ระบบยืม – คืนหนังสือ

Borrow and return books in the library

นางสาวไรบีนา	**มะเก๊ะ**	**รหัส**	**6806022510122**	**Sec 2**
นายคฤษฐ์	**เสือเดช**	**รหัส**	**6806022510637**	**Sec 2**
นางสาวชาลิสา	**ยิ้มลำใย**	**รหัส**	**6806022510653**	**Sec 2**

โครงงานเล่มนี้เป็นส่วนนึงของการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้า พระนครเหนือ

ปีการศึกษา 2568

คำนำ

การจัดทำโครงงาน “ระบบยืม–คืนหนังสือ” นี้เป็นส่วนหนึ่งของรายวิชา Computer Programming (060233115) หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมสารสนเทศและเครือข่าย มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ มีวัตถุประสงค์เพื่อให้นักศึกษาได้ฝึกฝนการวิเคราะห์ ออกแบบ และพัฒนาโปรแกรมด้วยภาษา Python เพื่อสร้างระบบที่สามารถใช้งานได้จริงและเป็นการประยุกต์ใช้ความรู้ที่ได้เรียนมาอย่างเป็นรูปธรรม

คณะผู้จัดทำหวังว่าโครงงานนี้จะเป็นประโยชน์ต่อผู้สนใจและผู้ที่กำลังศึกษาในสาขาที่เกี่ยวข้อง หากมีข้อบกพร่องประการใด คณะผู้จัดทำขอน้อมรับไว้และขออภัยมา ณ ที่นี้ด้วย

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 วัตถุประสงค์

1.1.1 เพื่อพัฒนาระบบยืม - คืนหนังสือได้อย่างมีประสิทธิภาพ

1.1.2 เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วยภาษา Python

1.1.3 เพื่อเรียนนำความรู้ที่ได้เรียนรู้มาประยุกต์ใช้

1.2 ขอบเขต

1.2.1 ระบบรองรับการทำงานสำหรับ ผู้ใช้ทั่วไป (สมาชิก) และ ผู้ดูแลระบบ (บรรณารักษ์/ผู้ดูแลหนังสือ)

1.2.2 การจัดการข้อมูลหลักในระบบประกอบด้วย

1) หนังสือ (Books): เพิ่ม, แสดงรายการ, แก้ไข และลบ (เปลี่ยนสถานะเป็น Inactive)
2) สมาชิก (Members): เพิ่ม, แสดงรายการ, แก้ไข และลบ (เปลี่ยนสถานะเป็น Blocked/Inactive)
3) การยืม-คืน (Loans): บันทึกการยืม, คืน, ตรวจสอบสถานะ และแสดงประวัติการยืม–คืน

1.2.3 ระบบจัดเก็บข้อมูลทั้งหมดใน Binary File ได้แก่ books.dat, members.dat, และ loans.dat

1.2.4 ฟังก์ชันเสริม เช่น รายงานสรุปของระบบ (Summary Report)

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ผู้ใช้สามารถยืม–คืนหนังสือได้สะดวกและเป็นระบบ

1.3.2 ผู้จัดทำได้ฝึกทักษะการเขียนโปรแกรมจริง

1.3.3 ผู้จัดทำสามารถประยุกต์ความรู้ที่เรียนมาและเรียนรู้การจัดการข้อมูลไฟล์

1.4 เครื่องมือที่ใช้

1.4.1 โปรแกรม Visual Studio Code

1.4.2 โปรแกรม Microsoft word

บทที่ 2
โครงสร้างแฟ้มข้อมูลและฟิลด์

ในการพัฒนาระบบยืม–คืนหนังสือนี้ ข้อมูลทั้งหมดจะถูกจัดเก็บในแฟ้มข้อมูลแบบ Binary File โดยแฟ้มข้อมูลหลักแบ่งออกเป็น 3 ส่วน คือ แฟ้มหนังสือ (Books), แฟ้มสมาชิก (Members) และ แฟ้มการยืม–คืน (Loans) แต่ละแฟ้มประกอบด้วยฟิลด์ที่ใช้เก็บข้อมูลสำคัญตามรายละเอียดดังนี้

2.1 แฟ้มหนังสือ (Books)

แฟ้มนี้ใช้จัดเก็บข้อมูลหนังสือทั้งหมดในระบบ ประกอบด้วยฟิลด์ดังนี้

ฟิลด์	ชนิดข้อมูล	ความหมาย
book_id	Integer	รหัสหนังสือ
title	String (50 bytes)	ชื่อหนังสือ
author	String (30 bytes)	ชื่อผู้แต่ง
year	Integer	ปีที่พิมพ์
copies	Integer	จำนวนสำเนาของหนังสือ
borrowed	Integer	จำนวนที่ถูกยืม
status	Integer	สถานะหนังสือ (1 = Active, 0 = Inactive)

2.2 แฟ้มสมาชิก (Member)

แฟ้มนี้ใช้จัดเก็บข้อมูลสมาชิก ประกอบด้วยฟิลด์ดังนี้

ฟิลด์	ชนิดข้อมูล	ความหมาย
member_id	Integer	รหัสสมาชิก
name	String (40 bytes)	ชื่อสมาชิก
email	String (40 bytes)	อีเมลสมาชิก
phone	String (15 bytes)	เบอร์โทรศัพท์
status	Integer	สถานะสมาชิก (1 = Active, = Blocked)
total_borrows	Integer	จำนวนครั้งที่สมาชิกเคยยืมหนังสือ

2.3 แฟ้มการยืม-คืน (Loans)

แฟ้มนี้ใช้จัดเก็บรายการยืม - คืนหนังสือ ประกอบด้วยฟิลด์ดังนี้

ฟิลด์	ชนิดข้อมูล	ความหมาย
borrow_id	Integer	รหัสรายการยืม
book_id	Integer	รหัสหนังสือที่ยืม
member_id	Integer	รหัสสมาชิกที่ยืม
loan_date	String (10 bytes)	วันที่ยืม (YYYY-MM-DD)
due_date	String (10 bytes)	วันที่กำหนดคืน (YYYY-MM-DD)
return_date	String (10 bytes)	วันที่คืนจริง (YYYY-MM-DD)
status	Integer	สถานะการยืม (0 = ยืมอยู่, 0 = คืนแล้ว)

บทที่ 3
วิธีการใช้งานระบบ

บทนี้เป็นการอธิบายขั้นตอนและวิธีการใช้งานระบบยืม–คืนหนังสือห้องสมุด โดยระบบพัฒนาด้วยภาษา Python และจัดเก็บข้อมูลทั้งหมดในรูปแบบ Binary File (แฟ้ม books.dat, members.dat, loans.dat) การใช้งานระบบแบ่งออกเป็นเมนูหลักและเมนูย่อยดังนี้

3.1 การเริ่มต้นใช้งานระบบ

3.1.1 เปิดโปรแกรมโดยรันไฟล์ Python (python librerySystem.py)

3.1.2 ระบบจะแสดงเมนูหลัก (Main Menu) ซึ่งมีตัวเลือกดังนี้

3.1.2.1 Manage Books : จัดการข้อมูลหนังสือ

3.1.2.2 Manage Members : จัดการข้อมูลสมาชิก

3.1.2.3 Borrow/Return : รายการยืม - คืน

3.1.2.4 Reports : แสดงรายงานสรุป

3.1.2.5 Exit : ออกจากระบบ

```
=== Library System ===
1. Manage Books
2. Manage Members
3. Borrow/Return
4. Reports
5. Exit
```

ภาพที่ 3-1 เมนูหลัก

3.2 การใช้งานเมนูจัดการหนังสือ (Manage Books)

3.2.1 เพิ่มข้อมูลหนังสือ (Add Book) : กรอกข้อมูลหนังสือ เช่น รหัสหนังสือ, ชื่อหนังสือ, ผู้แต่ง, ปีพิมพ์ และจำนวนสำเนา

3.2.1.1 กรอกหมายเลข 1 เพื่อเพิ่มข้อมูลหนังสือ ดังภาพที่ 3-2

```
--- Books Menu ---
1. Add Book
2. View Books
3. Update Book
4. Delete Book
5. Back
Choose: 1
```

ภาพที่ 3-2 เลือกเมนูหมายเลข 1 เพื่อเพิ่มข้อมูลหนังสือ

3.2.1.2 จากนั้นกรอกข้อมูลตามที่ปรากฏบนหน้าจอจนครบ ดังภาพที่ 3-3

```
How many books do you want to add? 1

--- Adding book 1 of 1 ---
Book ID: 1011
Title: Games
Author: Games H. Manhutt
Year: 2023
Copies: 3
Book added successfully.
```

ภาพที่ 3-3 กรอกข้อมูลเพื่อเพิ่มข้อมูลหนังสือ

3.2.2 ดูรายการหนังสือ (View Books) : แสดงรายการหนังสือทั้งหมดพร้อมสถานะและจำนวนคงเหลือ

3.2.2.1 กรอกหมายเลข 2 เพื่อดูรายการหนังสือ ดังภาพที่ 3-4

```
--- Books Menu ---
1. Add Book
2. View Books
3. Update Book
4. Delete Book
5. Back
Choose: 2
```

ภาพที่ 3-4 เลือกเมนูหมายเลข 2 เพื่อดูรายการหนังสือ

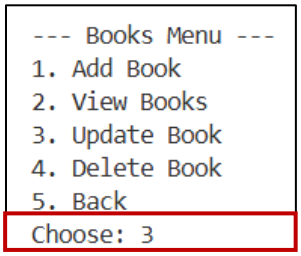
3.2.2.2 จากนั้นจะปรากฏรายการหนังสือทั้งหมดในรูปแบบตาราง ดังภาพที่ 3-5

