ASSIGNMENT NO.- 3

Library Inventory Manager – Project Report

NAME-ADITYA CHAUHAN

ROLL NO. – 2501730246

COURSE- BTECH CSE (AIML)

SECTION-C

SUBJECT- PROBLEM SOLVING USING PYTHON

FACULTY-SAMEER FAROOQ

Project Overview

The Library Inventory Manager is a lightweight, command-line

Python application designed to help campus libraries manage their

books efficiently. The program allows library staff to add, issue,

return, search, and view books in a catalog stored in a persistent

JSON file. The project demonstrates the use of Object-Oriented

Programming (OOP) principles, file handling, and modular

programming.

Objectives

- Design and implement a Book class with attributes and methods for book management.
- Create a LibraryInventory class to maintain and manipulate a collection of books.
- Implement JSON-based persistence to save and load the book catalog.
- Build a menu-driven CLI for user interaction.
- Ensure robust exception handling to

manage errors during file

operations and user input.

System Design

3.1 Classes

Book

- Attributes: title, author, isbn, status
- Methods:
    - issue() – Marks the book as issued.

- return_book() – Marks the book as available.
- to_dict() – Converts book details into a dictionary for JSON storage.
- __str__() – Returns a readable string representation of the book.

LibraryInventory

- Attributes: books (list of Book objects), filepath (JSON file path)
- Methods:
  - add_book(book) – Adds a new book to the inventory.
  - search_by_title(title) – Searches books by title keyword.
  - search_by_isbn(isbn) – Searches a book by ISBN.
  - display_all() – Returns all books in inventory.
  - save_data() – Saves the inventory to a JSON file.
  - load_data() – Loads the inventory from a JSON file.

Features

- Add Book: Add a new book with title, author, and ISBN.
- Issue Book: Mark a book as issued if available.
- Return Book: Mark an issued book as available.
- View All Books: Display all books with current status.
- Search Book: Search books by title keyword.
- Persistent Storage: JSON file stores all book records across sessions.

Technologies Used

- Python 3 – Core programming language.
- OOP Principles – Encapsulation, methods, and class-based design.
- JSON Module – For persistent storage of book records.
- Pathlib Module – For robust file path management.
- Command-Line Interface (CLI) – Interactive text-based user interface.

Implementation Highlights

- Single File Design: The entire program is implemented in a single, easy-to-understand Python file.
- Persistent Storage: Books are automatically saved in catalog.json.
- Robust Handling: The system handles missing or corrupted JSON files gracefully by creating a new catalog.
- User-Friendly: Clear menu and prompts guide the user through all operations.

How to Run

1. Ensure Python 3 is installed.
2. Save the program in a file named library_manager.py.
3. Run the program using the terminal:

```
python library_manager.py
```

4. Follow the menu prompts to manage the library books.

Sample Output

====== Library Inventory Manager ======

1. Add Book

2. Issue Book

3. Return Book

4. View All Books

5. Search Book

6. Exit

Enter choice: 1

Title: Python Basics

Author: John Doe

ISBN: 12345

Book added.

Enter choice: 4

Python Basics by John Doe (ISBN: 12345) — available

Challenges & Solutions

- File Not Found: Handled by creating a new JSON file if missing.
- Corrupted File: JSON decoding errors are caught and the catalog is reset.
- User Input Errors: Invalid menu choices are handled gracefully with prompts.

Conclusion

The Library Inventory Manager successfully demonstrates OOP

design, file handling, and CLI programming. It provides a practical
solution for small to medium libraries to maintain and track books efficiently. The modular structure allows for future ~~enhance~~ements,

such as GUI integration or multi-user support.

