Library Management System Report

Introduction:

The provided code implements a Library Management System in Python. It consists of several classes and functions to manage books, users, checkouts, and data storage.

How to run this code:

- Just run the “library.py file” on your system, rest other classes are defined within it.
- Make sure while adding a book, you enter a valid ISBN number, otherwise it will throw a warning of invalid ISBN.

  (To verify an ISBN, calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third digit and so on until we add 1 time the last digit. If the final number leaves no remainder when divided by 11, the code is a valid ISBN.)

Some examples of valid ISBN are:

- 1259060977
- 0747532745

Classes:

Book: Represents a book in the library. It has attributes such as title, author, ISBN, and a checked_out flag to track its availability status.
User: Represents a user of the library. It has attributes for name and user ID.
Library: Manages the collection of books, users, and checkouts. It provides functionalities to add, list, and delete books and users. It also includes methods for validating ISBNs and listing checkouts.
CheckManager: Manages the checkout and check-in processes for books. It ensures that books are correctly checked out and checked in by users.

Why are multiple classes beneficial here?

Using multiple classes in the Library Management System promotes modularity, enhances readability, and facilitates easier maintenance and scalability. Each class focuses on a specific responsibility, making the codebase more organized and intuitive.

For example, the User class focuses on only users, Book class focuses only on books and binds all the attributes related to it as a single entity in a class. This helps in scalability as later on we can modify these classes to add more features.

The separation allows for independent development and testing of features, making the system more robust and adaptable to future changes or expansions.

Validations and Checks:

ISBN Validation: The system validates ISBNs to ensure they consist of 10 digits and follow the ISBN checksum algorithm.

Availability Checks: Before checking out a book, the system verifies if the book is available and not already checked out. Similarly, during check-in, it ensures that the book is currently checked out before processing the return.

Deletion Check: It checks whether a book is available in the library while deleting, if it is checked out by someone and we try deleting it then it raises an error.

User Existence Verification: During the checkout process, the system verifies the existence of the user attempting to borrow the book. It checks if the provided user ID corresponds to a valid user in the system, safeguarding against attempts to check out books by non-existent users.

Error Handling for Invalid Inputs: Robust error handling mechanisms are implemented to catch and handle invalid inputs gracefully. The system provides informative error messages to users, guiding them on how to correct their inputs and proceed with their desired actions. For example if another user enters the same ID as some previous user, it will throw an error.

Similarly some other checks and validations are done throughout the program.

Data Storage:

The system utilizes JSON files for data storage. It saves information about books and users in separate JSON files when the program exits. It stores the data in the form of a dictionary. This allows the system to persist data between program runs.

Error Handling:

The code includes error handling mechanisms to catch and handle exceptions gracefully. It notifies users when input data is invalid or when operations cannot be performed due to certain conditions (e.g., attempting to delete a book that is currently checked out).