Table of Contents

1.Introduction
2. Objectives
3. System Overview
4. Data Structures
5. Functions
6. UML Diagram
7. Testing
8. Demo Script
9. Conclusion

Introduction

The Mini Library Management System demonstrates practical use of Python data structures and functions. It allows adding, searching, updating, deleting, borrowing, and returning books.

Objectives

- Implement book management using a dictionary keyed by ISBN.
- Manage members using a list of dictionaries.
- Use a tuple for a fixed set of valid genres.
- Provide CRUD operations and borrowing logic with constraints (max 3 books per member).

System Overview

Books are stored in a dictionary keyed by ISBN. Members are stored in a list of dictionaries with borrowed_books as a list. GENRES is a tuple to prevent modification.

Data Structures

Dictionary (Books), List (Members), Tuple (Genres) with code snippets:

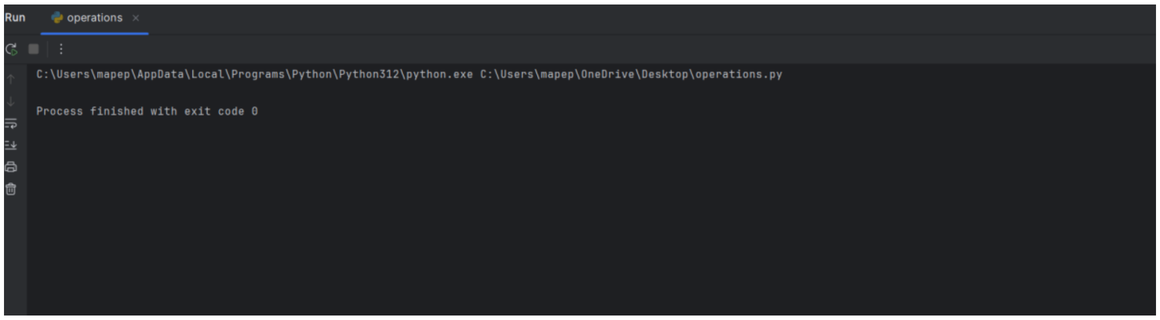
operations.py — Data Structures

```
GENRES = ("Fiction", "Non-Fiction", "Sci-Fi", "Romance", "Mystery")

books = {}

members = []

def find_member(member_id):

    for m in members:

        if m["member_id"] == member_id:

            return m

    return None

def add_book(isbn, title, author, genre, total_copies):

    if isbn in books:
        return "Book already exists."
    if genre not in GENRES:
        return "Invalid genre."
    books[isbn] = {
        "title": title,
        "author": author,
        "genre": genre,
        "total_copies": total_copies
    }
    return "Book added successfully."
def update_book(isbn, title=None, author=None, genre=None,
total_copies=None):
    if isbn not in books:
        return "Book not found."
    if genre and genre not in GENRES:
        return "Invalid genre."
    if title:
        books[isbn]["title"] = title
    if author:
        books[isbn]["author"] = author
    if genre:
        books[isbn]["genre"] = genre
    if total_copies is not None:
        books[isbn]["total_copies"] = total_copies
    return "Book updated successfully."
```

```
def delete_book(isbn):

    if isbn not in books:

        return "Book not found."

             for m in members:

        if isbn in m["borrowed_books"]:

            return "Cannot delete — book is currently borrowed."

    del books[isbn]

    return "Book deleted successfully."

def search_books(keyword):

         results = []

    for isbn, info in books.items():

        if keyword.lower() in info["title"].lower() or keyword.lower() in
info["author"].lower():

            results.append((isbn, info))

    return results or "No matches found."

def add_member(member_id, name, email):

         if find_member(member_id):

        return "Member already exists."

    members.append({

 "member_id": member_id,

        "name": name,

        "email": email,

        "borrowed_books": []

    })

    return "Member added successfully."
```