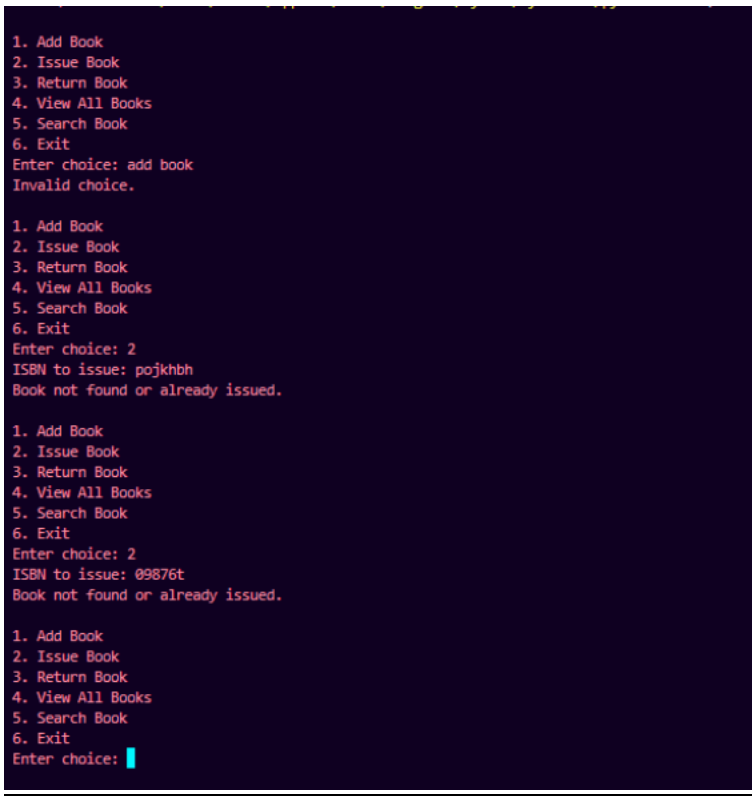
C

```python
import json
from pathlib import Path

class Book:
    def __init__(self, title, author, isbn, status="available"):
        self.title = title
        self.author = author
        self.isbn = isbn
        self.status = status

    def __str__(self):
        return f"{self.title} by {self.author} (ISBN: {self.isbn}) - {self.status}"

    def to_dict(self):
        return {"title": self.title, "author": self.author, "isbn": self.isbn, "status": self.status}

    def issue(self):
        if self.status == "available":
            self.status = "issued"
            return True
        return False

    def return_book(self):
        if self.status == "issued":
            self.status = "available"
            return True
        return False

class LibraryInventory:
    def __init__(self, filepath="catalog.json"):
        self.filepath = Path(filepath)
        self.books = []
        self.load_data()

    def add_book(self, book):
        self.books.append(book)
        self.save_data()

    def search_by_title(self, title):
        return [b for b in self.books if title.lower() in b.title.lower()]

    def search_by_isbn(self, isbn):
        for book in self.books:
            if book.isbn == isbn:
                return book
        return None

    def display_all(self):
        return self.books

    def save_data(self):
        data = [b.to_dict() for b in self.books]
        with open(self.filepath, "w") as f:
            json.dump(data, f, indent=4)
```

```
    def load_data(self):
        if not self.filepath.exists():
            self.save_data()
        try:
            with open(self.filepath, "r") as f:
                data = json.load(f)
            self.books = [Book(**d) for d in data]
        except:
            self.books = []

def menu():
    print("\n1. Add Book")
    print("2. Issue Book")
    print("3. Return Book")
    print("4. View All Books")
    print("5. Search Book")
    print("6. Exit")

def main():
    inventory = LibraryInventory()
    while True:
        menu()
        choice = input("Enter choice: ")
        if choice == "1":
            title = input("Title: ")
            author = input("Author: ")
            isbn = input("ISBN: ")
            inventory.add_book(Book(title, author, isbn))
            print("Book added.")
        elif choice == "2":
            isbn = input("ISBN to issue: ")
            book = inventory.search_by_isbn(isbn)
            if book and book.issue():
                print("Book issued.")
                inventory.save_data()
            else:
                print("Book not found or already issued.")
        elif choice == "3":
            isbn = input("ISBN to return: ")
            book = inventory.search_by_isbn(isbn)
            if book and book.return_book():
                print("Book returned.")
                inventory.save_data()
            else:
                print("Book not found or not issued.")
        elif choice == "4":
            for b in inventory.display_all():
                print(b)
        elif choice == "5":
            key = input("Enter title keyword: ")
            results = inventory.search_by_title(key)
            if results:
                for b in results:
```

```
                for b in results:
                    print(b)
            else:
                print("No books found.")
        elif choice == "6":
            print("Exiting...")
            break
        else:
            print("Invalid choice.")

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

OUTPUT:-


1. Add Book
2. Issue Book
3. Return Book
4. View All Books
5. Search Book
6. Exit
Enter choice: add book
Invalid choice.
1. Add Book
2. Issue Book
3. Return Book
4. View All Books
5. Search Book
6. Exit
Enter choice: 2
ISBN to issue: pojkhbh
Book not found or already issued.
1. Add Book
2. Issue Book
3. Return Book
4. View All Books
5. Search Book
6. Exit
Enter choice: 2
ISBN to issue: 09876t
Book not found or already issued.
1. Add Book
2. Issue Book
3. Return Book
4. View All Books
5. Search Book
6. Exit
Enter choice: