ABOUT THE LIBRARY MANAGEMENT SYSTEM

Overview

The Library Management System (LMS) is designed to manage a collection of books efficiently. It utilizes a linked list data structure for in-memory management of book records, allowing for dynamic memory allocation and easy manipulation of book entries.

Key Features

1. **Book Management**:
    - **Add Book**: Users can add new books by providing the title, author, and ISBN. The system checks for duplicate ISBNs to ensure data integrity.
    - **Delete Book**: Users can delete books from the collection using the ISBN. The linked list is updated accordingly to remove the book node.
    - **Update Book**: Users can modify existing book details, including title, author, and ISBN, ensuring that the linked list reflects these changes.
2. **PDF Management**:
    - **Upload PDF**: Users can associate PDF files with books, allowing for easy access to digital copies of the books.
    - **View PDF**: The system enables users to open and view the associated PDF files directly from the application.
3. **Undo Functionality**:
    - The system maintains an undo stack to allow users to revert the last add or delete operation, enhancing user experience and data management.
4. **Data Persistence**:
    - While the linked list manages books in memory, the system also integrates with an SQLite database for persistent storage. This ensures that book records are saved and can be retrieved even after the application is closed.

Data Structure

Linked List Implementation

- **Node Structure**: Each book is represented as a node in the linked list, containing the following attributes:
    - **title**: The title of the book.
    - **author**: The author of the book.
    - **isbn**: The unique ISBN of the book.

- **pdf_path**: The file path of the associated PDF (if any).
- **next**: A pointer to the next node in the linked list.

- **Head Pointer**: The linked list maintains a head pointer that points to the first book in the list. If the list is empty, the head pointer is **None**.

Operations on a Linked List

1. Adding a Book:

A new node is created. By doing so, the system seamlessly integrates a new book into the existing collection. To initialize the list, this new node assumes the role of the head in the absence of previous nodes.

2. Deleting a Book:

One prerequisite for deletion is finding the corresponding book. This process involves methodically traversing the linked list based on the unique identifier, ISBN. Once located, the node representing the specified book is meticulously removed. Simultaneously, the surrounding pointers are adjusted to ensure structural integrity within the list.

3. Updating a Book:

The system embarks on a search for a specific book by referencing its ISBN. Upon successful identification, the associated attributes undergo direct modification within the linked list, enhancing the existing information and reflecting changes accurately.

4. Viewing Books:

Detailed supervision is enabled through the system's ability to traverse the entire linked list. By doing so, it reliably retrieves and showcases comprehensive records, affording an informative overview of the collection, encompassing every book entry within the library's realm User Interface

The library management system (LMS) features an intuitive graphical user interface (GUI) developed with Tkinter, enabling users to engage with the system effortlessly.

Outcome:

The book management system utilized by the library, which is based on a linked list, offers an effective and adaptable approach to organizing and overseeing the library's collection of books. The linked list design enhances memory management efficiency, simplifying the tasks of adding, deleting, and altering book records. Alongside its user-friendly interface and reliable storage through SQLite, the LMS presents a trustworthy and efficient solution for library management.

HARDWARE & SOFTWARE REQUIREMENTS

Hardware Requirements

1. Processor:

Minimum: 1 GHz single-core processor

Recommended: Intel i3 or AMD Ryzen 3 for better performance

2. Memory (RAM):

Minimum: 2 GB

Recommended: 4 GB or more for smoother operation, especially with multiple applications

3. Storage:

Minimum: 10 GB of free disk space

Recommended: SSD for faster read/write speeds, especially when handling larger databases

4. Display:

Minimum: 1080p resolution for better visibility

Recommended: Larger screen or dual monitor setup for enhanced pro

Software Requirements

1. Operating System:

   * Windows, macOS, or Linux (any of these will work).

2. Python Environment:

   * **Python Version:** Python 3.x (latest version recommended).

   * **Package Manager:** pip (included with Python).

3. Required Libraries:

   * **Sqlite3 : A python** library that **simplifies** the use of **sqlite** databases for storing and managing **data.**

   * **Tkinter: a pre-built Python library that allows developers to create interactive graphical user interfaces (GUIs).**

* **Messagebox: a component of tkinter used to create pop-up message dialogs for user notifications**

* **Simpledialog is a tkinter feature that enables the creation of dialog boxes to gather basic user input.**

* **Filedialog is a tkinter utility that offers dialogs for users to choose files from their file system.**

* **Webbrowser is a Python module that allows users to open web pages in their preferred web browser.**

***OS**: **a** standard library in Python that **offers** functions for interacting with the operating system, **such as managing files** and **directories**

* **Pillow is a Python library that enables the opening, editing, and saving of different image formats, with Image and ImageTk used for incorporating images into tkinter applications**

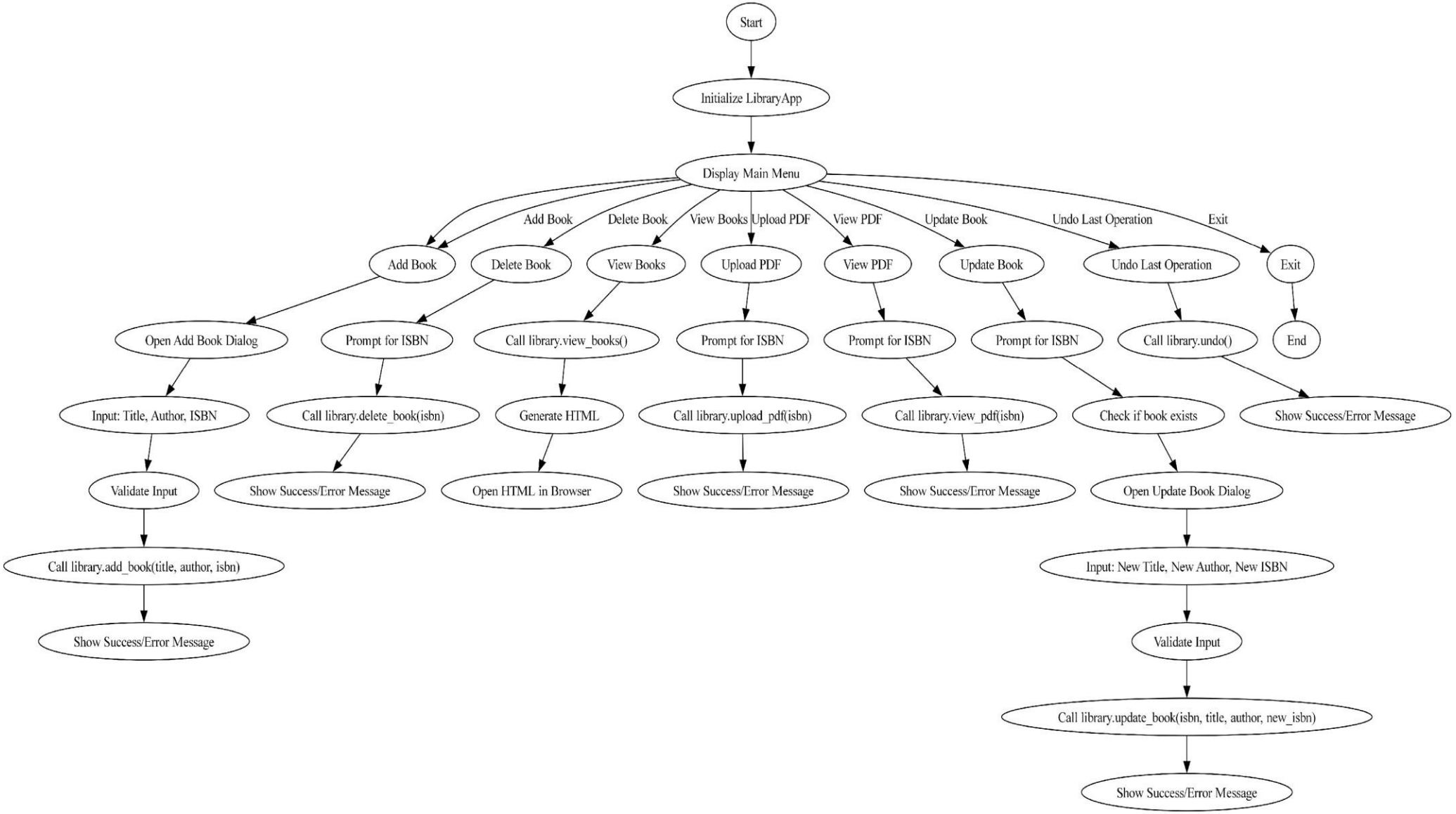
Development Tools:

* **IDE/Text Editor:** Recommended options include

  * PyCharm

  * Visual Studio Code

  * Jupyter Notebook


Start
Initialize LibraryApp
Display Main Menu
Add Book
Delete Book
View Books
Upload PDF
View PDF
Update Book
Undo Last Operation
Exit
Add Book
Delete Book
View Books
Upload PDF
View PDF
Update Book
Undo Last Operation
Exit
Open Add Book Dialog
Prompt for ISBN
Call library.view_books()
Prompt for ISBN
Prompt for ISBN
Prompt for ISBN
Call library.undo()
End
Input: Title, Author, ISBN
Call library.delete_book(isbn)
Generate HTML
Call library.upload_pdf(isbn)
Call library.view_pdf(isbn)
Check if book exists
Show Success/Error Message
Validate Input
Show Success/Error Message
Open HTML in Browser
Show Success/Error Message
Show Success/Error Message
Open Update Book Dialog
Call library.add_book(title, author, isbn)
Input: New Title, New Author, New ISBN
Show Success/Error Message
Validate Input
Call library.update_book(isbn, title, author, new_isbn)
Show Success/Error Message

CODE:

```
import sqlite3 import
tkinter as tk
from tkinter import messagebox, simpledialog, filedialog import
webbrowser
import os
from PIL import Image, ImageTk


class Book:
    def __init__(self, title, author, isbn, pdf_path=None):
        self.title = title
        self.author = author
        self.isbn = isbn
        self.pdf_path = pdf_path
        self.next = None


class Library:
    def _init_(self): self.head
        = None self.undo_stack
        = [] self.create_table()

    def connect_db(self):
        return sqlite3.connect("library_management.db")

    def create_table(self): conn
        = self.connect_db()
        cursor = conn.cursor()
        create_table_sql = '''
        CREATE TABLE IF NOT EXISTS books (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            title TEXT NOT NULL,
            author TEXT NOT NULL,
            isbn TEXT NOT NULL UNIQUE,
            pdf_path TEXT
        );
        '''
        cursor.execute(create_table_sql)
        conn.commit()
        conn.close()

    def add_book(self, title, author, isbn):
        conn = self.connect_db()
        cursor = conn.cursor()
        cursor.execute("SELECT COUNT(*) FROM books WHERE isbn=?", (isbn,))
        result = cursor.fetchone()
        conn.close()

        if result[0] > 0:
            messagebox.showerror("Error", f'Book with ISBN "{isbn}" already
exists.')
            return

        new_book = Book(title, author, isbn)
        if not self.head:
            self.head = new_book
        else:
            current = self.head
            while current.next:
                current = current.next
```

```
            current.next = new_book

        conn = self.connect_db()
        cursor = conn.cursor()
        try:
            cursor.execute("INSERT INTO books (title, author, isbn) VALUES (?,
?, ?)", (title, author, isbn))
            conn.commit() self.undo_stack.append(("add",
            new_book))
            messagebox.showinfo("Success", f'Book "{title}" added successfully.')
        except sqlite3.IntegrityError:
            messagebox.showerror("Error", f'Book with ISBN "{isbn}" already
exists.')
        finally:
            conn.close()

    def delete_book(self, isbn):
        current = self.head
        previous = None
        while current:
            if current.isbn == isbn:
                if previous:
                    previous.next = current.next
                else:
                    self.head = current.next

                conn = self.connect_db()
                cursor = conn.cursor()
                cursor.execute("DELETE FROM books WHERE isbn=?", (isbn,))
                conn.commit()
                conn.close() self.undo_stack.append(("delete",

                current))

                messagebox.showinfo("Success", f'Book "{current.title}" deleted
successfully.')
                return
            previous = current
            current = current.next
        messagebox.showwarning("Not Found", "Book not found!")

    def upload_pdf(self, isbn):
        conn = self.connect_db()
        cursor = conn.cursor()
        cursor.execute("SELECT COUNT(*) FROM books WHERE isbn=?", (isbn,))
        result = cursor.fetchone()
        if result[0] == 0:
            messagebox.showerror("Error", f"Incorrect ISBN: {isbn}. Book not
found.")
            return
        pdf_path = filedialog.askopenfilename(title="Select PDF File",
filetypes=[("PDF Files", "*.pdf")]) if
        pdf_path:
            cursor.execute("UPDATE books SET pdf_path=? WHERE isbn=?", (pdf_path,
isbn))
            conn.commit() conn.close()
            messagebox.showinfo("Success", f'PDF for book with ISBN "{isbn}"
uploaded successfully.')
        else:
            conn.close()

    def view_pdf(self, isbn):
        conn = self.connect_db()
```

```
        cursor = conn.cursor()
        cursor.execute("SELECT pdf_path FROM books WHERE isbn=?", (isbn,)) pdf_path =
        cursor.fetchone()
        conn.close()
        if pdf_path and pdf_path[0]:
            webbrowser.open(pdf_path[0])
        else:
            messagebox.showwarning("Not Found", "No PDF found for this book.")

    def undo(self):
        if not self.undo_stack:
            messagebox.showwarning("Undo", "No operations to undo!")
            return

        operation = self.undo_stack.pop()
        if operation[0] == "add":
            self.delete_book(operation[1].isbn) elif
        operation[0] == "delete":
            self.add_book(operation[1].title, operation[1].author, operation[1].isbn)

    def view_books(self):
        conn = self.connect_db()
        cursor = conn.cursor()
        cursor.execute("SELECT title, author, isbn, pdf_path FROM books") rows =
        cursor.fetchall()
        conn.close()

        if not rows:
            messagebox.showinfo("Books", "No books found.")
            return

        html_content = """
        <html>
        <head>
            <title>Library Books</title>
            <meta name="viewport" content="width=device-width, initial-
scale=1.0">
            <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=sw ap"
rel="stylesheet">
            <style>
                :root {
                    --primary-color: #3498db;
                    --secondary-color: #2ecc71;
                    --background-light: #f4f6f7;
                    --text-dark: #2c3e50;
                    --text-muted: #7f8c8d;
                    --border-radius: 12px;
                    --box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);
                }

                * {
                    margin: 0;
                    padding: 0;
                    box-sizing: border-box;
                }

                body {
                    font-family: 'Inter', -apple-system, BlinkMacSystemFont,
'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue',
sans-serif;
                    background: linear-gradient(135deg, var(--background-light)
0%, #e9ecef 100%);
```

```
                        color: var(--text-dark);
                        line-height: 1.6;
                        min-height: 100vh;
                        display: flex;
                        justify-content: center;
                        align-items: center;
                        padding: 2rem;
                    }

                    .container {
                        width: 100%;
                        max-width: 1200px;
                        background: white;
                        border-radius: var(--border-radius);
                        box-shadow: var(--box-shadow);
                        overflow: hidden;
                        transition: all 0.3s ease;
                        perspective: 1000px;
                    }

                    h1 {
                        text-align: center;
                        padding: 2rem 0;
                        background: linear-gradient(to right, var(--primary-color),
var(--secondary-color));
                        color: white;
                        margin-bottom: 1rem;
                        font-weight: 600;
                        letter-spacing: 1px;
                        text-transform: uppercase;
                        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
                    }

                    .header {
                        display: flex;
                        justify-content: flex-start;
                        padding: 1rem;
                    }

                    .search-container {
                        flex: 1;
                        text-align: left;
                        margin-right: 20px;
                    }

                    .search-input {
                        padding: 10px;
                        width: 100%;
                        max-width: 400px;
                        border: 1px solid #ccc;
                        border-radius: 6px;
                        font-size: 16px;
                    }

                    table {
                        width: 100%;
                        border-collapse: separate;
                        border-spacing: 0 10px;
                        padding: 0 20px;
                    }

                    th, td {
                        padding: 15px;
                        text-align: left;
```

```
        transition: all 0.3s ease;
    }

    th {
        background-color: var(--primary-color); color:
        white;
        text-transform: uppercase;
        font-weight: 600;
        letter-spacing: 1px;
        position: sticky;
        top: 0;
        z-index: 10;
    }

    tr {
        background-color: white;
        border-radius: var(--border-radius);
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.05);
        margin 0;
        -bottom: 10px;
        transition: all 0.3s ease;
    }

    tr:hover {
        transform: scale(1.02) translateY(-5px); box-
        shadow: 0 10px 20px rgba(0, 0, 0, 0.1);
    }

    td {
        color: var(--text-dark);
        border-bottom: 1px solid #ecf0f1;
    }

    .pdf-link {
        display: inline-flex;
        align-items: center;
        color: var(--primary-color);
        text-decoration: none;
        font-weight: 500;
        transition: all 0.3s ease;
        padding: 5px 10px;
        border-radius: 6px;
    }

    .pdf-link:hover {
        background-color: rgba(52, 152, 219, 0.1); color:
        var(--secondary-color);
        transform: translateX(5px);
    }

    .pdf-icon {
        width: 20px;
        height: 20px;
        margin-right: 8px;
        filter: drop-shadow(0 2px 4px rgba(0,  0, 0, 0.2));
    }

    @media screen and (max-width: 768px) {
        .container {
            margin: 1rem;
            padding: 0;
        }

        table {
```

```
                        font-size: 14px; padding:
                        0 10px;
                    }

                    th, td {
                        padding: 10px;
                    }
                }

                @keyframes fadeIn {
                    from { opacity: 0; transform: translateY(20px); } to
                    { opacity: 1; transform: translateY(0); }
                }

                tr {
                    animation: fadeIn 0.5s ease backwards;
                }

                tr:nth-child(even) {
                    animation-delay: 0.1s;
                }

                tr:nth-child(odd) {
                    animation-delay: 0.2s;
                }
            </style>
            <script>
                function searchBooks() {
                    const input =
document.getElementById('isbnSearch').value.toLowerCase();
                    const rows = document.querySelectorAll('table tbody tr');

                    rows.forEach(row => {
                        const isbnCell = row.cells[2].textContent.toLowerCase(); if
                        (isbnCell.includes(input)) {
                            row.style.display = '';
                        } else {
                            row.style.display = 'none';
                        }
                    });
                }
            </script>
        </head>
        <body>
            <div class="container">
                <h1>Library Books Collection</h1>
                <div class="header">
                    <div class="search-container">
                        <input type="text" id="isbnSearch" class="search-input"
placeholder="Search by ISBN..." onkeyup="searchBooks()">
                    </div>
                </div>
                <table>
                    <thead>
                        <tr>
                            <th>Title</th>
                            <th>Author</th>
                            <th>ISBN</th>
                            <th>PDF</th>
                        </tr>
                    </thead>
                    <tbody>
            """
```

```
        for row in rows:
            title, author, isbn, pdf_path = row if
            pdf_path:
                pdf_link = f'<a class="pdf-link"
href="file://{os.path.abspath(pdf_path)}" target="_blank"><img
src="https://img.icons8.com/ios-filled/50/000000/pdf-2.png" class="pdf-
icon"/>View PDF</a>'
            else:
                pdf_link = "No PDF available"

            html_content += f"""
            <tr>
                <td>{title}</td>
                <td>{author}</td>
                <td>{isbn}</td>
                <td>{pdf_link}</td>
            </tr>
            """

        html_content += """
                    </tbody>
                </table>
            </div>
        </body>
        </html>
        """

        html_file_path = "books_list.html" with
        open(html_file_path, "w") as file:
            file.write(html_content)

        webbrowser.open(f"file://{os.path.abspath(html_file_path)}")

    def load_books_from_db(self): conn
        = self.connect_db() cursor =
        conn.cursor()
        cursor.execute("SELECT title, author, isbn, pdf_path FROM books") rows =
        cursor.fetchall()

        for row in rows:
            new_book = Book(row[0], row[1], row[2], row[3]) if
            not self.head:
                self.head = new_book
            else:
                current = self.head
                while current.next:
                    current = current.next
                current.next = new_book

        conn.close()

    def update_book(self, isbn, title, author, new_isbn): conn
        = self.connect_db()
        cursor = conn.cursor()
        cursor.execute("UPDATE books SET title=?, author=?, isbn=? WHERE isbn=?",
                       (title, author, new_isbn, isbn))
        conn.commit()
        conn.close()
        messagebox.showinfo("Success", f'Book with ISBN "{isbn}" updated
successfully.')

    def get_book_by_isbn(self, isbn):
        conn = self.connect_db()
        cursor = conn.cursor()
```

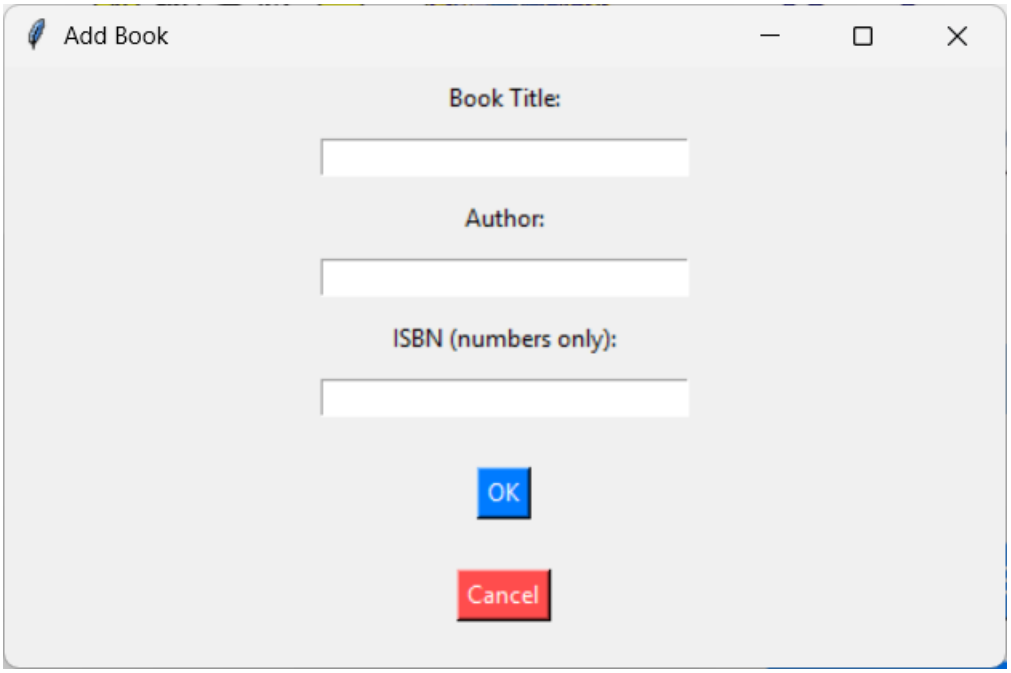
```
        cursor.execute("SELECT title, author, isbn FROM books WHERE isbn=?",
(isbn,))
        book = cursor.fetchone() conn.close()
        return book if book else None


class LibraryApp:
    def __init__(self, root): self.library
        = Library()
        self.library.load_books_from_db()

        self.window = root
        self.window.title("LIBRARY MANAGEMENT SYSTEM")
        self.window.geometry("1920x1080")
        self.window.config(bg="#f0f0f0")

        self.create_widgets()

    def create_widgets(self):
        original_image = Image.open("ks.png")
        resized_image = original_image.resize((1495, 200), Image.LANCZOS) self.image
        = ImageTk.PhotoImage(resized_image)

        self .image_label = tk.Label(self.window, image=self.image, bg="#f0f0f0")
        self.image_label.grid(row=0, column=0, columnspan=2, padx=20, pady=20)

        self.title_label = tk.Label(self.window, text="LIBRARY MANAGEMENT
SYSTEM", font=("Helvetica", 36, "bold"),
                                    bg="#f0f0f0", fg="#333")
        self.title_label.grid(row=1, column=0, columnspan=2, pady=20)

        button_frame = tk.Frame(self.window, bg="#f0f0f0") button_frame.grid(row=2,
        column=0, columnspan=2, pady=20)


        self.create_button(button_frame, "Add Book", self.add_book, 0, 0)
        self.create_button(button_frame, "Delete Book", self.delete_book, 0, 1)
        self.create_button(button_frame, "View Books", self.library.view_books,
1, 0)
        self.create_button(button_frame, "Upload PDF", self.upload_pdf, 1, 1)
        self.create_button(button_frame, "View PDF", self.view_pdf, 2, 0)
        self.create_button(button_frame, "Update Book", self.update_book, 2, 1)
        undo_button = tk.Button(button_frame, text="Undo Last Operation",
command=self.library.undo, width=20,
                                bg="#007bff", fg="white", font=("Helvetica", 14))
        undo_button.grid(row=3, column=0, columnspan=2, padx=20, pady=10) exit_button
        = tk.Button(button_frame, text="Exit",
command=self.window.quit, width=20, bg="#ff4d4d", fg="white",
                                font=("Helvetica", 14))
        exit_button.grid(row=4, column=0, columnspan=2, padx=20, pady=20)

    def create_button(self, parent, text, command, row, column):
        button = tk.Button(parent, text=text, command=command, width=20,
bg="#007bff", fg="white", font=("Helvetica", 14))
        button.grid(row=row, column=column, padx=20, pady=10) button.bind("<Enter>",
        lambda e: button.config(bg="#0056b3")) button.bind("<Leave>", lambda e:
        button.config(bg="#007bff"))

    def add_book(self):
        add_book_window = tk.Toplevel(self.window)
        add_book_window.title("Add Book")
        add_book_window.geometry("500x300")
        add_book_window.config(bg="#f0f0f0")
```

```
        tk.Label(add_book_window, text="Book Title:", bg="#f0f0f0").pack(pady=5)
        title_entry = tk.Entry(add_book_window, width=30) title_entry.pack(pady=5)

        tk.Label(add_book_window, text="Author:", bg="#f0f0f0").pack(pady=5)
        author_entry = tk.Entry(add_book_window, width=30) author_entry.pack(pady=5)

        tk.Label(add_book_window, text="ISBN (numbers only):",
bg="#f0f0f0").pack(pady=5)

        def validate_isbn_input(P):
            if P == "" or P.isdigit():
                return True
            else:
                return False

        validate_isbn = add_book_window.register(validate_isbn_input) isbn_entry

        = tk.Entry(add_book_window, width=30, validate="key",
validatecommand=(validate_isbn, '%P'))
        isbn_entry.pack(pady=5)

        def on_ok():
            title = title_entry.get()
            author = author_entry.get()
            isbn = isbn_entry.get()
            if title and author and isbn:
                self.library.add_book(title, author, isbn)
                add_book_window.destroy()

        ok_button = tk.Button(add_book_window, text="OK", command=on_ok,
bg="#007bff", fg="white")
        ok_button.pack(pady=20)

        cancel_button = tk.Button(add_book_window, text="Cancel",
command=add_book_window.destroy, bg="#ff4d4d", fg="white")
        cancel_button.pack(pady=5)

    def delete_book(self):
        isbn = simpledialog.askstring("Input", "Enter book ISBN to delete:") if
        isbn:
            self.library.delete_book(isbn)

    def upload_pdf(self):
        isbn = simpledialog.askstring("Input", "Enter book ISBN to upload PDF:") if
        isbn:
            self.library.upload_pdf(isbn)

    def view_pdf(self):
        isbn = simpledialog.askstring("Input", "Enter book ISBN to view PDF:") if
        isbn:
            self.library.view_pdf(isbn)

    def update_book(self):
        isbn = simpledialog.askstring ("Input", "Enter book ISBN to update:") if
        isbn:
            book = self.library.get_book_by_isbn(isbn) if
            book:
                self.update_book_dialog(isbn, book) else:
                messagebox.showerror("Error", "Book with this ISBN not found.") def

    update_book_dialog(self, isbn, book):
```

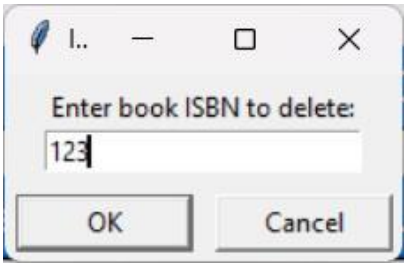
```
        update_book_window = tk.Toplevel(self.window)
        update_book_window.title("Update Book") update_book_window.geometry("500x300")
        update_book_window.config(bg="#f0f0f0")

        tk.Label(update_book_window, text="Book Title:",
bg="#f0f0f0").pack(pady=5)
        title_entry = tk.Entry(update_book_window, width=30)
        title_entry.insert(0, book[0]) title_entry.pack(pady=5)

        tk.Label(update_book_window, text="Author:", bg="#f0f0f0").pack(pady=5)
        author_entry = tk.Entry(update_book_window, width=30) author_entry.insert(0,
        book[1])
        author_entry.pack(pady=5)

        tk.Label(update_book_window, text="ISBN (numbers only):",
bg="#f0f0f0").pack(pady=5)

        def validate_isbn_input(P):
            if P == "" or P.isdigit():
                return True
            else:
                return False

        validate_isbn = update_book_window.register(validate_isbn_input) isbn_entry =

        tk.Entry(update_book_window, width=30, validate="key",
validatecommand=(validate_isbn, '%P'))
        isbn_entry.insert(0, book[2])
        isbn_entry.pack(pady=5)

        def on_ok():
            title = title_entry.get()
            author = author_entry.get()
            new_isbn = isbn_entry.get()
            if title and author and new_isbn:
                self.library.update_book(isbn, title, author, new_isbn)
                update_book_window.destroy()
            else:
                messagebox.showwarning("Input Error", "Please fill all fields.")

        ok_button = tk.Button(update_book_window, text="OK", command=on_ok,
bg="#007bff", fg="white")
        ok_button.pack(pady=20)

        cancel_button = tk.Button(update_book_window, text="Cancel",
command=update_book_window.destroy, bg="#ff4d4d", fg="white")
        cancel_button.pack(pady=5)


if __name__== "__main__": root
    = tk.Tk()
    app = LibraryApp(root)
    root.mainloop()
```

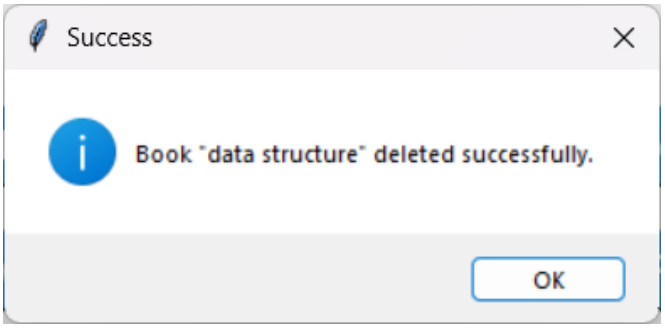
SCREENSHOT:

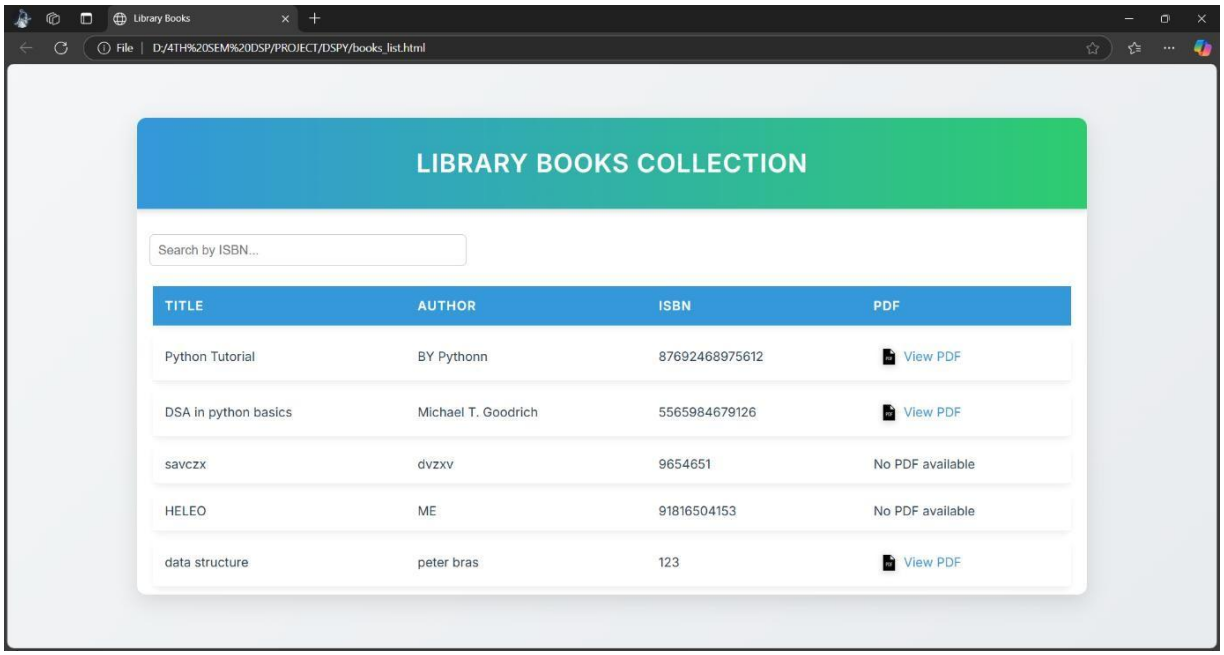

LIBRARY MANAGEMENT SYSTEM
Kammavari Sangham (R) 1952, K. S. Group of Institutions
K. S. POLYTECHNIC
No.15, Mallasandra, Off. Kanakapura Road, Bengaluru-560109, +919606064189
Approved by AICTE, New-Delhi and Recognized by DTE, Govt. of Karnataka
KSP
K S POLYTECHNIC
AICTE
LIBRARY MANAGEMENT SYSTEM
Add Book
Delete Book
View Books
Upload PDF
View PDF
Update Book
Undo Last Operation
Exit

