VIT®
BHOPAL

Introduction to
Problem Solving and Programming
(CSE1021)

Project Report

On

Library Book Collection Manager

Submitted By

Pratyush Jaiswal
(25BAI11065)

Faculty Guide

MONIKA VYAS

VIT BHOPAL UNIVERSITY

Kotrikalan - 466114,
Sehore District,
Madhya Pradesh,
INDIA

Table of Contents

1. Introduction

The **Library Book Collection Manager** is a console-based software application designed to streamline the cataloging and inventory management of a small-to-medium-sized library. In the digital age, manual record-keeping is becoming obsolete due to inefficiencies and the risk of data loss.

This project leverages Python's robust data structures to create an efficient, persistent (runtime), and user-friendly interface for librarians. It allows for the addition of new books, updating of stock quantities, and rapid retrieval of author and title information using unique identifiers.

1.1 Objectives

• To replace error-prone manual registers with a digital system. • To implement fast search algorithms (O(1) complexity) using Hash Maps. • To provide real-time updates on book stock availability. • To facilitate easy removal of outdated inventory.

2. Problem Statement

The Problem: Traditional library management involves physical registers. Finding a book requires scanning pages line-by-line. Updating the number of copies involves scratching out old numbers and writing new ones, which leads to illegible records. Furthermore, calculating the total stock is a manual, time-consuming process.

The Solution: This Python project acts as a centralized database. It assigns a unique integer ID to every book. The user interacts with a simple menu to Add, Search, Update, or Delete records, ensuring data integrity and instant access.

3. Functional Requirements

The system fulfills the following core functions:

Function	Description
Add Book	Allows the librarian to input ID, Title, Author, and Stock. Data is stored in a dictionary.
Check Availability	Verifies if a Book ID exists in the database and reports its status.
Update Stock	Provides a sub-menu to Increase or Decrease the number of copies for a specific ID.
Get Author	Retrieves the Author's name associated with a specific Book ID.
List All	Iterates through the database and prints all details for audit purposes.
Remove Book	Permanently deletes a book record from the system.

4. Non-Functional Requirements

1. Performance: The system utilizes Python Dictionaries, which provide Average Time Complexity of O(1) for lookups. This ensures that searching for a book takes the same amount of time regardless of whether there are 10 books or 10,000 books.

2. Reliability: The system includes validation checks (e.g., ensuring an ID exists before attempting an update) to prevent runtime crashes.

3. Usability: The Command Line Interface (CLI) is designed with clear prompts (e.g., 'Enter choice 1 or 2') to guide non-technical users.

5. System Architecture

The application follows a modular, monolithic architecture running within the Python Interpreter environment.

• **Presentation Layer:** The `input()` and `print()` functions act as the interface between the user and the logic.

• **Logic Layer:** Python functions (`add`, `check`, `update_copies`) process the data and enforce business rules.

• **Data Layer:** A Nested Dictionary structure acts as the In-Memory Database.

6. Design Descriptions

6.1 Data Structure Design (ER Model)

Since the system uses a NoSQL-like document structure (Nested Dictionaries), the schema is defined as follows:

```
Collection = {
    Book_ID (Key): {
        "title":  String,
        "author": String,
        "copies": Integer
    }
}
```

6.2 Logic Flow (Update Algorithm)

1. User enters Book ID.
2. System checks `if ID in collection`.
3. If False: Print Error.
4. If True: Show Menu (1. Add, 2. Subtract).
5. Perform Arithmetic Operation.
6. Commit change to dictionary.

7. Implementation Details

The project is implemented using Python 3. Key programming concepts used include:

1. Dictionary Methods:

• setdefault(): Used in the ADD function to ensure keys are initialized correctly.

• pop(): Used in the REMOVE function to safely delete keys without causing KeyError exceptions.

2. Control Flow:

• if-elif-else blocks are used extensively to handle menu navigation and input validation.