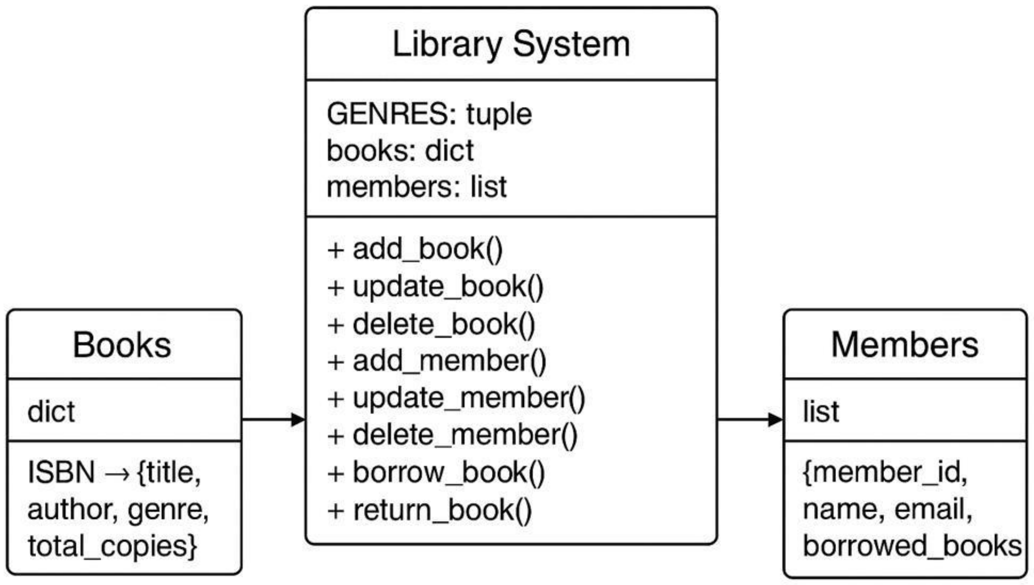
```
def update_member(member_id, name=None, email=None):
         m = find_member(member_id)
    if not m:
        return "Member not found."
    if name:
        m["name"] = name
    if email:
        m["email"] = email
    return "Member updated successfully."
def delete_member(member_id):
        m = find_member(member_id)
    if not m:
        return "Member not found."
    if m["borrowed_books"]:
        return "Cannot delete — member still has borrowed books."
    members.remove(m)
  return "Member deleted successfully."
def borrow_book(member_id, isbn):
      m = find_member(member_id)
    if not m:
        return "Member not found."
    if isbn not in books:
        return "Book not found."
    if len(m["borrowed_books"]) >= 3:
        return "Borrow limit reached (max 3)."
    if books[isbn]["total_copies"] <= 0:
        return "No copies available."
    books[isbn]["total_copies"] -= 1
    m["borrowed_books"].append(isbn)
    return f"{books[isbn]['title']} borrowed successfully."
def return_book(member_id, isbn):
      m = find_member(member_id)
    if not m:
        return "Member not found."
    if isbn not in m["borrowed_books"]:
        return "Book not borrowed by member."
    m["borrowed_books"].remove(isbn)
    books[isbn]["total_copies"] += 1
 return f"{books[isbn]['title']} returned successfully."
```

Terminal debug:



```
C:\Users\mapep\AppData\Local\Programs\Python\Python312\python.exe C:\Users\mapep\OneDrive\Desktop\operations.py

Process finished with exit code 0
```

UML Diagram



Testing

Unit tests are implemented in tests.py to verify correctness.

tests.py — Unit Tests

```
from operations import *

books.clear()

members.clear()

assert add_book("001", "OOP", "John Kargbo", "Non-Fiction", 3) == "Book added successfully."

assert add_book("001", "Duplicate", "Author", "Fiction", 2) == "Book already exists."

assert add_member("M001", "Tunde", "Tundejj23@mail.com") == "Member added successfully."
```

assert add_member("M001", "Keem Dup", "keem@mail.com") == "Member already exists."

assert borrow_book("M001", "001") == "OOP borrowed successfully."

