Lending and Returning Library (ระบบยืม–คืนหนังสือ)

จัดทำโดย

กณวรรธน์ เดชแสง	6806022510076
พิยดา มานพ	6706022610144
ภูริณัฐ วรศรี	6706022510301

คณะเทคโนโลยีการจัดการอุตสาหกรรม

สาขาวิศวกรรมสารสนเทศและเครือข่าย

เสนอ

อาจารย์ อนิราช มิ่งขวัญ

รายงานเล่นนี้เป็นส่วนหนึ่งของวิชา compro

รหัสวิชา 060233115

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ภาคการศึกษา 1/2567

หัวข้อโครงงาน ระบบยืม–คืนหนังสือ

นักศึกษา กณวรรธน์ เดชแสง

พิยดา มานพ

ภูริณัฐ วรศรี

พ.ศ. 2568

อาจารย์ที่ปรึกษาโครงงาน รองศาสตราจารย์ ดร.อนิราช มิ่งขวัญ

บทคัดย่อ

โครงงานเรื่อง “Lending and Returning Library” จัดทำขึ้นเพื่อพัฒนาระบบต้นแบบสำหรับการบริหารจัดการการยืม–คืนหนังสือในห้องสมุด โดยใช้ภาษา Python เป็นเครื่องมือหลักในการพัฒนา จุดประสงค์ของโครงงานนี้คือเพื่อแก้ไขปัญหาที่มักเกิดขึ้นจากการจัดการข้อมูลแบบดั้งเดิม เช่น การบันทึกด้วยเอกสารที่อาจสูญหาย การตรวจสอบสถานะหนังสือที่ล่าช้า และความผิดพลาดในการคำนวณค่าปรับ ระบบที่พัฒนาขึ้นสามารถจัดเก็บข้อมูลหนังสือ ข้อมูลสมาชิก และประวัติการยืม–คืนได้อย่างมีระเบียบ พร้อมทั้งสามารถคำนวณค่าปรับโดยอัตโนมัติหากมีการคืนหนังสือเกินกำหนด

ระบบที่พัฒนาขึ้นรองรับการทำงานเบื้องต้น ได้แก่ การเพิ่ม ลบ และค้นหาหนังสือ การบันทึกข้อมูลสมาชิกใหม่ การบันทึกการยืม–คืน รวมถึงการแสดงรายงานสรุปผลการยืม–คืนหนังสือ ปัจจุบันระบบเก็บข้อมูลในรูปแบบไฟล์ (.dat) ซึ่งง่ายต่อการใช้งานและเหมาะสมกับการศึกษาเบื้องต้น อีกทั้งยังสามารถต่อยอดไปสู่การพัฒนาเชื่อมต่อฐานข้อมูลหรือระบบออนไลน์ได้ในอนาคต

ผลการดำเนินงานพบว่า ระบบสามารถทำงานได้ตามที่ตั้งเป้าไว้ ช่วยลดความซ้ำซ้อนและข้อผิดพลาดในการจัดการข้อมูลห้องสมุดได้อย่างมีประสิทธิภาพ อีกทั้งยังเป็นการฝึกทักษะด้านการเขียนโปรแกรม การวิเคราะห์ระบบ และการแก้ปัญหาด้วยวิธีการเชิงโปรแกรมให้แก่ผู้พัฒนา ซึ่งสามารถนำไปประยุกต์ใช้ในการเรียนการสอนหรือต่อยอดเป็นระบบห้องสมุดดิจิทัลที่สมบูรณ์มากยิ่งขึ้นในอนาคต

Project Title	**Lending and Returning Library**
Project Members	**Kanawath Dechsang**
	Phiyada Manop
	Phurinat Worasri
A.D.	**2025**

Project Advisor Assoc. Prof. Dr. Anirach Mingkhwan

Abstract

The project **"Lending and Returning Library"** was developed as a prototype system for managing the borrowing and returning of books in a library, using Python as the primary development tool. The main objective of this project is to solve common issues found in traditional library management, such as the loss of paper-based records, delays in checking book availability, and errors in fine calculation. The proposed system can systematically store book information, member details, and borrowing–returning records, as well as automatically calculate fines for overdue returns.

The developed system supports basic operations including adding, deleting, and searching for books, registering new members, recording borrowing and returning transactions, and generating summary reports of library activities. At present, the system stores data in .dat files, which are easy to use and suitable for educational purposes. Moreover, the system can be further enhanced to connect with databases or be expanded into an online application in the future.

The results demonstrate that the system works as intended, effectively reducing redundancy and errors in managing library data. Furthermore, this project provides the developers with valuable experience in programming, system analysis, and problem-solving through computational approaches. It can be applied for educational purposes or serve as a foundation for developing a more comprehensive digital library system in the future.

สารบัญ

บทที่ 1 บทนำ 1

1.1 **ที่มาและความสำคัญของโครงงาน** 1

1.2 **วัตถุประสงค์ของโครงงาน** 2

1.3 **วัตถุประสงค์ของโครงงาน** 2

1.4 **ขอบเขตของโครงงาน** 2

1.5 **ประโยชน์ที่คาดว่าจะได้รับ** 3

บทที่ 2 เอกสารและทฤษฎีที่เกี่ยวข้อง 4

2.1 แนวคิดและทฤษฎีพื้นฐานของระบบห้องสมุดอัตโนมัติ 4

2.1.1 องค์ประกอบหลักของระบบห้องสมุดอัตโนมัติ 4

2.1.2 วงจรการไหลเวียนของทรัพยากร (Circulation Cycle) 4

2.2 ทฤษฎีที่เกี่ยวข้องกับการจัดการโครงสร้างข้อมูลและไฟล์ 5

2.2.1 แนวคิดฐานข้อมูลแบบ Flat-File 5

2.2.2 การออกแบบโครงสร้างข้อมูล (Data Structure Design) 5

2.2.3 ความสัมพันธ์ของข้อมูล (Data Relationship) 6

2.3 **ภาษา** Python **และเทคนิคการเขียนโปรแกรมที่เกี่ยวข้อง** 6

2.3.1 เหตุผลในการเลือกใช้ Python 6

2.3.2 เทคนิคการจัดการไฟล์ (File I/O) 7

2.3.3 การจัดการวันที่และเวลา (Datetime Module) 7

2.4 **ทฤษฎีการจัดการการยืม-คืนและการคำนวณค่าปรับ** 7

2.4.1 การจัดการสถานะหนังสือ (Status Management) 7

2.4.2 หลักการคำนวณค่าปรับ (Fine Calculation Principle) 8

2.5 **งานวิจัยและวรรณกรรมที่เกี่ยวข้อง** 9

2.5.1 โครงงานระบบยืม-คืนหนังสือที่ใช้ Python 9

2.5.2 การศึกษาการใช้งานระบบห้องสมุดดิจิทัล (Digital Library) 9
บทที่ 3 การวิเคราะห์และออกแบบระบบ 10
3.1 **การวิเคราะห์ความต้องการของระบบ (**System Requirement Analysis)..... 10
3.1.1 ความต้องการเชิงหน้าที่ (Functional Requirements)..... 10
3.1.2 ความต้องการเชิงไม่เป็นหน้าที่ (Non-Functional Requirements)..... 10
3.2 **การออกแบบโครงสร้างข้อมูล (**Data Structure Design)..... 11
3.2.1 books.dat (ข้อมูลหนังสือ)..... 11
3.2.2 ไฟล์ members.dat (ข้อมูลสมาชิก)..... 11
3.2.3 ไฟล์ borrows.dat (ข้อมูลการยืม-คืน) 12
3.3 **แผนภาพความสัมพันธ์เชิงเอนทิตี (**Entity-Relationship Diagram: E-R Diagram).. 12
บทที่ 4 อธิบายการทำงานของ Code..... 13
4.1 ไบนารีพื้นฐานในระบบยืม – คืนหนังสือห้องสมุด 17
4.2 ฟังชั่นในระบบยืม – คืนหนังสือห้องสมุด..... 18
บทที่ 5 สรุปผล อภิปรายผล และข้อเสนอแนะ 37
5.1 **สรุปผลการดำเนินโครงงาน**..... 37
5.1.1 สรุปผลตามวัตถุประสงค์ 37
5.2 **อภิปรายผล (**Discussion)..... 38
5.2.1 การอภิปรายผลด้านการจัดการข้อมูล 38
5.2.2 การอภิปรายผลด้านฟังก์ชันหลัก..... 38
5.2.3 ข้อจำกัดของระบบ 39
5.3 **ข้อเสนอแนะและแนวทางการพัฒนาในอนาคต** 39

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันห้องสมุดถือเป็นแหล่งเรียนรู้ที่สำคัญ ทั้งในสถานศึกษาและองค์กรต่าง ๆ โดยมีบทบาทในการเป็นแหล่งเก็บรวบรวมความรู้ หนังสือ เอกสาร และสื่อการเรียนรู้หลากหลายประเภท อย่างไรก็ตาม การจัดการข้อมูลหนังสือและการยืม–คืนยังคงเป็นปัญหาที่พบเจออยู่บ่อยครั้ง โดยเฉพาะห้องสมุดที่ยังใช้วิธีบันทึกข้อมูลด้วยกระดาษหรือแฟ้มเอกสาร ซึ่งอาจทำให้ข้อมูลสูญหาย ค้นหายาก หรือเกิดความผิดพลาดได้ง่าย นอกจากนี้ กระบวนการตรวจสอบสถานะหนังสือและการคำนวณค่าปรับกรณีคืนเกินกำหนดยังใช้เวลามาก และสร้างภาระงานให้แก่บรรณารักษ์

เพื่อแก้ไขปัญหาดังกล่าว จึงได้พัฒนาโครงงาน **“Lending and Returning Library”** โดยใช้ภาษา Python เป็นเครื่องมือหลัก เนื่องจาก Python เป็นภาษาที่มีความยืดหยุ่น ใช้งานง่าย และได้รับความนิยมอย่างแพร่หลายในงานด้านการพัฒนาซอฟต์แวร์ ระบบนี้มุ่งหวังที่จะช่วยให้การจัดการหนังสือ การบันทึกข้อมูลสมาชิก และการยืม–คืนเป็นไปอย่างมีประสิทธิภาพมากขึ้น

ภาพที่ 1.1 ห้องสมุด

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อพัฒนาระบบต้นแบบที่ช่วยจัดการการยืม–คืนหนังสือในห้องสมุดด้วยภาษา Python
2. เพื่อสร้างระบบที่สามารถเก็บข้อมูลหนังสือ ข้อมูลสมาชิก และประวัติการยืม–คืนได้อย่างเป็นระบบ
3. เพื่อลดความซ้ำซ้อน ความล่าช้า และข้อผิดพลาดที่เกิดจากการทำงานแบบเอกสาร
4. เพื่อคำนวณค่าปรับการคืนหนังสือเกินกำหนดโดยอัตโนมัติ
5. เพื่อเป็นต้นแบบในการพัฒนาระบบห้องสมุดดิจิทัลที่สามารถต่อยอดไปสู่การเชื่อมต่อฐานข้อมูลหรือใช้งานในรูปแบบออนไลน์

1.3 วัตถุประสงค์ของโครงงาน

1. เพื่อพัฒนาระบบต้นแบบที่ช่วยจัดการการยืม–คืนหนังสือในห้องสมุดด้วยภาษา Python
2. เพื่อสร้างระบบที่สามารถเก็บข้อมูลหนังสือ ข้อมูลสมาชิก และประวัติการยืม–คืนได้อย่างเป็นระบบ
3. เพื่อลดความซ้ำซ้อน ความล่าช้า และข้อผิดพลาดที่เกิดจากการทำงานแบบเอกสาร
4. เพื่อคำนวณค่าปรับการคืนหนังสือเกินกำหนดโดยอัตโนมัติ
5. เพื่อเป็นต้นแบบในการพัฒนาระบบห้องสมุดดิจิทัลที่สามารถต่อยอดไปสู่การเชื่อมต่อฐานข้อมูลหรือใช้งานในรูปแบบออนไลน์