ID	Title	Author	Year	Copies	Status
1001	Python	Mark Lutz	2021	5	Active
1002	Linux	William Shotts	2020	3	Active
1003	AI	Stuart Russell	2022	2	Inactive
1004	Java	Herbert Schildt	2019	3	Active
1005	HTML	Jon Duckett	2018	2	Active
1006	Data Communication	Bahrouz A. Forozan	2007	5	Inactive
1007	Physical	Cream Pathuy	2025	4	Active
1008	youtube	face	2018	2	Active
1009	royal	cheater	2045	6	Active
2003	no yummy	ewwe	2014	5	Inactive
10	ComPro	kit	2025	5	Inactive
1011	Games	Games H. Manhutt	2023	3	Active

ภาพที่ 3-5 ตารางแสดงรายการหนังสือทั้งหมด

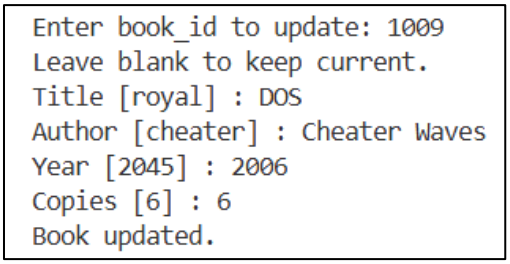
3.2.3 แก้ไขข้อมูลหนังสือ (Update Book) : สามารถแก้ไขข้อมูลของหนังสือที่มีอยู่

3.2.3.1 กรอกหมายเลข 3 เพื่อแก้ไขข้อมูลหนังสือ ดังภาพที่ 3-6



ภาพที่ 3-6 เลือกเมนูหมายเลข 3 เพื่อแก้ไขข้อมูลหนังสือ

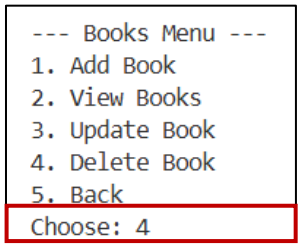
3.2.3.2 จากนั้นกรอกข้อมูลตามที่ปรากฏบนหน้าจอจน ดังภาพที่ 3-7



ภาพที่ 3-7 กรอกข้อมูลเพื่อแก้ไขข้อมูลหนังสือตามหัวข้อที่ต้องการ

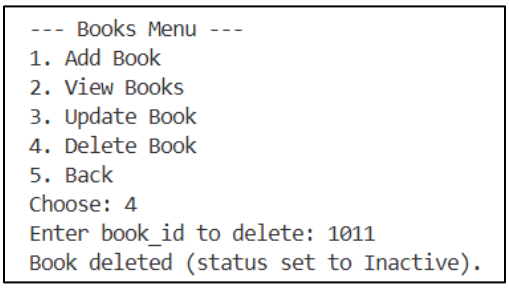
3.2.4 ลบรายการหนังสือ (Delete Book) : เปลี่ยนสถานะหนังสือเป็น Inactive หากหนังสือถูกยืมอยู่ จะไม่สามารถลบได้

3.2.4.1 กรอกหมายเลข 4 เพื่อลบรายการหนังสือ ดังภาพที่ 3-8



ภาพที่ 3-8 เลือกเมนูหมายเลข 4 เพื่อลบรายการหนังสือ

3.2.4.2 จากนั้นกรอกรหัสหนังสือเล่มที่ต้องการลบ ดังภาพที่ 3-9

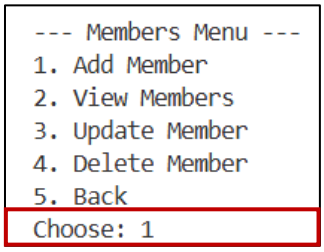


ภาพที่ 3-9 กรอกรหัสหนังสือเพื่อลบรายการหนังสือที่ต้องการ

3.3 การใช้งานเมนูจัดการสมาชิก (Manage Members)

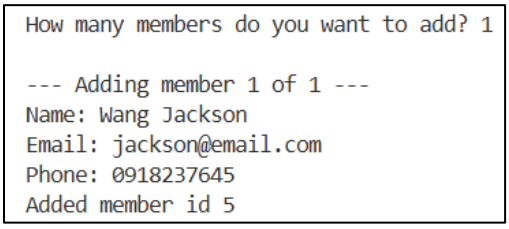
3.3.1 เพิ่มข้อมูลสมาชิก (Add Member) : ลงทะเบียนสมาชิกใหม่โดยใส่ชื่อ, อีเมล และเบอร์โทรศัพท์

3.3.1.1 กรอกหมายเลข 1 เพื่อเพิ่มข้อมูลสมาชิก ดังภาพที่ 3-10



ภาพที่ 3-10 เลือกเมนูหมายเลข 1 เพื่อเพิ่มข้อมูลสมาชิก

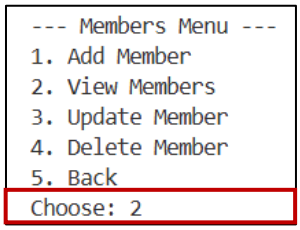
3.3.1.2 จากนั้นกรอกข้อมูลตามที่ปรากฏบนหน้าจอจนครบ ดังภาพที่ 3-11



ภาพที่ 3-11 กรอกข้อมูลเพื่อเพิ่มข้อมูลสมาชิก

3.3.2 ดูข้อมูลสมาชิกทั้งหมด (View Members) : แสดงรายการสมาชิกทั้งหมด พร้อมสถานะและจำนวนครั้งที่ยืม

3.3.2.1 กรอกหมายเลข 2 เพื่อเพิ่มข้อมูลสมาชิก ดังภาพที่ 3-12



ภาพที่ 3-12 เลือกเมนูหมายเลข 2 เพื่อดูข้อมูลสมาชิกทั้งหมด

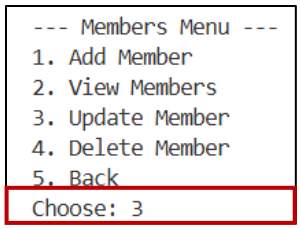
3.3.2.2 จากนั้นจะปรากฏข้อมูลสมาชิกทั้งหมดในรูปแบบตาราง ดังภาพที่ 3-13

MemberID	Name	Email	Phone	Status	TotalBorrows
1	Ribeena	ribeena@email.com	0987654321	Active	3
2	Karit	karit@email.com	0938271654	Active	8
3	Charisa	charisa@email.com	0987321654	Active	2
4	jam	jam@email.com	0333333333	Active	2
5	Wang Jackson	jackson@email.com	0918237645	Active	0

ภาพที่ 3-13 ตารางแสดงข้อมูลสมาชิกทั้งหมด

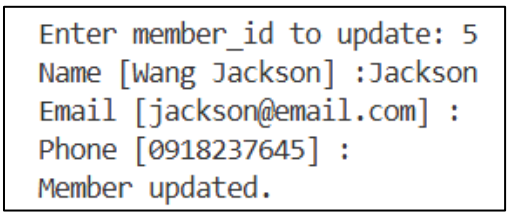
3.3.3 แก้ไขข้อมูลสมาชิก (Update Member) : แก้ไขข้อมูลสมาชิกที่มีอยู่

3.3.3.1 กรอกหมายเลข 3 เพื่อแก้ไขข้อมูลสมาชิก ดังภาพที่ 3-12



ภาพที่ 3-13 เลือกเมนูหมายเลข 3 เพื่อแก้ไขข้อมูลสมาชิก

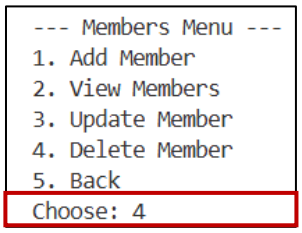
3.3.3.2 จากนั้นกรอกรหัสสมาชิกที่ต้องการแก้ไขข้อมูล ดังภาพที่ 3-14



ภาพที่ 3-14 กรอกรหัสสมาชิกเพื่อแก้ไขข้อมูลสมาชิกที่ต้องการ

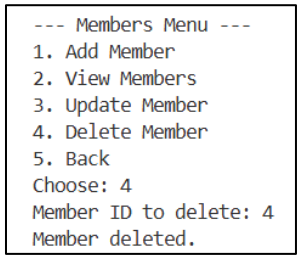
3.3.4 ลบข้อมูลสมาชิก (Delete Member) : เปลี่ยนสถานะสมาชิกเป็น Blocked

3.3.4.1 กรอกหมายเลข 4 เพื่อลบข้อมูลสมาชิก ดังภาพที่ 3-15



ภาพที่ 3-15 เลือกเมนูหมายเลข 4 เพื่อลบข้อมูลสมาชิก

3.3.4.2 จากนั้นกรอกรหัสสมาชิกที่ต้องการลบข้อมูล ดังภาพที่ 3-16

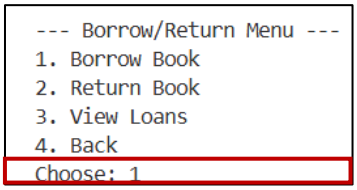


ภาพที่ 3-16 กรอกรหัสสมาชิกเพื่อลบข้อมูลสมาชิกที่ต้องการ

3.4 การใช้งานเมนูยืม-คืนหนังสือ (Borrow/Return)

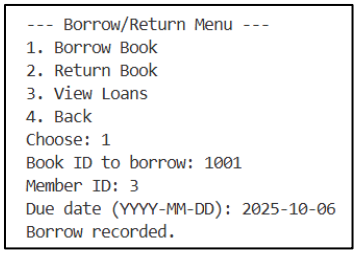
3.4.1 ยืมหนังสือ (Borrow Book) : ระบุรหัสหนังสือและรหัสสมาชิก

