## Introduction

This document outlines the design rationale and operational structure of a Python-based Library Management System. The system is built to manage books, members, and borrowing activities using simple data structures and functions. It is intended for small-scale library operations and educational purposes.

## Data Structures

### Books Dictionary

books = {ISBN: (title, author, genre, total\_copies)}

* **Purpose**: Stores book records using ISBN as the unique key.
* **Contents**: Each entry holds the title, author, genre, and number of available copies.
* **Advantages**: Enables fast lookup, update, and deletion of book records.

### Members List

members = [(member\_id, name, email, borrowed\_books), ...]

* **Purpose**: Maintains member information and their borrowed books.
* **Contents**: Each tuple includes member ID, name, email, and a list of borrowed ISBNs.
* **Advantages**: Simple structure for iteration and modification.

### Genres Tuple

genres = ("Fiction", "Non-Fiction", "Sci-Fi")

* **Purpose**: Defines valid genres for book classification.
* **Advantages**: Ensures genre consistency and validation.

## Functional Modules

### Add Book

add\_book(isbn, title, author, genre, total\_copies)

* Validates ISBN uniqueness and genre validity.
* Prevents negative copy counts.
* Adds the book to the books dictionary.

### Add Member

add\_member(member\_id, name, email)

* Ensures member ID is unique.
* Appends a new member with an empty borrowed list.

### Search Book

search\_book(query)

* Searches books by title or author (case-insensitive).
* Displays matching records with full details.

### Update Book

update\_book(isbn, title, author, genre, total\_copies)

* Validates ISBN existence and genre.
* Updates book details in the dictionary.

### Delete Book

delete\_book(isbn)

* Removes a book if the ISBN exists.
* Ensures safe deletion with existence check.

### Borrow Book

borrow\_book(member\_id, isbn)

* Verifies member and book existence.
* Checks for available copies.
* Updates book count and member's borrowed list.

### 

### Return Book

return\_book(member\_id, isbn)

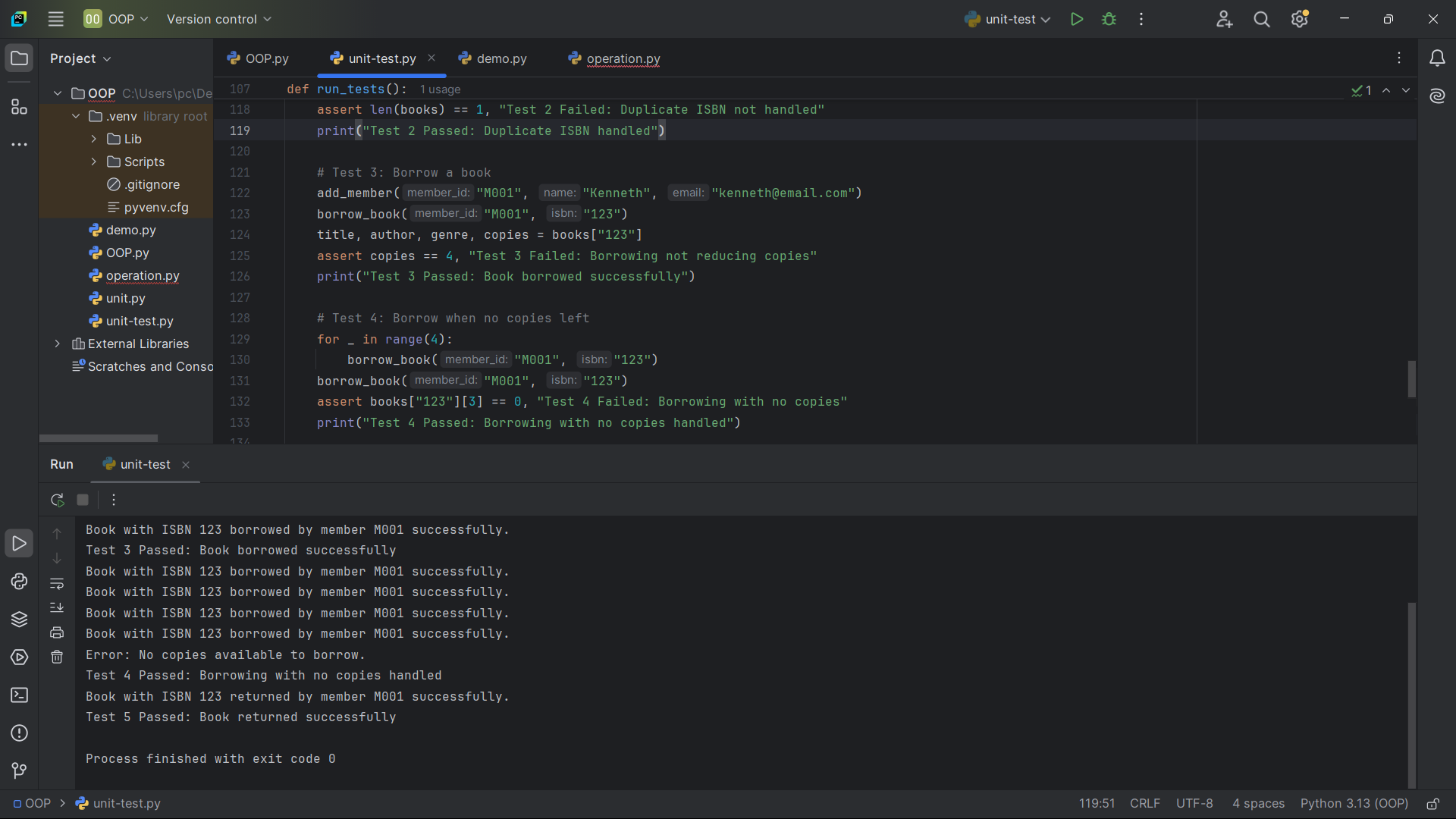
* Verifies member and book existence.
* Confirms the book was borrowed.
* Updates book count and removes ISBN from borrowed list.

