Introduction

This report presents a Python-based library automation system designed to enhance user experience and streamline library management processes. The system is user-friendly, accessible via a console interface, and provides four main options: adding new books, deleting books, displaying the book catalog, and exiting the program.

The motivation behind developing this library automation system in Python was to create a user-friendly and efficient tool for managing book records and user interactions. The system aims to streamline library management processes and enhance the user experience by providing intuitive options and robust data storage capabilities.

My efforts in developing this system involved carefully considering user requirements, selecting suitable technologies, and implementing the necessary functionalities to ensure a seamless and effective library management experience.

I believe that this system can greatly benefit libraries and individuals alike by simplifying book management tasks, facilitating quick and easy access to book information, and contributing to an organized and efficient library environment.

Functionality

- When a user selects option 1 (Add a New Book), they are prompted to enter the book's title, their own name, the number of days they plan to read the book, and the expected return date. This information is stored as a list within a list and recorded in a database for permanent storage, ensuring data persistence even after the program is closed.
- Option 2 (Delete a Book) allows users to remove a book from the database by entering its title. The system locates and deletes the book, notifying the user of the successful deletion.
- Selecting option 3 (Show the Books) displays a comprehensive list of all books in the database, including their respective information.
- Option 4 (Exit) gracefully terminates the program.

Technologies Used

The library automation system leverages several technologies:

1. **Python:** Python is a versatile, general-purpose programming language known for its readability and ease of use. It enables the development of clear, logical code for projects of varying scales.

2. **SQLite:** SQLite is a transactional SQL software engine widely used for its portability and reliability. It integrates seamlessly with Python, allowing for efficient data storage and retrieval.

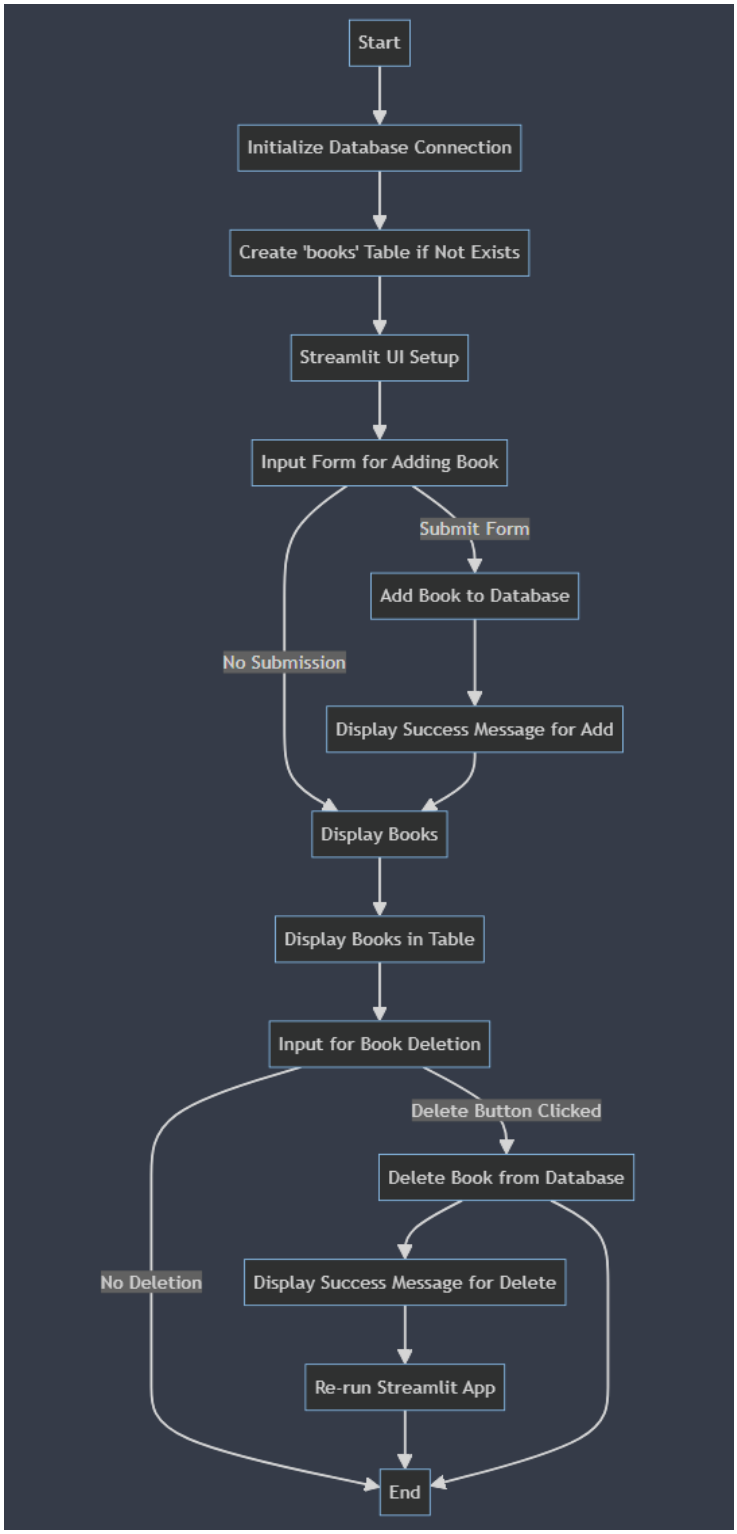
Related Applications

Several other applications offer similar functionality:

1. Google Keep: Google Keep is a note-taking app that supports tasks, reminders, and image recognition. It facilitates color coding, geofencing, and shareable notes.

