FAKULTI TEKNOLOGI MAKLUMAT DAN KOMUNIKASI

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

BITP1113 PROGRAMMING TECHNIQUE
SEMESTER 1 2024/2025

PROJECT PROPOSAL

GROUP NAME	**BOOM SQUAD**
MEMBER'S NAME & MATRIC NO	
	OSMAN ABAZAR OSMAN (B152510377)
	MOHAMED MUNIR HASSAN (B152510314)
	MUHAMMAD AJMAL ASYRAF BIN MOHD FARIF (B152510340)
PROJECT TITLE	**BOOK BORROWING SYSTEM**
SECTION/GROUP	**S1G2**
PROGRAM	**BITZ**

Title

1.0 Project Introduction

Introduction – explains the definition of important keywords involved in the project.

Describe the background of the studied problem.

The reason for the study and its importance.

1.1 Problem Statement

State the issues that want to be resolved in the development of this project.

1.3 Objective

Each project can contain more than one objective.

Represents the things that need to be achieved/implemented throughout the implementation of the project towards solving problems and achieving goals.

Project objectives should be written in point form in a simple, clear and measurable level of implementation and success of each one.

Table of Contents

Introduction

The **Book Borrowing System** is a computerized C++ application designed to **systematize** the management of book transactions. It replaces manual logging methods with a structured digital approach. To understand the technical scope of this project, it is essential to define the important keywords and concepts involved:

- **Data Validation:** The process of ensuring that input data such as dates follows logical rules before it is processed. This prevents errors like 30th February or negative months from entering the system.
- **Borrowing Duration:** The calculated time difference (in days) between the Borrow Date and the Return Date. This calculation is the foundation for determining if a book is late.
- **Overdue Status:** A classification assigned to a book when the Borrowing Duration exceeds the library's fixed limit (currently set to 7 days).
- **Fine Automation:** The algorithmic process of calculating financial penalties automatically rather than relying on manual multiplication.

In many small-scale libraries, borrowing records are still maintained using manual logbooks or simple spreadsheets. This traditional method lacks organization and safeguards. Librarians often struggle to keep records organized because handwriting can be illegible and there is no automatic way to check if a date is written correctly. Furthermore, determining if a book late requires the librarian to physically count days on a calendar, a process that is slow and prone to human error especially when transactions span across different months or leap years.

The primary reason for this study is to resolve the disorganization and inaccuracy inherent in manual systems by developing a C++ program that enforces a systematic structure on the borrowing process. The importance of this project lies in its ability to ensure **data integrity** by recording only valid dates and maintain **accuracy** by eliminating human error in day calculations. Furthermore, it guarantees **consistency** by automatically applying the 7-day rule to every transaction and significantly improves **efficiency** by instantly generating an organized final bill, thereby saving time for both library staff and customers.

Problem Statement

The development of this project addresses four specific issues inherent in manual library borrowing processes:

1. **Lack of Data Validation**
   Manual recording systems often fail to detect invalid or illogical entries immediately. This leads to record-keeping errors, such as recording non-existent dates or negative timeframes.

2. **Complex Date Calculations**
   Manually calculating the exact number of days between two dates is difficult and prone to human error, particularly when borrowing periods span across different months or involve leap years.

3. **Manual Status Tracking**
   Identifying whether a book is **On Time** or **Overdue** requires the librarian to manually compare every transaction against the library's policy, which is tedious and risks oversight.

4. **Inefficient Fine Assessment**
   Calculating the total fine for a customer with multiple overdue books requires repetitive mathematical steps. This slows down the checkout process and increases the likelihood of billing mistakes.

Objective

The main goal of this project is to develop a C++ program that **systematizes the book borrowing process**. The system aims to organize input data efficiently to ensure all records are accurate and structured, while simultaneously automating the calculation of overdue fines to eliminate manual processing errors. The specific objectives are:

1. To **design a robust input mechanism** that accepts and validates user data such as borrow and return dates to ensue logical consistency and data integrity. For example, the system correctly handling leap years.
2. To **implement a date-processing algorithm** that accurately calculates the total borrowing duration in days between any two given dates.
3. To **automate the status tracking** of borrowed items by comparing the calculated duration against the fixed 7-day allowed period.
4. To **compute financial penalties** automatically by multiplying the number of overdue days by the daily penalty rate (RM 2.00) and summing the results for a clear, organized final bill.