## Conclusion

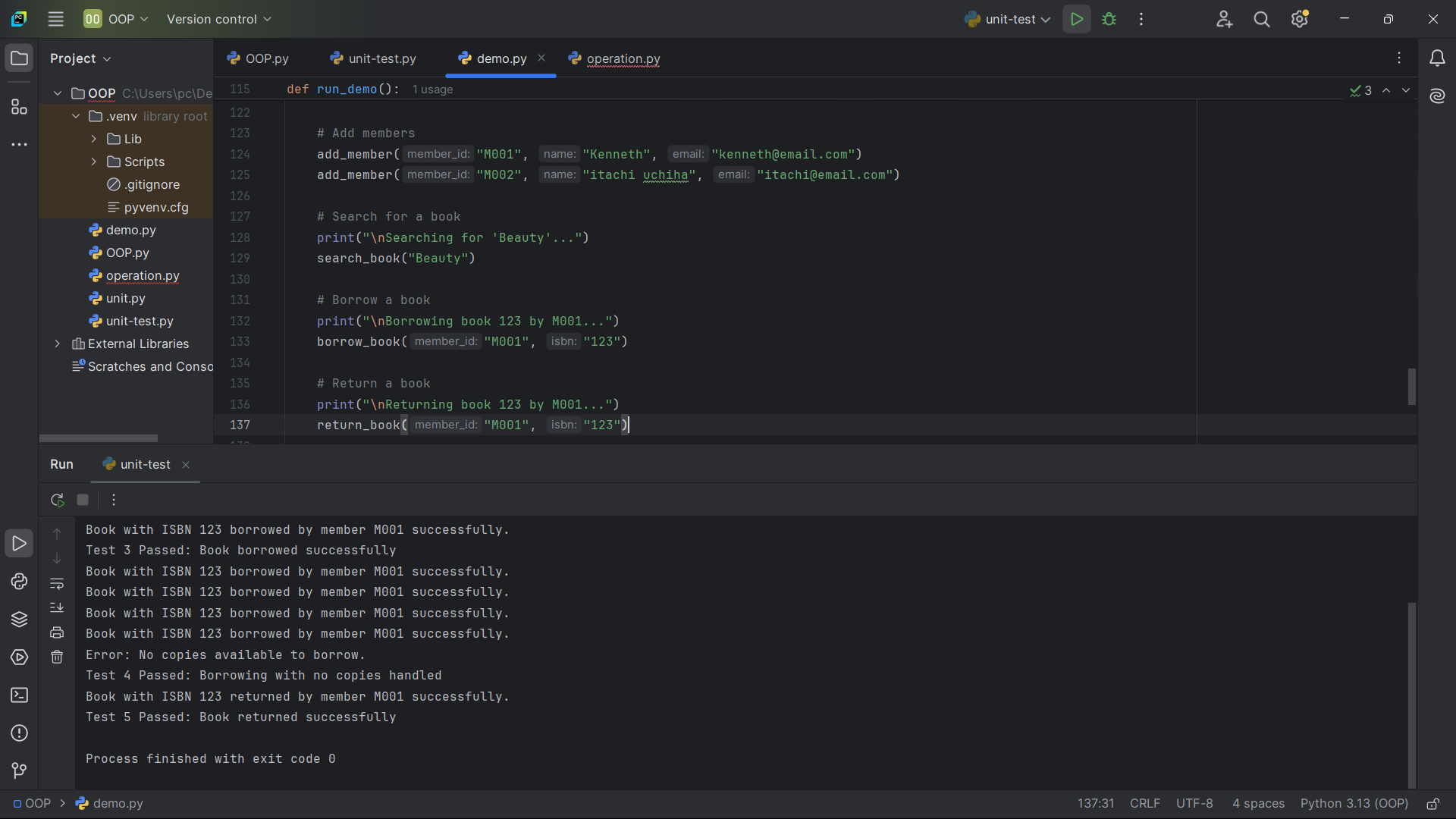
## This system provides a foundational framework for managing a library’s inventory and membership. With its clear structure and validation mechanisms, it serves as reliable tool for basic library operations. Future improvements can enhance scalability, persistence, and maintainability. Would you like me to help you format this into a downloadable document or refactor the code into a class-based version next?

