"Python Programming"

Assignment-3

Topic – Object-Oriented Design and Robust Programming in a Library Management System

Submitted by – Priyam Sharma

Roll no - 2501730184

Course – B. Tech CSE (AI & ML)

Section – B

Faculty name- Mr Sameer Farooq

Introduction

This project is a simple command-line Library Inventory Manager made using Python. It helps manage books by storing their details, tracking issued/available status, and saving all data in a JSON file. The program uses object-oriented programming, file handling, and basic error management to create a small but functional library system.

Objectives

-To design classes using OOP concepts.

-To store and load data using JSON.

-To build a menu-driven CLI for library operations.

-To apply exception handling and logging.

-To create a simple, real-world library management solution.

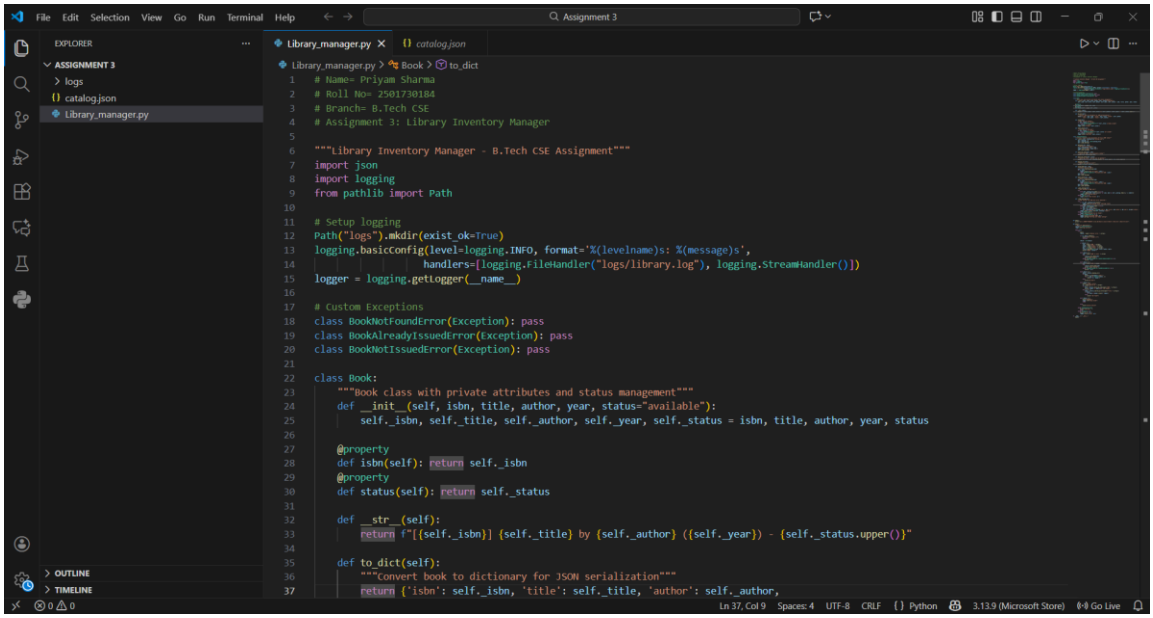
Program Description

The program contains two classes:
Book – stores title, author, ISBN, and status (issued/available).
LibraryInventory – manages a list of books, adds books, searches, issues, returns, and saves data to books.json.