Input, Process, Output (IPO)

Input

- Customer name
- Number of books (n)
- Book name
- Borrow date
- Return date

Process

1. **Input Validation**

- Validate number of books (Ensure it is a valid number)
- Validate Borrow date (Check for valid day/month/year)
- Validate Return date (Check for valid day/month/year)

2. **Calculation & Logic**

- Calculate the borrow duration (Return Date - Borrow Date)
- Calculate the overdue days (Duration - 7 days)
- Determine if the book is overdue (Check if Duration > 7)
- Calculate fine (Overdue days × Fine rate)
- Add fine to total fine

Output

- **Welcome** message
- Error messages (e.g., "Invalid no of books", "Invalid date")
- Borrowing Summary Header
- Book Details (Name, Dates, Duration)
- Book Status **[ON TIME]** or **[OVERDUE]**
- Individual Fine Amount
- Total Fine to Pay
- **Thank You** message

Input	Process	Output
Customer Data: • Customer Name	**Input Validation:** • Check if **Number of books** is a valid number. • Check if **Borrow/Return dates** are valid calendar dates (Day/Month/Year).	**User Prompts & Status:** • **Welcome** message. • Error messages (e.g., "Invalid date", "Invalid no of books")
Transaction Data: • Number of books (n)	**Calculations:** • Calculate **Borrow Duration** (Return Date - Borrow Date). • Calculate **Overdue Days** (Duration - 7 days). • Calculate **Fine Amount** (Overdue Days × Rate). • Accumulate **Total Fine**.	**Borrowing Summary:** • Customer Name • Book Title • Duration & Dates • Status **[ON TIME]** or **[OVERDUE]**
Book Data: *(Repeated for each book)* • Book Name • Borrow Date • Return Date	**Logic Decisions:** • Determine if **Borrow Duration** > 7 days (Overdue Check).	**Final Results:** • Individual Book Fine • Total Fine to Pay • **Thank You** message.

Table 1 shows the IPO table.

