# Design Rationale: Mini Library Management System

The Mini Library Management System was developed in Python to manage books and members using fundamental data structures — dictionaries, lists, and tuples. The program allows adding, searching, updating, deleting, borrowing, and returning books, while ensuring data integrity and efficient access. This design focuses on simplicity, readability, and compliance with the assignment’s requirement to use these specific Python data structures instead of classes or databases.

## 1. Use of Dictionary

Dictionaries are ideal for representing books in the library. Each book is identified by a unique key — the ISBN — which makes searching and updating information fast and efficient.

Example:

* books = {  
   "9781234567897": {  
   "title": "Python Programming",  
   "author": "John Doe",  
   "genre": "Non-Fiction",  
   "copies\_total": 3,  
   "copies\_available": 2  
   }  
  }

Reasons for choosing a dictionary:

• Unique identification: Each book has a unique ISBN, ideal for dictionary keys.

• Fast access and updates: Lookups, insertions, and deletions are efficient.

• Clear organization: Key–value pairs make book attributes easy to manage.

## 2. Use of List

A list was used to store multiple members of the library. Each member is represented as a dictionary containing their ID, name, and borrowed books.

Example:

* members = [  
   {"id": 1, "name": "Alice", "borrowed\_books": []},  
   {"id": 2, "name": "Bob", "borrowed\_books": ["9781234567897"]}  
  ]

Reasons for choosing a list:

• Collection of similar items: Lists store multiple members easily.

• Easy iteration: Looping through members is straightforward.

• Flexibility: Members can be added or removed as needed.

## 3. Use of Tuple

Tuples were used to store valid book genres, providing an immutable list of allowed categories.

Example:

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

Reasons for choosing a tuple:

• Immutability: Prevents accidental modification of genres.

• Consistency: Ensures only predefined genres are used.

• Lightweight: Tuples use less memory than lists.

## 4. Conclusion

The combination of dictionaries, lists, and tuples creates a well-balanced system that is efficient, organized, and reliable. Dictionaries handle fast lookups and updates, lists manage collections of members flexibly, and tuples secure fixed data like genres. This design satisfies the assignment requirements and demonstrates practical understanding of Python’s data structures in a real-world context.