**2. Design Rationale (2–3 pages summary)**

**Design Rationale for Mini Library Management System**

**Introduction**

The Mini Library Management System was developed in Python to manage books and members efficiently using core data structures such as **dictionary**, **list**, and **tuple**. The system supports adding, searching, updating, deleting, borrowing, and returning of books. These operations rely heavily on Python’s built-in data structures for data storage, retrieval, and manipulation.

**Choice of Data Structures**

1. **Dictionary for Books**
   * Books are stored in a dictionary where the **key** is the ISBN (unique identifier) and the **value** is another dictionary containing details such as title, author, genre, total copies, and available copies.
   * **Reason:**
     + Dictionaries offer **O(1)** average time complexity for lookup, addition, and deletion.
     + ISBNs are unique, which aligns naturally with the concept of a dictionary key.
     + Using nested dictionaries makes it easy to store multiple attributes for each book in a structured way.

**Example:**

books = {

"978001": {"title": "The Gamer", "author": "Amara kabba", "genre": "Sci-Fi", "total\_copies": 3, "available": 3}

}

This structure allows direct access to a book’s details using its ISBN, e.g., books["978001"]["title"].

1. **List of Dictionaries for Members**
   * Members are stored as a list of dictionaries, where each dictionary holds a member’s ID, name, email, and borrowed books.
   * **Reason:**
     + Lists are dynamic and can grow as new members are added.
     + Dictionaries within the list provide an organized representation of member data.

Searching and iterating through members is easy using loops or comprehensions.

**Example:**

members = [

{"member\_id": "M001", "name": "Alice", "email": "alice@example.com", "borrowed\_books": []}

]

This allows flexible member management, including adding, searching, and updating records.

1. **Tuple for Genres**
   * The system defines valid book genres using a tuple.
   * **Reason:**
     + Tuples are **immutable**, preventing accidental modification of valid genres during runtime.
     + They provide a quick way to validate user input when adding or updating books.

**Example:**

GENRES = ("Fiction", "Non-Fiction", "Sci-Fi", "Mystery", "Romance", "Biography")

This ensures consistency and restricts input to predefined categories.

**System Functions**

Each function in the system corresponds to a key operation in library management:

* add book () / add member () → Create new records
* search book () → Retrieve book information
* update book () / delete book () → Modify or remove records
* borrow book () / return\_book () → Track book lending activities

Functions ensure modularity, making the system easier to maintain and extend.

**Benefits of the Design**

* **Simplicity:** Built entirely with Python’s core data structures (no database needed).
* **Efficiency:** Fast lookups using dictionary keys and list iterations.
* **Scalability:** Easy to expand with file storage or GUI later.
* **Data Integrity:** Tuple ensures valid genres; borrow limits prevent misuse.

**Conclusion**

Using dictionaries, lists, and tuples provided a clean and efficient foundation for the Mini Library Management System. The approach balances simplicity with functionality, offering a strong conceptual model for future expansion (e.g., saving to files or integrating with databases).