Flowchart

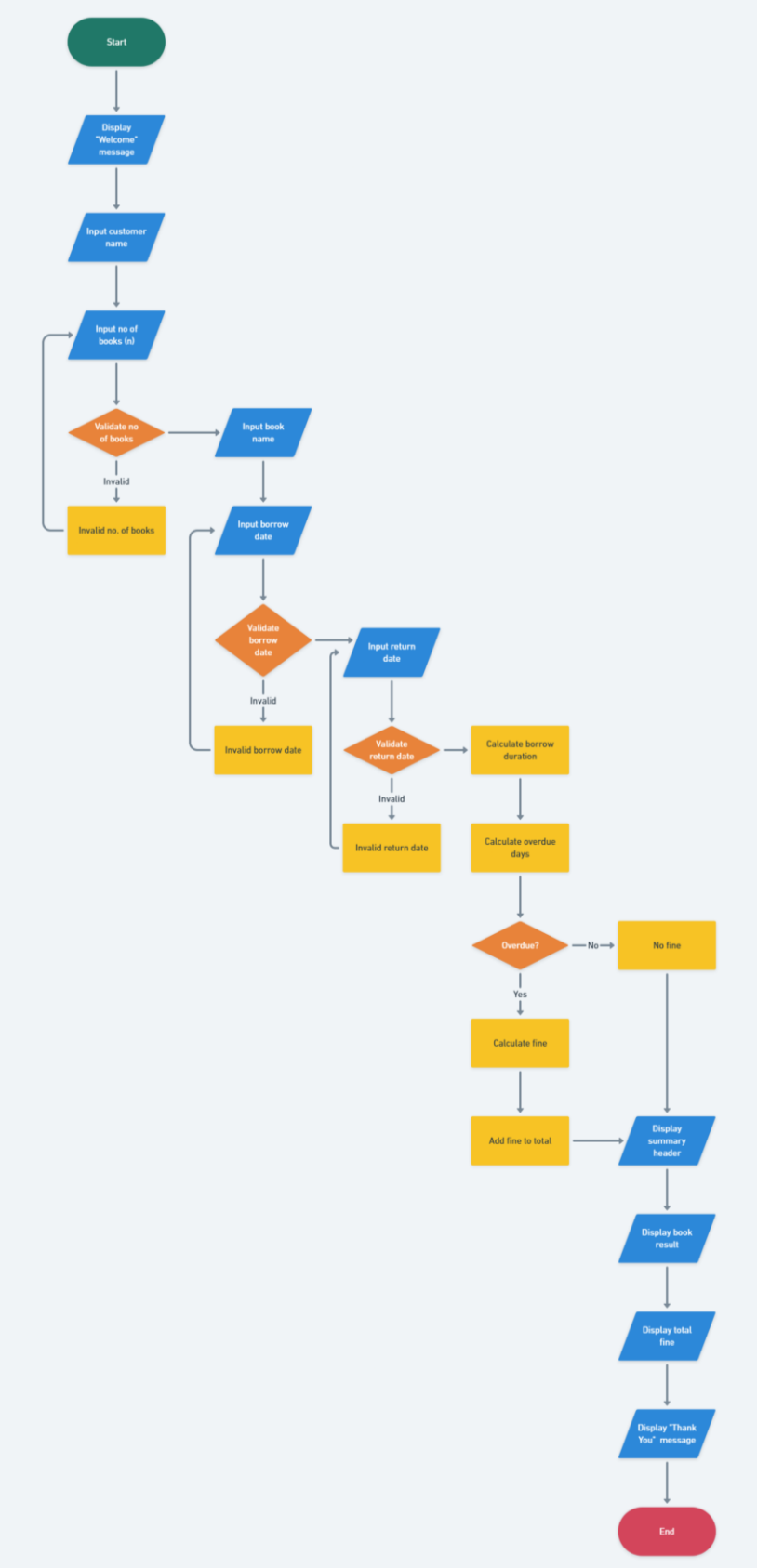

Figure 1 shows the flowchart of the program.

User Guide: Book Borrowing System

This User Guide provides step-by-step instructions on how to operate the **Book Borrowing System**. This console-based C++ program helps librarians record borrowing transactions, calculate durations and automatically determine if a customer owes a fine.

Step 1: Enter Customer Information

The system will prompt you for the customer's name.

- **Action:** Type the full name of the customer and press **Enter**.

Note: The system accepts names with spaces (e.g., "Ali Bin Abu").

Step 2: Enter Number of Books

The system will ask how many books the customer wishes to borrow.

- **Action:** Type a whole number (e.g., 3) and press **Enter**.
- **Validation:** If you type letters or symbols, the system will display an error: *"Error! Please enter numbers only"* and ask you to try again.

Step 3: Enter Book Details

For every book (based on the number you entered in Step 3), the system will ask for the following details sequentially:

1. **Book Name:**
    - Type the title of the book (e.g., *Intro to C++*) and press **Enter**.

2. **Borrow Date:**
    - Enter the date in the format **DD MM YYYY** separated by spaces (e.g., 01 02 2024).
    - *Validation:* The system will reject invalid dates (like 30 02 2024) and ask you to re-enter.
3. **Return Date:**
    - Enter the date the book was returned in the same **DD MM YYYY** format (e.g., 10 02 2024).

Step 4: View Borrowing Summary

Once all data is entered, the system will automatically process the information and display the **Borrowing Summary**.

- **Status Check:** Look for the status tag.
    - **[ON TIME]**: No fine is charged.
    - **[OVERDUE]**: The system displays the number of overdue days and the specific fine amount (RM).
- **Grand Total:** At the bottom, check the **"Total Fine to Pay"** to see the final amount the customer owes.

Step 5: End of Program

After displaying the total, the system shows a **"THANK YOU"** message and the program terminates. To process a new customer, restart the program.

