CUHK-SZ
Library Management System
Project Report

Diyuan DENG
123090079

November 17, 2024

1 Overall Project Description

This project is a **Library Management System** designed to efficiently manage book borrowing, returning, and reservations. The system distinguishes between three distinct user roles: **Admin(Librarian)**, **IT Admin**, and **User**. Each role has specific functions to streamline book management, user management, and borrowing processes. The system uses Python's `tkinter` library for the graphical user interface (GUI) and `sqlite3` for database management.

Key Features:

- Admin Functions: Add, edit, and delete books, manage borrowing and returning, view borrowing trends and other data analyses.
- User Functions: Borrow books, reserve books, check reservations, return books, renew books, view borrowing history, and view most popularly borrowed books.
- IT Admin Functions: Add, update, and delete user accounts, manage user roles.

2 Requirements Analysis

The system meets the following functional requirements:

Requirement	**Function Implemented**	**Description**
User Login	`login_user()`, `attempt_login()`	Validates user credentials and assigns roles
Admin Book Management	`add_book()`, `update_book()`, `delete_book()`	Admin can manage the book inventory
IT Admin User Management	`insert_user()`, `update_user()`, `delete_user()`	Manage user accounts and roles
Book Borrowing	`borrow_selected_book()`, `confirm_borrow()`	Allows users to borrow books
Book Returning	`return_selected_book()`	Handles book returns
Borrow History View	`show_borrow_history()`, `view_borrow_history()`	Displays user's borrowing history
Book Reservation	`confirm_borrow_reserved_book()`, `check_reservations()`	Users can reserve books
Borrowing Trend Analysis	`open_data_analytics_window()`	Displays data analytics of borrowing trends
Due Date Check	`check_due_dates()`	Notifies users of due dates
Front-End User Interface	`tkinter` components	Provides GUI for users

3 Database and SQL Design

3.1 Database Schema

The database is designed using `sqlite3` with the following tables:

Users Table

The table is used to store the users' information.

Column	Data Type	Description
user_id	INTEGER	Primary Key, Auto-increment
username	TEXT	Unique username
password	TEXT	Hashed password
role	TEXT	Role (Admin, IT_Admin, User)

Books Table

The table is used to store the information of the books in library.

Column	Data Type	Description
book_id	INTEGER	Primary Key, Auto-increment
title	TEXT	Book title
author	TEXT	Book author
nationality	TEXT	Author's nationality
isbn	TEXT	Unique ISBN
shelf_number	TEXT	Shelf location
total_quantity	INTEGER	Total copies available
available_quantity	INTEGER	Current available copies

BorrowedBooks Table

The table is used to store the borrowed books of each user.

Column	Data Type	Description
borrowed_id	INTEGER	Primary Key, Auto-increment
username	TEXT	Foreign Key referencing Users
book_id	INTEGER	Foreign Key referencing Books
title	TEXT	Title of the borrowed book
quantity	INTEGER	Quantity borrowed
borrow_date	DATE	Date when book was borrowed
return_date	DATE	Due date for returning
status	TEXT	Borrow status (borrowed/returned)

If the status is 'borrowed', it means the books is still borrowed by the user. Otherwise if the status is 'returned', it means the book is already returned. Both of the status can be shown in "Borrow History" table.

Reservations Table

If all copies of a book are borrowed, the user can reserve the book. And the reservation information will be stored in this table.

Column	Data Type	Description
reservation_id	INTEGER	Primary Key, username
username	TEXT	Primary Key, username
book_id	TEXT	Book title & Foreign Key referencing Books
reservation_date	TEXT	Book author
notified	TEXT	Author's nationality

3.2 SQL Queries example

When the user tries to search books, the program will run the following query as the graph shows:

```python
# Search Books
def search_books(query, search_by_title=False, search_by_author=False, search_by_isbn=False, search_by_nationality=False):
    conditions = []
    params = []

    if not query.strip():
        query_string = "SELECT * FROM Books"
    else:
        if search_by_title:
            conditions.append("title LIKE ?")
            params.append(f"%{query}%")
        if search_by_author:
            conditions.append("author LIKE ?")
            params.append(f"%{query}%")
        if search_by_isbn:
            conditions.append("isbn LIKE ?")
            params.append(f"%{query}%")
        if search_by_nationality:
            conditions.append("nationality LIKE ?")
            params.append(f"%{query}%")

        # 没有选择任何复选框时，默认搜索所有四个字段
        if not conditions:
            conditions.append("title LIKE ? OR author LIKE ? OR isbn LIKE ? OR nationality LIKE ?")
            params.extend([f"%{query}%"] * 4)

        # 创建查询语句，多个条件使用 OR 连接
        query_string = "SELECT * FROM Books WHERE " + " OR ".join(conditions)

    with sqlite3.connect('library.db') as conn:
        cursor = conn.cursor()
        cursor.execute(query_string, params)
        return cursor.fetchall()
```

