Project Report: Library Book Sorter & Tracker

Course: VITyarthi - Build Your Own Project

Student Name: Aryan Kuamr

Registration Number: 25BCE10362

Submission Date: 26/11/2025

1. Introduction

The **Library Book Sorter & Tracker** is a desktop application designed to streamline the management of library resources. In traditional small-scale libraries or personal collections, tracking book availability, due dates, and sorting titles often relies on manual ledgers or spreadsheets, which are prone to human error. This project digitizes these processes, providing a user-friendly Graphical User Interface (GUI) to manage book status, track lending periods for both single titles and multi-volume series, and organize the catalog efficiently.

2. Problem Statement

Managing a physical book collection manually presents several challenges:

- **Inefficiency:** Searching for specific books or checking availability status takes time.
- **Complex Inventories:** Tracking individual volumes within large series (e.g., *Harry Potter*) is difficult when treated as a single entry.
- **Data Integrity:** Manual records of checkout times and due dates are easily lost or miscalculated.
- **Organization:** Keeping a list sorted by different criteria (Author, Title, Availability) requires constant rewriting of physical lists.

This project addresses these issues by creating a centralized digital system that automates tracking, date calculation, series volume management, and sorting.

3. Functional Requirements

The system implements the following core functions:

1. **Book Inventory Management:**
    - Display a catalog of classic books and series with details (ID, Title, Author).
    - Real-time status visualization (Available, Checked Out, or Partial Series status like "3/7 Checked Out").
2. **Series Management:**

    - **Volume Tracking:** Support for multi-volume entries (e.g., *Lord of the Rings*).
    - **Granular Control:** Users can check out specific volumes (e.g., "The Two Towers") while leaving others available.
3. **Circulation Operations:**
    - **Check Out:** Users can borrow a book or a specific volume by specifying a duration. The system automatically calculates the checkout timestamp and due date.
    - **Return:** Users can return specific volumes or single books, resetting their status.
    - **Validation:** Prevents checking out an already borrowed item or returning an available one.
4. **Sorting & Organization:**
    - **Sort by Title/Author:** Alphabetical ordering.
    - **Sort by Status:** Groups books by their availability state, prioritizing available items.

4. Non-Functional Requirements

1. **Usability:** The interface uses a clean, modern GUI (Tkinter) with color-coded status indicators (Red for Checked Out, Green for Available, Orange for Partial Series) to ensure users can grasp the system state immediately.
2. **Performance:** Operations such as sorting and state updates occur instantly (< 100ms) due to optimized in-memory data structures.
3. **Reliability:** The system includes error handling to prevent invalid inputs (e.g., checking out a volume that is already out).
4. **Maintainability:** The code follows Object-Oriented Programming (OOP) principles, creating a flexible Book class that handles both single items and series logic transparently.

5. System Architecture

The application follows a monolithic architecture suitable for desktop GUI applications, structured around the **Model-View-Controller (MVC)** concept:

- **Model (Book Class):** Represents the data entity. It holds state (is_checked_out), properties (title, author), and complex logic for handling volumes and series_status.
- **View (Tkinter Widgets):** The Treeview table, buttons, custom selection dialogs, and labels.
- **Controller (LibraryApp Class):** Handles user events, manages the book list, and updates the View.

6. Design Diagrams

6.1 Use Case Diagram

```
usecaseDiagram
  actor Librarian
  Librarian --> (View Catalog)
```

```
Librarian --> (Sort Books)
Librarian --> (Check Out Item)
Librarian --> (Return Item)
(Check Out Item) .> (Select Volume) : extend
(Check Out Item) .> (Calculate Due Date) : include
(Return Item) .> (Select Volume) : extend
```

6.2 Class Diagram

```
classDiagram
    class Book {
        +int id
        +str title
        +str author
        +list volumes
        +dict series_status
        +check_out(days, volume_name)
        +return_book(volume_name)
        +get_status_display()
        +get_available_volumes()
    }

    class LibraryApp {
        +Tk root
        +list books
        +Treeview tree
        +ask_selection(title, prompt, options)
        +action_checkout()
        +action_return()
        +sort_books(criteria)
    }

    LibraryApp "1" *-- "Many" Book : manages
```

7. Design Decisions & Rationale

- **Data Structure for Series:** A dictionary (series_status) was used to map volume names to their specific checkout/due dates. This allows O(1) access to status checkouts and flexible tracking of partial availability.

- **Custom Dialogs:** Standard simpledialog was insufficient for selecting specific volumes. A custom Toplevel window with a Combobox was implemented (ask_selection) to provide a better user experience when choosing between multiple volumes.
- **Polymorphism in Logic:** The Book class handles both single books and series internally. The LibraryApp doesn't need to know *how* a book handles its status, it just calls check_out or return_book, simplifying the main application logic.