1.4 ขอบเขตของโครงงาน

โครงงานนี้มีขอบเขตการทำงานดังนี้

- **ข้อมูลหนังสือ**: สามารถเพิ่ม ลบ แก้ไข และค้นหาหนังสือ โดยเก็บข้อมูลเช่น รหัสหนังสือ ชื่อเรื่อง ผู้แต่ง สำนักพิมพ์ หมวดหมู่ ภาษา เลขที่ชั้นวาง และจำนวนคงเหลือ
- **ข้อมูลสมาชิก**: บันทึกข้อมูลสมาชิก เช่น รหัสสมาชิก ชื่อ–นามสกุล วันเกิด เพศ ที่อยู่ เบอร์โทรศัพท์ อีเมล และวันที่สมัคร
- **ข้อมูลการยืม–คืน**: เก็บประวัติการยืม–คืน โดยบันทึกวันที่ยืม วันที่ครบกำหนด วันที่คืนจริง และสถานะ พร้อมทั้งคำนวณค่าปรับ
- **รายงานการยืม–คืน**: ระบบสามารถออกรายงานสรุปการยืม–คืน เพื่อใช้ตรวจสอบและประเมินการใช้งานของสมาชิก

```
Library Borrow System – Summary Report
Generated At : 2025-09-17 19:05 (+07:00)
App Version  : 1.0
Encoding     : UTF-8 (fixed-length)

--------------------------------------------------------------------------------
BookID | Title                | Author     | Year | Copies | Borrowed | Status
--------------------------------------------------------------------------------
2001   | ComPro               | Kanawath   | 2021 | 10     | 1        | Active
2002   | Python Programming   | Unknown    | 2020 | 5      | 1        | Active
2003   | Networking Basics    | Unknown    | 2019 | 3      | 1        | Active
2004   | Database System      | Unknown    | 2022 | 4      | 0        | Active
2005   | Computer Concepts    | Unknown    | 2023 | 2      | 0        | Active
--------------------------------------------------------------------------------

Summary (Active Books Only)
- Total Books   : 5
- Active Books  : 5
- Deleted Books : 0
- Borrowed Now  : 3
- Available Now : 21
```

ภาพที่ 1.2 ตัวอย่างไฟล์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ระบบต้นแบบที่ช่วยจัดการข้อมูลการยืม–คืนหนังสือในห้องสมุดได้สะดวกและรวดเร็ว
2. ลดความผิดพลาดจากการจัดการข้อมูลด้วยวิธีดั้งเดิม
3. ผู้พัฒนามีความรู้ความเข้าใจในการเขียนโปรแกรมด้วยภาษา Python และการออกแบบระบบสารสนเทศ
4. ระบบสามารถนำไปประยุกต์ใช้จริงในห้องสมุดขนาดเล็ก หรือใช้เป็นต้นแบบในการต่อยอดสู่ระบบขนาดใหญ่

บทที่ 2
เอกสารและทฤษฎีที่เกี่ยวข้อง

2.1 แนวคิดและทฤษฎีพื้นฐานของระบบห้องสมุดอัตโนมัติ

2.1.1 องค์ประกอบหลักของระบบห้องสมุดอัตโนมัติ

ระบบยืม–คืน (Circulation) เป็นแกนหลักที่ต้องมีการเชื่อมโยงข้อมูลระหว่าง 3 ส่วนหลัก ดังที่ปรากฏในโครงสร้างไฟล์ข้อมูลของโครงงาน:

1. **การจัดการทรัพยากรสารสนเทศ (Cataloging Module):** เน้นการลงทะเบียนและจัดเก็บข้อมูลรายละเอียดของหนังสือแต่ละรายการ ข้อมูลสำคัญที่ต้องมีการจัดเก็บ ได้แก่

   book_id, title, author, publisher, year_pub, total_copies และสถานะของหนังสือ

2. **การจัดการสมาชิก (Member Module):** เกี่ยวข้องกับการบันทึกและดูแลข้อมูลส่วนตัวของสมาชิกเพื่อระบุตัวตนในการทำรายการยืม-คืน ข้อมูลหลักประกอบด้วย member_id, name_surname, mobile, และ Email_address

3. **การจัดการการยืม-คืน (Circulation Module):** เป็นส่วนที่บันทึกประวัติการทำรายการ โดยเชื่อมโยงรหัสหนังสือ (book_id) และรหัสสมาชิก (member_id) เข้าด้วยกัน เพื่อกำหนด **วันยืม (Date Out)**, **วันครบกำหนด (Date Due)**, และบันทึก **วันที่คืนจริง (Date Return)**

2.1.2 วงจรการไหลเวียนของทรัพยากร (Circulation Cycle)

การจัดการการยืม–คืนครอบคลุมตั้งแต่การตรวจสอบสิทธิ์สมาชิก การตรวจสอบสถานะหนังสือ (Available/Borrowed) การบันทึกการยืม การติดตามเมื่อหนังสือเลยกำหนด (Overdue) ไปจนถึงการคืนหนังสือและการคำนวณค่าปรับ ระบบจะต้องอัปเดตสถานะของหนังสือในตาราง

2.2 ทฤษฎีที่เกี่ยวข้องกับการจัดการโครงสร้างข้อมูลและไฟล์

โครงงานนี้เลือกใช้การจัดเก็บข้อมูลแบบไฟล์ข้อมูลแบน (Flat-File Database) ในรูปแบบ .dat ซึ่งเหมาะสมสำหรับระบบต้นแบบขนาดเล็กที่เน้นความเข้าใจง่ายในการจัดการไฟล์โดยตรงด้วยภาษาโปรแกรม

2.2.1 แนวคิดฐานข้อมูลแบบ Flat-File

Flat-File คือไฟล์ที่เก็บข้อมูลในรูปแบบตาราง โดยแต่ละแถว (Record) แทนรายการข้อมูลหนึ่งรายการ และแต่ละคอลัมน์ (Field) แทนคุณสมบัติหนึ่งของคุณสมบัตินั้น ข้อมูลในแต่ละฟิลด์จะถูกแยกออกจากกันด้วยตัวแบ่ง (Delimiter) หรือถูกกำหนดขนาดคงที่ (Fixed-Length)

2.2.2 การออกแบบโครงสร้างข้อมูล (Data Structure Design)

การออกแบบนี้เป็นไปตามหลักการของโมเดลความสัมพันธ์เชิงเอนทิตี (Entity-Relationship Model) แม้ว่าจะไม่ได้ใช้ระบบฐานข้อมูลเชิงสัมพันธ์เต็มรูปแบบ แต่มีการกำหนด **เอนทิตี (Entity)** และ **ฟิลด์ (Field)** อย่างชัดเจน:

1. เอนทิตีหนังสือ (books.dat)

- **คีย์หลัก (Primary Key):** book_id (Int, 4) ใช้สำหรับระบุหนังสือแต่ละเล่มโดยไม่ซ้ำกัน
- **โครงสร้างข้อมูล:** การกำหนดชนิดและขนาดที่ชัดเจน (เช่น title เป็น str ขนาด 100) ช่วยในการจัดการหน่วยความจำและสร้างความเป็นระเบียบในการจัดเก็บ

2. เอนทิตีสมาชิก (members.dat)

- **คีย์หลัก (Primary Key):** member_id (Int, 4) ใช้สำหรับระบุสมาชิกแต่ละคน
- **ความท้าทายของข้อมูล:** มีการเก็บข้อมูลที่สำคัญต่อการติดต่อ เช่น mobile (Int, 15) และ Email_address (str, 100) เพื่อใช้ในการแจ้งเตือน

3. เอนทิตีการยืม-คืน (borrows.dat)

- **คีย์นอก (Foreign Keys):** ใช้ member_id และ book_id เพื่อเชื่อมโยงไปยังเอนทิตี Member และ Book ตามลำดับ
- **ความสัมพันธ์:** เป็นความสัมพันธ์แบบ **ยืม (Borrow)** หรือ **คืน (Return)** ซึ่งเป็นส่วนที่สำคัญที่สุดในการบันทึกเหตุการณ์และประวัติการทำรายการ

2.2.3 ความสัมพันธ์ของข้อมูล (Data Relationship)

โครงงานนี้มีความสัมพันธ์แบบหนึ่งต่อกลุ่ม (**One-to-Many Relationship**) เช่น:

- **Book ↔ Borrow:** หนังสือ 1 เล่ม (ระบุด้วย book_id) สามารถถูกยืมได้หลายครั้งในตาราง borrows.dat
- **Member ↔ Borrow:** สมาชิก 1 คน (ระบุด้วย member_id) สามารถทำรายการยืม-คืนได้หลายรายการในตาราง borrows.dat

2.3 ภาษา Python และเทคนิคการเขียนโปรแกรมที่เกี่ยวข้อง

โครงงานนี้พัฒนาโดยใช้ภาษา **Python** ซึ่งเป็นภาษาโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ที่เน้นการออกแบบโค้ดให้มีความกระชับและอ่านง่าย

2.3.1 เหตุผลในการเลือกใช้ Python

1. **Readability และ Simplicity:** เหมาะสำหรับการพัฒนาระบบต้นแบบ เนื่องจากมีไวยากรณ์ที่ตรงไปตรงมา ช่วยให้สามารถพัฒนาฟังก์ชันหลักของระบบได้อย่างรวดเร็ว

2. **Standard Library ที่มีประสิทธิภาพ:** มีโมดูลมาตรฐานที่พร้อมใช้งานโดยไม่ต้องติดตั้งไลบรารีภายนอกจำนวนมาก โดยเฉพาะอย่างยิ่งในด้านการจัดการไฟล์ และวันที่/เวลา

2.3.2 เทคนิคการจัดการไฟล์ (File I/O)

หัวใจของการจัดการ Flat-File คือการทำ **File Input/Output (I/O)** ซึ่งในภาษา Python จะใช้คำสั่ง open() ในโหมดต่าง ๆ เช่น:

- **Reading (r):** เพื่ออ่านข้อมูลจากไฟล์ .dat เข้าสู่หน่วยความจำเพื่อประมวลผล
- **Writing (w):** เพื่อสร้างไฟล์ใหม่หรือเขียนทับไฟล์เดิม (ใช้สำหรับการลบข้อมูลทั้งหมด)
- **Appending (a):** เพื่อเพิ่มข้อมูลใหม่ต่อท้ายไฟล์ (ใช้สำหรับการเพิ่มรายการยืม-คืนหรือสมาชิกใหม่)
- **Reading and Writing (r+ หรือ w+):** สำหรับการแก้ไขข้อมูล (Update) ซึ่งต้องอ่านข้อมูลทั้งหมดเข้าสู่หน่วยความจำ แก้ไขรายการที่ต้องการ และเขียนข้อมูลทั้งหมดกลับลงในไฟล์ใหม่

2.3.3 การจัดการวันที่และเวลา (Datetime Module)

เพื่อตอบสนองความต้องการในการคำนวณค่าปรับโดยอัตโนมัติ ภาษา Python มีโมดูล

datetime ที่เป็นเครื่องมือสำคัญ:

1. **การแปลงข้อมูล:** ข้อมูลวันที่ในไฟล์ (2025-09-05) ต้องถูกแปลงจากรูปแบบ String ให้เป็นวัตถุ

datetime เพื่อให้สามารถทำการคำนวณทางคณิตศาสตร์ได้

2. **การคำนวณความแตกต่าง:** สามารถใช้การลบระหว่างวัตถุ datetime สองตัว (เช่น วันที่คืนจริง - วันครบกำหนด) เพื่อให้ได้ผลลัพธ์เป็นวัตถุ timedelta ซึ่งแสดงจำนวนวันที่ล่าช้าไป

2.4 ทฤษฎีการจัดการการยืม-คืนและการคำนวณค่าปรับ

2.4.1 การจัดการสถานะหนังสือ (Status Management)

การจัดการสถานะเป็นฟังก์ชันที่สำคัญในการป้องกันไม่ให้หนังสือที่ถูกยืมไปแล้วถูกยืมซ้ำอีก ระบบต้องมีการตรวจสอบและอัปเดตข้อมูล

จำนวนคงเหลือ โดยอ้างอิงจาก total_copies ใน books.dat และจำนวนรายการยืมที่ **สถานะ (Status)** เป็น Borrow ใน borrows.dat

Available Now=Total Copies−∑Borrowed Now

จากตัวอย่างรายงาน แสดงให้เห็นว่าระบบได้ทำการคำนวณสถานะคงเหลือแล้ว:

- Total Books: 5
- Borrowed Now: 3 (จากรายงานสรุป) และ 4 (จากรายงาน Borrowed Report)
- Available Now: 21 (ค่านี้อาจแสดงผลรวมของจำนวนสำเนาทั้งหมดที่ยังไม่ได้ถูกยืม)

2.4.2 หลักการคำนวณค่าปรับ (Fine Calculation Principle)

การคำนวณค่าปรับมีวัตถุประสงค์เพื่อจูงใจให้สมาชิกนำหนังสือมาคืนตามกำหนด

วันที่ครบกำหนด (Date Due)