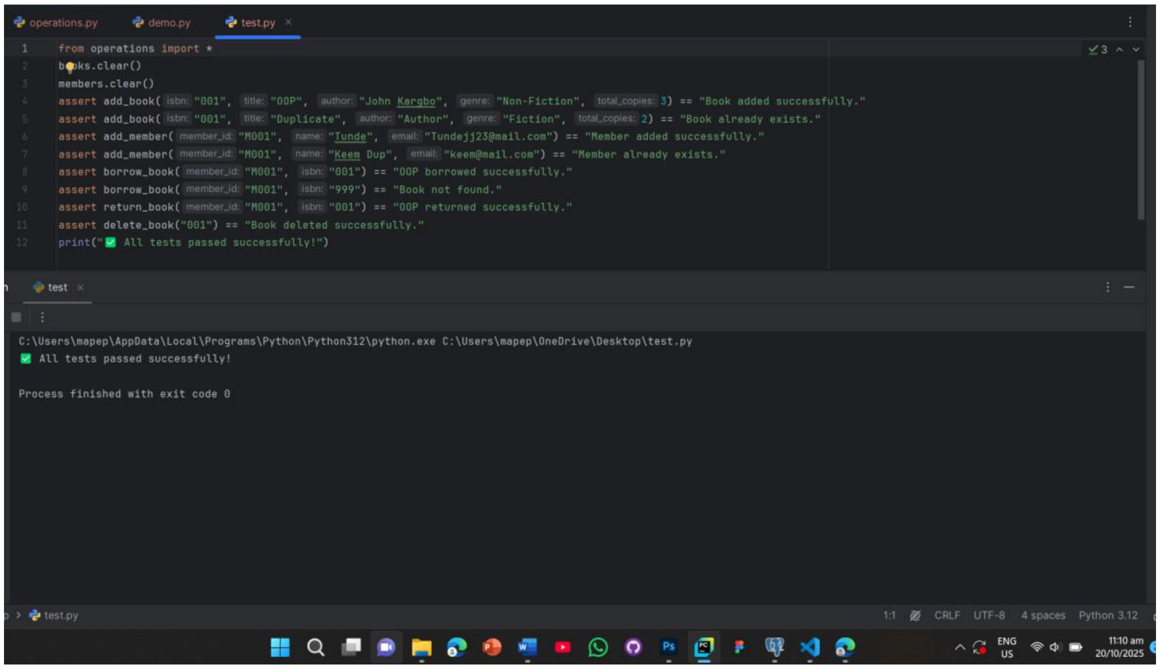
assert borrow_book("M001", "999") == "Book not found."

assert return_book("M001", "001") == "OOP returned successfully."

assert delete_book("001") == "Book deleted successfully."

print("☑ All tests passed successfully!")

Terminal debug:



```
from operations import *
books.clear()
members.clear()
assert add_book( isbn: "001",  title: "OOP",  author: "John Kargbo",  genre: "Non-Fiction",  total_copies: 3) == "Book added successfully."
assert add_book( isbn: "001",  title: "Duplicate",  author: "Author",  genre: "Fiction",  total_copies: 2) == "Book already exists."
assert add_member( member_id: "M001",  name: "Tunde",  email: "Tundejj23@mail.com") == "Member added successfully."
assert add_member( member_id: "M001",  name: "Keem Dup",  email: "keem@mail.com") == "Member already exists."
assert borrow_book( member_id: "M001",  isbn: "001") == "OOP borrowed successfully."
assert borrow_book( member_id: "M001",  isbn: "999") == "Book not found."
assert return_book( member_id: "M001",  isbn: "001") == "OOP returned successfully."
assert delete_book("001") == "Book deleted successfully."
print("✅ All tests passed successfully!")
```

```
C:\Users\mapep\AppData\Local\Programs\Python\Python312\python.exe C:\Users\mapep\OneDrive\Desktop\test.py
✅ All tests passed successfully!

Process finished with exit code 0
```

Demo Script

demo.py demonstrates adding books/members, borrowing, returning, and deleting.

