Design Rationale

1. Introduction

The Mini Library Management System is designed to perform basic library operations such as adding, searching, updating, deleting, borrowing, and returning books. The design is implemented in Python using fundamental data structures: dictionary, list, and tuple. These were chosen based on their efficiency, flexibility, and relevance to the type of data stored in a

2. Choice of Data Structures

a) Dictionary

A dictionary was used to store book records because it provides a fast and efficient way to access, update, or delete data using unique keys (ISBN).

Each book is stored as a key-value pair, where the key is the ISBN and the value is another dictionary containing book details such as title, author, genre, and number of copies.

This structure supports quick searching and updating since lookups in dictionaries are done in constant time O(1).

Example:

books = {

"9781234567890": {

"title": "Python Programming",

"author": "John Doe",

"genre": "Non-Fiction",

"total\_copies": 3

}

}

b) List

A list was used to store the members because the number of members can grow dynamically. Lists make it easy to iterate through all members and perform actions like searching by member ID or name.

Each member is represented as a dictionary inside the list, making it easy to access or modify their details.

Example:

members = [

{"member\_id": "M001", "name": "Alice", "email": "alice@gmail.com", "borrowed\_books": []}

]

Lists are also suitable for storing multiple dictionaries because they maintain order and allow easy looping when checking conditions (e.g., whether a member exists).

c) Tuple

A tuple was used for storing book genres since the set of valid genres is fixed and should not change during program execution. Tuples are immutable, meaning their contents cannot be modified — this ensures that only valid, predefined genres are used.

Example:

genres = ("Fiction", "Non-Fiction", "Science", "Romance", "Fantasy", "History")

Using a tuple provides data integrity and prevents accidental changes to the list of valid genres.

3. Functional Design (CRUD + Borrow/Return)

The system provides six major functions:

1. add\_book() – Adds a new book if ISBN is unique and genre is valid.

2. add\_member() – Registers a new library member.

3. search\_books() – Searches books by title or author.

4. update\_book() / update\_member() – Modifies existing information.

5. delete\_book() / delete\_member() – Removes entries if conditions are met.

6. borrow\_book() / return\_book() – Handles book borrowing (up to 3 at a time) and returns.

These functions ensure complete CRUD operation and enforce simple rules for borrowing and returning.

4. Design Benefits

Modular and Simple: Each function performs a specific task, making the program easy to understand and maintain.

Efficient: Dictionaries provide fast access and updates, while lists offer flexible iteration for members.

Data Integrity: Using tuples for genres prevents unintended changes.

Scalable: The design can easily be extended to include features like due dates, fines, or digital catalogs.

5. Conclusion

The choice of dictionary, list, and tuple makes the system well-structured, efficient, and easy to maintain. Dictionaries handle structured book data, lists handle dynamic member data, and tuples preserve fixed data like genres. This design balances flexibility and data safety while fulfilling all requirements for a mini library management system.