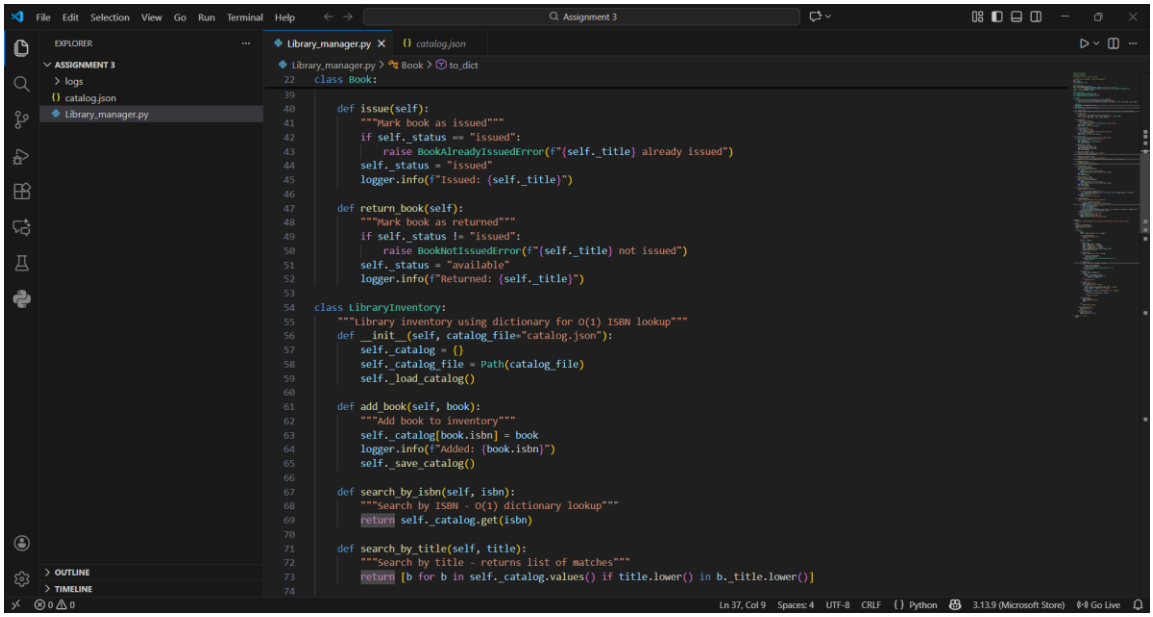
A menu interface allows the user to add, issue, return, view, and search books. JSON is used for permanent storage, and

logging/exception handling makes the program reliable. All features are implemented inside one single Python file.

