ระบบจัดการข้อมูลหนังสือ

Book Information Management System

นางสาวณัฏฐณิชา เจริญดี	รหัส 6806022610011	Sec1
นายจักรวาล ไวสติ	รหัส 6806022610445	Sec1
นายนิติภูมิ กาฬภักดี	รหัส 6806022610071	Sec1

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ปีการศึกษา 2568

บทคัดย่อ

โครงงานนี้มีวัตถุประสงค์เพื่อพัฒนาระบบห้องสมุดสำหรับการยืม – คืนหนังสือ โดยใช้ภาษา Python เป็นเครื่องมือหลักในการพัฒนา เพื่อแก้ไขปัญหาการจัดเก็บข้อมูลด้วยวิธีดั้งเดิมที่ยังขาดความเป็นระบบ ซึ่งส่งผลให้เกิดความล่าช้าและความผิดพลาดในการค้นหาและติดตามรายการหนังสือ ระบบที่พัฒนาขึ้นสามารถจัดเก็บข้อมูลหนังสือ ข้อมูลสมาชิก การยืม – คืน และประวัติการใช้งานได้อย่างมีประสิทธิภาพ โดยมุ่งเน้นการใช้งานที่ง่าย สะดวก และสามารถลดการใช้เอกสารในรูปแบบกระดาษ นอกจากนี้ยังช่วยเพิ่มความถูกต้องแม่นยำในการบริหารจัดการหนังสือภายในห้องสมุด พร้อมทั้งสามารถต่อยอดเป็นระบบออนไลน์ในอนาคต โครงงานนี้ช่วยให้นักศึกษาได้ฝึกฝนทักษะการเขียนโปรแกรม การออกแบบระบบ การวิเคราะห์ปัญหา และการทำงานร่วมกันเป็นทีม ซึ่งล้วนเป็นทักษะสำคัญในการประกอบวิชาชีพด้านวิศวกรรมสารสนเทศและเครือข่าย

กิตติกรรมประกาศ

คณะผู้จัดทำโครงงาน “ระบบห้องสมุดสำหรับการยืม – คืนหนังสือ” ขอขอบพระคุณอาจารย์ผู้สอนวิชา COMPUTER PROGRAMMING ทุกท่าน ที่ได้ให้คำแนะนำ ความรู้ และการสนับสนุนตลอดระยะเวลาการดำเนินโครงงานนี้ เป็นอย่างดี รวมทั้งผู้ที่มีส่วนเกี่ยวข้องทุกคนที่ให้ความช่วยเหลือ ไม่ว่าจะเป็นข้อมูล แหล่งอ้างอิง หรือคำปรึกษาต่าง ๆ ซึ่งช่วยให้การพัฒนาโครงงานนี้สำเร็จลุล่วงไปด้วยดี

คณะผู้จัดทำขอแสดงความขอบคุณอย่างสูง และหากมีข้อผิดพลาดหรือข้อบกพร่องประการใด คณะผู้จัดทำขอน้อมรับไว้เพื่อปรับปรุงแก้ไขในโอกาสต่อไป

คณะผู้จัดทำ

คำนำ

โครงงาน “ระบบห้องสมุดสำหรับการยืม-คืนหนังสือ” จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของรายวิชา COMPUTER PROGRAMMING ในหลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ โดยมีเป้าหมายเพื่อให้นักศึกษาได้นำความรู้ที่ได้รับจากการเรียนตลอดภาคการศึกษามาประยุกต์ใช้ในการพัฒนาโปรแกรมที่สามารถใช้งานได้จริง ผ่านการออกแบบและเขียนโปรแกรมด้วยภาษา Python ซึ่งเป็นภาษาหลักที่ใช้ในการเรียนการสอนของรายวิชานี้ เพื่อส่งเสริมทักษะด้านการคิดวิเคราะห์ การแก้ไขปัญหาเชิงเทคนิค และการทำงานอย่างมีระบบ ซึ่งล้วนเป็นพื้นฐานสำคัญสำหรับการประกอบอาชีพในสายงานวิศวกรรมสารสนเทศและเครือข่ายในอนาคต

คณะผู้จัดทำขอขอบพระคุณอาจารย์ผู้สอนที่ให้คำแนะนำและความรู้ตลอดการดำเนินโครงงาน และหากมีข้อผิดพลาดหรือข้อบกพร่องประการใด คณะผู้จัดทำขอน้อมรับไว้ด้วยความเคารพ พร้อมทั้งขออภัยมา ณ โอกาสนี้

สารบัญ

สารบัญภาพ

หน้า

สารบัญภาพ(ต่อ)

หน้า

สารบัญภาพ(ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

ในปัจจุบัน ระบบห้องสมุดในหลายแห่งยังคงใช้วิธีการบันทึกข้อมูลการยืม – คืนหนังสือด้วยกระดาษหรือไฟล์เอกสารทั่วไป ซึ่งเสี่ยงต่อการเกิดข้อผิดพลาด เช่น การบันทึกข้อมูลซ้ำ การสูญหายของข้อมูล หรือการค้นหาที่ล่าช้า ส่งผลให้การบริหารจัดการไม่เป็นระบบ

โครงงานนี้จึงมีความสำคัญในการพัฒนาระบบจัดการห้องสมุดแบบง่ายด้วยภาษา Python โดยใช้ไฟล์ไบนารี (.dat) เพื่อจัดเก็บข้อมูลอย่างมีโครงสร้าง สามารถเพิ่ม แก้ไข ลบข้อมูลหนังสือและสมาชิกได้อย่างมีประสิทธิภาพ รวมถึงรองรับฟังก์ชันการยืม–คืนหนังสือ และสามารถสร้างรายงานสรุปข้อมูลได้ ซึ่งช่วยอำนวยความสะดวกแก่ผู้ใช้งาน ลดการใช้เอกสาร และส่งเสริมการจัดการห้องสมุดให้เป็นระบบมากขึ้น

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อพัฒนาระบบการยืม – คืนหนังสือที่มีประสิทธิภาพ

1.2.2 เพื่อฝึกทักษะการเขียนโปรแกรมด้วยภาษา Python

1.2.3 เพื่อเรียนรู้วิธีการจัดการข้อมูลและไฟล์

1.3 ขอบเขตของโครงงาน

1.3.1 ระบบยืม - คืนหนังสือห้องสมุดมีฟังก์ชันพื้นฐานทั้งหมด 13 ฟังก์ชัน

1.3.1.1 เพิ่มหนังสือ

1.3.1.2 แก้ไขหนังสือ

1.3.1.3 ลบหนังสือ

1.3.1.4 ดูข้อมูลหนังสือทั้งหมด

1.3.1.5 เพิ่มสมาชิก

1.3.1.6 แก้ไขสมาชิก

1.3.1.7 ลบสมาชิก

1.3.1.8 ดูข้อมูลสมาชิกทั้งหมด

1.3.1.9 ยืมหนังสือ

1.3.1.10 คืนหนังสือ

1.3.1.11 ดูข้อมูลการยืม

1.3.1.12 เมนูกลางระบบการยืม - คืน

1.3.1.13 ออกจากระบบ

1.3.2 ระบบการยืม – คืนหนังสือห้องสมุดประกอบด้วย 4 ไฟล์ ได้แก่

1. แฟ้มข้อมูลหนังสือ books.dat

2. แฟ้มข้อมูลสมาชิก members.dat

3. แฟ้มข้อมูลการยืม - คืน loans.dat

1.3.3 ระบบการยืม – คืนหนังสือของห้องสมุดจะเก็บข้อมูลหนังสือไว้ในไฟล์ข้อความ (Text File) ชื่อ report ข้อมูลในไฟล์นี้ประกอบด้วยรหัสหนังสือ ชื่อหนังสือ ชื่อผู้เขียน ปีที่พิมพ์ ชื่อผู้ยืม จำนวนหนังสือทั้งหมด รายการผู้ยืม สถานะการยืม จำนวนหนังสือที่ถูกยืม จำนวนหนังสือที่เหลือให้ยืม และสถิติการยืมหนังสือ

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 พัฒนาระบบให้สามารถยืมและคืนหนังสือได้อย่างรวดเร็วและมีประสิทธิภาพ

1.4.2 เสริมทักษะการเขียนโปรแกรมและการแก้ไขปัญหา

1.4.3 เรียนรู้การจัดเก็บและจัดการข้อมูลในไฟล์อย่างเป็นระบบ

1.4.4 ฝึกการทำงานร่วมกับผู้อื่นในลักษณะทีมเวิร์ก

1.5 ข้อจำกัดของโครงงาน

1.5.1 ไม่รองรับการยืม – คืนแบบออนไลน์ผ่านอินเทอร์เน็ต

โครงงานนี้เป็น ระบบที่ทำงานแบบออฟไลน์ (Offline) และทำงานผ่าน Command Line Interface (CLI) เท่านั้น ไม่สามารถเข้าถึงหรือใช้งานผ่านเว็บเบราว์เซอร์หรือระบบเครือข่ายออนไลน์ได้

1.5.2 ไม่มีระบบยืนยันตัวตนหรือ Login ของผู้ใช้ (Authentication)

สมาชิกไม่สามารถเข้าสู่ระบบเพื่อจัดการข้อมูลตนเองได้ ต้องผ่านผู้ดูแลระบบเท่านั้น

1.5.3 ไม่มีระบบแจ้งเตือนการคืนหนังสือ

ระบบไม่มีฟีเจอร์แจ้งเตือนเมื่อใกล้ถึงกำหนดคืนหนังสือ เช่น Email หรือ SMS

1.5.4 ไม่รองรับการจัดการหนังสือที่สูญหายหรือชำรุด

ระบบไม่มีการบันทึกหรือจัดการสถานะ “สูญหาย” หรือ “เสียหาย” ของหนังสือ

1.5.5 ไม่มีระบบตรวจสอบหรือจำกัดการยืมซ้ำหนังสือเล่มเดิม

สมาชิกสามารถยืมหนังสือเล่มเดียวกันได้หลายครั้งติดต่อกันโดยไม่มีการตรวจสอบประวัติ

1.5.6 การจัดการข้อมูลทำงานผ่านไฟล์ไบนารี (.dat) เท่านั้น

ไม่ใช้งานฐานข้อมูลมาตรฐาน เช่น SQLite หรือ MySQL ทำให้ความสามารถในการขยายระบบมีข้อจำกัด

1.5.7 ไม่มีระบบสำรองข้อมูลอัตโนมัติ

หากไฟล์ `.dat` เสียหาย ข้อมูลอาจสูญหายโดยไม่สามารถกู้คืนได้

1.6 เครื่องมือที่คาดว่าจะใช้

1.6.1 โปรแกรม Visual Studio Code

1.6.2 Microsoft Office

บทที่ 2
ระบบยืม - คืนหนังสือห้องสมุด

2.1 แฟ้มข้อมูลหนังสือ books.dat

แฟ้มข้อมูลหนังสือประกอบด้วย 8 ฟิลด์สำคัญ แต่ละฟิลด์มีรายละเอียดและบทบาทในการจัดเก็บข้อมูลดังนี้

ฟิลด์	ขนิด	ขนาด(bytes)	คำอธิบาย
Book_ID	I, 1s	5	รหัสหนังสือ
Book_Title	50s	50	ชื่อหนังสือ
Book_Category	30s	30	หมวดหมู่หนังสือ
Author_Name	30s	30	ชื่อผู้แต่งหนังสือ
Publisher_Name	50s	50	ชื่อสำนักพิมพ์
Book_year	I	4	ปีที่พิมพ์
Book_copies	I	3	จำนวนหนังสือ
Book_status	I	4	สถานะ Active = 1 Deleted = 0

ตารางที่ 2-1 แฟ้มข้อมูลหนังสือ books.dat

2.1.1 Book_ID (รหัสหนังสือ) เป็นรหัสประจำหนังสือที่ใช้ในการระบุหนังสือแต่ละเล่มอย่างเฉพาะเจาะจงซึ่งมักจะเป็นตัวเลขที่ไม่ซ้ำกันในระบบ ซึ่งรูปแบบของฟิลด์ เป็นประเภทข้อมูล string, integer (ตัวอักษร,จำนวนเต็ม) เช่น B001, B002, B003 เป็นต้น ความสำคัญของฟิลด์คือ รหัสนี้มีความสำคัญเพราะใช้ในการค้นหาหรืออ้างอิงหนังสือในฐานข้อมูลได้ง่ายและช่วยในการป้องกันความสับสนระหว่างหนังสือหลายเล่มที่อาจมีชื่อเดียวกัน

2.1.2 Book_Title (ชื่อหนังสือ) ชื่อของหนังสือ ซึ่งสามารถระบุชื่อเรื่อง ซึ่งรูปแบบของฟิลด์ เป็นประเภทข้อมูล string (ข้อความ) เช่น “Basic Python” เป็นต้น ความสำคัญของฟิลด์คือ ชื่อหนังสือเป็นข้อมูลที่สำคัญสำหรับการค้นหาและการแสดงผลข้อมูลหนังสือโดยเฉพาะเมื่อผู้ใช้ต้องการค้นหาหนังสือที่ต้องการโดยใช้ชื่อ

2.1.3 Book_Category (หมวดหมู่ของหนังสือ) หมวดหมู่ของหนังสือ จะช่วยในการจัดกลุ่มหนังสือตามประเภท เช่น การเรียน (Education), นวนิยาย (Novel), เบ็ดเตล็ดหรือความรู้ทั่วไป (Generalities) เป็นต้น ซึ่งรูปแบบของฟิลด์ เป็นประเภทข้อมูล string

(ข้อความ) ซึ่งกำหนดไว้เป็น “Education”, “Novel”, “Generalities” ความสำคัญของฟิลด์คือ หมวดหมู่ที่ช่วยในการกรองหนังสือให้กับผู้ใช้ และทำให้การค้นหาและการจัดแสดงหนังสือทำได้ง่ายขึ้น

