชัน update_book มีหน้าที่สำหรับปรับปรุง (Update) ข้อมูลของหนังสือที่บันทึกอยู่ในไฟล์ไบนารี books.dat โดยผู้ใช้สามารถระบุรหัสหนังสือ (Book ID) ที่ต้องการแก้ไข แล้วเลือกกรอกค่าใหม่ลงไป หากไม่กรอกอะไร (ปล่อยว่าง) ระบบจะเก็บค่าของเดิมไว้

รายละเอียดการทำงานมีดังนี้

4.2.7.1 การรับรหัสหนังสือจากผู้ใช้

ฟังก์ชันเริ่มต้นด้วยการรับ Book ID ผ่านคำสั่ง `input()` และพยายามแปลงเป็นตัวเลขจำนวนเต็ม (int) หากผู้ใช้กรอกไม่ถูกต้อง (ไม่ใช่ตัวเลข) จะเกิด `ValueError` ระบบจะแสดง "Invalid Book ID." และออกจากฟังก์ชันทันที

4.2.7.2 การค้นหา record ตำแหน่งหนังสือ

ใช้ฟังก์ชัน `_find_record_pos_by_id(target_id, filename)` เพื่อค้นหาตำแหน่ง record ของหนังสือที่ต้องการแก้ไข หากไม่พบ หรือ record นั้นถูกลบไปแล้ว (เช็คจาก `from_i32(chunk[84:88]) != 0`) ระบบจะแจ้ง "Book not found." แล้วหยุดทำงาน

4.2.7.3 การอัปเดตข้อมูลที่ละฟิลด์

เมื่อพบ record แล้ว ระบบจะเปิดไฟล์ด้วยโหมด "r+b" (อ่าน/เขียนแบบ binary) และให้ผู้ใช้กรอกค่าข้อมูลใหม่ในแต่ละฟิลด์ โดยมีกฎ คือ ถ้าผู้ใช้กรอกค่าใหม่ ระบบจะเขียนทับ (update) ถ้าปล่อยว่าง ระบบจะเก็บค่าของเดิมไว้

การอัปเดตข้อมูลแต่ละฟิลด์ทำงานดังนี้:

- Title, Author, Publisher: รับข้อความ string ใหม่ ตรวจสอบว่าไม่ว่าง (if `new_title:`) แล้วเขียนทับลงไฟล์ด้วย `fix_str(..., 20)` เพื่อให้เป็นข้อความความยาวคงที่ 20 bytes
- Year Published: รับข้อมูลตัวเลข ถ้าเว้นว่างจะข้ามไป แต่ถ้ากรอกต้องเป็นตัวเลขจำนวนเต็ม (`isdigit()`) และมากกว่า 0 จากนั้นบันทึกลงไฟล์ด้วย `to_i32()`
- Total Copies: รับจำนวนเล่มรวมใหม่ ต้องเป็นจำนวนเต็มไม่ติดลบ (`>=0`) และต้องไม่น้อยกว่า available copies ปัจจุบัน หากถูกต้องจะอัปเดตที่ตำแหน่งไบต์ 68–72
- Available Copies: รับจำนวนเล่มที่เหลืออยู่ใหม่ ต้องเป็นจำนวนเต็มไม่ติดลบ และไม่เกินจำนวน total copies ปัจจุบัน (หรือค่าที่เพิ่งแก้ไข) หากถูกต้องจะอัปเดตที่ตำแหน่ง 72–76
- Price: รับราคาใหม่ แปลงเป็น float ถ้ามากกว่าหรือเท่ากับ 0 จะบันทึกในรูปแบบ string ความยาว 8 bytes เช่น "1200.00"

4.2.7.4 การตรวจสอบและ validation

เกือบทุกฟิลด์มีการตรวจสอบความถูกต้องของข้อมูล เช่น ปีต้องเป็นค่าบวกจำนวนเล่มรวมต้องไม่น้อยกว่าจำนวนเล่มที่มีอยู่ราคาไม่เป็นค่าติดลบหากกรอกผิด ระบบจะถามซ้ำจนกว่าจะถูกต้องหรือปล่อยว่างเพื่อใช้ค่าของเดิม

4.2.7.5 การบันทึกและสรุปผล

เมื่อจบการแก้ไขครบทุกฟิลด์แล้ว ระบบจะแสดงข้อความ "Book updated successfully." เพื่อยืนยันว่าข้อมูลถูกแก้ไขสำเร็จแล้ว

4.2.8 ฟังก์ชัน delete_book(filename="storage/books.dat")

ฟังก์ชัน delete_book มีหน้าที่สำหรับลบข้อมูลหนังสือออกจากไฟล์ books.dat โดยไม่ลบข้อมูลจริง แต่ทำการเปลี่ยนสถานะ (flag) ของ record เป็น "Deleted" เพื่อป้องกันการสูญหายของข้อมูลอย่างถาวร และยังสามารถตรวจสอบย้อนหลังได้

4.2.8.1 การรับค่า Book ID

ระบบเริ่มต้นด้วยการให้ผู้ใช้งานกรอกหมายเลขหนังสือ (Book ID) ที่ต้องการลบ โดยใช้ input() และแปลงเป็นตัวเลขจำนวนเต็ม (int) หากกรอกไม่ถูกต้องจะเกิด ValueError และระบบจะแจ้ง "Invalid Book ID." พร้อมออกจากฟังก์ชันทันที

4.2.8.2 การค้นหา record ของหนังสือ

ใช้ฟังก์ชัน _find_record_pos_by_id(target_id, filename) เพื่อตรวจสอบว่า Book ID ที่กรอกมีอยู่จริงหรือไม่ หากไม่พบ ระบบจะแจ้ง "Book not found." และหยุดทำงาน

4.2.8.3 การยืนยันการลบ (Confirmation)

ก่อนลบจริง ระบบจะแสดงข้อความถามยืนยัน เช่น "Are you sure you want to delete Book ID 1001? (y/n):" หากผู้ใช้ตอบ y (yes) เท่านั้น จึงจะดำเนินการลบ หากตอบเป็นอย่างอื่น (เช่น n) ระบบจะยกเลิกและแจ้ง "Delete cancelled."

4.2.8.4 การเปลี่ยนสถานะในไฟล์

ถ้าผู้ใช้ยืนยันการลบ ระบบจะเปิดไฟล์ books.dat ในโหมด "r+b" (อ่านและเขียน binary) แล้วเลื่อนไปที่ byte ตำแหน่ง 84 ซึ่งถูกใช้เก็บสถานะของหนังสือ (status) จากนั้นเขียนค่า 1 (Deleted) ลงไปโดยใช้ฟังก์ชัน to_i32(1)

4.2.8.4 การแสดงผลลัพธ์

เมื่อการเขียนสำเร็จ ระบบจะแจ้ง "Book deleted successfully." เพื่อยืนยันว่าหนังสือถูกลบออกจากระบบแล้ว แต่ถ้าไม่พบไฟล์เลย (เกิด FileNotFoundError) จะขึ้นข้อความ "No book records found."

```

1 def delete_book(filename="storage/books.dat"):
2     try:
3         target_id = int(input("Enter Book ID to delete: ").strip())
4     except ValueError:
5         print("Invalid Book ID.")
6         return
7
8     pos, chunk = _find_record_pos_by_id(target_id, filename)
9     if pos < 0 or chunk is None:
10        print("Book not found.")
11        return
12
13    confirm = input(f"Are you sure you want to delete Book ID {target_id}? (y/n): ").strip().lower()
14    if confirm != "y":
15        print("Delete cancelled.")
16        return
17
18    try:
19        with open(filename, "r+b") as f:
20            f.seek(pos * 84)
21            f.write(to_i32(1))
22            print("Book deleted successfully.")
23    except FileNotFoundError:
24        print("No book records found.")
25

```

ภาพที่ 4-10 การแสดงผลลัพธ์ เพื่อยืนยันว่าหนังสือถูกลบออกจากระบบแล้ว

4.2.9 ฟังก์ชัน book_exists_and_active(book_id, filename="storage/books.dat")

ฟังก์ชัน book_exists_and_active มีหน้าที่ตรวจสอบว่าหนังสือที่มีรหัส (book_id) ที่กำหนดนั้นมีอยู่ในไฟล์ books.dat จริงหรือไม่ และมีสถานะเป็น Active หรือไม่

```

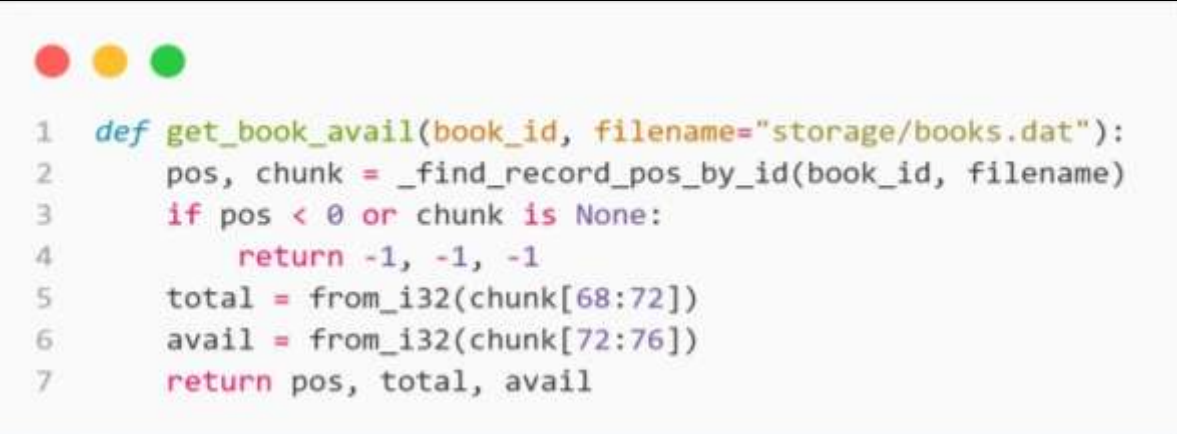
1 def book_exists_and_active(book_id, filename="storage/books.dat"):
2     pos, chunk = _find_record_pos_by_id(book_id, filename)
3     if pos < 0 or chunk is None:
4         return False
5     return from_i32(chunk[84:88]) == 0
6

```

ภาพที่ 4-11 ฟังก์ชัน book_exists_and_active

4.2.10 ฟังก์ชัน get_book_avail(book_id, filename="storage/books.dat")

ฟังก์ชัน get_book_avail มีหน้าที่ตรวจสอบจำนวนเล่มทั้งหมดและจำนวนเล่มที่ยังสามารถยืมได้ของหนังสือแต่ละเล่ม โดยอ้างอิงจาก book_id ที่ส่งเข้ามา ฟังก์ชันนี้ถูกใช้บ่อยในระบบยืม-คืนเพื่อควบคุมไม่ให้เกิดการยืมเกินจำนวนที่มีอยู่จริง



```

1 def get_book_avail(book_id, filename="storage/books.dat"):
2     pos, chunk = _find_record_pos_by_id(book_id, filename)
3     if pos < 0 or chunk is None:
4         return -1, -1, -1
5     total = from_i32(chunk[68:72])
6     avail = from_i32(chunk[72:76])
7     return pos, total, avail