3.4.1.1 กรอกหมายเลข 1 เพื่อยืมหนังสือ ดังภาพที่ 3-17



ภาพที่ 3-17 เลือกเมนูหมายเลข 1 เพื่อยืมหนังสือ

3.4.1.2 จากนั้นกรอกข้อมูลตามที่ปรากฏบนหน้าจอจนครบ ดังภาพที่ 3-18



ภาพที่ 3-18 กรอกข้อมูลยืมหนังสือ

3.4.2 คืนหนังสือ (Return Book) : ระบุรหัสรายการยืม (Borrow ID) ที่ต้องการคืน

3.4.2.1 กรอกหมายเลข 2 เพื่อคืนหนังสือ ดังภาพที่ 3-19

```
--- Borrow/Return Menu ---
1. Borrow Book
2. Return Book
3. View Loans
4. Back
Choose: 2
```

ภาพที่ 3-19 เลือกเมนูหมายเลข 2 เพื่อคืนหนังสือ

3.4.2.2 จากนั้นกรอกรหัสการยืมหนังสือที่ต้องการคืน ดังภาพที่ 3-20

```
--- Borrow/Return Menu ---
1. Borrow Book
2. Return Book
3. View Loans
4. Back
Choose: 2
Borrow ID to return: 5
Return success.
```

ภาพที่ 3-20 กรอกรหัสการยืมหนังสือที่ต้องการคืน

3.4.2.3 กรอกหมายเลข 3 เพื่อดูรายการยืม-คืนหนังสือ ดังภาพที่ 3-21

```
--- Borrow/Return Menu ---
1. Borrow Book
2. Return Book
3. View Loans
4. Back
Choose: 3
```

ภาพที่ 3-21 เลือกเมนูหมายเลข 3 เพื่อดูรายการยืม-คืนหนังสือ

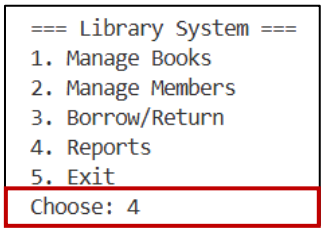
3.4.2.4 จากนั้นจะปรากฏรายการยืม-คืนหนังสือทั้งหมดในรูปแบบตาราง ดังภาพที่ 3-22

BorrowID	BookID	BookTitle	MemberID	MemberName	LoanDate	DueDate	ReturnDate	Status
1	1001	Python	1	Ribeena	2025-10-01	2025-10-04		Borrowed
2	1003	AI	2	Karit	2025-10-01	2025-10-06	2025-10-01	Returned
3	1005	HTML	3	Charisa	2025-10-01	2025-10-09	2025-10-01	Returned
4	1001	Python	4	jam	2025-10-01	2025-10-06	2025-10-01	Returned
5	1002	Linux	4	jam	2025-10-01	2025-10-06	2025-10-01	Returned
6	1004	Java	3	Charisa	2025-10-01	2025-10-5		Borrowed
7	1007	Physical	1	Ribeena	2025-10-01	2025-10-09		Borrowed
8	1008	youtube	1	Ribeena	2025-10-01	2025-10-04	2025-10-01	Returned
9	1009	DOS	2	Karit	2025-10-01	2025-10-06	2025-10-01	Returned
10	1001	Python	2	Karit	2025-10-01	2025-10-10		Borrowed
11	1002	Linux	2	Karit	2025-10-01	2025-10-15		Borrowed
12	1004	Java	2	Karit	2025-10-01	2025-10-03	2025-10-01	Returned
13	1005	HTML	2	Karit	2025-10-01	2025-10-16	2025-10-01	Returned
14	1007	Physical	2	Karit	2025-10-01	2025-10-13	2025-10-01	Returned
15	1008	youtube	2	Karit	2025-10-01	2025-10-09	2025-10-01	Returned
16	1001	Python	3	Charisa	2025-10-01	2025-10-06		Borrowed

ภาพที่ 3-22 ตารางแสดงรายการยืม-คืนหนังสือทั้งหมด

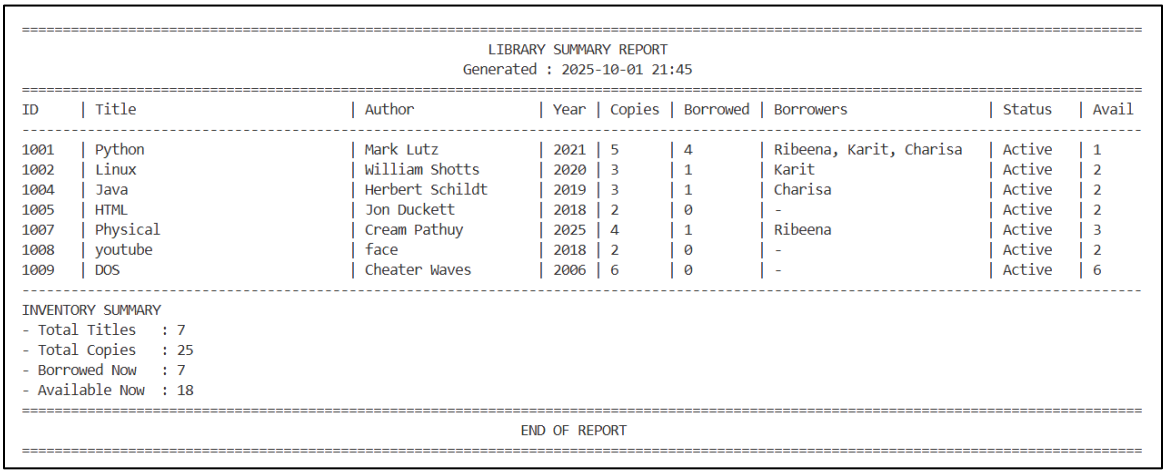
3.5 การใช้งานเมนูรายงานสรุป (Reports)

3.5.1 กรอกหมายเลข 4 ที่ส่วนของเมนูหลัก เพื่อดูรายงานสรุป ดังภาพที่ 3-23



ภาพที่ 3- 23 เลือกเมนูหมายเลข 4 เพื่อดูรายงานสรุป

3.5.2 จากนั้นจะปรากฏรายงานสรุปในรูปแบบตาราง ดังภาพที่ 3-24



LIBRARY SUMMARY REPORT
Generated : 2025-10-01 21:45

ID	Title	Author	Year	Copies	Borrowed	Borrowers	Status	Avail
1001	Python	Mark Lutz	2021	5	4	Ribeena, Karit, Charisa	Active	1
1002	Linux	William Shotts	2020	3	1	Karit	Active	2
1004	Java	Herbert Schildt	2019	3	1	Charisa	Active	2
1005	HTML	Jon Duckett	2018	2	0	-	Active	2
1007	Physical	Cream Pathuy	2025	4	1	Ribeena	Active	3
1008	youtube	face	2018	2	0	-	Active	2
1009	DOS	Cheater Waves	2006	6	0	-	Active	6

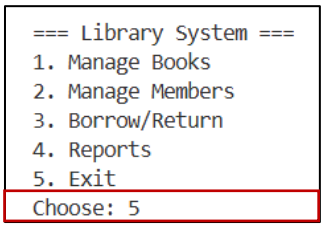
INVENTORY SUMMARY
- Total Titles : 7
- Total Copies : 25
- Borrowed Now : 7
- Available Now : 18

END OF REPORT

ภาพที่ 3-24 ตารางแสดงรายงานสรุป

3.6 การออกจากระบบ

3.6.1 กรอกหมายเลข 5 ที่ส่วนของเมนูหลัก เพื่อออกจากระบบ ดังภาพที่ 3-25



ภาพที่ 3-25 เลือกเมนูหมายเลข 5 เพื่อออกจากระบบ

บทที่ 4
อธิบายการทำงานของโค้ด

4.1 ฟังก์ชันไบนารีพื้นฐานในระบบยืม-คืนหนังสือ

4.1.1 Import Libraries ที่จำเป็นต่อการทำงานของระบบ โดยแต่ละตัวทำหน้าที่ที่แตกต่างกันไปดังนี้ struct ทำหน้าที่ แปลงข้อมูลระหว่าง Python dictionary (dict) กับ bytes fixed-length เพื่อให้เก็บในไฟล์ binary ได้, os ทำหน้าที่ ตรวจสอบว่าไฟล์มีอยู่ไหม, อ่าน/เขียนไฟล์, sys ทำหน้าที่ ช่วยให้ออกจากโปรแกรมอย่างปลอดภัย, datetime ทำหน้าที่ คืนวันที่/เวลาปัจจุบัน (today() / report) ดังภาพที่ 4-1

```
import struct, os, sys
from datetime import datetime
```

ภาพที่ 4-1 Import libraries

4.1.2 กำหนดชื่อไฟล์ข้อมูลที่ใช้สำหรับเก็บข้อมูลระบบ เพื่อให้ง่ายต่อการแก้ไขและเรียกใช้งาน โดยที่ตัวแปรใช้เก็บชื่อไฟล์ binary ของแต่ละ entity ดังภาพที่ 4-2

```
BOOK_FILE = "books.dat"
MEMBER_FILE = "members.dat"
LOAN_FILE = "loans.dat"
```

