## Course: PFP191 - Programming Fundamentals with Python

# Book management in the library

| **Group: 2** | |
|---|---|
| **Group Members** | Nguyễn Phúc Thịnh - SE200943<br>Chu Văn Tuấn Đạt - SE200915<br>Đoàn Hải Đăng - SE201996<br>Ngô Minh Nhựt - SE204149 |
| **Lecturer** | Lê Thị Hồng Nga |
| **Class** | IA2001 |

Ho Chi Minh, Nov – 2025

## Table of Contents

## 1. Introduction

### 1.1. Project Description

The project develops a **Book management system in the library** using the Python programming language.

The program allows Book Management, Add, Edit and Delete books, Borrowing and Returning books, Statistics and Reports, saving/reading data from files while applying object-oriented programming (OOP), exception handling, and templates.

### 1.2. Project Team

| # | Student ID | Full Name | Completion Level | Tasks Performed |
|---|---|---|---|---|
| 1 | SE200943 | Nguyễn Phúc Thịnh | 100 | Lab1: Data<br>Lab2: Function Borrowing and Returning<br>Lab3: Function Borrowing and Returning<br>Lab 4: Function Borrowing and Returning<br>Do a full review |
| 2 | SE200915 | Chu Văn Tuấn Đạt | 100 | Lab1: Function Book Management<br>Lab2: Function Book Management<br>Lab3: Function Book Management<br>Lab4: Function Book Management |
| 3 | SE201996 | Đoàn Hải Đăng | 100 | Lab1: Function File I/O and Testing<br>Lab2: Function File I/O<br>Lab3: Function File I/O<br>Lab4: Function File I/O |
| 4 | SE204149 | Ngô Minh Nhựt | 100 | Lab1: Function Borrowing and Returning & Statistics and Reports<br>Lab2: Function Statistics and Reports<br>Lab3: Function Statistics and Reports<br>Lab4: Function Statistics and Reports |

## 2. Requirement Analysis & Decomposition

### 2.1. Data

| # | Name | Description |
|---|---|---|
| 1 | books | List includes all books in library |
| 2 | ID | Unique identifier of the book |
| 3 | title | Title of the book |
| 4 | author | Author of the book |
| 5 | category | Book category |
| 6 | publication_Year | Year when the book was published |
| 7 | availability | Book status: available or borrowed |
| 8 | count | count the number of times borrowed |

### 2.2. Functions

| # | Function | Description |
|---|---|---|
| 1 | Book Management | Manage a collection of books |
| 2 | Borrowing and Returning | This function allows users to manage the borrowing and returning process of books. |
| 3 | Statistics and Reports | This function allows users to generate reports on books by category and most borrowed books. |
| 4 | File I/O | Save/load book data from .txt files |
| 5 | Testing | Verify that the main functions work correctly and reliably |

1.1 Addbook - Input required fields (e.g., ID, title, author, publicationYear, category,)

1.2 Editbook - Find book by ID and update specified fields

1.3 Deletebook - Remove by ID or mark as inactive (soft delete)

1.4 Displaybooklist - Show all books in library

1.5 Searchforbooks - Quick search by ID, title, or author ; UX features: autocomplete suggestions, highlight matched terms, and show number of results

2.1 borrowBook – Borrow a book from the library, updating its status (available/borrowed).

2.2 returnBook – Return a borrowed book, restoring its status (available for borrowing).

2.3 listBorrowedBooks – Display a list of all currently borrowed books.

3.1 booksByCategory – Display on the number of books by category (e.g., Science, Literature, History…).

3.2 mostBorrowedBooks – Display a list of the most borrowed books, helping track the popularity of books.

4.1 saveToFile - Save all book data to a text file (book.txt)

4.2 loadFromFile - Load book data from a text file into the program

5.1 testAddBook - Verify adding a valid book works correctly.

5.2 testSearchBook - Check searching by ID/Title returns results.

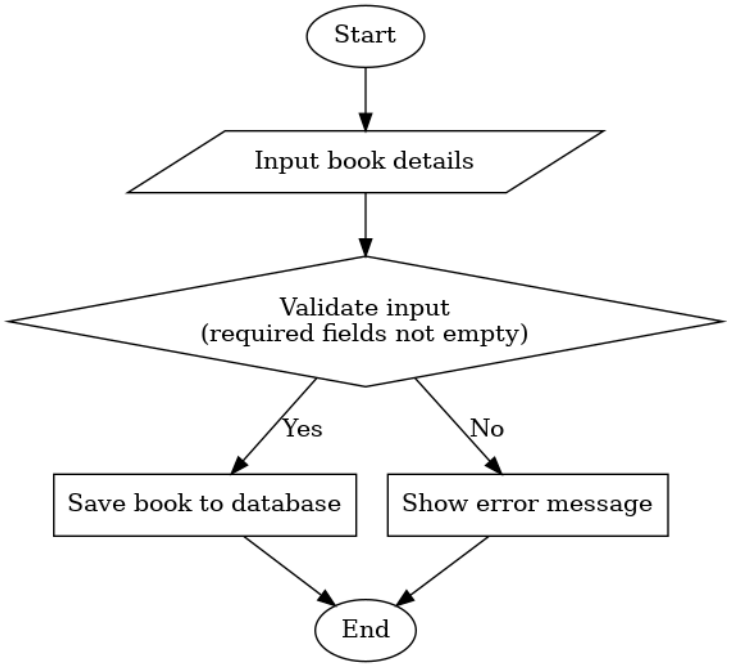
5.3 testBorrowBook - Ensure borrowing a book updates availability.

5.4 testReturnBook - Ensure returning a book resets availability.

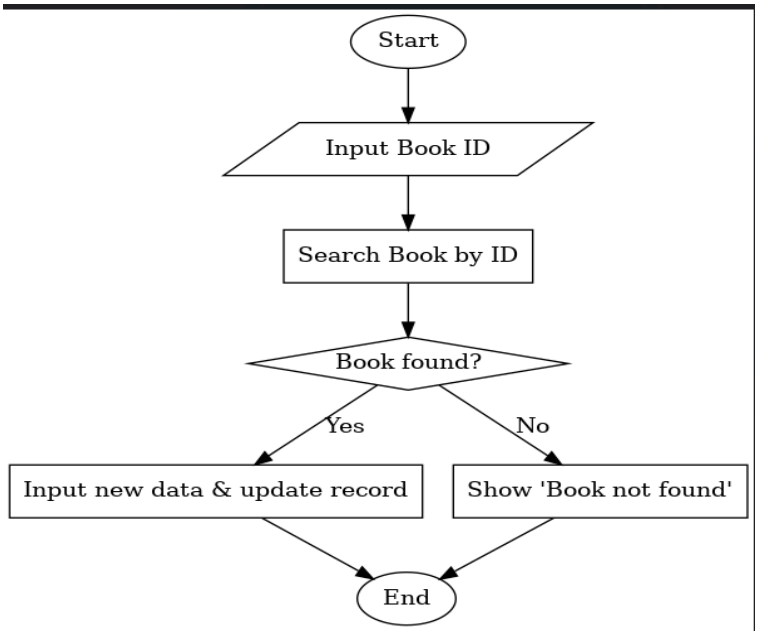
5.5 testSaveLoad - Confirm data is saved to file and loaded back.