```

ภาพที่ 4-12 สิ้นสุดฟังก์ชันการทำงานต่างๆของการจัดการรายการหนังสือ

4.2.11 ฟังก์ชัน set_book_avail_at_pos

ฟังก์ชัน set_book_avail_at_pos มีหน้าที่อัปเดตจำนวนเล่มที่ยังสามารถให้ยืมได้ (Available Copies) ของหนังสือ โดยอ้างอิงจากตำแหน่งของ record (pos) ที่ได้จากฟังก์ชันอื่น เช่น get_book_avail หรือ _find_record_pos_by_id



```

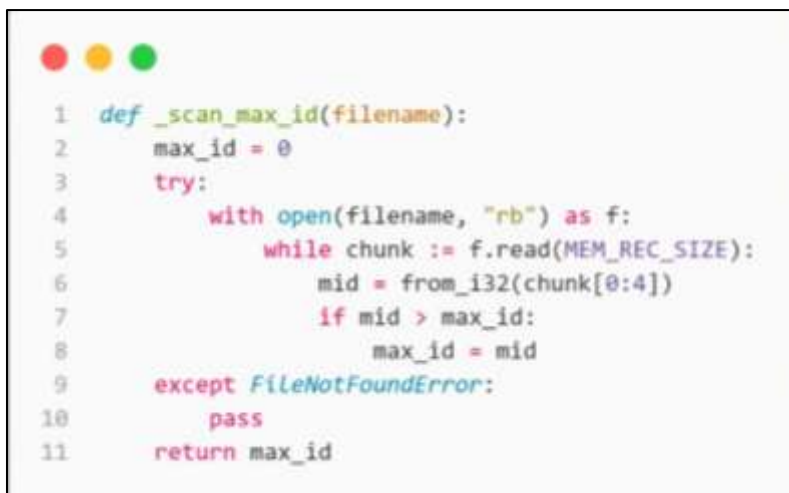
1 def set_book_avail_at_pos(pos, new_avail, filename="storage/books.dat"):
2     with open(filename, "r+b") as f:
3         f.seek(pos + 72)
4         f.write(to_i32(new_avail))

```

ภาพที่ 4-13 เริ่มฟังก์ชันการทำงานต่างๆของการจัดการรายการสมาชิก

ฟังก์ชันนี้มีหน้าที่ค้นหา รหัสสมาชิก (Member ID) ที่มีค่ามากที่สุด จากไฟล์ข้อมูลสมาชิก (members.dat) เพื่อใช้เป็นพื้นฐานในการกำหนดค่า ID ถัดไปเวลามีการเพิ่มสมาชิกใหม่ โดยวิธีการทำงานคือ ฟังก์ชันจะพยายามเปิดไฟล์ที่กำหนดด้วยโหมด "rb" (อ่านแบบ binary) และอ่านข้อมูลออกมาทีละ record โดยมีขนาด record ละ 92 ไบต์ (ตามที่กำหนดใน MEM_REC_SIZE) จากนั้นจะใช้คำสั่ง from_i32(chunk[0:4]) เพื่อแปลง 4 ไบต์แรกของ record ให้เป็นจำนวนเต็ม ซึ่งตรงกับรหัสสมาชิก (member_id) ของ record นั้น แล้วนำมาเปรียบเทียบกับค่าปัจจุบันของ max_id ถ้าค่าใหม่ที่ย่านมาใหญ่กว่า ระบบจะอัปเดต max_id ให้เป็นค่านั้น เมื่ออ่านจนครบทุก record จะคืนค่าของ max_id กลับไป (member_id) ของ record นั้น แล้วนำมาเปรียบเทียบกับค่าปัจจุบันของ max_id ถ้าค่าใหม่ที่ย่านมาใหญ่

กว่า ระบบจะอัปเดต max_id ให้เป็นค่านั้น เมื่ออ่านจนครบทุก record จะคืนค่าของ max_id กลับไป หากไฟล์ไม่ถูกพบ (เกิด FileNotFoundError) ฟังก์ชันจะไม่ทำอะไรและคืนค่า 0 แทน นั่นหมายถึงยังไม่มีข้อมูลสมาชิกในระบบ การทำงานลักษณะนี้ช่วยให้ทุกครั้งที่เพิ่มสมาชิกใหม่ จะสามารถกำหนด Member ID ที่ไม่ซ้ำและเพิ่มขึ้นเรื่อย ๆ เช่น ถ้า max_id ล่าสุดคือ 5003 สมาชิกใหม่ที่จะเพิ่มเข้ามาจะได้รหัส 5004 โดยอัตโนมัติ



```

1 def _scan_max_id(filename):
2     max_id = 0
3     try:
4         with open(filename, "rb") as f:
5             while chunk := f.read(MEM_REC_SIZE):
6                 mid = from_i32(chunk[0:4])
7                 if mid > max_id:
8                     max_id = mid
9     except FileNotFoundError:
10         pass
11     return max_id

```

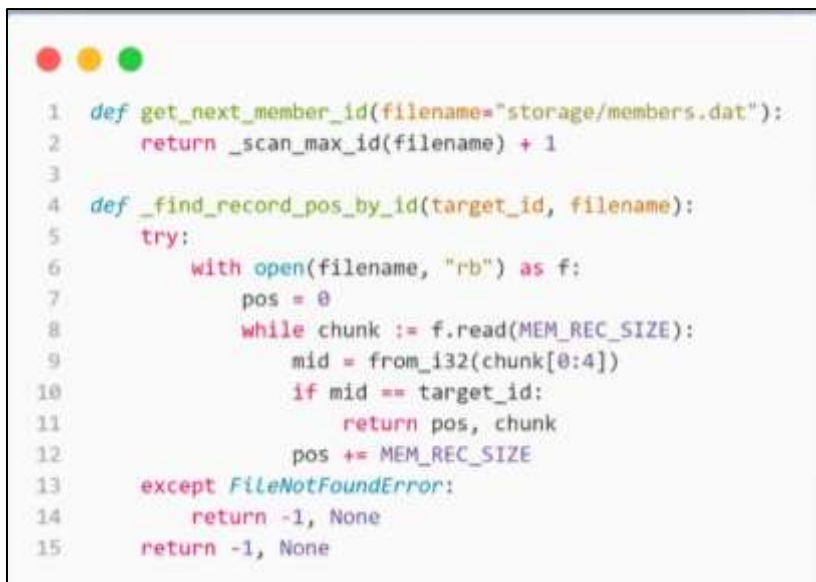
ภาพที่ 4-14 สร้างรหัสสมาชิกใหม่โดยอัตโนมัติ (Auto Increment)

4.2.12 ฟังก์ชัน get_next_member_id(filename="storage/members.dat")

ฟังก์ชันนี้ทำหน้าที่สร้างรหัสสมาชิกใหม่โดยอัตโนมัติ (Auto Increment) ทุกครั้งที่ต้องการเพิ่มสมาชิกเข้ามาในระบบ โดยมันจะเรียกใช้งาน _scan_max_id(filename) เพื่อตรวจสอบว่าในไฟล์ members.dat มีรหัสสมาชิกสูงสุดเท่าไร จากนั้นจะคืนค่าที่มากกว่าหนึ่ง (+ 1) กลับมาเป็นรหัสสมาชิกใหม่ ตัวอย่างเช่น ถ้าค่าสูงสุดที่มีอยู่คือ 5003 ฟังก์ชันนี้จะคืนค่า 5004 ให้เป็นรหัสสำหรับสมาชิกคนถัดไป วิธีการนี้ช่วยป้องกันการซ้ำกันของรหัส และทำให้ไม่ต้องอาศัยการกรอก ID ด้วยมือ

4.2.13 ฟังก์ชัน _find_record_pos_by_id(target_id, filename)

ฟังก์ชันนี้ใช้เพื่อค้นหาตำแหน่ง record ของสมาชิกในไฟล์ members.dat จากรหัสสมาชิกที่กำหนด (target_id) โดยระบบจะเปิดไฟล์ด้วยโหมด "rb" เพื่ออ่านแบบ binary และทำการวนลูปอ่านข้อมูลที่ละ record ขนาด 92 ไบต์ (ตามที่กำหนดไว้ใน MEM_REC_SIZE) จากนั้นจะใช้ from_i32(chunk[0:4]) แปลง 4 ไบต์แรกของ record เป็นตัวเลขจำนวนเต็มซึ่งก็คือ member_id ถ้า member_id ที่อ่านได้ตรงกับค่า target_id ที่กำหนด ฟังก์ชันจะคืนค่ากลับเป็น (pos, chunk) โดย pos คือ byte offset ของ record นั้นในไฟล์ และ chunk คือข้อมูลดิบของ record ที่อ่านมา แต่ถ้าอ่านจนหมดไฟล์แล้วยังไม่พบ หรือไฟล์ไม่มีอยู่เลย ฟังก์ชันจะคืนค่า (-1, None) เพื่อบอกว่าไม่พบข้อมูล



```

1 def get_next_member_id(filename="storage/members.dat"):
2     return _scan_max_id(filename) + 1
3
4 def _find_record_pos_by_id(target_id, filename):
5     try:
6         with open(filename, "rb") as f:
7             pos = 0
8             while chunk := f.read(MEM_REC_SIZE):
9                 mid = from_i32(chunk[0:4])
10                if mid == target_id:
11                    return pos, chunk
12                pos += MEM_REC_SIZE
13    except FileNotFoundError:
14        return -1, None
15    return -1, None

```

ภาพที่ 4-15 ค้นหาตำแหน่ง record ของสมาชิกในไฟล์

4.2.14 ฟังก์ชัน add_member(filename="storage/members.dat")

ฟังก์ชันนี้ทำหน้าที่เพิ่มข้อมูลสมาชิกใหม่ลงในไฟล์ members.dat โดยใช้แนวทางการทำงานกับไฟล์ไบนารีและระเบียบแบบความยาวคงที่ (Fixed-Length Record) เพื่อให้การจัดเก็บข้อมูลเป็นระบบและง่ายต่อการค้นหาในภายหลัง การทำงานเริ่มจากการเรียก get_next_member_id(filename) เพื่อกำหนดรหัสสมาชิกใหม่โดยอัตโนมัติ (Auto Increment) และแสดงให้ผู้ใช้ทราบว่ารหัสใดถูกกำหนดให้

จากนั้นฟังก์ชันจะวนถามข้อมูลสำคัญของสมาชิกพร้อมการตรวจสอบความถูกต้อง (Validation) แต่ละฟิลด์ ได้แก่ ปีที่เข้าร่วม (ต้องเป็นตัวเลขบวก), ชื่อ-นามสกุล (ต้องไม่ว่าง), ที่อยู่ (ต้องไม่ว่าง), หมายเลขโทรศัพท์ (ต้องเป็นตัวเลขความยาว 7-10 หลัก) และอีเมล (ต้องมีสัญลักษณ์ @ และ . อย่างน้อยหนึ่งตำแหน่งและความยาวเกิน 5 ตัวอักษร) หากผู้ใช้กรอกข้อมูลไม่ถูกต้อง ระบบจะวนถามซ้ำจนกว่าจะได้ค่าที่ถูกต้อง ฟิลด์ to_i32(0) ที่ท้ายสุด ใช้เป็น สถานะ (status) โดยค่าเริ่มต้นคือ 0 หมายถึง Active (ใช้งานได้)

