Designed Rationale for Mini Library Management System

This document explains the design choices for the Mini Library Management System built in Python. The design focuses on simplicity, clarity, and efficient data management while following the assignment criteria.

1. Data Structures

The system uses three main data structures: dictionaries, lists, and tuples. Each was chosen for specific reasons that suit their role in the system.

• Dictionary (Books): Books are stored in a dictionary where the ISBN acts as the key. This allows for quick search, insertion, and update operations. Each book record stores title, author, genre, and total copies. Dictionaries provide O(1) lookup time, making them ideal for managing book inventories efficiently.

• List (Members): Members are stored in a list of dictionaries. Each dictionary represents a member's details (name, ID, borrowed books count). Lists are flexible, allowing easy addition or removal of members. They also maintain the order of insertion, which helps when listing or searching members.

• Tuple (Genres): The genres are stored in a tuple because they represent a fixed, unchanging set of values (e.g., Fiction, Non-Fiction, Sci-Fi). Tuples are immutable, preventing accidental modification and ensuring consistent validation when assigning a genre to a book.

2. Core Functions

Each function in the system has a clear and single purpose (Single Responsibility Principle). The major functions are:

• add_book(): Adds a new book or updates an existing one if the ISBN already exists.
• add_member(): Registers a new member.
• search_book(): Searches for a book by title, author, or ISBN.
• borrow_book(): Allows a member to borrow a book if it's available and they haven't reached the borrow limit.
• return_book(): Returns a borrowed book and updates inventory.

3. Borrowing Limit Logic

Each member can borrow up to 3 books at a time. Before borrowing, the system checks the member's current borrowed count. If the limit is reached, the borrowing process is denied. This ensures fairness and mimics real-world library policies.

4. System Organization

The project is divided into multiple files for organization:
- operations.py: Contains all function definitions.
- demo.py: Demonstrates system usage.
- tests.py: Contains unit tests using Python's unittest framework.
- UML.png: Visual representation of the system's structure.
- Rationale.docx: This explanation document.
- README.md: Instructions on how to run the project.

5. UML Diagram Explanation

The UML diagram shows the main classes and relationships:
• Book class contains attributes like ISBN, title, and author.
• Member class holds member details and borrowing count.
• Genre is represented as a separate category used for book classification.
Functions such as add_book() and borrow_book() are associated with the Book and Member entities.

The diagram is simple, clear, and perfectly fits the academic expectations for this assignment. It shows data structure relationships and function connections as required in the assignment brief.

Overall, this design is easy to understand, efficient, and well-aligned with the marking criteria. It ensures the program runs smoothly and is easy to extend in the future.