## 3. Algorithm Design & Flowcharts

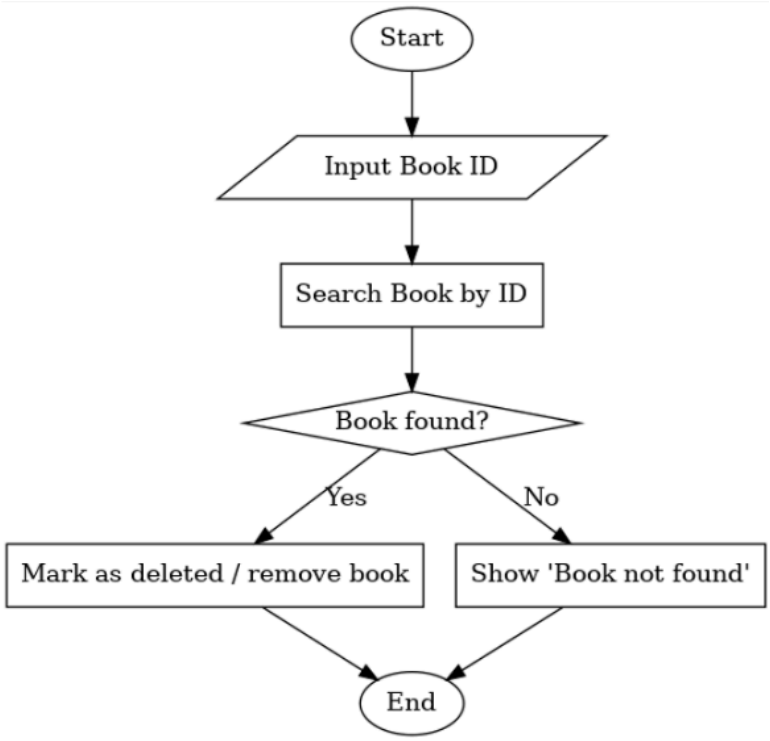
1.1 Addbook



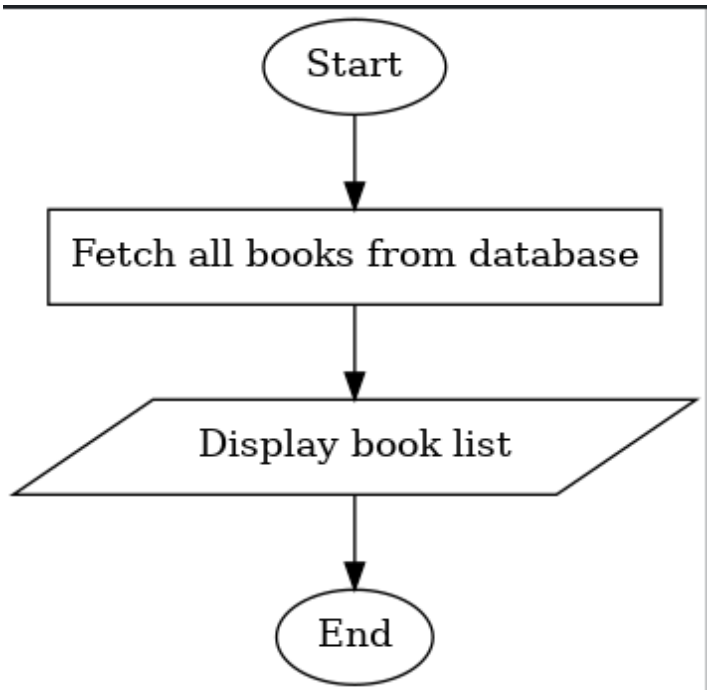

1.2 Edit book



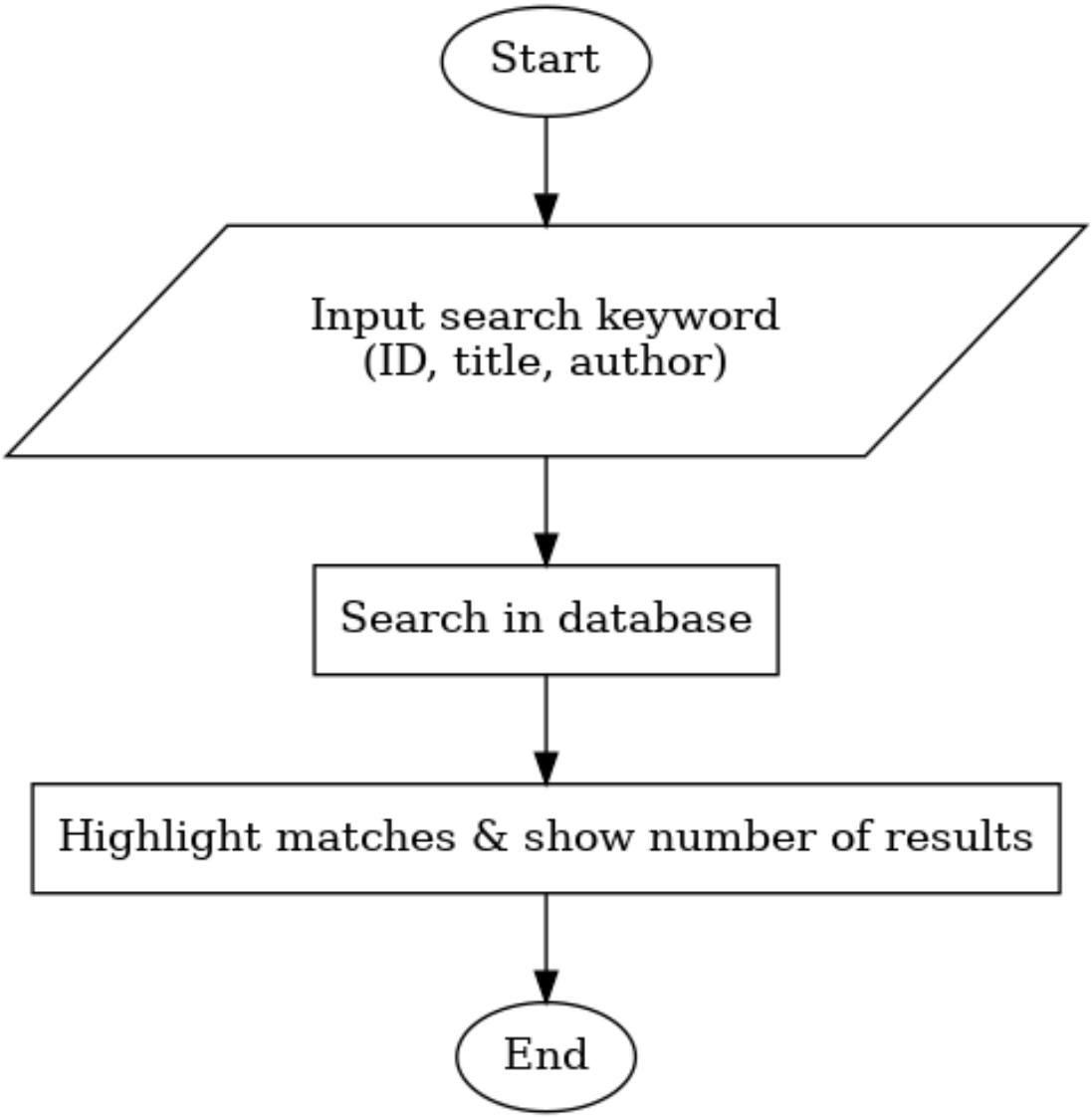
1.3 Delete book



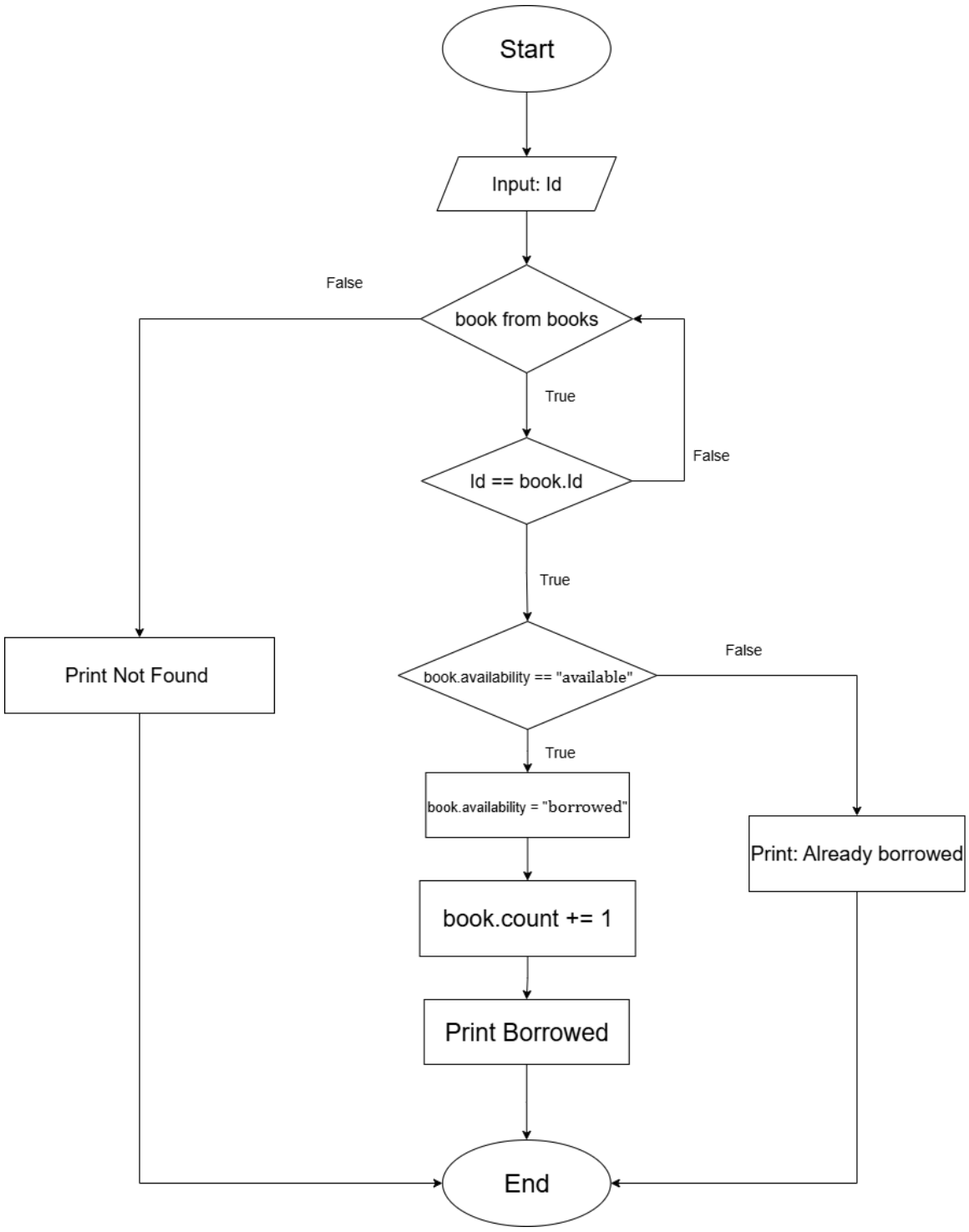

1.4 Displaybooklist



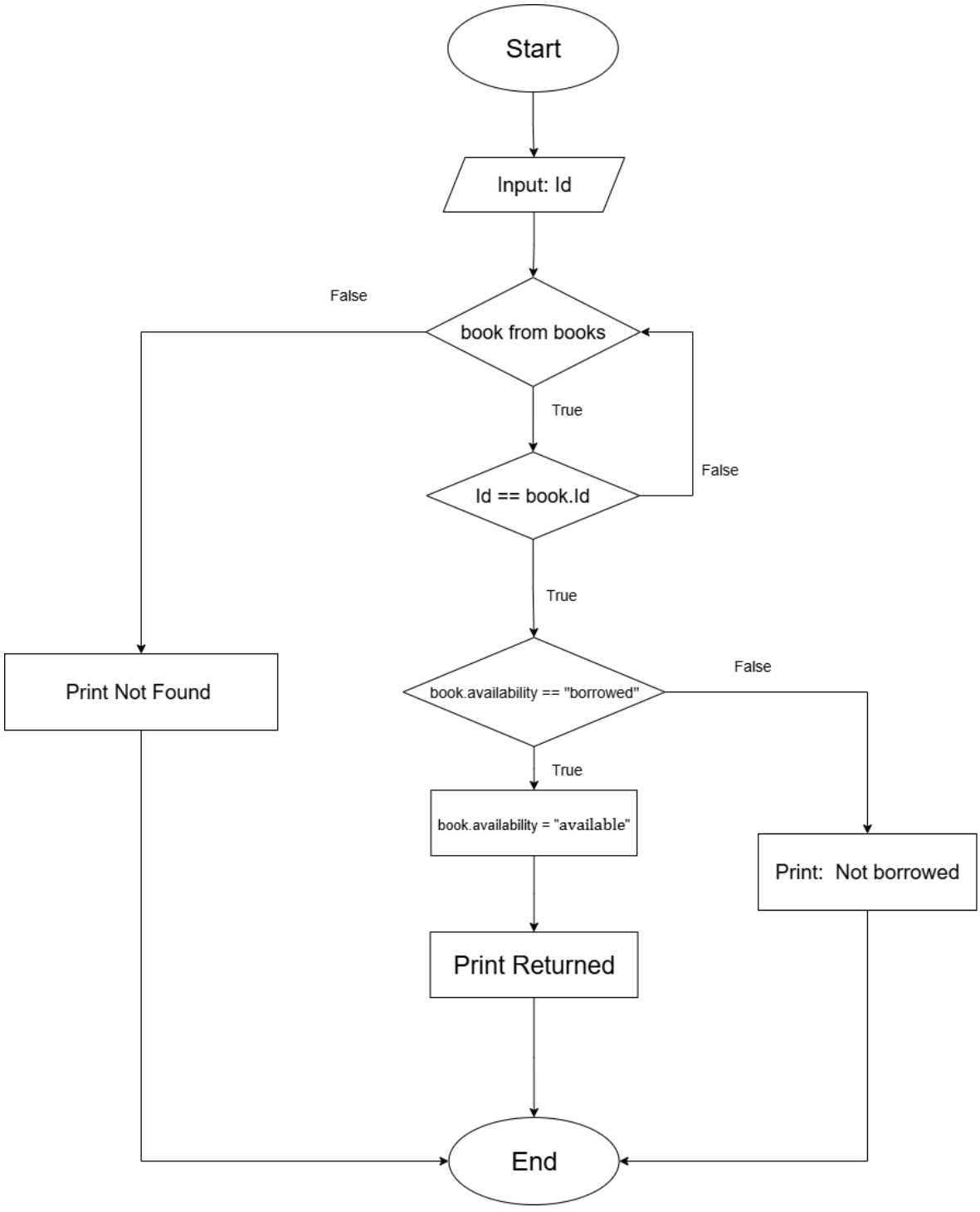
1.5 Searchforbooks



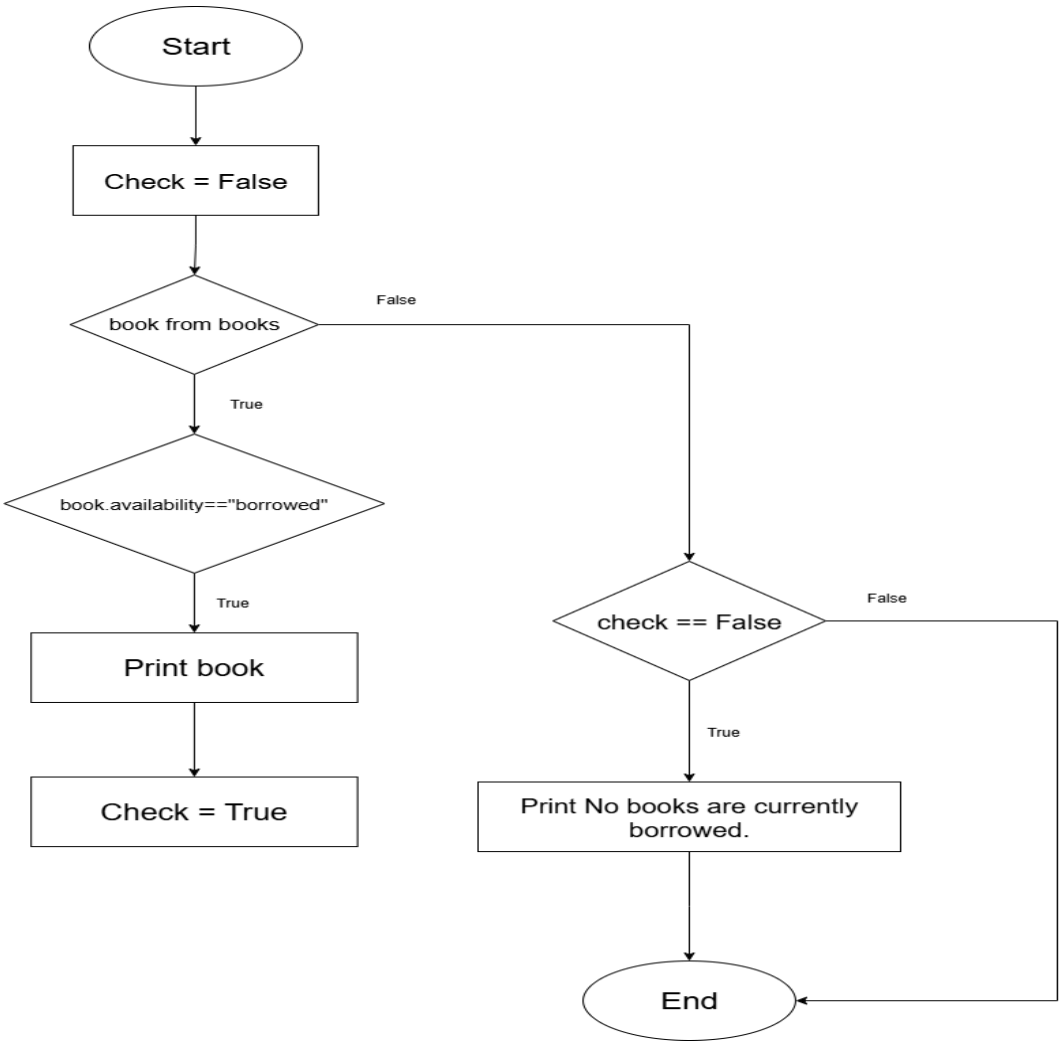
2.1 borrowBook



