Project Documentation:

Library Management System

**Introduction**

GitHub: <https://github.com/DmitryGanotskiy/LibraryTkinter/tree/master/venv>

The Library Management System is an application to automate library operations. It offers a user-friendly graphical interface and library staff to efficiently manage the library's resources. This documentation provides an overview of the system's architecture, functionalities, and usage.

**Setup Module (setup.py)**

The setup module serves as the app structure, and contains the "TK" class, which represents the main application window. This class encapsulates various attributes and methods crucial for initialising and configuring the application environment. Key functionalities include:

1. **Window Configuration:** The "TK" class initialises and configures the main application window, setting attributes such as background colour, size, and title to create a functional interface.
2. **GUI Element Creation:** Through the "\_createElements" method, the class dynamically generates GUI elements such as buttons, frames, and canvas, facilitating user interaction and content display within the application.
3. **Book Entry Management:** The "TK" class manages book entries within the system, maintaining lists to store book details, borrowed books, and borrower information. Methods like "\_addBook", "\_RemoveVar", "\_show", "\_userBooks", "\_serviceForm", "\_borrow", "\_return", and "\_exit" handle various aspects of book management, including addition, removal, display, borrowing, returning, and application exit.

**Book Entry Module (bookEntry.py)**

The BookEntry module defines the "BookEntry" class, which represents individual book entries within the library. This class encapsulates book-related attributes and functionalities, including:

1. **Initialisation:** Upon creation, each "BookEntry" object receives an ID, name, and availability status, providing a structured representation of book data.
2. **Accessor Methods:** The class provides accessor methods such as "getName" and "getAvailable" to retrieve the name and availability status of a book entry.
3. **Update Methods:** Methods like "updateAvailability" and "updateName" enable the modification of a book entry's availability status and name, respectively.

**Main Module**

The main module serves as the entry point for the Library Management System, orchestrating the initialisation and execution of the application. Key functionalities include:

1. **Initialisation:** Upon execution, the module creates an instance of the "TK" class, triggering the setup and configuration of the main application window.
2. **Execution:** The "run" method of the "TK" instance initiates the event loop, allowing the application to respond to user input and interactions effectively.

**Detailed Functionality**

1. **Adding Book Entries:** Users can add new book entries to the system via the "Add" button. Upon addition, each book entry includes fields for ID, name, and availability, facilitating organised storage and retrieval of book information.
2. **Removing Book Entries:** The system supports the removal of existing book entries through the "Remove" button. Removal operations update the entry count and reorder remaining entries, ensuring data integrity and consistency.
3. **Displaying Book Entries:** The system provides functionality to display all existing book entries within a scrollable frame, presenting relevant details such as ID, name, and availability for each entry.
4. **Managing Borrowed Books:** Users can borrow and return books from the library using the "Take or Return" button. Borrowing operations update availability status and borrower information, facilitating efficient tracking of borrowed resources.
5. **Service Form:** The system includes a service form accessible via the "Take or Return" button, allowing users to perform borrowing and returning actions by entering their name and the book's ID or name.
6. **Exiting the Application:** Users can gracefully exit the application using the "Exit" button, terminating the event loop, and closing the main application window.

**Methods used**

1. **Encapsulation:**

OOP encapsulation is evident throughout the Library Management System's design, primarily using classes and methods.

The TK class encapsulates attributes and methods related to the main application window, including book entry management, GUI creation, and event handling.

1. **Abstraction:**

Abstraction is achieved by hiding the internal complexities of the system's components and providing a simplified interface for interaction.

Users interact with the system through intuitive GUI elements and functionalities, abstracting away the underlying implementation details.

Methods such as addBook, removeBook, borrowBook, and returnBook provide abstracted interfaces for performing common library management tasks.

1. **Inheritance:**

Could be employed to create specialised subclasses of the BookEntry class if additional features or behaviours are required.

For example, subclasses could be created to represent specific types of books.

1. **Polymorphism:**

The system demonstrates polymorphic behaviour using uniform method names such as updateAvailability across different book entry instances, allowing for consistent interaction regardless of the specific book being manipulated.

Utilisation of Python Dictionaries:

1. **Storage of Borrower Information**

The borrowerDict attribute within the TK class serves as a dictionary where keys represent book IDs and values represent dictionaries containing borrower names and borrowed book names.

This dictionary structure enables quick retrieval and modification of borrower information based on book IDs, enhancing the system's performance and flexibility.