เงื่อนไขและสูตรการคำนวณ:

1. **เงื่อนไข:** รายการ borrows.dat จะถูกตรวจสอบเมื่อมีการบันทึก date_return
2. **การตรวจสอบ:** หาก date_return ล่าช้ากว่า date_due จะถือว่าเกิดค่าปรับ
3. **สูตร:**

   Fine Amount=(Date Return−Date Due)×Fine Rate Per Day

   ค่า

   fine_amount ที่บันทึกไว้ในรายงานการยืม (เช่น $20.00, $22.00, $165.50, $40.00) แสดงให้เห็นว่าระบบได้ประมวลผลการคำนวณนี้และมีการจัดเก็บผลลัพธ์ในฟิลด์

   fine_amount ที่เป็นชนิดข้อมูล float 4,2

2.5 งานวิจัยและวรรณกรรมที่เกี่ยวข้อง

การทบทวนงานวิจัยที่เกี่ยวข้องมุ่งเน้นไปที่การพัฒนาระบบห้องสมุดขนาดเล็กและกลาง โดยเฉพาะที่ใช้ภาษาโปรแกรมในการจัดการฐานข้อมูล

2.5.1 โครงงานระบบยืม-คืนหนังสือที่ใช้ Python

งานวิจัยและโครงงานนักศึกษาจำนวนมากได้พัฒนาโปรแกรมจัดการระบบห้องสมุดโดยใช้ภาษา Python ในการจัดการไฟล์ข้อมูล CSV หรือ TXT ซึ่งสอดคล้องกับแนวคิดการใช้ Flat-File ของโครงงานนี้ โครงงานเหล่านี้มักเน้นการสร้างเมนูคำสั่งแบบข้อความ (**Console Application**) ซึ่งง่ายต่อการพัฒนาและทดสอบฟังก์ชันการทำงานหลักของระบบ (CRUD: Create, Read, Update, Delete) โดยตรง

2.5.2 การศึกษาการใช้งานระบบห้องสมุดดิจิทัล (Digital Library)

แม้ว่าโครงงานนี้จะเริ่มต้นด้วยระบบจัดการไฟล์แบบออฟไลน์ แต่มีวัตถุประสงค์เพื่อเป็นต้นแบบในการต่อยอดไปสู่ระบบห้องสมุดดิจิทัลที่สามารถเชื่อมต่อฐานข้อมูล การศึกษาการใช้งาน (Usability Study) ของระบบห้องสมุดดิจิทัลที่มุ่งเน้นประสบการณ์ของผู้ใช้งาน (UX) เช่น การค้นหาที่รวดเร็ว (Searching efficiency) และการแสดงสถานะที่ชัดเจน (Status clarity) จะเป็นแนวทางสำคัญในการพัฒนาต่อยอดในบทถัดไป

บทที่ 3
การวิเคราะห์และออกแบบระบบ

3.1 การวิเคราะห์ความต้องการของระบบ (System Requirement Analysis)

ขั้นตอนนี้เป็นการแปลงวัตถุประสงค์และขอบเขตที่กำหนดให้เป็นคุณสมบัติที่ระบบต้องสามารถปฏิบัติได้จริง เพื่อเป็นพื้นฐานในการออกแบบและพัฒนาโปรแกรม

3.1.1 ความต้องการเชิงหน้าที่ (Functional Requirements)

คือสิ่งที่ระบบต้องทำเพื่อให้ผู้ใช้สามารถดำเนินงานหลักของห้องสมุดได้สำเร็จ

ลำดับ	ฟังก์ชันหลัก (Module)	ฟังก์ชันย่อย (Sub-Function)	อ้างอิงข้อมูล
1	**การจัดการหนังสือ**	1.1 เพิ่มข้อมูลหนังสือใหม่	`books.dat`
		1.2 แก้ไขข้อมูลหนังสือที่มีอยู่	`books.dat`
		1.3 ลบข้อมูลหนังสือ	`books.dat`
		1.4 ค้นหาหนังสือ (เช่น ตามชื่อเรื่อง, ผู้แต่ง, ISBN)	`books.dat`
2	**การจัดการสมาชิก**	2.1 เพิ่มข้อมูลสมาชิกใหม่	`members.dat`
		2.2 แก้ไขข้อมูลสมาชิกที่มีอยู่	`members.dat`
		2.3 ค้นหาข้อมูลสมาชิก (เช่น ตามรหัส, ชื่อ)	`members.dat`
3	**การทำรายการยืม-คืน**	3.1 บันทึกรายการยืมหนังสือ (Date Out, Date Due)	`borrows.dat`
		3.2 บันทึกรายการคืนหนังสือ (Date Return)	`borrows.dat`
		3.3 คำนวณค่าปรับอัตโนมัติหากคืนเกินกำหนด	`borrows.dat`
		3.4 อัปเดตสถานะหนังสือ (Available/Borrowed)	`books.dat`, `borrows.dat`
4	**การสร้างรายงาน**	4.1 รายงานสรุปสถานะหนังสือ (รวม/คงเหลือ/ถูกยืม)	รายงานสรุป
		4.2 รายงานหนังสือที่ถูกยืมอยู่ (Borrowed Report)	รายงานการยืม

3.1.2 ความต้องการเชิงไม่เป็นหน้าที่ (Non-Functional Requirements)

- **ความสามารถในการทำงานร่วมกัน (Compatibility):** ระบบต้องสามารถทำงานได้บนสภาพแวดล้อมที่รองรับภาษา Python และการจัดการไฟล์ .dat
- **ความถูกต้องแม่นยำ (Accuracy):** การคำนวณค่าปรับต้องมีความถูกต้องตามจำนวนวันที่เกินกำหนดและอัตราค่าปรับที่กำหนดไว้

- **ความน่าเชื่อถือ (Reliability):** การบันทึกข้อมูลต้องเป็นไปในรูปแบบ **UTF-8 (fixed-length)** เพื่อให้การจัดเก็บและเรียกใช้ข้อมูลคงที่และเป็นระเบียบตามที่ระบุในไฟล์รายงาน

3.2 **การออกแบบโครงสร้างข้อมูล (**Data Structure Design)

โครงสร้างข้อมูลอ้างอิงตามรูปแบบไฟล์

.dat แบบ **Fixed-Length Encoding** ที่กำหนดขนาดของแต่ละฟิลด์อย่างชัดเจน เพื่อความรวดเร็วในการจัดเก็บและค้นหา

3.2.1 books.dat (ข้อมูลหนังสือ)

ฟิลด์ (Field)	ชนิดข้อมูล	ขนาด (Size)	ตัวอย่าง	หมายเหตุ
book_id	Int	4	2001	คีย์หลัก (Primary Key)
title	String	100	ComPro	
author	String	100	Kanawath	
publisher	String	100	Siam Book Co.	
year_pub	Int	4	2021	
category	String	50	Computer	
language	String	50	Thai	
shelf_no	String	20	A3-15	
total_copies	Int	4	10	จำนวนสำเนาทั้งหมด

3.2.2 ไฟล์ members.dat (ข้อมูลสมาชิก)

ฟิลด์ (Field)	ชนิดข้อมูล	ขนาด (Size)	ตัวอย่าง	หมายเหตุ
member_id	Int	4	1001	คีย์หลัก (Primary Key)
name_surname	String	100	Kana Dech	
birth_date	Date/Int	10	2002-05-12	
gender	String	1	M	
home_address	String	200	1/ Bangkok	
mobile	Int	15	0888888888	
Email_address	String	100	Test@email.com	
reg_date	Date/Int	4	2023-01-01	

3.2.3 ไฟล์ borrows.dat (ข้อมูลการยืม-คืน)

ฟิลด์ (Field)	ชนิดข้อมูล	ขนาด (Size)	ตัวอย่าง	หมายเหตุ
member_id	Int	4	1001	คีย์นอก (Foreign Key) เชื่อมสมาชิก
book_id	Int	4	2001	คีย์นอก (Foreign Key) เชื่อมหนังสือ
date_out	Date/Int	10	2025-09-05	วันที่ยืม
date_due	Date/Int	10	2025-09-05	วันที่ครบกำหนด
date_return	Date/Int	10	2025-09-07	วันที่คืนจริง (เป็นค่าว่างถ้ายังไม่คืน)
status	String	20	Return	สถานะ (Borrow/Return)
fine_amount	Float	4,2	0.00	ค่าปรับที่ถูกคำนวณ
notes	String	200	-	

3.3 **แผนภาพความสัมพันธ์เชิงเอนทิตี (**Entity-Relationship Diagram: E-R Diagram)

แผนภาพ E-R แสดงความสัมพันธ์ระหว่างข้อมูลหลัก 3 เอนทิตี (Entity) ในระบบ

- **Member (E1):** ข้อมูลสมาชิก (**Primary Key: member_id)**
- **Book (E2):** ข้อมูลหนังสือ (**Primary Key: book_id)**
- **Borrow (E3):** ข้อมูลการยืม-คืน (**Composite Key: member_id, book_id, date_out** ใช้ระบุรายการยืมที่ไม่ซ้ำกัน)

ความสัมพันธ์ (Relationship):

1. **Member ↔ Borrow:** เป็นความสัมพันธ์แบบ **หนึ่งต่อกลุ่ม (One-to-Many: 1:N)** สมาชิก **1** คนสามารถทำรายการยืมได้หลายรายการ
2. **Book ↔ Borrow:** เป็นความสัมพันธ์แบบ **หนึ่งต่อกลุ่ม (One-to-Many: 1:N)** หนังสือ **1** เล่มสามารถถูกยืมได้หลายรายการตลอดเวลา

บทที่ 4
อธิบายการทำงานของ Code

```
import struct
import os
import datetime
```

4.1 **ไบนารีพื้นฐานในระบบยืม – คืนหนังสือห้องสมุด**

4.1.1 **Module Struct** เป็นโมดูลใน Python ที่ใช้สำหรับการจัดการข้อมูลแบบไบนารีเช่น การแปลงข้อมูลจากรูปแบบ Python (เช่น string,integer, float) ไปเป็นไบต์ หรือการแปลงข้อมูลจากไบต์กลับมาเป็นรูปแบบ Python อีกครั้ง โมดูลนี้สำคัญเมื่อเราต้องการทำงานกับไฟล์หรือข้อมูลที่อยู่ในรูปแบบไบนารี เช่นไฟล์

4.1.2 **Module os** เป็น **โมดูลมาตรฐานของ Python** ที่ใช้สำหรับ **ทำงานกับระบบปฏิบัติการ (Operating System)** ช่วยให้โปรแกรม Python สามารถจัดการไฟล์ โฟลเดอร์ เส้นทางไฟล์ และเรียกใช้คำสั่งของระบบได้ โดยไม่ต้องพึ่งโปรแกรมภายนอก ทำให้โค้ดของเราสามารถทำงานกับระบบไฟล์หรือระบบปฏิบัติการได้โดยตรง

4.1**.2 Module datetime** เป็น **โมดูลมาตรฐาน** ของ Python ที่ใช้สำหรับ **จัดการวันที่และเวลา** ทำให้สามารถสร้าง ตรวจสอบ คำนวณ และจัดรูปแบบวันที่-เวลาได้อย่างสะดวก

4.1.3 **Struct Definitions**

```
# ---------------- Struct definitions (little-endian '<') ----------------
BOOK_STRUCT = struct.Struct("<i100s100s100si50s50s20si")
MEMBER_STRUCT = struct.Struct("<i100s10s1s200s15s100s10s")
BORROW_STRUCT = struct.Struct("<ii10s10s10s20sf200s")

BOOK_FILE = "books.dat"
MEMBER_FILE = "members.dat"
BORROW_FILE = "borrows.dat"
```

- **Struct** กำหนด **รูปแบบข้อมูลไบนารี**
- < little-endian (ตัวเลขเก็บแบบ byte น้อยไปมาก)
- i integer 4 bytes
- s string ขนาดคงที่ (เช่น 100s = string 100 bytes)
- **ความหมายของแต่ละ struct:**

1. **BOOK_STRUCT** ⟶ ข้อมูลหนังสือ: book_id, title, author, publisher, year_pub, category, language, shelf_no, total_copies
2. **MEMBER_STRUCT** ⟶ ข้อมูลสมาชิก: member_id, name_surname, birth_date, gender, address, mobile, email, reg_date
3. **BORROW_STRUCT** ⟶ ข้อมูลการยืม: member_id, book_id, date_out, date_due, date_return, status, fine_amount, notes