2.1.4 Author_Name (ชื่อผู้แต่งหนังสือ) ชื่อของผู้แต่ง สามารถระบุชื่อผู้แต่งของหนังสือแต่ละเล่ม ซึ่งรูปแบบของฟิลด์ เป็นประเภทข้อมูล string (ข้อความ) เช่น “Albert Camus”, “Herman Melville” เป็นต้น ความสำคัญของฟิลด์คือ ชื่อของผู้แต่งเป็นข้อมูลสำคัญสำหรับผู้ใช้งานที่ต้องการหาหนังสือของผู้แต่งหนังสือแต่ละท่าน

2.1.5 Publisher_Name (ชื่อสำนักพิมพ์) ชื่อของสำนักพิมพ์ สามารถระบุชื่อของสำนักพิมพ์ของหนังสือแต่ละเล่ม ซึ่งรูปแบบของฟิลด์ เป็นประเภทข้อมูล string (ข้อความ) เช่น “ASIABOOKS”, “avocadobooks” เป็นต้น ความสำคัญของฟิลด์คือ ชื่อสำนักพิมพ์เป็นข้อมูลสำคัญสำหรับนักวิจัยหรือผู้อ่าน ที่ต้องการหาหนังสือจากสำนักพิมพ์ที่ตนเองไว้ใจ

2.1.6 Book_year (ปีที่วางขายหนังสือ) ปีที่วางขายหนังสือ สามารถระบุปีของหนังสือที่วางขายแต่ละเล่ม ซึ่งรูปแบบฟิลด์คือ Integer (ตัวเลข) กำหนดรูปแบบเป็น “YYYY” หรือ “1998” ความสำคัญของฟิลด์คือ เพื่อให้ผู้อ่านตรวจสอบว่าข้อมูลของหนังสือที่ตนเองต้องการ เป็นข้อมูลที่เก่าหรือใหม่

2.1.7 Book_copies (จำนวนหนังสือ) จำนวนหนังสือ สามารถระบุจำนวนหนังสือแต่ละเล่มที่มีอยู่ในระบบและจำนวนหนังสือที่สามารถยืมได้ ซึ่งรูปแบบฟิลด์คือ Integer (ตัวเลข) เช่น 5, 6 ความสำคัญของฟิลด์คือ จำนวนหนังสือเป็นข้อมูลสำคัญที่จะทำให้รับรู้ว่าหนังสือแต่ละเล่มในระบบเหลือเท่าไหร่ และมีจำนวนที่สามารถยืมได้อีกกี่เล่ม

2.1.8 Book_status (สถานะหนังสือ) สถานะหนังสือ สามารถตรวจสอบว่าหนังสืออยู่ในระบบหรือไม่ ซึ่งรูปแบบฟิลด์คือ String (ข้อความ) กำหนดรูปแบบเป็น “Active”, “Deleted” ความสำคัญของฟิลด์คือ เพื่อให้ผู้ใช้งานตรวจสอบว่าหนังสือแต่ละเล่ม มีอยู่ในระบบหรือไม่

2.2 แฟ้มข้อมูลสมาชิก members.dat

แฟ้มข้อมูลสมาชิกมี 5 ฟิลด์สำคัญ โดยแต่ละฟิลด์มีรายละเอียดและหน้าที่ดังต่อไปนี้

ฟิลด์	ชนิด	ขนาด(bytes)	คำอธิบาย
Member_ID	I, 1s	5	รหัสสมาชิก
Member_Namae	50s	50	ชื่อ – นามสกุล
Member_Birth	I	4	วัน - เดือน – ปีเกิด
Max_loan	I	4	จำนวนหนังสือสูงสุดที่ยืมได้
Member_status	I	4	สถานะ Active = 1 Deleted = 0

ตารางที่ 2-2 แฟ้มข้อมูลสมาชิก members.dat

2.2.2 Member_ID รหัสสมาชิก

Member_ID เป็นรหัสที่ใช้ระบุสมาชิกแต่ละคนในระบบ ฟิลด์นี้เป็นตัวอักษรและตัวเลขจำนวนเต็ม (String,Integer) มีความเป็นเอกลักษณ์ไม่ซ้ำกัน เช่น M001, M002 การมีรหัสสมาชิกช่วยให้ระบบสามารถจัดการข้อมูลสมาชิกจำนวนมากได้อย่างถูกต้องและรวดเร็ว

2.2.3 Member_Name ชื่อสมาชิก

Member_Name เป็นฟิลด์เก็บข้อความ (String) ความยาวสูงสุด 50 ตัวอักษร ใช้เก็บชื่อ - นามสกุลของสมาชิก เช่น "เอกภพ ยอดดี", "สมหมาย หมายปอง" ฟิลด์นี้เป็นข้อมูลสำคัญเพื่อแสดงผลและยืนยันตัวตนของสมาชิก

2.2.4 Member_Birth วัน-เดือน-ปีเกิด

Member_Birth เป็นฟิลด์เก็บจำนวนเต็ม (Integer) ใช้เก็บวันเดือนปีเกิดของสมาชิก เช่น 05-12-2005, 03-07-2003 ข้อมูลนี้อาจถูกนำไปใช้ในการตรวจสอบอายุ การจัดกลุ่มสมาชิกตามช่วงวัย หรือใช้กำหนดสิทธิพิเศษบางอย่างสำหรับสมาชิกที่เกิดในวันหรือเดือนนั้น ๆ

2.2.5 Max_loan จำนวนหนังสือสูงสุดที่ยืมได้

Max_oan เป็นฟิลด์จำนวนเต็ม (Integer) ที่กำหนดจำนวนสูงสุดของหนังสือที่สมาชิกแต่ละคนสามารถยืมได้ เช่น 4,5,6 ช่วยควบคุมการยืมหนังสือเพื่อป้องกันไม่ให้มีการยืมเกินขีดจำกัดที่ระบบกำหนด

2.2.6 Member_status สถานะ Active,Deleted

Member_status เป็นฟิลด์จำนวนเต็ม (Integer) ใช้เก็บสถานะของสมาชิก เช่น 1 = Active, 0 = Delete ข้อมูลนี้ช่วยให้สามารถตรวจสอบได้ว่าสมาชิกที่ตรวจสอบ ยังอยู่ในระบบหรือไม่

2.3 แฟ้มข้อมูลการยืม – คืน loans.dat

แฟ้มข้อมูลการยืม - คืน มี 8 ฟิลด์สำคัญ โดยแต่ละฟิลด์มีรายละเอียดและหน้าที่ดังต่อไปนี้

ฟิลด์	ชนิด	ขนาด (bytes)	คำอธิบาย
Loan_ID	I, 1s	5	แยก record แต่ละครั้ง เพราะสมาชิกอาจยืมเล่มเดิมได้หลายครั้ง
Operation_type	I	4	1 = add, 2 = update, 3 = delete, 4 = view
Member_ID	I, 1s	5	ใช้เชื่อมโยงกับ members.dat
Book_ID	I, 1s	5	ใช้เชื่อมโยงกับ book.dat
Loan_Date	I	8	วันที่ยืม (YYYY-MM-DD)
Due_Date	I	8	กำหนดวันคืน
Return_Date	I	8	วันที่คืน
Loan_Status	I	1	แสดงสถานะว่าถูกยืมอยู่ หรือว่าคืนแล้ว

ตารางที่ 2-3 แฟ้มข้อมูลการยืม – คืน loans.**dat**

2.3.1 Loan_ID (รหัสการยืม - คืน)

เป็นฟิลด์จำนวนเต็ม (integer ขนาด 5 ไบต์) ใช้ในการระบุเอกลักษณ์ของการยืมแต่ละครั้ง เนื่องจากสมาชิกอาจทำการยืมหนังสือซ้ำได้หลายครั้ง ดังนั้นฟิลด์นี้จึงมีความสำคัญในการแยกแต่ละธุรกรรมออกจากกัน

2.3.2 Operation_type (รหัสดำเนินการ)

เป็นฟิลด์จำนวนเต็ม (integer ขนาด 4 ไบต์) ใช้ระบุประเภทของการทำงาน ได้แก่

1 = เพิ่มข้อมูล (ADD)

2 = แก้ไขข้อมูล (UPDATE)

3 = ลบข้อมูล (DELETE)

4 = เรียกดูข้อมูล (VIEW)

ช่วยในการตรวจสอบการเปลี่ยนแปลงที่เกิดขึ้นภายในระบบยืม–คืนหนังสือ

2.3.3 Member_ID (รหัสสมาชิก)

เป็นฟิลด์จำนวนเต็ม (integer ขนาด 5 ไบต์) ใช้เก็บรหัสสมาชิกซึ่งเชื่อมโยงกับไฟล์ members.dat เพื่อระบุว่าสมาชิกคนใดเป็นผู้ดำเนินการยืม–คืน

2.3.4 Book_ID (รหัสหนังสือ)

เป็นฟิลด์จำนวนเต็ม (integer ขนาด 5 ไบต์) ใช้เก็บรหัสหนังสือซึ่งเชื่อมโยงกับไฟล์ books.dat เพื่อบ่งบอกว่าการยืม–คืนครั้งนี้เกี่ยวข้องกับหนังสือเล่มใด

2.3.5 Loan_Date (วันที่ยืม)

เป็นฟิลด์ข้อความ (string ขนาด 8 ไบต์) ใช้เก็บวันที่ที่สมาชิกยืมหนังสือ โดยใช้รูปแบบ YYYY-MM-DD เช่น 2025-09-30 ข้อมูลนี้ใช้ตรวจสอบวันเริ่มต้นการยืม

2.3.6 Due_Date (กำหนดวันคืน)

เป็นฟิลด์ข้อความ (string ขนาด 8 ไบต์) ใช้เก็บวันที่ครบกำหนดคืนหนังสือในรูปแบบ YYYY-MM-DD เช่น 2025-10-07 เพื่อใช้ตรวจสอบว่าหนังสือถูกคืนตรงกำหนดหรือไม่

2.3.7 Return_Date (วันที่คืน)

เป็นฟิลด์ข้อความ (string ขนาด 8 ไบต์) ใช้เก็บวันที่ทำการคืนหนังสือจริงในรูปแบบ YYYY-MM-DD หากยังไม่ได้คืน ฟิลด์นี้จะเว้นว่าง

2.3.8 Loan_Status (สถานะการยืม)

เป็นฟิลด์จำนวนเต็ม (integer ขนาด 1 ไบต์) ใช้เก็บสถานะปัจจุบันของการยืม เช่น

1 = กำลังถูกยืม

0 = คืนแล้ว

2.4 ไฟล์ report.txt

ไฟล์ report.txt ใช้เพื่อสร้างรายงานสรุปเกี่ยวกับการยืม – คืนหนังสือของห้องสมุด โดยรวบรวมข้อมูลจากแฟ้ม members.dat, books.dat และ loans.dat แล้วจัดทำออกมาเป็นรายงานในรูปแบบข้อความ (Text Report)

```
Library Borrow System - Summary Report
Generated At : 2025-10-02 10:55:23 (+07:00)
App Version  : 1.0
Encoding     : UTF-8

MemberID | MemberName  | BookID         | Titles                | LoanDate   | DueDate    | ReturnDate | Status
---------+-------------+----------------+-----------------------+------------+------------+------------+---------
M004     | Jakkawal    | B002,B003      | -,English I           | 2025-09-30 | 2025-10-07 | 2025-09-30 | Returned
M001     | Natthanicha | B003           | English I             | 2025-09-30 | 2025-10-07 | -          | Borrowed
M003     | Phatchanoon | B001,B005,B003 | Compro,html,English I | 2025-10-01 | 2025-10-08 | 2025-10-01 | Returned

Summary (Active Books Only)
- Total Books : 5
- Active Books : 5
- Deleted Books : 0
- Borrowed Now : 1
- Available Now : 28

Borrow Statistics (Active only)
- Most Borrowed Book : English I (B003) (1 times)
- Currently Borrowed : 1
- Active Members : 3
```

ภาพที่ 2-1 ไฟล์ report.txt

2.4.1 header_text ชื่อส่วนหัวของรายงาน

เป็นฟิลด์ชนิดข้อความ (string ความยาวไม่เกิน 100 ไบต์) ใช้สำหรับเก็บชื่อของรายงาน เช่น "Library Borrow System – Summary Report" เพื่อให้ผู้อ่านเข้าใจได้ทันทีว่าเอกสารนี้เป็นรายงานประเภทใด หรือมีวัตถุประสงค์อย่างไร

2.4.2 generated_at วันที่และเวลาที่จัดทำรายงาน

ฟิลด์นี้เป็นข้อความ (string ขนาดไม่เกิน 25 ไบต์) ใช้สำหรับบันทึกวันและเวลาที่รายงานถูกสร้างขึ้น โดยใช้รูปแบบ "YYYY-MM-DD HH:MM" เช่น "2025-10-01 09:30" เพื่อใช้ในการอ้างอิงหรือบันทึกประวัติการสร้างรายงาน

2.4.3 app_version เวอร์ชันของแอปพลิเคชัน

ฟิลด์นี้มีขนาดข้อความไม่เกิน 10 ไบต์ ใช้เก็บหมายเลขเวอร์ชันของโปรแกรมที่สร้างรายงาน เช่น "1.0", "2.1.5" เป็นต้น เพื่อตรวจสอบว่าไฟล์รายงานนี้ถูกสร้างขึ้นจากเวอร์ชันใดของระบบ

2.4.4 encoding รูปแบบการเข้ารหัส

ฟิลด์ข้อความ (string ความยาวสูงสุด 20 ไบต์) นี้ใช้สำหรับระบุรูปแบบของการเข้ารหัสข้อมูลในรายงาน เช่น "UTF-8" หรือ "ISO8859-1" เพื่อให้แน่ใจว่าเมื่อเปิดไฟล์จะแสดงผลถูกต้องตามภาษาที่ใช้

2.4.5 book_table_header หัวคอลัมน์ของข้อมูลหนังสือ

เป็นฟิลด์ข้อความ (string ความยาวไม่เกิน 80 ไบต์) สำหรับเก็บชื่อหัวคอลัมน์ของตารางหนังสือ เช่น "BookID | Title | Author | Year | Copies | Borrowed By | Status" ซึ่งกำหนดโครงสร้างของข้อมูลที่จะแสดงในรายงาน