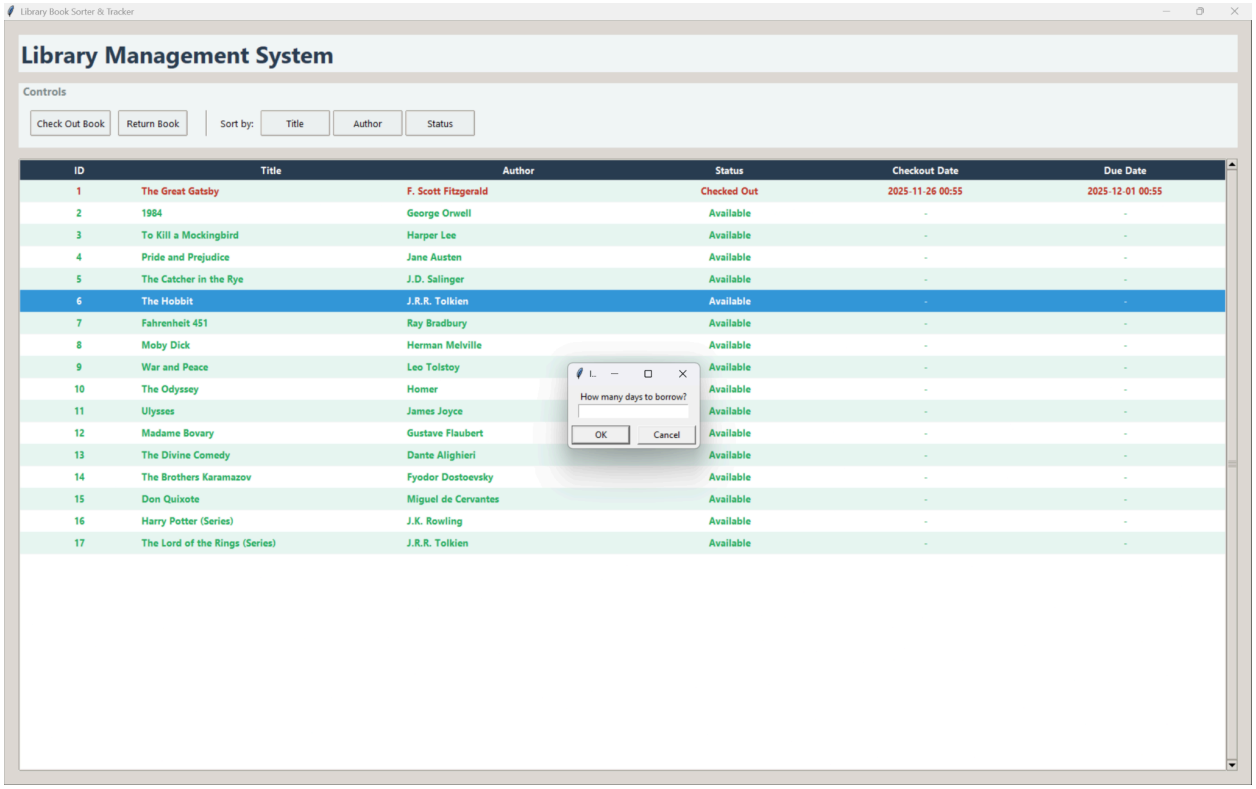
8. Implementation Details

The project is implemented in a single modular Python file (library_sorter.py).

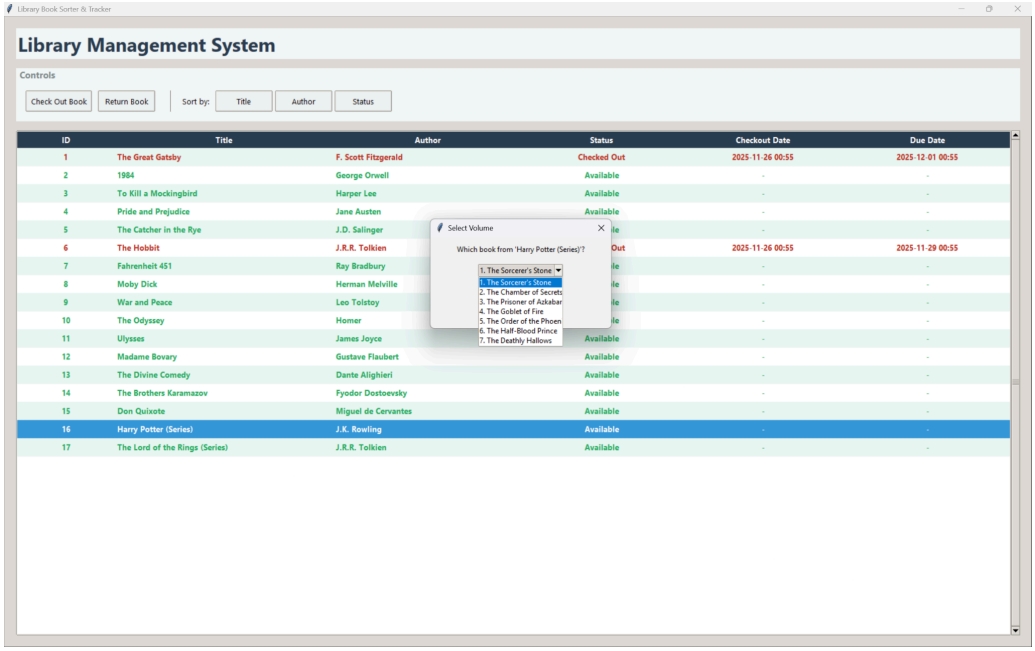
- **Series Logic:** The Book class detects if volumes are provided. If so, it switches to "Series Mode", where is_checked_out is ignored in favor of the series_status dictionary.
- **Status Display:** A complex helper method get_status_display() dynamically generates strings like "Available", "All Checked Out", or "2/3 Checked Out" based on the internal state.
- **UI Construction:** Uses ttk.Treeview with custom tagging to color-code rows based on three states: Available (Green), Checked Out (Red), and Partially Available (Orange).