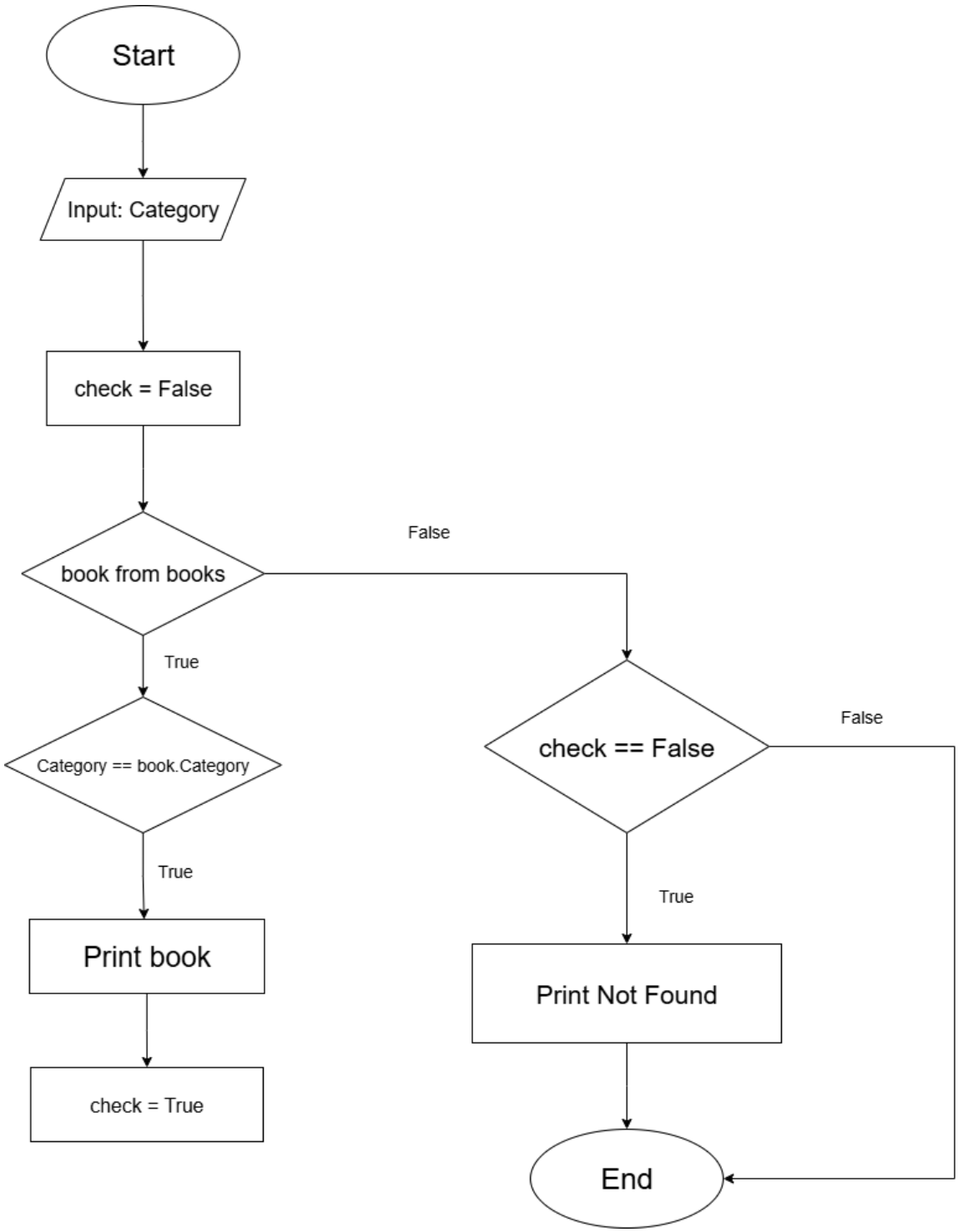
2.2 returnBook



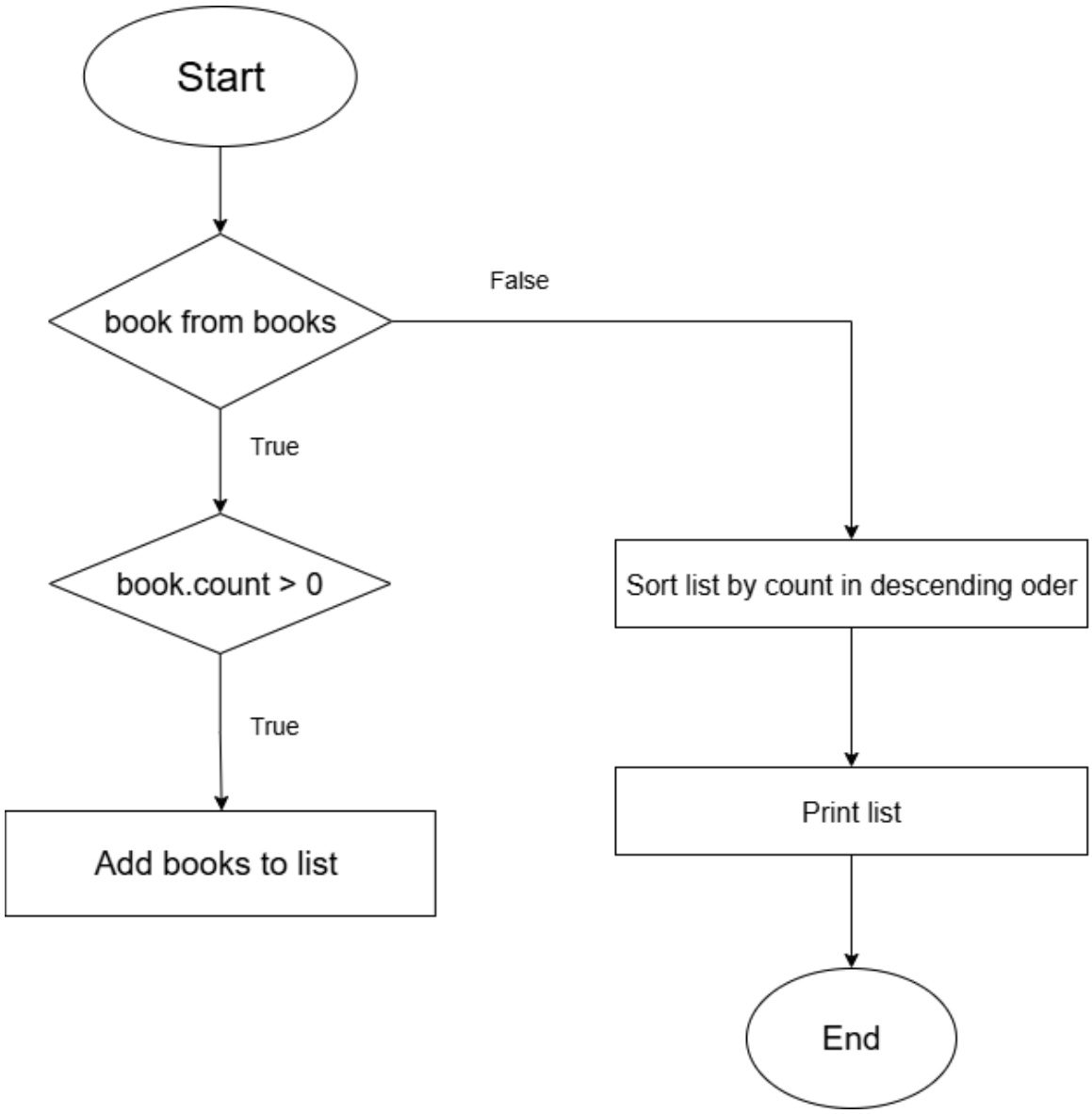
2.3 listBorrowedBooks



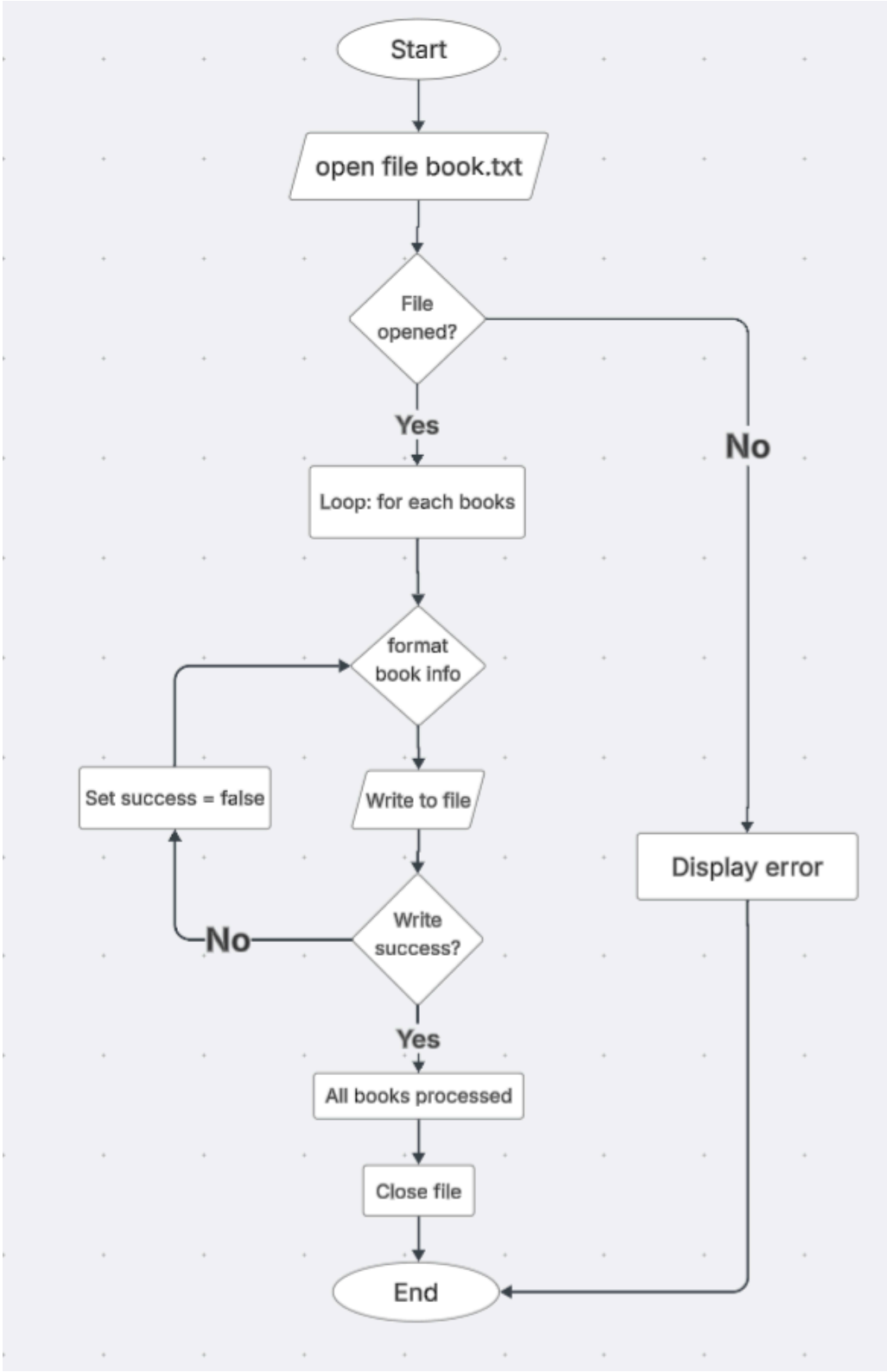
3.1booksByCategory



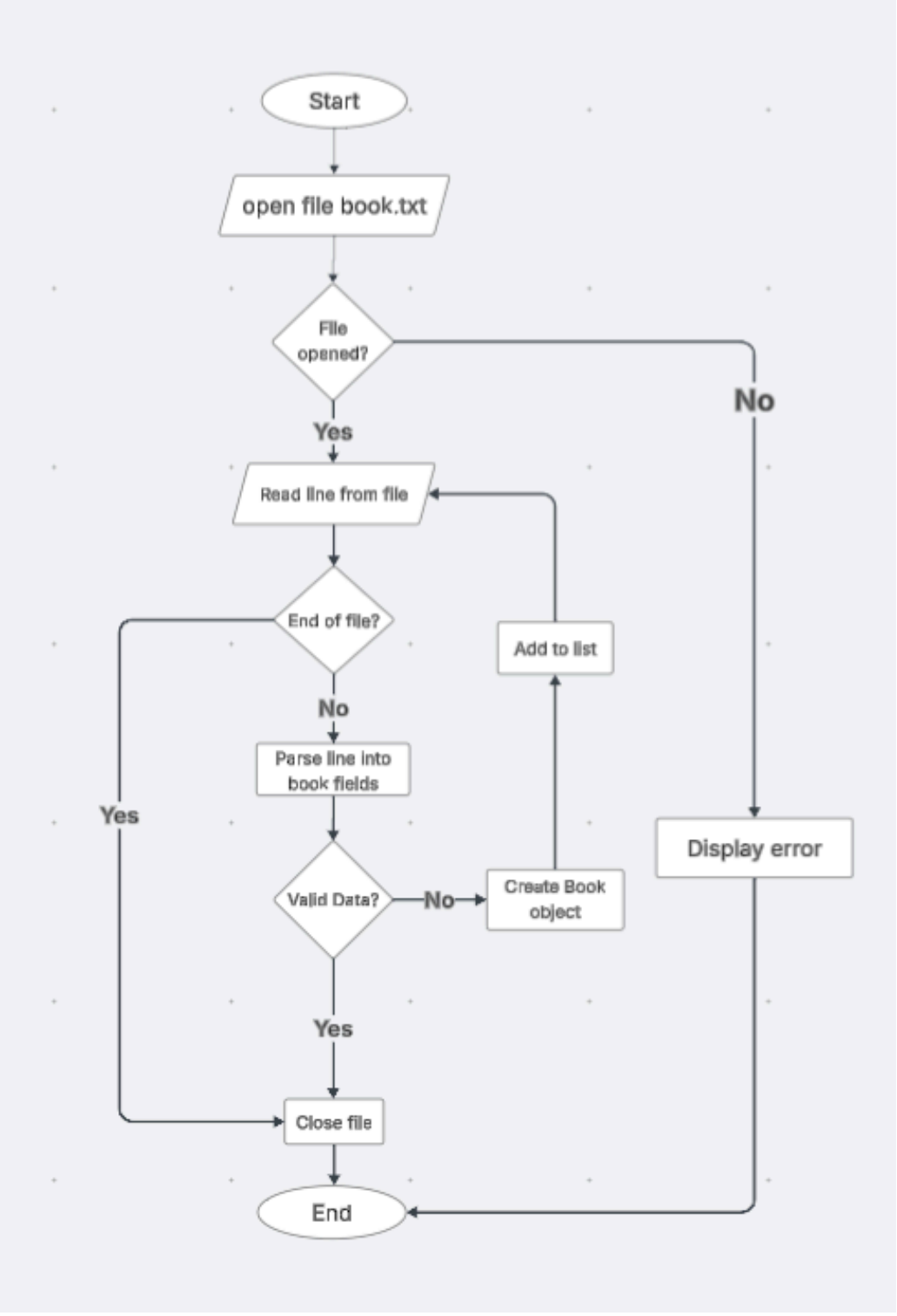
3.2mostBorrowedBooks



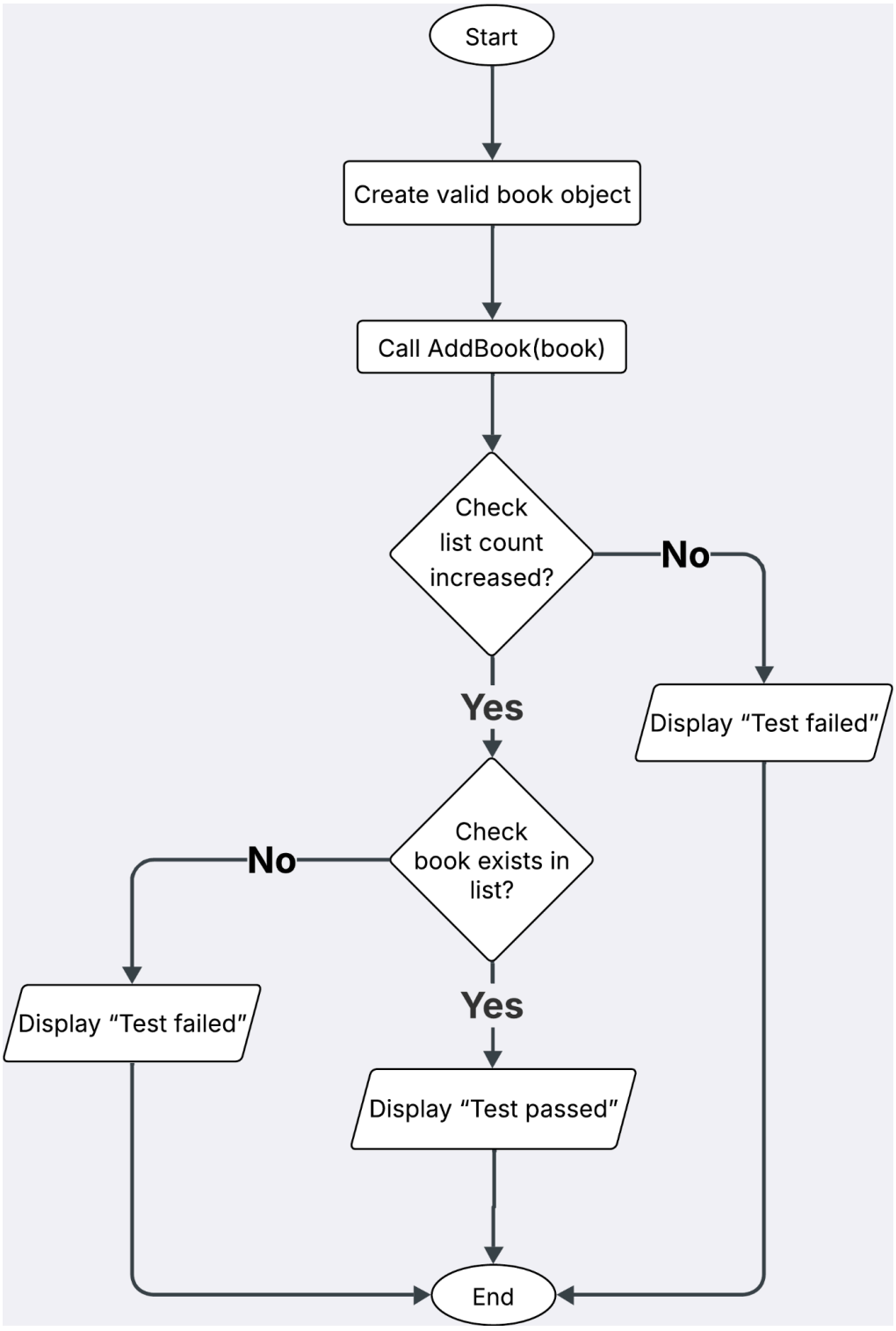
4.1 saveToFile



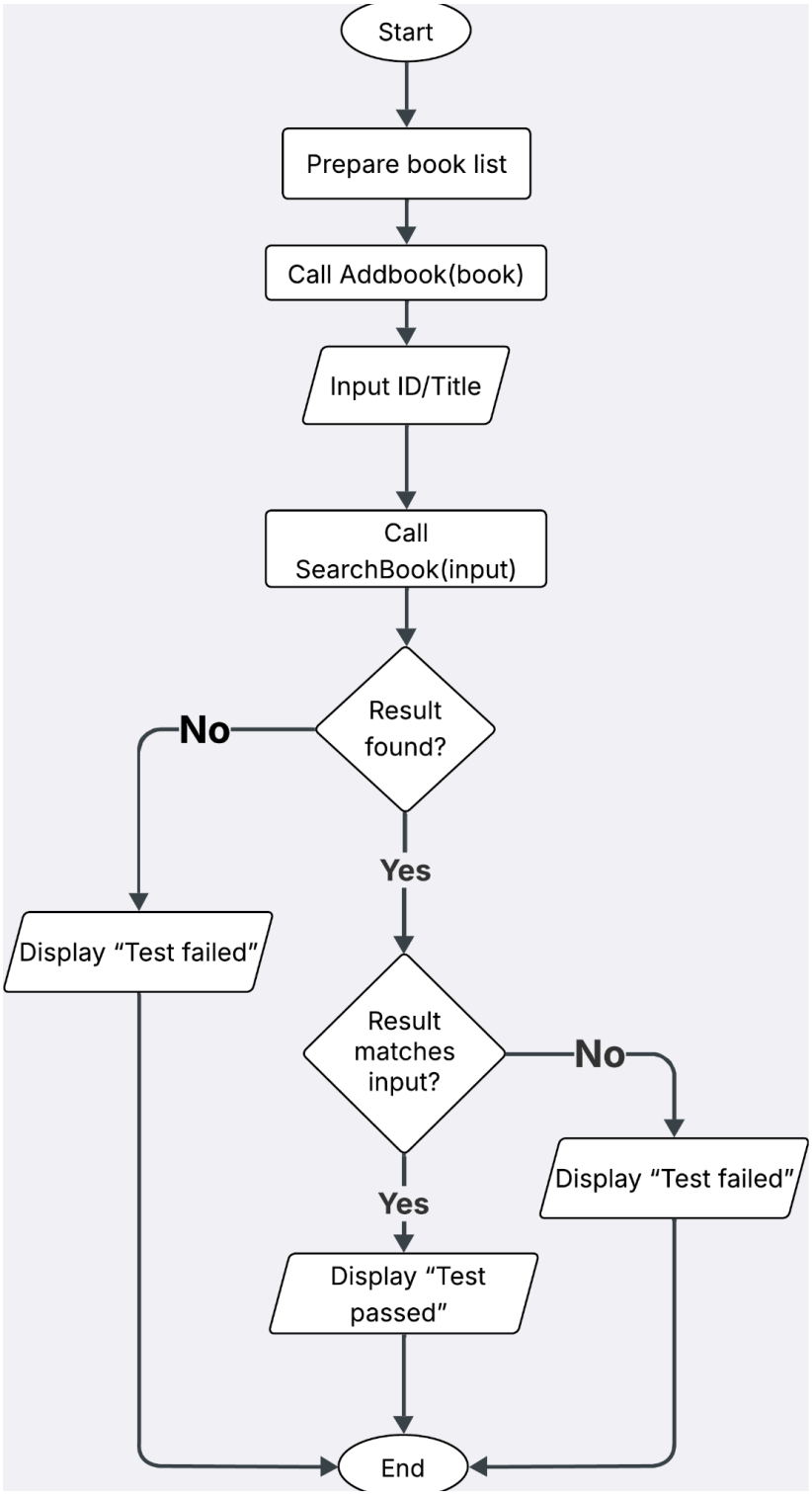
4.2 loadFromFile