Input & Output of the Program

Test Data

Prompt	Input to Type
Customer Name	Ali Bin Abu
How many books?	3
[Book 1] Name	**Intro to C++**
Borrow Date	01 02 2024
Return Date	05 02 2024
[Book 2] Name	**Data Structures**
Borrow Date	25 01 2024
Return Date	05 02 2024
[Book 3] Name	**Practical Malware Analysis**
Borrow Date	27 02 2024
Return Date	10 03 2024

Table 2 shows the sample data used in this program.

The table above outlines the input values used to validate the functionality of the **Book Borrowing System**. The test case involves a single customer, *Ali Bin Abu,* borrowing three distinct books to test different logical scenarios:

- **Case 1: On Time Return (Intro to C++)**
  Tests a standard, short-term borrowing period (4 days) to verify that **On Time** returns incur no fines.
- **Case 2: Overdue Calculation (Data Structures)**
  Tests a longer borrowing period (11 days) that spans across two different months (January to February) to verify overdue calculation logic.

- **Case 3: Leap Year Logic Verification (Practical Malware Analysis)**
  Tests the system's "Leap Year" capability by using dates in February 2024. This ensures the algorithm correctly accounts for the extra day (February 29th) when calculating the total duration and subsequent fines.

Input

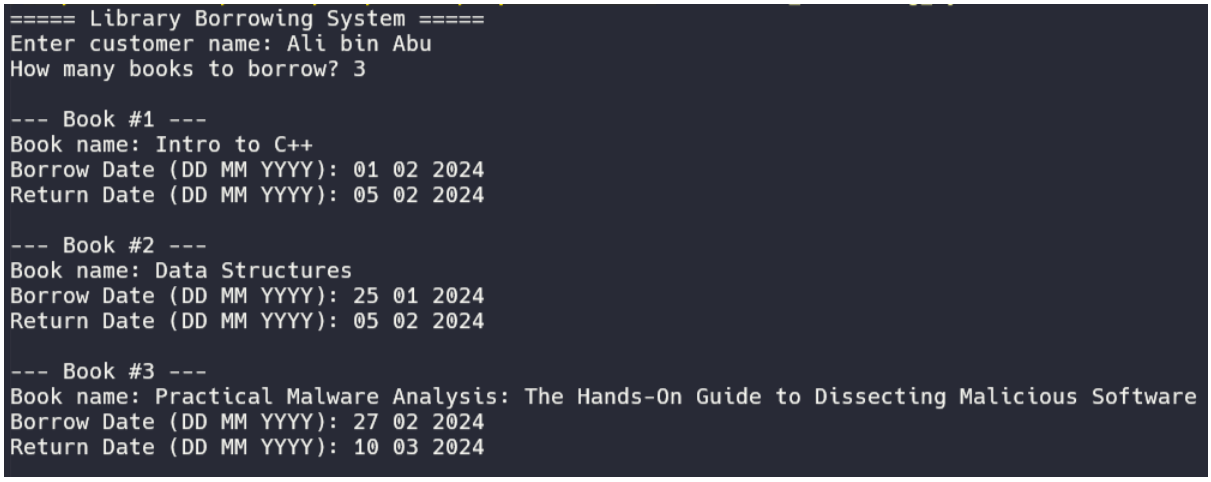

Figure 2 shows the sample data inserted in the program (Input).