จากนั้น record ที่ประกอบเสร็จจะถูกบันทึกลงไฟล์ members.dat ด้วยโหมด "ab" (append binary) เพื่อเพิ่มข้อมูลใหม่ต่อท้ายไฟล์โดยไม่กระทบข้อมูลเก่า สุดท้ายเมื่อการบันทึกเสร็จสิ้น ระบบจะแสดงข้อความ "Member added successfully." เพื่อยืนยันว่าการเพิ่มสมาชิกเสร็จสมบูรณ์แล้ว

```

1 def add_member(filename="storage/members.dat"):
2     member_id = get_next_member_id(filename)
3     print(f"Assigned Member ID: {member_id}")
4
5     while True:
6         year_str = input("Enter Join Year: ").strip()
7         if year_str.isdigit() and int(year_str) > 0:
8             join_year = int(year_str)
9             break
10        print("Invalid join year. Please enter a positive number.")
11
12    while True:
13        full_name = input("Enter Full Name: ").strip()
14        if full_name:
15            break
16        print("Full name cannot be empty.")
17
18    while True:
19        address = input("Enter Address: ").strip()
20        if address:
21            break
22        print("Address cannot be empty.")
23
24    while True:
25        phone = input("Enter Phone (10 digits): ").strip()
26        if phone.isdigit() and 7 <= len(phone) <= 10:
27            break
28        print("Invalid phone number. Must be digits, 7-10 characters.")
29
30    while True:
31        email = input("Enter Email: ").strip()
32        if "@" in email and "." in email and len(email) > 5:
33            break
34        print("Invalid email address.")
35
36    record = {
37        to_i32(member_id) +
38        fix_str(full_name, 30) +
39        fix_str(address, 20) +
40        fix_str(phone, 10) +
41        fix_str(email, 30) +
42        to_i32(join_year) +
43        to_i32(0)
44    }
45    with open(filename, "ab") as f:
46        f.write(record)
47    print("Member added successfully.")
48

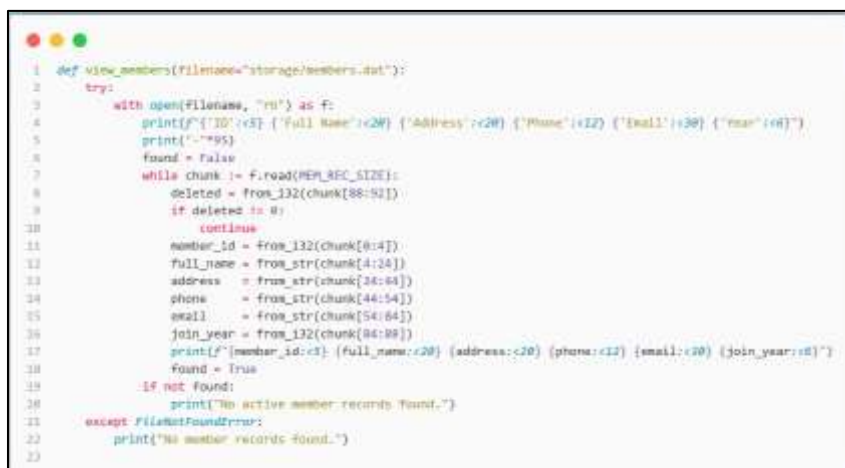
```

ภาพที่ 4-16 เพิ่มข้อมูลสมาชิกใหม่ลงในไฟล์ members.dat

4.2.15 ฟังก์ชัน view_members(filename="storage/members.dat")

ฟังก์ชันนี้ทำหน้าที่แสดงรายการสมาชิกที่บันทึกอยู่ในไฟล์ members.dat ออกมาในรูปแบบตารางที่อ่านง่าย โดยเน้นแสดงเฉพาะสมาชิกที่ยัง Active อยู่ (ไม่ถูกลบ) การทำงานเริ่มจากการพยายามเปิดไฟล์ members.dat ด้วยโหมด "rb" (read binary) ถ้าไฟล์ไม่มีอยู่ ระบบจะแจ้งข้อความ "No member records found." แล้วจบการทำงานทันที เมื่อเปิดไฟล์ได้สำเร็จ ฟังก์ชันจะแสดงหัวตารางที่จัดรูปแบบไว้ชัดเจน เช่น ID, Full Name, Address, Phone, Email, และ Year พร้อมขีดเส้นคั่นเพื่อความสวยงาม จากนั้นเริ่มวนอ่านข้อมูลทีละ record โดยใช้ขนาด record ที่กำหนดไว้ (MEM_REC_SIZE = 92 ไบต์) ทุกครั้งที่อ่านได้หนึ่ง record ระบบจะตรวจสอบฟิลด์สถานะที่เก็บอยู่ที่ bytearray88-92(deleted=from_i32(chunk[88:92])) ถ้าค่าไม่เท่ากับ 0 หมายความว่าสมาชิกคนนั้นถูกลบแล้ว (Deleted)

ระบบจะข้ามไปและไม่แสดงผล แต่ถ้าค่าเท่ากับ 0 หมายถึง Active อยู่ ข้อมูลจะถูกนำมาแปลงและแสดงผลต่อไป การแปลงข้อมูลแต่ละฟิลด์ทำโดยใช้ from_i32() สำหรับตัวเลข (เช่น member_id, join_year) และ from_str() สำหรับข้อความความยาวคงที่ (เช่น full_name, address, phone, email) เมื่อแปลงเสร็จแล้วจะพิมพ์ออกมาในรูปแบบตารางโดยจัดตำแหน่งคอลัมน์ให้สวยงามตรงกันทุก record เมื่อวนอ่านจนครบ หากไม่พบสมาชิกที่ Active เลย ระบบจะแสดง "No active member records found." เพื่อแจ้งให้ผู้ใช้ทราบ แต่ถ้ามีข้อมูลจะแสดงออกมาเต็มตารางทั้งหมด



```

1 def view_members(filename="storage/members.dat"):
2     try:
3         with open(filename, "rb") as f:
4             print(f"ID:<3> {<20>} {<20>} {<12>} {<20>} {<8>}")
5             print("-" * 95)
6             found = False
7             while chunk := f.read(PMEM_REC_SIZE):
8                 deleted = from_i32(chunk[88:92])
9                 if deleted != 0:
10                     continue
11                 member_id = from_i32(chunk[0:4])
12                 full_name = from_str(chunk[4:24])
13                 address = from_str(chunk[24:64])
14                 phone = from_str(chunk[64:84])
15                 email = from_str(chunk[84:104])
16                 join_year = from_i32(chunk[104:108])
17                 print(f"member_id:<3> {<20>} {<20>} {<12>} {<20>} {<8>}")
18                 found = True
19             if not found:
20                 print("No active member records found.")
21     except FileNotFoundError:
22         print("No member records found.")
23

```

ภาพที่ 4-17แสดงรายการสมาชิกที่บันทึกอยู่ในไฟล์ members.dat

4.2.16 ฟังก์ชัน update_member(filename="storage/members.dat")

ฟังก์ชันนี้มีหน้าที่สำหรับแก้ไข (Update) ข้อมูลสมาชิกที่บันทึกอยู่ในไฟล์ members.dat โดยผู้ใช้สามารถเลือกกรอกค่าข้อมูลใหม่สำหรับแต่ละฟิลด์ หากเว้นว่าง ระบบจะเก็บค่าของเดิมไว้ไม่เปลี่ยนแปลง การทำงานเริ่มจากการให้ผู้ใช้กรอกรหัสสมาชิก (Member ID) ที่ต้องการแก้ไข ถ้ากรอกไม่ถูกต้อง (ไม่ใช่ตัวเลข) ระบบจะแจ้งข้อความ "Invalid Member ID." และยกเลิกการทำงานทันที เมื่อได้ค่า Member ID ที่ถูกต้อง ฟังก์ชันจะเรียก _find_record_pos_by_id(target_id, filename) เพื่อตรวจสอบตำแหน่งของ record ในไฟล์ ถ้าไม่พบข้อมูล หรือสมาชิกนั้นถูกลบไปแล้ว (from_i32(chunk[88:92]) != 0) ระบบจะแจ้ง "Member not found." และไม่ดำเนินการต่อ

หากพบข้อมูล ระบบจะแสดงข้อความ "Leave blank to keep old value." เพื่อบอกผู้ใช้ว่าสามารถกด Enter เพื่อข้ามไม่แก้ไขฟิลด์นั้น ๆ ได้ จากนั้นจะเปิดไฟล์ members.dat ในโหมด "r+b" (อ่าน/เขียนแบบ binary) และเริ่มแก้ไขข้อมูลตามฟิลด์ต่าง ๆ ผู้ใช้สามารถกรอกค่าใหม่สำหรับ ชื่อ-นามสกุล (Full Name) และ ที่อยู่ (Address) ถ้ากรอกข้อมูล ระบบจะใช้ fix_str() แปลงเป็นข้อความความยาวคงที่และ

เขียนทับลงในไฟล์ที่ตำแหน่งที่เหมาะสม สำหรับ เบอร์โทรศัพท์ (Phone) ระบบจะตรวจสอบว่าค่าที่กรอกต้องเป็นตัวเลขทั้งหมดและมีความยาว 7-10 หลัก ถ้าไม่ถูกต้อง ระบบจะแจ้งเตือนและให้กรอกใหม่จนกว่าจะถูกหรือกด Enter เพื่อเข้าไปในส่วนของ อีเมล (Email) ระบบตรวจสอบว่ามีเครื่องหมาย @ และ . รวมถึงความยาวมากกว่า 5 ตัวอักษร ถ้าไม่ถูกต้องจะวนถามซ้ำเช่นกัน สำหรับ ปีที่เข้าร่วม (Join Year) ต้องเป็นตัวเลขจำนวนเต็มบวก หากกรอกผิดระบบก็จะวนถามซ้ำเช่นกัน เมื่อผู้ใช้กรอกข้อมูลเสร็จสิ้น ระบบจะบันทึกการแก้ไขลงไฟล์ จากนั้นแสดงข้อความ "Member updated successfully." เพื่อยืนยันว่าการแก้ไขเสร็จสมบูรณ์



```

1 def update_member(filename="storage/members.dat"):
2     try:
3         target_id = int(input("Enter Member ID to update: ").strip())
4     except ValueError:
5         print("Invalid Member ID.")
6     return

```

ภาพที่ 4-18 สำหรับแก้ไข (Update) ข้อมูลสมาชิกที่บันทึกอยู่ในไฟล์ members.dat

4.2.17 ฟังก์ชัน `delete_member(filename="storage/members.dat")`

ฟังก์ชันนี้มีหน้าที่สำหรับลบข้อมูลสมาชิกจากไฟล์ `members.dat` โดยการเปลี่ยนสถานะของ record แทนการลบข้อมูลถาวร การทำงานเริ่มจากการให้ผู้ใช้กรอกรหัสสมาชิก (Member ID) ที่ต้องการลบ ถ้ากรอกไม่ถูกต้อง (ไม่ใช่ตัวเลข) จะเกิด `ValueError` และระบบจะแจ้ง "Invalid Member ID." พร้อมยกเลิกการทำงาน

เมื่อได้ค่า Member ID ที่ถูกต้อง ระบบจะเรียก `_find_record_pos_by_id(target_id, filename)` เพื่อตรวจสอบว่ามี record ของสมาชิกนี้อยู่หรือไม่ ถ้าไม่พบ (`pos < 0` หรือ `chunk is None`) จะขึ้นข้อความ "Member not found." และจบการทำงานทันที

หากพบข้อมูล ระบบจะถามยืนยันการลบด้วยข้อความ "Are you sure you want to delete Member ID {target_id}? (y/n):" เพื่อป้องกันการลบโดยไม่ตั้งใจ ถ้าผู้ใช้ตอบเป็น `y` ระบบจึงจะดำเนินการต่อ แต่ถ้าตอบเป็นอย่างอื่น ระบบจะขึ้น "Delete cancelled." และไม่ลบข้อมูล

เมื่อผู้ใช้ยืนยันการลบ ระบบจะเปิดไฟล์ในโหมด "`r+b`" (อ่าน/เขียนแบบ binary) แล้วเลื่อนไปที่ `byte offset 88` ซึ่งเป็นตำแหน่งฟิลด์สถานะ (status) ของ record นั้น จากนั้นเขียนค่า `1` ลงไปโดยใช้ `to_i32(1)` เพื่อเปลี่ยนสถานะเป็น Deleted และเมื่อบันทึกสำเร็จ ระบบจะแจ้ง "Member deleted successfully." แต่ถ้าไฟล์ไม่พบตั้งแต่แรก จะขึ้นข้อความ "No member records found."