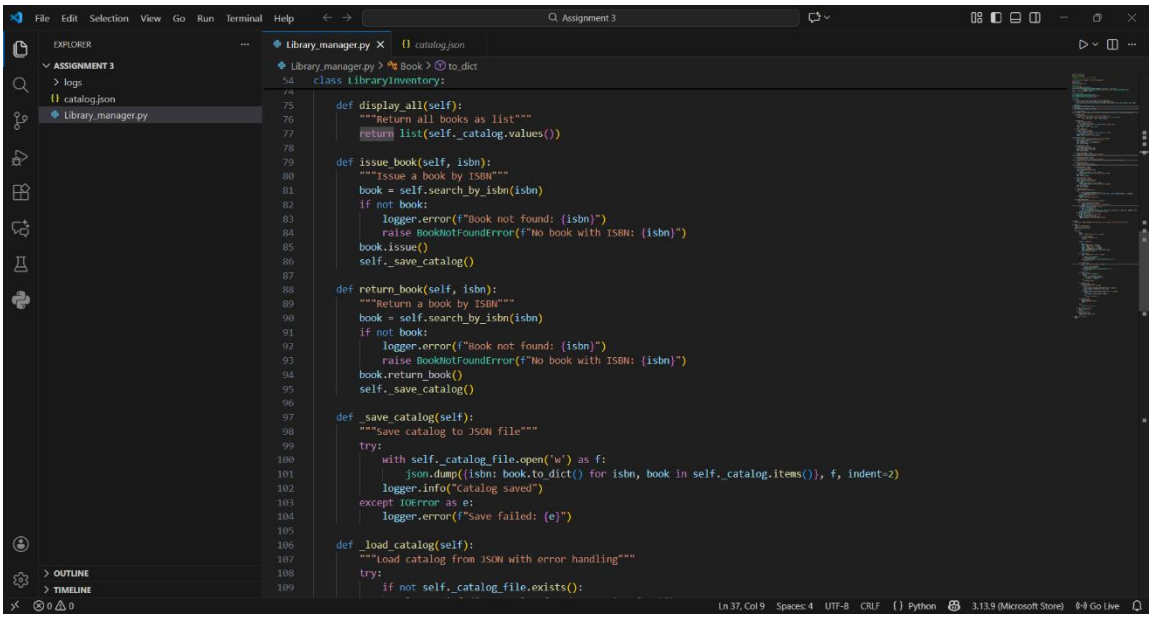
Program Code



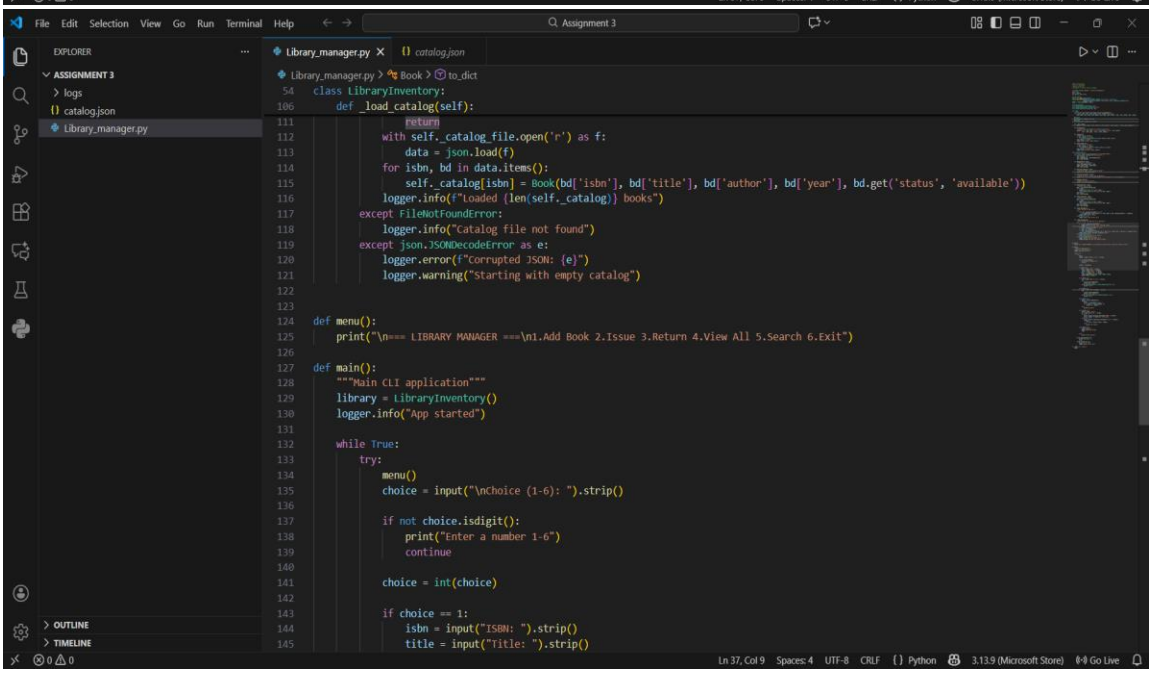
```python
# Name= Priyam Sharma
# Roll No= 2501730184
# Branch= B.Tech CSE
# Assignment 3: Library Inventory Manager

"""Library Inventory Manager - B.Tech CSE Assignment"""
import json
import logging
from pathlib import Path

# Setup logging
Path("logs").mkdir(exist_ok=True)
logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s',
                    handlers=[logging.FileHandler("logs/library.log"), logging.StreamHandler()])
logger = logging.getLogger(__name__)

# Custom Exceptions
class BookNotFoundError(Exception): pass
class BookAlreadyIssuedError(Exception): pass
class BookNotIssuedError(Exception): pass

class Book:
    """Book class with private attributes and status management"""
    def __init__(self, isbn, title, author, year, status="available"):
        self._isbn, self._title, self._author, self._year, self._status = isbn, title, author, year, status

    @property
    def isbn(self): return self._isbn
    @property
    def status(self): return self._status

    def __str__(self):
        return f"[{self._isbn}] {self._title} by {self._author} ({self._year}) - {self._status.upper()}"

    def to_dict(self):
        """Convert book to dictionary for JSON serialization"""
        return {'isbn': self._isbn, 'title': self._title, 'author': self._author,
```