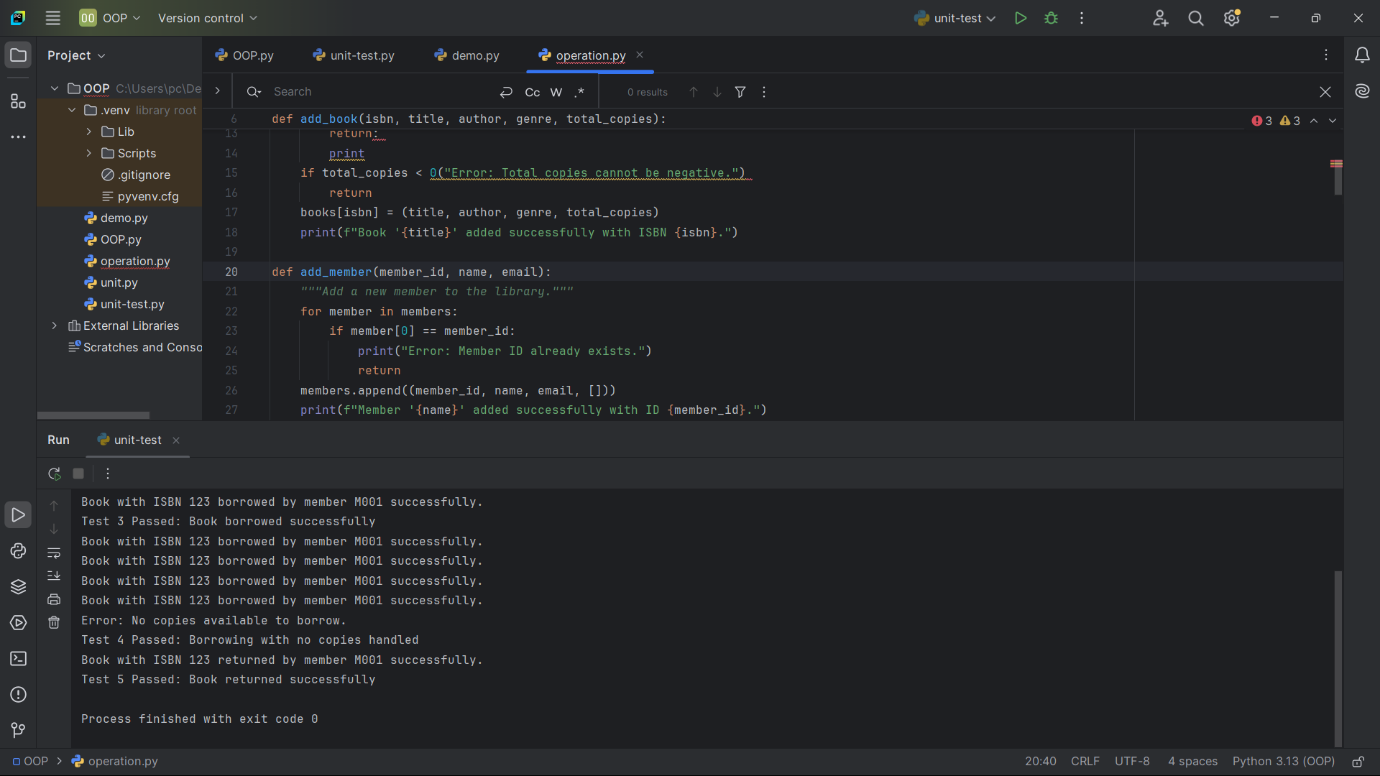
# Data structures  
books = {} # Dictionary: {ISBN: (title, author, genre, total\_copies)}  
members = [] # List: [(member\_id, name, email, borrowed\_books), ...]  
genres = ("Fiction", "Non-Fiction", "Sci-Fi") # Tuple of valid genres  
  