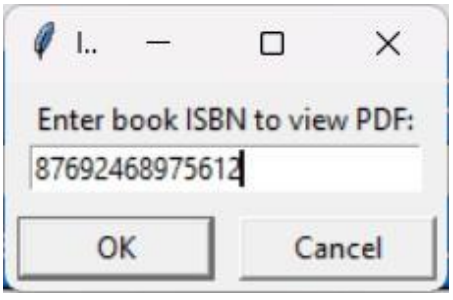

Add Book
Book Title:
Author:
ISBN (numbers only):
OK
Cancel

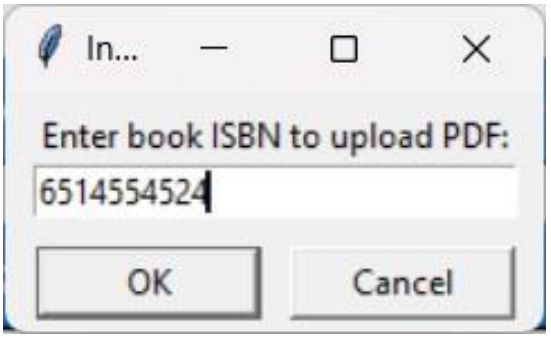

Success
Book "DS" added successfully.
OK

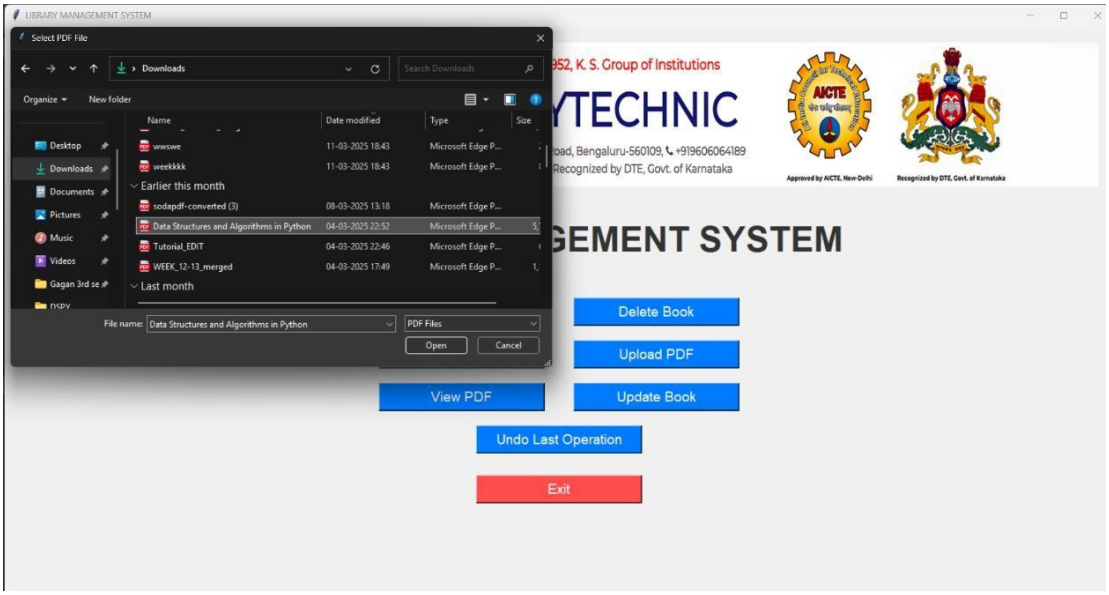

I..
Enter book ISBN to delete:
123
OK
Cancel

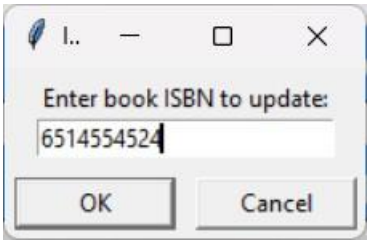

Success
Book "data structure" deleted successfully.
OK

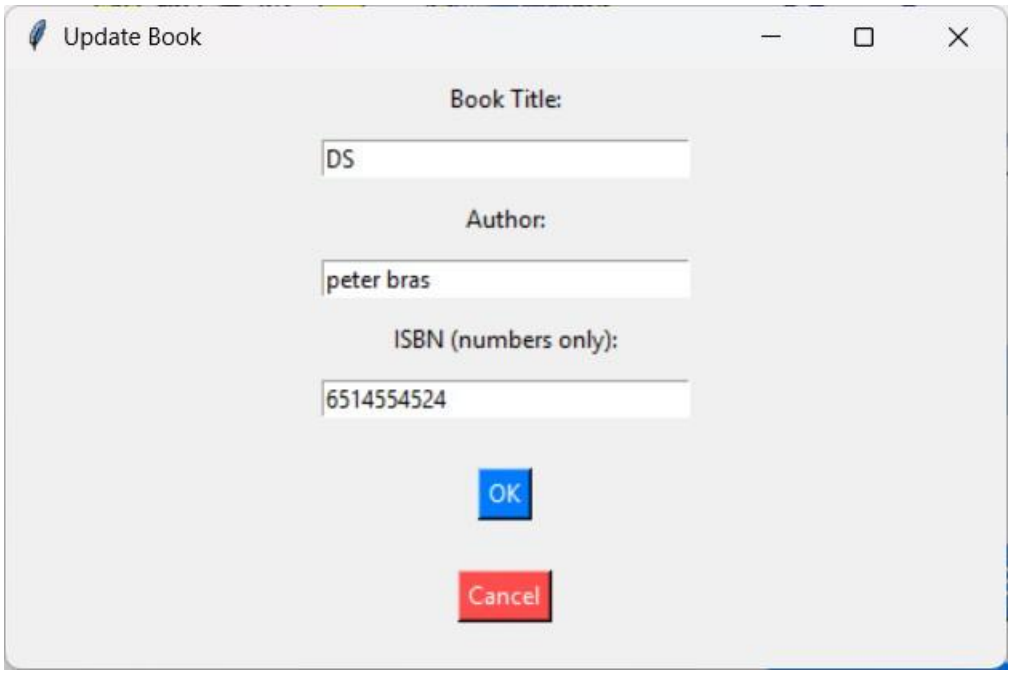

Library Books
File | D:/4TH%20SEM%20DSP/PROJECT/DSPY/books_list.html
LIBRARY BOOKS COLLECTION
Search by ISBN...
TITLE
AUTHOR
ISBN
PDF
Python Tutorial
BY Pythonn
87692468975612
View PDF
DSA in python basics
Michael T. Goodrich
5565984679126
View PDF
savczx
dvzxv
9654651
No PDF available
HELEO
ME
91816504153
No PDF available
data structure
peter bras
123
View PDF