2.4.6 book_records ข้อมูลหนังสือในรูปแบบตาราง

ฟิลด์นี้เป็นข้อความแบบความยาวคงที่ (string 120 * N ไบต์ โดย N คือจำนวนรายการหนังสือ) ใช้เก็บข้อมูลแต่ละรายการของหนังสือ เช่น รหัสหนังสือ, ชื่อเรื่อง, ผู้แต่ง, ปีที่พิมพ์, จำนวนเล่ม, ผู้ที่ยืม และสถานะของหนังสือนั้น ๆ

2.4.7 summary_section สรุปภาพรวมของข้อมูล

เป็นข้อความ (string ขนาด 150 ไบต์) สำหรับบันทึกข้อมูลโดยสรุป เช่น จำนวนหนังสือทั้งหมด, จำนวนหนังสือที่ยังใช้งานได้, หนังสือที่ถูกลบ, ที่ถูกยืม และที่ยังว่างอยู่ ช่วยให้ผู้ใช้งานเห็นภาพรวมของคลังหนังสือ

2.4.8 statistics_section ข้อมูลเชิงสถิติการยืม

เป็นฟิลด์ข้อความ (string 150 ไบต์) ใช้เก็บข้อมูลทางสถิติ เช่น หนังสือที่มีการยืมมากที่สุด, จำนวนหนังสือที่กำลังถูกยืมในปัจจุบัน และจำนวนสมาชิกที่ยังมีการยืมอยู่ ข้อมูลในส่วนนี้ช่วยวิเคราะห์แนวโน้มการใช้งานของผู้ใช้ และช่วยวางแผนการจัดการทรัพยากรหนังสือในอนาคต

บทที่ 3
การใช้งานระบบยืม – คืนหนังสือ

โปรแกรมระบบยืม – คืนหนังสือห้องสมุดถูกพัฒนาขึ้นเพื่อช่วยให้การจัดการข้อมูลหนังสือและสมาชิกเป็นไปอย่างสะดวกและมีประสิทธิภาพมากขึ้น โดยระบบสามารถเพิ่ม แก้ไข ค้นหา และลบข้อมูลหนังสือและสมาชิกได้ รวมถึงรองรับการทำรายการยืม – คืนอย่างเป็นระบบ พร้อมทั้งสร้างรายงานสรุปในรูปแบบไฟล์ข้อความ เพื่อใช้ในการตรวจสอบและเก็บเป็นหลักฐาน ทำให้การบริหารจัดการห้องสมุดมีความถูกต้อง รวดเร็ว และเป็นระเบียบมากยิ่งขึ้น

สำหรับผู้ใช้งานโปรแกรม

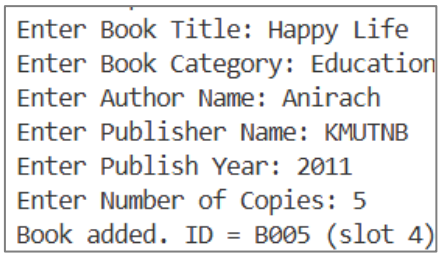
3.1 การใช้งานโปรแกรมระบบยืม – คืนหนังสือ

```
===== Library Management =====
1. Add Book
2. Update Book
3. Delete Book
4. View Books
5. Add Member
6. Update Member
7. Delete Member
8. View Members
9. Borrow Book
10. Return Book
11. View Loans
12. Generate Report (.txt)
0. Exit
Choose option:
```

ภาพที่ 3-1 การเลือกใช้งานฟังก์ชัน

3.1.1 การเพิ่มข้อมูลหนังสือ (add_book)

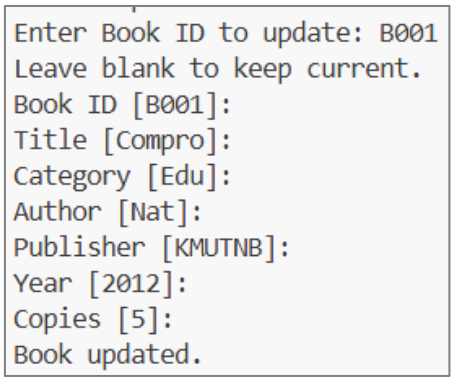
ผู้ใช้สามารถกดหมายเลข 1 ภายในเมนูหลักเพื่อเรียกฟังก์ชัน add_book ระบบจะแสดงคำสั่งให้กรอกข้อมูลของหนังสือ เมื่อกรอกข้อมูลครบ ระบบจะสร้างรหัสประจำหนังสือ (Book ID) อัตโนมัติ และบันทึกข้อมูลลงในฐานข้อมูล พร้อมแสดงข้อความยืนยัน เช่น “Book added. ID = B001



ภาพที่ 3-2 การเพิ่มข้อมูลหนังสือ

3.1.2 การแก้ไขข้อมูลหนังสือ (update_book)

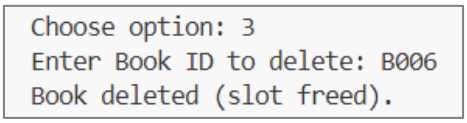
ผู้ใช้กดหมายเลข 2 เพื่อแก้ไขข้อมูลหนังสือ ระบบจะให้กรอกรหัสหนังสือ (Book ID) ที่ต้องการแก้ไข หากพบข้อมูล ระบบจะแสดงข้อมูลเดิมให้ผู้ใช้กรอกใหม่ หรือเว้นว่างเพื่อคงค่าดังกล่าวไว้ เมื่อแก้ไขเสร็จสิ้น ระบบจะแสดงข้อความ “Book updated.”



ภาพที่ 3-3 การแก้ไขข้อมูลหนังสือในฟังก์ชัน update_book

3.1.3 การลบข้อมูลหนังสือ (delete_book)

ผู้ใช้กดหมายเลข 3 และกรอกรหัสหนังสือที่ต้องการลบ เมื่อยืนยันแล้ว ระบบจะเปลี่ยนสถานะของหนังสือเป็น Deleted และจัดเก็บตำแหน่งข้อมูลไว้เพื่อใช้ซ้ำในอนาคต พร้อมแสดงข้อความ “Book deleted (slot freed).”



ภาพที่ 3-4 การลบข้อมูลหนังสือในฟังก์ชัน delete_book

3.1.4 การแสดงรายการหนังสือ (view_books)

เมื่อผู้ใช้กดหมายเลข 4 ระบบจะแสดงรายการหนังสือทั้งหมดในรูปแบบตาราง โดยแสดงข้อมูล ID, ชื่อเรื่อง, หมวดหมู่, ผู้แต่ง, จำนวนเล่ม และสถานะ (Active หรือ Deleted)

```
--- Books ---
ID   | Title                          | Category      | Author         | Copies | Status
-------------------------------------------------------------------------------------------
B001 | Compro                         | Edu           | Nat            | 5      | Active
B003 | English I                      | Edu           | Nitigan        | 9      | Active
B006 | Python                         | edu           | anirach        | 5      | Active
B005 | html                           | edu           | nitigan        | 4      | Active
B007 | INE                            | edu           | niti           | 5      | Active
```

ภาพที่ 3-5 แสดงข้อมูลหนังสือทั้งหมดในฟังก์ชัน books.dat

3.1.5 การเพิ่มข้อมูลสมาชิก (add_member)

ผู้ใช้กดหมายเลข 5 เพื่อเพิ่มสมาชิก ระบบจะแสดงข้อความให้กรอกข้อมูล ได้แก่ ชื่อสมาชิก และวันเกิด (YYYY-MM-DD) ระบบจะสร้าง Member ID ให้อัตโนมัติ เช่น M001 และแสดงข้อความยืนยัน เช่น “Member added. ID = M001. Max loan = 5”

```
Enter Member Name:
Enter Birth Date (YYYY-MM-DD):
Member added. ID = M005 (slot 3). Max loan = 5
```

ภาพที่ 3-6 การเพิ่มข้อมูลสมาชิกในฟังก์ชัน add_member

3.1.6 การแก้ไขข้อมูลสมาชิก (update_member)

เมื่อกดหมายเลข 6 ผู้ใช้สามารถแก้ไขชื่อหรือวันเกิดของสมาชิกได้ โดยกรอก Member ID ที่ต้องการแก้ไข ระบบจะแสดงค่าปัจจุบันเพื่อให้ผู้ใช้เลือกแก้ไขหรือละเว้น

```
Enter Member ID to update: M001
Leave blank to keep current.
Name [Natthanicha]:
Birth [2006-11-29]:
Member updated.
```

ภาพที่ 3-7 การแก้ไขข้อมูลสมาชิกในฟังก์ชัน update_member

3.1.7 การลบสมาชิก (delete_member)

การกดหมายเลข 7 จะเป็นการลบสมาชิก โดยระบบจะเปลี่ยนสถานะเป็น Deleted และเก็บตำแหน่งเพื่อใช้ซ้ำ พร้อมข้อความยืนยันการลบ

```
Enter Member ID to delete:
Member not found.
```

ภาพที่ 3-8 การลบข้อมูลสมาชิกในฟังก์ชัน delete_member

3.1.8 การแสดงรายการสมาชิก (view_members)

เมื่อกดหมายเลข 8 ระบบจะแสดงรายชื่อสมาชิกทั้งหมด ทั้ง Active และ Deleted พร้อมรายละเอียด เช่น Member ID, ชื่อ, วันเกิด, สิทธิ์ในการยืมสูงสุด และสถานะ

```
--- Members ---
ID   | Name                          | Birth      | MaxLoan | Status
-------------------------------------------------------------------------------
M001 | Natthanicha                   | 2006-11-29 | 5       | Active
M003 | Phatchanoon                   | 2016-12-26 | 5       | Active
M004 | Jakkawal                      | 2020       | 5       | Active
M005 | Nitipoom                      | 2006-12-22 | 5       | Active
```

ภาพที่ 3-9 ดูสมาชิกทั้งหมดใน members.dat

3.1.9 การยืมหนังสือ (borrow_book)

ผู้ใช้กดหมายเลข 9 ระบบจะแจ้งให้กรอก Member ID และ Book IDs ที่ต้องการยืม (สามารถกรอกหลายเล่มคั่นด้วยเครื่องหมายจุลภาค ,) หากสมาชิกยังไม่ถึงจำนวนสูงสุดที่สามารถยืมได้ และหนังสือมีเล่มว่าง ระบบจะหักจำนวนเล่มและสร้างรายการยืม (Loan ID) ให้โดยอัตโนมัติ พร้อมกำหนดวันครบกำหนดคืน (Due Date) เช่น 7 วันถัดไป

```
Choose option: 9
Enter Member ID: M005
Enter Book IDs to borrow (comma separated): B001
Borrow successful. Loan ID = L007 (slot 6). Due date: 2025-10-09
```

ภาพที่ 3-10 การยืมหนังสือในฟังก์ชัน borrow_book

3.1.10 การคืนหนังสือ (return_book)

เมื่อกดหมายเลข 10 ระบบจะแจ้งให้กรอก Loan ID ที่ต้องการคืน หากพบข้อมูลระบบจะปรับสถานะเป็นคืนแล้ว (Returned) และเพิ่มจำนวนเล่มของหนังสือกลับไป พร้อมข้อความยืนยัน เช่น “Book B001 returned successfully. Loan L001 closed.”

```
Choose option: 10
Enter Loan IDs to return (comma separated): L007
Book B001 returned successfully. Loan L007 closed.
```

ภาพที่ 3-11 การคืนหนังสือในฟังก์ชัน return_book

3.1.11 การแสดงรายการยืม – คืน (view_loans)

การกดหมายเลข 11 จะแสดงประวัติการยืมและคืนหนังสือทั้งหมด โดยมีรายละเอียด Loan ID, Member ID, Book ID, วันยืม, วันครบกำหนดคืน, วันคืน และสถานะ (Borrowed หรือ Returned)

```
--- Loans ---
LoanID | MemID | BookID | LoanDate   | DueDate    | ReturnDate | Status
-------------------------------------------------------------------------------------
L001   | M004  | B002   | 2025-09-30 | 2025-10-07 | 2025-09-30 | Returned
L002   | M004  | B003   | 2025-09-30 | 2025-10-07 | 2025-09-30 | Returned
L003   | M001  | B003   | 2025-09-30 | 2025-10-07 | -          | Borrowed
L004   | M003  | B001   | 2025-10-01 | 2025-10-08 | 2025-10-01 | Returned
L005   | M003  | B005   | 2025-10-01 | 2025-10-08 | 2025-10-01 | Returned
L006   | M003  | B003   | 2025-10-01 | 2025-10-08 | 2025-10-01 | Returned
L007   | M005  | B001   | 2025-10-02 | 2025-10-09 | 2025-10-02 | Returned
```

ภาพที่ 3-12 แสดงรายการยืมทั้งหมดใน loans.dat

3.1.12 การสร้างรายงาน (generate_report)

เมื่อเลือกหมายเลข 12 ระบบจะประมวลผลและสร้างไฟล์ report.txt ซึ่งมีข้อมูลสถิติของห้องสมุด

3.1.13 การออกจากระบบ (Exit)

เมื่อกดหมายเลข 0 ระบบจะปิดการทำงานทันที และแสดงข้อความ “Exiting. Bye.”