เหตุผล: ใช้ struct เพื่อให้สามารถ **อ่าน/เขียนข้อมูลลงไฟล์ไบนารี** แบบมีขนาดคงที่

4.2 ฟังชั่นในระบบยืม – คืนหนังสือห้องสมุด

4.2.1 pack/unpack strings

```
# ---------------- Helpers: packing/unpacking fixed-length strings ----------------
def pack_str(s: str, length: int) -> bytes:
    if s is None:
        s = ""
    b = s.encode("utf-8")[:length]
    return b.ljust(length, b' ')

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

- **pack_str** ⟶ แปลง string ⟶ bytes fixed-length
    - ตัด string ถ้ายาวเกิน length
    - เติมช่องว่าง (padding) ถ้าสั้น
- **unpack_str** ⟶ แปลง bytes ⟶ string
    - ลบช่องว่างและ null padding

เหตุผล: เพราะ struct ต้องการ **byte ขนาดคงที่** ทำให้เขียนไฟล์ไบนารีง่ายและอ่านกลับได้ถูกต้อง

```
def get_int(prompt: str, minv=None, maxv=None) -> int:
    while True:
        v = input(prompt).strip()
        if v == "":
            print("กรุณากรอกข้อมูล")
            continue
        try:
            n = int(v)
        except ValueError:
            print(" ต้องกรอกตัวเลขจำนวนเต็มเท่านั้น – ลองใหม่")
            continue
        if minv is not None and n < minv:
            print(f" ค่าต้อง >= {minv}")
            continue
        if maxv is not None and n > maxv:
            print(f" ค่าต้อง <= {maxv}")
            continue
        return n
```

4.2.2 get_int

- รับค่า integer จากผู้ใช้

- ตรวจสอบว่ากรอกตัวเลขหรือไม่

- ตรวจสอบช่วง minv และ maxv
- **เหตุผล:** ป้องกัน input ผิดพลาด

```python
def get_float(prompt: str, minv=None, maxv=None) -> float:
    while True:
        v = input(prompt).strip()
        if v == "":
            print("กรุณากรอกข้อมูล")
            continue
        try:
            f = float(v)
        except ValueError:
            print(" ต้องกรอกตัวเลข (ทศนิยมได้) เท่านั้น — ลองใหม่")
            continue
        if minv is not None and f < minv:
            print(f" ค่าต้อง >= {minv}")
            continue
        if maxv is not None and f > maxv:
            print(f" ค่าต้อง <= {maxv}")
            continue
        return f
```

4.2.3 get_float

- รับค่า float
- ตรวจสอบ input และช่วงค่า

```python
def get_date(prompt: str) -> str:
    while True:
        s = input(prompt).strip()
        try:
            datetime.datetime.strptime(s, "%Y-%m-%d")
            return s
        except Exception:
            print(" รูปแบบวันที่ต้องเป็น YYYY-MM-DD เช่น 2025-09-07")
```

4.2.4 get_date

- รับวันที่ในรูปแบบ YYYY-MM-DD
- ตรวจสอบรูปแบบถูกต้องด้วย datetime.strptime

```python
def get_str(prompt: str, maxlen: int, allow_empty=False) -> str:
    while True:
        s = input(prompt).rstrip()
        if not allow_empty and s == "":
            print(" ห้ามเว้นว่าง — ลองใหม่")
            continue
        if len(s.encode("utf-8")) > maxlen:
            print(f" ความยาวเกิน {maxlen} bytes — จะถูกตัด")
            s = s[:maxlen]
        return s
```

4.2.5 get_str

- รับ string
- ตรวจสอบความยาว byte
- allow_empty ⟶ กำหนดว่าข้อมูลว่างได้หรือไม่

เหตุผล: ฟังก์ชันเหล่านี้ช่วย **ลด error จากผู้ใช้** และทำให้ข้อมูลที่เก็บลงไฟล์มีรูปแบบถูกต้อง

```python
def add_record(filename: str, st: struct.Struct, packed_tuple: tuple):
    os.makedirs(os.path.dirname(filename) or ".", exist_ok=True)
    with open(filename, "ab") as f:
        f.write(st.pack(*packed_tuple))
```

4.2.6 add_record

- เปิดไฟล์ filename แบบ append ("ab")
- เขียนข้อมูลที่ pack แล้วลงไฟล์

```python
def read_raw_records(filename: str, st: struct.Struct):
    records = []
    if not os.path.exists(filename):
        return records
    size = st.size
    with open(filename, "rb") as f:
        while True:
            chunk = f.read(size)
            if not chunk:
                break
            if len(chunk) != size:
                break
            records.append(st.unpack(chunk))
    return records
```

4.2.7 read_raw_records

- อ่านไฟล์ทีละ record ขนาด st.size

- แปลงแต่ละ record เป็น tuple

```python
def write_raw_records(filename: str, st: struct.Struct, records: list):
    with open(filename, "wb") as f:
        for r in records:
            f.write(st.pack(*r))
```

4.2.8 write_raw_records

- เขียน **record ทั้งหมด** ลงไฟล์ (overwrite)

เหตุผล: ฟังก์ชันเหล่านี้ทำให้ **CRUD ข้อมูลในไฟล์ไบนารี** ง่ายและเป็นระบบ

```python
# --------------- Conversion helpers ---------------
def decode_record(raw_tuple):
    return tuple(unpack_str(x) if isinstance(x, (bytes, bytearray)) else x for x in raw_tuple)
```

4.2.9 decode_record

- แปลง tuple ที่อ่านจากไฟล์

- bytes ⟶ string

- int/float ⟶ คงเดิม

- **เหตุผล:** เพื่อให้ง่ายต่อการแสดงข้อมูลและใช้งาน

```python
def add_book():
    print("\n== Add Book ==")
    book_id = get_int("Book ID (ตัวเลข): ")
    if any(b[0] == book_id for b in read_raw_records(BOOK_FILE, BOOK_STRUCT)):
        print(" Book ID นี้มีอยู่แล้ว")
        return
    title = get_str("Title: ", 100)
    author = get_str("Author: ", 100, allow_empty=True)
    publisher = get_str("Publisher: ", 100, allow_empty=True)
    year_pub = get_int("Year (เช่น 2023): ", 0, 9999)
    category = get_str("Category: ", 50, allow_empty=True)
    language = get_str("Language: ", 50, allow_empty=True)
    shelf_no = get_str("Shelf No.: ", 20, allow_empty=True)
    total_copies = get_int("Total copies: ", 0)
    packed = (
        book_id, pack_str(title, 100), pack_str(author, 100), pack_str(publisher, 100),
        year_pub, pack_str(category, 50), pack_str(language, 50),
        pack_str(shelf_no, 20), total_copies
    )
    add_record(BOOK_FILE, BOOK_STRUCT, packed)
    print(" เพิ่มหนังสือสำเร็จ")
```

4.2.10 add_book

ฟังก์ชัน add_book มีหน้าที่สร้างข้อมูลหนังสือ 1 เล่มแล้วบันทึกลงไฟล์books.dat ใน

รูปแบบไบนารี โดยเริ่มจากการเก็บข้อมูล ได้แก่ รหัสหนังสือ (book_id), ชื่อ หนังสือ (title), สถานะ (status), ผู้แต่ง (author), ปีพิมพ์ (year), จำนวนเล่ม (copies), และเวลา (created_at, updated_at) มาจัดรูปแบบตามโครงสร้างที่กำหนดใน books_struck.pack โดยจะมี การแปลงข้อความเป็น bytes และปรับความยาวให้คงที่ด้วยการเติมค่า \x00 เพื่อให้แต่ละเรคอร์ดมี ขนาดเท่ากันเสมอ สุดท้ายเปิดไฟล์books.dat ในโหมด append binary ("ab") แล้วเขียนข้อมูลเรคอร์ดที่สร้างขึ้นต่อท้ายไฟล์ ทำให้สามารถเก็บข้อมูลหนังสือเพิ่มได้ทีละเล่มโดยไม่ไปทับข้อมูลเดิมที่มี

อยู่ก่อนหน้า

```python
def view_books():
    print("\n== View Books ==")
    raws = read_raw_records(BOOK_FILE, BOOK_STRUCT)
    if not raws:
        print("ไม่มีข้อมูลหนังสือ")
        return
    print(f"{'ID':<6} {'Title':<30} {'Author':<20} {'Year':<6} {'Copies':<6}")
    print("-" * 80)
    for r in raws:
        rr = decode_record(r)
        print(f"{rr[0]:<6} {rr[1][:30]:<30} {rr[2][:20]:<20} {rr[4]:<6} {rr[8]:<6}")
```

4.2.11 view_books

ฟังก์ชัน view_books ใช้สำหรับแสดงรายการหนังสือทั้งหมดจากไฟล์ books.dat โดยเริ่มจากเรียกฟังก์ชัน read_all_books เพื่อดึงข้อมูลหนังสือทั้งหมดเข้ามาเป็นลิสต์หากไม่พบหนังสือจะแจ้งข้อความ "No books found." แล้วออกจากฟังก์ชัน จากนั้นพิมพ์หัวตารางและเส้นคั่นเพื่อจัดรูปแบบให้สวยงาม สำหรับแต่ละหนังสือในลิสต์จะแปลงสถานะเป็นข้อความ"Active" หากสถานะเป็น 1 หรือ"Deleted" หากสถานะเป็น 0 แล้วพิมพ์ข้อมูลหนังสือแต่ละรายการ
เช่น รหัสหนังสือ, ชื่อ, ผู้แต่ง, ปีพิมพ์, จำนวนเล่ม และสถานะ โดยจัดเรียงให้อ่านง่ายด้วยการจัดความกว้างคอลัมน์ สุดท้ายพิมพ์เส้นคั่นเพื่อปิดท้ายตาราง ทำให้ผู้ใช้สามารถมองเห็นรายการหนังสือทั้งหมดพร้อมสถานะได้อย่างชัดเจน

```
def update_book():
    print("\n== Update Book ==")
    book_id = get_int("Book ID ที่ต้องการแก้ไข: ")
    raws = read_raw_records(BOOK_FILE, BOOK_STRUCT)
    found = False
    for idx, r in enumerate(raws):
        if r[0] == book_id:
            found = True
            rr = decode_record(r)
            print("ข้อมูลเดิม:", rr)
            # ... (Logic to get new values)
            new_title = get_str("New Title (Enter=ไม่เปลี่ยน): ", 100, allow_empty=True) or rr[1]
            new_author = get_str("New Author (Enter=ไม่เปลี่ยน): ", 100, allow_empty=True) or rr[2]
            # ... (Rest of the fields)
            raws[idx] = (
                book_id, pack_str(new_title, 100), pack_str(new_author, 100), r[3], r[4], r[5], r[6], r[7], r[8]
            ) # Simplified for brevity
            write_raw_records(BOOK_FILE, BOOK_STRUCT, raws)
            print(" แก้ไขเรียบร้อย")
            break
    if not found:
        print(" ไม่พบ Book ID")
```

4.2.12 update_book

ฟังก์ชัน update_book ใช้สำหรับแก้ไขข้อมูลหนังสือในไฟล์books.dat โดยเริ่มจากเรียกฟังก์ชัน update_bookเพื่อแสดงรายการหนังสือทั้งหมดให้ผู้ใช้เลือก จากนั้นรับรหัสหนังสือ (book_id) ที่ต้องการแก้ไข หากผู้ใช้ป้อนค่าไม่ถูกต้องจะหยุดฟังก์ชันและแจ้งเตือน หลังจากนั้นอ่านข้อมูลหนังสือทั้งหมดเข้าลิสต์books ในรูปแบบไบนารีและแปลงเป็น list ของแต่ละเรคอร์ด ถัดไปจะค้นหาหนังสือที่ตรงกับรหัสและมีสถานะ Active (1) หากพบ จะให้ผู้ใช้ป้อนข้อมูลใหม่ เช่น ชื่อหนังสือ (title), ผู้แต่ง (author), ปีพิมพ์ (year) และจำนวนเล่ม (copies) โดยสามารถเว้นว่างเพื่อคงค่าของเดิมได้ หลังจากรับข้อมูลแล้ว จะอัปเดตค่าต่าง ๆ ในเรคอร์ดรวมถึงปรับ updated_at เป็นเวลาปัจจุบัน เมื่อแก้ไขเสร็จแล้ว เขียนข้อมูลทั้งหมดกลับลงไฟล์ในโหมดเขียนทับ ("wb") และแจ้งผู้ใช้ว่าการแก้ไขสำเร็จ หากไม่พบหนังสือหรือไม่ใช่ Active จะแจ้งข้อความว่าไม่พบหรือไม่สามารถแก้ไขได้ ฟังก์ชันนี้ช่วยให้ผู้ใช้สามารถปรับปรุงข้อมูลหนังสือได้อย่างสะดวกและปลอดภัย