1. **Dynamic Book Entry Management:**

The entries attribute within the TK class stores book entry objects as values, with their corresponding IDs serving as keys.

This dictionary structure allows for efficient addition, removal, and retrieval of book entries based on their unique IDs, ensuring accurate and organised management of library resources.

The dictionary-based approach enhances the system's responsiveness and scalability, particularly in scenarios involving large datasets or frequent data updates.

**Usage Instructions**

1. Execute the "main.py".
2. Window will appear, providing access to library functionalities.
3. Interact with the GUI elements as needed to add, remove, borrow, and return books, ensuring efficient management of library resources.
4. To exit the application, click the "Exit" button.

**Pseudocode**

Initialization:

1. Create an empty list of "entries" to store book entries.
2. Create an empty list "borrowed" to store labels representing book availability.
3. Create an empty dictionary "borrowerDict" to store information about borrowers and their borrowed books (borrower name as key, dictionary of borrowed book names with book ID as key as value).
4. Initialise a counter "count" to keep track of the number of book entries (starts at 0).
5. Create the main application window.

Set the window title.

Configure window size and background colour.

Create a scrollable frame with a canvas and a scrollbar.

1. Create buttons:

"Add": Triggers adding a new book entry.

"Show": Displays detailed information about all book entries.

"Take or Return": Presents a form for borrowing or returning books.

"Borrowed": Lists books borrowed by each user.

"Exit": Terminates the application.

Initially, disable the "Show", "Borrowed", and "Take or Return" buttons.

Adding a Book:

1. If the number of book entries is less than 11:

Create a new frame to represent the book entry.

Generate a unique book ID using the current "count" value.

Create Tkinter widgets for book names and a boolean variable to track availability.

Create a "BookEntry" object with the generated ID, name entry, and availability variable.

Append the "BookEntry" object to the "entries" list.

Increment the "count" value.

Pack the entry frame, name label, name entry, availability label, checkbox, and remove button within the book entry frame.

Enable "Show", "Borrowed", and "Take or Return" buttons.

Removing a Book:

1. Get the index of the entry frame to be removed.
2. If the index is valid:

Destroy the entry frame, effectively removing it from the GUI.

Remove the corresponding "BookEntry" instance from the "entries" list at the same index.

Reassign IDs for remaining book entries to maintain consistency (iterate through remaining entries and update their "id" attribute with the current index).

Showing Book Details:

1. Destroy any previously displayed frames.
2. Create a new frame to display book details.
3. Iterate through all "BookEntry" objects in the "entries" list.

For each entry, create a new frame.

Retrieve book ID, name, and availability from the "BookEntry" instance.

Display book information (ID, name, and availability) as labels within the new frame.

Update the corresponding label in the "borrowed" list based on the book's availability.

Handle potential duplicate book names: If a name is encountered multiple times, append "\_copy{number}" to the original name, where "{number}" represents the occurrence count within a temporary name list.

Borrowing a Book:

1. Retrieve borrower name and book ID/name from the service form.
2. Iterate through all "BookEntry" objects in the "entries" list.

If a match is found based on the provided book ID/name and the book is available:

Update the book's availability to False.

Update the corresponding label in the "borrowed" list to reflect the change.

Add borrower and book information to the "borrowerDict":

Key: Book ID from the "BookEntry" object.

Value: Dictionary with borrower name as key and borrowed book name as the value.

Returning a Book:

1. Retrieve returner name and book ID/name from the service form.
2. Iterate through the "borrowerDict" to find the borrower and the specific borrowed book.

If a match is found:

Update the book's availability to True.

Remove the book entry from the borrower's list in the "borrowerDict".

If there are no more borrowed books for the borrower, remove the borrower's entry from the "borrowerDict".

Listing Borrowed Books:

1. Iterate through the "borrowerDict" dictionary.

For each borrower:

Create a new frame.

Display borrower name as a label.

Create a listbox to display borrowed book names.

Iterate through the borrowed books list associated with the borrower and insert each book name

**Conclusion**

The Library Management System offers a robust solution for libraries and educational institutions seeking to streamline their resource management processes. The system delivers a user-friendly interface coupled with comprehensive functionalities for efficient book addition, removal, borrowing, and returning. With its modular architecture and intuitive design, the system caters to the diverse needs of library administrators, empowering them to optimise library operations effectively.