2. Microsoft OneNote: OneNote is a digital notebook that enables users to organize notes, images, and scanned documents. It features a powerful search function and seamless integration with the Microsoft Office suite.

Flowchart



Conclusion

The library automation system developed in Python provides a robust solution for managing book records and user interactions. Its user-friendly interface and efficient data storage capabilities make it a valuable tool for libraries and individuals alike.

Overview

This code builds a web-based application using the Streamlit library to manage a library database. Here's the core functionality:

- **Database Interaction:** It handles connecting to an SQLite database, making changes (adding and deleting books), and retrieving data for display.
- **User Interface:** Streamlit creates a web form for users to input book details and a table to display the book inventory.

Step-by-Step Explanation

Imports

streamlit: The core library for building web interfaces.

pandas: Used to display the book data in a tabular format.

sqlite3: For interacting with the SQLite database.

Database Functions

def init_db(conn):

... Creates the 'books' table if it doesn't exist

def add_book(conn, ...):

... Inserts a new book record into the database

def delete_book(conn, ...):

... Deletes a book record based on its ID

def get_all_books(conn):

... Fetches all the book records from the database

These functions handle the interaction with the 'library.db' SQLite database.

App Setup

Initialize the database

conn = sqlite3.connect('library.db')

init_db(conn)

Streamlit UI

st.title("Library Management System MVP")

- **Connects to the database:** Creates the database if it doesn't exist.
- **Sets the app title:** Displays the title of the web application.

Input Form

with st.form("book_form", clear_on_submit=True):

... Input fields for book details

submit_button = st.form_submit_button("Add Book")

if submit_button:

add_book(conn, ...)

st.success("Book added successfully!")

- **st.form("book_form"):** Creates a form labeled "book_form".
- **Input Fields (st.text_input):** Creates spaces for users to enter book information.
- **Submit Button (st.form_submit_button):** A button to trigger adding the book.
- **Conditional (if submit_button:):** If the button is pressed, the add_book function is called and a success message is displayed.

Display Books

books = get_all_books(conn)

books_df = pd.DataFrame(books, ...)

st.write(books_df)

- **Retrieves books from the database (**get_all_books**)**
- **Uses Pandas DataFrame to format the data for display.**
- st.write **displays the data as a table.**

Delete Functionality

book_id_to_delete = st.selectbox(...)

if st.button("Delete Book"):

```
delete_book(conn, book_id_to_delete)

st.success(...)

st.experimental_rerun()
```

- st.selectbox**:** Creates a dropdown to select a book ID for deletion.
- **Delete Button:** When pressed, calls delete_book and displays a success message.
- st.experimental_rerun()**:** Refreshes the app to reflect the changes.

Closing the Database

```
conn.close()
```

- Closes the database connection to release resources.

Database Setup (Desktop version)

- **import sqlite3**: Imports the library for working with SQLite databases.
- **vt = sqlite3.connect('database.db')**: Connects to a database file named 'database.db' (creates it if it doesn't exist).
- **im = vt.cursor()**: Prepares a cursor (like a pointer) for interacting with the database.
- **im.execute("create TABLE IF NOT EXISTS tablo (d1, d2,d3,d4)")**: Creates a table named "tablo" with columns "d1", "d2", "d3", and "d4" if the table doesn't already exist.

Tkinter Window Setup

- **root = Tk()**: Creates the main window for the application.
- **root.title("My Library")**: Sets the title of the window to "My Library".
- **root.geometry("530x400")**: Sets the size of the window to 530 pixels wide and 400 pixels tall.

Initial Data Loading

- **im.execute("SELECT * FROM tablo ")**: Fetches all existing data from the "tablo" database table.
- **data = im.fetchall():** Stores the fetched data in the data variable.
- **...** (code to format data) Prepares the data for display in the listbox.

Listbox for Displaying Data

- **label = Label(...)**: Creates a label with headings for the information to be displayed (Book Name, Book No, etc.).
- **listbox = Listbox(...)**: Creates a listbox to display the book and student records.
- **for i in saves: ...:** Adds the formatted data to the listbox.

User Input Fields

- **number = Label(...) **name = Label(...) **surname = Label(...) phone = Label(...)**: Creates labels for the input fields ("Book Name", "Book No", etc.)
- ****no_entry = Entry(...) **n_entry = Entry(...) **s_entry = Entry(...) tel_entry = Entry(...)**: Creates entry fields where the user can type information.

Buttons

- **add = Button(..., command=add)**: Creates an "Add" button that, when clicked, runs the add function.
- **delete = Button(..., command=delete)**: Creates a "Delete" button that, when clicked, runs the delete function.

Functions to Modify Data

- **def add():**
  - Retrieves data from the entry fields.
  - Uses **im.execute()** to insert a new record into the database table.
  - Adds the new information to the listbox for display.
  - Commits (saves) the changes to the database with **vt.commit()**.
- **def delete():**
  - Gets the selected item from the listbox.
  - Deletes the corresponding record from the database using **im.execute()**.
  - Removes the item from the listbox.
  - Commits (saves) the changes to the database with **vt.commit()**.

Running the Application

- **mainloop()**: Starts the Tkinter event loop, making the window appear and respond to user actions.