Mini Project Report(KCS752) on

Advance Library Management System

Submitted in partial fulfilment for award of

BACHELOR OF TECHNOLOGY

Degree

In

COMPUTER SCIENCE & ENGINEERING

2024-25

Submitted By: Vansh Kabaria (2100330100245)

Under the Guidance of: Mr. Sachin Shah

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

RAJ KUMAR GOEL INSTITUTE OF TECHNOLOGY

DELHI-MEERUT ROAD, GHAZIABAD

Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow

TABLE OF CONTENTS

CHAPTER-1

INTRODUCTION—ENVIRONMENT AND TOOLS USED:

The **Advanced Library Management System** is a comprehensive solution designed to automate and manage the core operations of a library. Unlike traditional library management systems that focus primarily on basic functionalities such as book cataloging, issuing, and returning, this advanced system incorporates several modern technologies to enhance efficiency, accuracy, and user experience.

Objectives

The primary goal of this project is to provide a seamless and automated library management experience, addressing not only the traditional needs of book management but also integrating new-age technologies to meet the evolving demands of modern libraries. This system aims to:

- Simplify and automate routine tasks like book issuing, returning, and inventory management.
- Integrate QR codes for unique identification and tracking of books.
- Provide an intuitive and user-friendly interface for librarians and administrators.
- Enhance data security and error handling.
- Generate comprehensive reports on library operations.

Key Features

This system goes beyond the traditional library management systems by incorporating the following advanced features:

- **QR Code Integration**: Each book in the library is assigned a unique QR code, which simplifies the process of tracking and verifying book details. This feature also ensures that book issuance and returns are handled efficiently.
- **Fine Management**: Automated fine calculation for overdue books, ensuring that penalties are managed systematically.
- **Book History Retention**: A comprehensive history of all issued and returned books is maintained, including timestamps and fine details, which helps in tracking and auditing.
- **Search and Sort Functionality**: Users can easily search for books and sort them based on various criteria, making it easier to manage large book inventories.

- **PDF Export Functionality**: Users can export data, such as book records and issued book history, into PDF format for reporting and documentation purposes.

This advanced system is designed with the future of library management in mind. By incorporating QR codes, automated fine management, and comprehensive history tracking, it offers significant improvements over traditional systems. These enhancements not only reduce manual work for librarians but also provide a more accurate and reliable system for managing library resources.

1.1 Technologies Used:

1. Python (Tkinter)

Explanation:

Python is a high-level, versatile programming language known for its readability and ease of use. In your Advanced Library Management System, you used Python to implement the core logic and functionality of the application. Python's extensive standard library and third-party packages allowed you to handle various tasks like database connectivity, file handling, and more. Python's simplicity and flexibility make it an ideal choice for both beginners and experienced developers.

Tkinter Details:

Tkinter is Python's standard GUI (Graphical User Interface) library. It provides tools to create windows, dialogs, buttons, labels, and other graphical elements that make up the user interface. Tkinter is lightweight, easy to use, and comes bundled with Python, eliminating the need for separate installation. It uses a widget-based approach, where each GUI element is a widget that can be customized and placed within the application window. You used Tkinter to create the interface for your library management system, providing a user-friendly way for librarians to interact with the system.

2. MySQL

Explanation:

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to manage and manipulate data. It is known for its reliability, ease of use, and performance, making it one of the most popular databases worldwide. In your project, MySQL is used to store and manage all the data related to books, users, issued books, fines, and history.

MySQL Details:

MySQL organizes data into tables, each containing rows and columns, making it easy to model real-world entities and their relationships. It supports a wide range of SQL operations like querying, inserting, updating, and deleting data. MySQL also provides security features such as user authentication and permissions, ensuring that only authorized users can access or modify data. Its scalability allows it to handle large datasets efficiently, making it suitable for your library management system, where the database might grow as more books and users are added.

3. PyMySQL

Explanation:

PyMySQL is a Python library that allows you to connect to and interact with a MySQL database. It serves as the interface between your Python code and the MySQL database, enabling you to execute SQL queries, fetch results, and handle database transactions directly from your Python application.

PyMySQL Details:

PyMySQL provides a straightforward API for connecting to a MySQL database. It supports various operations like establishing a connection, executing queries, and fetching data. In your project, PyMySQL is used to interact with the database for tasks like adding new books, issuing books, updating records, and fetching data for display in the GUI. It also supports error handling, ensuring that any issues with database operations are caught and managed appropriately. PyMySQL's flexibility allows you to perform complex queries and transactions, making it a crucial part of your library management system.

4. qrcode Module

Explanation:

The qrcode module in Python is used to generate QR codes, which are two-dimensional barcodes that store information in a compact, scannable format. In your library management system, QR codes are generated for each book, containing details like the book's title, author, and unique identifier, simplifying the process of book tracking and verification.

qrcode Module Details:

The qrcode module provides an easy-to-use interface for generating QR codes in Python. It allows you to encode information into a QR code, customize its size, color, and error correction level, and save it as an image file. QR codes are widely used for their ability to store a large amount of information in a small space, and they can be scanned using smartphones or dedicated QR code scanners. In your project, QR codes are used to uniquely identify each book, streamlining the process of issuing and returning books by simply scanning the QR code rather than manually entering book details.

5. PIL (Python Imaging Library)

Explanation:

PIL, or the Python Imaging Library, is a library that adds image processing capabilities to your Python interpreter. It allows you to open, manipulate, and save many different image file formats. In your project, PIL is used to handle images, particularly for displaying and saving QR codes generated for books.

PIL Details:

PIL, now known as Pillow (a fork of the original PIL), supports a wide range of image file formats, including JPEG, PNG, BMP, and GIF. It provides tools for basic image processing tasks like resizing, cropping, rotating, and converting images. In your library management system, PIL is used to process and display QR code images. When a QR code is generated, PIL can save it as an image file, which can then be displayed in the application or printed for physical labeling. PIL's simplicity and flexibility make it a powerful tool for image handling in your project.

6. Datetime Module

Explanation:

The datetime module in Python provides classes for manipulating dates and times. It is used in your project to handle various date and time-related tasks, such as recording the issue and return dates of books, calculating fines, and generating timestamps for actions.

Datetime Module Details:

The datetime module provides a variety of classes, such as datetime, date, time, and timedelta, which allow you to work with dates and times in a precise and efficient manner. It supports operations like calculating the difference between two dates, formatting dates and times, and converting between different time zones. In your library management system, the datetime module is used to track when books are issued and returned, calculate overdue periods, and apply fines accordingly. It also helps in generating accurate timestamps for logging activities, ensuring that all actions are properly recorded with their respective dates and times.

7. PDF Generation

Explanation:

PDF generation in Python can be handled using libraries like ReportLab or FPDF, which allow you to create PDF files programmatically. In your library management system, PDF generation is used to export data such as book records, issued book history, and reports, making it easy to share and print this information.

PDF Generation Details:

Libraries like ReportLab provide a rich set of tools for creating PDFs from scratch. You can define the layout, add text, images, tables, and other elements, and customize the formatting. In your project, PDF generation is used to create reports that can be shared with librarians, administrators, or other stakeholders. For example, after searching for books or viewing issued book history, users can export the data into a PDF format, which is ideal for documentation or printing purposes. PDF files are widely used due to their portability and consistent formatting across different devices, making them a valuable addition to your system.

8. Error Handling with try-except

Explanation:

Error handling in Python is managed using try-except blocks, which allow you to catch and manage exceptions that may occur during the execution of your code. In your library management system, error handling is crucial for ensuring that the application runs smoothly, even when unexpected issues arise, such as database connection failures or invalid user inputs.

Error Handling Details:

A try-except block in Python allows you to test a block of code for errors (try), and handle the error if one occurs (except). This prevents the program from crashing and allows you to provide user-friendly error messages or take corrective actions. In your project, error handling is implemented throughout the code to manage issues like database errors, file handling errors, or invalid data entries. For example, when connecting to the database, a try-except block can catch connection errors and prompt the user to check their database settings. This makes your application more robust and reliable, as it can gracefully handle errors without disrupting the user experience.

CHAPTER 2
HARDWARE AND SOFTWARE REQUIREMENTS

1. Hardware Requirements

Minimum Hardware Requirements:

- **Processor**: Intel Core i3 or equivalent (2.0 GHz)
- **RAM**: 4 GB
- **Hard Disk**: 500 GB HDD
- **Monitor**: 15-inch display (1024x768 resolution)
- **Input Devices**: Keyboard and Mouse
- **QR Code Scanner (Optional)**: For scanning book QR codes (if hardware scanning is used)
- **Printer (Optional)**: For printing reports and QR codes
- **Internet Connection**: For downloading software dependencies and updates (optional if local setup is sufficient)

Recommended Hardware Requirements:

- **Processor**: Intel Core i5 or equivalent (3.0 GHz or higher)
- **RAM**: 8 GB or more
- **Hard Disk**: 1 TB HDD or SSD for faster performance
- **Monitor**: 19-inch display (1280x1024 resolution or higher)
- **Input Devices**: Keyboard and Mouse (Ergonomic)
- **QR Code Scanner (Optional)**: Dedicated barcode/QR code scanner for efficient book management
- **Printer (Optional)**: High-speed printer for generating reports and labels
- **Internet Connection**: High-speed broadband for updates, cloud integration, and online database management (if applicable)

---

2. Software Requirements

Operating System:

- **Minimum**:

    - Windows 7 or higher
    - Linux (Ubuntu 18.04 or higher)
    - macOS 10.13 or higher

- **Recommended**:
    - Windows 10 or 11
    - Latest stable versions of Linux distributions like Ubuntu 20.04 LTS or Fedora
    - macOS 11 Big Sur or higher

Software Components:

1. **Python**:
    - **Version**: Python 3.7 or higher (Recommended: Python 3.10+)
    - **Purpose**: Core programming language for building the application.
    - **Installation**: Ensure Python is installed with Tkinter, PIP, and other required packages.
2. **Tkinter**:
    - **Purpose**: Default GUI toolkit for Python, used for creating the application interface.
    - **Installation**: Comes pre-installed with Python; no separate installation required.
3. **MySQL Server**:
    - **Version**: MySQL 5.7 or higher (Recommended: MySQL 8.0+)
    - **Purpose**: Database management system for storing library data.
    - **Installation**: Install the MySQL server and MySQL Workbench for database management.
4. **PyMySQL**:
    - **Version**: Latest stable version via PIP.
    - **Purpose**: A Python library for connecting to the MySQL database.
    - **Installation**: Install using pip install PyMySQL.
5. **qrcode Module**:
    - **Version**: Latest stable version via PIP.
    - **Purpose**: Used to generate QR codes for books.

   - **Installation**: Install using pip install qrcode[pil].

6. **Pillow (PIL)**:
   - **Version**: Latest stable version via PIP.
   - **Purpose**: Python Imaging Library (Pillow) for handling images, including QR code generation.
   - **Installation**: Install using pip install Pillow.
7. **Datetime Module**:
   - **Purpose**: Python standard library for handling date and time operations.
   - **Installation**: Comes with Python; no separate installation required.
8. **PDF Generation Library** (e.g., ReportLab or FPDF):
   - **Version**: Latest stable version via PIP.
   - **Purpose**: For exporting data and reports as PDFs.
   - **Installation**: Install using pip install reportlab or pip install fpdf.
9. **Web Browser** (Optional):
   - **Purpose**: For viewing online documentation, downloading resources, or integrating with online services.
10. **Text Editor/IDE**:
    - **Minimum**: Notepad++ or Sublime Text.
    - **Recommended**: Visual Studio Code, PyCharm, or any full-featured Python IDE.
    - **Purpose**: For writing, editing, and debugging Python code.
11. **Git** (Optional):
    - **Version**: Latest stable version.
    - **Purpose**: Version control system for managing and tracking code changes.
    - **Installation**: Install using package manager (e.g., apt-get install git for Linux, or download from git-scm.com for Windows/macOS).
12. **PDF Viewer**:
    - **Purpose**: For viewing generated PDF reports.
    - **Examples**: Adobe Acrobat Reader, Foxit Reader.

Additional Software (Optional):

- **XAMPP/WAMP/LAMP (Optional for MySQL)**:
    - A local server environment to run MySQL if you prefer using a pre-packaged environment.
- **MySQL Workbench**:
    - **Purpose**: GUI tool for managing MySQL databases, useful for visual database management.

Network Requirements (Optional):

- **LAN/Wi-Fi Network**: For multi-user or networked installations where the database might be hosted on a central server, allowing multiple clients to access it.

CHAPTER 3 APPLICATION ARCHITECTURE

The architecture of your Advanced Library Management System can be described as a multi-tiered, modular structure that separates different functionalities into distinct layers. This approach ensures scalability, maintainability, and ease of understanding. The architecture generally consists of three main layers: Presentation Layer, Business Logic Layer, and Data Layer.

---

1. Presentation Layer (User Interface)

Purpose:

This layer is responsible for interacting with the user. It includes the graphical user interface (GUI) elements and handles user inputs and outputs.

Components:

- **Tkinter-based GUI:** The interface is built using Python's Tkinter library, providing users with a window-based environment to interact with the system.
  - **Main Dashboard:** Displays different options such as viewing books, issuing books, returning books, managing users, etc.
  - **Forms and Dialogs:** For adding new books, issuing books, managing users, etc.

  - **QR Code Integration:** Users can scan QR codes for faster book identification.
  - **PDF Export:** Allows users to export data into PDF format, such as book lists or issued book history.
  - **Search and Sort Functionalities:** Enhances user experience by allowing them to quickly find and organize information.

Interaction:

- The Presentation Layer interacts with the Business Logic Layer to process user commands (e.g., adding a book, issuing a book) and display the results.
- The GUI captures user inputs (like book details or student IDs) and sends them to the Business Logic Layer for processing.

---

2. Business Logic Layer (Application Logic)

Purpose:

This layer contains the core logic and functionalities of the application. It processes user inputs, performs operations, and communicates with the Data Layer to retrieve or store information.

Components:

- **Book Management:** Functions for adding, updating, deleting, and viewing books. This includes logic for generating unique book serial numbers and QR codes.
- **User Management:** Handles librarian and admin functionalities, including adding, deleting, and managing user access.
- **Book Issue and Return:** Manages the issuing and returning of books. This includes updating the database and managing fines for late returns.
- **Fine Calculation:** Implements logic for calculating fines based on the return date and due date.
- **Report Generation:** Generates reports in PDF format for various operations like book inventory, issued book history, etc.
- **Validation and Error Handling:** Ensures that user inputs are validated before processing and handles any exceptions that might occur.

Interaction:

- The Business Logic Layer communicates with the Data Layer to fetch or update data in the database.
- It acts as a middleman between the Presentation Layer and the Data Layer, ensuring that user actions result in appropriate data operations.

---

3. Data Layer (Database Management)

Purpose:

This layer is responsible for managing the storage, retrieval, and updating of data within the system. It uses a MySQL database to store all the necessary information related to books, users, issued books, and other relevant details.

Components:

- **MySQL Database:** The central storage for all the data, including tables for books, users, issued book history, fines, etc.
  - Books Table: Stores book details like serial number, title, author, subject, and quantity.
  - Users Table: Stores information about librarians, admins, and possibly students.
  - Issued Book History Table: Tracks all issued books, including issue date, return date, fines, and timestamps.
  - Fines Table: Tracks fines associated with late returns.
- **PyMySQL Connector:** This component acts as the interface between your Python application and the MySQL database. It handles executing SQL queries, fetching results, and managing database connections.

Interaction:

- The Data Layer provides data to the Business Logic Layer when requested (e.g., fetching a list of all books) and updates the database based on operations performed in the Business Logic Layer (e.g., issuing a book or returning a book).

4. Optional Components

QR Code Generator:

- Functionality: Generates QR codes for books using the qrcode Python module and saves them as image files with the help of the Pillow library.
- Usage: Integrates with both the Presentation Layer (displaying QR codes) and the Business Logic Layer (storing QR code information in the database).

PDF Generator:

- Functionality: Allows exporting of various data into PDF format using libraries like ReportLab or FPDF.
- Usage: Can be invoked from the GUI to generate printable reports of book inventories, issued book history, etc.

Flow of Operations

1. **User Interaction (Presentation Layer):**
   - The user interacts with the GUI, for example, by clicking the "Issue Book" button or entering details in a form.
2. **Business Logic Execution (Business Logic Layer):**
   - The input from the user is processed by the business logic. For instance, the system checks if the book is available, calculates the due date, and updates the issued book history.

3. **Database Operations (Data Layer):**
    - The business logic then communicates with the MySQL database via PyMySQL to update the records, such as marking a book as issued and recording the user who borrowed it.
4. **Feedback to User (Presentation Layer):**
    - Once the operation is complete, the GUI is updated with the result, such as confirming that the book has been issued or displaying an error message if something went wrong.

Benefits of the Architecture

- **Separation of Concerns:** Each layer focuses on a specific aspect of the system, making it easier to manage and maintain.
- **Scalability:** The architecture can be scaled easily by adding more features to the Business Logic Layer or expanding the database schema.
- **Maintainability:** Modular design allows for easy updates and debugging, as changes in one layer usually don't affect the others.
- **Reusability:** Components like the PDF generator, QR code generator, and MySQL connector can be reused in other projects or extended further.
- **Security:** By managing database connections and queries through PyMySQL, the architecture ensures that sensitive operations are handled securely.

CHAPTER 4

PROJECT MODULES DESIGN

1. User Management Module

Purpose:

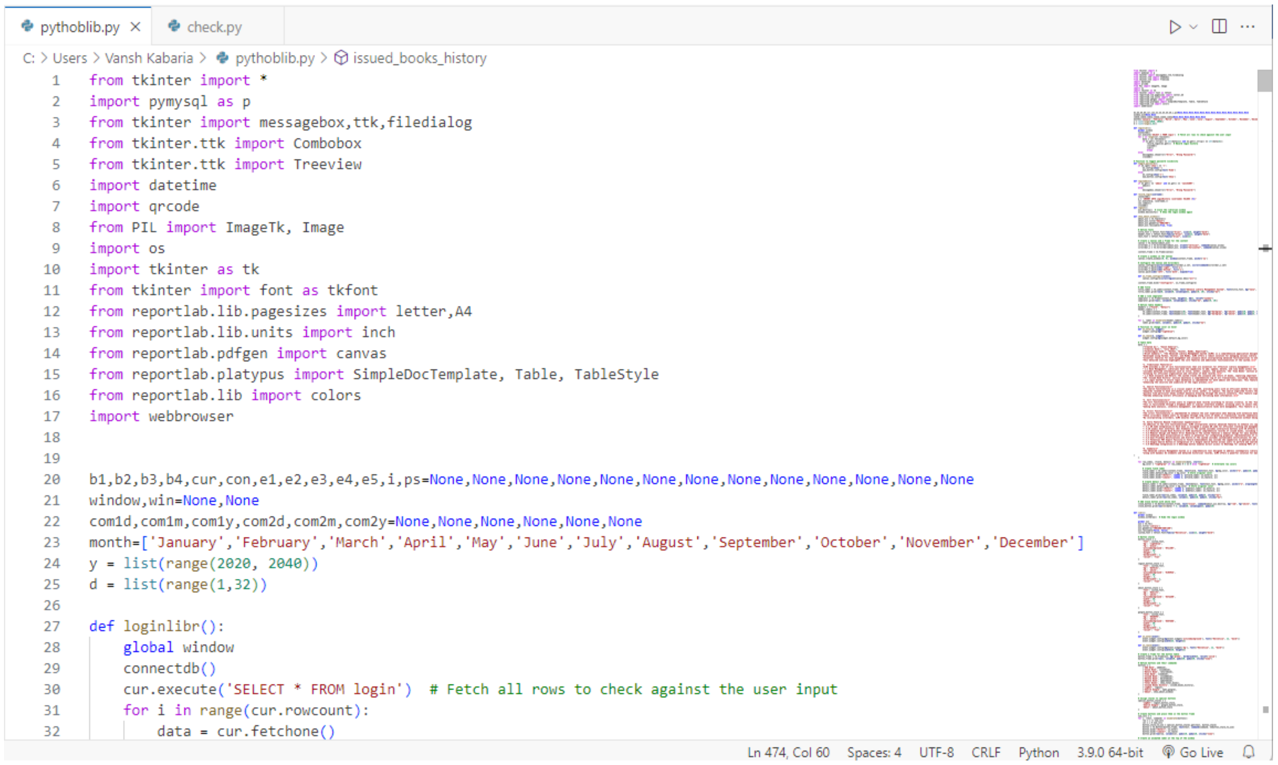
This module handles user-related operations, including authentication, authorization, and management of user accounts.

Key Features:

- **Admin Login**: Allows administrators to log in and manage the system.
- **Librarian Login**: Provides librarians with access to the library management functionalities after successful login.
- **User CRUD Operations**: Admin can create, read, update, and delete librarian accounts.

Functions:

- admin_login(): Verifies admin credentials and grants access to the admin dashboard.
- librarian_login(): Authenticates librarians and grants access to librarian-specific functionalities.
- add_librarian(): Allows the admin to add new librarian accounts.
- delete_librarian(): Enables the admin to remove librarian accounts.


pythoblib.py × check.py

C: > Users > Vansh Kabaria > pythoblib.py > issued_books_history

```python
from tkinter import *
import pymysql as p
from tkinter import messagebox,ttk,filedialog
from tkinter.ttk import Combobox
from tkinter.ttk import Treeview
import datetime
import qrcode
from PIL import ImageTk, Image
import os
import tkinter as tk
from tkinter import font as tkfont
from reportlab.lib.pagesizes import letter,A4
from reportlab.lib.units import inch
from reportlab.pdfgen import canvas
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
from reportlab.lib import colors
import webbrowser


b1,b2,b3,b4,cur,con,e1,e2,e3,e4,e5,i,ps=None,None,None,None,None,None,None,None,None,None,None,None,None
window,win=None,None
com1d,com1m,com1y,com2d,com2m,com2y=None,None,None,None,None,None
month=['January','February','March','April','May','June','July','August','September','October','November','December']
y = list(range(2020, 2040))
d = list(range(1,32))

def loginlibr():
    global window
    connectdb()
    cur.execute('SELECT * FROM login')  # Fetch all rows to check against the user input
    for i in range(cur.rowcount):
        data = cur.fetchone()
```

Ln 474, Col 60   Spaces: 4   UTF-8   CRLF   Python   3.9.0 64-bit   Go Live

2. Book Management Module

Purpose:

This module manages the library's collection of books, including adding, updating, deleting, and viewing book records.

Key Features:

- **Add Book**: Adds new books to the library with details like title, author, and subject.
- **Update Book**: Allows for editing book details such as title and quantity.
- **Delete Book**: Removes books from the library's database.
- **View Books**: Displays a list of books with options to search and sort.

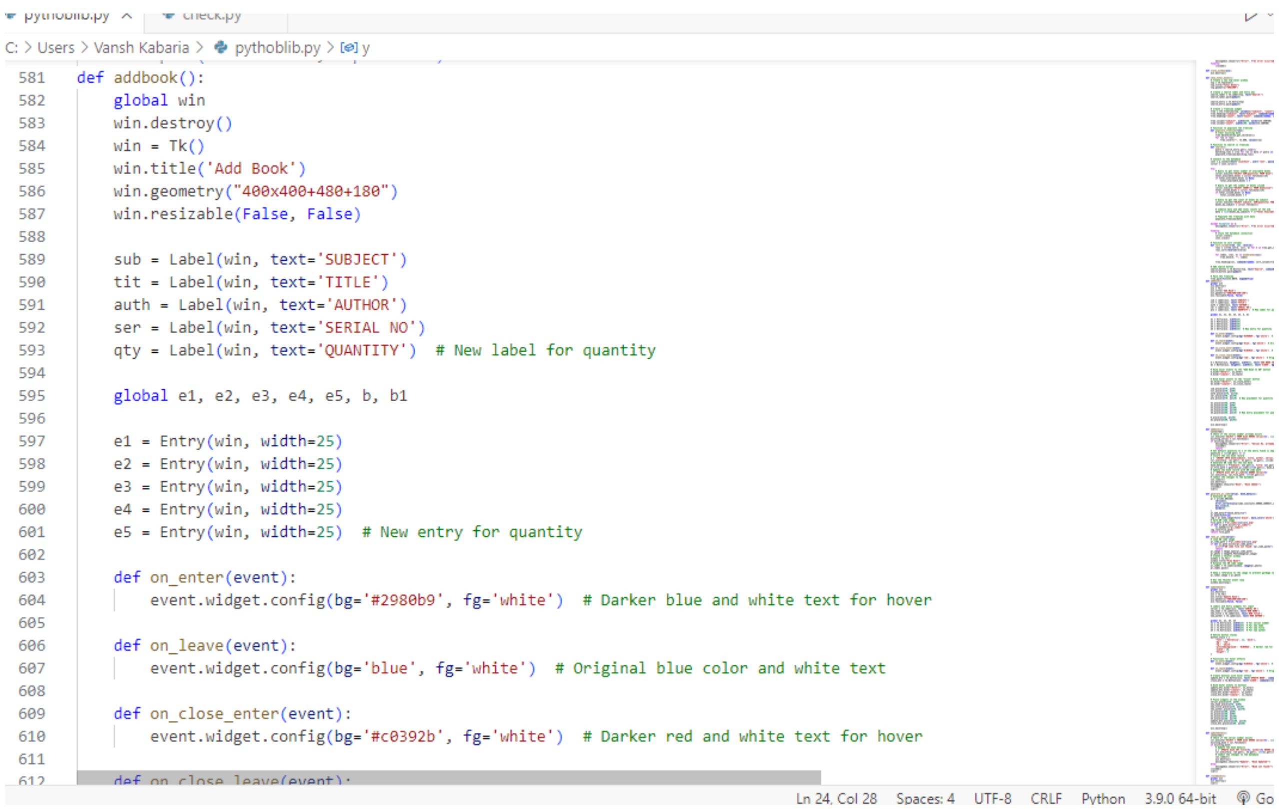
Functions:

- add_book(): Inserts a new book record into the database and generates a unique serial number and QR code.

- update_book(): Updates existing book details in the database.

- delete_book(): Deletes a book record from the database.

- view_books(): Retrieves and displays all books from the database with search and sort options.



C: > Users > Vansh Kabaria > pythoblib.py > y

```python
def addbook():
    global win
    win.destroy()
    win = Tk()
    win.title('Add Book')
    win.geometry("400x400+480+180")
    win.resizable(False, False)

    sub = Label(win, text='SUBJECT')
    tit = Label(win, text='TITLE')
    auth = Label(win, text='AUTHOR')
    ser = Label(win, text='SERIAL NO')
    qty = Label(win, text='QUANTITY')  # New label for quantity

    global e1, e2, e3, e4, e5, b, b1

    e1 = Entry(win, width=25)
    e2 = Entry(win, width=25)
    e3 = Entry(win, width=25)
    e4 = Entry(win, width=25)
    e5 = Entry(win, width=25)  # New entry for quantity

    def on_enter(event):
        event.widget.config(bg='#2980b9', fg='white')  # Darker blue and white text for hover

    def on_leave(event):
        event.widget.config(bg='blue', fg='white')  # Original blue color and white text

    def on_close_enter(event):
        event.widget.config(bg='#c0392b', fg='white')  # Darker red and white text for hover

    def on_close_leave(event):
```

Ln 24, Col 28   Spaces: 4   UTF-8   CRLF   Python   3.9.0 64-bit

3. Book Issue/Return Module

Purpose:

This module manages the issuing and returning of books, including tracking which books have been borrowed and by whom.

Key Features:

- **Issue Book**: Allows librarians to issue books to students, updating the database accordingly.

- **Return Book**: Manages the return of books, updating the system and calculating any applicable fines.

- **Fine Calculation**: Automatically calculates fines for overdue books.

- **Issued Book History**: Maintains a history of all issued and returned books.

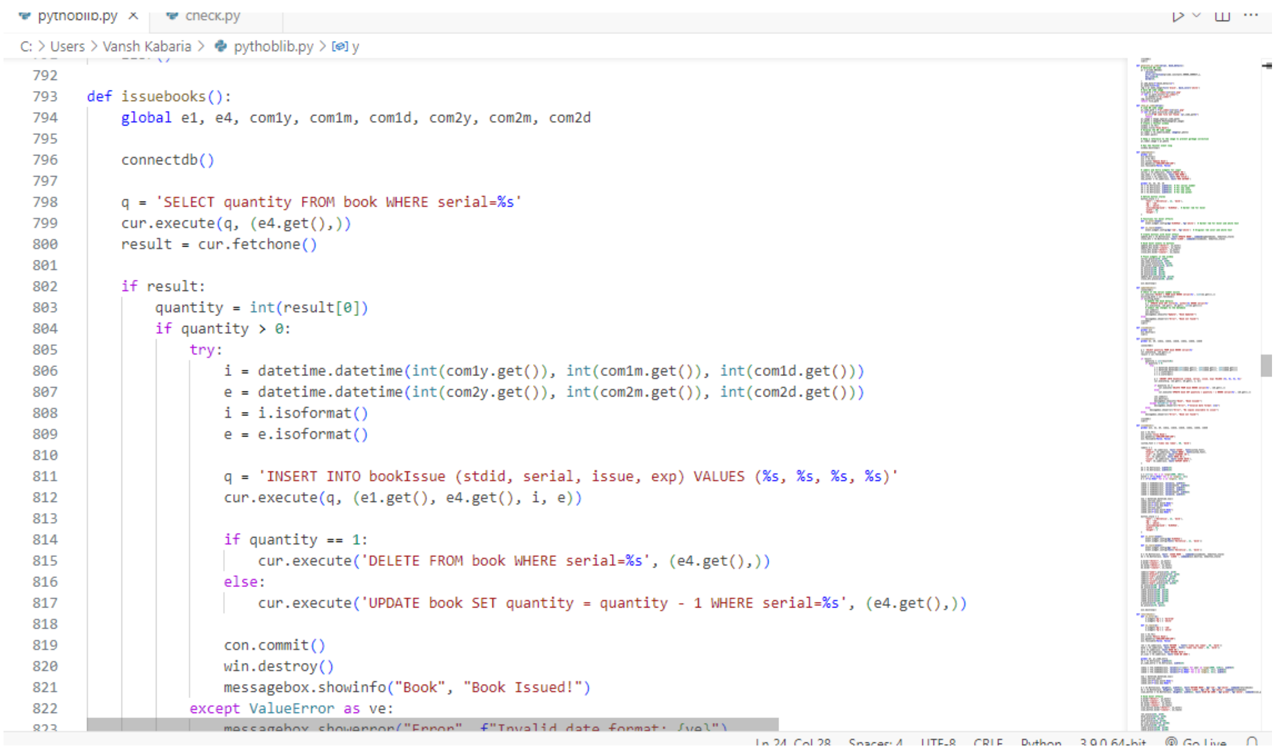
Functions:

- issue_book(): Issues a book to a student and records the transaction in the database.

- return_book(): Processes the return of a book and updates the issued book history.

- calculate_fine(): Calculates any fines due for overdue books based on the return date.

- view_issued_books(): Displays a list of currently issued books and their details.


pythoblib.py ×
check.py
C: > Users > Vansh Kabaria > pythoblib.py > y
Ln 24, Col 28 Spaces: 4 UTF-8 CRLF Python 3.9.0 64-bit Go Live

```python

def issuebooks():
    global e1, e4, com1y, com1m, com1d, com2y, com2m, com2d

    connectdb()

    q = 'SELECT quantity FROM book WHERE serial=%s'
    cur.execute(q, (e4.get(),))
    result = cur.fetchone()

    if result:
        quantity = int(result[0])
        if quantity > 0:
            try:
                i = datetime.datetime(int(com1y.get()), int(com1m.get()), int(com1d.get()))
                e = datetime.datetime(int(com2y.get()), int(com2m.get()), int(com2d.get()))
                i = i.isoformat()
                e = e.isoformat()

                q = 'INSERT INTO bookIssue (stdid, serial, issue, exp) VALUES (%s, %s, %s, %s)'
                cur.execute(q, (e1.get(), e4.get(), i, e))

                if quantity == 1:
                    cur.execute('DELETE FROM book WHERE serial=%s', (e4.get(),))
                else:
                    cur.execute('UPDATE book SET quantity = quantity - 1 WHERE serial=%s', (e4.get(),))

                con.commit()
                win.destroy()
                messagebox.showinfo("Book", "Book Issued!")
            except ValueError as ve:
                messagebox.showerror("Error", f"Invalid date format: {ve}")
```

4. QR Code Integration Module

Purpose:

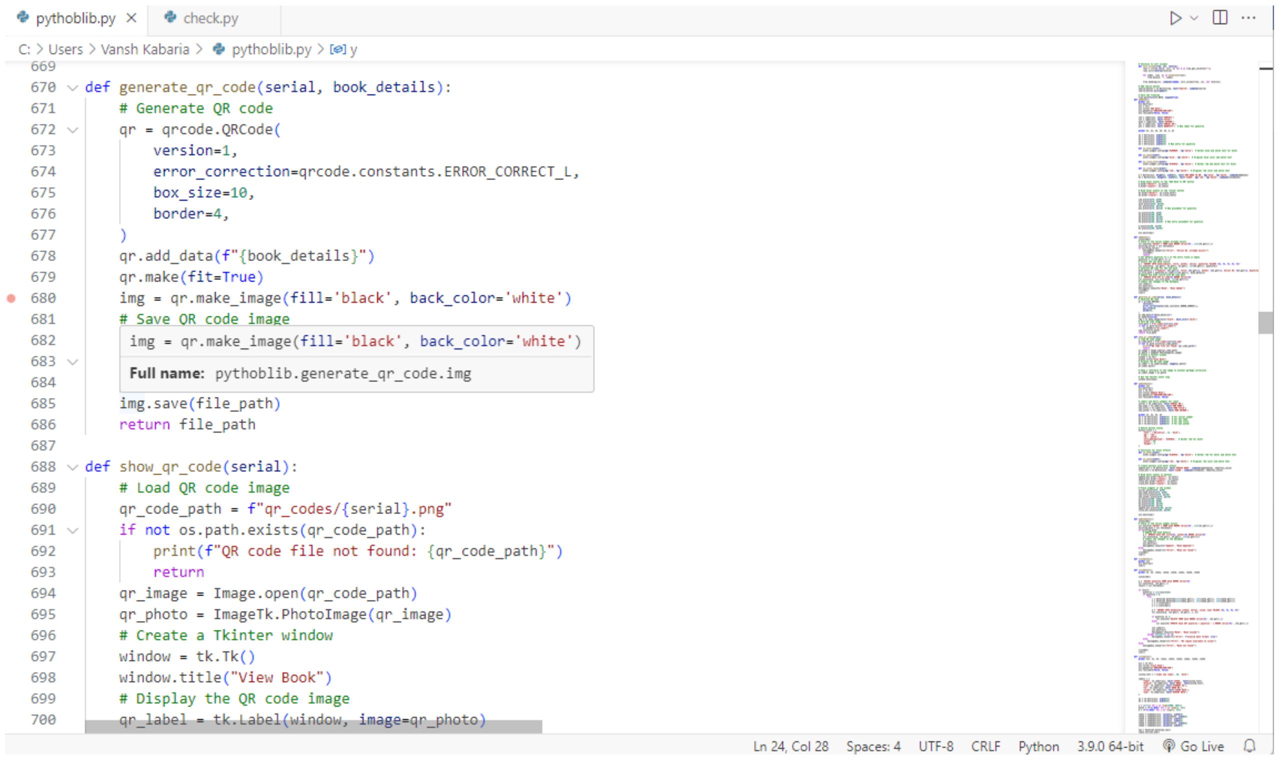
This module handles the generation and management of QR codes for books, allowing for quick identification and verification.

Key Features:

- **QR Code Generation**: Automatically generates QR codes for each book when it is added to the system.
- **QR Code Display**: Displays the QR code in the book details section for easy access.
- **QR Code Scanning**: (Optional) Allows for scanning of QR codes to quickly retrieve book details.

Functions:

- generate_qr_code(book_id): Generates a QR code for a book using its unique serial number or ID.
- display_qr_code(book_id): Displays the QR code in the GUI when viewing book details.



```python
def generate_qr_code(serial, book_details):
    # Generate QR code
    qr = qrcode.QRCode(
        version=1,
        error_correction=qrcode.constants.ERROR_CORRECT_L,
        box_size=10,
        border=4,
    )
    qr.add_data(f"{book_details}")
    qr.make(fit=True)
    img = qr.make_image(fill='black', back_color='white')
    # Save QR code image
    img = qr.make_image(fill='black', back_color='white')
    Full name: pythoblib.generate_qr_code.img
    img.save(file_path)
    return file_path

def show_qr_code(serial):
    # Load QR code image
    qr_code_path = f"qr_codes/{serial}.png"
    if not os.path.exists(qr_code_path):
        print(f"QR code file not found: {qr_code_path}")
        return
    qr_image = Image.open(qr_code_path)
    qr_photo = ImageTk.PhotoImage(qr_image)
    # Create a Tkinter window
    window = tk.Tk()
    window.title("View Book")
    # Display the QR code image
    qr_label = tk.Label(window, image=qr_photo)
```

5. PDF Report Generation Module

Purpose:

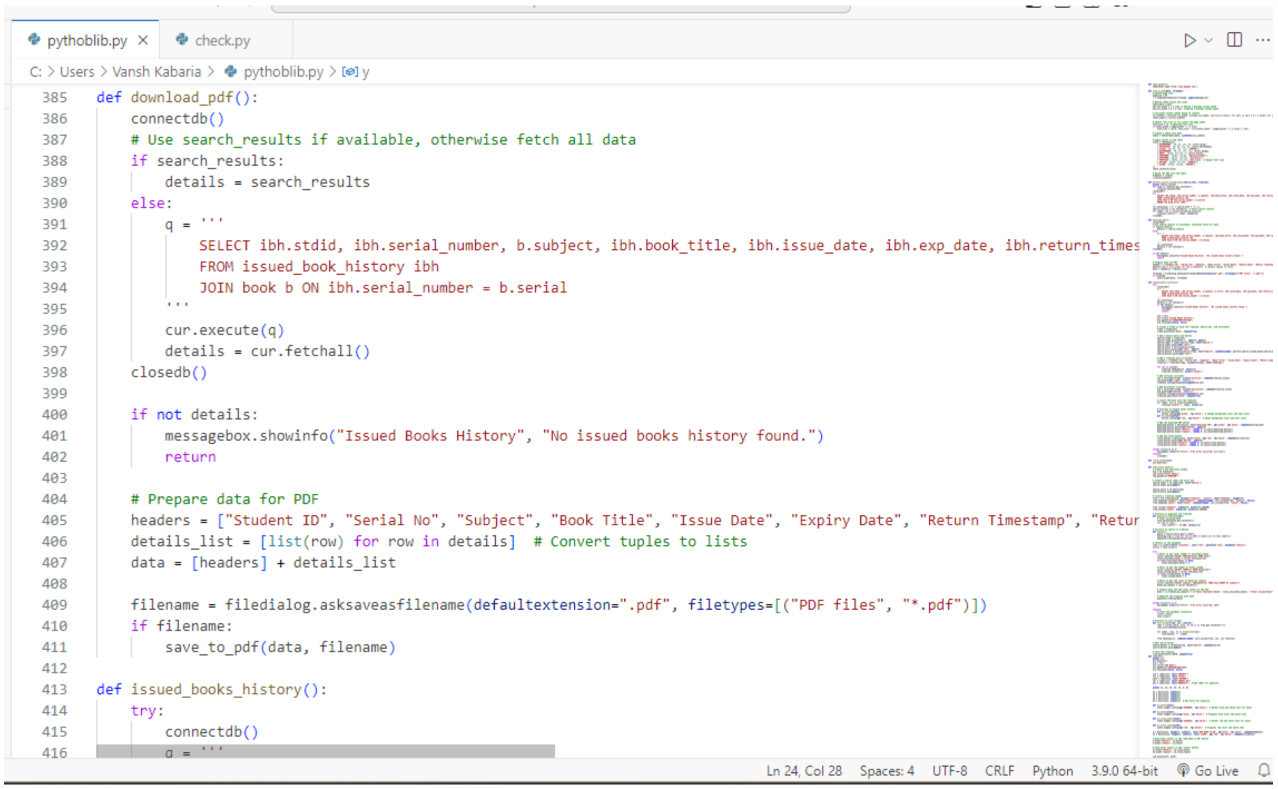
This module provides functionality for generating reports in PDF format, such as book inventories or issued book histories.

Key Features:

- **Generate PDF Reports**: Allows users to export data into PDF format for offline viewing or printing.
- **Customizable Reports**: Users can select which data to include in the reports.

Functions:

- generate_pdf_report(data): Creates a PDF report from the provided data, such as a list of books or issued book history.
- download_pdf(): Provides an option for users to download the generated PDF report.


pythoblib.py
check.py
C: > Users > Vansh Kabaria > pythoblib.py > y
Ln 24, Col 28 Spaces: 4 UTF-8 CRLF Python 3.9.0 64-bit Go Live

```python
def download_pdf():
    connectdb()
    # Use search_results if available, otherwise fetch all data
    if search_results:
        details = search_results
    else:
        q = '''
            SELECT ibh.stdid, ibh.serial_number, b.subject, ibh.book_title, ibh.issue_date, ibh.exp_date, ibh.return_times
            FROM issued_book_history ibh
            JOIN book b ON ibh.serial_number = b.serial
        '''
        cur.execute(q)
        details = cur.fetchall()
    closedb()

    if not details:
        messagebox.showinfo("Issued Books History", "No issued books history found.")
        return

    # Prepare data for PDF
    headers = ["Student ID", "Serial No", "Subject", "Book Title", "Issue Date", "Expiry Date", "Return Timestamp", "Retur
    details_list = [list(row) for row in details]  # Convert tuples to lists
    data = [headers] + details_list

    filename = filedialog.asksaveasfilename(defaultextension=".pdf", filetypes=[("PDF files", "*.pdf")])
    if filename:
        save_to_pdf(data, filename)

def issued_books_history():
    try:
        connectdb()
        q = '''
```

6. Search and Sort Module

Purpose:

This module enhances the usability of the system by allowing users to search and sort records, making it easier to find specific information.

Key Features:

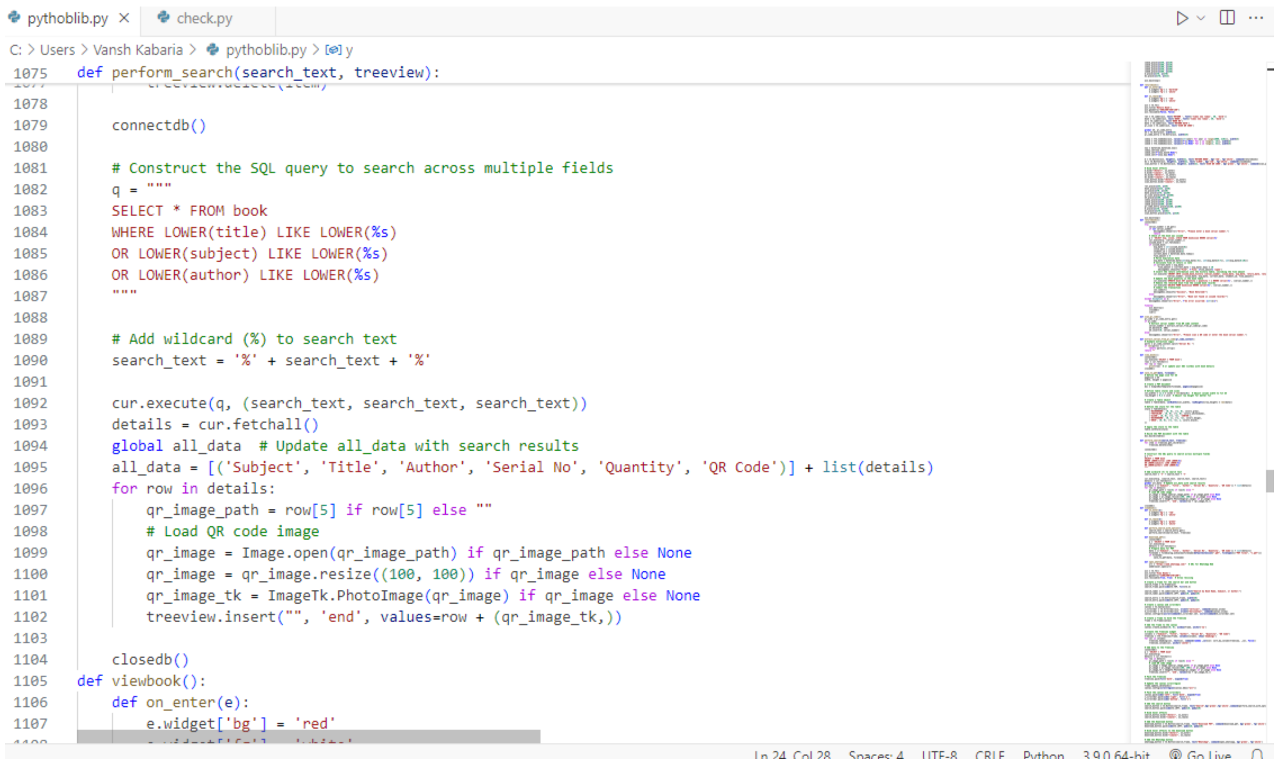
- **Search Functionality**: Allows users to search for books, users, or issued books by various criteria (e.g., title, author, student ID).

- **Sort Functionality**: Enables sorting of records based on different columns (e.g., by title, author, date issued).

Functions:

- search_books(query): Searches the book database for records that match the query.

- sort_books(criteria): Sorts the list of books based on the selected criteria (e.g., alphabetically by title).


pythoblib.py × check.py

C: > Users > Vansh Kabaria > pythoblib.py > [@] y

```python
def perform_search(search_text, treeview):

    connectdb()

    # Construct the SQL query to search across multiple fields
    q = """
    SELECT * FROM book
    WHERE LOWER(title) LIKE LOWER(%s)
    OR LOWER(subject) LIKE LOWER(%s)
    OR LOWER(author) LIKE LOWER(%s)
    """

    # Add wildcard (%) to search text
    search_text = '%' + search_text + '%'

    cur.execute(q, (search_text, search_text, search_text))
    details = cur.fetchall()
    global all_data  # Update all_data with search results
    all_data = [('Subject', 'Title', 'Author', 'Serial No', 'Quantity', 'QR Code')] + list(details)
    for row in details:
        qr_image_path = row[5] if row[5] else ""
        # Load QR code image
        qr_image = Image.open(qr_image_path) if qr_image_path else None
        qr_image = qr_image.resize((100, 100)) if qr_image else None
        qr_image_tk = ImageTk.PhotoImage(qr_image) if qr_image else None
        treeview.insert("", 'end', values=row + (qr_image_tk,))

    closedb()
def viewbook():
    def on_enter(e):
        e.widget['bg'] = 'red'
```

Ln 24, Col 28 Spaces: 4 UTF-8 CRLF Python 3.9.0 64-bit Go Live

---

7. Database Management Module

Purpose:

This module interacts with the MySQL database, managing data storage, retrieval, and updates.

Key Features:

- **Data Persistence**: Ensures that all data (books, users, transactions) is stored in a MySQL database.
- **Secure Data Handling**: Manages connections to the database securely, ensuring data integrity.
- **Error Handling**: Implements error handling to manage database-related exceptions.

Functions:

- connect_to_db(): Establishes a connection to the MySQL database.
- execute_query(query): Executes SQL queries on the database.
- fetch_results(query): Retrieves data from the database based on a query.

---

8. Fine Management Module

Purpose:

This module manages the calculation and tracking of fines for overdue books.

Key Features:

- **Fine Calculation**: Automatically calculates fines based on the number of days a book is overdue.
- **Fine Payment Tracking**: Tracks payments of fines and updates the database accordingly.

Functions:

- calculate_fine(issue_date, return_date): Calculates the fine amount based on the difference between the issue and return dates.
- update_fine_status(student_id, fine_amount): Updates the fine status in the database after payment.

---

9. History Management Module

Purpose:

This module maintains historical records of all transactions, including book issues and returns, along with timestamps.

Key Features:

- **Issued Book History**: Stores a record of all issued books, including the date of issue and return.

- **Activity Logs**: Logs all significant activities in the system, such as user logins and book transactions.

Functions:

- log_activity(user_id, activity): Logs a specific activity performed by a user.

- view_issue_history(): Retrieves and displays the history of issued and returned books.

Module Interaction

- **Presentation Layer**: Interacts with all modules through the user interface, capturing user inputs and displaying results.

- **Business Logic Layer**: Implements the core logic of the system, coordinating between the Presentation Layer and Database Management Module.

- **Database Layer**: Handles data storage and retrieval, ensuring persistence of all records.

CHAPTER 5

PROJECT SNAPSHOT

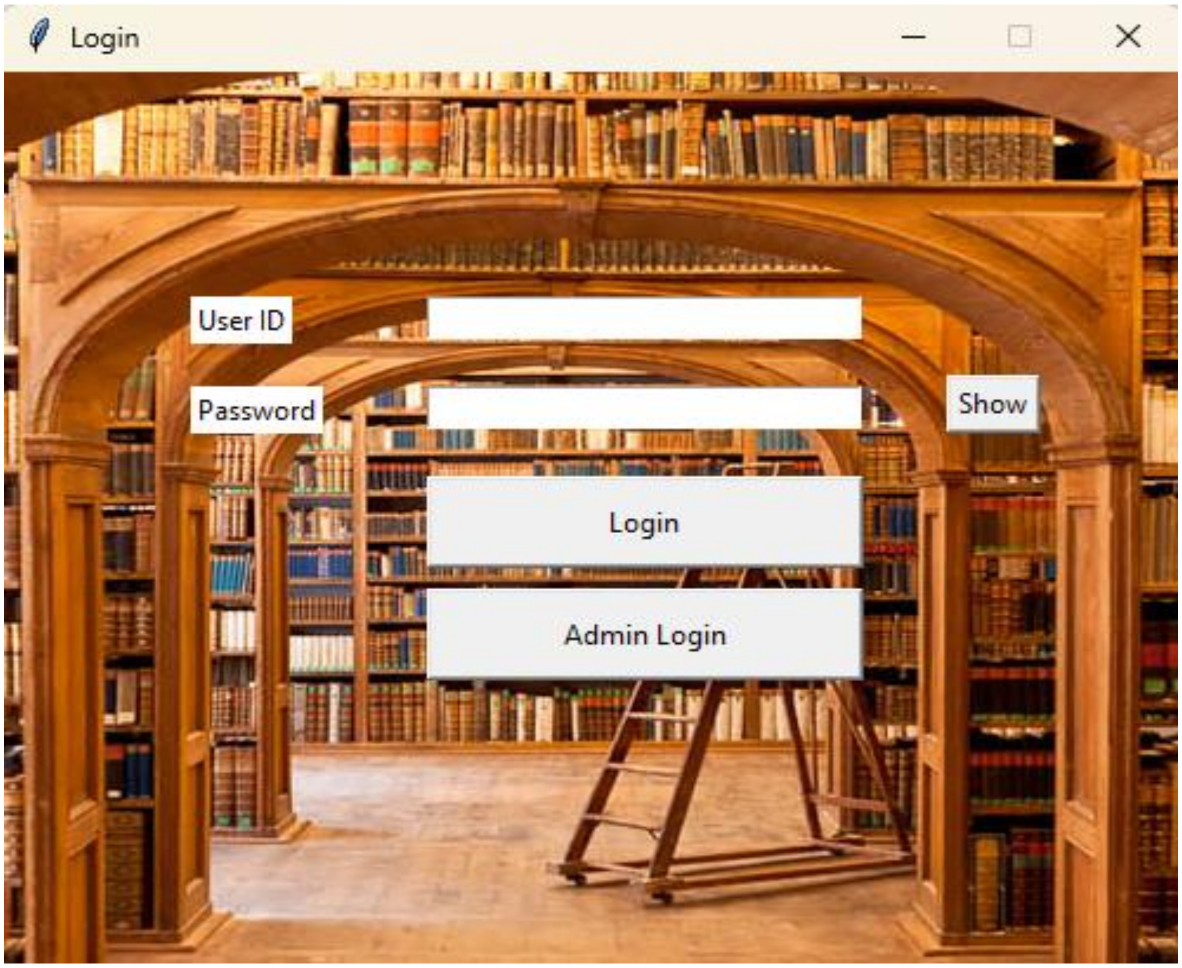


Fig.5.1

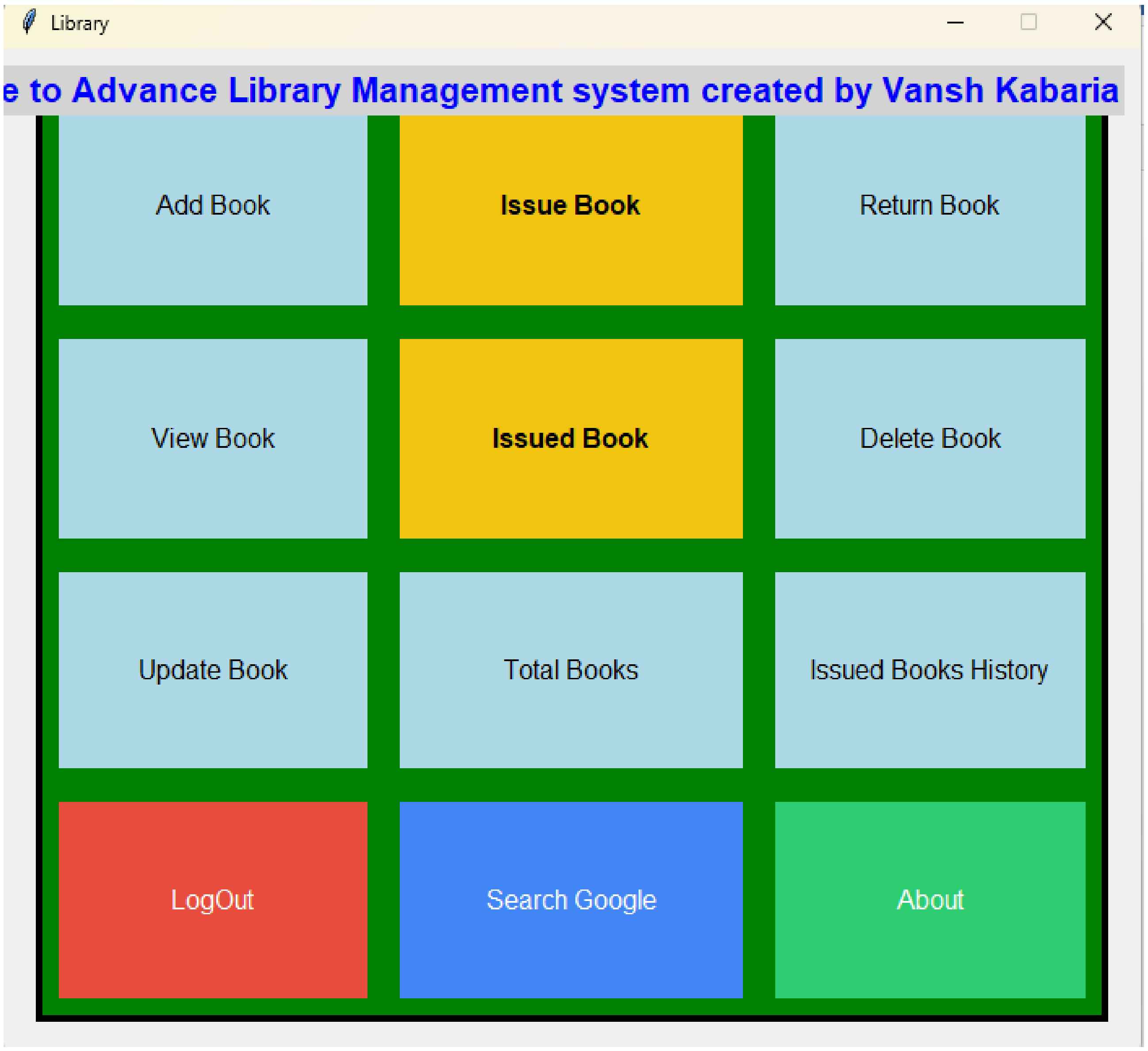
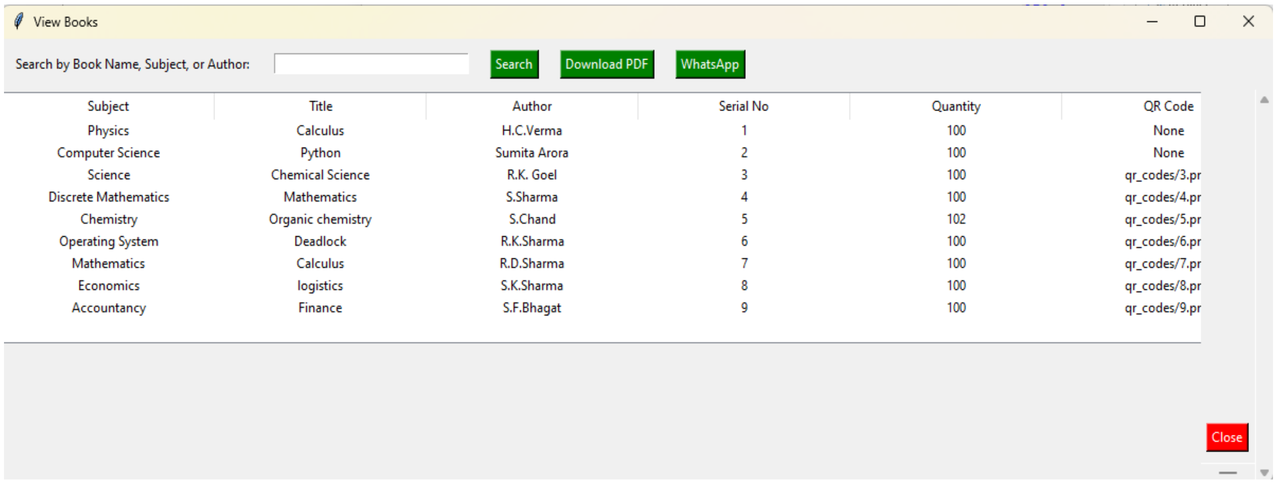

