**Design Rationale**

**Title: Design Rationale for Mini Library Management System (Python)**

**1. Introduction**

This design rationale explains the reasoning behind the use of Python data structures — dictionaries, lists, and tuples — in the implementation of the Mini Library Management System. The goal of the system is to manage books and members effectively while maintaining simplicity and readability.

**2. Data Structure Choices**

**a. Dictionary (for Books):** Books are stored in a dictionary where the ISBN acts as a unique key, and the book details (title, author, genre, total copies) form the value.

**Reason:** ISBN values are unique and ideal for dictionary keys, Allows quick search, update, and deletion using key lookups (O(1) access time). Keeps each book’s data structured in one place.

**b. List (for Members)**

Members are stored as a list of dictionaries, with each dictionary containing details like member\_id, name, email, and borrowed\_books.

**Reason:** Lists allow easy iteration through multiple members. The list of dictionaries provides flexibility to add, search, and update members without complex indexing. Suited for small to medium datasets common in school assignments.

**c. Tuple (for Genres)**

A tuple is used to store valid genres such as Fiction, Non-Fiction, and Sci-Fi.

**Reason:** Tuples are immutable, meaning the list of genres cannot be accidentally changed.

It ensures data integrity and stability for fixed categories.

**UML DIAGRAM SKETCH (Hand Drawn)**

**Conclusion**

In conclusion, the Mini Library Management System demonstrates how fundamental Python data structures and functions can be combined to create an effective management tool. By utilizing dictionaries for books, lists for members, and tuples for genres, the system ensures easy data access and manipulation. The implementation of CRUD operations, borrowing, and returning processes further showcases the practical application of programming logic in solving real-world problems. This project not only strengthens programming and debugging skills but also lays a strong foundation for developing larger and more advanced systems in the future. Through comprehensive testing and validation, the system proves capable of handling essential library operations including book cataloging, member registration, borrowing transactions with copy limitations, and book returns. The modular design of the functions ensures maintainability and scalability for future enhancements.

This project reinforces fundamental programming concepts including data structure selection, function design, error handling, and systematic testing—all critical skills for software development. The system provides a solid foundation that could be extended with additional features such as due date tracking, fine calculations, or database integration for persistent storage.

**References**

1. Python Software Foundation. (2025). Python Documentation: Data Structures. Retrieved from https://docs.python.org/3/tutorial/datastructures.html

2. Lutz, M. (2013). Learning Python (5th ed.). O'Reilly Media.

3. McKinney, W. (2022). Python for Data Analysis (3rd ed.). O'Reilly Media.

4. Python Software Foundation. (2025). unittest — Unit testing framework. Retrieved from <https://docs.python.org/3/library/unittest.html>

5. Python Software Foundation. (2025). Python 3.12 Documentation. Retrieved from https://docs.python.org/3/

6. W3Schools. (2025). Python Dictionaries, Lists, and Tuples. Retrieved from https://www.w3schools.com/python/

7. GeeksforGeeks. (2025). Library Management System using Python. Retrieved from https://www.geeksforgeeks.org/