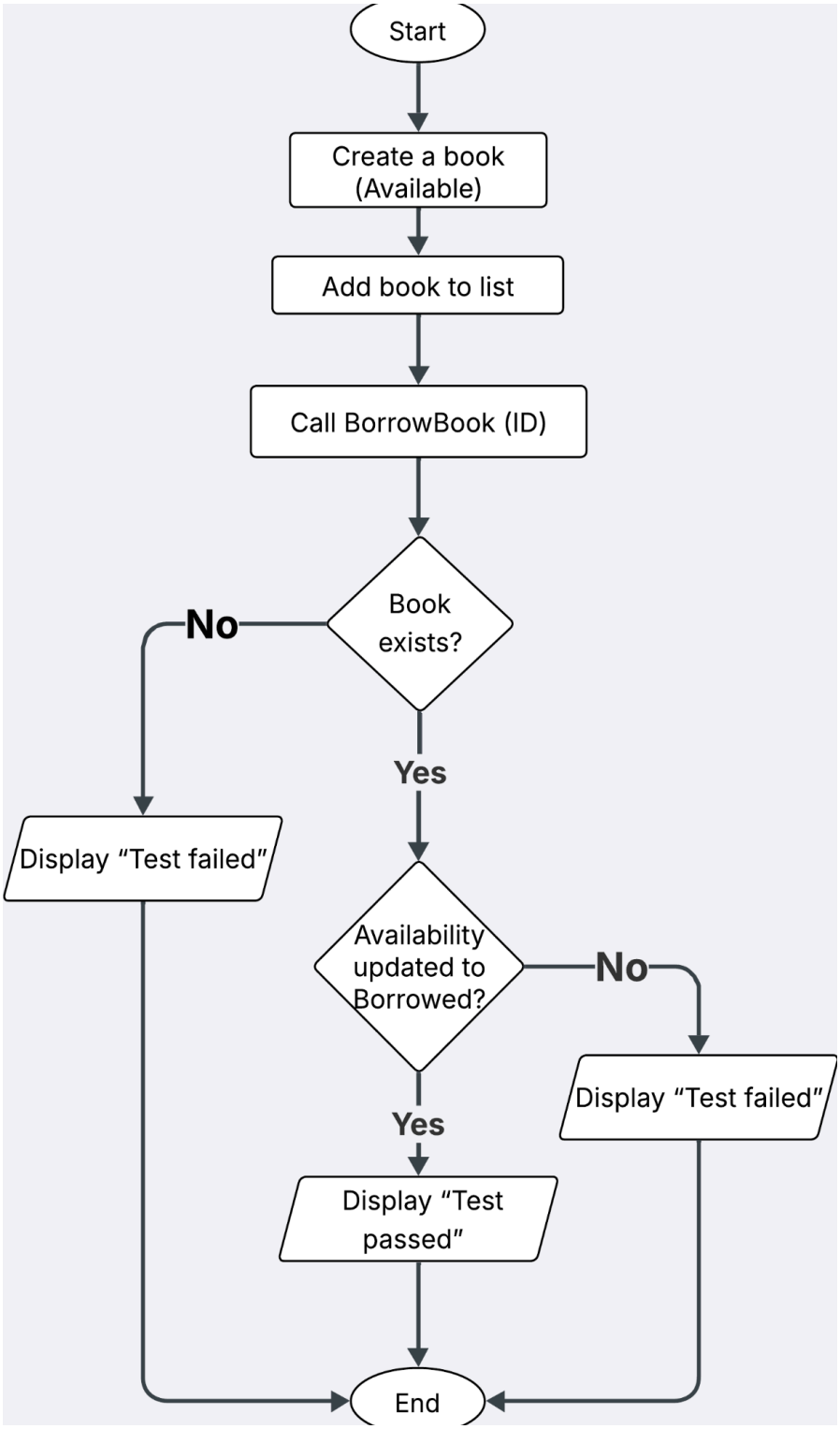
5.1 testAddBook



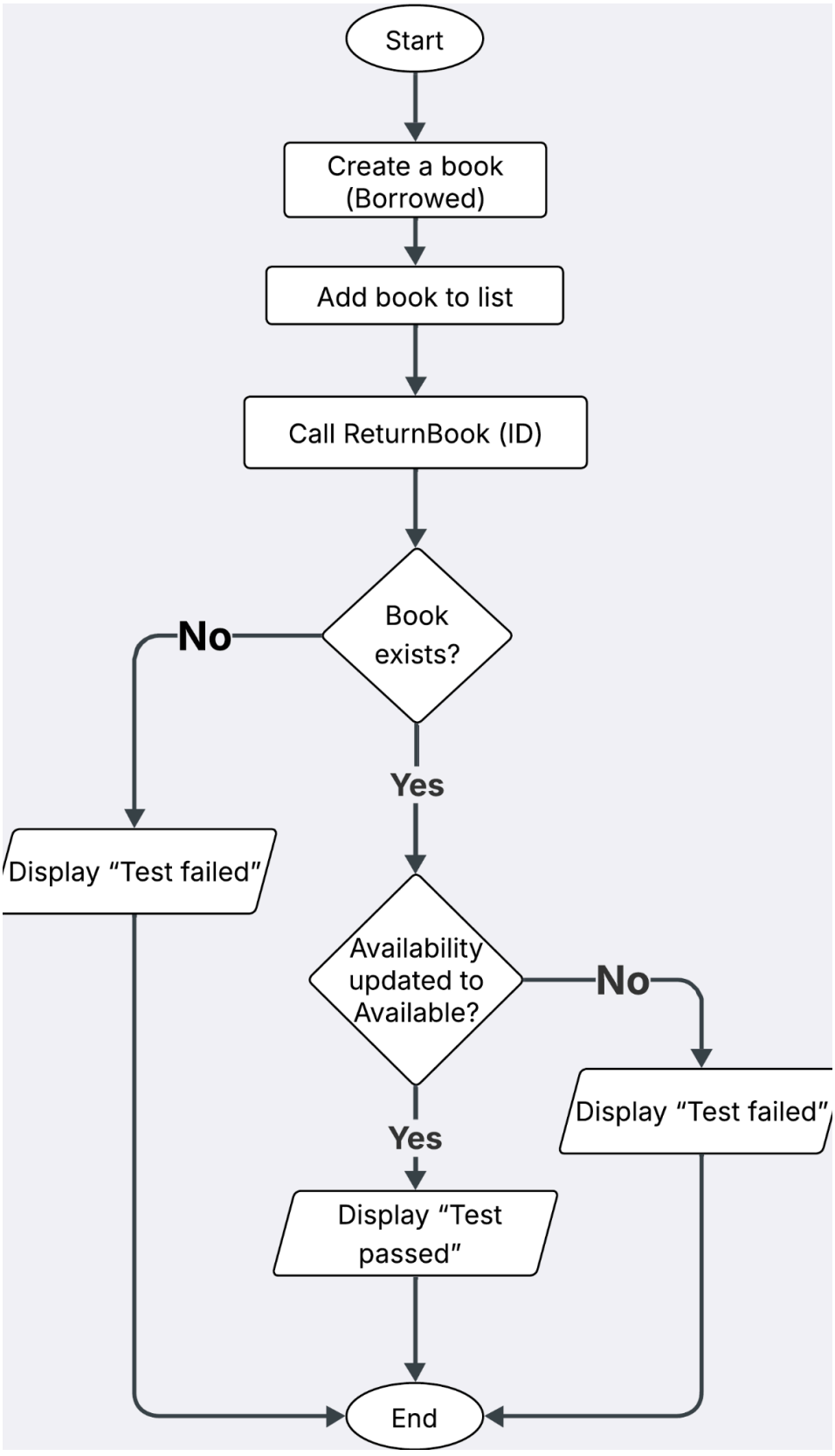
5.2 testSearchBook



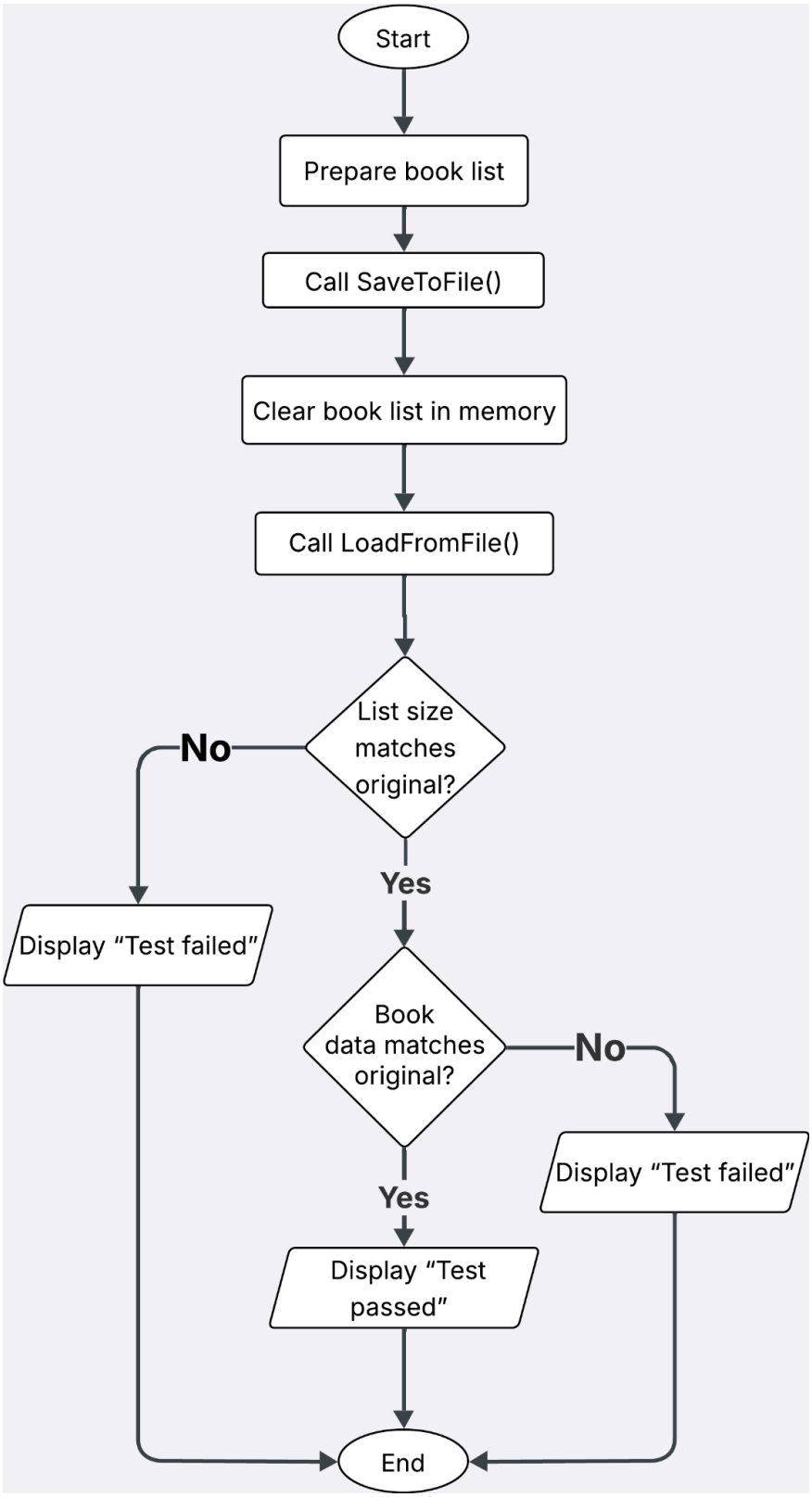
5.3 testBorrowBook



5.4 testReturnBook



5.5 testSaveLoad



## 4. Implementation of Basic Functions

1.1 Addbook

```
def Addbook():
    book = {}
    while True:
        try:
            book['ID'] = int(input("Enter Book ID: "))
            break
        except ValueError:
            print("Invalid input! Please enter a number for Book ID.")
    book['author'] = input("Enter Author: ")
    while True:
        try:
            book['publicationYear'] = int(input("Enter publicationYear: "))
            break
        except ValueError:
            print(" Invalid input! Please enter a number for publicationYear.")
    book['category'] = input("Enter Category: ")
    book['title'] = input("Enter Title: ")
    book['availability'] = input("Enter Availability: ")
    while True:
        try:
            book['count'] = int(input("Enter Count: "))
            break
        except ValueError:
            print("Invalid input! Please enter a number for Count.")
    books.append(book)
    print("Book added successfully")
```