ภาพที่ 4-2 การกำหนดชื่อไฟล์ให้กับแต่ละเอนทิตี้

4.1.3 กำหนดรูปแบบข้อมูลของฟิลด์ในไฟล์ไบนารี ตามการใช้งานในการเก็บข้อมูลของระบบ โดยที่ในระบบยืม-คืนหนังสือนี้จะมีการกำหนดรูปแบบให้กับฟิลด์ข้อมูลหนังสือ (books) ,สมาชิก (members), การยืม-คืนหนังสือ (loans) ในแต่ละส่วนแตกต่างกันซึ่งแต่ละตัวจะถูกกำหนดให้กับฟิลด์ที่ใช้ในการเก็บข้อมูลดังนี้ BOOK_STRUCT เป็นการกำหนดรูปแบบให้กับฟิลด์ book_id, title, author, year, copies, borrowed, status ตามลำดับ, MEMBER_STRUCT เป็นการกำหนดรูปแบบให้กับฟิลด์ member_id, name, email, phone, status, total_borrows ตามลำดับ, LOAN_STRUCT เป็นการกำหนดรูปแบบให้กับฟิลด์ borrow_id, book_id, member_id, loan_date, due_date, return_date, status ตามลำดับ โดยที่ i หมายถึง Integer , s หมายถึง String และเลขข้างหน้า s หมายถึง จำนวนไบต์ของ String นั้นๆ ดังภาพที่ 4-3

```
BOOK_STRUCT = struct.Struct("i50s30siiii")
MEMBER_STRUCT = struct.Struct("i40s40s15sii")
LOAN_STRUCT = struct.Struct("iii10s10s10si")
```

ภาพที่ 4-3 การกำหนดรูปแบบข้อมูลให้กับฟิลด์ไฟล์ไบนารี

4.1.4 กำหนดรูปแบบของการ pack/unpack ข้อมูล โดยที่การ pack จะเป็นการ แปลง String เป็น ไบต์แบบ fixed length จากนั้นตัดส่วนที่เกินและเติม \x00 ส่วน unpack จะเป็นการแปลง ไบต์ ให้เป็น String จากนั้นลบ padding \x00 ดังภาพที่ 4-4

```
def pack_str(s: str, length: int) -> bytes:
    return s.encode("utf-8")[:length].ljust(length, b"\x00")

def unpack_str(b: bytes) -> str:
    return b.decode("utf-8", errors="ignore").rstrip("\x00")
```

ภาพที่ 4-4 การกำหนดรูปแบบของการ pack/unpack ข้อมูล

4.1.5 การใช้งาน pack/unpack สำหรับไฟล์ต่างๆ โดยที่ไฟล์ books นั้น จะมีการทำงานดังนี้ pack_book(d) : รับข้อมูลหนังสือในรูปแบบ dict แล้วแปลงเป็นไบต์ แบบ fixed-length ตาม BOOK_STRUCT เพื่อเขียนลงไฟล์ไบนารี ส่วน unpack_book(raw) : รับ raw bytes (จำนวนไบต์เท่ากับขนาด struct) แล้วแปลงกลับเป็น dict ที่อ่านง่ายใน Python ดังภาพที่ 4-5 และในส่วนของ members และ loans ก็มีการใช้งานในรูปแบบเดียวกัน

```
def pack_book(d):
    return BOOK_STRUCT.pack(
        int(d["book_id"]),
        pack_str(d.get("title",""),50),
        pack_str(d.get("author",""),30),
        int(d.get("year",0)),
        int(d.get("copies",0)),
        int(d.get("borrowed",0)),
        int(d.get("status",1))
    )

def unpack_book(raw):
    r = BOOK_STRUCT.unpack(raw)
    return {
        "book_id": r[0],
        "title": unpack_str(r[1]),
        "author": unpack_str(r[2]),
        "year": r[3],
        "copies": r[4],
        "borrowed": r[5],
        "status": r[6]
    }
```

ภาพที่ 4-5 การใช้งาน pack/unpack กับไฟล์ไบนารี

4.1.6 ฟังก์ชัน read_all ทำหน้าที่อ่านไฟล์แบบ binary (rb) แล้วแบ่งข้อมูลออกเป็นก้อน (chunk) ตามขนาด size ที่กำหนด จากนั้นเก็บไว้ใน list แล้วส่งกลับออกมา ดังภาพที่ 4-6

```
def read_all(filename, size):
    if not os.path.exists(filename):
        return []
    items = []
    with open(filename,"rb") as f:
        while True:
            chunk = f.read(size)
            if not chunk or len(chunk) < size:
                break
            items.append(chunk)
    return items
```

ภาพที่ 4-6 ฟังก์ชัน read_all

4.1.7 ฟังก์ชัน load_books ใช้ read_all เพื่ออ่านไฟล์ BOOK_FILE ออกมาเป็น chunks ที่มีขนาดเท่ากับ BOOK_STRUCT.size จากนั้นวนลูปเอาแต่ละ chunk ไปแปลงกลับเป็น object หรือ dictionary ของหนังสือ ด้วย unpack_book และ คืนค่าเป็น list ของหนังสือทั้งหมด ดังภาพที่ 4-7

```
def load_books():
    chunks = read_all(BOOK_FILE, BOOK_STRUCT.size)
    return [unpack_book(c) for c in chunks]
```

ภาพที่ 4-7 ฟังก์ชัน load_books

4.1.8 ฟังก์ชัน save_books เปิดไฟล์ BOOK_FILE ในโหมดเขียนทับ (wb) ลบของเก่าแล้วเขียนใหม่ทั้งหมด จากนั้นวนลูป list_books แล้วแปลงหนังสือแต่ละเล่มด้วย pack_book(b) ดังภาพที่ 4-8

```
def save_books(list_books):
    with open(BOOK_FILE,"wb") as f:
        for b in list_books:
            f.write(pack_book(b))
```

ภาพที่ 4-8 ฟังก์ชัน save_books

4.1.9 ฟังก์ชัน append_book_record เปิดไฟล์ BOOK_FILE ในโหมด append binary (ab) แปลงข้อมูลหนังสือ b ด้วย pack_book แล้วเขียนต่อท้ายไฟล์ (เพิ่มหนังสือใหม่ทีละ record โดยไม่กระทบข้อมูลเดิม) ดังภาพที่ 4-9

```
def append_book_record(b):
    with open(BOOK_FILE,"ab") as f:
        f.write(pack_book(b))
```

ภาพที่ 4-9 ฟังก์ชัน append_book_record

4.1.10 ฟังก์ชัน load_members เรียก read_all เพื่ออ่านไฟล์ MEMBER_FILE ซึ่งขนาดการอ่านแต่ละครั้งเท่ากับ MEMBER_STRUCT.size (ขนาด 1 record ของสมาชิก) จะได้ผลเป็น chunks ของข้อมูลดิบ แล้วเอาแต่ละ chunk ไปถอดรหัสเป็น object หรือ dict ด้วย unpack_member แล้วคืนค่าเป็น list ของสมาชิกทั้งหมด ดังภาพที่ 4-10

```
def load_members():
    chunks = read_all(MEMBER_FILE, MEMBER_STRUCT.size)
    return [unpack_member(c) for c in chunks]
```

ภาพที่ 4-10 ฟังก์ชัน load_members

4.1.11 ฟังก์ชัน save_members เปิดไฟล์สมาชิกใหม่ในโหมดเขียนทับ (wb) จากนั้นวนลูป list_members แล้วใช้ pack_member(m) แปลง object เป็น bytes ที่ตรงกับ struct ดังภาพที่ 4-11

```
def save_members(list_members):
    with open(MEMBER_FILE,"wb") as f:
        for m in list_members:
            f.write(pack_member(m))
```

ภาพที่ 4-11 ฟังก์ชัน save_members

4.1.12 ฟังก์ชัน append_member_record เปิดไฟล์สมาชิกในโหมดต่อท้าย (ab) แล้วเขียนข้อมูลสมาชิกใหม่ m ที่ถูกแปลงเป็น bytes ด้วย pack_member ต่อท้ายไฟล์ ดังภาพที่ 4-12

```
def append_member_record(m):
    with open(MEMBER_FILE,"ab") as f:
        f.write(pack_member(m))
```

ภาพที่ 4-12 ฟังก์ชัน append_member_record

4.1.13 ฟังก์ชัน load_loans ใช้ read_all อ่านไฟล์ LOAN_FILE ออกมาเป็นก้อนๆ (chunk) ตามขนาดที่ LOAN_STRUCT.size กำหนด จากนั้นนำแต่ละ chunk ไปถอดรหัสด้วย unpack_loan และคืนค่าเป็น list ของข้อมูลการยืมทั้งหมด ดังภาพที่ 4-13

```
def load_loans():
    chunks = read_all(LOAN_FILE, LOAN_STRUCT.size)
    return [unpack_loan(c) for c in chunks]
```

ภาพที่ 4-13 ฟังก์ชัน load_loans

4.1.14 ฟังก์ชัน save_loans เปิดไฟล์ LOAN_FILE ในโหมดเขียนทับ (wb) จากนั้นวนลูป list_loans แล้วแปลง loan แต่ละอันเป็น bytes ด้วย pack_loan(l) ดังภาพที่ 4-14

```
def save_loans(list_loans):
    with open(LOAN_FILE,"wb") as f:
        for l in list_loans:
            f.write(pack_loan(l))
```

ภาพที่ 4-14 ฟังก์ชัน save_loans

4.1.15 ฟังก์ชัน append_loan_record เปิดไฟล์ LOAN_FILE ในโหมดต่อท้าย (ab) จากนั้นแปลงข้อมูลการยืมใหม่ด้วย pack_loan(l) แล้วเขียนต่อท้ายไฟล์ (เพิ่มข้อมูลการยืมใหม่ลงไฟล์ โดยไม่กระทบข้อมูลเดิม) ดังภาพที่ 4-15

```
def append_loan_record(l):
    with open(LOAN_FILE,"ab") as f:
        f.write(pack_loan(l))
```

ภาพที่ 4-15 ฟังก์ชัน append_loan_record

4.1.16 ฟังก์ชัน next_id ใช้สร้าง รหัสใหม่ (id) โดยดูจาก list ของ items ที่มีอยู่แล้ว ถ้า items ว่างให้เริ่มที่ 1 ถ้ามีข้อมูลอยู่แล้วให้หาค่า id สูงสุด (max(i[key])) แล้วบวกเพิ่ม 1 ดังภาพที่ 4-16

```
def next_id(items, key):
    if not items:
        return 1
    return max(i[key] for i in items) + 1
```