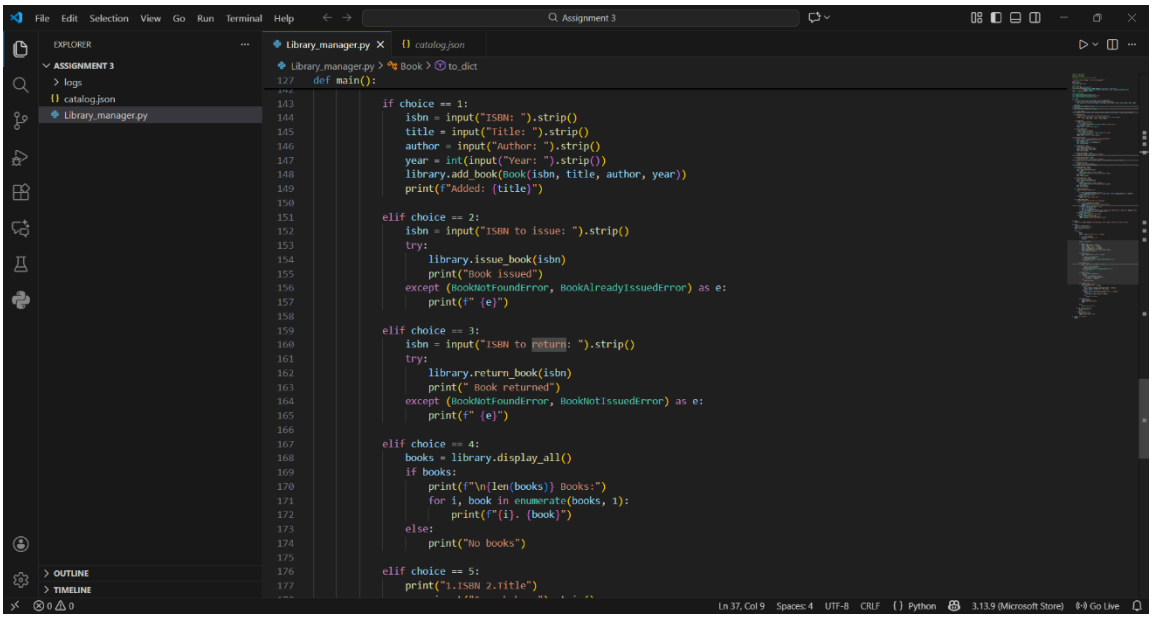
```python
class Book:

    def issue(self):
        """Mark book as issued"""
        if self._status == "issued":
            raise BookAlreadyIssuedError(f"{self._title} already issued")
        self._status = "issued"
        logger.info(f"Issued: {self._title}")

    def return_book(self):
        """Mark book as returned"""
        if self._status != "issued":
            raise BookNotIssuedError(f"{self._title} not issued")
        self._status = "available"
        logger.info(f"Returned: {self._title}")

class LibraryInventory:
    """Library inventory using dictionary for O(1) ISBN lookup"""
    def __init__(self, catalog_file="catalog.json"):
        self._catalog = {}
        self._catalog_file = Path(catalog_file)
        self._load_catalog()

    def add_book(self, book):
        """Add book to inventory"""
        self._catalog[book.isbn] = book
        logger.info(f"Added: {book.isbn}")
        self._save_catalog()

    def search_by_isbn(self, isbn):
        """Search by ISBN - O(1) dictionary lookup"""
        return self._catalog.get(isbn)

    def search_by_title(self, title):
        """Search by title - returns list of matches"""
        return [b for b in self._catalog.values() if title.lower() in b._title.lower()]
```