ฟังก์ชัน `member_exists_and_active(member_id, filename="storage/members.dat")`

ฟังก์ชันนี้มีหน้าที่ตรวจสอบว่าสมาชิกที่มีรหัส (member_id) ที่ส่งเข้ามานั้นมีอยู่จริงในไฟล์ `members.dat` และยังมีสถานะ Active หรือไม่ การทำงานเริ่มจากเรียก

`(_find_record_pos_by_id(member_id, filename))` เพื่อค้นหาตำแหน่งและข้อมูล record ของสมาชิก หากไม่พบ (`pos < 0` หรือ `chunk is None`) ฟังก์ชันจะคืนค่า `False` ทันที หมายถึงไม่มีสมาชิกคนนั้นในระบบ แต่ถ้าพบ record แล้ว ฟังก์ชันจะอ่านค่าฟิลด์สถานะที่เก็บไว้ใน `byte offset 88-92` ของ record โดยใช้ `from_i32()` เพื่อแปลงค่า 4 ไบต์เป็นจำนวนเต็ม ถ้าค่านี้เท่ากับ `0` หมายความว่าสมาชิกยัง Active อยู่ (ใช้งานได้) ฟังก์ชันจะคืนค่า `True` แต่ถ้าไม่ใช่ `0` (เช่นค่า = `1` หมายถึงถูกลบ/ไม่ใช้งานแล้ว) ฟังก์ชันจะคืนค่า `False` การทำงานลักษณะนี้ช่วยให้ระบบสามารถมั่นใจได้ว่า การทำงานต่อไป เช่น การยืมหนังสือ (`add_borrow`) หรือการคืนหนังสือ (`return_book`) จะไม่อ้างอิงไปยังสมาชิกที่ถูกลบหรือไม่อยู่ในระบบอีกต่อไป



```

1 def member_exists_and_active(member_id, filename="storage/members.dat"):
2     pos, chunk = _find_record_pos_by_id(member_id, filename)
3     if pos < 0 or chunk is None:
4         return False
5     return from_i32(chunk[88:92]) == 0

```

ภาพที่ 4-19 สำหรับลบข้อมูลสมาชิกจากไฟล์ members.dat

4.3 ฟังก์ชันเมนูจัดการสมาชิก (Members)

4.3.1 ฟังก์ชัน menu_members

ฟังก์ชัน menu_members มีหน้าที่จัดการข้อมูลสมาชิกภายในระบบ โดยรวมการทำงานที่เกี่ยวข้องกับการเพิ่มสมาชิกใหม่ การดูรายการสมาชิก การแก้ไขข้อมูล และการลบสมาชิกออกจากระบบ โครงสร้างของฟังก์ชันถูกเขียนในรูปแบบลูป while True เพื่อให้ผู้ใช้สามารถดำเนินการซ้ำได้หลายครั้ง จนกว่าจะเลือกกลับไปเมนูหลัก

เมื่อเข้าสู่ฟังก์ชัน ระบบจะแสดงหัวข้อ "--- Manage Members ---" พร้อมรายการเมนูย่อยดังนี้

Add Member – เรียกใช้ฟังก์ชัน members.add_member() สำหรับเพิ่มสมาชิกใหม่เข้าสู่ระบบ ผู้ใช้ต้องกรอกข้อมูล เช่น รหัสสมาชิก (member_id), ชื่อ-นามสกุล (full_name), ที่อยู่ (address), เบอร์โทรศัพท์ (phone), อีเมล (email) และวันที่สมัครสมาชิก (join_date) เมื่อป้อนครบแล้ว ระบบจะบันทึกข้อมูลลงไฟล์ members.dat

View Members – เรียกใช้ฟังก์ชัน members.view_members() เพื่อดูข้อมูลสมาชิกทั้งหมด ข้อมูลนี้จะถูกอ่านจากไฟล์ members.dat และแสดงออกมาในรูปแบบตาราง ประกอบด้วยรหัสสมาชิก ชื่อ ที่อยู่ เบอร์โทร และอีเมล ซึ่งช่วยให้ผู้ดูแลสามารถตรวจสอบรายการสมาชิกได้อย่างสะดวก

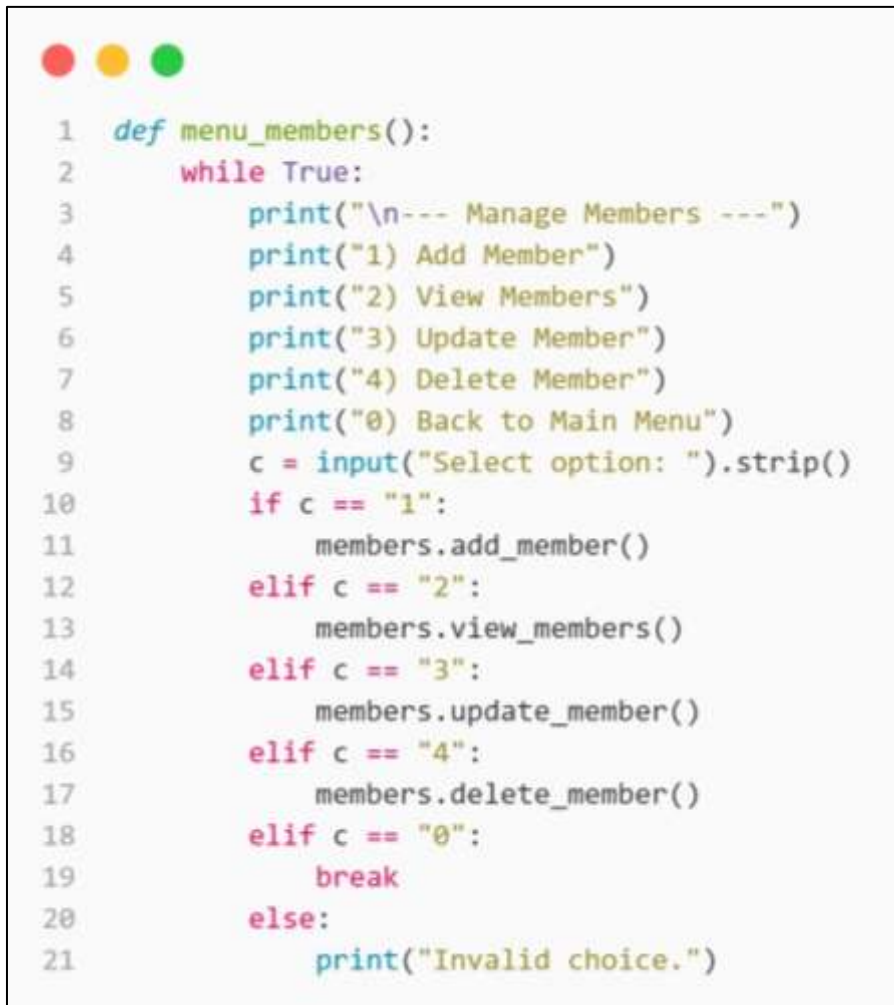
Update Member – เรียกใช้ฟังก์ชัน members.update_member() สำหรับแก้ไขข้อมูลสมาชิกที่มีอยู่เดิม เช่น เปลี่ยนที่อยู่หรือเบอร์โทรศัพท์ โดยระบบจะให้ผู้ใช้ป้อนรหัสสมาชิก (member_id) ที่ต้องการแก้ไข แล้วจึงป้อนข้อมูลใหม่ จากนั้นเขียนทับข้อมูลเดิมลงในไฟล์ members.dat

Delete Member – เรียกใช้ฟังก์ชัน members.delete_member() สำหรับลบข้อมูลสมาชิกออกจากระบบ การลบนี้มักมีการถามยืนยันก่อนเสมอเพื่อป้องกันข้อผิดพลาด ข้อมูลที่ถูกลบอาจตั้งสถานะเป็น "Deleted" หรือถูกลบออกจากไฟล์โดยตรง

Back to Main Menu – เมื่อเลือกตัวเลือกนี้ ฟังก์ชันจะใช้คำสั่ง break เพื่อตัดการทำงานของลูปและย้อนกลับไปเมนูหลักของระบบ

ถ้าผู้ใช้ป้อนค่าที่ไม่ตรงกับตัวเลือกที่กำหนด ฟังก์ชันจะแสดงข้อความ "Invalid choice." เพื่อแจ้งเตือนว่าป้อนข้อมูลไม่ถูกต้อง

กล่าวโดยสรุป ฟังก์ชัน menu_members มีบทบาทเป็น ศูนย์รวมการจัดการสมาชิก ที่ครอบคลุมทั้งการเพิ่ม ดู แก้ไข และลบข้อมูล ทำให้ผู้ดูแลระบบสามารถจัดการฐานข้อมูลสมาชิกได้ครบวงจร



```

1  def menu_members():
2      while True:
3          print("\n--- Manage Members ---")
4          print("1) Add Member")
5          print("2) View Members")
6          print("3) Update Member")
7          print("4) Delete Member")
8          print("0) Back to Main Menu")
9          c = input("Select option: ").strip()
10         if c == "1":
11             members.add_member()
12         elif c == "2":
13             members.view_members()
14         elif c == "3":
15             members.update_member()
16         elif c == "4":
17             members.delete_member()
18         elif c == "0":
19             break
20         else:
21             print("Invalid choice.")