ภาพที่ 4-16 ฟังก์ชัน next_id

4.1.17 ฟังก์ชัน input_int ใช้รับค่าจากผู้ใช้ (input) ที่ต้องเป็นตัวเลข (int) ถ้าใส่ว่าง ("") และมีค่า default กำหนดไว้ให้คืนค่า default ถ้าใส่ค่าไม่ใช่ตัวเลขจะแจ้งเตือนแล้วให้ใส่ใหม่จนกว่าจะถูก ดังภาพที่ 4-17

```
def input_int(prompt, default=None):
    while True:
        s = input(prompt).strip()
        if s == "" and default is not None:
            return default
        try:
            return int(s)
        except:
            print("Please enter a valid number.")
```

ภาพที่ 4-17 ฟังก์ชัน input_int

4.1.18 ฟังก์ชัน input_str ใช้รับข้อความ (string) จากผู้ใช้ ถ้าผู้ใช้กด Enter โดยไม่พิมพ์อะไรให้ใช้ค่า default แทน ดังภาพที่ 4-18

```
def input_str(prompt, default=""):
    s = input(prompt)
    return s if s != "" else default
```

ภาพที่ 4-18 ฟังก์ชัน input_str

4.1.19 ฟังก์ชัน today คืนค่าวันที่ปัจจุบันในรูปแบบ YYYY-MM-DD ดังภาพที่ 4-19

```
def today():
    return datetime.now().strftime("%Y-%m-%d")
```

ภาพที่ 4-19 ฟังก์ชัน today

4.2 ฟังก์ชันเมนูระบบยืม-คืนหนังสือ

4.2.1 ฟังก์ชัน add_book ทำหน้าที่เพิ่มหนังสือใหม่เข้าสู่ระบบ โดยเริ่มจากการโหลดรายการหนังสือทั้งหมดจากไฟล์เก็บข้อมูล จากนั้นถามผู้ใช้ว่าจะเพิ่มหนังสือกี่เล่ม ถ้าไม่กรอกจะถือว่าเพิ่ม 1 เล่มเป็นค่าเริ่มต้น ภายในลูปแต่ละรอบจะให้ผู้ใช้กรอกรหัสหนังสือ และตรวจสอบไม่ให้ซ้ำกับรหัสที่มีอยู่แล้ว หากซ้ำจะให้กรอกใหม่จนกว่าจะไม่ซ้ำ เมื่อได้รหัสที่ถูกต้องแล้วจึงให้กรอกข้อมูลชื่อหนังสือ ผู้แต่ง ปีที่พิมพ์ และจำนวนเล่มที่มี โดยค่าปีที่พิมพ์หากไม่กรอกจะเป็น 0 และจำนวนเล่มหากไม่กรอกจะเป็น 1 หลังจากนั้นจะสร้างข้อมูลหนังสือเป็นดิกชันนารี พร้อมกำหนดค่าเริ่มต้นว่ามีจำนวนที่ถูกยืมเป็น 0 และสถานะเป็น 1 หมายถึงใช้งานได้ แล้วบันทึกข้อมูลนี้ต่อท้ายไฟล์จริงด้วย append_book_record พร้อมทั้งเพิ่มเข้าไปในรายการหนังสือที่เก็บไว้ในหน่วยความจำ และเมื่อเสร็จสิ้นการเพิ่มแต่ละเล่มจะแสดงข้อความแจ้งว่าการเพิ่มสำเร็จ ดังภาพที่ 4-20

```python
def add_book():
    books = load_books()
    count = input_int("How many books do you want to add? ", default=1)

    for idx in range(1, count+1):
        print(f"\n--- Adding book {idx} of {count} ---")
        while True:
            bid = input_int("Book ID: ")
            if any(b["book_id"] == bid for b in books):
                print("This Book ID already exists. Please enter a different ID.")
            else:
                break

        title = input_str("Title: ")
        author = input_str("Author: ")
        year = input_int("Year: ", default=0)
        copies = input_int("Copies: ", default=1)

        b = {
            "book_id": bid,
            "title": title,
            "author": author,
            "year": year,
            "copies": copies,
            "borrowed": 0,
            "status": 1
        }
        append_book_record(b)
        books.append(b)
        print(f"Book added successfully.")
```

ภาพที่ 4-20 ฟังก์ชัน add_book

4.2.2 ฟังก์ชัน view_books ทำหน้าที่แสดงรายการหนังสือทั้งหมดที่มีอยู่ในระบบ โดยเริ่มจากการโหลดข้อมูลหนังสือด้วย load_books ถ้าไม่มีหนังสือเลยก็จะแสดงข้อความว่า "No books." แล้วจบการทำงานทันที แต่ถ้ามีหนังสืออยู่จะเตรียมรูปแบบการแสดงผลด้วย header_fmt และ row_fmt ซึ่งกำหนดความกว้างของแต่ละคอลัมน์ เช่น ID 6 ตัวอักษร Title 30 ตัวอักษร Author 20 ตัวอักษร และ Year, Copies, Status ตามลำดับ จากนั้นพิมพ์หัวตารางและเส้นคั่นเพื่อความชัดเจน แล้ววนลูปทีละหนังสือเพื่อพิมพ์รายละเอียด โดยจะตัดข้อความชื่อหนังสือให้ไม่เกิน 30 ตัวอักษร และชื่อผู้แต่งไม่เกิน 20 ตัวอักษร พร้อมทั้งตรวจสอบสถานะ ถ้า status เท่ากับ 1 จะพิมพ์เป็น "Active" ถ้าไม่ใช่จะพิมพ์เป็น "Inactive" เมื่อแสดงข้อมูลครบทุกเล่มแล้วจะพิมพ์เส้นคั่นอีกครั้งเพื่อปิดท้ายตาราง ดังภาพที่ 4-21

```python
def view_books():
    books = load_books()
    if not books:
        print("No books.")
        return

    header_fmt = "{:<6} | {:<30} | {:<20} | {:<4} | {:<6} | {:<8}"
    row_fmt    = header_fmt

    print()
    print(header_fmt.format("ID","Title","Author","Year","Copies","Status"))
    print("-"*90)

    for b in books:
        status = "Active" if b['status']==1 else "Inactive"
        print(row_fmt.format(
            b['book_id'],
            b['title'][:30],
            b['author'][:20],
            b['year'],
            b['copies'],
            status
        ))

    print("-"*90)
```

ภาพที่ 4-21 ฟังก์ชัน view_books

4.2.3 ฟังก์ชัน update_book ทำหน้าที่แก้ไขข้อมูลหนังสือที่มีอยู่ในระบบ โดยเริ่มจากการโหลดข้อมูลหนังสือทั้งหมดด้วย load_books จากนั้นให้ผู้ใช้ป้อนรหัสหนังสือ (book_id) ที่ต้องการแก้ไข ระบบจะค้นหาหนังสือในรายการทีละเล่ม ถ้าพบรหัสที่ตรงกัน ระบบจะแสดงข้อความว่า "Leave blank to keep current." เพื่อบอกว่าผู้ใช้สามารถกด Enter เพื่อคงค่าข้อมูลเดิมไว้ได้ จากนั้นจะแสดงค่าปัจจุบันในวงเล็บเหลี่ยม เช่น Title [เดิม] : เพื่อให้ผู้ใช้ป้อนข้อมูลใหม่ หากผู้ใช้กรอกค่าลงไปก็จะถูกแทนที่ข้อมูลเก่า แต่ถ้าปล่อยว่างไว้จะใช้ค่าดั้งเดิมแทน โดยฟิลด์ที่สามารถแก้ไขได้ ได้แก่ ชื่อเรื่อง ผู้แต่ง ปีที่พิมพ์ และจำนวนเล่ม เมื่อแก้ไขเสร็จจะอัปเดตค่ากลับไปยังตำแหน่งเดิมในลิสต์ และเมื่อวนเสร็จระบบจะบันทึกข้อมูลที่เปลี่ยนแปลงด้วย save_books พร้อมแสดงข้อความว่า "Book updated." แต่ถ้าไม่พบหนังสือที่มีรหัสตรงกับที่ป้อนเข้ามา ระบบจะแสดงข้อความว่า "Not found." ดังภาพที่ 4-21

```python
def update_book():
    books = load_books()
    bid = input_int("Enter book_id to update: ")
    found = False
    for i,b in enumerate(books):
        if b["book_id"] == bid:
            found = True
            print("Leave blank to keep current.")
            title = input_str(f"Title [{b['title']}] : ", default=b['title'])
            author = input_str(f"Author [{b['author']}] : ", default=b['author'])
            year = input_int(f"Year [{b['year']}] : ", default=b['year'])
            copies = input_int(f"Copies [{b['copies']}] : ", default=b['copies'])
            b["title"]=title; b["author"]=author; b["year"]=year; b["copies"]=copies
            books[i]=b
            break
    if found:
        save_books(books)
        print("Book updated.")
    else:
        print("Not found.")
```