Output

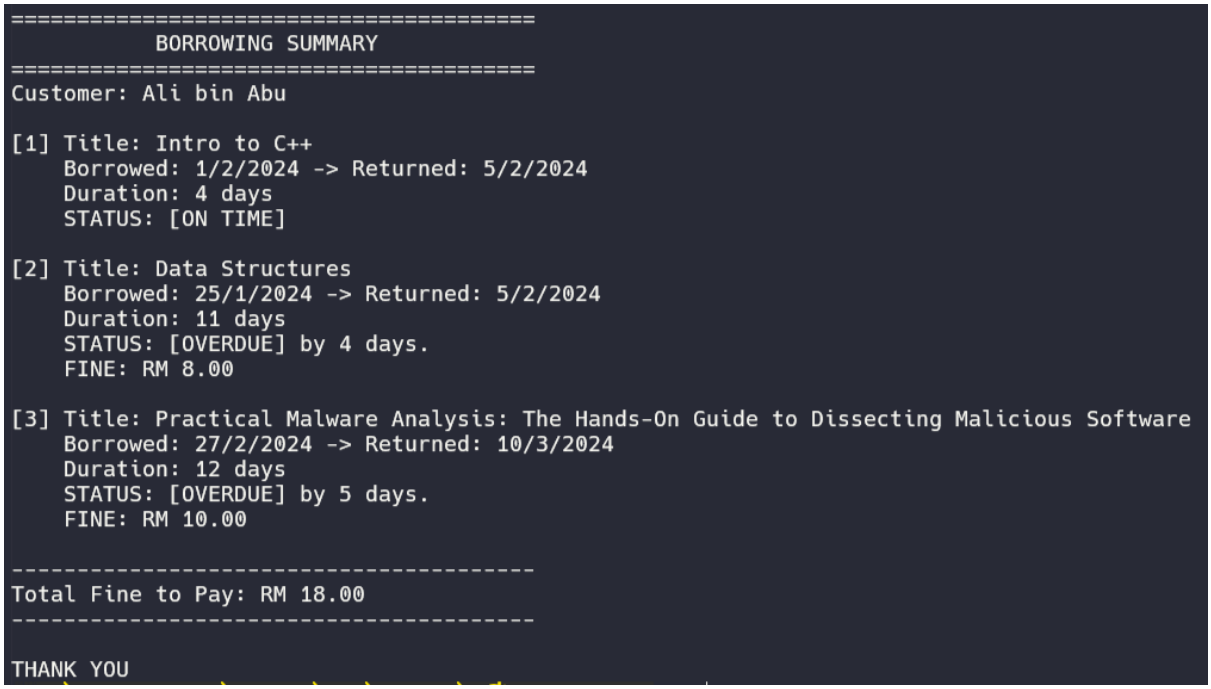

Figure 3 shows the output of the program.

The screenshot above demonstrates the successful execution of the Book Borrowing System for customer *Ali bin Abu*. The output verifies that the program correctly processes dates, calculates durations, and applies fines according to the system rules:

1. **Case 1: On-Time Return (Intro to C++)**
    - For the book *Intro to C++*, the borrowing duration was calculated as **4 days** (Feb 1 to Feb 5).
    - Since this is within the allowed 7-day limit, the system correctly assigned the status **[ON TIME]** and applied **no fine**.

2. **Case 2: Overdue Calculation (Data Structures)**
    - For *Data Structures*, the book was held for **11 days**.
    - The system accurately identified it as overdue. It calculated the excess time as **4 days** and multiplied this by the fine rate (RM 2.00) to generate a fine of **RM 8.00**.

3. **Case 3: Leap Year Logic Verification (Practical Malware Analysis)**
    - For *Practical Malware Analysis* book, the transaction spanned from **27th February 2024** to **10th March 2024**.
    - Because 2024 is a leap year, February has 29 days. The system correctly accounted for this extra day to calculate a total duration of **12 days**.

Conclusion

In conclusion, the development of the **Book Borrowing System** has successfully achieved its primary objective of systematizing the book transaction process. By transitioning from a manual recording method to a structured C++ application, the project has effectively resolved the issues of data disorganization and human error identified in the problem statement.

Through the implementation of robust programming techniques specifically input validation and modular date algorithms, the system ensures that all recorded data is consistent and accurate. A significant technical achievement verified during testing is the system's ability to correctly handle Leap Year calculations, ensuring that borrowing durations are precise regardless of calendar irregularities (as seen in the Practical Malware Analysis test case). Furthermore, the automation of the Overdue Status and Fine Calculation guarantees that the library's 7-day policy is enforced fairly and efficiently, without requiring manual intervention.

Ultimately, this project demonstrates that a computational approach significantly enhances administrative workflows. The system not only saves time for library staff by generating instant, error-free bills but also provides a transparent and organized experience for customers, fully meeting the project's goals of accuracy, integrity, and efficiency.