```

ภาพที่ 4-20 ฟังก์ชัน menu_members จัดการข้อมูลสมาชิกภายในระบบ

4.4 ฟังก์ชันเมนูยืม-คืนหนังสือ (Borrows/Returns)

4.4.1 ฟังก์ชัน menu_borrows

ฟังก์ชัน menu_borrows เป็นเมนูย่อยที่ใช้ควบคุมการทำงานเกี่ยวกับการ ยืมและคืนหนังสือ ของสมาชิกในห้องสมุด โดยออกแบบให้ทำงานในลักษณะวนลูป while True เพื่อให้ผู้ใช้สามารถทำรายการต่อเนื่องได้หลายครั้งจนกว่าจะเลือกกลับไปเมนูหลัก

เมื่อเข้าสู่ฟังก์ชัน ระบบจะแสดงหัวข้อ "--- Manage Borrows/Returns ---" พร้อมเมนูย่อย ดังนี้

Add Borrow – เรียกใช้ borrows.add_borrow() สำหรับบันทึกการยืมหนังสือใหม่ ผู้ใช้จะต้องระบุรหัสสมาชิก (member_id) และรหัสหนังสือ (book_id) จากนั้นระบบจะตรวจสอบว่า สมาชิกยังไม่เกินสิทธิ์การยืม

หนังสือมีจำนวนเล่มว่างอยู่ (available_copies > 0) หากผ่านเงื่อนไข ระบบจะบันทึกรายการยืมลงไฟล์ borrows.dat และอัปเดตจำนวนเล่มว่างใน books.dat ให้อัตโนมัติ

Return Book – เรียกใช้ borrows.return_book() เพื่อบันทึกการคืนหนังสือ โดยระบบจะตรวจสอบว่าในไฟล์ borrows.dat มีรายการที่ยังไม่ถูกคืน (is_returned = 0) เมื่อผู้ใช้ป้อนข้อมูลรหัสการยืมหรือรหัสหนังสือ ระบบจะบันทึกวันที่คืน (return_date) กำหนดสถานะการคืน (is_returned = 1) และอัปเดตจำนวนเล่มว่างใน books.dat

View Borrows – เรียกใช้ borrows.view_borrows() เพื่อแสดงรายการยืม-คืนทั้งหมด โดยจะแสดงข้อมูล เช่น รหัสยืม (borrow_id), ชื่อสมาชิก, รหัสหนังสือ, วันที่ยืม, วันที่ครบกำหนด และสถานะการคืน (Active/Returned) ทำให้ผู้ดูแลสามารถติดตามการยืม-คืนได้ง่ายขึ้น

Back to Main Menu – เลือกตัวเลือกนี้เพื่อออกจากลูปและย้อนกลับไปยังเมนูหลักของระบบ หากผู้ใช้กรอกตัวเลือกไม่ตรงกับที่มี ระบบจะแสดงข้อความ "Invalid choice." เพื่อแจ้งข้อผิดพลาด



```

1  def menu_borrows():
2      while True:
3          print("\n--- Manage Borrows/Returns ---")
4          print("1) Add Borrow")
5          print("2) Return Book")
6          print("3) View Borrows")
7          print("0) Back to Main Menu")
8          c = input("Select option: ").strip()
9          if c == "1":
10             borrows.add_borrow()
11          elif c == "2":
12             borrows.return_book()
13          elif c == "3":
14             borrows.view_borrows()
15          elif c == "0":
16             break
17          else:
18             print("Invalid choice.")

```

ภาพที่ 4-21 เมนูย่อยที่ใช้ควบคุมการทำงานเกี่ยวกับการ ยืมและคืนหนังสือ


from module.utils import fix_str, from_str, to_i32, from_i32 บรรทัดนี้คือการ import ฟังก์ชัน จากไฟล์ utils.py ที่อยู่ในโฟลเดอร์ module เข้ามาใช้งานในไฟล์ปัจจุบัน ฟังก์ชันเหล่านี้ช่วยจัดการเรื่อง การเข้ารหัส/ถอดรหัสข้อมูล ระหว่าง string ↔ bytes และ integer ↔ bytes เพื่อให้สามารถบันทึกลง binary file ได้อย่างถูกต้อง

fix_str(text, length) → แปลงข้อความ (string) เป็น bytes ความยาวคงที่ (fixed-length) โดยตัดหรือเติมค่า \x00 ให้ครบตามความยาวที่กำหนด ใช้เวลาจะบันทึกข้อความลงไฟล์ binary

from_str(b) → แปลงข้อมูล bytes ที่อ่านออกมาจากไฟล์ ให้กลับมาเป็นข้อความ (string) โดยตัด \x00 หรือ space ด้านหลังออก

to_i32(n) → แปลงตัวเลข (int) ให้เป็น bytes ขนาด 4 byte (32 bit, little-endian) เพื่อเขียนเก็บลงไฟล์ binary from_i32(b) → แปลงข้อมูล bytes 4 byte กลับมาเป็นตัวเลข (int) เวลาที่อ่านไฟล์ binary

from_i32(b) → แปลงข้อมูล bytes 4 byte กลับมาเป็นตัวเลข (int) เวลาที่อ่านไฟล์ binary



```
1 from module.utils import fix_str, from_str, to_i32, from_i32
2 BOOK_REC_SIZE = 88
```

ภาพที่ 4-22 การ import ฟังก์ชันให้ในโฟลเดอร์ module เข้ามาใช้งานในไฟล์ปัจจุบัน

4.4.2 ฟังก์ชัน member_exists_and_active("storage/members.dat")

ฟังก์ชันนี้มีหน้าที่ตรวจสอบว่าสมาชิกที่มีรหัส (member_id) ที่ส่งเข้ามานั้นมีอยู่จริงในไฟล์ members.dat และยังมีสถานะ Active หรือไม่ การทำงานเริ่มจากเรียก

_find_record_pos_by_id(member_id, filename) เพื่อค้นหาตำแหน่งและข้อมูล record ของสมาชิก หากไม่พบ (pos < 0 หรือ chunk is None) ฟังก์ชันจะคืนค่า False ทันที หมายถึงไม่มีสมาชิกคนนี้ในระบบ แต่ถ้าพบ record แล้ว ฟังก์ชันจะอ่านค่าฟิลด์สถานะที่เก็บไว้ใน byte offset 88-92 ของ record โดยใช้ from_i32() เพื่อแปลงค่า 4 ไบต์เป็นจำนวนเต็ม ถ้าค่านี้เท่ากับ 0 หมายความว่าสมาชิกยัง Active อยู่ (ใช้งานได้) ฟังก์ชันจะคืนค่า True แต่ถ้าไม่ใช่ 0 (เช่นค่า = 1 หมายถึงถูกลบ/ไม่ใช้งานแล้ว) ฟังก์ชันจะคืนค่า False การทำงานลักษณะนี้ช่วยให้ระบบสามารถมั่นใจได้ว่า การทำงานต่อไป เช่น การยืมหนังสือ (add_borrow) หรือการคืนหนังสือ (return_book) จะไม่อ้างอิงไปยังสมาชิกที่ถูกลบหรือไม่อยู่ในระบบอีกต่อไป



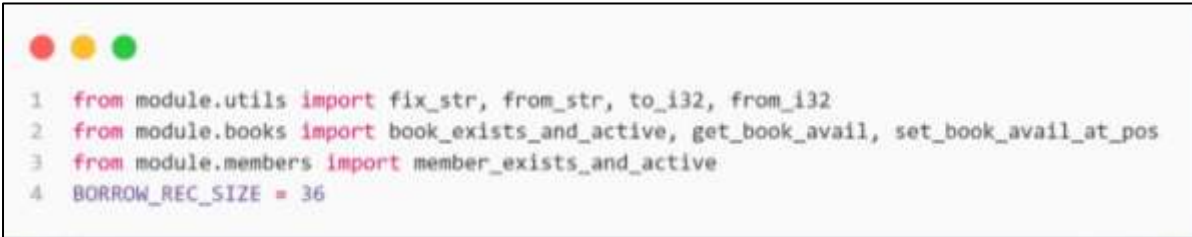
```
1 def member_exists_and_active(member_id, filename="storage/members.dat"):
2     pos, chunk = _find_record_pos_by_id(member_id, filename)
3     if pos < 0 or chunk is None:
4         return False
5     return from_i32(chunk[88:92]) == 0
```

ภาพที่ 4-23 ตรวจสอบว่าสมาชิกที่มีรหัส (member_id)

โค้ดส่วนนี้เป็นการเตรียมโมดูลที่เอาไว้จัดการการยืม-คืนหนังสือ โดยเริ่มจากการ import ฟังก์ชันพื้นฐานจาก utils อย่าง fix_str, from_str, to_i32, และ from_i32 ซึ่งมีหน้าที่แปลงข้อมูลข้อความและตัวเลขไปมาในรูปแบบ fixed-length record เพื่อให้บันทึกลงไฟล์ไบนารีได้อย่างสม่ำเสมอ ต่อมา มีการ import ฟังก์ชันจาก books เช่น book_exists_and_active, get_book_avail, และ set_book_avail_at_pos ซึ่งทั้งหมดนี้จะช่วยตรวจสอบว่าสามารถยืมหนังสือได้หรือไม่ รวมถึงปรับลดหรือเพิ่มจำนวน

เล่มคงเหลือเมื่อมีการยืมหรือคืนหนังสือ นอกจากนี้ยังมีการ import member_exists_and_active จาก members เพื่อยืนยันว่าสมาชิกที่กำลังจะทำการยืมอยู่จริงและยังใช้งานได้

สุดท้ายมีการกำหนด BORROW_REC_SIZE = 36 เพื่อระบุว่าข้อมูลการยืม-คืนแต่ละ record จะมีขนาดตายตัวที่ 36 ไบต์ในไฟล์ borrows.dat ทำให้การอ่านและเขียนข้อมูลสามารถทำได้อย่างตรงตำแหน่งและมีประสิทธิภาพ รวม ๆ แล้วโค้ดนี้จึงเป็นเหมือนฐานรากของระบบยืม-คืน ที่เตรียมทุกอย่างไว้พร้อม ทั้งเครื่องมือจัดการข้อมูล การตรวจสอบความถูกต้องของหนังสือและสมาชิก และการกำหนดโครงสร้าง record ชัดเจน เพื่อให้ฟังก์ชันที่จะสร้างต่อจากนี้ เช่น add_borrow() หรือ return_book() ทำงานได้อย่างสมบูรณ์และน่าเชื่อถือ



```

1 from module.utils import fix_str, from_str, to_i32, from_i32
2 from module.books import book_exists_and_active, get_book_avail, set_book_avail_at_pos
3 from module.members import member_exists_and_active
4 BORROW_REC_SIZE = 36

```

ภาพที่ 4-24 ตรวจสอบว่าสมาชิกที่มีรหัส (member_id)

4.4.3 def _scan_max_id(filename="storage/borrows.dat"):

ฟังก์ชันนี้ใช้สำหรับค้นหา รหัสสูงสุด (max ID) ของรายการยืม-คืน (borrow_id) ที่มีอยู่ในไฟล์ borrows.dat เพื่อใช้ในการกำหนดรหัสใหม่เวลาเพิ่มข้อมูลการยืม-คืนใหม่เข้าระบบ

การทำงานของฟังก์ชันเริ่มจากการเปิดไฟล์ borrows.dat ในโหมดอ่านไบนารี (rb) และวนอ่านข้อมูลในไฟล์ทีละระเบียน โดยแต่ละระเบียนมีขนาดคงที่ตามที่กำหนดใน BORROW_REC_SIZE เมื่ออ่านข้อมูลของแต่ละระเบียนเสร็จ จะนำ 4 ไบต์แรกมาตีความเป็นจำนวนเต็มด้วย from_i32() ซึ่งคือรหัสของการยืม (borrow_id) แล้วเปรียบเทียบกับค่ารหัสที่มากที่สุดที่เจออยู่ในตอนนี้ (max_id) หากค่าที่อ่านได้ใหม่มากกว่า max_id จะอัปเดตค่า max_id ให้เท่ากับค่านั้น ทำซ้ำไปเรื่อย ๆ จนกว่าจะอ่านครบทุกระเบียนในไฟล์



```

1 def _scan_max_id(filename="storage/borrows.dat"):
2     max_id = 0
3     try:
4         with open(filename, "rb") as f:
5             while chunk := f.read(BORROW_REC_SIZE):
6                 bid = from_i32(chunk[0:4])
7                 if bid > max_id:
8                     max_id = bid
9     except FileNotFoundError:
10         pass
11     return max_id
12
13 def get_next_borrow_id(filename="storage/borrows.dat"):
14     return _scan_max_id(filename) + 1
15

```

ภาพที่ 4-25 ค้นหาห้สสูงสุด (max ID) ของรายการยืม-คืน (borrow_id)

4.4.4 def _find_record_pos_by_id(target_id, filename="storage/borrows.dat"):