บทที่ 4
อธิบายการทำงานของ Code

4.1 ฟังก์ชันพื้นฐานในระบบจัดการข้อมูลหนังสือ

4.2.1 ฟังก์ชัน main_menu()

ฟังค์ชัน main_menu() เป็นฟังค์ชันหลักของโปรแกรมระบบจัดการข้อมูลหนังสือ ทำหน้าที่หลักในการแสดงเมนูการจัดการข้อมูลหนังสือ และรับคำสั่งจากผู้ใช้เพื่อนำไปเรียกใช้ฟังก์ชันย่อยอื่น ๆ ตามที่เลือกจากเมนู ฟังก์ชันนี้ทำงานวนซ้ำในลูปจนกว่าผู้ใช้จะเลือกออกจากโปรแกรม โดยรายละเอียดชองฟังก์ชันมีดังนี้

4.2.1.1 โครงสร้างและการทำงานของฟังก์ชัน main_menu()

แสดงเมนูหลักเมื่อโปรแกรมเริ่มต้น ฟังก์ชัน main_menu() จะทำการแสดงเมนูหลักที่มีตัวเลือกต่าง ๆ ให้กับผู้ใช้

ผู้ใช้สามารถเลือกตัวตามหมายเลขที่ต้องการ

กด 1 เพื่อเพิ่มหนังสือ

กด 2 เพื่อแก้ไขหนังสือ

กด 3 เพื่อลบหนังสือ

กด 4 เพื่อดูหนังสือทั้งหมด

กด 5 เพื่อเพิ่มผู้ใช้บริการ

กด 6 เพื่อแก้ไขผู้ใช้บริการ

กด 7 เพื่อลบผู้ใช้บริการ

กด 8 เพื่อดูผู้ใช้บริการทั้งหมด

กด 9 เพื่อยืมหนังสือ

กด 10 เพื่อคืนหนังสือ

กด 11 เพื่อดูข้อมูลการคืนหนังสือ

กด 12 เพื่อสร้างไฟล์ Report (.txt)

กด 0 เพื่อออกจากโปรแกรม

การรับค่าการเลือกจากผู้ใช้

ฟังก์ชัน input() จะถูกใช้ในการรับค่าจากผู้ใช้ โดยผู้ใช้ต้องใส่หมายเลขที่สอดคล้องกับฟังก์ชันที่ต้องการทำงาน

การเรียกใช้ฟังก์ชันต่าง ๆ ตามการเลือกของผู้ใช้

เมื่อผู้ใช้ใส่หมายเลขเลือกแล้ว โปรแกรมจะตรวจสอบค่า และเรียกใช้ฟังก์ชันที่สอดคล้องตามการเลือกนั้น หากผู้ใช้เลือกหมายเลขที่ไม่ถูกต้อง จะมีการแจ้งเตือนให้ผู้ใช้ทราบและขอให้ใส่หมายเลขใหม่

การทำงานต่อเนื่อง

หลังจากการทำงานของฟังก์ชันที่ถูกเลือกเสร็จสิ้น ฟังก์ชัน main_menu() จะถูกเรียกซ้ำเพื่อแสดงเมนูอีกครั้ง ทำให้โปรแกรมทำงานต่อเนื่อ จนกว่าผู้ใช้จะเลือกออกจากโปรแกรม

4.2.1.2 โค้ดของฟังก์ชัน main_menu()

```
def main_menu():
    init_all_files()
    while True:
        print("\n===== Library Management =====")
        print("1. Add Book")
        print("2. Update Book")
        print("3. Delete Book")
        print("4. View Books")
        print("5. Add Member")
        print("6. Update Member")
        print("7. Delete Member")
        print("8. View Members")
        print("9. Borrow Book")
        print("10. Return Book")
        print("11. View Loans")
        print("12. Generate Report (.txt)")
        print("0. Exit")
```

ภาพที่ 4-1 โค้ดของฟังก์ชัน main_menu()

```
        choice = input("Choose option: ").strip()
        if choice == "1":
            add_book()
        elif choice == "2":
            update_book()
        elif choice == "3":
            delete_book()
        elif choice == "4":
            view_books()
        elif choice == "5":
            add_member()
        elif choice == "6":
            update_member()
        elif choice == "7":
            delete_member()
        elif choice == "8":
            view_members()
        elif choice == "9":
            borrow_book()
        elif choice == "10":
            return_book()
        elif choice == "11":
            view_loans()
        elif choice == "12":
            generate_report()
        elif choice == "0":
            print("Exiting. Bye.")
            sys.exit(0)
        else:
            print("Invalid choice.")

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

ภาพที่ 4-2 โค้ดของฟังก์ชัน main_menu() 2

4.2.1.3 แสดงหัวข้อของโปรแกรม

เมื่อโปรแกรมเริ่มทำงาน ฟังก์ชัน main_menu() จะพิมพ์หัวข้อและเมนูหลักของโปรแกรม จากนั้นจะแสดงชื่อโปรแกรมและเมนูการจัดการข้อมูลหนังสือ โดยการพิมพ์ข้อความถูกพิมพ์ด้วยคำสั่ง print() ซึ่งประกอบด้วย

- ชื่อโปรแกรม “Library Management”
- เมนูการจัดการต่าง ๆ จะถูกแสดงหลังจากนี้

```
def main_menu():
    init_all_files()
    while True:
        print("\n===== Library Management =====")
```

ภาพที่ 4-3 แสดงหัวข้อของฟังก์ชัน main_menu()

4.2.1.4 แสดงเมนูให้ผู้ใช้เลือก

โปรแกรมจะเข้าสู่ลูป while True เพื่อแสดงเมนูซ้ำ ๆ และรับอินพุตจากผู้ใช้เรื่อย ๆ จนกว่าผู้ใช้จะเลือกเมนู “Exit” เพื่อออกจากโปรแกรม โดยลูป while True เป็นลูปที่ไม่มีเงื่อนไขสิ้นสุด ทำให้โปรแกรมทำงานต่อเนื่องโดยไม่หยุด และรับอินพุตจากผู้ใช้ ใช้คำสั่ง input() เพื่อให้ผู้ใช้เลือกหมายเลขเมนู จากนั้นแปลงข้อมูลที่ได้รับเป็นประเภทตัวเลขเป็นตำนวนเต็ม (int)

เมนูที่ผู้ใช้สามารถเลือกได้

1. Add Book: เพิ่มหนังสือใหม่
2. Update Book: แก้ไขข้อมูลหนังสือ
3. Delete Book: ลบหนังสือ
4. View Books: ดูหนังสือทั้งหมด
5. Add Member: เพิ่มผู้ใช้บริการใหม่
6. Update Member: แก้ไขข้อมูลผู้ใช้บริการ
7. Delete Member: ลบผู้ใช้บริการ
8. View Members: ดูผู้ใช้บริการทั้งหมด
9. Borrow Book: ยืมหนังสือ
10. Return Book: คืนหนังสือ
11. View Loan: ดูข้อมูลการคืนหนังสือ
12. Generate Report (.txt): สร้างไฟล์ Report (.txt)
0. Exit: ออกจากโปรแกรม

```
while True:
    print("\n===== Library Management =====")
    print("1. Add Book")
    print("2. Update Book")
    print("3. Delete Book")
    print("4. View Books")
    print("5. Add Member")
    print("6. Update Member")
    print("7. Delete Member")
    print("8. View Members")
    print("9. Borrow Book")
    print("10. Return Book")
    print("11. View Loans")
    print("12. Generate Report (.txt)")
    print("0. Exit")
```

ภาพที่ 4-4 แสดงเมนูให้ผู้ใช้เลือกของฟังก์ชัน main_menu()

4.2.1.5 ตรวจสอบการเลือกของผู้ใช้

เมื่อผู้ใช้เลือกหมายเลขจากเมนู ฟังก์ชันจะตรวจสอบหมายเลขที่ผู้ใช้เลือก และเรียกใช้ฟังก์ชันที่สอดคล้องกับเมนูนั้น ๆ โดยใช้โครงสร้างควบคุม if-elif-else ฟังก์ชันที่ถูกเรียกใช้:

1. Add Book: เรียกใช้ฟังก์ชันเพื่อเพิ่มหนังสือใหม่
2. Update Book: เรียกใช้ฟังก์ชันเพื่อแก้ไขข้อมูลหนังสือที่มีอยู่
3. Delete Book: เรียกใช้ฟังก์ชันเพื่อลบข้อมูลหนังสือ
4. View Books: เรียกใช้ฟังก์ชันเพื่อแสดงข้อมูลหนังสือทั้งหมดที่จัดเก็บในไฟล์
5. Add Member: เรียกใช้ฟังก์ชันเพื่อเพิ่มข้อมูลผู้ใช้บริการใหม่
6. Update Member: เรียกใช้ฟังก์ชันเพื่อแก้ไขข้อมูลผู้ใช้บริการที่มีอยู่
7. Delete Member: เรียกใช้ฟังก์ชันเพื่อลบข้อมูลผู้ใช้บริการ
8. View Members: เรียกใช้ฟังก์ชันเพื่อแสดงข้อมูลผู้ใช้บริการทั้งหมดที่จัดเก็บในไฟล์
9. Borrow Book: เรียกใช้ฟังก์ชันเพื่อยืมหนังสือ
10. Return Book: เรียกใช้ฟังก์ชันเพื่อคืนหนังสือ
11. View Loan: เรียกใช้ฟังก์ชันเพื่อแสดงข้อมูลการคืนหนังสือ
12. Generate Report (.txt): เรียกใช้ฟังก์ชันเพื่อสร้างไฟล์ Report (.txt)
0. Exit: เรียกใช้ฟังก์ชันเพื่อยืนยันการออกจากโปรแกรม

หากผู้ใช้ป้อนหมายเลขที่ไม่อยู่ในช่วง 1 ถึง 0 โปรแกรมจะพิมพ์ข้อความเตือนว่า “Invalid choice.” เพื่อแจ้งว่าหมายเลขที่เลือกไม่ถูกต้อง

```
choice = input("Choose option: ").strip()
if choice == "1":
    add_book()
elif choice == "2":
    update_book()
elif choice == "3":
    delete_book()
elif choice == "4":
    view_books()
elif choice == "5":
    add_member()
elif choice == "6":
    update_member()
elif choice == "7":
    delete_member()
elif choice == "8":
    view_members()
elif choice == "9":
    borrow_book()
elif choice == "10":
    return_book()
elif choice == "11":
    view_loans()
elif choice == "12":
    generate_report()
elif choice == "0":
    print("Exiting. Bye.")
    sys.exit(0)
else:
    print("Invalid choice.")
```

ภาพที่ 4-5 ตรวจสอบการเลือกของผู้ใช้ฟังก์ชัน main_menu()

4.2.1.6 การวนลูปกลับสู่เมนูหลัก

เนื่องจากลูป while True: จะทำงานต่อเนื่อง โปรแกรมจึงจะแสดงเมนูให้ผู้ใช้เลือกใหม่เรื่อย ๆ หลังจากการทำงานเสร็จในแต่ละฟังก์ชัน เช่น เมื่อผู้ใช้แสดงรายการหนังสือหรือเพิ่มหนังสือเสร็จแล้ว โปรแกรมจกลับมาที่เมนูหลักโดยอัตโนมัติ จนกว่าผู้ใช้จะเลือกเมนู “Exit (0)” เพื่อออกจากโปรแกรม

4.2.1.7 การออกจากโปรแกรม

เมื่อผู้ใช้เลือกเมนู 0 ฟังก์ชัน sys.exit(0) โปรแกรมจะหยุดทำงานในทันที

```
elif choice == "0":
    print("Exiting. Bye.")
    sys.exit(0)
```

ภาพที่ 4-6 การออกจากโปรแกรมฟังก์ชัน main_menu()

4.2.1.8 สรุปการทำงานของฟังก์ชัน main_menu()

ฟังก์ชัน main_menu() เป็นศูนย์กลางการทำงานของโปรแกรมทำหน้าที่แสดงเมนูและรับคำสั่งจากผู้ใช้มีการตรวจสอบอินพุตจากผู้ใช้และเรียกใช้ฟังก์ชันย่อยต่าง ๆ ตามเมนูที่เลือกและโปรแกรมจะวนลูปไปเรื่อย ๆ จนกว่าผู้ใช้จะเลือกออกจากโปรแกรม และยังมีการจัดการข้อผิดพลาดเพื่อให้โปรแกรมทำงานได้อย่างราบรื่น

4.2.1.9 ผลลัพธ์ของฟังก์ชัน main_menu()

```
===== Library Management =====
1. Add Book
2. Update Book
3. Delete Book
4. View Books
5. Add Member
6. Update Member
7. Delete Member
8. View Members
9. Borrow Book
10. Return Book
11. View Loans
12. Generate Report (.txt)
0. Exit
Choose option:
```

ภาพที่ 4-7 ผลลัพธ์ของฟังก์ชัน main_menu()

4.2.2 ฟังก์ชัน add_book()

ฟังก์ชัน add_book() ทำหน้าที่เพิ่มข้อมูลหนังสือใหม่เข้าสู่ระบบฐานข้อมูล โดยมีการบันทึกข้อมูลลงในไฟล์ books.dat ในรูปแบบไฟล์ไบนารี (Binary File) ข้อมูลที่ถูกบันทึกประกอบด้วย รหัสหนังสือ ชื่อหนังสือ ประเภท ผู้แต่ง สำนักพิมพ์ ปีที่พิมพ์ จำนวนเล่ม และสถานะการใช้งาน ทั้งนี้ฟังก์ชันยังรองรับการจัดการพื้นที่ว่างในไฟล์ (Free List) เพื่อให้การใช้พื้นที่เก็บข้อมูลมีประสิทธิภาพสูงสุด

4.2.2.1 โค้ดของฟังก์ชัน add_book()

```
def add_book():
    ensure_file(BOOKS_FILE, BOOK_STRUCT)
    with open(BOOKS_FILE, "r+b") as f:
        num, free_head = read_header(f)
        if free_head != -1:
            book_id = fmt_id("B", free_head + 1)
        else:
            book_id = fmt_id("B", num + 1)

        title = input("Enter Book Title: ").strip()
        category = input("Enter Book Category: ").strip()
        author = input("Enter Author Name: ").strip()
        publisher = input("Enter Publisher Name: ").strip()
        year = input("Enter Publish Year: ").strip()[:4]
        try:
            copies = int(input("Enter Number of Copies: ").strip())
        except ValueError:
            print("Invalid copies number. Cancel.")
            return
        packed = pack_book(book_id, title, category, author, publisher, year, copies, 1, -1)
        idx = append_or_reuse(f, packed, BOOK_STRUCT)
        print(f"Book added. ID = {book_id} (slot {idx})")
```

ภาพที่ 4-8 โค้ดของฟังก์ชัน add_book()

4.2.2.2 กระบวนการทำงานของฟังก์ชัน

4.2.2.2.1 ตรวจสอบไฟล์จัดเก็บข้อมูล

เรียกใช้ฟังก์ชันเพื่อตรวจสอบการมีอยู่ของไฟล์ books.dat หากไฟล์ยังไม่มี ระบบจะทำการสร้างไฟล์ใหม่และเขียนข้อมูลส่วนหัว (Header) เริ่มต้น

4.2.2.2.2 เปิดไฟล์และอ่านข้อมูลส่วนหัว (Header)

เปิดไฟล์ด้วยโหมด r+b (อ่านและเขียนแบบไบนารี) และเรียกใช้ฟังก์ชัน read_header(f) เพื่ออ่านจำนวนระเบียน (Record) ที่มีอยู่ (num) และตำแหน่งว่าง (free_head) เพื่อนำมาใช้ในการบันทึกข้อมูลใหม่