9. Screenshots / Results

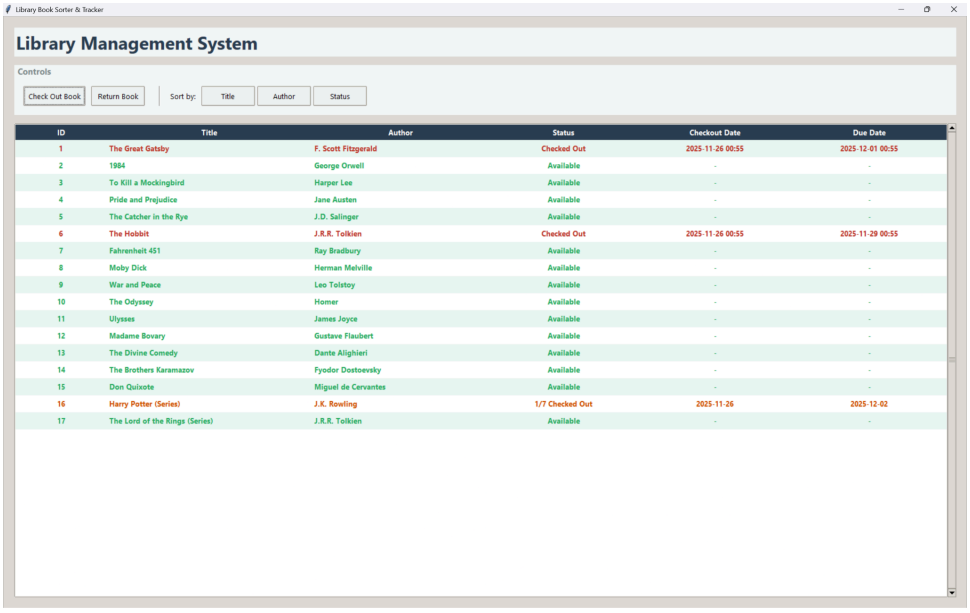
1. **Main Interface:** Shows the table with mixed single books and series.



2. **Volume Selection:** Shows the popup dropdown asking "Which book from 'Harry Potter'?".



3. **Partial Status:** Shows a series row in orange with text "1/7 Checked Out".



10. Testing Approach

Manual testing was conducted to verify functional correctness:

- **Single Book Testing:**
  - Check out/Return standard books -> Verified status updates and color changes.
- **Series Testing:**

- Select "Harry Potter" -> Check Out -> Select "Chamber of Secrets" -> Enter 7 days.
- **Verify:** Status changes to "1/7 Checked Out" (Orange text).
- Check Out again -> Verify "Chamber of Secrets" is NOT in the dropdown list (preventing double booking).
- Return -> Select "Chamber of Secrets" -> Verify status resets to "Available".

11. Challenges Faced

- **Complex Status Logic:** Differentiating between a single book being "Checked Out" and a series being "Partially Checked Out" required careful logic in the get_status_display method.
- **Dynamic UI Interaction:** Creating a blocking popup dialog (ask_selection) that waits for user input before proceeding with the checkout logic required understanding Tkinter's wait_window method.

12. Learnings & Key Takeaways

- **Data Modeling:** Learned how to model complex real-world objects (like book series) within a class structure.
- **UI UX Design:** Improved understanding of how to guide users through multi-step processes (Select Book -> Select Volume -> Enter Days).
- **Code Reusability:** The ask_selection function was designed to be generic, used for both Checkout and Return operations.

13. Future Enhancements

1. **Persistence:** Save the library state to a file so data isn't lost on close.
2. **Search:** Add a text filter to find books instantly.
3. **Late Fees:** Calculate fines based on the difference between due_date and current time.

14. References

- Python Documentation (Tkinter): https://docs.python.org/3/library/tkinter.html
- Python Documentation (Datetime): https://docs.python.org/3/library/datetime.html