ฟังก์ชันนี้ทำหน้าที่ค้นหารายการยืม-คืนที่มีรหัส (borrow_id) ตรงกับรหัสที่ผู้ใช้ต้องการ (target_id) ภายในไฟล์ borrows.dat และส่งกลับตำแหน่งของข้อมูลนั้นในไฟล์ พร้อมทั้งเนื้อหาของระเบียน (record) ที่อ่านเจอ

การทำงานเริ่มจากการเปิดไฟล์ borrows.dat ในโหมดอ่านแบบไบนารี (rb) จากนั้นฟังก์ชันจะวนอ่านไฟล์ทีละระเบียน โดยแต่ละระเบียนมีขนาดคงที่ตาม BORROW_REC_SIZE ตัวแปร pos จะเก็บตำแหน่งไบต์เริ่มต้นของระเบียนนั้นในไฟล์ เริ่มจาก 0 แล้วเพิ่มขึ้นเรื่อย ๆ ตามขนาดของระเบียนที่อ่านไปแล้ว

เมื่ออ่านข้อมูลระเบียนออกมา ฟังก์ชันจะนำ 4 ไบต์แรกมาตีความเป็นจำนวนเต็มด้วย from_i32() เพื่อให้ได้ค่าของ borrow_id จากนั้นเปรียบเทียบกับ target_id ถ้าพบค่าตรงกัน ฟังก์ชันจะคืนค่ากลับเป็น ตำแหน่ง (pos) และข้อมูลระเบียน (chunk) ที่พบในทันที

ถ้าไฟล์ไม่มีอยู่ (เช่นยังไม่เคยสร้างไฟล์มาก่อน) ฟังก์ชันจะดัก FileNotFoundError แล้วคืนค่า (-1, None) กลับมาเพื่อบอกว่าไม่พบข้อมูลใด ๆ เช่นเดียวกัน หากวนครบทั้งไฟล์แล้วยังไม่เจอรหัสที่ตรงกับ target_id ก็คืนค่า (-1, None) กลับมาเช่นกัน

```

1 def _find_record_pos_by_id(target_id, filename="storage/borrows.dat"):
2     try:
3         with open(filename, "rb") as f:
4             pos = 0
5             while chunk := f.read(BORROW_REC_SIZE):
6                 borid = from_i32(chunk[0:4])
7                 if borid == target_id:
8                     return pos, chunk
9                 pos += BORROW_REC_SIZE
10    except FileNotFoundError:
11        return -1, None
12    return -1, None
13

```

ภาพที่ 4-26 ค้นหาการยืม-คืนที่มีรหัส (borrow_id)

4.4.5 ฟังก์ชัน _active_duplicate_exists

มีหน้าที่ตรวจสอบว่าสมาชิกที่กำหนด (member_id) กำลังยืมหนังสือเล่มเดิม (book_id) อยู่ในปัจจุบันหรือไม่ และยังไม่ได้คืนหนังสือเล่มนั้น ฟังก์ชันนี้ถูกออกแบบมาเพื่อป้องกันไม่ให้ผู้สามารถยืมหนังสือเล่มเดียวกันซ้ำได้ก่อนที่จะทำการคืน

```

1 def _active_duplicate_exists(member_id: int, book_id: int, filename="storage/borrows.dat") -> bool:
2     """Return True if this member currently borrows the same book and has not returned it yet."""
3     try:
4         with open(filename, "rb") as f:
5             while chunk := f.read(BORROW_REC_SIZE):
6                 mid = from_i32(chunk[4:8])
7                 bid = from_i32(chunk[8:12])
8                 ret = from_str(chunk[28:36])
9                 if mid == member_id and bid == book_id and ret == "00000000":
10                    return True
11    except FileNotFoundError:
12        pass
13    return False
14

```

ภาพที่ 4-27 ตรวจสอบว่าสมาชิกที่กำหนด (member_id) กำลังยืมหนังสือเล่มเดิม (book_id)

4.4.6 ฟังก์ชัน add_borrow(filename="storage/borrows.dat")

“ยืมหนังสือ” ใหม่ลงไฟล์ borrows.dat โดยเริ่มจากสร้างหมายเลขยืม (borrow_id) แบบอัตโนมัติผ่าน get_next_borrow_id() แล้วแจ้งผู้ใช้ให้ทราบ จากนั้นระบบจะวนถามรหัสสมาชิกและรหัสหนังสือที่ละรายการ โดยบังคับให้ป้อนเป็นตัวเลขและตรวจสอบความมีอยู่จริงและสถานะ Active ด้วย member_exists_and_active(member_id) และ book_exists_and_active(book_id) หากค่าใดไม่ผ่านจะให้กรอกใหม่ เมื่อผ่านแล้วจะมีการป้องกันการยืมซ้ำด้วย _active_duplicate_exists(member_id, book_id, filename) ถ้าพบว่าบุคคลเดียวกันยังถือครองหนังสือเล่มเดิมอยู่และยังไม่คืน ฟังก์ชันจะยุติทันทีและแจ้งเหตุผล ต่อมาเป็นการรับวันที่ยืมและกำหนดวันครบกำหนด โดยใช้ตัวช่วย _input_date_yyyymmdd(label) ซึ่งคืนค่าวันที่ในรูปแบบ YYYYMMDD ที่ผ่านการตรวจสอบแล้ว และบังคับให้วันครบกำหนดต้องไม่ก่อนวันยืม จากนั้นดึงสถานะสต็อกของหนังสือด้วย get_book_avail(book_id) เพื่อตรวจสอบว่ามีเล่มว่าง (avail) มากกว่าศูนย์หรือไม่ ถ้าไม่พอจะปฏิเสธการยืม หากมีเพียงพอจึงประกอบเรคอร์ดยืมขึ้นมาด้วยลำดับฟิลด์คงที่: borrow_id, member_id, book_id (ทั้งหมดเก็บเป็นจำนวนเต็ม 4 ไบต์ด้วย to_i32), ตามด้วย borrow_date, due_date และ return_date (เก็บเป็นสตริง 8 ไบต์แบบ fixed ด้วย fix_str) โดยกำหนด return_date เป็น "00000000" เพื่อสื่อว่ายังไม่คืน แล้วเขียนเรคอร์ดต่อท้ายไฟล์ borrows.dat ด้วยโหมด "ab" เสร็จแล้วจึงอัปเดตจำนวนเล่มว่างของหนังสือโดยลดลงหนึ่งผ่าน set_book_avail_at_pos(bpos, avail - 1) ซึ่งใช้ตำแหน่งเรคอร์ด (bpos) ที่ได้จาก get_book_avail สุดท้ายจะแจ้งว่าเพิ่มรายการยืมสำเร็จ ฟังก์ชันนี้จึงทำหน้าที่ครบทั้งตรวจสอบผู้ยืมและหนังสือ ความถูกต้องของวันที่ การกันยืมซ้ำ และการตัดสต็อกก่อนบันทึกเรคอร์ดแบบ fixed-length ลงไฟล์ไบนารีอย่างสอดคล้องกับโครงสร้างข้อมูลของระบบ


```

1 def add_borrow(filename="storage/borrows.dat"):
2     borrow_id = get_next_borrow_id(filename)
3     print(f"Assigned Borrow ID: {borrow_id}")
4
5     while True:
6         mid = input("Enter Member ID: ").strip()
7         if mid.isdigit():
8             member_id = int(mid)
9             if member_exists_and_active(member_id):
10                 break
11             else:
12                 print("Member does not exist or has been deleted.")
13         else:
14             print("Invalid Member ID. Must be digits.")
15
16     while True:
17         bid = input("Enter Book ID: ").strip()
18         if bid.isdigit():
19             book_id = int(bid)
20             if book_exists_and_active(book_id):
21                 break
22             else:
23                 print("Book does not exist or has been deleted.")
24         else:
25             print("Invalid Book ID. Must be digits.")
26
27     if _active_duplicate_exists(member_id, book_id, filename):
28         print("This member already has this book borrowed and not yet returned.")
29         return
30
31     borrow_date = _input_date_YYYYMMDD("Borrow")
32
33     while True:
34         due_date = _input_date_YYYYMMDD("Due")
35         if due_date >= borrow_date:
36             break
37         print("Due date cannot be earlier than borrow date.")
38
39     bpos, total, avail = get_book_avail(book_id)
40     if avail <= 0:
41         print("No available copies for this book.")
42         return
43
44     record = (
45         to_i32(borrow_id) +
46         to_i32(member_id) +
47         to_i32(book_id) +
48         fix_str(borrow_date, 8) +
49         fix_str(due_date, 8) +
50         fix_str("00000000", 8)
51     )
52     with open(filename, "ab") as f:
53         f.write(record)
54
55     set_book_avail_at_pos(bpos, avail - 1)
56     print("Borrow record added successfully.")
57

```

ภาพที่ 4-28 “ยืมหนังสือ” ใหม่ลงไฟล์ borrows.dat

4.4.7 ฟังก์ชัน return_book(filename="storage/borrows.dat")

มีหน้าที่จัดการกระบวนการ “คืนหนังสือ” ในระบบ โดยทำงานกับไฟล์ไบนารี borrows.dat ที่เก็บรายการการยืมแบบ fixed-length ขนาด 36 ไบต์ต่อเรคอร์ด เมื่อเริ่มทำงาน ระบบจะวนรับค่ารหัสการยืม (Borrow ID) จากผู้ใช้ โดยตรวจสอบให้แน่ใจว่าเป็นตัวเลขล้วน จากนั้นนำค่าดังกล่าวไปค้นหาในไฟล์ผ่าน `_find_record_pos_by_id(target_id, filename)` เพื่อดึงตำแหน่ง (pos) และข้อมูลเรคอร์ด (chunk) ถ้าไม่พบหรือหาไม่เจอ จะหยุดการทำงานทันทีพร้อมแจ้งข้อความว่าไม่พบรายการยืม ต่อมาจะตรวจสอบว่าหนังสือในรายการนี้ถูกคืนไปแล้วหรือไม่ โดยดูจากฟิลด์ return_date

ซึ่งอยู่ในช่วงไบต์ที่ 28–35 ของเรคอร์ด หากไม่ใช่ "00000000" แสดงว่ามีการคืนแล้ว ฟังก์ชันจะไม่ทำซ้ำและจบการทำงานทันที หากยังไม่ได้คืน ระบบจะขอรับ return_date จากผู้ใช้โดยใช้ `_input_date_YYYYMMDD("Return")` ซึ่งเป็นฟังก์ชันที่ตรวจสอบความถูกต้องของรูปแบบวันที่ (YYYYMMDD) และบังคับว่าต้องไม่เร็วกว่าวันที่ยืม (borrow_date จากไบต์ที่ 12–19) เมื่อวันที่ถูกต้องแล้ว จะให้ผู้ใช้ยืนยันความตั้งใจคืนหนังสือผ่านข้อความแจ้งเตือน หากปฏิเสธ (ไม่พิมพ์ y) การคืนจะถูกยกเลิก แต่หากยืนยันจะทำการเพิ่มจำนวนเล่มว่างของหนังสือกลับคืน โดยเริ่มจากดึง book_id จากไบต์ที่ 8–11 ของเรคอร์ด จากนั้นใช้ `get_book_avail(book_id)` เพื่อดึงตำแหน่งและจำนวนเล่มว่าง แล้วใช้ `set_book_avail_at_pos(bpos, avail + 1)` อัปเดตให้เพิ่มกลับมา 1 เล่ม

```

1 def return_book(filename="storage/borrows.dat"):
2     while True:
3         target = input("Enter Borrow ID to return: ").strip()
4         if target.isdigit():
5             target_id = int(target)
6             break
7         print("Invalid Borrow ID. Must be digits.")
8
9     pos, chunk = _find_record_pos_by_id(target_id, filename)
10    if pos < 0 or chunk is None:
11        print("Borrow record not found.")
12        return
13
14    cur_return = from_str(chunk[28:36])
15    if cur_return != "00000000":
16        print("This borrow has already been returned.")
17        return
18
19    while True:
20        return_date = _input_date_YYYYMMDD("Return")
21        borrow_date = from_str(chunk[12:20])
22        if return_date <= borrow_date:
23            break
24        print("Return date cannot be earlier than borrow date.")
25
26    confirm = input(f"Are you sure you want to return Borrow ID {target_id}? (y/n): ").strip().lower()
27    if confirm != "y":
28        print("Return cancelled.")
29        return
30
31    book_id = from_int(chunk[8:12])
32    bpos, total, avail = get_book_avail(book_id)
33    if bpos >= 0:
34        set_book_avail_at_pos(bpos, avail + 1)
35
36    with open(filename, "rb") as f:
37        f.seek(pos + 28)
38        f.write(to_str(return_date, 8))
39
40    print("Book returned successfully.")
41