When the user logs in, the program will check his /her reservations, and it will run the following query:

```sql
SELECT Reservations.book_id, Books.title
    FROM Reservations
    JOIN Books ON Reservations.book_id = Books.book_id
    WHERE Reservations.username = ? AND Books.available_quantity > 0
       AND Reservations.notified = 0
```

4 Front-End and Back-End Design

4.1 Front-End Design

The front-end uses Python's `tkinter` library to create a graphical interface for different user roles.

4.2 Back-End Design

The back-end uses Python with `sqlite3` for database operations. Functions handle CRUD operations for books, users, and borrowing records.

Sample Implementation Table

Requirement	Function	Description
User Login	`login_user()`	Validates user credentials
Add Book	`add_book()`	Admin adds a new book
Borrow Book	`borrow_selected_book()`	User borrows a book
Return Book	`return_selected_book()`	User returns a book
View Borrow History	`show_borrow_history()`	Displays borrowing history
Reservation Notification	`check_reservations()`	Notifies users of available reservations

5 How to Use the System

5.1 Environment Setup

1. First you need to configure the environment. You need to install the three necessary libraries first by installing them in PowerShell (for MacOS or Linux, you need to install them with Terminal). Otherwise, the code cannot be run correctly.
```
pip install matplotlib
pip install datetime
pip install pandas
```

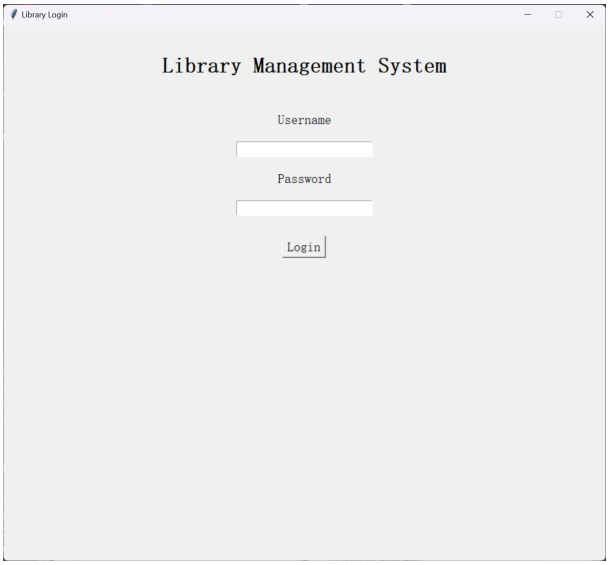
2. Then you can compile and run the code. The program will lead you to the login page automatically.

5.2 Login Page

1. If you input the librarian's (administrator's) username and password, you will be led to the admin dashboard.
2. If you input the normal user's username and password, you will be led to the admin dashboard.
3. If you input the IT_admin's username and password, you will be led to the admin dashboard.

