

课程设计报告

课程名称 计算机程序设计基础 2

班 级 无14
学 号 2021012719
姓 名 董相宏

2024 年 6 月 30 日

一、设计内容与设计要求

1. 课程设计目的

面向对象程序设计课程设计是集中实践性环节之一，是学习完《面向对象程序设计》课程后进行的一次全面的综合练习。要求学生达到熟练掌握 C++语言的基本知识和技能；基本掌握面向对象程序设计的思想和方法；能够利用所学的基本知识和技能，解决简单的面向对象程序设计问题，从而提高动手编程解决实际问题的能力。**尤其重视创新思维培养。**

2. 课题题目

- 1) 图书管理系统

3. 文档设计要求

3.1 设计课题题目：每个同学都单独完成 1 道课题。后面有范题，仅供同学们参考，不列入本次课程设计的课题。

3.2 对于程设题目，按照范题的格式。自行虚构软件需求。并按照第 4 点要求，编写设计文档。基本要求系统中设计的类的数目不少于 4 个，每个类中要有各自的属性（多于 3 个）和方法（多于 3 个）；需要定义一个抽象类，采用继承方式派生这些类。并设计一个多重继承的派生类。在程序设计中，引入虚函数的多态性、运算符重载等机制。

4. 程序设计的基本要求：

4.1 要求利用面向对象的方法以及 C++的编程思想来完成系统的设计；
4.2 要求在设计的过程中，建立清晰的类层次；
4.3 根据课题完成以下主要工作：①完成系统需求分析：包括系统设计目的与意义；系统功能需求（系统流程图）；输入输出的要求。②完成系统总体设计：包括系统功能分析；系统功能模块划分与设计（系统功能模块图）。③完成系统详细设计：数据文件；类层次图；界面设计与各功能模块实现。④系统调试：调试出现的主要问题，编译语法错误及修改，重点是运行逻辑问题修改和调整。⑤使用说明书及编程体会：说明如何使用你编写的程序，详细列出每一步的操作步骤。⑥关键源程序（带注释）

4.4 图书管理系统**应至少包含以下功能：**

- ① 用户信息管理：存储用户信息，如用户名、借阅历史等。支持对用户的信息进行查询、增加、删除、修改等操作；
- ② 图书信息管理：存储图书信息，如书名、作者、类别、关键字、简介、借阅状态等。支持对图书的信息进行查询、增加、删除、修改等操作，且要求图书查询功能具有一定模糊查询的能力，例如用户输入一个字，可查询到包含该字的图书；
- ③ 借还书记录管理：记录用户借还书的时间等信息、查看用户或图书的借阅历史等；
- ④ 统计分析功能：根据借还书记录进行简单的统计分析，比如最受欢迎的书籍、借阅最多的用户、用户借阅量随时间的变化趋势等。可参考某些 app 的年度报告；
- ⑤ 特殊情况处理：对非法输入等特殊情况进行合适的处理；
- ⑥ 你认为需要的其他功能。

4.5 自己设计测试数据，将将测试数据存在文件中，通过文件来进行数据读写来测试；

4.6 按规定格式完成课程设计报告，并在网络学堂上按时提交；
4.7 不得抄袭他人程序、课程设计报告，每个人应独立完成，在程序和设计报告中体现自己的个性设计。

5. 进度安排

小学期 第 1、2 周	学习使用 Qt Designer 设计 ui 界面	学习 Qt 编程中的信号与槽	学习 Qt 编程中的组件和布局
	设计程序各个界面的 ui 样式	下载程序所需要的图标	在 VS 中配置 Qt 编程环境
	编写管理员界面，实现堆栈窗口逻辑	编写用户界面，实现堆栈窗口逻辑	实现各功能接口
	调试，修改，完善各功能	添加运算符重载和虚函数多态实例	打包代码生成可执行文件

注： 1、一定要保留自己那个课题的完整任务书在课程设计报告里面。

2、“评分表”放在“附录：源程序清单”的后面。

目 录

一、图书管理系统	1
1. 系统需求分析	1
2. 总体设计	2
3. 详细设计	3
(0) 数据库设计	3
(1) 类的设计概览	4
(2) 功能模块	5
4. 系统调试	10
5. 使用说明/功能介绍	11
(1) 登录界面	11
(2) 用户主界面	15
(3) 管理员主页面	26
6. 总结	44
二、源程序清单	45
1. 说明	45
2. 头文件	45
(1) add_singleuser.h	45
(2) au_singlebook.h	46
(3) base.h	47
(4) cell_analysis.h	48
(5) cell_bkbrw.h	49
(6) cell_bkmgr.h	50
(7) cell_rcdmgr.h	51
(8) cell_self.h	52
(9) cell_usermgr.h	53
(10) dlg_login.h	54
(11) msgbox.h	55
(12) page_admin.h	56
(13) page_user.h	57
(14) sqlmgr.h	58
3. 源文件	60
(1) add_singleuser.cpp	60
(2) au_singlebook.cpp	61
(3) base.cpp	62

(4)	cell_analysis.cpp	63
(5)	cell_bkbrw.cpp	66
(6)	cell_bkmgr.cpp	69
(7)	cell_rcdmgr.cpp	73
(8)	cell_self.cpp	76
(9)	cell_usermgr.cpp	80
(10)	dlg_login.cpp	84
(11)	msgbox.cpp	85
(12)	page_admin.cpp	86
(13)	page_user.cpp	88
(14)	sqlmgr.cpp	90
(15)	main.cpp	97
4.	资源文件	98
	Resource.qrc	98
三、	评分表	99

