**GitHub Rationale (README.md Section)**

**Project Title**

**Mini Library Management System (Python)**

**Overview**

The **Mini Library Management System** is a lightweight Python application designed to manage books, members, and borrowing activities using only Python’s built-in data structures — **dictionaries, lists, tuples, and functions**.  
It’s perfect for demonstrating **CRUD operations**, **data integrity**, and **modular programming principles** without relying on databases or external frameworks.

**Objectives**

* Build a simple and functional library management tool using Python fundamentals.
* Implement core library operations: Add, Search, Update, Delete, Borrow, and Return.
* Demonstrate modular, readable, and testable code architecture.
* Validate user actions and maintain accurate system state.

**System Architecture**

| **Component** | **File** | **Description** |
| --- | --- | --- |
| **Core Logic** | operations.py | Contains all CRUD and transaction functions. |
| **Demo Script** | demo.py | Simulates real-world use cases and interactions. |
| **Test Cases** | tests.py | Uses assert statements to validate correctness. |

**Data Structures Used**

| **Data Type** | **Purpose** | **Reason** |
| --- | --- | --- |
| **Dictionary (dict)** | Books | Enables fast ISBN lookups and updates. |
| **List of Dictionaries** | Members | Allows flexible storage of user details and borrowed books. |
| **Tuple (tuple)** | Genres | Maintains a fixed set of valid categories, ensuring consistency. |

**Core Functionalities**

* **Add Book / Add Member** – Registers new records with validation.
* **Search Book** – Retrieves books by title or author.
* **Update / Delete** – Safely modifies or removes entries.
* **Borrow / Return** – Tracks loaned books, with a 3-book limit per member.
* **Testing** – Validates system behavior using lightweight unit tests.

**Testing Approach**

* Implemented using assert statements for clarity and simplicity.
* Covers all major operations — addition, borrowing, returning, and deletion.
* Ensures data integrity and proper error handling.

**Future Improvements**

* Integrate database (SQLite/MySQL) for persistence.
* Add GUI with **Tkinter** or web dashboard via **Flask**.
* Implement authentication and role-based access.
* Enhance search and reporting features.

**Design Rationale**

This system follows clean coding and academic software design principles:

* **Simplicity:** Minimal dependencies, easy to understand and modify.
* **Scalability:** Can evolve into a fully-featured digital library system.
* **Maintainability:** Modular structure, clear naming conventions, and inline documentation.
* **Reliability:** Enforced data validation and assert-based testing.

**References**

* [Python Official Docs](https://docs.python.org/3/)
* [W3Schools Python Tutorial](https://www.w3schools.com/python/)
* GeeksforGeeks – Data Structures
* [PlantUML Class Diagrams](https://plantuml.com/class-diagram)