```
    with open(BOOKS_FILE, "r+b") as f:
        num, free_head = read_header(f)
```

ภาพที่ 4-9 การเปิดไฟล์และอ่านข้อมูลของฟังก์ชัน add_book()

4.2.2.2.3 สร้างรหัสหนังสืออัตโนมัติ

เรียกใช้ฟังก์ชันเพื่อสร้างรหัสประจำหนังสือ (Book_ID) ใหม่ โดยมีรูปแบบรหัส เช่น B001, B002 เป็นต้น

```
books = list_all_records(BOOKS_FILE, BOOK_STRUCT, unpack_book)
max_id = 0
for b in books:
    try:
        n = int(b["Book_ID"][1:])
        if n > max_id:
            max_id = n
    except:
        continue
return fmt_id("B", max_id + 1)
```

ภาพที่ 4-10 การสร้างรหัสหนังสืออัตโนมัติในฟังก์ชัน add_book()

4.2.2.2.4 รับข้อมูล

ระบบจะแสดงข้อความให้ผู้ใช้กรอกข้อมูล ได้แก่

- ชื่อหนังสือ (Title)
- ประเภท (Category)
- ผู้แต่ง (Author)
- สำนักพิมพ์ (Publisher)
- ปีที่พิมพ์ (Year) โดยจำกัดความยาว 4 หลัก
- จำนวนเล่ม (Copies) โดยตรวจสอบว่าต้องเป็นค่าตัวเลข หากไม่ถูกต้องระบบจะยกเลิกการทำงานทันที

```
title = input("Enter Book Title: ").strip()
category = input("Enter Book Category: ").strip()
author = input("Enter Author Name: ").strip()
publisher = input("Enter Publisher Name: ").strip()
year = input("Enter Publish Year: ").strip()[:4]
```

ภาพที่ 4-11 การรับข้อมูลหนังสือในฟังก์ชัน add_book()

4.2.2.2.5 บรรจุข้อมูลให้อยู่ในรูปแบบระเบียน (Record Packing)

เมื่อข้อมูลครบถ้วน ฟังก์ชัน pack_book() จะทำการบรรจุข้อมูลทั้งหมดให้อยู่ในรูปแบบไบนารีตามโครงสร้าง BOOK_STRUCT

4.2.2.2.6 บันทึกข้อมูลลงไฟล์

เรียกใช้ฟังก์ชัน append_or_reuse() เพื่อตัดสินใจบันทึกข้อมูลใหม่

- หากมีพื้นที่ว่างใน Free List : ระบบจะนำช่องว่างนั้นมาใช้ซ้ำ
- หากไม่มีพื้นที่ว่าง : ระบบจะบันทึกข้อมูลต่อท้ายไฟล์

4.2.2.2.7 ปรับปรุงข้อมูลส่วนหัว (Header Update)

หากเป็นการเพิ่มหนังสือใหม่ จะมีการปรับปรุงจำนวนระเบียน (num) ใน Header เพิ่มขึ้นหนึ่งระเบียน

4.2.2.2.8 แสดงผลลัพธ์การทำงาน

เมื่อเพิ่มหนังสือสำเร็จ ระบบจะแสดงข้อความยืนยัน พร้อมแจ้งรหัสหนังสือ (Book ID) และตำแหน่งที่ถูกจัดเก็บ (Slot)

```
Enter Book Title: Happy Life
Enter Book Category: Education
Enter Author Name: Anirach
Enter Publisher Name: KMUTNB
Enter Publish Year: 2011
Enter Number of Copies: 5
Book added. ID = B005 (slot 4)
```

ภาพที่ 4-12 ผลลัพธ์การทำงานของฟังก์ชัน add_book()

4.2.2.2.9 สรุปการทำงาน

ฟังก์ชัน add_book() เป็นฟังก์ชันสำคัญของระบบจัดการห้องสมุด เนื่องจากใช้สำหรับเพิ่มหนังสือใหม่เข้าสู่ฐานข้อมูล โดยมีการตรวจสอบความถูกต้องของข้อมูล ป้องกันข้อผิดพลาด และใช้โครงสร้าง Free List เพื่อจัดการพื้นที่จัดเก็บข้อมูลอย่างมีประสิทธิภาพ

4.2.3 ฟังก์ชัน view_books()

ทำหน้าที่แสดงข้อมูลหนังสือทั้งหมดที่ถูกบันทึกอยู่ในระบบ โดยจะแสดงในรูปแบบตาราง (Tabular Form) เพื่อให้ง่ายต่อการอ่านและตรวจสอบสถานะของหนังสือ

4.2.3.1 โค้ดของฟังก์ชัน view_book

```python
def view_books():
    recs = list_all_records(BOOKS_FILE, BOOK_STRUCT, unpack_book)
    print("\n--- Books ---")
    print("ID   | Title                          | Category      | Author         | Copies | Status")
    print("-"*100)
    for r in recs:
        status = "Active" if r["Book_status"] == 1 else "Deleted"
        print(f"{r['Book_ID']:<4} | {r['Book_Title'][:30]:<30} | {r['Book_Category'][:12]:<12} | {r['Author_Name'][:14]:<14} | {r['Book_copies']:<6} | {status}")
    print()
```

ภาพที่ 4-13 โค้ดฟังก์ชัน view_book

4.3.2.2 กระบวนการทำงานของฟังก์ชัน

4.2.3.2.1 ดึงข้อมูลหนังสือจากไฟล์

ฟังก์ชัน list_all_records(BOOKS_FILE, BOOK_STRUCT, unpack_book) ทำหน้าที่อ่านข้อมูลหนังสือทั้งหมดจากไฟล์ books.dat คืนค่ากลับมาเป็นรายการ (List) ของ Dictionary ที่เก็บ รายละเอียดหนังสือแต่ละเล่ม เช่น Book_ID, Book_Title, Book_Category, Author_Name, Book_copies, Book_status

4.2.3.2.2 แสดงหัวตาราง

ใช้คำสั่ง print() เพื่อกำหนดหัวตาราง ได้แก่ ID, Title, Category, Author, Copies, Status

4.2.3.2.3 แสดงข้อมูลแต่ละเรคอร์ด

- วนลูปข้อมูลหนังสือทีละรายการ
- ตรวจสอบค่าฟิลด์ Book_status
  - ถ้า 1 = Active
  - ถ้า 0 = Deleted
- จัดรูปแบบการแสดงผลให้อ่านง่าย (กำหนดความกว้างตัวอักษรของแต่ละคอลัมน์)

4.2.3.2.4 สรุปการทำงาน

ฟังก์ชัน view_books() ช่วยให้ผู้ใช้สามารถตรวจสอบหนังสือทั้งหมดที่บันทึกไว้ในระบบได้อย่างสะดวกและชัดเจน โดยแสดงรายละเอียดในรูปแบบตาราง พร้อมทั้งระบุสถานะของหนังสือว่าอยู่ในระบบ (Active) หรือถูกลบไปแล้ว (Deleted)

4.2.4 ฟังก์ชัน update_book()

ฟังก์ชัน update_book() ใช้สำหรับแก้ไขข้อมูลหนังสือที่มีอยู่แล้วในระบบ โดยผู้ใช้สามารถป้อนรหัสหนังสือ (Book_ID) เพื่อตรวจสอบและเลือกแก้ไขรายละเอียดที่ต้องการ เช่น ชื่อหนังสือ ผู้แต่ง สำนักพิมพ์ หรือจำนวนเล่ม

4.2.4.1 โค้ดของฟังก์ชัน update_book()

```python
def update_book():
    bid = input("Enter Book ID to update: ").strip()
    res = find_record_by_id(BOOKS_FILE, BOOK_STRUCT, unpack_book, "Book_ID", bid)
    if not res:
        print("Book not found.")
        return
    idx, rec = res
    if rec["Book_status"] != 1:
        print("Book is deleted/inactive.")
        return

    print("Leave blank to keep current.")

    BookID = input(f"Book ID [{rec['Book_ID']}]: ").strip() or rec['Book_ID']
    title = input(f"Title [{rec['Book_Title']}]: ").strip() or rec['Book_Title']
    category = input(f"Category [{rec['Book_Category']}]: ").strip() or rec['Book_Category']
    author = input(f"Author [{rec['Author_Name']}]: ").strip() or rec['Author_Name']
    publisher = input(f"Publisher [{rec['Publisher_Name']}]: ").strip() or rec['Publisher_Name']
    year = input(f"Year [{rec['Book_year']}]: ").strip() or rec['Book_year']
    copies_str = input(f"Copies [{rec['Book_copies']}]: ").strip()

    try:
        copies = int(copies_str) if copies_str else rec['Book_copies']
    except ValueError:
        print("Invalid copies. Aborted.")
        return

    packed = pack_book(rec['Book_ID'], title, category, author, publisher, year, copies, 1, rec['next_free'])
    with open(BOOKS_FILE, "r+b") as f:
        write_record_at(f, idx, packed, BOOK_STRUCT)
    print("Book updated.")
```

ภาพที่ 4-14 โค้ดของฟังก์ชัน update_book()

4.2.4.2 กระบวนการทำงาน

4.2.4.2.1 รับค่ารหัสหนังสือจากผู้ใช้

ผู้ใช้ป้อน Book_ID เพื่อระบุหนังสือที่ต้องการแก้ไข

4.2.4.2.2 ค้นหาหนังสือในไฟล์

- โปรแกรมอ่านข้อมูลทั้งหมดจาก books.dat
- ตรวจสอบว่ามีรหัสตรงกับที่ผู้ใช้ป้อนไหม

4.2.4.2.3 แก้ไขข้อมูล

ถ้าพบหนังสือ

- แสดงข้อมูลเก่า
- ให้ผู้ใช้ป้อนข้อมูลใหม่ (หากเว้นว่างไว้จะใช้ค่าเดิม)
- อัปเดตข้อมูลลงไฟล์

4.2.4.2.4 แจ้งผลลัพธ์

- ถ้าแก้ไขสำเร็จจะแจ้ง “Book updated successfully!”
- ถ้าไม่พบรหัสจะแจ้ง “Book not found!”

```
Choose option: 2
Enter Book ID to update: B001
Leave blank to keep current.
Book ID [B001]:
Title [Computer Programming]:
Category [Education]:
Author [Nitikan]:
Publisher [KMUTNB]:
Year [2012]:
Copies [9]:
Book updated.
```

ภาพที่ 4-15 ผลลัพธ์ของฟังก์ชัน update_book()

4.2.5 ฟังก์ชัน delete_book()

ฟังก์ชัน delete_book() ใช้สำหรับลบหนังสือออกจากระบบ โดยการเปลี่ยนสถานะหนังสือจาก “Active” เป็น “Deleted” แทนการลบข้อมูลจริง

4.2.5.1 โค้ดของฟังก์ชัน delete_book()

```python
def delete_book():
    bid = input("Enter Book ID to delete: ").strip()
    res = find_record_by_id(BOOKS_FILE, BOOK_STRUCT, unpack_book, "Book_ID", bid)
    if not res:
        print("Book not found.")
        return
    idx, rec = res
    if rec["Book_status"] == 0:
        print("Book already deleted.")
        return

    with open(BOOKS_FILE, "r+b") as f:
        num, free_head = read_header(f)
        packed = pack_book(rec['Book_ID'], rec['Book_Title'], rec['Book_Category'], rec['Author_Name'], rec['Publisher_Name'], rec['Book_year'], rec['Book_copies'], 0, free_head)

        write_record_at(f, idx, packed, BOOK_STRUCT)
        write_header(f, num, idx)
    print("Book deleted (slot freed).")
```

ภาพที่ 4-16 โค้ดของฟังก์ชัน delete_book

4.2.5.2 กระบวนการทำงาน

4.2.5.2.1 รับรหัสหนังสือจากผู้ใช้

โปรแกรมรับรหัสสมาชิกจากผู้ใช้ผ่านคำสั่ง input() และใช้ .strip() เพื่อตัดช่องว่างหัว–ท้ายออก เพื่อป้องกันความผิดพลาดในการค้นหาและให้รหัสที่ได้ตรงกับข้อมูลจริงในไฟล์

4.2.5.2.2 ค้นหาหนังสือในไฟล์

เมื่อได้รหัสหนังสือมาแล้ว โปรแกรมจะเปิดไฟล์ books.dat ในโหมดอ่าน (rb) จากนั้น วนลูปอ่านข้อมูลหนังสือทีละระเบียน (record) เพื่อตรวจสอบว่า Book ID ตรงกับที่ผู้ใช้ป้อนไว้หรือไม่ถ้าพบหนังสือที่มีรหัสตรงกัน

แสดงข้อมูลของหนังสือ เช่น ชื่อเรื่อง ผู้แต่ง จำนวนเล่ม ฯลฯ ถ้าไม่พบ แจ้งข้อความว่า "ไม่พบหนังสือ"

4.2.5.2.3 ถ้าพบ ⟶ ถามยืนยัน (y/n)

1. ถ้าเลือก y ⟶ เปลี่ยนสถานะเป็น Deleted
2. ถ้าเลือก n ⟶ ยกเลิก

4.2.5.2.4 แจ้งผลลัพธ์

```
Choose option: 3
Enter Book ID to delete: B005
Book deleted (slot freed).
```

ภาพที่ 4-17 ผลลัพธ์ update_book

4.2.6 ฟังก์ชัน add_member()

ฟังก์ชัน add_member() ใช้สำหรับเพิ่มข้อมูลสมาชิกใหม่เข้าสู่ระบบ เช่น รหัสสมาชิก ชื่อ–นามสกุล และสถานการณ์ใช้งาน

4.2.6.1 โค้ดของฟังก์ชัน add_member()