ภาพที่ 4-21 ฟังก์ชัน update_book

4.2.4 ฟังก์ชัน delete_book ทำหน้าที่ลบหนังสือออกจากระบบ โดยเริ่มจากการโหลดข้อมูลหนังสือทั้งหมดด้วย load_books และโหลดข้อมูลการยืมด้วย load_loans จากนั้นรับรหัสหนังสือ (book_id) ที่ต้องการลบจากผู้ใช้ เมื่อได้รหัสมาแล้ว ระบบจะตรวจสอบก่อนว่ามีการยืมหนังสือเล่มนั้นที่ยังไม่คืนอยู่หรือไม่ โดยดูจากตารางการยืมที่มี status เท่ากับ 0 ถ้ามีการยืมค้างอยู่ ระบบจะแสดงข้อความว่า "Cannot delete this book. It is currently borrowed by someone." และหยุดการทำงานทันที แต่ถ้าไม่มีการยืมค้างอยู่ ระบบจะค้นหาหนังสือในรายการทั้งหมดทีละเล่ม หากพบหนังสือที่มีรหัสตรงกัน ระบบจะตรวจสอบต่อว่าหนังสือเล่มนี้ถูกลบไปแล้วหรือไม่ (status = 0) ถ้าลบไปแล้วจะแจ้งว่า "This book has already been deleted (Inactive). Cannot delete again." และหยุดการทำงาน แต่ถ้ายังไม่ถูกลบ ระบบจะเปลี่ยนสถานะของหนังสือจาก 1 (Active) ให้เป็น 0 (Inactive) แล้วอัปเดตข้อมูลกลับไปยังรายการ จากนั้นบันทึกข้อมูลใหม่ด้วย save_books และแสดงข้อความว่า Book deleted (status set to Inactive)." แต่ถ้าค้นหาหนังสือแล้วไม่พบรหัสที่ตรงกันเลย ระบบจะแสดงข้อความว่า "Not found." ดังภาพที่ 4-22

```python
def delete_book():
    books = load_books()
    loans = load_loans()
    bid = input_int("Enter book_id to delete: ")

    active_loans = [l for l in loans if l["book_id"] == bid and l["status"] == 0]
    if active_loans:
        print("Cannot delete this book. It is currently borrowed by someone.")
        return

    found = False
    for i, b in enumerate(books):
        if b["book_id"] == bid:
            if b["status"] == 0:
                print("This book has already been deleted (Inactive). Cannot delete again.")
                return
            b["status"] = 0   # mark inactive
            books[i] = b
            found = True
            break

    if found:
        save_books(books)
        print("Book deleted (status set to Inactive).")
    else:
        print("Not found.")
```

ภาพที่ 4-22 ฟังก์ชัน delete_book

4.2.5 ฟังก์ชัน add_member ทำหน้าที่เพิ่มสมาชิกใหม่เข้าสู่ระบบ โดยเริ่มจากการโหลดข้อมูลสมาชิกทั้งหมดด้วย load_members จากนั้นถามผู้ใช้ว่าต้องการเพิ่มกี่คน ถ้าไม่ตอบจะใช้ค่าเริ่มต้นคือ 1 คน ระบบจะวนลูปตามจำนวนที่ผู้ใช้กำหนดเพื่อเพิ่มสมาชิกทีละคน โดยในแต่ละรอบจะสร้างรหัสสมาชิกใหม่ (member_id) ด้วย next_id ซึ่งจะให้ค่าต่อเนื่องจากรหัสที่มีอยู่แล้ว จากนั้นให้ผู้ใช้ป้อนข้อมูลของสมาชิก ได้แก่ ชื่อ อีเมล และเบอร์โทรศัพท์ เมื่อได้ข้อมูลครบ ระบบจะสร้าง dictionary สำหรับเก็บข้อมูลสมาชิกใหม่ โดยใส่รายละเอียดทั้งหมดพร้อมทั้งกำหนดค่า status เป็น 1 (หมายถึง Active หรือใช้งานได้) และ total_borrows เป็น 0 (ยังไม่เคยยืมหนังสือ) จากนั้นบันทึกข้อมูลสมาชิกใหม่ลงไฟล์ด้วย append_member_record และเพิ่มเข้าไปในรายการ members พร้อมแสดงข้อความยืนยันว่าได้เพิ่มสมาชิกสำเร็จ โดยบอกหมายเลขรหัสสมาชิกที่ถูกสร้างขึ้น ดังภาพที่ 4-23

```python
def add_member():
    members = load_members()
    count = input_int("How many members do you want to add? ", default=1)

    for idx in range(1, count+1):
        print(f"\n--- Adding member {idx} of {count} ---")
        mid = next_id(members, "member_id")
        name = input_str("Name: ")
        email = input_str("Email: ")
        phone = input_str("Phone: ")

        m = {
            "member_id": mid,
            "name": name,
            "email": email,
            "phone": phone,
            "status": 1,
            "total_borrows": 0
        }
        append_member_record(m)
        members.append(m)
        print(f"Added member id {mid}")
```

ภาพที่ 4-23 ฟังก์ชัน add_member

4.2.6 ฟังก์ชัน view_members ทำหน้าที่แสดงรายชื่อสมาชิกทั้งหมดในระบบ โดยเริ่มจากการโหลดข้อมูลสมาชิกด้วย load_members ถ้าไม่พบสมาชิกเลย ระบบจะแสดงข้อความว่า "No members." แล้วจบการทำงานทันที แต่ถ้ามีข้อมูลสมาชิกอยู่ จะเตรียมรูปแบบการแสดงผลด้วย header_fmt และ row_fmt ซึ่งกำหนดความกว้างของแต่ละคอลัมน์ เช่น MemberID 8 ตัวอักษร, Name 30 ตัวอักษร, Email 30 ตัวอักษร, Phone 12 ตัวอักษร, Status 8 ตัวอักษร และ TotalBorrows 12 ตัวอักษร จากนั้นพิมพ์หัวตารางและเส้นคั่น ต่อมาจะวนลูปแสดงสมาชิกทีละคน โดยตัดชื่อและอีเมลให้ไม่เกิน 30 ตัวอักษร ตรวจสอบสถานะ ถ้า status เท่ากับ 1 จะแสดงเป็น "Active" แต่ถ้าไม่ใช่จะแสดงเป็น "Blocked" จากนั้นพิมพ์รายละเอียดทั้งหมดของสมาชิกตามรูปแบบที่กำหนดไว้ และเมื่อแสดงครบทุกคนแล้วจะพิมพ์เส้นคั่นอีกครั้งเพื่อปิดท้ายตาราง ดังภาพที่ 4-24

```python
def view_members():
    members = load_members()
    if not members:
        print("No members.")
        return

    header_fmt = "{:<8} | {:<30} | {:<30} | {:<12} | {:<8} | {:<12}"
    row_fmt    = header_fmt

    print()
    print(header_fmt.format("MemberID","Name","Email","Phone","Status","TotalBorrows"))
    print("-"*113)

    for m in members:
        status = "Active" if m["status"]==1 else "Blocked"
        print(row_fmt.format(
            m['member_id'], m['name'][:30], m['email'][:30],
            m['phone'], status, m['total_borrows']
        ))
    print("-"*113)
```

ภาพที่ 4-24 view_members

4.2.7 ฟังก์ชัน update_member ทำหน้าที่แก้ไขข้อมูลสมาชิกที่มีอยู่ในระบบ โดยเริ่มจากการโหลดข้อมูลสมาชิกทั้งหมดด้วย load_members จากนั้นรับหมายเลขสมาชิก (member_id) ที่ต้องการแก้ไขจากผู้ใช้ ระบบจะค้นหาสมาชิกในรายการทีละคน หากพบว่ามีรหัสตรงกัน จะให้ผู้ใช้ป้อนข้อมูลใหม่สำหรับ ชื่อ อีเมล และเบอร์โทรศัพท์ โดยจะแสดงค่าปัจจุบันไว้ในวงเล็บเหลี่ยม เช่น Name [เดิม] : เพื่อบอกว่าถ้ากด Enter โดยไม่พิมพ์อะไร ระบบจะเก็บค่าดั้งเดิมเอาไว้ แต่ถ้ามีการป้อนข้อมูลใหม่ก็จะนำมาแทนที่ข้อมูลเก่า เมื่อแก้ไขข้อมูลเรียบร้อยแล้ว จะบันทึกการเปลี่ยนแปลงกลับเข้าสู่รายการสมาชิก และเซฟลงไฟล์ด้วย save_members พร้อมแสดงข้อความว่า "Member updated." แต่ถ้าค้นหาไม่พบสมาชิกที่มีรหัสตรงกับที่ป้อนเข้ามา ระบบจะแสดงข้อความว่า "Not found." ดังภาพที่ 2-25

```python
def update_member():
    members = load_members()
    mid = input_int("Enter member_id to update: ")
    found=False
    for i,m in enumerate(members):
        if m["member_id"]==mid:
            found=True
            name = input_str(f"Name [{m['name']}] : ", default=m['name'])
            email = input_str(f"Email [{m['email']}] : ", default=m['email'])
            phone = input_str(f"Phone [{m['phone']}] : ", default=m['phone'])
            m['name']=name; m['email']=email; m['phone']=phone
            members[i]=m
            break
    if found:
        save_members(members)
        print("Member updated.")
    else:
        print("Not found.")
```

ภาพที่ 2-25 ฟังก์ชัน update_member

4.2.8 ฟังก์ชัน delete_member ทำหน้าที่ลบสมาชิกออกจากระบบ (ในที่นี้คือเปลี่ยนสถานะเป็นลบ ไม่ได้ลบข้อมูลจริง) โดยเริ่มจากการโหลดข้อมูลสมาชิกทั้งหมดด้วย load_members จากนั้นรับหมายเลขสมาชิก (member_id) ที่ต้องการลบจากผู้ใช้ ระบบจะค้นหาสมาชิกในรายการทีละคน หากพบว่ามีรหัสสมาชิกตรงกัน จะเปลี่ยนค่าฟิลด์ "status" ของสมาชิกนั้นเป็น 0 เพื่อระบุว่าสมาชิกถูกลบ จากนั้นบันทึกข้อมูลสมาชิกทั้งหมดกลับลงไฟล์ด้วย save_members และแสดงข้อความว่า "Member deleted." ถ้าไม่พบสมาชิกที่มีรหัสตรงกับที่ป้อนเข้ามา ระบบจะแสดงข้อความว่า "Not found." ดังภาพที่ 4-16

```python
def delete_member():
    members = load_members()
    mid = input_int("Member ID to delete: ")
    found=False
    for i,m in enumerate(members):
        if m["member_id"]==mid:
            m["status"]=0
            members[i]=m
            found=True
            break
    if found:
        save_members(members)
        print("Member deleted.")
    else:
        print("Not found.")
```