Library
e to Advance Library Management system created by Vansh Kabaria
Add Book
Issue Book
Return Book
View Book
Issued Book
Delete Book
Update Book
Total Books
Issued Books History
LogOut
Search Google
About

Fig. 5.2

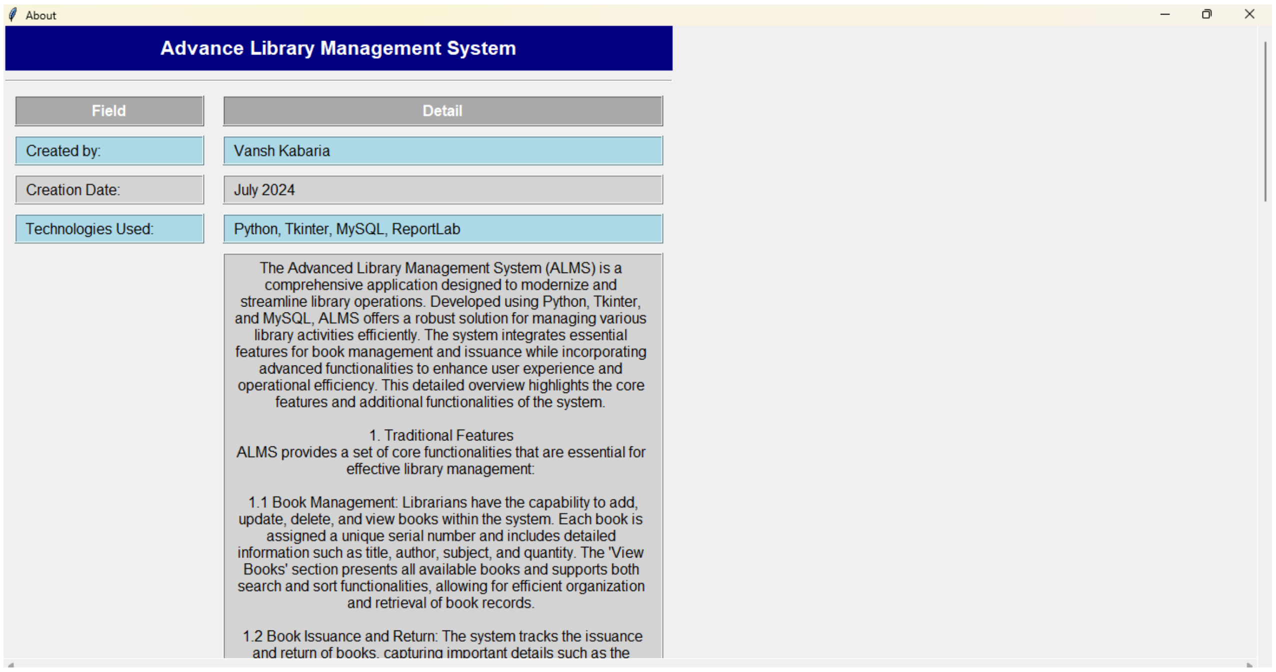


View Books

Search by Book Name, Subject, or Author:

Subject	Title	Author	Serial No	Quantity	QR Code
Physics	Calculus	H.C.Verma	1	100	None
Computer Science	Python	Sumita Arora	2	100	None
Science	Chemical Science	R.K. Goel	3	100	qr_codes/3.pr
Discrete Mathematics	Mathematics	S.Sharma	4	100	qr_codes/4.pr
Chemistry	Organic chemistry	S.Chand	5	102	qr_codes/5.pr
Operating System	Deadlock	R.K.Sharma	6	100	qr_codes/6.pr
Mathematics	Calculus	R.D.Sharma	7	100	qr_codes/7.pr
Economics	logistics	S.K.Sharma	8	100	qr_codes/8.pr
Accountancy	Finance	S.F.Bhagat	9	100	qr_codes/9.pr

Fig.5.3



About

Advance Library Management System

Field	Detail
Created by:	Vansh Kabaria
Creation Date:	July 2024
Technologies Used:	Python, Tkinter, MySQL, ReportLab

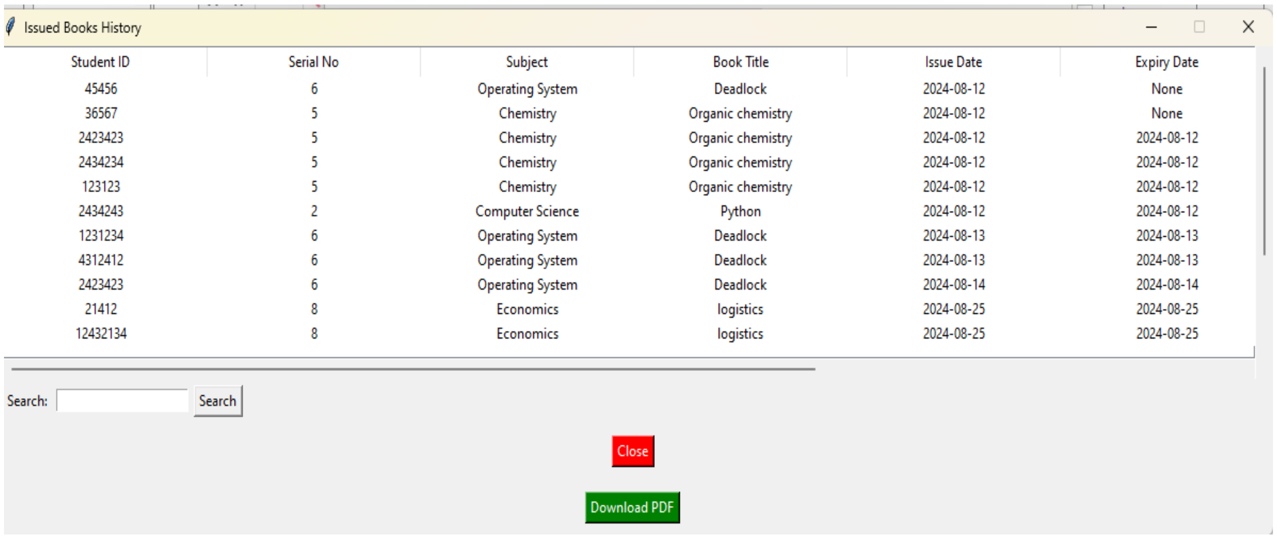
The Advanced Library Management System (ALMS) is a comprehensive application designed to modernize and streamline library operations. Developed using Python, Tkinter, and MySQL, ALMS offers a robust solution for managing various library activities efficiently. The system integrates essential features for book management and issuance while incorporating advanced functionalities to enhance user experience and operational efficiency. This detailed overview highlights the core features and additional functionalities of the system.

1. Traditional Features
ALMS provides a set of core functionalities that are essential for effective library management:

1.1 Book Management: Librarians have the capability to add, update, delete, and view books within the system. Each book is assigned a unique serial number and includes detailed information such as title, author, subject, and quantity. The 'View Books' section presents all available books and supports both search and sort functionalities, allowing for efficient organization and retrieval of book records.

1.2 Book Issuance and Return: The system tracks the issuance and return of books, capturing important details such as the

Fig.5.4



Student ID	Serial No	Subject	Book Title	Issue Date	Expiry Date
45456	6	Operating System	Deadlock	2024-08-12	None
36567	5	Chemistry	Organic chemistry	2024-08-12	None
2423423	5	Chemistry	Organic chemistry	2024-08-12	2024-08-12
2434234	5	Chemistry	Organic chemistry	2024-08-12	2024-08-12
123123	5	Chemistry	Organic chemistry	2024-08-12	2024-08-12
2434243	2	Computer Science	Python	2024-08-12	2024-08-12
1231234	6	Operating System	Deadlock	2024-08-13	2024-08-13
4312412	6	Operating System	Deadlock	2024-08-13	2024-08-13
2423423	6	Operating System	Deadlock	2024-08-14	2024-08-14
21412	8	Economics	logistics	2024-08-25	2024-08-25
12432134	8	Economics	logistics	2024-08-25	2024-08-25

Fig.5.5

CHAPTER 6

LIMITATIONS

Advanced library management systems (ALMS) offer sophisticated functionalities and integrations to manage library operations more efficiently. However, they come with their own set of limitations that organizations should consider:

1. **Complexity of Implementation**

    - **Integration Challenges**: Integrating advanced features such as automated cataloging, digital asset management, and inter-library loan systems can be complex and may require significant customization.

    - **Learning Curve**: Advanced systems often have a steeper learning curve for staff, necessitating extensive training and adaptation periods.

2. **Cost**

    - **High Initial Investment**: The upfront cost of advanced library management systems can be substantial, including software licensing, hardware requirements, and implementation services.

    - **Ongoing Maintenance**: Continuous maintenance, updates, and support can add to the total cost of ownership over time.