```python
def add_member():
    ensure_file(MEMBERS_FILE, MEM_STRUCT)
    with open(MEMBERS_FILE, "r+b") as f:
        num, free_head = read_header(f)

        member_id = get_next_member_id()  # ใช้ ID ใหม่

        name = input("Enter Member Name: ").strip()
        birth = input("Enter Birth Date (YYYY-MM-DD): ").strip()[:10]
        max_loan = DEFAULT_MAX_LOAN

        packed = pack_member(member_id, name, birth, max_loan, 1, -1)
        idx = append_or_reuse(f, packed, MEM_STRUCT)  # ใช้ slot ว่างถ้ามี

        if free_head == -1:
            write_header(f, num + 1, -1)  # อัพเดตจำนวนสมาชิก

        print(f"Member added. ID = {member_id} (slot {idx}). Max loan = {max_loan}")
```

ภาพที่ 4-18 โค้ดฟังก์ชันของ add_member

4.2.6.2 กระบวนการทำงาน

- รับค่าข้อมูลสมาชิกจากผู้ใช้

- จัดเก็บข้อมูลในรูปแบบ dictionary
- ใช้ฟังก์ชัน append_record() เพื่อบันทึกลงไฟล์ members.dat
- แจ้งผลลัพธ์ว่าเพิ่มสมาชิกสำเร็จ

```
packed = pack_member(member_id, name, birth, max_loan, 1, -1)
idx = append_or_reuse(f, packed, MEM_STRUCT)
```

ภาพที่ 4-19 ฟังก์ชันที่ใช้บันทึกลงไฟล์ members.dat

4.2.7 ฟังก์ชัน view_members()

ฟังก์ชัน view_members() ใช้สำหรับแสดงรายชื่อสมาชิกทั้งหมดที่ถูกบันทึกในระบบ

4.2.7.1 โค้ดของฟังก์ชัน view_members()

```
def view_members():
    recs = list_all_records(MEMBERS_FILE, MEM_STRUCT, unpack_member)
    print("\n--- Members ---")
    print("ID   | Name                          | Birth      | MaxLoan | Status")
    print("-"*80)
    for r in recs:
        status = "Active" if r["Member_status"] == 1 else "Deleted"
        print(f"{r['Member_ID']:<4} | {r['Member_Name'][:30]:<30} | {r['Member_Birth']:<10} | {r['Max_loan']:<7} | {status}")
    print()
```

ภาพที่ 4-20 โค้ดฟังก์ชัน view_members()

4.2.7.2 กระบวนการทำงาน

4.2.7.2.1 ดึงข้อมูลสมาชิกทั้งหมดจากไฟล์ members.dat

- ฟังก์ชันเรียกใช้เพื่ออ่านข้อมูลทุกเรคคอร์ดจากไฟล์ members.dat
- ข้อมูลที่อ่านออกมาจะถูกถอดรหัส (unpack) ด้วย unpack_member เพื่อให้ได้เป็นรูปแบบ dictionary ที่อ่านง่าย

```
recs = list_all_records(MEMBERS_FILE, MEM_STRUCT, unpack_member)
```

ภาพที่ 4-21 ฟังก์ชัน unpack_member ให้เป็น dictionnary

4.2.7.2.2 แสดงหัวตาราง (Header)

- ใช้คำสั่ง print() เพื่อพิมพ์หัวตารางให้อ่านง่าย
- หัวตารางประกอบด้วย ID | Name | Phone | Status
- มีเส้นคั่นด้วย "-"*60 เพื่อจัดรูปแบบให้อ่านง่ายเหมือนตาราง

```
print("\n--- Members ---")
print("ID   | Name                          | Birth      | MaxLoan | Status")
print("-"*80)
```

ภาพที่ 4-22 แสดง Header ของฟังก์ชัน view_member

4.2.7.2.3 วนลูปแสดงข้อมูลสมาชิกทีละรายการ

- ใช้คำสั่ง for r in recs: เพื่อวนซ้ำผ่านรายการสมาชิกทั้งหมด
- ข้อมูลแต่ละแถวจะแสดงเป็นรูปแบบตาราง โดยใช้ f-string และการจัดตำแหน่ง

4.2.7.2.4 แสดงสถานะสมาชิก (Active/Deleted)

- ตรวจสอบค่าของ Member_status
  - ถ้า 1 ⟶ แสดงเป็น "Active"
  - ถ้า 0 ⟶ แสดงเป็น "Deleted"
- ทำให้ผู้ใช้สามารถรู้ได้ว่าสมาชิกคนนั้นยังใช้งานได้อยู่หรือถูกลบออกจากระบบแล้ว

```
for r in recs:
    status = "Active" if r["Member_status"] == 1 else "Deleted"
    print(f"{r['Member_ID']:<4} | {r['Member_Name'][:30]:<30} | {r['Member_Birth']:<10} | {r['Max_loan']:<7} | {status}")
```

ภาพที่ 4-23 แสดงสถานะ Active Deleted ของฟังก์ชัน view_member

4.2.8 ฟังก์ชัน update_member()

ฟังก์ชัน update_book() มีหน้าที่ในการแก้ไขข้อมูลสมาชิก (Member) ภายในระบบ โดยใช้ รหัสสมาชิก (Member ID) เป็นตัวระบุหลัก ข้อมูลจะถูกอ่าน แก้ไข และบันทึกกลับไปยังไฟล์เก็บข้อมูลสมาชิก (members_file) ตามโครงสร้างที่กำหนดไว้ (mem_struct)

4.2.8.1 โค้ดของฟังก์ชัน update_member

```
def update_member():
    mid = input("Enter Member ID to update: ").strip()
    res = find_record_by_id(MEMBERS_FILE, MEM_STRUCT, unpack_member, "Member_ID", mid)
    if not res:
        print("Member not found.")
        return
    idx, rec = res
    if rec["Member_status"] != 1:
        print("Member deleted/inactive.")
        return

    print("Leave blank to keep current.")
    name = input(f"Name [{rec['Member_Name']}]: ").strip() or rec['Member_Name']
    birth = input(f"Birth [{rec['Member_Birth']}]: ").strip() or rec['Member_Birth']

    packed = pack_member(rec['Member_ID'], name, birth, rec['Max_loan'], 1, rec['next_free'])
    with open(MEMBERS_FILE, "r+b") as f:
        write_record_at(f, idx, packed, MEM_STRUCT)
    print("Member updated.")
```

ภาพที่ 4-24 โค้ดของฟังก์ชัน update_member

4.2.8.2 กระบวนการทำงาน

4.2.8.2.1 รับค่ารหัสสมาชิก (Member ID) ที่ต้องการแก้ไข

- ผู้ใช้กรอกรหัสสมาชิกผ่าน input()
- ค่าที่รับมาจะถูกนำไปค้นหาภายในไฟล์ด้วยฟังก์ชัน fine_record_by_id()

```
name = input(f"Name [{rec['Member_Name']}]: ").strip() or rec['Member_Name']
birth = input(f"Birth [{rec['Member_Birth']}]: ").strip() or rec['Member_Birth']
```

ภาพที่ 4-25 รับข้อมูลผ่าน Input ในฟังก์ชัน update_member

```
res = find_record_by_id(MEMBERS_FILE, MEM_STRUCT, unpack_member, "Member_ID", mid)
```

ภาพที่ 4-26 ค้นหาด้วยฟังก์ชัน fine_record_by_id()

4.2.8.2.2 ตรวจสอบการมีอยู่ของสมาชิก

- ถ้าไม่พบข้อมูลสมาชิกในไฟล์ จะพิมพ์ข้อความ "Member not found." และหยุดการทำงาน
- ถ้าพบข้อมูล แต่สถานะ (Member_status) ไม่เท่ากับ 1 (ซึ่งหมายถึงถูกลบหรือไม่ใช้งาน) จะพิมพ์ข้อความ "Member deleted/inactive." และหยุดการทำงาน

```
if not res:
    print("Member not found.")
    return
idx, rec = res
if rec["Member_status"] != 1:
    print("Member deleted/inactive.")
    return
```

ภาพที่ 4-27 ตรวจสอบสมาชิก

4.2.8.2.3 ให้ผู้ใช้เลือกแก้ไขเฉพาะบางข้อมูลได้

- ระบบจะแสดงข้อมูลเดิม เช่น ชื่อ (Member_Name) และวันเกิด (Member_Birth)

- ผู้ใช้สามารถกรอกข้อมูลใหม่ หรือเว้นว่างเพื่อคงค่าเดิมไว้

4.2.8.2.4 บันทึกข้อมูลใหม่กลับไปยังไฟล์

- ข้อมูลใหม่จะถูกจัดรูปแบบให้อยู่ในรูปแบบ Binary Record โดยใช้ฟังก์ชัน pack_member()

- จากนั้นเขียนทับกลับไปยังไฟล์ที่ตำแหน่งเดิม ด้วย write_record_at()

```
packed = pack_member(rec['Member_ID'], name, birth, rec['Max_loan'], 1, rec['next_free'])
with open(MEMBERS_FILE, "r+b") as f:
    write_record_at(f, idx, packed, MEM_STRUCT)
print("Member updated.")
```

ภาพที่ 4-28 กรอกข้อมูลใหม่และเขียนทับลงในไฟล์เดิม

4.2.8.2.5 ยันยืนผลการอัปเดต

- เมื่อแก้ไขเสร็จสมบูรณ์ จะแสดงข้อความ "Member updated." เพื่อยืนยันการทำงาน

4.2.9 ฟังก์ชัน delete_member()

ฟังก์ชัน delete_member() ทำหน้าที่ลบข้อมูลสมาชิก (Member) ออกจากระบบ โดยอาศัย รหัสสมาชิก (Member ID) เป็นตัวระบุหลัก ข้อมูลการลบจะไม่ถูกลบออกจากไฟล์จริงทั้งหมด แต่จะมีการปรับสถานะและเชื่อมโยงพื้นที่ว่างเข้ากับรายการช่องว่าง (Free List) เพื่อให้สามารถนำพื้นที่นั้นกลับมาใช้เก็บข้อมูลสมาชิกใหม่ได้ในอนาคต

4.2.9.1 โค้ดของฟังก์ชัน

```python
def delete_member():
    mid = input("Enter Member ID to delete: ").strip()
    res = find_record_by_id(MEMBERS_FILE, MEM_STRUCT, unpack_member, "Member_ID", mid)
    if not res:
        print("Member not found.")
        return
    idx, rec = res
    if rec["Member_status"] == 0:
        print("Member already deleted.")
        return

    with open(MEMBERS_FILE, "r+b") as f:
        num, free_head = read_header(f)
        packed = pack_member(rec['Member_ID'], rec['Member_Name'], rec['Member_Birth'], rec['Max_loan'], 0, free_head)
        write_record_at(f, idx, packed, MEM_STRUCT)
        write_header(f, num, idx)
    print("Member deleted (slot freed).")
```

ภาพที่ 4-29 โค้ดของฟังก์ชัน deleted_member

4.2.9.2 กระบวนการทำงาน

4.2.9.2.1 รับค่ารหัสสมาชิก (Member ID) ที่ต้องการลบ

- ผู้ใช้กรอกรหัสสมาชิกผ่าน input()
- ค่าที่ได้จะถูกใช้เป็นตัวค้นหาภายในไฟล์ข้อมูลสมาชิก (MEMBERS_FILE)

```python
mid = input("Enter Member ID to delete: ").strip()
res = find_record_by_id(MEMBERS_FILE, MEM_STRUCT, unpack_member, "Member_ID", mid)
```

ภาพที่ 4-30 รับค่ารหัสสมาชิก

4.2.9.2.2 ค้นหาข้อมูลสมาชิก

- ใช้ฟังก์ชัน find_record_by_id() เพื่อค้นหาสมาชิกจากไฟล์ โดยใช้ Member_ID เป็นตัวระบุ
- ถ้าไม่พบข้อมูล จะพิมพ์ข้อความ "Member not found." และหยุดการทำงาน

4.2.9.2.3 ตรวจสอบสถานะของสมาชิก

- ถ้าข้อมูลสมาชิกนั้นมีค่า Member_status เท่ากับ 0 (หมายถึงถูกลบไปแล้ว) ระบบจะแจ้ง "Member already deleted." และหยุดการทำงาน

```
if not res:
    print("Member not found.")
    return
idx, rec = res
if rec["Member_status"] == 0:
    print("Member already deleted.")
    return
```

ภาพที่ 4-31 ตรวจสอบสถานะของสมาชิก

4.2.9.2.4 ลบสมาชิกและปรับโครงสร้างไฟล์

- เปิดไฟล์ในโหมดอ่าน/เขียนแบบไบนารี (r+b)

- อ่านข้อมูลส่วนหัว (Header) ของไฟล์ด้วย read_header() ซึ่งจะได้จำนวนสมาชิกทั้งหมด และตำแหน่งหัวของ Free List

- เขียนข้อมูลใหม่กลับไปที่ตำแหน่งเดิมของระเบียนสมาชิก ด้วย write_record_at()

- ปรับปรุงส่วนหัวของไฟล์ (write_header()) โดยกำหนดให้ตำแหน่ง Free List ชี้มายังระเบียนที่เพิ่งถูกลบ (idx)

```
with open(MEMBERS_FILE, "r+b") as f:
    num, free_head = read_header(f)
    packed = pack_member(rec['Member_ID'], rec['Member_Name'], rec['Member_Birth'], rec['Max_loan'], 0, free_head)
    write_record_at(f, idx, packed, MEM_STRUCT)
    write_header(f, num, idx)
```

ภาพที่ 4-32 ลบสมาชิกและปรับโครงสร้างไฟล์

4.2.9.2.5 ยืนยันการลบ

- เมื่อการลบเสร็จสิ้น ระบบจะแสดงข้อความ "Member deleted (slot freed)."

4.2.10 ฟังก์ชัน borrow_book()

ฟังก์ชัน borrow_book() มีหน้าที่ในการจัดการกระบวนการ ยืมหนังสือ (Book Borrowing Process) ของระบบห้องสมุด โดยมีการตรวจสอบสิทธิ์ของสมาชิก ความพร้อมใช้งานของหนังสือ และการบันทึกข้อมูลการยืมลงไฟล์ข้อมูลการยืม (LOANS_FILE) ตามโครงสร้างข้อมูลที่กำหนดไว้

4.2.10.1 โค้ดของฟังก์ชัน borrow_book()