ภาพที่ 4-26 ฟังก์ชัน delete_member

4.2.9 ฟังก์ชัน borrow_book ทำหน้าที่บันทึกการยืมหนังสือของสมาชิกในระบบ โดยเริ่มจากการโหลดข้อมูลหนังสือ สมาชิก และรายการยืมทั้งหมดด้วย load_books, load_members และ load_loans ตามลำดับ จากนั้นรับหมายเลขหนังสือ (book_id) และหมายเลขสมาชิก (member_id) ที่ต้องการยืม จากนั้นระบบจะตรวจสอบว่าหนังสือมีอยู่และสามารถยืมได้ (status == 1) และว่ามีสำเนาว่างสำหรับยืมหรือไม่ (borrowed < copies) รวมถึงตรวจสอบว่าสมาชิกมีสถานะใช้งานได้ (status == 1) ถ้าไม่ผ่านเงื่อนไขใด ๆ ระบบจะแสดงข้อความแจ้ง เช่น "Book not available.", "Member not valid." หรือ "No available copies." แล้วออกจากฟังก์ชัน ถ้าข้อมูลถูกต้อง ระบบจะสร้างบันทึกการยืมใหม่ โดยกำหนดรหัสยืม (borrow_id) วันที่ยืม (loan_date) และวันที่กำหนดคืน (due_date) แล้วบันทึกลงรายการยืมด้วย append_loan_record หลังจากนั้นจะอัปเดตสถานะหนังสือ โดยเพิ่มจำนวนครั้งที่ถูกยืม (borrowed += 1) และบันทึกกลับไปยังไฟล์ด้วย save_books พร้อมกับอัปเดตข้อมูลสมาชิก เพิ่มจำนวนครั้งที่สมาชิกยืมหนังสือทั้งหมด (total_borrows += 1) และบันทึกกลับด้วย save_members เมื่อบันทึกสำเร็จ ระบบจะแสดงข้อความ "Borrow recorded." เพื่อยืนยันว่าการยืมเสร็จสมบูรณ์ ดังภาพที่ 4-27

```
def borrow_book():
    books = load_books()
    members = load_members()
    loans = load_loans()
    book_id = input_int("Book ID to borrow: ")
    member_id = input_int("Member ID: ")
    book = next((b for b in books if b["book_id"]==book_id and b["status"]==1), None)
    member = next((m for m in members if m["member_id"]==member_id and m["status"]==1), None)
    if not book:
        print("Book not available.")
        return
    if not member:
        print("Member not valid.")
        return
    if book["borrowed"] >= book["copies"]:
        print("No available copies.")
        return
    borrow_id = next_id(loans, "borrow_id")
    loan_date = today()
    due_date = input_str("Due date (YYYY-MM-DD): ", default="")
    loan = {
        "borrow_id": borrow_id,
        "book_id": book_id,
        "member_id": member_id,
        "loan_date": loan_date,
        "due_date": due_date,
        "return_date": "",
        "status": 0  # 0 = Borrowed
    }
    append_loan_record(loan)
    # update book
    for i,b in enumerate(books):
        if b["book_id"]==book_id:
            b["borrowed"] += 1
            books[i]=b
            break
    save_books(books)
    # update member total borrows
    for i,m in enumerate(members):
        if m["member_id"]==member_id:
            m["total_borrows"] += 1
            members[i]=m
            break
    save_members(members)
    print("Borrow recorded. " \
    "")
```

ภาพที่ 4-27 ฟังก์ชัน borrow_book

4.2.10 ฟังก์ชัน return_book ทำหน้าที่บันทึกการคืนหนังสือของสมาชิก โดยเริ่มจากโหลดข้อมูลการยืมหนังสือ และสมาชิกทั้งหมด จากนั้นรับรหัสการยืม (borrow_id) ที่ต้องการคืนระบบจะค้นหารายการยืมที่ตรงกับรหัสและยังอยู่ในสถานะยืม (status == 0) หากพบ จะบันทึกวันที่คืน (return_date) และเปลี่ยนสถานะเป็นคืนแล้ว (status = 1) พร้อมอัปเดตจำนวนหนังสือที่ถูกยืม (borrowed) ของหนังสือให้น้อยลง แต่ไม่ต่ำกว่าหลังจากอัปเดตแล้ว จะบันทึกข้อมูลกลับไปยังไฟล์ด้วย save_loans และ save_books แล้วแสดงข้อความ "Return success." ถ้าไม่พบรายการยืมที่ตรงตามเงื่อนไข ระบบจะแสดงข้อความ "Active loan not found." ดังภาพที่ 4-28

```
def return_book():
    loans = load_loans()
    books = load_books()
    members = load_members()
    borrow_id = input_int("Borrow ID to return: ")
    found=False
    for i,l in enumerate(loans):
        if l["borrow_id"]==borrow_id and l["status"]==0:
            found=True
            l["return_date"] = today()
            l["status"] = 1  # returned
            loans[i]=l
            # update book borrowed
            for j,b in enumerate(books):
                if b["book_id"]==l["book_id"]:
                    b["borrowed"] = max(0, b["borrowed"] - 1)
                    books[j]=b
                    break
            break
    if found:
        save_loans(loans)
        save_books(books)
        print("Return success.")
    else:
        print("Active loan not found.")
```

ภาพที่ 4-28 ฟังก์ชัน return_book

4.2.11 ฟังก์ชัน view_loans ทำหน้าที่แสดงรายการยืมหนังสือทั้งหมดในระบบ โดยเริ่มจากโหลดข้อมูลการยืม หนังสือ และสมาชิกถ้าไม่มีรายการยืม ระบบจะแสดงข้อความ "No loans." แล้วออกจากฟังก์ชัน จากนั้นจะสร้าง map ของรหัสหนังสือกับชื่อหนังสือ และรหัสสมาชิกกับชื่อสมาชิก เพื่อให้สามารถแสดงชื่อแทนรหัสได้ ต่อมาแสดงตารางรายการยืม โดยมีคอลัมน์ BorrowID, BookID, BookTitle, MemberID, MemberName, LoanDate, DueDate, ReturnDate, Status และในแต่ละแถวจะแสดงข้อมูลการยืม พร้อมสถานะ "Borrowed" ถ้ายังไม่คืน หรือ "Returned" ถ้าคืนแล้ว ดังภาพที่ 4-29

```python
def view_loans():
    loans = load_loans()
    books = load_books()
    members = load_members()

    if not loans:
        print("No loans.")
        return

    book_map = {b["book_id"]: b["title"] for b in books}
    member_map = {m["member_id"]: m["name"] for m in members}

    header_fmt = "{:<8} | {:<6} | {:<30} | {:<8} | {:<25} | {:<10} | {:<10} | {:<10} | {:<10}"
    row_fmt    = header_fmt

    print()
    print(header_fmt.format(
        "BorrowID", "BookID", "BookTitle", "MemberID", "MemberName",
        "LoanDate", "DueDate", "ReturnDate", "Status"
    ))
    print("-"*140)

    for l in loans:
        status = "Borrowed" if l["status"]==0 else "Returned"
        book_title = book_map.get(l["book_id"], f"Book{l['book_id']}")
        member_name = member_map.get(l["member_id"], f"Member{l['member_id']}")

        print(row_fmt.format(
            l['borrow_id'],
            l['book_id'], book_title[:30],
            l['member_id'], member_name[:25],
            l['loan_date'], l['due_date'], l['return_date'], status
        ))

    print("-"*140)
```