3. **Scalability Concerns**

    - **Performance Issues**: As the volume of data and number of users increase, advanced systems may face performance bottlenecks, potentially impacting response times and system efficiency.

- **Resource Intensive**: Advanced systems often require substantial computing resources, which might not be scalable without additional investments in infrastructure.

4. **System Integration**
   - **Compatibility Issues**: Integrating with other existing systems (e.g., financial software, academic databases) may pose compatibility challenges, requiring custom solutions or middleware.
   - **Data Migration**: Migrating data from legacy systems to advanced systems can be complex and prone to data integrity issues.
5. **User Interface and Usability**
   - **Complex User Interfaces**: Advanced features and extensive functionalities can make the user interface complex, potentially overwhelming users who are not tech-savvy.
   - **Customization Needs**: To ensure user-friendliness, significant customization of the user interface may be needed, which can add to the development time and costs.
6. **Security and Privacy**
   - **Data Security**: Advanced systems that handle sensitive user and transactional data need robust security measures. Any vulnerabilities can lead to data breaches and unauthorized access.
   - **Privacy Concerns**: Ensuring compliance with privacy regulations (e.g., GDPR) can be challenging, especially with systems that track extensive user activity and interactions.
7. **Technical Support and Maintenance**

- **Support Availability**: Access to timely and effective technical support can be a concern, particularly if the system is highly specialized or proprietary.
- **System Updates**: Regular updates and patches are necessary to address security vulnerabilities and improve functionality, but they may disrupt normal operations if not managed properly.

8. **Dependence on Technology**
   - **Hardware Dependence**: Advanced systems often require specific hardware configurations, which may not be readily available or affordable for all libraries.
   - **Software Dependencies**: The reliance on specific software environments or versions can create issues if upgrades or changes are required.
9. **Limited Customization Flexibility**
   - **Rigid Frameworks**: Some advanced systems may offer limited flexibility for customization, making it difficult to tailor the system to specific library needs or workflows.
   - **Complex Customization**: Customizing the system to fit unique requirements may require specialized skills or external consultancy.
10. **Accessibility and Inclusivity**
    - **Limited Mobile Support**: Not all advanced systems provide mobile or remote access, which can limit usability for users who prefer or require access from various devices.
    - **User Inclusivity**: Systems that are not designed with accessibility in mind may not be usable by all members of the library community, including those with disabilities.

11. **Data Management**

   - **Data Overload**: Managing large volumes of data and ensuring its accuracy and relevance can be challenging, potentially leading to information overload or data management issues.

   - **Data Backup and Recovery**: Ensuring reliable data backup and recovery processes are in place is crucial but can be complex and costly.

\

CHAPTER 7
FUTURE SCOPE

As libraries continue to evolve in the digital age, the future scope of advanced library management systems (ALMS) involves integrating emerging technologies and addressing current limitations to enhance functionality, user experience, and operational efficiency. Here are several key areas of future development for ALMS:

---

1. Integration with Emerging Technologies

- **Artificial Intelligence (AI) and Machine Learning**: Implement AI-powered features for personalized recommendations, predictive analytics for book acquisitions, and automated cataloging. Machine learning algorithms can improve search accuracy and automate repetitive tasks.
- **Blockchain Technology**: Use blockchain for secure and transparent tracking of book transactions, ownership, and intellectual property rights. It can also ensure tamper-proof records and streamline inter-library loans and document sharing.

2. Enhanced User Experience

- **Advanced User Interfaces**: Develop intuitive and user-friendly interfaces with modern design principles. Implement responsive designs to ensure compatibility with various devices, including tablets and smartphones.
- **Voice and Natural Language Processing**: Integrate voice commands and natural language processing (NLP) to allow users to search for books, manage their accounts, and interact with the system using conversational language.

3. Improved Accessibility and Inclusivity

- **Universal Design**: Incorporate features that improve accessibility for users with disabilities, such as screen readers, voice control, and customizable interface options.
- **Mobile and Remote Access**: Enhance mobile applications and remote access features to provide users with the flexibility to interact with the library system from any location or device.

4. Advanced Data Analytics and Reporting

- **Big Data Integration**: Utilize big data technologies to analyze large volumes of data for insights into user behavior, book usage patterns, and library trends. This can inform decision-making and improve library services.
- **Interactive Dashboards**: Develop interactive dashboards and visualizations to present data in a more comprehensible and actionable format, helping librarians and administrators make informed decisions.

5. Automation and Efficiency Improvements

- **Robotic Process Automation (RPA)**: Implement RPA to automate routine administrative tasks such as data entry, report generation, and inventory management, reducing manual effort and errors.
- **Smart Shelving**: Use RFID technology and IoT (Internet of Things) to create smart shelves that automatically track book inventory, location, and status, and assist in real-time book management.

6. Enhanced Security and Privacy

- **Advanced Security Protocols**: Adopt state-of-the-art security measures, including multi-factor authentication (MFA), end-to-end encryption, and regular security audits to safeguard user data and system integrity.
- **Privacy Enhancements**: Implement features to comply with global privacy regulations, ensuring that user data is handled with the highest levels of confidentiality and control.

7. Cloud Integration and Scalability

- **Cloud-Based Solutions**: Migrate to cloud-based systems to provide greater scalability, flexibility, and accessibility. Cloud solutions can offer on-demand resources, automatic updates, and reduced hardware requirements.
- **Hybrid Models**: Explore hybrid models that combine local and cloud resources to balance performance, security, and cost-efficiency.

8. Enhanced Interoperability

- **API Integration**: Develop and integrate APIs to facilitate communication with other systems, such as academic databases, educational tools, and community resources, enhancing the overall functionality of the library management system.
- **Standardization**: Adopt industry standards and protocols to improve interoperability with other library systems and services, facilitating easier data exchange and collaboration.

9. Gamification and User Engagement

- **Gamified Learning**: Introduce gamification elements to engage users, such as badges, leaderboards, and interactive challenges related to reading and library activities.
- **Community Building**: Implement features that foster community engagement, such as user forums, book clubs, and event management tools.

10. Environmental Sustainability

- **Green IT Practices**: Promote sustainability by adopting green IT practices, including energy-efficient data centers, digital-only communication, and reducing paper usage through electronic records and reports.
- **Eco-Friendly Technologies**: Invest in technologies and practices that minimize environmental impact, such as energy-efficient servers and sustainable materials for hardware.

11. Customizable and Modular Designs

- **Modular Architecture**: Design systems with modular components that can be easily customized or upgraded to meet specific library needs without requiring a complete overhaul of the system.
- **User-Defined Customization**: Provide users with tools to customize the system's features and interface according to their preferences and requirements.

12. Collaboration and Resource Sharing

- **Inter-Library Collaboration**: Enhance capabilities for collaboration between libraries, including shared catalogs, resource pooling, and joint digital collections.
- **Global Networks**: Develop networks that connect libraries globally, enabling access to a wider range of resources and collaborative opportunities.

Conclusion

The future scope of advanced library management systems is expansive, with numerous opportunities for innovation and improvement. By embracing new technologies, enhancing user experiences, and addressing current limitations, libraries can build more robust and adaptable systems that better serve their communities and support the evolving landscape of information management.

References

- **Research on The Framework of Library Management System Based on Internet of Things**

Wang Zhigang

2021 13th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)

Year: 2021 | Conference Paper | Publisher: IEEE

Abstract:

In view of the traditional library management method, with low efficiency and single operation mode, this paper designs the information infrastructure and key mechanism of library management system, and discusses the innovation of system service mode from the aspects of readers' information acquisition and modification, fast management and delivery of books. With RFID technology as the core, through the comprehensive design of relevant subsystems, the key problems related to library management are solved, and the intelligent application scheme and overall architecture based on RFID are proposed. Finally, while realizing the basic functions, the RFID system of public library will be established into SAAS and cloud computing platform, and a set of integrated library information management system based on IoTs is designed. The system running test results show that the scheme has good information acquisition ability, which meets the needs of readers, and has a good development prospect for the future management activities of intelligent library.

- **The university library management system based on radio frequency identification**

Jin Feng Zhang;Chang Ji Wen

2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)

Year: 2017 | Conference Paper | Publisher: IEEE

Abstract:

Traditional library management is time consuming, laborious and low library circulation rate. The Radio Frequency Identification(RFID) has the characteristics of waterproof, anti-magnetic read distance and the label data can be encrypted, large storage data capacity and other technical features. This paper focuses on the design plan of university library management system based on RFID and elaborates the overall structure design of the system including the system hardware and software environment. The paper introduces the function and the use of each module in the system which emphatically studies the label conversion subsystem and self-help borrowing book subsystem. Through the design of the system the RFID brings automation and intelligence to the library management. The system is based on RFID and depends on RFID middle ware as the media to achieve the organic combination of the advanced RFID and library management and offers very effective technical means to the library management. The innovation of this paper is the use case diagram to explain the overall function of the system and its sub functions, and realize the intelligent management from the book entry to the circulation of books.

- **Comparing Public Library Management under Designated Administrator System with Direct Management: Forcusing on Reference Service**

Yuhiro Mizunuma;Keita Tsuji

2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)

Year: 2016 | Conference Paper | Publisher: IEEE

Abstract:

In Japan, public libraries have long been managed by local governments. However, with the introduction of the designated administrator system (DAS) in 2003, other organizations including private companies began to take over library management. In this study, we examine the differences between public libraries managed under DAS and those managed directly by local government (LG), with a focus on reference services. We compare the contents of the services provided and the number of reference questions received. The results show that LG libraries tend to answer users' questions directly, whereas DAS libraries tend to develop environments where users can find answers for themselves, such as by providing PCs connected to the Web and by offering classes on information-seeking skills. Our analysis shows that the DAS libraries receive more reference questions than LG libraries.

- https://github.com/kabariavansh55/Advance-library-Management.git