```python
def borrow_book():
    member_id = input("Enter Member ID: ").strip()
    res_m = find_record_by_id(MEMBERS_FILE, MEM_STRUCT, unpack_member, "Member_ID", member_id)
    if not res_m:
        print("Member not found or inactive.")
        return
    m_idx, member = res_m
    if member["Member_status"] != 1:
        print("Member inactive.")
        return

    loans = list_all_records(LOANS_FILE, LOAN_STRUCT, unpack_loan)
    current_borrowed = sum(1 for l in loans if l["Member_ID"] == member_id and l["Loan_Status"] == 1)

    book_ids = input("Enter Book IDs to borrow (comma separated): ").strip().split(",")

    for book_id in [b.strip() for b in book_ids if b.strip()]:
        if current_borrowed >= member["Max_loan"]:
            print(f"Member reached max loan limit ({member['Max_loan']}). Cannot borrow {book_id}.")
            continue
```

ภาพที่ 4-33 โค้ดฟังก์ชัน borrow_book()

```python
res_b = find_record_by_id(BOOKS_FILE, BOOK_STRUCT, unpack_book, "Book_ID", book_id)
if not res_b:
    print(f"Book {book_id} not found.")
    continue
b_idx, book = res_b
if book["Book_status"] != 1:
    print(f"Book {book_id} inactive.")
    continue
if book["Book_copies"] <= 0:
    print(f"No available copies for {book_id}.")
    continue

book["Book_copies"] -= 1
with open(BOOKS_FILE, "r+b") as bf:
    packed_book = pack_book(book["Book_ID"], book["Book_Title"], book["Book_Category"],
                            book["Author_Name"], book["Publisher_Name"], book["Book_year"],
                            book["Book_copies"], 1, book["next_free"])
    write_record_at(bf, b_idx, packed_book, BOOK_STRUCT)

ensure_file(LOANS_FILE, LOAN_STRUCT)
with open(LOANS_FILE, "r+b") as lf:
    loan_id = next_loan_id()
    loan_date = datetime.now().strftime("%Y-%m-%d")
    due_date = (datetime.now() + timedelta(days=7)).strftime("%Y-%m-%d")
    packed_loan = pack_loan(loan_id, 1, member_id, book_id, loan_date, due_date, "-", 1, -1)
    loan_idx = append_or_reuse(lf, packed_loan, LOAN_STRUCT)
    print(f"Borrow successful. Loan ID = {loan_id} (slot {loan_idx}). Due date: {due_date}")

current_borrowed += 1
```

ภาพที่ 4-34 โค้ดฟังก์ชัน borrow_book(2)

4.2.10.2 กระบวนการทำงาน

4.2.10.2.1 รับรหัสสมาชิก (Member ID)

- ระบบให้ผู้ใช้กรอก Member ID ผ่าน input()

- จากนั้นใช้ฟังก์ชัน find_record_by_id() เพื่อค้นหาสมาชิก0 จากไฟล์เก็บข้อมูลสมาชิก (MEMBERS_FILE)

4.2.10.2.2 ตรวจสอบข้อมูลสมาชิก

- ถ้าไม่พบข้อมูล จะแสดง "Member not found or inactive." และหยุดการทำงาน

- ถ้าพบข้อมูล แต่สถานะ (Member_status) ไม่เท่ากับ 1 (หมายถึงสมาชิกถูกลบหรือไม่ใช้งาน) จะแสดง "Member inactive." และหยุดการทำงาน

```
if not res_m:
    print("Member not found or inactive.")
    return
m_idx, member = res_m
if member["Member_status"] != 1:
    print("Member inactive.")
    return
```

ภาพที่ 4-35 ตรวจสอบข้อมูลสมาชิก

4.2.10.2.3 ตรวจสอบจำนวนการยืมปัจจุบัน

- อ่านข้อมูลการยืมทั้งหมดจากไฟล์ (LOANS_FILE) ด้วย list_all_records()

- คำนวณจำนวนเล่มที่สมาชิกคนนั้นกำลังยืมอยู่ (โดยมี Loan_Status = 1)

- เก็บผลลัพธ์ในตัวแปร current_borrowed

```
loans = list_all_records(LOANS_FILE, LOAN_STRUCT, unpack_loan)
current_borrowed = sum(1 for l in loans if l["Member_ID"] == member_id and l["Loan_Status"] == 1)
```

ภาพที่ 4-36 ตรวจสอบการยืมปัจจุบัน

4.2.10.2.4 รับรหัสหนังสือที่ต้องการยืม

- ระบบให้ผู้ใช้ป้อน Book IDs แบบคั่นด้วยเครื่องหมายจุลภาค

- ทำการแยกรหัสแต่ละเล่ม และนำไปตรวจสอบทีละรายการ

4.2.10.2.5 ตรวจสอบสิทธิ์การยืมและสถานะหนังสือ (ทีละเล่ม)

สำหรับหนังสือแต่ละเล่มที่สมาชิกต้องการยืม :

- ตรวจสอบจำนวนยืมสูงสุด (Max Loan Limit) ถ้าสมาชิกยืมครบจำนวนที่กำหนด (Max_loan) ระบบจะแจ้งว่าไม่สามารถยืมเพิ่มได้
- ตรวจสอบหนังสือ
- ถ้าไม่พบรหัสหนังสือ แจ้ง "Book not found."
- ถ้าหนังสือถูกปิดใช้งาน (Book_status != 1) แจ้ง "Book inactive."
- ถ้าหนังสือไม่มีเล่มคงเหลือ (Book_copies <= 0) แจ้ง "No available copies."

```
if not res_b:
    print(f"Book {book_id} not found.")
    continue
b_idx, book = res_b
if book["Book_status"] != 1:
    print(f"Book {book_id} inactive.")
    continue
if book["Book_copies"] <= 0:
    print(f"No available copies for {book_id}.")
    continue
```

ภาพที่ 4-37 ตรวจสอบสิทธิ์การยืมและสถานะหนังสือ

4.2.10.2.6 ปรับปรุงจำนวนที่ยืมปัจจุบัน

เพิ่มค่า current_borrowed ขึ้น 1 เพื่อสะท้อนสถานะล่าสุด

4.2.11 ฟังก์ชัน return_book()

ฟังก์ชัน return_book() มีหน้าที่จัดการกระบวนการ คืนหนังสือ (Book Return Process) ในระบบ โดยตรวจสอบการยืมจากไฟล์ข้อมูลการยืม (LOANS_FILE) และปรับปรุงสถานะการยืม รวมถึงจำนวนหนังสือคงเหลือในไฟล์เก็บข้อมูลหนังสือ (BOOKS_FILE) ให้ถูกต้องตามการคืน

4.2.11.1 โค้ดของฟังก์ชัน return_book()

```
def return_book():
    loan_ids = input("Enter Loan IDs to return (comma separated): ").strip().split(",")
    loan_ids = [lid.strip() for lid in loan_ids if lid.strip()]

    for loan_id in loan_ids:
        res = find_record_by_id(LOANS_FILE, LOAN_STRUCT, unpack_loan, "Loan_ID", loan_id)
        if not res:
            print(f"Loan {loan_id} not found.")
            continue

        l_idx, loan = res
        if loan["Loan_Status"] == 0:
            print(f"Loan {loan_id} already returned.")
            continue

        return_date = datetime.now().strftime("%Y-%m-%d")
        packed_return = pack_loan(
            loan["Loan_ID"],
            2,  # Operation_type = 2 = return
            loan["Member_ID"],
            loan["Book_ID"],
            loan["Loan_Date"],
            loan["Due_Date"],
            return_date,
            0,  # Loan_Status = 0 = returned
            -1
        )
```

ภาพที่ 4-38 โค้ดฟังก์ชัน return_book()

```
with open(LOANS_FILE, "r+b") as lf:
    write_record_at(lf, l_idx, packed_return, LOAN_STRUCT)
res_b = find_record_by_id(BOOKS_FILE, BOOK_STRUCT, unpack_book, "Book_ID", loan["Book_ID"])
if res_b:
    b_idx, book = res_b
    book["Book_copies"] += 1
    with open(BOOKS_FILE, "r+b") as bf:
        packed_book = pack_book(
            book["Book_ID"], book["Book_Title"], book["Book_Category"],
            book["Author_Name"], book["Publisher_Name"], book["Book_year"],
            book["Book_copies"], 1, book["next_free"]
        )
        write_record_at(bf, b_idx, packed_book, BOOK_STRUCT)

print(f"Book {loan['Book_ID']} returned successfully. Loan {loan_id} closed.")
```

ภาพที่ 4-39 โค้ดฟังก์ชัน return_book()2

4.2.11.2 กระบวนการทำงาน

4.2.11.2.1 รับรหัสการยืม (Loan IDs)

- ระบบให้ผู้ใช้กรอก Loan IDs ที่ต้องการคืน โดยคั่นด้วยเครื่องหมายจุลภาค (,)

- ทำการแยกค่าที่กรอก และลบช่องว่างส่วนเกินเพื่อนำไปตรวจสอบทีละรายการ

4.2.11.2.2 ตรวจสอบการมีอยู่ของรายการยืม

สำหรับแต่ละ Loan ID ที่ผู้ใช้ป้อน:

- ใช้ฟังก์ชัน find_record_by_id() ค้นหาในไฟล์ LOANS_FILE

- ถ้าไม่พบ จะแสดงข้อความ เช่น "Loan L001 not found." และข้ามไปทำรายการถัดไป

```
for loan_id in loan_ids:
    res = find_record_by_id(LOANS_FILE, LOAN_STRUCT, unpack_loan, "Loan_ID", loan_id)
    if not res:
        print(f"Loan {loan_id} not found.")
        continue
```

ภาพที่ 4-40 ตรวจสอบการมีอยู่ของรายการยืม

4.2.11.2.3 ตรวจสอบสถานการณ์ยืม

- ถ้ารายการยืม (Loan_Status) เท่ากับ 0 แสดงว่าถูกคืนไปแล้ว

- ระบบจะแจ้งข้อความ เช่น "Loan L001 already returned." และข้ามไปทำรายการถัดไป

```
l_idx, loan = res
if loan["Loan_Status"] == 0:
    print(f"Loan {loan_id} already returned.")
    continue
```

ภาพที่ 4-41 ตรวจสอบสถานการณ์ยืม

4.2.11.2.4 อัปเดตสถานะการคืน (Loan Record Update)

- บันทึกวันคืน (Return_Date) เป็นวันปัจจุบัน

- จัดเก็บข้อมูลใหม่ด้วย pack_loan() โดยมีรายละเอียดดังนี้:

- Loan_ID : รหัสการยืมเดิม

- Operation_type = 2 : ระบุว่าเป็นการคืน (return)

- Loan_Status = 0 : แสดงว่าสถานะถูกปิด (คืนแล้ว)

- ข้อมูลอื่น ๆ เช่น Member_ID, Book_ID, Loan_Date, Due_Date คงค่าเดิม

- เขียนระเบียนใหม่กลับไปยังไฟล์ LOANS_FILE ด้วย write_record_at()

```
return_date = datetime.now().strftime("%Y-%m-%d")
packed_return = pack_loan(
    loan["Loan_ID"],
    2,  # Operation_type = 2 = return
    loan["Member_ID"],
    loan["Book_ID"],
    loan["Loan_Date"],
    loan["Due_Date"],
    return_date,
    0,  # Loan_Status = 0 = returned
    -1
)
```

ภาพที่ 4-42 อัปเดตสถานการณ์ยืม - คืน

4.2.11.2.5 ปรับปรุงข้อมูลหนังสือ

- ค้นหาหนังสือที่ถูกคืนจากไฟล์ BOOKS_FILE ด้วย Book_ID

- ถ้าพบระเบียน จะเพิ่มจำนวนเล่มคงเหลือ (Book_copies += 1)

- ใช้ pack_book() เพื่อแพ็กข้อมูลใหม่ และบันทึกกลับไปยังไฟล์ด้วย write_record_at()

4.2.11.2.6 ยืนยันผลการคืน

- เมื่อการอัปเดตเสร็จสิ้น ระบบจะแสดงข้อความ เช่น "Book B001 returned successfully. Loan L001 closed."

4.2.12 ฟังก์ชัน view_loans()

ฟังก์ชัน view_loans() มีหน้าที่ แสดงรายการยืม-คืนหนังสือทั้งหมดในระบบ (Loan Records) โดยดึงข้อมูลจากไฟล์เก็บข้อมูลการยืม (LOANS_FILE) และนำมาจัดรูปแบบให้อ่านง่ายในลักษณะตาราง (Tabular Format)

4.2.12.1 โค้ดของฟังก์ชัน view_loans()