def add\_book(isbn, title, author, genre, total\_copies):  
 *"""Add a new book to the library."""* if isbn in books:  
 print("Error: ISBN already exists.")  
 return  
 if genre not in genres:  
 print("Error: Invalid genre.")  
 return  
 if total\_copies < 0:  
 print("Error: Total copies cannot be negative.")  
 return  
 books[isbn] = (title, author, genre, total\_copies)  
 print(f"Book '{title}' added successfully with ISBN {isbn}.")  
  
def add\_member(member\_id, name, email):  
 *"""Add a new member to the library."""* for member in members:  
 if member[0] == member\_id:  
 print("Error: Member ID already exists.")  
 return  
 members.append((member\_id, name, email, []))  
 print(f"Member '{name}' added successfully with ID {member\_id}.")  
  
def search\_book(query):  
 *"""Search for books by title or author."""* results = [  
 (isbn, title, author, genre, total\_copies)  
 for isbn, (title, author, genre, total\_copies) in books.items()  
 if query.lower() in title.lower() or query.lower() in author.lower()  
 ]  
 if results:  
 for isbn, title, author, genre, total\_copies in results:  
 print(f"Found: ISBN {isbn}, Title: {title}, Author: {author}, Genre: {genre}, Copies: {total\_copies}")  
 else:  
 print("No books found matching the query.")  
  
def update\_book(isbn, title, author, genre, total\_copies):  
 *"""Update book details."""* if isbn not in books:  
 print("Error: ISBN does not exist.")  
 return  
 if genre not in genres:  
 print("Error: Invalid genre.")  
 return  
 if total\_copies < 0:  
 print("Error: Total copies cannot be negative.")  
 return  
 books[isbn] = (title, author, genre, total\_copies)  
 print(f"Book with ISBN {isbn} updated successfully.")  
  
def delete\_book(isbn):  
 *"""Delete a book from the library."""* if isbn not in books:  
 print("Error: ISBN does not exist.")  
 return  
 books.pop(isbn)  
 print(f"Book with ISBN {isbn} deleted successfully.")  
  
def borrow\_book(member\_id, isbn):  
 *"""Allow a member to borrow a book."""* # Find member  
 member\_found = None  
 for member in members:  
 if member[0] == member\_id:  
 member\_found = member  
 break  
 if not member\_found:  
 print("Error: Member ID does not exist.")  
 return  
  
 # Check if book exists and has available copies  
 if isbn not in books:  
 print("Error: ISBN does not exist.")  
 return  
 title, author, genre, total\_copies = books[isbn]  
 if total\_copies <= 0:  
 print("Error: No copies available to borrow.")  
 return  
  
 # Update book copies and member's borrowed list  
 books[isbn] = (title, author, genre, total\_copies - 1)  
 member\_found[3].append(isbn) # Add ISBN to member's borrowed\_books  
 print(f"Book with ISBN {isbn} borrowed by member {member\_id} successfully.")  
  
def return\_book(member\_id, isbn):  
 *"""Allow a member to return a book."""* # Find member  
 member\_found = None  
 for member in members:  
 if member[0] == member\_id:  
 member\_found = member  
 break  
 if not member\_found:  
 print("Error: Member ID does not exist.")  
 return  
  
 # Check if book exists  
 if isbn not in books:  
 print("Error: ISBN does not exist.")  
 return  
 title, author, genre, total\_copies = books[isbn]  
  
 # Check if member borrowed the book  
 if isbn not in member\_found[3]:  
 print("Error: Member did not borrow this book.")  
 return  
  
 # Update book copies and remove ISBN from member's borrowed list  
 books[isbn] = (title, author, genre, total\_copies + 1)  
 member\_found[3].remove(isbn)  
 print(f"Book with ISBN {isbn} returned by member {member\_id} successfully.")

**TEST.PY**

****

**DEMO.PY**

****

**OPERATION.PY**