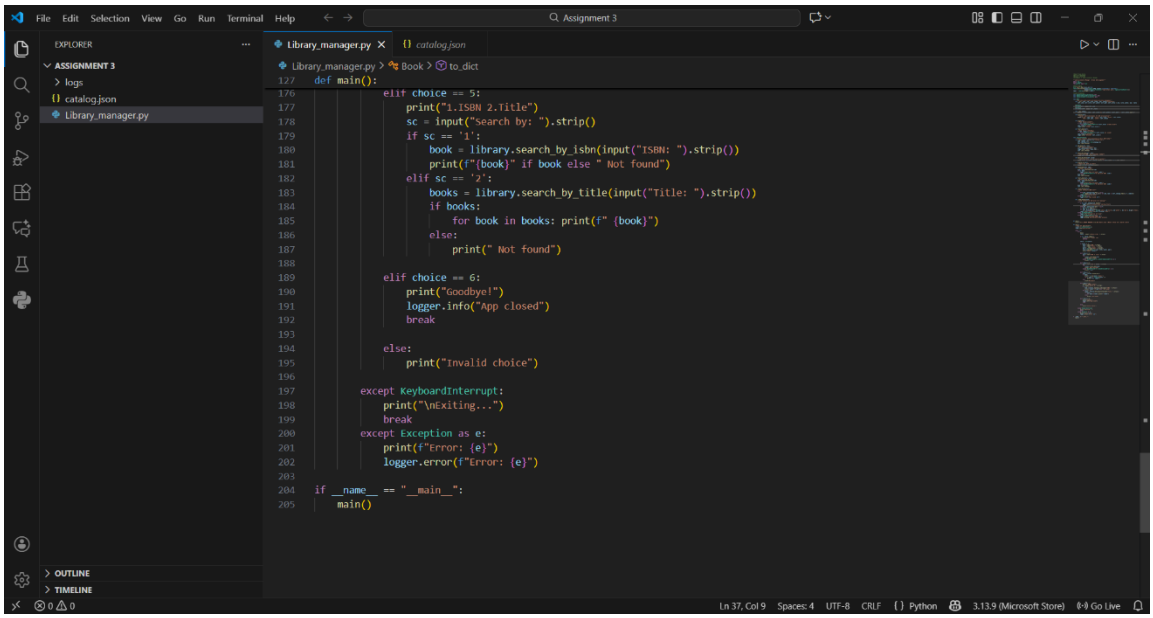


```
    def display_all(self):
        """Return all books as list"""
        return list(self._catalog.values())

    def issue_book(self, isbn):
        """Issue a book by ISBN"""
        book = self.search_by_isbn(isbn)
        if not book:
            logger.error(f"Book not found: {isbn}")
            raise BookNotFoundError(f"No book with ISBN: {isbn}")
        book.issue()
        self._save_catalog()

    def return_book(self, isbn):
        """Return a book by ISBN"""
        book = self.search_by_isbn(isbn)
        if not book:
            logger.error(f"Book not found: {isbn}")
            raise BookNotFoundError(f"No book with ISBN: {isbn}")
        book.return_book()
        self._save_catalog()

    def _save_catalog(self):
        """Save catalog to JSON file"""
        try:
            with self._catalog_file.open('w') as f:
                json.dump({isbn: book.to_dict() for isbn, book in self._catalog.items()}, f, indent=2)
            logger.info("Catalog saved")
        except IOError as e:
            logger.error(f"Save failed: {e}")

    def _load_catalog(self):
        """Load catalog from JSON with error handling"""
        try:
            if not self._catalog_file.exists():
```

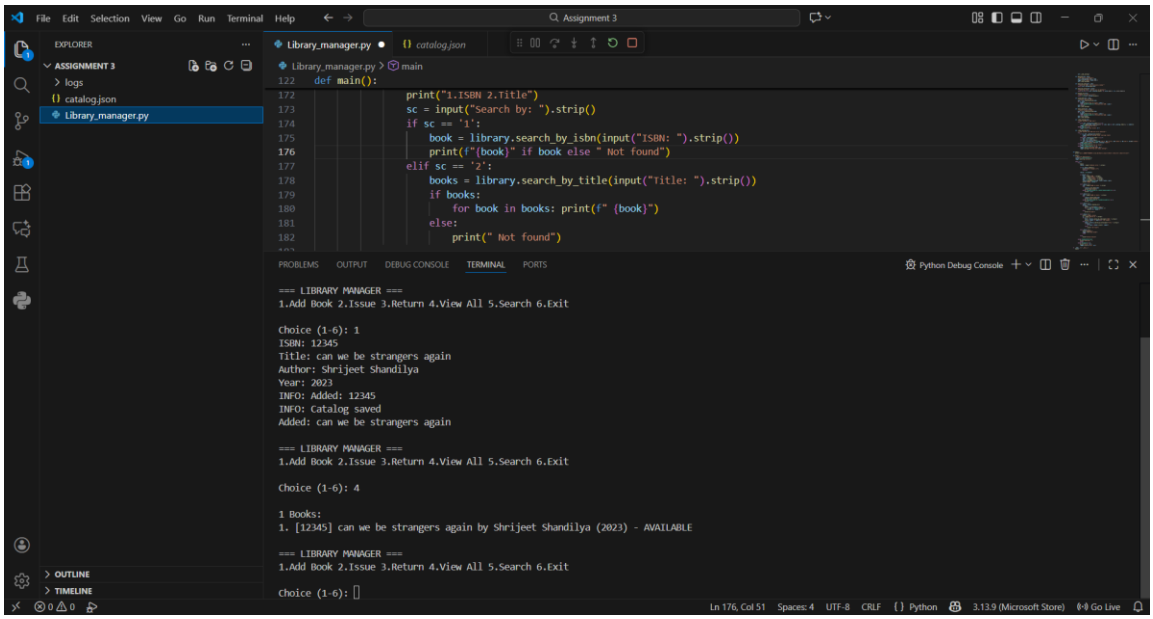


```
    def _load_catalog(self):
                return
            with self._catalog_file.open('r') as f:
                data = json.load(f)
            for isbn, bd in data.items():
                self._catalog[isbn] = Book(bd['isbn'], bd['title'], bd['author'], bd['year'], bd.get('status', 'available'))
            logger.info(f"Loaded {len(self._catalog)} books")
        except FileNotFoundError:
            logger.info("Catalog file not found")
        except json.JSONDecodeError as e:
            logger.error(f"Corrupted JSON: {e}")
            logger.warning("Starting with empty catalog")


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

def main():
    """Main CLI application"""
    library = LibraryInventory()
    logger.info("App started")

    while True:
        try:
            menu()
            choice = input("\nChoice (1-6): ").strip()

            if not choice.isdigit():
                print("Enter a number 1-6")
                continue

            choice = int(choice)

            if choice == 1:
                isbn = input("ISBN: ").strip()
                title = input("Title: ").strip()
```


