**Design Rationale — Mini Library Management System**

**Overview**

This mini library system uses simple, built-in Python data structures to represent books, members, and genres. The focus was on clarity and educational value rather than advanced design patterns.

**Data Structures**

**Dictionaries for Books**

Books are stored in a dictionary keyed by ISBN. Reasons:

* ISBN is a natural unique identifier for books.
* Dictionary lookups by key are O(1), making search by ISBN fast and intuitive.
* Each book's metadata (title, author, genre, counts) naturally maps to a nested dictionary so it’s easy to extend.

I included available\_copies in addition to total\_copies to track real-time availability (borrowed vs. present copies). This makes borrow/return logic straightforward.

**Lists for Members**

Members are stored as a list of dictionaries where each member dictionary contains member\_id, name, email, and borrowed\_books (a list of ISBNs).

* Lists are simple to use and iterate over — suitable for small to medium collections often used in assignments.
* borrowed\_books is a list of ISBN strings; this keeps history simple and easy to display.

Note: for larger systems a dictionary keyed by member\_id would be more efficient for lookups.

**Tuple for Genres**

Genres are stored in a tuple to indicate a fixed set of valid options:

* Tuples are immutable, preventing accidental changes to the allowed genres.
* It’s small and expressive for demonstration.

**Functions and Behavior**

Functions are written to be small and single-purpose:

* add\_book, add\_member handle validation and insertion.
* borrow\_book, return\_book update both the books and members structures so data stays consistent.
* update\_ and delete\_ functions include simple safety checks (e.g., cannot delete a book that is currently borrowed).

**Simplicity and Extensibility**

* The design favors readability: helpers like \_find\_member\_index encapsulate lookup logic.
* To extend: swap members to a dict keyed by member\_id for speed, add persistence (saving to JSON/DB), or add borrowing limits / due dates.

**Conclusion**

This mix of dictionaries, lists, and tuples uses each structure where it is most natural: dictionaries for keyed fast access, lists for ordered collections that are easy to iterate or display, and tuples for fixed configuration data. The design balances clarity and correctness for an introductory university assignment.