```

ภาพที่ 4-29 จัดการกระบวนการ “คืนหนังสือ” ในระบบ

4.5 view_borrows(filename="storage/borrows.dat")

ใช้สำหรับแสดงข้อมูลการยืม-คืนหนังสือทั้งหมดจากไฟล์ borrows.dat โดยจะเปิดไฟล์แบบไบนารี อ่านข้อมูลที่ละเรคอร์ด แล้วแปลงออกมาเป็นค่าตัวเลขและข้อความ เช่น BorrowID, MemberID, BookID, วันที่ยืม, กำหนดคืน และวันที่คืนจริง จากนั้นนำมาแสดงในรูปตาราง หากไฟล์ไม่มีข้อมูลหรือไม่พบไฟล์ จะแจ้งว่าไม่มีรายการยืมอยู่ในระบบ ฟังก์ชันนี้ช่วยให้ตรวจสอบสถานะการยืม-คืนได้ง่ายและเป็นระเบียบ

```

1 def view_borrows(filename="storage/borrows.dat"):
2     try:
3         with open(filename, "rb") as f:
4             print(f'{"BorrowID":<8} {"MemberID":<8} {"BookID":<8} {"BorrowDate":<10} {"DueDate":<10} {"ReturnDate":<10}')}
5             print("-"*60)
6             found = False
7             while chunk := f.read(BORROW_REC_SIZE):
8                 borrow_id = from_i32(chunk[0:4])
9                 member_id = from_i32(chunk[4:8])
10                book_id = from_i32(chunk[8:12])
11                borrow_date = from_str(chunk[12:20])
12                due_date = from_str(chunk[20:28])
13                return_date = from_str(chunk[28:36])
14                print(f'{"borrow_id":<8} {"member_id":<8} {"book_id":<8} {"borrow_date":<10} {"due_date":<10} {"return_date":<10}')}
15                found = True
16            if not found:
17                print("No borrow records found.")
18        except FileNotFoundError:
19            print("No borrow records found.")

```

ภาพที่ 4-30แสดงข้อมูลการยืม-คืนหนังสือทั้งหมดจากไฟล์ borrows.dat

import os ใช้สำหรับเรียกใช้ฟังก์ชันเกี่ยวกับระบบปฏิบัติการ เช่น การสร้างโฟลเดอร์ใหม่ ลบไฟล์ ตรวจสอบเส้นทาง หรือจัดการชื่อไฟล์ ซึ่งในโปรเจกต์นี้อาจถูกใช้ตอนสร้างโฟลเดอร์สำหรับเก็บรายงานอัตโนมัติ

from datetime import datetime นำเข้าคลาส datetime จากโมดูลมาตรฐาน datetime เพื่อใช้งานวันที่และเวลา เช่น การบันทึกเวลาที่สร้างรายงาน การแปลง timestamp เป็นรูปแบบอ่านง่าย หรือการกำหนดชื่อไฟล์รายงานให้มีวันที่เวลา

from module.utils import from_i32, from_str คือการนำเข้าฟังก์ชันที่เขียนไว้ในไฟล์ utils.py ซึ่งอยู่ในโฟลเดอร์ module โดย from_i32 ใช้แปลงข้อมูลจากไบต์ 4 ตัวกลับเป็นจำนวนเต็ม (int) และ from_str ใช้แปลงข้อมูลจาก fixed-length string ที่เก็บในไฟล์ไบนารีกลับเป็นข้อความปกติ



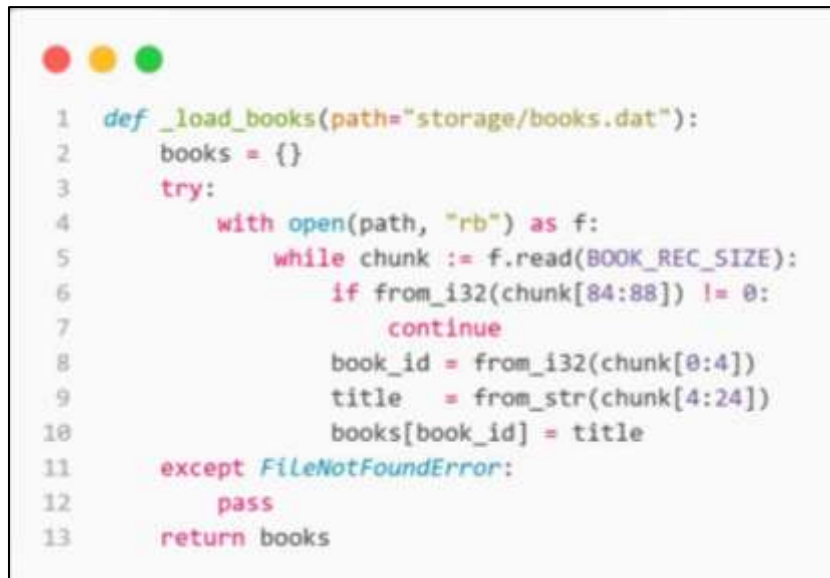
```
1 import os
2 from datetime import datetime
3 from module.utils import from_i32, from_str
```

ภาพที่ 4-31 สำหรับเรียกใช้ฟังก์ชันเกี่ยวกับระบบปฏิบัติการ

4.6 ฟังก์ชัน load

4.6.1 ฟังก์ชัน _load_books

มีหน้าที่อ่านไฟล์ books.dat ซึ่งเก็บข้อมูลหนังสือในรูปแบบ binary fixed-length record แล้วดึงเฉพาะ BookID และ Title มาเก็บไว้ใน dictionary เพื่อใช้งานต่อไปในโปรแกรม เช่น การสร้างรายงาน หรือการแสดงชื่อหนังสือจากรหัส การทำงานเริ่มจากการสร้าง dictionary ว่างชื่อ books จากนั้นพยายามเปิดไฟล์ books.dat แบบ binary (rb) และวนลูปอ่านเรคคอร์ดทีละก้อนตามขนาดที่กำหนดใน BOOK_REC_SIZE สำหรับแต่ละเรคคอร์ด จะตรวจสอบสถานะการลบที่ตำแหน่งไบต์ 84-87 ถ้าค่าไม่ใช่ 0 หมายความว่าถูกลบไปแล้ว จึงข้ามไป แต่ถ้าเป็น 0 แสดงว่ายัง active อยู่ จึงอ่าน book_id จากไบต์ 0-3 และ title จากไบต์ 4-23 แล้วบันทึกลง dictionary โดยใช้ book_id เป็น key และ title เป็น value หากไฟล์ไม่ถูกสร้างหรือหาไม่เจอ (FileNotFoundError) ฟังก์ชันจะไม่ทำอะไรและคืน dictionary ว่างๆ กลับมา



```

1  def _load_books(path="storage/books.dat"):
2      books = {}
3      try:
4          with open(path, "rb") as f:
5              while chunk := f.read(BOOK_REC_SIZE):
6                  if from_i32(chunk[84:88]) != 0:
7                      continue
8                  book_id = from_i32(chunk[0:4])
9                  title = from_str(chunk[4:24])
10                 books[book_id] = title
11     except FileNotFoundError:
12         pass
13     return books

```

ภาพที่ 4-32 อ่านไฟล์ books.dat เก็บข้อมูลหนังสือ

4.6.2 ฟังก์ชัน _load_members

มีหน้าที่อ่านไฟล์ members.dat ซึ่งเก็บข้อมูลสมาชิกในรูปแบบ binary fixed-length record แล้วเลือกเอาเฉพาะข้อมูลที่ยังใช้งานอยู่ (Active) มาเก็บในรูปแบบ dictionary โดยมี member_id เป็น key และ value เป็น tuple (name, phone)

การทำงานเริ่มต้นด้วยการสร้าง dictionary วางชื่อ members จากนั้นพยายามเปิดไฟล์ members.dat แบบ binary (rb) แล้วอ่านเรคคอร์ดทีละก้อนตามขนาด MEM_REC_SIZE ภายในแต่ละเรคคอร์ดจะตรวจสอบฟิลด์สถานะที่ไบต์ 88-91 ถ้าค่าไม่ใช่ 0 แสดงว่าถูกลบ (Deleted) จึงข้ามไป แต่ถ้าเป็น 0 แปลว่าข้อมูลยัง Active อยู่ จึงอ่าน member_id จากไบต์ 0-3, อ่าน name จากไบต์ 4-23 และอ่าน phone จากไบต์ 44-53 แล้วบันทึกลงใน dictionary โดยจัดเก็บเป็น {member_id: (name, phone)}



```

1  def _load_members(path="storage/members.dat"):
2      members = {}
3      try:
4          with open(path, "rb") as f:
5              while chunk := f.read(MEM_REC_SIZE):
6                  if from_i32(chunk[88:92]) != 0:
7                      continue
8                  member_id = from_i32(chunk[0:4])
9                  name      = from_str(chunk[4:24])
10                 phone     = from_str(chunk[44:54])
11                 members[member_id] = (name, phone)
12     except FileNotFoundError:
13         pass
14     return members

```

ภาพที่ 4-33 อ่านไฟล์ members.dat เก็บข้อมูลสมาชิก

4.6.3 ฟังก์ชัน _load_borrows

มีหน้าที่อ่านข้อมูลจากไฟล์ borrows.dat ซึ่งเก็บรายการการยืม-คืนหนังสือในรูปแบบ binary fixed-length record แล้วนำแต่ละเรคอร์ดมาเก็บในลิสต์ records โดยแต่ละเรคอร์ดถูกแปลงให้อยู่ในรูปแบบ dictionary ที่เข้าใจง่ายและเข้าถึงด้วย key ได้ทันที การทำงานคือเปิดไฟล์ borrows.dat แบบ binary (rb) และวนอ่านข้อมูลที่ละก้อนขนาด BORROW_REC_SIZE ทุกครั้งที่อ่านได้ จะดึงค่าต่าง ๆ ออกมาได้แก่ borrow_id, member_id, book_id (อ่านจากไบต์และแปลงเป็น int ด้วย from_i32) และ borrow_date, due_date, return_date (อ่านจากไบต์ string fixed-length แล้วแปลงกลับเป็นข้อความด้วย from_str) จากนั้นสร้าง dictionary ที่เก็บข้อมูลของเรคอร์ดนั้น แล้วเพิ่มเข้าไปในลิสต์ records หากไฟล์ยังไม่มีอยู่จริง (FileNotFoundError) ฟังก์ชันจะไม่ทำอะไรและคืนลิสต์ว่างแทน



```

1 def _load_borrows(path="storage/borrows.dat"):
2     records = []
3     try:
4         with open(path, "rb") as f:
5             while chunk := f.read(BORROW_REC_SIZE):
6                 records.append({
7                     "borrow_id": from_i32(chunk[0:4]),
8                     "member_id": from_i32(chunk[4:8]),
9                     "book_id": from_i32(chunk[8:12]),
10                    "borrow_date": from_str(chunk[12:20]),
11                    "due_date": from_str(chunk[20:28]),
12                    "return_date": from_str(chunk[28:36]),
13                })
14     except FileNotFoundError:
15         pass
16     return records

