Design Rationale – Mini Library Management System

This design rationale explains the logic and reasoning behind the structure, data organization, and functional design of the Mini Library Management System implemented in Python. The project focuses on the use of lists, tuples, dictionaries, and functions.

1. Data Structure Selection

Different data structures were used based on their strengths and efficiency in representing the required entities.

• Dictionaries were used for books because each book has a unique ISBN number that serves as a key, allowing fast lookups, updates, and deletions. Each dictionary value stores another dictionary containing details like title, author, genre, and available copies.

• Lists were used for members because they are ideal for storing ordered collections. Each member is represented as a dictionary, and all members are stored in a list for easy iteration and searching.

• Tuples were used for genres since they contain a fixed set of predefined book categories. Tuples are immutable, ensuring that the list of valid genres cannot be accidentally modified during program execution.

2. Functional Design

The system is built entirely around functions, following a modular and procedural programming approach. Each function performs a specific, clearly defined task such as adding, deleting, borrowing, or returning books. This makes the program easy to maintain, test, and extend.

Functions like add_book() and add_member() handle input validation to prevent duplicates, while borrow_book() and return_book() manage borrowing logic and update book availability dynamically. This separation of logic ensures clarity and prevents code repetition.

3. Validation and Error Handling

Input validation is included in each function to ensure data integrity. For example, the add_book() function checks if the ISBN already exists or if the genre is valid. The borrow_book() function enforces borrowing limits and checks book availability before confirming a borrow request.

4. Simplicity and Scalability

The system is designed to be simple enough for educational purposes while maintaining scalability. New features like tracking due dates, adding fines, or managing librarians can easily be added later without restructuring the existing code.

5. Conclusion

Overall, the design emphasizes simplicity, clarity, and logical structure. By using lists, tuples, and dictionaries effectively, the system achieves both flexibility and maintainability. Functions provide modularity, making the Mini Library Management System a clear example of well-structured procedural programming in Python.