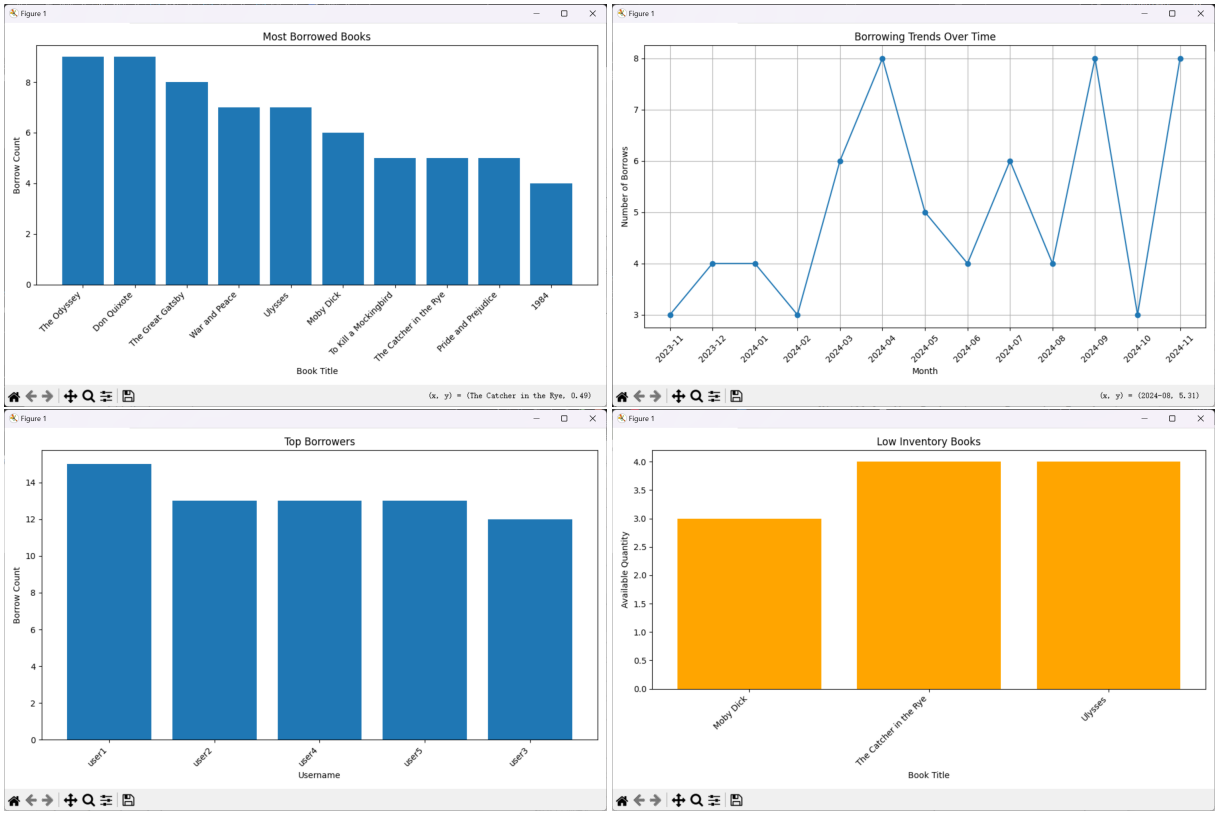
5.3 Dashboard - Key Features

- Login Page:
  - Login page
    - Input the username and password and the user will be led to **Admin Dashboard/ (Normal) User Dashboard/ IT Admin Dashboard**



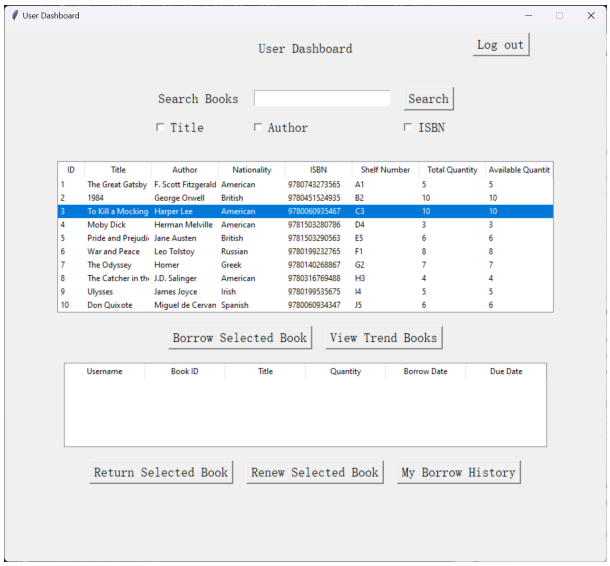
- **Admin Dashboard** - Admin Functions (Can be used by librarians)
  - Search Books
    - Broad Search
    - Search by title/ author/ ISBN
    - Click Title/ Author/ Nationality, and the book list will be shown in ascending order.
  - Add Books
  - Edit Book's Information (can also delete books).
  - Manage Lending and Returning

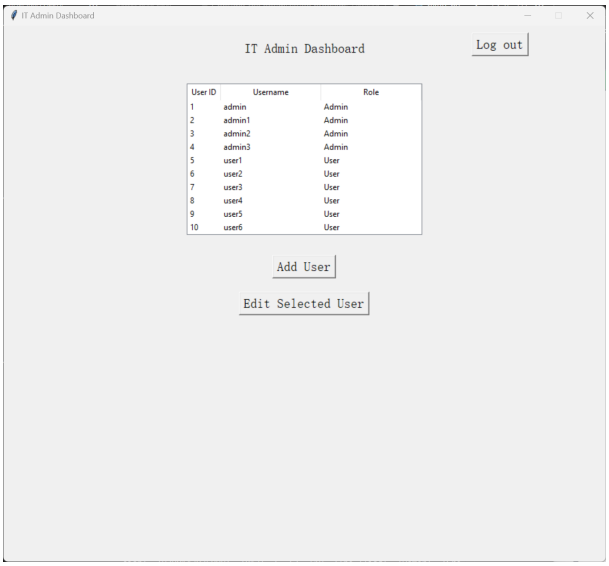
    - Lend books to users
    - Search by title/ author/ ISBN
  - View all the users who have borrowed the same specific book.
  - View all the the currently lending books.
    - Admin can help users return /renew books in this page.
  - View all the borrowing histories of all users.
  - Data Analysis
    - View Most Borrowed Books
    - Borrowing Trends Over Time
    - View Top Borrowers (users who borrowed the most books)
    - Low Inventory Books