File Edit Selection View Go Run Terminal Help
Assignment 3
EXPLORER
ASSIGNMENT 3
logs
catalog.json
Library_manager.py
Library_manager.py
catalog.json
Library_manager.py > Book > to_dict
def main():
if choice == 1:
isbn = input("ISBN: ").strip()
title = input("Title: ").strip()
author = input("Author: ").strip()
year = int(input("Year: ").strip())
library.add_book(Book(isbn, title, author, year))
print(f"Added: {title}")
elif choice == 2:
isbn = input("ISBN to issue: ").strip()
try:
library.issue_book(isbn)
print("Book issued")
except (BookNotFoundError, BookAlreadyIssuedError) as e:
print(f" {e}")
elif choice == 3:
isbn = input("ISBN to return: ").strip()
try:
library.return_book(isbn)
print(" Book returned")
except (BookNotFoundError, BookNotIssuedError) as e:
print(f" {e}")
elif choice == 4:
books = library.display_all()
if books:
print(f"\n{len(books)} Books:")
for i, book in enumerate(books, 1):
print(f"{i}. {book}")
else:
print("No books")
elif choice == 5:
print("1.ISBN 2.Title")
OUTLINE
TIMELINE
Ln 37, Col 9 Spaces: 4 UTF-8 CRLF Python 3.13.9 (Microsoft Store) Go Live


File Edit Selection View Go Run Terminal Help
Assignment 3
EXPLORER
ASSIGNMENT 3
logs
catalog.json
Library_manager.py
Library_manager.py
catalog.json
Library_manager.py > Book > to_dict
def main():
elif choice == 5:
print("1.ISBN 2.Title")
sc = input("Search by: ").strip()
if sc == '1':
book = library.search_by_isbn(input("ISBN: ").strip())
print(f"{book}" if book else " Not found")
elif sc == '2':
books = library.search_by_title(input("Title: ").strip())
if books:
for book in books: print(f" {book}")
else:
print(" Not found")
elif choice == 6:
print("Goodbye!")
logger.info("App closed")
break
else:
print("Invalid choice")
except KeyboardInterrupt:
print("\nExiting...")
break
except Exception as e:
print(f"Error: {e}")
logger.error(f"Error: {e}")
if __name__ == "__main__":
main()
OUTLINE
TIMELINE
Ln 37, Col 9 Spaces: 4 UTF-8 CRLF Python 3.13.9 (Microsoft Store) Go Live

Sample Output



Conclusion

This project successfully demonstrates OOP, JSON file handling, CLI design, and error management in Python. It provides a simple and efficient way to manage a small library system and fulfills all assignment requirements.