```
def view_loans():
    recs = list_all_records(LOANS_FILE, LOAN_STRUCT, unpack_loan)
    print("\n--- Loans ---")
    print("LoanID | MemID | BookID | LoanDate   | DueDate    | ReturnDate | Status")
    print("-"*90)
    for r in recs:
        status = "Borrowed" if r["Loan_Status"] == 1 else "Returned"
        print(f"{r['Loan_ID']:<6} | {r['Member_ID']:<5} | {r['Book_ID']:<5} | {r['Loan_Date']:<10} | {r['Due_Date']:<10} | {r['Return_Date']:<10} | {status}")
    print()
```

ภาพที่ 4-43 โค้ดฟังก์ชัน view_loans()

4.2.12.2 กระบวนการทำงาน

4.2.12.2.1 ดึงข้อมูลรายการยืมทั้งหมด

- ใช้ฟังก์ชัน list_all_records() เพื่ออ่านข้อมูลทั้งหมดจากไฟล์ LOANS_FILE
- แต่ละระเบียนจะถูกถอดรหัสด้วย unpack_loan และถูกเก็บไว้ใน recs

4.2.12.2.2 แสดงหัวตาราง (Table Header)

- แสดงข้อความหัวเรื่อง --- Loans ---
- แสดงหัวคอลัมน์ดังนี้:
  - LoanID : รหัสการยืม
  - MemID : รหัสสมาชิก
  - BookId : รหัสหนังสือ
  - LoanDate : วันที่ยืม
  - DueDate : วันที่กำหนดคืน
  - ReturnDate : วันที่คืนจริง
  - Status : สถานะ (Borrowed/Returned)
- วาดเส้นคั่นด้วย "-"*90 เพื่อจัดรูปแบบให้อ่านง่าย

4.2.12.2.3 วนลูปแสดงข้อมูลแต่ละรายการ (Record Display)

- สำหรับแต่ละระเบียนการยืม (r) ใน recs:
  - ตรวจสอบค่า Loan_Status:
    - ถ้า Loan_Status = 1 ⟶ แสดง "Borrowed"
    - ถ้าไม่ใช่ (เช่น 0) ⟶ แสดง "Returned"
  - แสดงผลในรูปแบบตาราง โดยใช้การจัดตำแหน่ง (Alignment)

4.2.12.2.4 ปิดท้ายการแสดงผล

- แทรกบรรทัดว่าง (print()) เพื่อความสวยงามและการอ่านที่ชัดเจน

4.2.13 ฟังก์ชัน generate_report()

ฟังก์ชัน generate_report() มีหน้าที่ในการ สร้างรายงานสรุป (Summary Report) ของระบบจัดการการยืม-คืนหนังสือ โดยจะดึงข้อมูลจากไฟล์เก็บข้อมูลทั้งหมด ได้แก่

- BOOKS_FILE : ข้อมูลหนังสือ
- MEMBERS_FILE : ข้อมูลสมาชิก
- LOANS_FILE : ข้อมูลการยืม-คืน

และทำการประมวลผลเพื่อสร้างไฟล์รายงาน (REPORT_FILE) ในรูปแบบข้อความ (Text Report)

4.2.13.1 โค้ดของฟังก์ชัน generate_report()

```
def generate_report():
    ensure_file(BOOKS_FILE, BOOK_STRUCT)
    ensure_file(MEMBERS_FILE, MEM_STRUCT)
    ensure_file(LOANS_FILE, LOAN_STRUCT)

    books = list_all_records(BOOKS_FILE, BOOK_STRUCT, unpack_book)
    members = list_all_records(MEMBERS_FILE, MEM_STRUCT, unpack_member)
    loans = list_all_records(LOANS_FILE, LOAN_STRUCT, unpack_loan)

    active_books = [b for b in books if b["Book_status"] == 1]
    deleted_books = [b for b in books if b["Book_status"] == 0]
    borrowed_now = sum(1 for l in loans if l["Loan_Status"] == 1)
    available_now = sum(b["Book_copies"] for b in active_books)

    grouped = {}
    for l in loans:
        key = (l["Member_ID"], l["Loan_Date"], l["Due_Date"], l["Return_Date"], l["Loan_Status"])
        if key not in grouped:
            grouped[key] = []
        grouped[key].append(l["Book_ID"])

    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    lines = []
    lines.append("Library Borrow System - Summary Report")
    lines.append(f"Generated At : {now} (+07:00)")
    lines.append("App Version  : 1.0")
    lines.append("Encoding     : UTF-8")
    lines.append("")
```

ภาพที่ 4-44 โค้ดฟังก์ชัน generate_report()

```
headers = ["MemberID", "MemberName", "BookID", "Titles", "LoanDate", "DueDate", "ReturnDate", "Status"]
rows = []
for (mid, loan_date, due_date, return_date, status), book_ids in grouped.items():
    member_name = next((m["Member_Name"] for m in members if m["Member_ID"] == mid), "-")
    titles = [next((b["Book_Title"] for b in books if b["Book_ID"] == bid), "-") for bid in book_ids]
    status_str = "Borrowed" if status == 1 else "Returned"
    rows.append([
        mid,
        member_name,
        ",".join(book_ids),
        ",".join(titles),
        loan_date,
        due_date,
        return_date,
        status_str
    ])

all_rows = [headers] + rows
col_widths = [max(len(str(row[i])) for row in all_rows) for i in range(len(headers))]

header_line = " | ".join(f"{headers[i]:<{col_widths[i]}}" for i in range(len(headers)))
sep_line = "-+-".join("-" * col_widths[i] for i in range(len(headers)))
lines.append(header_line)
lines.append(sep_line)

for row in rows:
    line = " | ".join(f"{str(row[i]):<{col_widths[i]}}" for i in range(len(row)))
    lines.append(line)
```

ภาพที่ 4-45 โค้ดฟังก์ชัน generate_report()2

```
lines.append("")
lines.append("Summary (Active Books Only)")
lines.append(f"- Total Books : {len(books)}")
lines.append(f"- Active Books : {len(active_books)}")
lines.append(f"- Deleted Books : {len(deleted_books)}")
lines.append(f"- Borrowed Now : {borrowed_now}")
lines.append(f"- Available Now : {available_now}")
lines.append("")
lines.append("Borrow Statistics (Active only)")

# borrow counts
borrow_count = {}
for l in loans:
    bid = l["Book_ID"]
    borrow_count[bid] = borrow_count.get(bid, 0) + (1 if l["Operation_type"] == 1 else 0)

most_borrowed_book = "-"
most_borrowed_count = 0
if borrow_count:
    top_bid = max(borrow_count, key=borrow_count.get)
    most_borrowed_count = borrow_count[top_bid]
    title = next((b["Book_Title"] for b in books if b["Book_ID"] == top_bid), "-")
    most_borrowed_book = f"{title} ({top_bid})"

lines.append(f"- Most Borrowed Book : {most_borrowed_book} ({most_borrowed_count} times)")
lines.append(f"- Currently Borrowed : {borrowed_now}")
lines.append(f"- Active Members : {len([m for m in members if m['Member_status']==1])}")

with open(REPORT_FILE, "w", encoding="utf-8") as rf:
    rf.write("\n".join(lines))
    rf.flush()
    os.fsync(rf.fileno())
print(f"Report generated: {REPORT_FILE}")
```

ภาพที่ 4-46 โค้ดฟังก์ชัน generate_report()3

4.2.13.2 กระบวนการทำงาน

4.2.13.2.1 ตรวจสอบและสร้างไฟล์ข้อมูลที่จำเป็น

- ใช้ ensure_file() เพื่อให้แน่ใจว่าไฟล์ BOOKS_FILE, MEMBERS_FILE, และ LOANS_FILE ถูกสร้างและพร้อมใช้งาน

4.2.13.2.2 โหลดข้อมูลจากไฟล์

- ใช้ list_all_records() เพื่ออ่านข้อมูลหนังสือ (books), สมาชิก (members), และการยืม (loans)

4.2.13.2.3 วิเคราะห์ข้อมูลเบื้องต้น

- แยกหนังสือเป็น 2 กลุ่ม:
    - active_books = หนังสือที่สถานะใช้งานอยู่ (Book_status = 1)
    - deleted_books = หนังสือที่ถูกลบออกแล้ว (Book_status = 0)
- คำนวณจำนวนการยืมที่ยังไม่คืน (borrowed_now)
- คำนวณจำนวนสำเนาหนังสือที่ยังมีอยู่ทั้งหมด (available_now)

4.2.13.2.4 จัดกลุ่มข้อมูลการยืม

- จัดกลุ่มตาม (Member_ID, Loan_Date, Due_Date, Return_Date, Loan_Status)
- เพื่อรวมหนังสือที่ยืมหลายเล่มในการทำธุรกรรมเดียว

4.2.13.2.5 สร้างรายงาน

- ใส่ข้อมูลเบื้องต้น เช่น วันเวลาที่สร้างรายงาน, เวอร์ชันของแอปพลิเคชัน, การเข้ารหัส (UTF-8)
- แสดงตารางสรุปการยืม-คืน โดยประกอบด้วยคอลัมน์:
    - MemberID, MemberName, BookID, Titles, LoanDate, DueDate, ReturnDate, Status

- จัดความกว้างของคอลัมน์อัตโนมัติเพื่อให้อ่านง่าย

4.2.13.2.6 สรุปสถิติ (Summary Section)

- แสดงจำนวนหนังสือทั้งหมด, หนังสือที่ใช้งาน, หนังสือที่ลบ, หนังสือที่ถูกยืม, และหนังสือที่มีสำเนาเหลือ

4.2.13.2.7 สถิติการยืม (Borrow Statistics)

- คำนวณจำนวนครั้งที่หนังสือแต่ละเล่มถูกยืม

- หาหนังสือที่ถูกยืมมากที่สุด (Most Borrowed Book)

- แสดงจำนวนการยืมที่ยังไม่คืน (Currently Borrwed)

- แสดงจำนวนสมาชิกที่ยังใช้งานอยู่ (Active Members)

4.2.13.2.8 บันทึกผลลัพธ์ลงไฟล์

- เขียนข้อมูลทั้งหมดลงใน REPORT_FILE โดยใช้การเข้ารหัส UTF-8

- ทำการ flush และ os.fsync() เพื่อให้แน่ใจว่าข้อมูลถูกบันทึกลงดิสก์จริง

4.2.13.2.9 แจ้งผลการทำงาน

- แสดงข้อความในคอนโซลว่า "Report generated: report.txt"

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

ระบบสำหรับการจัดการการยืมและคืนหนังสือของห้องสมุดที่ได้พัฒนาขึ้น ช่วยให้การบริหารข้อมูลหนังสือ ข้อมูลสมาชิก และข้อมูลการยืม-คืน มีประสิทธิภาพยิ่งขึ้น โดยระบบมีการจัดเก็บข้อมูลในรูปแบบไฟล์ไบนารี พร้อมด้วยเมนูในการเพิ่ม แก้ไข ลบ และแสดงผลข้อมูลต่าง ๆ ได้อย่างครบถ้วน นอกจากนี้ ยังสามารถตรวจสอบสถานะของหนังสือได้ว่าอยู่ในระหว่างการยืมหรือยังว่างอยู่ รวมถึงสามารถควบคุมจำนวนเล่มที่ถูกยืมออกไป และจัดทำรายงานสรุปผลการใช้งาน เช่น หนังสือที่ถูกยืมบ่อยที่สุด รายชื่อผู้ที่กำลังยืมหนังสืออยู่ และสถิติการใช้งานโดยรวม ซึ่งช่วยให้การบริหารจัดการในห้องสมุดเป็นไปอย่างสะดวก รวดเร็ว และลดความผิดพลาดจากการบันทึกแบบเอกสารกระดาษแบบเดิม

5.2 ปัญหาและอุปสรรคในการดำเนินงาน

ตลอดกระบวนการพัฒนาระบบ พบอุปสรรคสำคัญคือ ความยุ่งยากในการจัดการไฟล์ข้อมูลแบบไบนารี ซึ่งต้องอาศัยการใช้โครงสร้างข้อมูลแบบคงที่ (struct) ที่มีความเสี่ยงในการเกิดข้อผิดพลาด หากกระบวนการเข้ารหัสหรือถอดรหัสข้อมูลไม่ถูกต้อง อีกทั้งยังมีข้อจำกัดในการแสดงผล โดยเฉพาะในส่วนของชื่อหนังสือหรือชื่อผู้ใช้งานที่ถูกจำกัดความยาวตามที่กำหนดไว้ นอกจากนี้ ระบบในเวอร์ชันปัจจุบันยังไม่รองรับการเชื่อมต่อกับฐานข้อมูลจริง ทำให้ไม่สามารถรองรับปริมาณข้อมูลขนาดใหญ่ หรือการเข้าใช้งานพร้อมกันจากหลายผู้ใช้ได้อย่างมีประสิทธิภาพ

5.3 ข้อเสนอแนะ

5.3.1 พัฒนาให้รองรับการเชื่อมต่อกับฐานข้อมูลเชิงสัมพันธ์ (Relational Database) เช่น MySQL หรือ SQLite เพื่อเพิ่มขีดความสามารถในการจัดการข้อมูลจำนวนมากและรองรับผู้ใช้งานหลายคนในเวลาเดียวกัน

5.3.2 เพิ่มความสามารถในการค้นหาและกรองข้อมูล เช่น ค้นหาตามชื่อหนังสือ ผู้แต่ง หรือปีที่พิมพ์ เพื่อเพิ่มความสะดวกแก่ผู้ใช้งาน

5.3.3 ปรับปรุงระบบตรวจสอบและยืนยันตัวตนของสมาชิก พร้อมกับกำหนดสิทธิ์การเข้าถึงของผู้ใช้งานแต่ละระดับ เพื่อความปลอดภัยของข้อมูล

5.3.4 พัฒนาระบบให้มีอินเทอร์เฟซแบบกราฟิก (GUI) หรือรูปแบบเว็บแอปพลิเคชัน เพื่อให้ใช้งานได้ง่ายและเข้าถึงได้สะดวกมากยิ่งขึ้น

5.4 สิ่งที่ผู้จัดทำได้รับจากการพัฒนาโครงงาน

จากการดำเนินงานในโครงงานนี้ ผู้จัดทำได้รับความรู้และประสบการณ์ในการออกแบบระบบและเขียนโปรแกรมด้วยภาษา Python รวมถึงการประยุกต์ใช้โครงสร้างข้อมูลแบบไบนารี นอกจากนี้ยังได้พัฒนาทักษะในการคิดวิเคราะห์ แก้ปัญหาเชิงตรรกะ และการทำงานร่วมกับผู้อื่นเป็นทีม ฝึกการจัดสรรหน้าที่ความรับผิดชอบ และบริหารเวลาให้สอดคล้องกับแผนงานที่วางไว้ ส่งผลให้ผู้จัดทำเข้าใจกระบวนการพัฒนาระบบซอฟต์แวร์ได้ชัดเจนมากขึ้น และสามารถนำความรู้ที่ได้รับไปใช้ในการทำโครงการหรือการทำงานในอนาคตได้อย่างมีประสิทธิภาพ