Library Management System

Project Overview

The **Library Management System (LMS)** is a Python-based application designed to manage the operations of a library. It offers core functionality such as adding new books, searching for available books, borrowing books, and returning borrowed books. This system is implemented using **lists** and **dictionaries** to store data, and it provides a simple command-line menu for user interaction.

The project demonstrates the effective use of fundamental Python concepts, including data structures (lists and dictionaries), functions, control flow (conditionals and loops), and input/output handling. The system is lightweight and easy to use, making it ideal for small library operations or as a learning tool for Python developers.

Features

1. Add a Book

- Users can add new books to the library by providing the book title and author name.
- The book details are stored in the system for future operations such as searching and borrowing.

2. Search for a Book

- Users can search for books in the library by entering the book title.
- The system will return the author of the book if it is found in the library. Otherwise, it will notify the user that the book is unavailable.

3. Borrow a Book

- Users can borrow available books by entering the book title and their name.
- Once a book is borrowed, it is removed from the available library books and stored in a borrowed books list along with the borrower's name.

4. Return a Book

- Users can return borrowed books by entering the book title.
- The system will then add the book back to the library's available books and remove it from the list of borrowed books.

5. Menu System

- The system provides an interactive menu to guide users through available actions: adding books, searching for books, borrowing, returning, or exiting the program.

How It Works

Data Structures

- **library_books (list)**:
    - This list stores all the available books in the library. Each book is stored as a dictionary with the keys title and author.

    Example:

library_books = [{"title": "Masterminds", "author": "JCornellius M}, {"title": " Wild one ", "author": "J ordin Wills "}]

- **borrowed_books (dictionary)**:
    - This dictionary stores the titles of books that have been borrowed, along with the name of the person who borrowed them.

- The keys are the book titles, and the values are the names of the borrowers.

Example:

borrowed_books = {"Kifo Kisimani": "Joy"}

Functions

1. add_book(book_title, author)

- **Purpose**: Adds a new book to the library's list of available books.
- **Parameters**:
  - book_title (string): The title of the book to be added.
  - author (string): The author of the book.
- **Functionality**:
  - The function creates a dictionary representing the book and adds it to the library_books list.
  - It then prints a confirmation message.
- **Example Call**:

```
Library Menu:
1. Add a Book
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Exit
Enter your choice (1-5):  1
Enter the book title:  Kifo Kisimani
Enter the author:  Wallah Bin wallah
Book "Kifo Kisimani" by Wallah Bin wallah added to the library.
```

2. search_book(book_title)

- **Purpose**: Searches for a book by its title in the library.
- **Parameter**:
  - book_title (string): The title of the book to be searched.

- **Functionality**:
  - The function iterates through the library_books list and checks if the book is available.
  - If found, it prints the title and author; otherwise, it displays a "not found" message.
- **Example Call**:

```
Library Menu:
1. Add a Book
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Exit
Enter your choice (1-5):  2
Enter the book title:  Kifo Kisimani
Book found: Kifo Kisimani by Wallah Bin wallah
```

3. borrow_book(book_title, user_name)

- **Purpose**: Allows a user to borrow a book from the library.
- **Parameters**:
  - book_title (string): The title of the book to be borrowed.
  - user_name (string): The name of the person borrowing the book.
- **Functionality**:
  - The function checks if the book is available in the library_books list. If found, it removes the book from the list and adds it to the borrowed_books dictionary along with the borrower's name.
  - If the book is already borrowed or not available, it notifies the user.
- **Example Call**:

```
Library Menu:
1. Add a Book
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Exit
Enter your choice (1-5):  3
Enter the book title:  Motivation of the midnight sun
Enter your name:  Joy
Book "Motivation of the midnight sun" borrowed by Joy.
```

4. return_book(book_title)

- **Purpose**: Allows a user to return a borrowed book.

- **Parameter**:
  - book_title (string): The title of the book to be returned.
- **Functionality**:
  - The function checks if the book is in the borrowed_books dictionary. If found, it removes the book from the borrowed list and adds it back to the library_books list.
  - It assumes the author's name may be unknown when returned, so it sets the author to "Unknown".
  - If the book is not borrowed, it notifies the user.
- **Example Call**:

```
Library Menu:
1. Add a Book
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Exit
Enter your choice (1-5):  4
Enter the book title:  Motivation of the midnight sun
Book "Motivation of the midnight sun" returned by Joy.
```

5. library_menu()

- **Purpose**: Provides a menu for users to select actions and interact with the library system.
- **Functionality**:
  - The function displays a list of options (Add Book, Search Book, Borrow Book, Return Book, Exit) and handles user input to perform the selected operation.
- **Example Call**:

```
Library Menu:
1. Add a Book
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Exit
Enter your choice (1-5):
```

Code Flow

1. **User Interaction**:

a. When the program starts, the library_menu() function displays the available options to the user.
b. The user selects an option by entering a number (1-5).

2. **Executing Actions**:
    a. Based on the user's choice, the corresponding function is executed:
        i. **Add a Book**: The user enters the book title and author to add a new book.
        ii. **Search for a Book**: The user enters the book title to check if it's available.
        iii. **Borrow a Book**: The user enters the book title and their name to borrow a book.
        iv. **Return a Book**: The user enters the book title to return a borrowed book.
3. **Looping**:
    a. After completing an action, the menu is displayed again for further operations until the user chooses to exit.

Sample Interaction

Example 1: Adding a Book

```
Library Menu:
1. Add a Book
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Exit
Enter your choice (1-5):  1
Enter the book title:  Masterminds
Enter the author:  Cornellius M
Book "Masterminds" by Cornellius M added to the library.
```

Example 2: Borrowing a Book

```
Library Menu:
1. Add a Book
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Exit
Enter your choice (1-5):  3
Enter the book title:  Motivation of the midnight sun
Enter your name:  Joy
Book "Motivation of the midnight sun" borrowed by Joy.
```

Potential Enhancements

1. **Multiple Copies**:
    a. Track the number of copies available for each book, allowing multiple copies of the same book to be borrowed simultaneously.
2. **User Registration**:
    a. Add a user authentication system to allow for more detailed tracking of borrowers.
3. **Persistent Storage**:
    a. Integrate a database (e.g., SQLite) for persistent storage of library data to ensure the library records are not lost when the program ends.
4. **Graphical User Interface (GUI)**:
    a. Develop a GUI using libraries such as Tkinter or PyQt to make the system more user-friendly.

Conclusion

The **Library Management System** is a simple and functional project that highlights the use of Python's core concepts such as lists, dictionaries, and functions. It allows users to perform common library operations such as adding, searching, borrowing, and returning

books. The project is easily extendable and can be improved by incorporating additional features like multiple copies of books, user authentication, and persistent storage.

This project is a valuable tool for learning Python and serves as a foundation for more advanced applications in data management and system development.