📚 Rockview Library Management System Documentation

Overview

The Rockview Library Management System is a cross-platform desktop and mobile application designed to manage a small library's book collection and transactions. It is built using **Python** and the **KivyMD** framework, providing a modern, touch-friendly user interface.
All book data, including availability and borrowing history, is stored locally in a library_records.json file.

🌟 Key Features

- **Book List & Search:** A main view that lists all books. Users can filter the collection instantly by Book ID, Title, or Author.
- **Add New Records:** Dedicated screen for adding new books, enforcing unique Book IDs.
- **Transaction Management:** Screens for easily recording the **Borrowing** and **Returning** of books.
- **Data Persistence:** Automatic saving and loading of the entire library catalogue to and from a local JSON file.
- **Safe Deletion:** A confirmation dialogue is required before permanently deleting a book, and deletion is blocked if the book is currently borrowed.
- **Responsive UI:** Built with KivyMD for a clean look, utilizing a **Navigation Drawer** for easy screen switching.

🛠️ Prerequisites

To run this application, you must have **Python** and the **KivyMD** library installed on your system.

Installation Steps

1. **Python 3.x** is required.
2. Install Kivy and KivyMD using pip:
   ```
   pip install kivy kivymd
   ```

🚀 How to Run the Application

1. Save the provided code as a file named Library_Rockview.py.
2. Open your terminal or command prompt.
3. Navigate to the directory where you saved the file.
4. Execute the application using the Python interpreter:
   ```
   python Library_Rockview.py
   ```

A desktop application window will open. Upon the first run, a new data file (library_records.json) will be initialized.

📂 Project Structure and Data Flow

The entire application is self-contained within the single Library_Rockview.py file, utilizing Kivy's built-in KV language for UI definition.

Data Persistence (library_records.json)

Book records are stored as a list of dictionaries in a local JSON file. This file is managed by the application's core methods:

- load_data(): Reads the JSON file into the in-memory self.books list upon application startup (build()).
- save_data(): Writes the current state of self.books back to the JSON file, called after any change (Add, Borrow, Return, Delete) and when the application stops (on_stop()).

Book Record Structure

Each book record stored in the system follows this JSON structure:

```
{
    "book_id": "L001",
    "title": "The Great Gatsby",
    "author": "F. Scott Fitzgerald",
    "year_published": 1925,
    "availability": "Available" or "Borrowed",
    "borrower_name": "John Doe" or "",
    "date_borrowed": "2025-11-25" or ""
}
```

⚙️ Core Application Logic

The BookManagerApp class contains the methods that handle all user interactions and data manipulation:

Method	Screen / Context	Function
add_book_ui()	Add Book Screen	Validates user input (ID uniqueness, year format), creates a new book record, adds it to self.books, and calls save_data().
load_books()	Book List Screen	Filters and sorts the self.books list, then dynamically generates OneLineIconListItem widgets to display the results in the UI.
borrow_book_ui()	Transactions Screen	Finds the book by ID. If

Method	Screen / Context	Function
		available, updates the record's availability to "Borrowed" and fills in the borrower_name and current date. Calls save_data().
return_book_ui()	Transactions Screen	Finds the book by ID. If borrowed, resets availability to "Available" and clears the borrower details. Calls save_data().
_execute_delete(book_id)	Transactions Screen	Called after confirmation. Removes the book from self.books list if it is currently "Available" and calls save_data().
show_dialog()	Global Utility	Displays modal dialogue boxes for alerts, errors, and confirmation requests (replacing standard Python alert/confirm which are restricted in KivyMD).