- **User Dashboard** - User Functions: Borrow books, return books, view borrowing history, renew books, and check reservations.
  - Search Books
    - Broad Search
    - Search by title/ author/ ISBN
    - Click Title/ Author/ Nationality, and the book list will be shown in ascending order.
  - Borrow Books
  - Reserve Books (if no copies available)
  - Book Returning Reminder (if the due date is within 7 days)
  - Return Books
  - Renew Books

- View all the personal borrowing histories.
- Data Analysis
  - View Most Borrowed Books(The part is the same as 'Most Borrowed Books' in the admin dashboard.)





- **IT Admin Dashboard** - IT Admin Functions:
  - Add Users
  - Edit Users' Information (can also delete users)

6 Sample SQL Queries for Functions

Initialization: Create Table

Setup User Database

```
CREATE TABLE IF NOT EXISTS Users (
    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT UNIQUE NOT NULL,
    password TEXT NOT NULL,
    role TEXT NOT NULL
)
```

Setup Borrowed Books Database

```
CREATE TABLE IF NOT EXISTS BorrowedBooks (
    borrowed_id INTEGER UNIQUE PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL,
    book_id INTEGER NOT NULL,
    title TEXT NOT NULL,
    quantity INTEGER NOT NULL CHECK (quantity > 0),
    borrow_date DATE NOT NULL,
    return_date DATE NOT NULL,
    status TEXT DEFAULT 'borrowed',
    FOREIGN KEY (book_id) REFERENCES Books(book_id)
)
```

User Login

When the user tries to login, the program will run the following SQL Query.
If the user inputs username = `user1` and password = `password123`, whose SHA-256 is `EF92B778BAFE771E89245B89ECBC08A44A4E166C06659911881F383D4473E94F`, then the following query will be executed.

```
SELECT role FROM Users WHERE username = 'user1' AND password =
    'EF92B778BAFE771E89245B89ECBC08A44A4E166C06659911881F383D4473E94F';
```

If no match, then the user need to re-input the username and the password. Otherwise, the query will return the role of the user, and the program will lead the user to the corresponding dashboard.

Create User

The code will store the user's info in the database 'Users', and to guarantee the information safety, the database will only store the SHA-256 processed password.

```python
def insert_user(username, password, role):
    hashed_password = hashlib.sha256(password.encode()).hexdigest()
    try:
        with sqlite3.connect('library.db') as conn:
            cursor = conn.cursor()
            cursor.execute('''
            INSERT INTO Users (username, password, role) VALUES (?, ?, ?)
            ''', (username, hashed_password, role))
            conn.commit()
    except sqlite3.IntegrityError:
        print("User already exists.")
```

```
INSERT INTO Users (username, password, role) VALUES (?, ?, ?)

-- The three '?'s stand for the input username, (encrypted) password,
    and the role of user respectively.
```

Add New Book

```
INSERT INTO Books (title, author, nationality, isbn, shelf_number,
    total_quantity, available_quantity)
VALUES (?, ?, ?, ?, ?, ?, ?);

-- The six '?' s mean the input book title, author, nationality of the
    author, ISBN of the book, which shelf is the book on, total quantity
     of the book, and the available quantity (which can be borrowed at
    present).
```

Borrow Book

```
UPDATE Books SET available_quantity = available_quantity - 1
    WHERE book_id = ? AND available_quantity > 0;
INSERT INTO BorrowedBooks (username, book_id, title, quantity,
    borrow_date, return_date, status)
    VALUES (?, ?, ?, 1, ?, ?, 'borrowed');
```

7 Demo Instructions

1. Demonstrates to login with different roles.

2. Showed Admin dashboard's, User dashboard 's and IT admin dashboard's functionalities.
3. Demonstrate User functionalities such as borrowing and returning books.
4. Display data analysis graphs, including
   (a) Most Borrowed Books
   (b) Borrowing Trends
   (c) Top Borrowers
   (d) Low Inventory Books
5. Ensure all tables have a reasonable amount of data.