## 1.2 Editbook

```
def Editbook():
    while True:
        book_ID = input("Enter Book ID to edit: ")
        if book_ID.isdigit():
            book_ID = int(book_ID)
            break
        else:
            print("Invalid input! Please enter a number for Book ID.")
    for book in books:
        if book['ID'] == book_ID:
            print(f"Editing book: {book['ID']}")
            book['author'] = input("New Author : ")
            while True:
                publicationYear_input = input("New publicationYear: ")
                if publicationYear_input.isdigit():
                    book['publicationYear'] = int(publicationYear_input)
                    break
                else:
                    print(" Invalid input! Please enter a number for publicationYear.")
            book['category'] = input("New Category : ")
            book['title'] = input("New Title: ")
            while True:
                try:
                    book['count'] = int(input("Enter Count: "))
                    break
                except ValueError:
                    print("Invalid input! Please enter a number for Count.")
            book['availability'] = input("Enter Availability: ")

            print("Book updated")
            return
    print("Book not found")
```

## 1.3 Deletebook

```
def Deletebook():
    while True:
        book_ID = input("Enter Book ID to delete: ")
        if book_ID.isdigit():
            book_ID = int(book_ID)
            break
        else:
            print("Invalid input! Please enter a number for Book ID.")
    for book in books:
        if book['ID'] == book_ID:
            books.remove(book)
            print("Book deleted")
            return
    print(" Book not found")
```

## 1.4 Displaybooklist

```
def Displaybooklist():
    if not books:
        print("No book in books")
        return
    print("Books List")
    for book in books:
        print(f"{book['ID']} | {book['title']} | {book['author']} | {book['publicationYear']} | {book['category']} | {book['availability']} | {book['count']} |")
```

## 1.5 Searchforbooks

```
def Searchforbooks():
    keyword = input("(ID or Category or Author or Year): ").lower()
    results = []
    for b in books:
        if (keyword == str(b['ID']) or keyword == b['category'].lower() or keyword == b['author'].lower() or keyword == str(b['publicationYear']) or keyword == b['title'].lower()):
            results.append(b)
    if results:
        print(f"Found {len(results)} result(s):")
        for b in results:
            print(f"{b['ID']} | {b['title']} | {b['author']} | {b['publicationYear']} | {b['category']} | {b['availability']} | {b['count']} |")
    else:
        print("No matching books found.")
```

## 2.1 borrowBook

```
def borrowBook():  1 usage
    while True:
        id = input("Enter Book ID to borrow: ")
        if id.isdigit():
            id = int(id)
            break
        else:
            print("Invalid input! Please enter a number for Book ID.")
    for book in books:
        if book['ID'] == id:
            if book['availability'] == 'available':
                book['availability'] = 'borrowed'
                book['count'] += 1
                print('Borrowed')
            else:
                print('Already borrowed')
            return
    print('Not Found')
```

## 2.2 returnBook

```
def returnBook():  1 usage
    while True:
        id = input("Enter Book ID to return: ")
        if id.isdigit():
            id = int(id)
            break
        else:
            print("Invalid input! Please enter a number for Book ID.")
    for book in books:
        if book['ID'] == id:
            if book['availability'] == 'borrowed':
                book['availability'] = 'available'
                print('Returned')
            else:
                print('Not borrowed')
            return
    print('Not Found')
```

2.3 listBorrowedBooks

```
def listBorrowedBooks():  1 usage
    check = False
    for book in books:
        if book['availability'] == 'borrowed':
            print(f"{book['ID']},{book['title']},{book['author']},{book['category']}")
            check = True
    if check == False:
        print('No books are current borrowed')
```

3.1 booksByCategory – Display on the number of books by category (e.g., Science, Literature, History…).

```
# 3.1 booksByCategory - Display the number of books by category (e.g., Science, Literature, Hist

def booksByCategory(books):
    check = False
    time = 0
    category = input('input category of book:')
    for i in books:
        if i ['category'].lower() == category.lower():
            print(f"ID: {i['ID']}, Title: {i['title']}, Availability: {i['availability']}")
            time += 1
            check = True

    if check:
        print (f"total book {category}:{time}")
    else:
        print (f"not found: {category}")
```

3.2 mostBorrowedBooks – Display a list of the most borrowed books, helping track the popularity of books.

```
#3.2 mostBorrowedBooks - Display a list of the most borrowed books, helping track the popularity

def mostBorrowedBooks():
    list_borrowed = []
    for book in books:
        if book['count'] > 0:
            list_borrowed.append(book)

    sorted_list= sorted(list_borrowed, key= lambda b: b['count'], reverse= True)
    for book in sorted_list:
        print(f" ID: {book['ID']}, Title: {book['title']}, Count: {book['count']}")
```

4.1 saveToFile

```python
#saveToFile
def saveToFile():
    filename = input("Enter name of file to save: ")

    try:
        with open(filename, "w", encoding="utf-8") as file:
            file.write("ID,title,author,category,publicationYear,availability,count\n")

            for book in books:
                line = f"{book['ID']},{book['title']},{book['author']}," \
                       f"{book['category']},{book['publicationYear']}," \
                       f"{book['availability']},{book['count']}\n"
                file.write(line)

        print("All books saved successfully!")
        return True

    except Exception as e:
        print(f"Error: Cannot save file ({e})")
        return False
```

4.2 loadFromFile

```python
#loadFromFile
def loadFromFile():
    filename = input("Enter name of file: ").strip()

    try:
        file = open(filename, 'r', encoding='utf-8')
    except FileNotFoundError:
        print(f"Error: cannot open file {filename}")

    for line in file:
        line = line.strip()
        if not line:
            continue

        try:
            parts = line.split(',')
            if len(parts) != 6:
                print(f"Invalid data format")
                continue
            books.append(parts)
        except Exception as e:
            print(f"Error reading line: {e}")

    file.close()
    print(f'Loaded {len(books)} books successfully!')
```

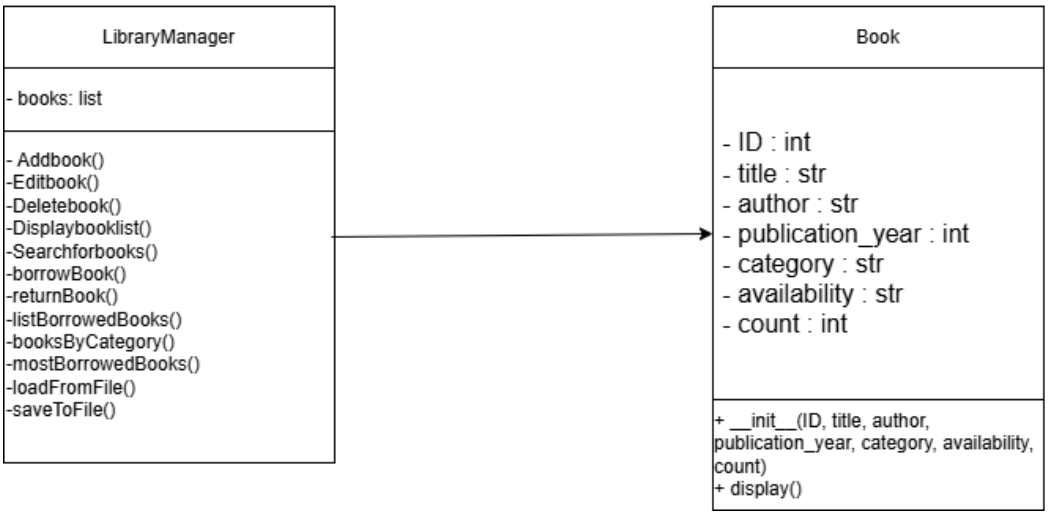
5. Control menu

```
while True:
    try:
        choice = int(input(
            'What do you want to do? Please select from the following options \n'
            '1 : Add book \n'
            '2 : Edit book \n'
            '3 : Delete book \n'
            '4 : Display all books \n'
            '5 : Search book \n'
            '6 : Borrow book \n'
            '7 : Return book \n'
            '8 : Display all borrowed book \n'
            '9 : Display books following the category \n'
            '10: Show most popular books in descending order \n'
            '11: Export data book to file .txt \n'
            '12: Import data book from file .txt \n'
            'Enter your selection: '
        ))
    except:
        print('Please enter number from 1 to 12')
        continue
    if choice not in range(1,13):
        print('Please enter number from 1 to 12')
        continue

    if choice == 1:
        Addbook()
    if choice == 2:
        Editbook()
    if choice == 3:
        Deletebook()
    if choice == 4:
        Displaybooklist()
    if choice == 5:
        Searchforbooks()
    if choice == 6:
        borrowBook()
    if choice == 7:
        returnBook()
    if choice == 8:
        listBorrowedBooks()
    if choice == 9:
        booksByCategory()
    if choice == 10:
        mostBorrowedBooks()
    if choice == 11:
        saveToFile()
    if choice == 12:
        loadFromFile()
    yesno = input('Do you want to continue? Yes/No ')
    if yesno.lower() == "yes":
        continue
    else:
        break

print('Goodbye')
```

## 5. Object-Oriented Design (OOP)




Class Book

```python
class Book:
    def __init__(self, ID, title, author, publication_year, category, availability, count):
        self._ID = ID
        self.title = title
        self.author = author
        self.publication_year = publication_year
        self.category = category
        self.availability = availability
        self.count = count

    def get_ID(self):
        return self._ID

    def set_ID(self, value):
        self._ID = value

    def get_title(self):
        return self.title

    def set_title(self, value):
        self.title = value

    def get_author(self):
        return self.author

    def set_author(self, value):
        self.author = value

    def get_publication_year(self):
        return self.publication_year

    def set_publication_year(self, value):
        self.publication_year = value

    def get_category(self):
        return self.category

    def set_category(self, value):
        self.category = value

    def get_availability(self):
        return self.availability

    def set_availability(self, value):
        self.availability = value

    def get_count(self):
        return self.count

    def set_count(self, value):
        self.count = value

    def display(self):
        print(f"{self._ID} | {self.title} | {self.author} | {self.publication_year} | "
              f"{self.category} | {self.availability} | {self.count} |")
```

Class LIbraryManager:

```
class LibraryManager:
    def __init__(self,books):
        self.books = books
```