ภาพที่ 4-29 ฟังก์ชัน view_loans

4.3 เมนูทั้งหมดที่ใช้ในระบบยืม-คืนหนังสือ

4.3.1 ฟังก์ชัน show_summary_report ทำหน้าที่แสดง รายงานสรุปสถานะห้องสมุด โดยเริ่มจากโหลดข้อมูลหนังสือ สมาชิก และรายการยืมทั้งหมด ซึ่งระบบจะนับสถิติพื้นฐานของหนังสือ เช่น จำนวนหัวข้อหนังสือทั้งหมด (Total Titles), จำนวนสำเนาทั้งหมด (Total Copies), จำนวนหนังสือที่กำลังถูกยืม (Borrowed Now) และจำนวนสำเนาที่ว่าง (Available Now) จากนั้นจะแสดงตารางรายละเอียดหนังสือแต่ละเล่ม โดยแสดงคอลัมน์ ID, Title, Author, Year, Copies, Borrowed, Borrowers, Status, Avail สำหรับหนังสือที่มีสถานะใช้งาน (Active) และตรวจสอบว่ามีผู้ยืมหนังสือเล่มนั้น ๆ กี่คน เพื่อแสดงชื่อสมาชิกที่กำลังยืม ถ้าไม่มีผู้ยืมจะแสดงเป็น "-" สุดท้ายจะแสดง Reports พร้อมวันที่และเวลาที่สร้างรายงาน และปิดด้วยข้อความ "END OF REPORT" เพื่อให้เห็นภาพรวมของสถานะหนังสือทั้งหมดในห้องสมุด ทั้งรายละเอียดแต่ละเล่มและสถิติรวม ดังภาพที่ 4-30

```python
def show_summary_report():
    books = load_books()
    members = load_members()
    loans = load_loans()
    member_map = {m["member_id"]: m["name"] for m in members}

    total_titles = 0
    total_copies = 0
    total_borrowed_now = 0

    now = datetime.now().strftime("%Y-%m-%d %H:%M")
    print("\n" + "="*137)
    print("LIBRARY SUMMARY REPORT".center(135))
    print(f"Generated : {now}".center(135))
    print("="*137)

    header_fmt = "{:<6} | {:<30} | {:<20} | {:<4} | {:<6} | {:<8} | {:<25} | {:<8} | {:<5}"
    row_fmt    = header_fmt

    print(header_fmt.format("ID","Title","Author","Year","Copies","Borrowed","Borrowers","Status","Avail"))
    print("-"*137)

    for b in books:
        if b["status"] != 1:
            continue
        total_titles += 1
        total_copies += b["copies"]

        active_loans = [ln for ln in loans if ln["book_id"]==b["book_id"] and ln["status"]==0]
        borrower_names = [member_map.get(ln["member_id"], f"Member{ln['member_id']}") for ln in active_loans]
        borrowers_display = ", ".join(borrower_names) if borrower_names else "-"

        borrowed_count = b["borrowed"]
        total_borrowed_now += borrowed_count
        avail = b["copies"] - borrowed_count
        status = "Active" if b["status"]==1 else "Inactive"

        print(row_fmt.format(
            b['book_id'], b['title'][:30], b['author'][:20], b['year'],
            b['copies'], borrowed_count, borrowers_display[:25], status, avail
        ))

    print("-"*137)
    print("INVENTORY SUMMARY")
    print(f"- Total Titles   : {total_titles}")
    print(f"- Total Copies   : {total_copies}")
    print(f"- Borrowed Now   : {total_borrowed_now}")
    print(f"- Available Now  : {total_copies - total_borrowed_now}")
    print("="*137)
    print("END OF REPORT".center(135))
    print("="*137 + "\n")
```

ภาพที่ 4-30 ฟังก์ชัน show_summary_report

4.3.2 ฟังก์ชัน books_menu ทำหน้าที่แสดงเมนูจัดการหนังสือ โดยให้ผู้ใช้เลือกดำเนินการ เช่น เพิ่มหนังสือ ดูรายการหนังสือ แก้ไข หรือลบหนังสือ ระบบจะวนลูปรอคำสั่งจนกว่าผู้ใช้จะเลือกกลับไปยังเมนูก่อนหน้า และจะแสดงข้อความ "Invalid" หากป้อนตัวเลือกไม่ถูกต้อง ดังภาพที่ 4-31

```python
def books_menu():
    while True:
        print("\n--- Books Menu ---")
        print("1. Add Book")
        print("2. View Books")
        print("3. Update Book")
        print("4. Delete Book")
        print("5. Back")
        c = input("Choose: ").strip()
        if c=="1": add_book()
        elif c=="2": view_books()
        elif c=="3": update_book()
        elif c=="4": delete_book()
        elif c=="5": break
        else: print("Invalid")
```

ภาพที่ 4-31 ฟังก์ชัน books_menu

4.3.3 ฟังก์ชัน members_menu ทำหน้าที่แสดงเมนูจัดการสมาชิก โดยให้ผู้ใช้เลือกดำเนินการ เช่น เพิ่มสมาชิก ดูรายการสมาชิก แก้ไข หรือลบสมาชิก ระบบจะวนลูปรอคำสั่งจนกว่าผู้ใช้จะเลือกกลับไปยังเมนูก่อนหน้า และจะแสดงข้อความ "Invalid" หากป้อนตัวเลือกไม่ถูกต้อง ดังภาพที่ 4-32

```python
def members_menu():
    while True:
        print("\n--- Members Menu ---")
        print("1. Add Member")
        print("2. View Members")
        print("3. Update Member")
        print("4. Delete Member")
        print("5. Back")
        c = input("Choose: ").strip()
        if c=="1": add_member()
        elif c=="2": view_members()
        elif c=="3": update_member()
        elif c=="4": delete_member()
        elif c=="5": break
        else: print("Invalid")
```

ภาพที่ 4-32 ฟังก์ชัน members_menu

4.3.4 ฟังก์ชัน loans_menu ทำหน้าที่แสดงเมนูยืมและคืนหนังสือ โดยให้ผู้ใช้เลือกดำเนินการ เช่น ยืมหนังสือ คืนหนังสือ หรือดูรายการยืม ระบบจะวนลูปรอคำสั่งจนกว่าผู้ใช้จะเลือกกลับไปยังเมนูก่อนหน้า และจะแสดงข้อความ "Invalid" หากป้อนตัวเลือกไม่ถูกต้อง ดังภาพที่ 4-33

```python
def loans_menu():
    while True:
        print("\n--- Borrow/Return Menu ---")
        print("1. Borrow Book")
        print("2. Return Book")
        print("3. View Loans")
        print("4. Back")
        c = input("Choose: ").strip()
        if c=="1": borrow_book()
        elif c=="2": return_book()
        elif c=="3": view_loans()
        elif c=="4": break
        else: print("Invalid")
```

ภาพที่ 4-33 ฟังก์ชัน loans_menu

4.3.5 ฟังก์ชัน main_menu ทำหน้าที่แสดงเมนูหลักของระบบห้องสมุด โดยให้ผู้ใช้เลือกดำเนินการ เช่น จัดการหนังสือ จัดการสมาชิก ยืม/คืนหนังสือ หรือดูรายงาน ระบบจะวนลูปรอคำสั่งจนกว่าผู้ใช้จะเลือกออก (Exit) และจะแสดงข้อความ "Invalid choice" หากป้อนตัวเลือกไม่ถูกต้อง ดังภาพที่ 4-34

```python
def main_menu():
    while True:
        print("\n=== Library System ===")
        print("1. Manage Books")
        print("2. Manage Members")
        print("3. Borrow/Return")
        print("4. Reports")
        print("5. Exit")
        c = input("Choose: ").strip()
        if c=="1": books_menu()
        elif c=="2": members_menu()
        elif c=="3": loans_menu()
        elif c=="4": show_summary_report()
        elif c=="5":
            print("Bye.")
            sys.exit(0)
        else:
            print("Invalid choice")
```

ภาพที่ 4-34 ฟังก์ชัน main_menu

บทที่ 5
สรุปผลการดำเนินงานและสรุปผล

5.1 สรุปผลการดำเนินงาน

จากการพัฒนาโครงงานระบบยืม–คืนหนังสือ ผู้จัดทำสามารถสร้างระบบที่มีฟังก์ชันการทำงานหลักได้ครบถ้วน ไม่ว่าจะเป็นการจัดการข้อมูลหนังสือ การจัดการข้อมูลสมาชิก การบันทึกการยืมและการคืนหนังสือ รวมถึงการออกรายงานสรุปผลการทำงานของระบบ ระบบสามารถทำงานได้จริงตามวัตถุประสงค์ที่ตั้งไว้ และช่วยให้ผู้จัดทำได้ฝึกทักษะการวิเคราะห์ การออกแบบ และการเขียนโปรแกรมด้วยภาษา Python ตลอดจนการจัดการข้อมูลในรูปแบบ Binary File ซึ่งถือเป็นการประยุกต์ใช้ความรู้จากการเรียนมาอย่างเป็นรูปธรรม

5.2 ปัญหาและอุปสรรคในการดำเนินงาน

แม้ว่าระบบจะสามารถพัฒนาได้สำเร็จ แต่ก็พบปัญหาและอุปสรรคบางประการ เช่น การจัดเก็บข้อมูลด้วย Binary File มีข้อจำกัดในการแก้ไขหรืออัปเดต เนื่องจากเมื่อมีการเปลี่ยนแปลงข้อมูลจำเป็นต้องอ่านและเขียนไฟล์ใหม่เกือบทั้งหมด อีกทั้งการออกแบบฟังก์ชันการแปลงข้อมูล (pack/unpack) ต้องทำอย่างถูกต้องตามโครงสร้าง มิฉะนั้นจะไม่สามารถอ่านข้อมูลได้ นอกจากนี้ยังมีปัญหาในด้านการตรวจสอบความถูกต้องของข้อมูลที่ผู้ใช้กรอกเข้ามา ซึ่งต้องทำอย่างรอบคอบเพื่อป้องกันข้อผิดพลาด และระบบยังคงเป็นแบบ Command Line Interface (CLI) ที่อาจไม่สะดวกต่อผู้ใช้ทั่วไปที่คุ้นเคยกับระบบแบบกราฟิก

5.3 ข้อเสนอแนะ

เพื่อให้ระบบมีความสมบูรณ์และใช้งานได้จริงในอนาคต ผู้จัดทำมีข้อเสนอแนะดังนี้ ควรเปลี่ยนจากการใช้ Binary File ไปใช้ฐานข้อมูล เช่น MySQL หรือ SQLite เพื่อเพิ่มประสิทธิภาพในการจัดการข้อมูล ควรพัฒนาระบบให้มีส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) หรือทำเป็นเว็บแอปพลิเคชันเพื่อให้ผู้ใช้เข้าถึงได้สะดวกมากขึ้น นอกจากนี้ควรเพิ่มฟังก์ชันการแจ้งเตือนกำหนดคืนหนังสือผ่านอีเมลหรือข้อความเพื่อช่วยลดปัญหาการคืนล่าช้า ควรกำหนดสิทธิ์ผู้ใช้งานตามบทบาท เช่น บรรณารักษ์และสมาชิก เพื่อให้ระบบมีความปลอดภัยมากขึ้น และควรพัฒนาให้ระบบสามารถเก็บประวัติการใช้งานเพื่อใช้วิเคราะห์แนวโน้มการยืมหนังสือและวางแผนการจัดการทรัพยากรห้องสมุดได้อย่างมีประสิทธิภาพ