demo.py — Demo Script

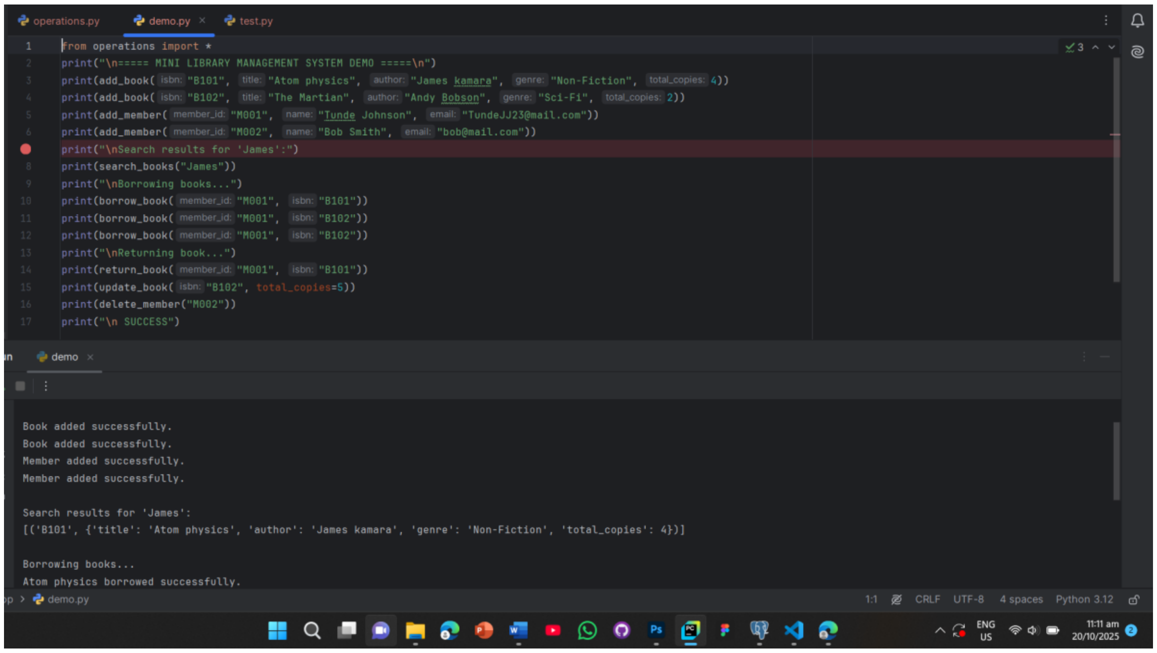
from operations import *

print("\n===== MINI LIBRARY MANAGEMENT SYSTEM DEMO =====\n")

print(add_book("B101", "Atom physics", "James kamara", "Non-Fiction", 4))

```
print(add_book("B102", "The Martian", "Andy Bobson", "Sci-Fi", 2))

print(add_member("M001", "Tunde Johnson", "TundeJJ23@mail.com"))

print(add_member("M002", "Bob Smith", "bob@mail.com"))

print("\nSearch results for 'James':")

print(search_books("James"))

print("\nBorrowing books...")

print(borrow_book("M001", "B101"))

print(borrow_book("M001", "B102"))

print(borrow_book("M001", "B102"))

print("\nReturning book...")

print(return_book("M001", "B101"))

print(update_book("B102", total_copies=5))

print(delete_member("M002"))

print("\n SUCCESS")
```

Terminal debug:



```
from operations import *
print("\n===== MINI LIBRARY MANAGEMENT SYSTEM DEMO =====\n")
print(add_book( isbn: "B101",  title: "Atom physics",  author: "James kamara",  genre: "Non-Fiction",  total_copies: 4))
print(add_book( isbn: "B102",  title: "The Martian",  author: "Andy Bobson",  genre: "Sci-Fi",  total_copies: 2))
print(add_member( member_id: "M001",  name: "Tunde Johnson",  email: "TundeJJ23@mail.com"))
print(add_member( member_id: "M002",  name: "Bob Smith",  email: "bob@mail.com"))
print("\nSearch results for 'James':")
print(search_books("James"))
print("\nBorrowing books...")
print(borrow_book( member_id: "M001",  isbn: "B101"))
print(borrow_book( member_id: "M001",  isbn: "B102"))
print(borrow_book( member_id: "M001",  isbn: "B102"))
print("\nReturning book...")
print(return_book( member_id: "M001",  isbn: "B101"))
print(update_book( isbn: "B102", total_copies=5))
print(delete_member("M002"))
print("\n SUCCESS")
```

```
Book added successfully.
Book added successfully.
Member added successfully.
Member added successfully.

Search results for 'James':
[('B101', {'title': 'Atom physics', 'author': 'James kamara', 'genre': 'Non-Fiction', 'total_copies': 4})]

Borrowing books...
Atom physics borrowed successfully.
```

Conclusion

The system uses appropriate Python data structures for clarity, maintainability, and testability. Dictionaries allow O(1) access, lists store members simply, and tuples ensure fixed genres. Ready for extensions like GUI or persistent storage.