```

ภาพที่ 4-34 อ่านข้อมูลจากไฟล์ borrows.dat ซึ่งเก็บรายการการยืม-คืนหนังสือ

4.7 ฟังก์ชัน generate_report()

เป็นตัวสร้างรายงานสรุปการยืม-คืนของระบบห้องสมุด โดยโครงสร้างมันทำงานเป็นขั้นตอนดังนี้ครับ เมื่อเรียกใช้งาน ฟังก์ชันจะตรวจสอบก่อนว่ามีโฟลเดอร์ report อยู่หรือยัง ถ้าไม่มีจะสร้างใหม่ขึ้นมาทันที จากนั้นสร้างชื่อไฟล์รายงานโดยอิงตามวันและเวลาปัจจุบัน เช่น report_2025-10-04_13-25-44.txt เพื่อให้ไฟล์แต่ละครั้งไม่ซ้ำกัน ขั้นตอนมา ระบบจะโหลดข้อมูลทั้งหมดจากไฟล์ในบาริ 3 กลุ่ม ได้แก่ _load_books() เพื่อเอาข้อมูลหนังสือ (id และชื่อเรื่อง), _load_members() เพื่อเอาข้อมูลสมาชิก (id, ชื่อ, เบอร์โทร), และ _load_borrows() เพื่อเอาข้อมูลการยืม-คืนทั้งหมด หลังจากนั้นจะสรุปจำนวนสถิติคร่าว ๆ เช่น ยืมทั้งหมดกี่ครั้ง (total_borrows), ยืมที่ยังไม่คืน (active_borrows), และจำนวนที่คืนไปแล้ว (returned_borrows) ตัวไฟล์รายงานจะถูกสร้างขึ้นในรูปแบบข้อความ (.txt)

โดยเริ่มจากพิมพ์หัวเรื่อง มีบรรทัดเส้นแบ่งสวยงาม และบอกเวลาที่สร้างรายงาน รวมถึงข้อมูลว่าไฟล์นี้ใช้การเข้ารหัส UTF-8 และเป็นรายงานที่มีเฉพาะสรุปการยืม-คืน จากนั้นเข้าสู่ส่วน “BORROW/RETURN RECORDS” ซึ่งจะพิมพ์หัวตาราง เช่น BorrowID, MemberID, Member Name, Phone, BookID, Book Title, Borrow, Due, Return และ Status แล้ววนลูปเติมแถวของข้อมูลการยืมแต่ละรายการ แต่ละเรคอร์ดที่พิมพ์ออกมาจะเชื่อมโยง id กับชื่อจริง โดย member_id จะถูกแทนด้วยชื่อและเบอร์โทรจาก _load_members ถ้าไม่เจอจะแสดงเป็น (unknown) ส่วน book_id จะเชื่อมกับชื่อหนังสือจาก _load_books ถ้าไม่เจอจะแสดงว่า (unknown) ข้อมูลวันที่ (borrow_date, due_date, return_date)

จะถูกจัดรูปแบบให้เป็น YYYY-MM-DD ด้วย `_fmt_date8()` ถ้าวันคืนยังเป็น "00000000" แปลว่าหนังสือยังไม่ถูกคืน ฟังก์ชันจะแสดงสถานะเป็น "Active" ถ้ามีวันที่คืนแล้วก็จะเปลี่ยนเป็น "Returned"

เมื่อจบตารางจะแสดงส่วนสรุป (SUMMARY) ที่บอกจำนวนรายการทั้งหมด จำนวนที่กำลังยืมอยู่ และจำนวนที่คืนแล้ว

```

1 def generate_report():
2     if not os.path.exists("report"):
3         os.makedirs("report")
4
5     ts = datetime.now()
6     filename = f"report/report_{ts.strftime('%Y-%m-%d-%H-%M-%S')}.txt"
7
8     books = _load_books()
9     members = _load_members()
10    borrows = _load_borrows()
11
12    total_borrows = len(borrows)
13    active_borrows = sum(1 for b in borrows if b["return_date"] == "00000000")
14    returned_borrows = total_borrows - active_borrows
15
16    with open(filename, "w", encoding="utf-8") as f:
17        f.write("\n" * 100)
18        f.write("LIBRARY MANAGEMENT SYSTEM - SUMMARY/RETURN SUMMARY\n")
19        f.write("\n" * 100)
20        f.write(f"Generated At : {ts.strftime('%Y-%m-%d %H:%M:%S')}\n")
21        f.write(f"Encoding : UTF-8\n")
22        f.write(f"Content : Borrow/Return summary table\n")
23        f.write("\n" * 100)
24
25        f.write("SUMMARY/RETURN RECORDS\n")
26        f.write("\n" * 100)
27        f.write(f"{'BorrowID':<8} {'MemberID':<8} {'Member Name':<20} {'Phone':<12} "
28              f"{'BookID':<8} {'Book Title':<20} {'Borrow':<10} {'Date':<10} {'Return':<10} {'Status':<8}\n")
29        f.write("\n" * 100)
30
31        if not borrows:
32            f.write("(no borrow records)\n")
33        else:
34            for r in borrows:
35                mid = r["member_id"]
36                bid = r["book_id"]
37
38                name, phone = members.get(mid, ["unknown", ""])
39                title = books.get(bid, "unknown")
40
41                status = "Active" if r["return_date"] == "00000000" else "Returned"
42                bdate = _fmt_date8(r["borrow_date"])
43                ddate = _fmt_date8(r["due_date"])
44                rdate = _fmt_date8(r["return_date"])
45
46                f.write(f"{'BorrowID':<8} {mid:<8} {name:<20} {phone:<12} "
47                      f"{'BookID':<8} {title:<20} {bdate:<10} {ddate:<10} {rdate:<10} {status:<8}\n")
48
49            f.write("\n" * 100)
50            f.write("SUMMARY\n")
51            f.write("\n" * 100)
52            f.write(f"Total Borrow Records : {total_borrows}\n")
53            f.write(f"Currently Borrowed : {active_borrows}\n")
54            f.write(f"Returned : {returned_borrows}\n")
55            f.write("\n" * 100)
56
57    print(f"Report generated: {filename}")
58

```

ภาพที่ 4-35 ฟังก์ชันส่วนเสริม Utility

4.8 ฟังก์ชัน `fix_str`

ใช้สำหรับจัดการข้อความ (string) ให้กลายเป็น ข้อมูลแบบไบนารีความยาวคงที่ (fixed-length bytes) เพื่อเก็บลงในไฟล์ไบนารี โดยหลักการคือ ข้อความที่รับเข้ามาจะถูกแปลงเป็นไบนารีด้วยการเข้ารหัส UTF-8 (`text.encode("utf-8")`) ตัดความยาวให้ไม่เกินค่าที่กำหนด (`[:length]`) เพื่อป้องกันข้อความยาวเกินพื้นที่ที่จัด

ไว้ในเรคอร์ด. ถ้าข้อความสั้นเกินไปจะถูกเติมท้ายด้วยค่า `\x00` (null byte) จนมีความยาวเท่ากับ `length` (`ljust(length, b"\x00")`)



```

1 def fix_str(text, length):
2     """Convert string to fixed-length bytes (UTF-8, padded with 0x00)."""
3     return text.encode("utf-8")[:length].ljust(length, b"\x00")

```

ภาพที่ 4-36 ฟังก์ชัน `fix_str`

4.9 ฟังก์ชัน `from_str`

ทำงานตรงข้ามกับ `fix_str` ครึ่ง คือใช้สำหรับ แปลงข้อมูลไบต์ที่มีความยาวคงที่ (fixed-length bytes) กลับมาเป็นข้อความ (string) ที่อ่านได้ตามปกติ การทำงานเริ่มจากการถอดรหัสไบต์ (`b.decode("utf-8")`) เพื่อคืนค่าเป็นข้อความ จากนั้นใช้ `.strip("\x00 ")` เพื่อลบตัวอักษรที่ไม่จำเป็นออก ได้แก่ `\x00` (null byte) ที่ถูกเติมไว้ตอนเขียนไฟล์ด้วย `fix_str` และช่องว่างที่อาจถูกใช้เติมให้ครบขนาดฟิลด์ ผลลัพธ์ที่ได้จึงเป็นข้อความดิบที่ถูกบันทึกจริงๆ โดยไม่มี padding เกินมา



```

1 def from_str(b):
2     """Convert fixed-length bytes to string (strip 0x00 and spaces)."""
3     return b.decode("utf-8").strip("\x00 ")

```

ภาพที่ 4-37 ฟังก์ชัน `from_str`

บทที่ 5

สรุปผล อภิปราย และข้อเสนอแนะ

5.1 สรุปผล

- 5.1.1 ระบบยืม-คืนหนังสือห้องสมุดต้นแบบ ที่ทำงานได้จริง
- 5.1.2 ผู้จัดทำได้ฝึกทักษะการเขียนโปรแกรมด้วย Python และการจัดการไฟล์
- 5.1.3 วัตถุประสงค์ทุกข้อบรรลุผลสำเร็จ

5.2 ปัญหาและอุปสรรค

- 5.2.1 พบข้อผิดพลาดจากการเขียนไฟล์ไบนารี เช่น การกำหนดขนาดไม่ตรง
- 5.2.2 ผู้ใช้ต้องพิมพ์ข้อมูลด้วยตนเอง อาจเกิดการสะกดผิดหรือใส่ข้อมูลไม่ครบ

5.3 ข้อเสนอแนะ

- 5.3.1 ในอนาคตควรปรับปรุงระบบให้เป็น Web Application หรือ GUI Application
- 5.3.2 เพิ่มระบบ ตรวจสอบข้อมูลอัตโนมัติ เช่น ไม่ให้ใส่รหัสซ้ำ
- 5.3.3 อาจเชื่อมต่อกับฐานข้อมูล เช่น SQLite หรือ MySQL

5.4 สิ่งที่ได้รับในการพัฒนาโครงการ

จากการพัฒนาโครงการครั้งนี้ ผู้จัดทำได้รับความรู้และประสบการณ์ด้านการออกแบบระบบการเขียนโปรแกรมด้วยภาษา Python การใช้โครงสร้างข้อมูลแบบไบนารี รวมถึงการคิดวิเคราะห์และแก้ไขปัญหาเชิงตรรกะ นอกจากนี้ยังได้ฝึกทักษะการทำงานเป็นทีม การแบ่งหน้าที่รับผิดชอบ และการจัดการเวลาให้สอดคล้องกับแผนงาน ทำให้ผู้จัดทำมีความเข้าใจในการบวนการพัฒนาระบบซอฟต์แวร์มากยิ่งขึ้น และสามารถนำไปประยุกต์ใช้ในโครงการหรืองานจริงในอนาคตได้