I..
Enter book ISBN to view PDF:
87692468975612
OK
Cancel


In...
Enter book ISBN to upload PDF:
6514554524
OK
Cancel


LIBRARY MANAGEMENT SYSTEM
Select PDF File
Downloads
Search Downloads
Organize
New folder
Name
Date modified
Type
Size
Desktop
Downloads
Documents
Pictures
Music
Videos
Gagan 3rd se
wwswe
11-03-2025 18:43
Microsoft Edge P...
weekkkk
11-03-2025 18:43
Microsoft Edge P...
Earlier this month
sodapdf-converted (3)
08-03-2025 13:18
Microsoft Edge P...
Data Structures and Algorithms in Python
04-03-2025 22:52
Microsoft Edge P...
Tutorial_EDIT
04-03-2025 22:46
Microsoft Edge P...
WEEK_12-13_merged
04-03-2025 17:49
Microsoft Edge P...
Last month
File name: Data Structures and Algorithms in Python
PDF Files
Open
Cancel
952, K. S. Group of Institutions
YTECHNIC
oad, Bengaluru-560109, +919606064189
Recognized by DTE, Govt. of Karnataka
Approved by AICTE, New Delhi
Recognized by DTE, Govt. of Karnataka
GEMENT SYSTEM
Delete Book
Upload PDF
View PDF
Update Book
Undo Last Operation
Exit


I..
Enter book ISBN to update:
6514554524
OK
Cancel


Update Book
Book Title:
DS
Author:
peter bras
ISBN (numbers only):
6514554524
OK
Cancel


Success
Book with ISBN "6514554524" updated successfully.
OK

ADVANTAGES OF LIBRARY MANAGEMENT SYSTEM

1.Dynamic Memory Allocation:
- No need for pre-defined memory size; the linked list can dynamically grow or shrink as the number of books changes.

2.Efficient Insertion and Deletion:
- Adding/removing books is efficient, especially at the beginning or end, as it does not require shifting elements like in an array.

3.Memory Efficiency:
- Memory is allocated only as needed, and no memory is wasted, unlike fixed-size arrays.

4.Flexible Data Structure:
- Different types of linked lists (singly, doubly) can be used based on system needs, allowing for flexible traversal and operations.

5.No Wasted Space:
- The system doesn't reserve extra space, making it ideal for a system where the number of books may vary over time.

DISADVANTAGES OF LIBRARY MANAGEMENT SYSTEM

1.Increased Memory Usage:
- Each node requires extra memory for storing pointers (next or previous), which increases memory usage compared to arrays.

2.Sequential Access:
- Searching or accessing a specific book requires traversing the list from the beginning, making it slower (O(n)O(n)O(n)) compared to direct access in arrays.

3.Complex Implementation:
- Linked lists are more complex to implement and manage, requiring careful pointer handling to avoid errors.

4.No Random Access:
- Unlike arrays, linked lists do not allow direct indexing, which makes accessing specific books slower.

5.Fragmentation:
- The memory allocation is dynamic, potentially causing fragmentation in the system over time.

FUTURE ENHANCEMENT

1.Person authentication and role control.

- set up a at ease login mechanism that accommodates various person types

2. Advanced search and filtering options.

- improve the hunt feature to permit customers to discover books based totally on a couple of standards, which includes title, writer, and isbn moreover, incorporate filtering abilities to assist users in refining their search effects with the aid of deciding on unique categories, guide years, or availability popularity

3.The e book category and tagging system is a way used to categorize and prepare books based totally on their content material and problem be counted.

- implement a system that permits customers to categorize books into specific genres or tags, improving employer and making it simpler for customers to find out books that suit their hobbies

4.The process of managing book borrowing and returns.

- create a machine that video display units the borrowing and returning of books, keeping song of every e-book's status and keeping person bills to log borrowing history

5. Automatic signals and activates.

- expand a notification gadget that informs customers approximately upcoming due dates for borrowed books and indicators them while a asked e book is ready for checkout

6. The software includes a feature that enables customers to access and view pdf documents instantly within the utility.

- allow users to view pdf documents directly inside the software, getting rid of the need to open them in an external web browser, doubtlessly by using utilizing an included pdf viewer

7. Database backup and restoration skills.

- introduce alternatives for backing up the database and restoring it in case of facts loss, which include functionalities for exporting and uploading statistics files

8. Facts analytics and reporting equipment.

- offer analytical insights into e book utilization styles, which includes figuring out the most frequently borrowed titles, popular authors, and developments in person pastime, that could useful resource in effective stock control

9. A user-friendly interface design that is appealing to nine people.

- don't forget growing a cell-responsive model of the application or ensuring that the existing layout is optimized to be used on smartphones and pills

10. Improvements to the user interface.

- revamp the person interface with modern design factors, animations, and advanced navigation to elevate the general consumer enjoy

11. Integration with online bookstores.

- facilitate customers' ability to search for books in online bookstores or libraries, offering direct links for purchasing or borrowing options.

CONCLUSION

The establishment of the library management system (lms) signifies a significant advancement in the organization and interaction of library collections and user engagements. This project has successfully created a comprehensive solution that tackles long-standing issues faced by libraries while incorporating modern technology to enhance user experience and streamline operations.

The Library Management System (LMS) includes essential features like user authentication and role management, which ensure secure access to confidential information. Its enhanced search and filtering capabilities allow users to quickly locate books, while the classification and tagging system enhances the organization of the library's inventory. Furthermore, the borrowing and return management system enables precise tracking of book statuses and user accounts, along with automated alerts that notify users about due dates and availability.

A major focus has been given to user experience, with a design that adjusts to different devices, guaranteeing accessibility for a broad spectrum of users. The enhancements made to the user interface increase engagement, and adherence to accessibility standards guarantees that individuals with disabilities can navigate the system without any difficulties.

Looking ahead, there are numerous possibilities for further improvements, such as incorporating online bookstore features, establishing a system for tracking overdue items, and introducing user feedback mechanisms. These advancements can enhance library operations and foster a sense of community among its users.

In conclusion, the library management system has the potential to transform libraries into dynamic community hubs that provide not only books but also a wide range of information and resources specifically designed to cater to the diverse needs of their patrons. By embracing technological advancements and nurturing a culture of innovation, libraries can remain relevant and significant in an ever-evolving world, fulfilling their crucial role in promoting knowledge, education, and literacy for all.