## 1.1 Addbook

```
    def Addbook(self):
        while True:
            try:
                ID = int(input("Enter Book ID: "))
                break
            except ValueError:
                print("Invalid input! Please enter a number for Book ID.")
        for book in self.books:
            if book.get_ID() == ID:
                print(f"Book with ID {book.get_ID()} already exists! Skipping...")
                return
        author = input("Enter Author: ")
        while True:
            try:
                publication_year = int(input("Enter Publication Year: "))
                break
            except ValueError:
                print("Invalid input! Please enter a number for Publication Year.")
        category = input("Enter Category: ")
        title = input("Enter Title: ")
        while True:
            availability = input("Enter Availability (available/borrowed): ").lower()
            if availability in ["available", "borrowed"]:
                break
            else:
                print("Invalid input! Please enter only 'available' or 'borrowed'.")
        count = 0

        new_book = Book(ID, title, author, publication_year, category, availability, count)
        self.books.append(new_book)
        print("Book added successfully!")
```

## 1.2 Editbook

```
def Editbook(self):
    while True:
        ID = input("Enter Book ID to edit: ")
        if ID.isdigit():
            ID = int(ID)
            break
        else:
            print("Invalid input! Please enter a number for Book ID.")
    for book in self.books:
        if book.get_ID() == ID:
            print(f"Editing book: {book.title}")
            book.author = input("New Author: ")
            while True:
                try:
                    book.publication_year = int(input("New Publication Year: "))
                    break
                except ValueError:
                    print("Invalid input! Please enter a number for Publication Year.")
            book.category = input("New Category: ")
            book.title = input("New Title: ")
            while True:
                try:
                    book.count = int(input("New Count: "))
                    break
                except ValueError:
                    print("Invalid input! Please enter a number for Count.")
            while True:
                availability = input("Enter Availability (available/borrowed): ").strip().lower()
                if availability in ["available", "borrowed"]:
                    book.availability = availability
                    break
                else:
                    print("Invalid input! Please enter only 'available' or 'borrowed'.")
            print(" Book updated successfully!")
            return
    print(" Book not found!")
```

## 1.3 Deletebook

```
def Deletebook(self):
    while True:
        ID = input("Enter Book ID to delete: ")
        if ID.isdigit():
            ID = int(ID)
            break
        else:
            print("Invalid input! Please enter a number for Book ID.")
    for book in self.books:
        if book.get_ID() == ID:
            self.books.remove(book)
            print(" Book deleted successfully!")
            return
    print(" Book not found!")
```

1.4 Displaybooklist

```
def Displaybooklist(self):
    if not self.books:
        print(" No books available.")
        return
    self.books.sort(key = lambda b: b.get_ID())
    print("\n=== Books List ===")
    for book in self.books:
        book.display()
```

1.5 Searchforbooks

```
def Searchforbooks(self):
    keyword = input("Enter keyword (ID / Category / Author / Publication_Year / Title / Availability): ").lower()
    results = []
    for book in self.books:
        if (keyword == str(book.get_ID()) or
                keyword == book.category.lower() or
                keyword == book.author.lower() or
                keyword == str(book.publication_year) or
                keyword == book.title.lower() or
                keyword == book.availability.lower()):
            results.append(book)

    if results:
        print(f" Found {len(results)} result(s):")
        for book in results:
            book.display()
    else:
        print(" No matching books found.")
```

2.1 borrowBook

```
def borrowBook(self):
    while True:
        id = input("Enter Book ID to borrow: ")
        if id.isdigit():
            id = int(id)
            break
        else:
            print("Invalid input! Please enter a number for Book ID.")

    for book in self.books:
        if book.get_ID() == id:
            if book.availability == 'available':
                book.availability = 'borrowed'
                book.count += 1
                print('Borrowed successfully!')
            else:
                print('Already borrowed!')
            return

    print('Not Found')
```

2.2 returnBook

```
def returnBook(self):
    while True:
        id = input("Enter Book ID to return: ")
        if id.isdigit():
            id = int(id)
            break
        else:
            print("Invalid input! Please enter a number for Book ID.")

    for book in self.books:
        if book.get_ID() == id:
            if book.availability == 'borrowed':
                book.availability = 'available'
                print('Returned successfully!')
            else:
                print('Not borrowed!')
            return

    print('Not Found')
```

2.3 listBorrowedBooks

```
def listBorrowedBooks(self):
    check = False
    for book in self.books:
        if book.availability == 'borrowed':
            book.display()
            check = True

    if check == False:
        print('No books are current borrowed')
```

3.1 booksByCategory

```
def booksByCategory(self):
    category = input("Input category of book: ").lower()
    found_books = []
    for b in self.books:
        if b.category.lower() == category.lower():
            found_books.append(b)

    if not found_books:
        print(f"Not found: {category}")
        return

    print(f"Books in category '{category}':")
    for b in found_books:
        b.display()
    print(f"Total books in '{category.title()}': {len(found_books)}")
```

3.2 mostBorrowedBooks.

```
def mostBorrowedBooks(self):
    borrowed_books = []
    for b in self.books:
        if b.count > 0:
            borrowed_books.append(b)

    if not borrowed_books:
        print("No borrowed books found.")
        return

    sorted_books = sorted(borrowed_books, key=lambda b: b.count, reverse=True)

    print("Most Borrowed Books:")
    for b in sorted_books:
        b.display()
```

4.1 saveToFile

```
def saveToFile(self):
    filename = input("Enter name of file to save: ")

    try:
        with open(filename, "w", encoding="utf-8") as file:
            file.write("ID,title,author,category,publication_Year,availability,count\n")

            for b in self.books:
                line = f"{b.get_ID()},{b.title},{b.author}," \
                       f"{b.category},{b.publication_year},{b.availability},{b.count}\n"
                file.write(line)

        print("All books saved successfully!")
        return True

    except Exception as e:
        print(f"Error: Cannot save file ({e})")
        return False
```

4.2 loadFromFile

```
    def loadFromFile(self):
        filename = input('Enter name of file to load (e.g., book.txt): ').strip()
        try:
            dem = 0
            with open(filename, 'r', encoding='utf-8') as f:
                for line in f:
                    line = line.strip()
                    if not line:
                        continue
                    parts = line.split(',')
                    if len(parts) != 7:
                        print(f'Invalid line format: {line}')
                        continue
                    ID, title, author, year, category, available, count = parts
                    ID = int(ID)
                    available = available
                    year = int(year)
                    count = int(count)
                    book = Book(ID, title, author, year, category, available, count)
                    self.books.append(book)
                    dem += 1
            print(f'Loaded {dem} books from {filename}')
        except FileNotFoundError:
            print(f"Error: File '{filename}' not found.")
        except Exception as e:
            print(f'Error reading file: {e}')
```

5 Main

```
from Book import Book
from LibraryManager import LibraryManager

books = [
    Book(1, "Python Basics", "John Smith", 2020, "Programming", "available", 3),
    Book(2, "Data Structures", "Alice Brown", 2018, "Computer Science", "borrowed", 2),
    Book(3, "Network Security", "James Lee", 2022, "Cybersecurity", "borrowed", 0),
    Book(4, "Linear Algebra", "David Kim", 2019, "Mathematics", "available", 5),
]

if __name__ == "__main__":
    manager = LibraryManager(books)

    while True:
        try:
            choice = int(input(
                '====================== LIBRARY MANAGEMENT ======================\n'
                'What do you want to do? Please select from the following options \n'
                '1 : Add book \n'
                '2 : Edit book \n'
                '3 : Delete book \n'
                '4 : Display all books \n'
                '5 : Search book \n'
                '6 : Borrow book \n'
                '7 : Return book \n'
                '8 : Display all borrowed book \n'
                '9 : Display books following the category \n'
                '10: Show most popular books in descending order \n'
                '11: Export data book to file .txt \n'
                '12: Import data book from file .txt \n'
                '================================================================\n'
                'Enter your selection: '

            ))
        except:
            print('Please enter number from 1 to 12')
            continue
        if choice not in range(1, 13):
            print('Please enter number from 1 to 12')
            continue
```

```
        if choice == 1:
            manager.Addbook()
        if choice == 2:
            manager.Editbook()
        if choice == 3:
            manager.Deletebook()
        if choice == 4:
            manager.Displaybooklist()
        if choice == 5:
            manager.Searchforbooks()
        if choice == 6:
            manager.borrowBook()
        if choice == 7:
            manager.returnBook()
        if choice == 8:
            manager.listBorrowedBooks()
        if choice == 9:
            manager.booksByCategory()
        if choice == 10:
            manager.mostBorrowedBooks()
        if choice == 11:
            manager.saveToFile()
        if choice == 12:
            manager.loadFromFile()

        yesno = input('Do you want to continue? Yes/No ')
        if yesno.strip().lower() == "yes":
            continue
        else:
            break

    print('Goodbye')
```

## 6. File I/O & Testing

```
def saveToFile(self):
    filename = input("Enter name of file to save: ")

    try:
        with open(filename, "w", encoding="utf-8") as file:
            file.write("ID,title,author,category,publication_Year,availability,count\n")

            for b in self.books:
                line = f"{b.get_ID()},{b.title},{b.author}," \
                       f"{b.category},{b.publication_year},{b.availability},{b.count}\n"
                file.write(line)

        print("All books saved successfully!")
        return True

    except Exception as e:
        print(f"Error: Cannot save file ({e})")
        return False
```

```
   def loadFromFile(self):
       filename = input('Enter name of file to load (e.g., book.txt): ').strip()
       try:
           dem = 0
           with open(filename, 'r', encoding='utf-8') as f:
               for line in f:
                   line = line.strip()
                   if not line:
                       continue
                   parts = line.split(',')
                   if len(parts) != 7:
                       print(f'Invalid line format: {line}')
                       continue
                   ID, title, author, year, category, available, count = parts
                   ID = int(ID)
                   available = available
                   year = int(year)
                   count = int(count)
                   book = Book(ID, title, author, year, category, available, count)
                   self.books.append(book)
                   dem += 1
           print(f'Loaded {dem} books from {filename}')
       except FileNotFoundError:
           print(f"Error: File '{filename}' not found.")
       except Exception as e:
           print(f'Error reading file: {e}')
```

## 7. Experimental Results

```
====================== LIBRARY MANAGEMENT ======================
What do you want to do? Please select from the following options
1 : Add book
2 : Edit book
3 : Delete book
4 : Display all books
5 : Search book
6 : Borrow book
7 : Return book
8 : Display all borrowed book
9 : Display books following the category
10: Show most popular books in descending order
11: Export data book to file .txt
12: Import data book from file .txt
================================================================
Enter your selection:
```

## 8. Appendix

Book.py

LibraryManager.py

main.py

book.txt

https://github.com/NPT0306/PFP191-Project-Manager-Library