# K.R. Mangalam University

## School of Engineering & Technology

---

**Course Code:** ETCCPP102

**Course Name: PROGRAMMING FOR PROBLEM SOLVING USING PYTHON**

**Course:** B.Tech. CSE (Specialization in AI & ML)

**Semester:** 1

---

# Library Management System – Detailed Project Report

**Submitted To: MR. Sameer Farooqi**

---

**Submitted By: Prachi Manwal**
**Roll No:** 2501730365

**Section:** 'B'

## 1. Introduction

This project presents a fully functional Library Management System built using Python. It demonstrates Object-Oriented Programming (OOP), JSON file persistence, exception handling,

logging, and modular project architecture. The system allows library staff or students to manage books interactively through a command-line interface.

The primary goal of the system is to automate common library operations such as:

• Adding new books

• Issuing books

• Returning books

• Viewing inventory

• Searching books by title or ISBN

## 2. Objectives

The main objectives of the Library Management System are:

• Apply OOP concepts to represent books and inventory

• Implement file handling using JSON

• Create an interactive menu-driven CLI

• Add robust error handling

• Use logging to track system operations

• Organize files into a professional, GitHub-ready package

## 3. System Architecture

The system is organized into separate modules for clarity:

• book.py – Contains the Book class

• inventory.py – Handles the collection of books and JSON operations

• main.py – Command-line interface

This modular design follows clean coding principles and makes the project scalable.

## 4. Class Diagram (Textual Representation)

Book Class:

---

Attributes:

• title

• author

• isbn

• status

Methods:

• issue()

• return_book()

• is_available()

• to_dict()

LibraryInventory Class:

---

Attributes:

• books (list of Book objects)

Methods:

• add_book()

• search_by_title()

• search_by_isbn()

• save_to_file()

• load_from_file()

• display_all()

## 5. JSON File Persistence

The system uses JSON to save and load book data. JSON is chosen because:

• It's lightweight

• Human-readable

• Easy for Python's json module to handle

Missing or corrupted files are handled using try–except blocks.

## 6. Logging System

The logging module records:

• INFO logs for successful operations

• ERROR logs for failures such as missing files or invalid inputs

This helps in debugging and maintaining a usage history.

## 7. Detailed Code Explanation

The Book class manages book data and status. It validates operations (e.g., cannot issue an already issued book). The LibraryInventory class holds a list of Book objects and interacts with the JSON file. The CLI in main.py presents a menu for user operations.

## 8. Sample Outputs / Screenshots (Text Mode)

Example Menu:

---

1. Add Book

2. Issue Book

3. Return Book

4. View All Books

5. Search Book

6. Exit

Example Book Entry:

Enter title: Atomic Habits

Enter author: James Clear

Enter ISBN: 12345

Book added successfully!

## 9. Future Scope

The project can be extended with:

• SQLite/MySQL database instead of JSON

• Tkinter or PyQt5 graphical interface

• Barcode scanning for ISBN

• Cloud-based library syncing

• User login and role-based access

## 10. Conclusion

This Library Management System successfully demonstrates practical use of Python OOP, file handling, logging, and modular design. It is scalable, easy to extend, and structured as a professional GitHub repository.

# Appendix A – Full Python Source Code

## book.py

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

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

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

    def is_available(self):
        return self.status == "available"

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

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

## inventory.py

```
import json
```

```
from pathlib import Path
from .book import Book

class LibraryInventory:
    def __init__(self, file_path="books.json"):
        self.file_path = Path(file_path)
        self.books = []
        self.load_from_file()

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

    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 b in self.books:
            if b.isbn == isbn:
                return b
        return None

    def display_all(self):
        return [str(b) for b in self.books]

    def save_to_file(self):
        try:
            data = [b.to_dict() for b in self.books]
            self.file_path.write_text(json.dumps(data, indent=4))
        except Exception as e:
            print("Error saving file:", e)

    def load_from_file(self):
        try:
            if self.file_path.exists():
                data = json.loads(self.file_path.read_text())
                self.books = [Book(**entry) for entry in data]
        except Exception as e:
            print("Error loading file:", e)
```

## main.py

```
from library_manager.inventory import LibraryInventory
from library_manager.book import Book

def menu():
    print("\nLibrary Management System")
    print("1. 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("Enter ISBN to issue: ")
            book = inventory.search_by_isbn(isbn)
            if book and book.issue():
                inventory.save_to_file()
                print("Book issued.")
            else:
                print("Book not available or invalid.")

        elif choice == "3":
            isbn = input("Enter ISBN to return: ")
            book = inventory.search_by_isbn(isbn)
            if book and book.return_book():
```

```
            inventory.save_to_file()
            print("Book returned.")
        else:
            print("Invalid operation.")

    elif choice == "4":
        for b in inventory.display_all():
            print(b)

    elif choice == "5":
        title = input("Enter title: ")
        results = inventory.search_by_title(title)
        for r in results:
            print(r)

    elif choice == "6":
        break

    else:
        print("Invalid choice.")

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

## GitHub Repository link : https://github.com/prachiii0418-star/library_inventory