```python
def delete_book():
    print("\n== Delete Book ==")
    book_id = get_int("Book ID ที่ต้องการลบ: ")
    raws = read_raw_records(BOOK_FILE, BOOK_STRUCT)
    new = [r for r in raws if r[0] != book_id]
    if len(new) == len(raws):
        print(" ไม่พบ Book ID")
        return
    write_raw_records(BOOK_FILE, BOOK_STRUCT, new)
    print(" ลบสำเร็จ")
```

4.2.13 delete_book

ฟังก์ชัน delete_book ใช้สำหรับลบหนังสือจากไฟล์ books.dat โดยจะอ่านข้อมูลหนังสือทั้งหมดเข้ามาในลิสต์ก่อน จากนั้นค้นหาหนังสือที่ตรงกับรหัส book_id และมีสถานะยังใช้งานอยู่ ถ้าพบจะเปลี่ยนสถานะเป็น 0 (ปิดการใช้งาน) และปรับค่าupdated_at เป็นเวลาปัจจุบัน จากนั้นเขียนข้อมูลทั้งหมดกลับลงไฟล์แทนของเดิม หากไม่พบหนังสือหรือถูกลบไปแล้วจะแจ้งผู้ใช้ ฟังก์ชันนี้เป็นการลบแบบ soft delete ทำให้ข้อมูลยังอยู่ในไฟล์แต่ระบบจะถือว่าหนังสือเล่มนั้นถูกลบเรียบร้อยแล้ว

```python
def add_member():
    print("\n== Add Member ==")
    member_id = get_int("Member ID (ตัวเลข): ")
    if any(m[0] == member_id for m in read_raw_records(MEMBER_FILE, MEMBER_STRUCT)):
        print(" Member ID นี้มีอยู่แล้ว")
        return
    name = get_str("Name Surname: ", 100)
    birth_date = get_date("Birth date (YYYY-MM-DD): ")
    gender = get_str("Gender (M/F/O): ", 1)
    address = get_str("Address: ", 200, allow_empty=True)
    mobile = get_str("Mobile: ", 15, allow_empty=True)
    email = get_str("Email: ", 100, allow_empty=True)
    reg_date = get_date("Reg date (YYYY-MM-DD): ")
    packed = (
        member_id, pack_str(name, 100), pack_str(birth_date, 10), pack_str(gender, 1),
        pack_str(address, 200), pack_str(mobile, 15), pack_str(email, 100),
        pack_str(reg_date, 10)
    )
    add_record(MEMBER_FILE, MEMBER_STRUCT, packed)
    print(" เพิ่มสมาชิกสำเร็จ")
```

4.2.13 add_member

ฟังก์ชัน add_members มีหน้าที่บันทึกข้อมูลสมาชิกใหม่ลงไฟล์members.dat โดยใช้โครงสร้างที่กำหนดใน members_struck เพื่อให้ข้อมูลแต่ละเรคอร์ดมีขนาดคงที่ เริ่มต้นด้วยการเก็บเวลาปัจจุบัน (now) ในรูปแบบ Unix timestamp สำหรับใช้เป็นค่า created_at และ updated_at จากนั้นนำข้อมูลที่รับเข้ามา ได้แก่ รหัสสมาชิก (member_id), สถานะ (status), ชื่อ (name), ปีเกิด (birth_year) และจำนวนยืมสูงสุด (max_loan) มาจัดรูปแบบให้ตรงกับโครงสร้าง โดยชื่อจะถูกแปลงเป็นข้อมูลแบบ bytes และบังคับความยาว 50 ไบต์ด้วยการเติม \x00 จากนั้นข้อมูลทั้งหมดถูก pack เป็นเรคอร์ดไบนารี ก่อนจะเปิดไฟล์ members.dat ในโหมด append binary ("ab") และเขียนข้อมูลเรคอร์ดลงไปท้ายไฟล์ ทำให้สามารถเพิ่มสมาชิกใหม่ได้ต่อเนื่องโดยไม่กระทบกับข้อมูลเดิมที่มีอยู่แล้ว.

```python
def view_members():
    print("\n== View Members ==")
    raws = read_raw_records(MEMBER_FILE, MEMBER_STRUCT)
    if not raws:
        print("ไม่มีข้อมูลสมาชิก")
        return
    print(f"{'ID':<6} {'Name':<25} {'Birth Date':<12} {'Mobile':<15} {'Email':<25}")
    print("-" * 90)
    for r in raws:
        rr = decode_record(r)
        print(f"{rr[0]:<6} {rr[1][:25]:<25} {rr[2]:<12} {rr[5]:<15} {rr[6][:25]:<25}")
```

4.2.14 view_members

ฟังก์ชัน view_members ใช้สำหรับแสดงรายการสมาชิกทั้งหมดจากไฟล์ members.dat โดยเริ่มจากเรียก read_all_members เพื่อดึงข้อมูลสมาชิกเข้ามาเป็นลิสต์ หากไม่พบสมาชิกจะแจ้งข้อความ "No members found." แล้วออกจากฟังก์ชัน จากนั้นพิมพ์หัวตารางและเส้นคั่นเพื่อจัดรูปแบบให้สวยงาม สำหรับแต่ละสมาชิกในลิสต์ จะตรวจสอบสถานะ หาก status เป็น 1 จะแสดงเป็น "Active" หากเป็น 0 จะแสดงเป็น "Deleted" แล้วพิมพ์ข้อมูลสมาชิก เช่น รหัสสมาชิก, ชื่อ, ปีเกิด, จำนวนครั้งที่สามารถยืมสูงสุด และสถานะ โดยจัดความกว้างคอลัมน์ให้อ่านง่าย สุดท้ายพิมพ์เส้นคั่นเพื่อปิดท้ายตาราง ทำให้ผู้ใช้สามารถเห็นข้อมูลสมาชิกทั้งหมดพร้อมสถานะได้อย่างชัดเจน

```python
def update_member():
    print("\n== Update Member ==")
    member_id = get_int("Member ID ที่ต้องการแก้ไข: ")
    raws = read_raw_records(MEMBER_FILE, MEMBER_STRUCT)
    found = False
    for idx, r in enumerate(raws):
        if r[0] == member_id:
            found = True
            rr = decode_record(r)
            print("ข้อมูลเดิม:", rr)
            # ... (Logic to get new values)
            new_name = get_str("New Name (Enter=ไม่เปลี่ยน): ", 100, allow_empty=True) or rr[1]
            # ... (Rest of the fields)
            raws[idx] = (
                member_id, pack_str(new_name, 100), r[2], r[3], r[4], r[5], r[6], r[7]
            ) # Simplified for brevity
            write_raw_records(MEMBER_FILE, MEMBER_STRUCT, raws)
            print(" แก้ไขข้อมูลสมาชิกเรียบร้อย")
            break
    if not found:
        print(" ไม่พบ Member ID")
```

4.2.15 update_member

ทำหน้าที่ **แก้ไขข้อมูลของสมาชิก** ที่มีอยู่แล้วในระบบ

เมื่อฟังก์ชันนี้ถูกเรียกใช้งาน จะมีขั้นตอนการทำงานดังนี้:

1. **แสดงส่วนหัว**: เริ่มต้นด้วยการแสดงข้อความ "== Update Member ==" เพื่อให้ผู้ใช้รู้ว่ากำลังจะทำการแก้ไขข้อมูลสมาชิก
2. **รับ Member ID**: โปรแกรมจะขอให้ผู้ใช้ป้อน "Member ID" ของสมาชิกที่ต้องการแก้ไขข้อมูล
3. **ค้นหาข้อมูล**: จากนั้น โปรแกรมจะเข้าไปอ่านข้อมูลสมาชิกทั้งหมดจากไฟล์ MEMBER_FILE เพื่อค้นหาว่ามี Member ID ตรงกับที่ผู้ใช้ป้อนมาหรือไม่
4. **ตรวจสอบผลการค้นหา**:
    - **ถ้าเจอ**: โปรแกรมจะแสดงข้อมูลเดิมของสมาชิกคนนั้นให้ผู้ใช้เห็นก่อน จากนั้นจะเปิดโอกาสให้ผู้ใช้ป้อนข้อมูลใหม่ในแต่ละช่อง (เช่น ชื่อใหม่) หากผู้ใช้ไม่ต้องการแก้ไขช่องไหน ก็สามารถกด Enter เพื่อข้ามไปได้ (ซึ่งจะยังคงใช้ข้อมูลเดิม) เมื่อผู้ใช้แก้ไขข้อมูลเสร็จสิ้น โปรแกรมจะนำข้อมูลใหม่นี้ไปเขียนทับข้อมูลเก่าในไฟล์ แล้วแสดงข้อความว่า "แก้ไขข้อมูลสมาชิกเรียบร้อย"
    - **ถ้าไม่เจอ**: หากค้นหาแล้วไม่พบ Member ID ที่ระบุ โปรแกรมจะแสดงข้อความว่า "ไม่พบ Member ID" แล้วจบการทำงานในส่วนนี้

```python
def delete_member():
    print("\n== Delete Member ==")
    member_id = get_int("Member ID ที่ต้องการลบ: ")
    raws = read_raw_records(MEMBER_FILE, MEMBER_STRUCT)
    new_raws = [r for r in raws if r[0] != member_id]
    if len(raws) == len(new_raws):
        print(" ไม่พบ Member ID")
    else:
        write_raw_records(MEMBER_FILE, MEMBER_STRUCT, new_raws)
        print(" ลบข้อมูลสมาชิกสำเร็จ")
```

4.2.16 delete_member

มีหน้าที่**ลบข้อมูลของสมาชิก**ออกจากระบบ

อธิบายการทำงานเป็นขั้นตอนได้ดังนี้:

1. **แสดงส่วนหัว**: โปรแกรมจะแสดงข้อความ == Delete Member == เพื่อบอกผู้ใช้ว่ากำลังจะเข้าสู่กระบวนการลบข้อมูล
2. **รับ Member ID**: โปรแกรมจะขอให้ผู้ใช้ป้อน "Member ID" ของสมาชิกที่ต้องการลบ
3. **อ่านข้อมูลทั้งหมด**: โค้ดจะอ่านข้อมูลสมาชิกทั้งหมดจากไฟล์ MEMBER_FILE มาเก็บไว้ในลิสต์ที่ชื่อ raws
4. **กรองข้อมูล**: โปรแกรมจะสร้างลิสต์ใหม่ชื่อ new_raws โดยคัดลอกข้อมูลจาก raws มาทั้งหมด **ยกเว้น** ข้อมูลของสมาชิกที่มี Member ID ตรงกับที่ผู้ใช้ป้อนเข้ามา
5. **ตรวจสอบและดำเนินการ**:
    - **ถ้าไม่พบ ID**: โปรแกรมจะเปรียบเทียบจำนวนข้อมูลในลิสต์เก่า (raws) กับลิสต์ใหม่ (new_raws) ถ้าจำนวนเท่ากัน แสดงว่าไม่มีข้อมูลใดถูกลบออก (เพราะหา ID ที่ระบุไม่เจอ) ก็จะแสดงข้อความว่า ไม่พบ Member ID
    - **ถ้าพบ ID**: ถ้าจำนวนข้อมูลในลิสต์ใหม่น้อยกว่าลิสต์เก่า แสดงว่ามีข้อมูลถูกลบออกไป โปรแกรมจะเขียนข้อมูลใน new_raws (ซึ่งไม่มีข้อมูลของสมาชิกที่ต้องการลบแล้ว) กลับลงไปที่ไฟล์ MEMBER_FILE เป็นการบันทึกทับของเดิม จากนั้นจึงแสดงข้อความ ลบข้อมูลสมาชิกสำเร็จ

```python
def add_borrow():
    print("\n== Add Borrow (Multiple books) ==")
    member_id = get_int("Member ID: ")
    if not any(r[0] == member_id for r in read_raw_records(MEMBER_FILE, MEMBER_STRUCT)):
        print(" ไม่พบ Member ID")
        return

    books_to_borrow = []
    while True:
        book_id_str = input(f"Book ID ที่จะยืม (สำหรับ Member {member_id}) (พิมพ์ 'done' เพื่อสิ้นสุด): ").strip().lower()
        if book_id_str == 'done':
            if not books_to_borrow:
                print("ยังไม่ได้เพิ่มหนังสือเลย ยกเลิกการยืม")
            break

        try:
            book_id = int(book_id_str)
            if not any(b[0] == book_id for b in read_raw_records(BOOK_FILE, BOOK_STRUCT)):
                print(f" ไม่พบ Book ID {book_id}")
                continue
            books_to_borrow.append(book_id)
            print(f"  เพิ่ม Book ID {book_id} ในรายการ")
        except ValueError:
            print(" ต้องกรอก Book ID เป็นตัวเลข")

    if books_to_borrow:
        date_out = get_date("Date out (YYYY-MM-DD) สำหรับหนังสือทั้งหมด: ")
        date_due = get_date("Date due (YYYY-MM-DD) สำหรับหนังสือทั้งหมด: ")

        for book_id in books_to_borrow:
            packed = (
                member_id, book_id, pack_str(date_out, 10), pack_str(date_due, 10),
                pack_str("", 10), pack_str("Borrow", 20), 0.0, pack_str("", 200)
            )
            add_record(BORROW_FILE, BORROW_STRUCT, packed)
        print(f"\nเพิ่มการยืมหนังสือ {len(books_to_borrow)} เล่มสำหรับ Member ID {member_id} สำเร็จ")
```

4.2.17 add_borrow

ฟังก์ชัน add_borrow ใช้สำหรับบันทึกการยืมหนังสือใหม่ลงในไฟล์ borrows.dat โดยเริ่มจากพิมพ์หัวข้อเพื่อบอกผู้ใช้ว่ากำลังอยู่ในเมนูเพิ่มการยืม จากนั้นจะให้ผู้ใช้กรอกรหัสสมาชิก (Member ID) และตรวจสอบว่ามีอยู่จริงในไฟล์ members.dat หรือไม่ โดยใช้ read_raw_records อ่านข้อมูลสมาชิกทั้งหมดแล้วตรวจสอบด้วยเงื่อนไข any(...) หากไม่พบรหัสที่ตรงกัน โปรแกรมจะแสดงข้อความ “ไม่พบ Member ID” และหยุดทำงานทันที หลังจากนั้นจะให้กรอกรหัสหนังสือ (Book ID) แล้วตรวจสอบด้วยวิธีเดียวกันกับไฟล์ books.dat เพื่อยืนยันว่าหนังสือมีอยู่จริง หากไม่พบก็จะแจ้งว่า “ไม่พบ Book ID” และออกจากฟังก์ชัน

ถ้าทั้งรหัสสมาชิกและรหัสหนังสือถูกต้อง ฟังก์ชันจะให้ผู้ใช้กรอกวันที่ยืม (date_out) และวันที่กำหนดคืน (date_due) โดยบังคับรูปแบบเป็น YYYY-MM-DD ส่วนวันที่คืน (date_return) จะตั้งค่าเป็นค่าว่างไว้ก่อนเพราะยังไม่ได้คืนจริง จากนั้นรับค่าระบุสถานะ (status) เช่น Borrow หรือ Return พร้อมทั้งค่าปรับ (fine_amount) ที่ต้องเป็นตัวเลขไม่ติดลบ และบันทึกหมายเหตุ (notes) เพิ่มเติมได้ตามต้องการ

เมื่อรวบรวมข้อมูลครบแล้วจะนำค่าทั้งหมดมาทำการ pack ให้เป็นรูปแบบ bytes ตามโครงสร้าง BORROW_STRUCT ได้แก่ member_id, book_id, วันที่ยืม, วันที่กำหนดคืน, วันที่คืน, สถานะ, ค่าปรับ และหมายเหตุ แล้วเรียกใช้ฟังก์ชัน add_record เพื่อเขียนข้อมูลการยืมนี้ลงไฟล์ borrows.dat แบบต่อท้าย สุดท้ายพิมพ์ข้อความ “เพิ่มการยืมสำเร็จ” เพื่อยืนยันกับผู้ใช้ว่าข้อมูลถูกบันทึกเรียบร้อย

```
def view_borrows():
    print("\n== View Borrows (Grouped) ==")
    borrows_raw = read_raw_records(BORROW_FILE, BORROW_STRUCT)
    if not borrows_raw:
        print("ไม่มีข้อมูลการยืม")
        return

    grouped_borrows = {}
    for r in borrows_raw:
        rr = decode_record(r)
        member_id = rr[0]
        if member_id not in grouped_borrows:
            grouped_borrows[member_id] = []
        grouped_borrows[member_id].append(rr)

    members_map = {m[0]: decode_record(m) for m in read_raw_records(MEMBER_FILE, MEMBER_STRUCT)}
    books_map = {b[0]: decode_record(b) for b in read_raw_records(BOOK_FILE, BOOK_STRUCT)}

    for member_id, borrows in grouped_borrows.items():
        member_info = members_map.get(member_id, (None, "Unknown Member"))
        print("-" * 80)
        print(f"Member ID: {member_id} | Name: {member_info[1]}")
        print(f"{'':<4}{'BookID':<7} | {'Title':<40} | {'Status'}")

        for borrow_item in borrows:
            book_id = borrow_item[1]
            book_info = books_map.get(book_id, (None, "Unknown Book"))
            title = book_info[1]
            status = borrow_item[5]
            print(f"{'':<4}{book_id:<7} | {title[:40]:<40} | {status}")
    print("-" * 80)
```

4.2.18 view_borrows

ฟังก์ชัน view_borrows ใช้สำหรับแสดงรายการข้อมูลการยืมหนังสือทั้งหมดที่ถูกบันทึกไว้ในไฟล์ borrows.dat โดยเริ่มจากการพิมพ์หัวข้อบอกผู้ใช้ว่าอยู่ในเมนูแสดงการยืม หลังจากนั้นฟังก์ชันจะเรียก read_raw_records พร้อมกำหนดชื่อไฟล์ BORROW_FILE และโครงสร้างข้อมูล BORROW_STRUCT เพื่อนำข้อมูลดิบทั้งหมดที่เก็บอยู่ในไฟล์มาเก็บไว้ในตัวแปร raws ถ้าไม่พบข้อมูลเลย หรือไฟล์ยังว่างเปล่า ฟังก์ชันจะแสดงข้อความ “ไม่มีข้อมูลยืม” และหยุดทำงานทันที หากมีข้อมูลยืมอยู่ ฟังก์ชันจะจัดรูปแบบตารางเพื่อแสดงผล โดยพิมพ์หัวตารางที่ประกอบไปด้วยคอลัมน์ Member ID, Book ID, Date Out, Date Due, Date Return, Status และ Fine จากนั้นพิมพ์เส้นคั่นด้วย “–” ความยาว 80 ตัวอักษรเพื่อให้การแสดงผลเป็นระเบียบและอ่านง่าย

ขั้นตอนต่อไปจะวนลูปใน raws เพื่อนำข้อมูลแต่ละ record ออกมาแสดง โดยแต่ละ record จะถูกแปลงจาก bytes ให้เป็น string หรือค่าตัวเลขที่อ่านได้ง่ายด้วยฟังก์ชัน decode_record แล้วนำมาจัดรูปแบบตามคอลัมน์ที่กำหนด เช่น รหัสสมาชิก (rr[0]), รหัสหนังสือ (rr[1]), วันที่ยืม (rr[2]), วันที่กำหนดคืน (rr[3]), วันที่คืน (rr[4]), สถานะ (rr[5]) และค่าปรับ (r[6]) ซึ่งค่าปรับจะถูกแสดงเป็นทศนิยมสองตำแหน่งโดยใช้รูปแบบ {r[6]:<6.2f} เพื่อให้ผลลัพธ์ออกมาเป็นตัวเลขชัดเจนและสวยงาม สุดท้ายเมื่อวนลูปแสดงข้อมูลทั้งหมดเสร็จ ผู้ใช้ก็จะเห็นตารางการยืมหนังสืออย่างครบถ้วนทั้งรหัสสมาชิก รหัสหนังสือ วันเวลาต่าง ๆ สถานะการยืม และจำนวนค่าปรับที่เกี่ยวข้อง

```python
def update_borrow():
    print("\n== Update Borrow Record ==")
    view_borrows() # แสดงข้อมูลทั้งหมดก่อน
    borrows_raw = read_raw_records(BORROW_FILE, BORROW_STRUCT)
    if not borrows_raw:
        return

    member_id_to_edit = get_int("ใส่ Member ID ที่ต้องการแก้ไข: ")

    # กรองรายการยืมเฉพาะของสมาชิกคนนี้
    member_borrows_raw = [r for r in borrows_raw if r[0] == member_id_to_edit]
    if not member_borrows_raw:
        print(f"ไม่พบรายการยืมสำหรับ Member ID {member_id_to_edit}")
        return

    books_map = {b[0]: decode_record(b) for b in read_raw_records(BOOK_FILE, BOOK_STRUCT)}
    print(f"\nรายการหนังสือสำหรับ Member ID {member_id_to_edit}:")
    for i, r in enumerate(member_borrows_raw):
        rr = decode_record(r)
        book_info = books_map.get(rr[1], (None, "Unknown Book"))
        print(f"  {i+1}: Book ID {rr[1]} ({book_info[1][:30]}) - Status: {rr[5]}")

    rec_num = get_int("เลือกลำดับหนังสือที่ต้องการแก้ไข: ", minv=1, maxv=len(member_borrows_raw))
    record_to_update_raw = member_borrows_raw[rec_num - 1]

    # หา index ของ record นี้ใน list ดั้งเดิมทั้งหมด
    original_index = -1
    for i, r in enumerate(borrows_raw):
        if r == record_to_update_raw:
            original_index = i
            break

    if original_index == -1:
        print("เกิดข้อผิดพลาด: ไม่พบข้อมูลที่ต้องการแก้ไขในไฟล์หลัก")
        return

    rr = decode_record(record_to_update_raw)
    print("\nข้อมูลเดิม:", rr)

    new_date_out = get_date("New Date Out (Enter=ไม่เปลี่ยน): ", allow_empty=True) or rr[2]
    new_date_due = get_date("New Date Due (Enter=ไม่เปลี่ยน): ", allow_empty=True) or rr[3]
    new_date_return = get_date("New Date Return (Enter=ไม่เปลี่ยน): ", allow_empty=True) or rr[4]
    new_status = get_str("New Status (Enter=ไม่เปลี่ยน): ", 20, allow_empty=True) or rr[5]

    new_fine_str = input(f"New Fine (Enter=ไม่เปลี่ยน, Current={rr[6]}): ").strip()
    fine_amount = float(new_fine_str) if new_fine_str else rr[6]

    new_notes = get_str("New Notes (Enter=ไม่เปลี่ยน): ", 200, allow_empty=True) or rr[7]

    new_packed = (
        rr[0], rr[1],
        pack_str(new_date_out, 10), pack_str(new_date_due, 10), pack_str(new_date_return, 10),
        pack_str(new_status, 20), fine_amount, pack_str(new_notes, 200)
    )

    borrows_raw[original_index] = new_packed
    write_raw_records(BORROW_FILE, BORROW_STRUCT, borrows_raw)
    print(" แก้ไขข้อมูลการยืมเรียบร้อย")
```

4.2.19 update_borrow

มีหน้าที่**แก้ไขรายละเอียดของการยืมหนังสือแต่ละรายการ**

เมื่อฟังก์ชันนี้ทำงาน จะมีขั้นตอนที่ค่อนข้างละเอียดดังนี้ครับ:

1. **แสดงข้อมูลทั้งหมดก่อน**: เริ่มแรก โปรแกรมจะแสดงข้อมูลการยืม-คืนทั้งหมดที่มีอยู่ในระบบ เพื่อให้ผู้ใช้เห็นภาพรวม
2. **รับ Member ID**: โปรแกรมจะขอให้ผู้ใช้ป้อน "Member ID" ของสมาชิกที่ต้องการแก้ไขรายการยืม
3. **ค้นหารายการยืมของสมาชิก**: โค้ดจะกรอง (filter) ข้อมูลการยืมทั้งหมด ให้เหลือเฉพาะรายการที่เป็นของ Member ID ที่ระบุ
    - ถ้าไม่พบรายการยืมของสมาชิกคนนั้นเลย โปรแกรมจะแจ้งเตือนและจบการทำงาน
4. **แสดงรายการหนังสือที่ยืม**: โปรแกรมจะแสดงรายการหนังสือทั้งหมดที่สมาชิกคนนั้นยืมอยู่ โดยมีลำดับหมายเลขกำกับ (เช่น 1, 2, 3) พร้อมแสดงชื่อหนังสือและสถานะการยืม เพื่อให้ผู้ใช้เลือกได้ง่ายขึ้น
5. **ให้ผู้ใช้เลือกรายการที่ต้องการแก้ไข**: ผู้ใช้จะต้องป้อน "ลำดับ" ของหนังสือที่ต้องการจะแก้ไขข้อมูล
6. **แสดงข้อมูลเดิมและรับข้อมูลใหม่**:
    - โปรแกรมจะแสดงข้อมูลเดิมของรายการยืมที่ผู้ใช้เลือก (เช่น วันที่ยืม, วันกำหนดส่ง, สถานะ, ค่าปรับ)
    - จากนั้น จะไล่ถามข้อมูลใหม่ในแต่ละช่อง ผู้ใช้สามารถป้อนข้อมูลใหม่ที่ต้องการ หรือกด Enter เพื่อข้ามและใช้ข้อมูลเดิมได้
7. **อัปเดตข้อมูล**: หลังจากผู้ใช้ป้อนข้อมูลใหม่ครบแล้ว โปรแกรมจะนำข้อมูลใหม่นี้ไปแทนที่ข้อมูลเก่าในตำแหน่งที่ถูกต้อง
8. **บันทึกข้อมูลลงไฟล์**: สุดท้าย โปรแกรมจะเขียนข้อมูลการยืมทั้งหมด (ทั้งที่แก้ไขและไม่ได้แก้ไข) กลับลงไปในไฟล์ BORROW_FILE เพื่อบันทึกการเปลี่ยนแปลง
9. **แจ้งผล**: เมื่อบันทึกเรียบร้อย จะแสดงข้อความว่า "แก้ไขข้อมูลการยืมเรียบร้อย"