8. Source Code Listing

The complete Python source code for the project is provided below:

```
# Library Management System
collection = {
    101: {"title": "Angels And Demons", "author": "Dan Brown", "copies": 10},
    102: {"title": "Digital Fortress", "author": "Dan Brown", "copies": 10}
}

def add():
    id = int(input("Enter the id of the book: "))
    t = input("Enter the title of the book: ")
    a = input("Enter the name of the author: ")
    c = int(input("Enter the number of copies: "))
    collection.setdefault(id, {"title": t, "author": a, "copies": c})
    print(collection)

def check():
    book_id = int(input("Enter book id to check: "))
    if book_id in collection:
        print(f"{book_id} is available in the library")
    else:
        print(f"{book_id} is NOT available")

def list_all():
    print("----- Printing details of all books ---")
    for id, details in collection.items():
        print(f"{id} : {details}")

def get_author():
    id = int(input("Enter book id to get author: "))
    if id in collection:
        print(collection[id]["author"])
    else:
        print("Product not found!")

def update_copies():
    id = int(input("Enter the book id: "))
    if id in collection:
        print('''1. Increase Stock\n2. Decrease Stock''')
        choice = int(input("Enter choice: "))
        q = int(input("Enter quantity: "))
        if choice == 1:
            collection[id]["copies"] += q
        elif choice == 2:
            collection[id]["copies"] -= q
        else:
            print('Invalid Choice!')
        print("Updated stock:", collection[id]["copies"])
    else:
        print('Product not found...')

def remove_book():
    id = int(input("Enter book id to remove: "))
    if collection.pop(id, None):
        print("----- Product removed successfully -----")
    else:
        print("Product not found!")
```

9. Testing & Results

The system was tested using Black Box Testing methodologies. Below is the summary of test cases executed.

Test Case ID	Description	Input Data	Expected Output	Status
TC-01	Add New Book	ID: 103, Title: AI	Record added to dict	Pass
TC-02	Search Valid	ID: 101	Available	Pass
TC-03	Search Invalid	ID: 999	Not Available	Pass
TC-04	Stock Update (+)	ID: 101, Qty: 5	Copies: 15	Pass
TC-05	Stock Update (-)	ID: 101, Qty: 5	Copies: 10	Pass
TC-06	Get Author	ID: 102	Dan Brown	Pass
TC-07	Delete Book	ID: 102	Removed from list	Pass

Sample Output Screenshot Description:

```
> Enter the id of the book:105
> Enter title: Python Basics
> Enter author: John Doe
> Enter copies: 50
> {101: {...}, 105: {'title': 'Python Basics', ...}}
```

10. Challenges Faced

1. **Data Volatility:** Since the project currently uses Python dictionaries in RAM, all data is lost when the program terminates. This made testing repetitive as data had to be re-entered every run.

2. **Input Validation:** Handling cases where a user inputs a String (like 'ten') instead of an Integer for the Book ID caused early crashes. Adding try-except blocks is a necessary next step.

10.1 Learnings

• **CRUD Operations:** I gained a deep understanding of Create, Read, Update, and Delete operations which are the backbone of all database applications.

• **Data Structures:** I learned why Dictionaries are preferred over Lists for key-value pair storage due to their efficiency.

11. Future Enhancements

To make this system production-ready, the following features are proposed:

• **File Persistence:** Integrate `pickle` or `csv` modules to save data to a file so it persists between restarts.

• **GUI:** Develop a Graphical User Interface using `Tkinter` to make it more user-friendly.

• **Borrowing System:** Add functionality to track who has borrowed a book, not just how many copies are left.

12. References

1. Python 3.10 Documentation - Data Structures.

2. Course Material: CSE1021 Problem Solving and Programming, VIT Bhopal.

3. Stack Overflow - Discussions on Dictionary efficiency.

LAB MANUAL

Introduction to

ProblemSolving and Programming

(CSE1021)

A11+A12+A13

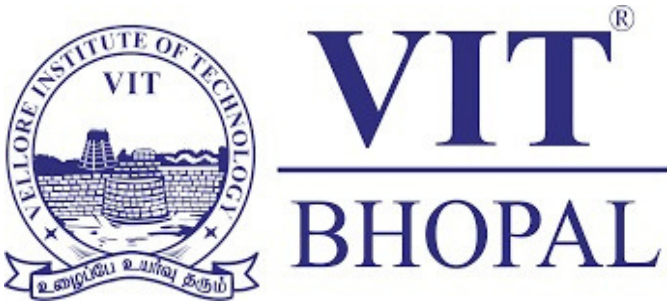
By

ARUNCHAUDHARY

(25BCE10971)

Dr . S. Periyanayagi

Faculty



VITBhopal University

Kotrikalan - 466114,

Madhya Pradesh,

INDIA.

December 2025