Design Rationale for Mini Library Management System

1. Introduction

This report outlines the design decisions behind the Mini Library Management System developed for PROG211 - Object-Oriented Programming 1. The system manages book inventory, member records, and borrowing operations using fundamental Python data structures. The design prioritizes efficiency, simplicity, and adherence to object-oriented principles while meeting all specified requirements.

2. Choice of Data Structures

Dictionary for Books

Rationale The dictionary was chosen for book storage because ISBN numbers provide natural unique keys. This enables O(1) time complexity for essential operations like searching, updating, and deleting books. The key-value mapping perfectly represents the relationship between ISBNs and book details, making it ideal for rapid lookups and efficient data management in a library context.

List for Members

Rationale A list was selected for member storage because members don't require key-based access patterns. Sequential processing adequately serves most member operations like registration updates and borrowing history checks. Lists provide simplicity for iteration and maintenance while offering sufficient performance for the expected scale of operations.

Tuple for Genres

Rationale A tuple was implemented for genre storage due to its immutable nature. Genres represent fixed, predefined categories that shouldn't change during program execution. This immutability prevents accidental modifications and ensures data integrity while providing memory efficiency for small, constant datasets.

3. Core Functionalities

The system implements comprehensive CRUD (Create, Read, Update, Delete) operations with robust validation

Book Management Includes add, search, update, and delete functions with ISBN uniqueness checks and genre validation. The search functionality supports both title and author queries for user convenience.

Member Management Handles member registration, updates, and deletion with proper constraints to prevent removal of members with active borrows.

Borrowing System Implements a sophisticated borrowing mechanism with multiple validations

· Copy availability checks

· Maximum limit enforcement (3 books per member)

· Prevention of duplicate borrows

· Automatic inventory adjustment

Error Handling Comprehensive validation ensures data integrity through

· Input sanitization

· Business rule enforcement

· Clear user feedback messages

4. Testing Approach

The testing strategy employs Python's assert statements to verify system correctness

Functional Testing Validates all core operations including bookmember management and borrowing workflows.

Edge Case Testing Tests boundary conditions like duplicate entries, unavailable copies, and maximum borrowing limits.

Integration Testing Ensures components work together correctly, particularly the interaction between book availability and member borrowing status.

Validation Testing Confirms proper error handling for invalid inputs and constraint violations.

5. Conclusion

The Mini Library Management System successfully demonstrates effective use of Python's core data structures to solve real-world problems. The dictionary-list-tuple combination provides an optimal balance of performance, simplicity, and maintainability. The system meets all specified requirements while maintaining clean code organization and comprehensive functionality. This design serves as a solid foundation that could be extended with additional features like persistent storage or enhanced reporting capabilities in future iterations.