```python
def delete_borrow():
    print("\n== Delete Borrow Record ==")
    view_borrows() # แสดงข้อมูลทั้งหมดก่อน
    borrows_raw = read_raw_records(BORROW_FILE, BORROW_STRUCT)
    if not borrows_raw:
        return

    member_id_to_delete = get_int("ใส่ Member ID ที่ต้องการลบรายการ: ")

    member_borrows_raw = [r for r in borrows_raw if r[0] == member_id_to_delete]
    if not member_borrows_raw:
        print(f"ไม่พบรายการยืมสำหรับ Member ID {member_id_to_delete}")
        return

    books_map = {b[0]: decode_record(b) for b in read_raw_records(BOOK_FILE, BOOK_STRUCT)}
    print(f"\nรายการหนังสือสำหรับ Member ID {member_id_to_delete}:")
    for i, r in enumerate(member_borrows_raw):
        rr = decode_record(r)
        book_info = books_map.get(rr[1], (None, "Unknown Book"))
        print(f"  {i+1}: Book ID {rr[1]} ({book_info[1][:30]}) - Status: {rr[5]}")

    rec_num = get_int("เลือกลำดับหนังสือที่ต้องการลบ: ", minv=1, maxv=len(member_borrows_raw))
    record_to_delete_raw = member_borrows_raw[rec_num - 1]

    confirm = input(f"ต้องการลบรายการยืมนี้ใช่หรือไม่? (y/n): ").strip().lower()
    if confirm == 'y':
        borrows_raw.remove(record_to_delete_raw)
        write_raw_records(BORROW_FILE, BORROW_STRUCT, borrows_raw)
        print(" ลบข้อมูลการยืมสำเร็จ")
    else:
        print("ยกเลิกการลบ")
```

4.2.20 delete_borrow

นี้มีหน้าที่**ลบรายการยืมหนังสือทีละรายการ**ออกจากระบบ

อธิบายขั้นตอนการทำงานได้ดังนี้:

1. **แสดงข้อมูลภาพรวม**: โปรแกรมจะเริ่มด้วยการแสดงข้อมูลการยืม-คืนทั้งหมดที่มีอยู่ เพื่อให้ผู้ใช้เห็นภาพรวมก่อน
2. **รับ Member ID**: จากนั้นจะขอให้ผู้ใช้ป้อน "Member ID" ของสมาชิกที่ต้องการจะลบรายการยืม
3. **ค้นหาและแสดงรายการของสมาชิก**:
    - โปรแกรมจะค้นหารายการยืมทั้งหมดที่ผูกกับ Member ID นั้น
    - หากไม่พบ ระบบจะแจ้งเตือนและจบการทำงาน
    - หากพบ จะแสดงรายการหนังสือที่สมาชิกคนนั้นยืมอยู่ทั้งหมด โดยมีลำดับหมายเลขกำกับไว้ (เช่น 1, 2, 3) เพื่อให้ผู้ใช้เลือกได้สะดวก

4. **ให้ผู้ใช้เลือกรายการที่จะลบ**: โปรแกรมจะขอให้ผู้ใช้ป้อน "ลำดับ" ของหนังสือที่ต้องการลบข้อมูลการยืม
5. **ยืนยันการลบ**: เพื่อป้องกันความผิดพลาด โปรแกรมจะถามเพื่อยืนยันอีกครั้งว่า "ต้องการลบรายการยืมนี้ใช่หรือไม่? (y/n)"
6. **ดำเนินการลบหรือยกเลิก**:
    - **ถ้ายืนยัน (y)**: โปรแกรมจะลบรายการที่เลือกออกจากข้อมูลทั้งหมด แล้วเขียนข้อมูลที่เหลือกลับลงไปในไฟล์ BORROW_FILE จากนั้นแจ้งว่า "ลบข้อมูลการยืมสำเร็จ"
    - **ถ้าไม่ยืนยัน (ตัวอักษรอื่น ๆ)**: โปรแกรมจะยกเลิกการทำงานและแสดงข้อความว่า "ยกเลิกการลบ"

```python
def generate_report():
    print("\nGenerating reports...")
    books_raw = read_raw_records(BOOK_FILE, BOOK_STRUCT)
    books = [decode_record(r) for r in books_raw]
    books_map = {b[0]: b for b in books}

    members_map = {m[0]: decode_record(m) for m in read_raw_records(MEMBER_FILE, MEMBER_STRUCT)}
    borrows_raw = read_raw_records(BORROW_FILE, BORROW_STRUCT)
    borrows = [decode_record(r) for r in borrows_raw]

    now = datetime.datetime.now()

    with open("books_report.txt", "w", encoding="utf-8") as f:
        # ... (same as before)
        f.write("Library Borrow System - Book Summary Report\n")
        f.write(f"Generated At : {now.strftime('%Y-%m-%d %H:%M')} (+07:00)\n\n")
        f.write(f"{'BookID':<6} | {'Title':<30} | {'Author':<20} | {'Year':<5} | {'Copies':<7} | {'Borrowed':<9} | {'Status'}\n")
        f.write("-" * 95 + "\n")
        borrowed_counts = {b_id: 0 for b_id in books_map.keys()}
        active_borrows_list = [br for br in borrows if br[5].lower() == 'borrow']
        for br in active_borrows_list:
            book_id = br[1]
            if book_id in borrowed_counts:
                borrowed_counts[book_id] += 1
        total_copies_sum = sum(b[8] for b in books)
        for b in books:
            book_id, title, author, _, year, _, _, _, total_copies = b
            borrowed = borrowed_counts.get(book_id, 0)
            f.write(f"{book_id:<6} | {title[:30]:<30} | {author[:20]:<20} | {str(year):<5} | {total_copies:<7} | {borrowed:<9} | {'Active'}\n")
        f.write("\n\nSummary (Active Books Only)\n")
        f.write(f"- Total Book Titles : {len(books)}\n")
        f.write(f"- Total Copies      : {total_copies_sum}\n")
        f.write(f"- Borrowed Now      : {sum(borrowed_counts.values())}\n")
        f.write(f"- Available Now     : {total_copies_sum - sum(borrowed_counts.values())}\n")

    with open("borrows_report.txt", "w", encoding="utf-8") as f:
        # ... (same as before, grouped)
        f.write("Library Borrow System - Borrowed Report\n")
        f.write(f"Generated At : {now.strftime('%Y-%m-%d %H:%M')} (+07:00)\n\n")
        grouped_borrows = {}
        for br in active_borrows_list:
            member_id = br[0]
            if member_id not in grouped_borrows:
                grouped_borrows[member_id] = []
            grouped_borrows[member_id].append(br)
        for member_id, borrows_list in grouped_borrows.items():
            member_info = members_map.get(member_id)
            if not member_info: continue
            f.write("-" * 120 + "\n")
            f.write(f"MemberID: {member_id:<5} | Name: {member_info[1]:<30} | Email: {member_info[6]}\n")
            f.write(f"{'':<4}{'BookID':<7} | {'Title':<40} | {'Author':<20} | {'Date Out':<12} | {'Due Date':<12} | {'Fine'}\n")
            f.write(f"{'':<4}{'-'*110}\n")
            for item in borrows_list:
                book_id, book_info = item[1], books_map.get(item[1], (None, "N/A", "N/A"))
                title, author = book_info[1], book_info[2]
                date_out, date_due, fine = item[2], item[3], item[6]
                f.write(f"{'':<4}{book_id:<7} | {title[:40]:<40} | {author[:20]:<20} | {date_out:<12} | {date_due:<12} | {fine:.2f}\n")
            f.write("\n")
        f.write("-" * 120 + "\n")
        f.write("Summary (Borrowed Only)\n")
        f.write(f"- Total Borrowed Books : {len(active_borrows_list)}\n")
        f.write(f"- Members with Borrows : {len(grouped_borrows)}\n")

    print(" รายงานถูกสร้าง: books_report.txt, borrows_report.txt")
# ---------------- Report ----------------
def generate_report():
    books = [decode_record(r) for r in read_raw_records(BOOK_FILE, BOOK_STRUCT)]
    borrows = [decode_record(r) for r in read_raw_records(BORROW_FILE, BORROW_STRUCT)]
    now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

4.2.21 generate_report

ฟังก์ชันสำหรับสร้างรายงาน (generate report) โดยทำงานหลัก ๆ คือรวบรวมข้อมูลจากไฟล์ที่เก็บบันทึกหนังสือและไฟล์ที่เก็บข้อมูลการยืมออกมา เพื่อเอามาประมวลผลแล้วจัดทำเป็นรายงาน โดยเริ่มแรกฟังก์ชันจะอ่านข้อมูลดิบที่บันทึกในไฟล์ BOOK_FILE ซึ่งถูกกำหนดโครงสร้างไว้ตาม BOOK_STRUCT แล้วถอดรหัส (decode) ข้อมูลที่อ่านออกมาให้อยู่ในรูปแบบที่ใช้งานง่าย เช่น

รายชื่อหนังสือ ผู้แต่ง สำนักพิมพ์ เป็นต้น หลังจากนั้นก็ทำในลักษณะเดียวกันกับข้อมูลการยืม โดยอ่านจาก BORROW_FILE ตามโครงสร้าง BORROW_STRUCT แล้วถอดรหัสออกมาเพื่อให้ได้ข้อมูลว่าใครยืมหนังสืออะไร วันที่ยืมคือเมื่อไหร่ กำหนดส่งเมื่อไหร่ คืนแล้วหรือยัง และค่าปรับเท่าไหร่
เมื่อได้ข้อมูลทั้งสองส่วนมาแล้ว ฟังก์ชันจะสร้างตัวแปร now เพื่อเก็บวันที่และเวลาปัจจุบันโดยใช้ datetime.datetime.now() แล้วแปลงให้อยู่ในรูปแบบตัวอักษรที่อ่านง่าย เช่น 2025-10-02 14:30:00 เพื่อเอาไปแสดงในหัวรายงานว่าเป็นรายงานที่สร้างขึ้นเมื่อเวลาใด ซึ่งช่วยให้รายงานนั้นมีความสมบูรณ์และอ้างอิงได้ว่าเป็นข้อมูล ณ เวลาปัจจุบันจริง ๆ
สรุปแล้วโครงสร้างฟังก์ชันนี้มีวัตถุประสงค์เพื่อดึงข้อมูลที่จัดเก็บไว้ในไฟล์ของระบบออกมาใช้ในการสร้างรายงานแบบรวม ทั้งในส่วนหนังสือและการยืม แล้วกำกับไว้ด้วยวันและเวลาปัจจุบันเพื่อบอกช่วงเวลาที่รายงานถูกสร้าง ซึ่งในขั้นตอนต่อไปอาจมีการนำข้อมูลเหล่านี้ไปแสดงผลบนหน้าจอหรือบันทึกออกเป็นไฟล์รายงานก็ได้ ขึ้นอยู่กับการเขียนโค้ดส่วนที่เหลือของระบบ

```python
def main_menu():
    while True:
        print("\n===== Library System =====")
        print("1. Books")
        print("2. Members")
        print("3. Borrows")
        print("4. Generate Report")
        print("0. Exit")
        c = input("เลือก: ").strip()
        if c == "1":
            # ... (Book Menu)
            while True:
                print("\n-- Books Menu --")
                print("1. Add Book")
                print("2. View Books")
                print("3. Update Book")
                print("4. Delete Book")
                print("0. Back")
                cc = input("เลือก: ").strip()
                if cc == "1": add_book()
                elif cc == "2": view_books()
                elif cc == "3": update_book()
                elif cc == "4": delete_book()
                elif cc == "0": break
        elif c == "2":
            # ... (Member Menu)
            while True:
                print("\n-- Members Menu --")
                print("1. Add Member")
                print("2. View Members")
                print("3. Update Member")
                print("4. Delete Member")
                print("0. Back")
                cc = input("เลือก: ").strip()
                if cc == "1": add_member()
                elif cc == "2": view_members()
                elif cc == "3": update_member()
                elif cc == "4": delete_member()
                elif cc == "0": break
        elif c == "3":
            while True:
                print("\n-- Borrows Menu --")
                print("1. Add Borrow")
                print("2. View Borrows")
                print("3. Update Borrow")
                print("4. Delete Borrow")
                print("0. Back")
                cc = input("เลือก: ").strip()
                if cc == "1": add_borrow()
                elif cc == "2": view_borrows()
                elif cc == "3": update_borrow()
                elif cc == "4": delete_borrow()
                elif cc == "0": break
                else: print(" เลือกไม่ถูกต้อง")
        elif c == "4":
            generate_report()
        elif c == "0":
            print("Bye")
            break
        else:
            print(" เลือกไม่ถูกต้อง")

if __name__ == "__main__":
    main_menu()
```

4.2.22 main_menu

การทำงานโดยละเอียด

1. **เมนูหลัก (Main Menu)** เมื่อรันโปรแกรมจะมีการวนลูป (while True) เพื่อแสดงเมนูหลักขึ้นมาตลอดเวลา:
    - 1. Books: สำหรับจัดการข้อมูลหนังสือ
    - 2. Members: สำหรับจัดการข้อมูลสมาชิก
    - 3. Borrows: สำหรับจัดการข้อมูลการยืม-คืนหนังสือ
    - 4. Generate Report: สำหรับสร้างรายงาน
    - 0. Exit: สำหรับออกจากโปรแกรม
2. **เมนูย่อย (Sub-menus)** เมื่อผู้ใช้เลือกเมนู 1, 2, หรือ 3 โปรแกรมจะเข้าไปยัง **เมนูย่อย** ของหัวข้อนั้นๆ ซึ่งมีลักษณะการทำงานที่คล้ายกันคือ:
    - 1. Add: เพิ่มข้อมูลใหม่ (เช่น add_book(), add_member())
    - 2. View: ดูข้อมูลทั้งหมด (เช่น view_books(), view_members())
    - 3. Update: แก้ไขข้อมูลที่มีอยู่ (เช่น update_book(), update_member())
    - 4. Delete: ลบข้อมูล (เช่น delete_book(), delete_member())
    - 0. Back: เพื่อกลับไปยัง **เมนูหลัก**
3. **การเรียกใช้ฟังก์ชัน**
    - เมื่อผู้ใช้เลือกรายการในเมนูย่อย โปรแกรมจะเรียกใช้ฟังก์ชันที่เกี่ยวข้อง เช่น ถ้าผู้ใช้อยู่ในเมนู Books และเลือก "1. Add Book" โปรแกรมก็จะเรียกฟังก์ชัน add_book() ให้ทำงาน
    - **ข้อสังเกต:** โค้ดที่ให้มานี้เป็นเพียง **โครงสร้างของเมนู** เท่านั้น ตัวฟังก์ชันที่ทำงานจริงๆ (เช่น add_book(), view_members()) ยังไม่ได้ถูกเขียนขึ้นมา เป็นเพียงการเรียกชื่อฟังก์ชันไว้เฉยๆ
4. **จุดเริ่มต้นของโปรแกรม**
    - บรรทัด if __name__ == "__main__": คือจุดเริ่มต้นของโปรแกรม ซึ่งจะสั่งให้ฟังก์ชัน main_menu() เริ่มทำงานเป็นอันดับแรก

บทที่ 5
สรุปผล อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการดำเนินโครงงาน

โครงงาน "Lending and Returning Library" ได้รับการพัฒนาขึ้นตามวัตถุประสงค์ที่กำหนดไว้ในบทที่ 1 โดยใช้ภาษา **Python 3.x** เป็นเครื่องมือหลักในการสร้างระบบจัดการข้อมูลการยืม–คืนหนังสือในห้องสมุด โดยใช้แนวคิด **Flat-File Database** ในรูปแบบไฟล์ .dat เพื่อจัดเก็บข้อมูลอย่างเป็นระบบ

5.1.1 สรุปผลตามวัตถุประสงค์

1. **เพื่อพัฒนาระบบต้นแบบที่ช่วยจัดการการยืม–คืนหนังสือในห้องสมุดด้วยภาษา Python:**
    - **สำเร็จ:** ได้พัฒนาระบบ Console Application ที่สามารถดำเนินการตามฟังก์ชันการทำงานหลักของการยืม–คืนได้ครบถ้วน โดยใช้โค้ดภาษา Python ในการจัดการไฟล์ข้อมูลโดยตรง
2. **เพื่อสร้างระบบที่สามารถเก็บข้อมูลหนังสือ ข้อมูลสมาชิก และประวัติการยืม–คืนได้อย่างเป็นระบบ:**
    - **สำเร็จ:** ระบบสามารถจัดเก็บข้อมูลอย่างเป็นระบบในรูปแบบ **Fixed-Length Encoding** ใน 3 ไฟล์หลัก คือ books.dat, members.dat, และ borrows.dat ซึ่งช่วยให้การค้นหาและการอ้างอิงข้อมูลระหว่างกันทำได้อย่างมีประสิทธิภาพ
3. **เพื่อลดความซ้ำซ้อน ความล่าช้า และข้อผิดพลาดที่เกิดจากการทำงานแบบเอกสาร:**
    - **สำเร็จ:** กระบวนการยืม–คืนถูกเปลี่ยนจากการบันทึกด้วยกระดาษมาเป็นการประมวลผลอัตโนมัติ ทำให้ลดโอกาสเกิดข้อผิดพลาดในการบันทึกข้อมูลและเพิ่มความรวดเร็วในการให้บริการ
4. **เพื่อคำนวณค่าปรับการคืนหนังสือเกินกำหนดโดยอัตโนมัติ:**
    - **สำเร็จ:** ระบบสามารถคำนวณจำนวนวันที่เกินกำหนดและแสดงผลค่าปรับ (fine_amount) ที่เกิดขึ้นในแต่ละรายการยืม–คืนได้อย่างถูกต้องและอัตโนมัติ ซึ่งปรากฏในรายงานการยืม (Borrowed Report)

5. **เพื่อเป็นต้นแบบในการพัฒนาระบบห้องสมุดดิจิทัลที่สามารถต่อยอดไปสู่การเชื่อมต่อฐานข้อมูลหรือใช้งานในรูปแบบออนไลน์:**

- **สำเร็จ:** ระบบนี้เป็นรากฐานที่มั่นคงในการจัดการตรรกะและโครงสร้างข้อมูลการยืม–คืน ซึ่งสามารถนำไปปรับใช้กับฐานข้อมูลเชิงสัมพันธ์ขนาดใหญ่ในอนาคตได้

5.2 **อภิปรายผล (**Discussion)

การดำเนินโครงงานในครั้งนี้แสดงให้เห็นถึงความสำเร็จในการประยุกต์ใช้ภาษา Python เพื่อแก้ไขปัญหาการจัดการทรัพยากรห้องสมุดด้วยวิธีดั้งเดิม

5.2.1 การอภิปรายผลด้านการจัดการข้อมูล

- ประสิทธิภาพของ Fixed-Length File: การเลือกใช้ไฟล์ .dat แบบ Fixed-Length ในโครงงานนี้ช่วยให้การจัดการข้อมูลขนาดเล็กทำได้อย่างมีประสิทธิภาพ เนื่องจากสามารถคำนวณตำแหน่งของข้อมูลแต่ละฟิลด์ได้อย่างรวดเร็วโดยไม่ต้องอาศัยตัวแบ่ง (Delimiter) อย่างไรก็ตาม วิธีนี้มีข้อจำกัดด้านความยืดหยุ่นในการแก้ไขโครงสร้างข้อมูลในภายหลัง
- การจัดการสถานะที่ถูกต้อง: การที่ระบบสามารถแสดงผลลัพธ์ Borrowed Now เป็น 4 และ Available Now เป็น 21 (อ้างอิงจากรายงานสรุปและรายงานการยืม) แสดงให้เห็นว่าตรรกะในการอัปเดตจำนวนสำเนาคงเหลือ (total_copies) และการนับสถานะ 'Borrow' ในไฟล์ borrows.dat ทำงานได้อย่างแม่นยำ

5.2.2 การอภิปรายผลด้านฟังก์ชันหลัก

- **ความถูกต้องของการคำนวณค่าปรับ:** ระบบมีความโดดเด่นในการใช้โมดูล datetime ของ Python ในการคำนวณค่าปรับที่ซับซ้อนโดยอัตโนมัติ โดยรายงานการยืมได้ยืนยันความสามารถนี้ด้วยการแสดงค่าปรับที่หลากหลาย เช่น 20.00,22.00,165.50, และ 40.00 ซึ่งเป็นการลดภาระงานของบรรณารักษ์ในการคำนวณด้วยมือได้อย่างชัดเจน

5.2.3 ข้อจำกัดของระบบ

ข้อจำกัดที่สำคัญของระบบต้นแบบนี้คือ:

1. **รูปแบบการใช้งาน:** ระบบเป็นแบบ Console Application ทำให้ขาดความสวยงามและใช้งานง่ายเมื่อเทียบกับระบบที่มีส่วนติดต่อผู้ใช้แบบกราฟิก (GUI)

2. **ข้อจำกัดด้านความสามารถในการขยาย (Scalability):** การจัดการข้อมูลแบบ Flat-File จะมีประสิทธิภาพลดลงอย่างมากเมื่อปริมาณข้อมูลเพิ่มขึ้นเป็นจำนวนมาก เนื่องจากทุกครั้งที่มีการแก้ไขข้อมูล จะต้องทำการเขียนทับไฟล์ข้อมูลทั้งหมดใหม่

5.3 ข้อเสนอแนะและแนวทางการพัฒนาในอนาคต

1. การเปลี่ยนไปใช้ฐานข้อมูลเชิงสัมพันธ์ (RDBMS): ควรย้ายการจัดเก็บข้อมูลจากไฟล์ .dat ไปสู่ระบบฐานข้อมูล เช่น SQLite, PostgreSQL, หรือ MySQL เพื่อแก้ไขปัญหาด้านประสิทธิภาพและความยืดหยุ่นในการขยายระบบและการจัดการข้อมูลจำนวนมาก

2. การพัฒนาส่วนติดต่อผู้ใช้แบบกราฟิก (GUI): ควรพัฒนาส่วนติดต่อผู้ใช้โดยใช้ไลบรารีของ Python เช่น Tkinter, PyQt, หรือ Streamlit เพื่อให้ระบบมีความน่าสนใจ ใช้งานง่ายขึ้น และเข้าถึงผู้ใช้ได้หลากหลายขึ้น

3. การเพิ่มฟังก์ชันการแจ้งเตือน: พัฒนาระบบให้สามารถแจ้งเตือนสมาชิกทางอีเมล (Email) หรือ SMS เมื่อหนังสือใกล้ถึงกำหนดคืน หรือเมื่อเลยกำหนดคืนแล้ว เพื่อช่วยลดปัญหาหนังสือค้างยืม

4. การเพิ่มโมดูลสำหรับผู้ดูแลระบบ: พัฒนาระบบการเข้าสู่ระบบ (Login) และการกำหนดสิทธิ์การเข้าถึงข้อมูล เพื่อเพิ่มความปลอดภัยและความน่าเชื่อถือให้กับระบบ