一、图书管理系统

1. 系统需求分析

图书管理系统集成用户端和管理员端，并且记录如下信息：用户ID、用户名、用户密码、用户借阅总量、图书ID、图书名称、作者、图书类别、在架数量、被借总数、借阅记录ID、借阅人ID、借阅书籍ID、借阅书籍名称、借阅起始时间、应还时间、借阅状态。该图书管理系统需要实现如下功能：

(0) 登录界面

用户端：

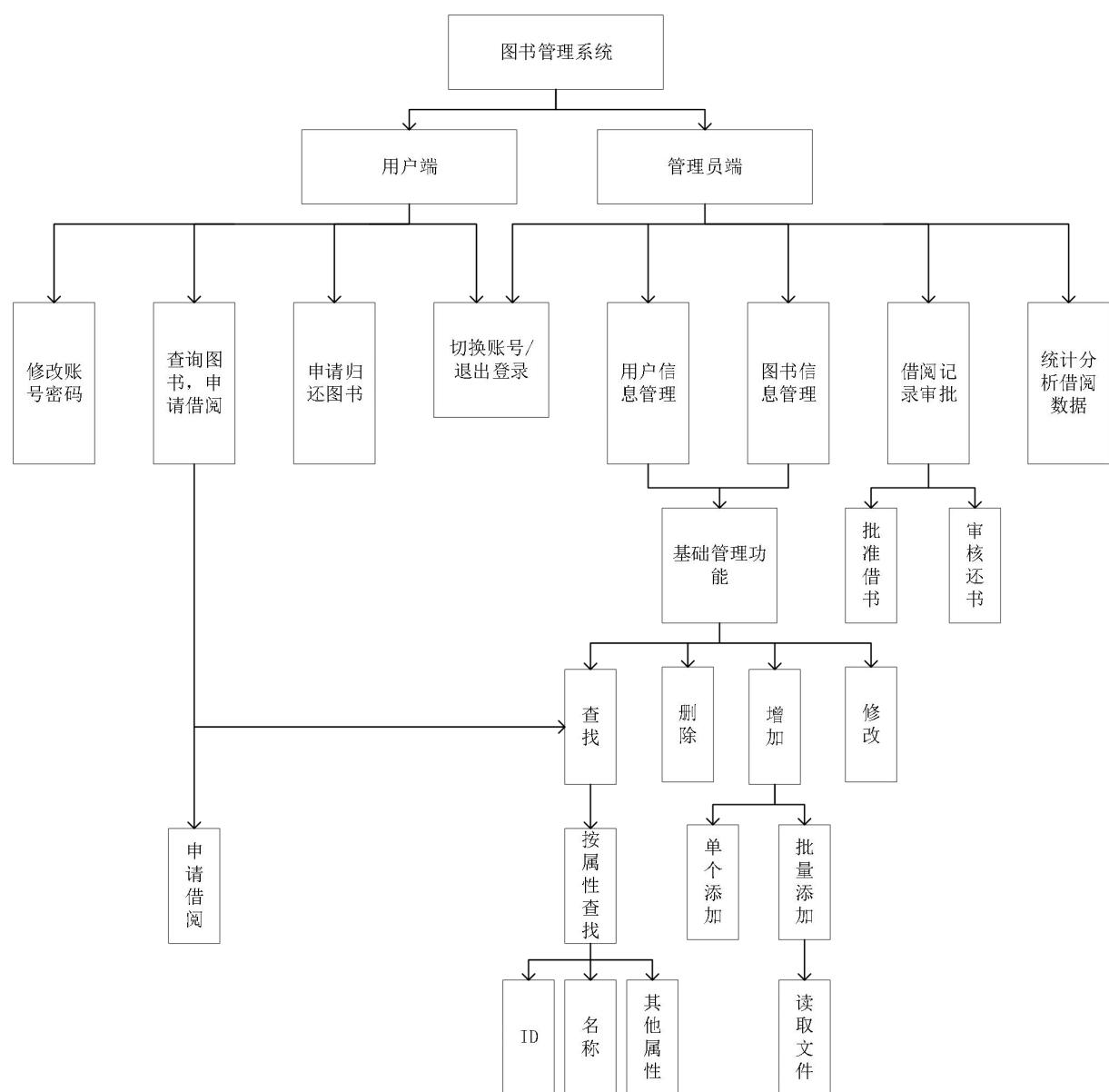
- (1) 随时修改账户和密码，也即个人信息
- (2) 查找图书并申请借阅
- (3) 申请归还已借未还的图书
- (4) 切换账号
- (5) 退出登录

管理员端：

- (1) 管理用户信息，可增加、删除、查找、修改用户
- (2) 管理图书信息，可增加、删除、查找、修改图书
- (3) 管理借阅记录，可批准借书申请，审核还书申请
- (4) 简单的统计分析，可查看借书最多的用户，被借最多的图书，用户借阅数随时间的变化情况
- (5) 切换账号
- (6) 退出登录

2. 总体设计

经过以上系统需求分析，可以绘制出学生管理系统的功能模块图如下：



3. 详细设计

(0) 数据库设计

项目数据库(SQLite)文件bk_mgr.db中共有3个表，内容如下：

book表

字段	bookid	book-name	author	type1	type2	access	cnt
属性	integer	text	text	text	integer	integer	integer

access为在架数量，cnt为被借次数

user表

字段	userid	username	password	brwcnt	admin
属性	integer	text	text	integer	text

brwcnt为借阅总数，admin为用户权限

record表

字段	recordid	userid	bookid	bookname	start	end	request
属性	integer	integer	integer	text	text	text	text

start为借阅起始时间，end为应还日期，request为借阅状态

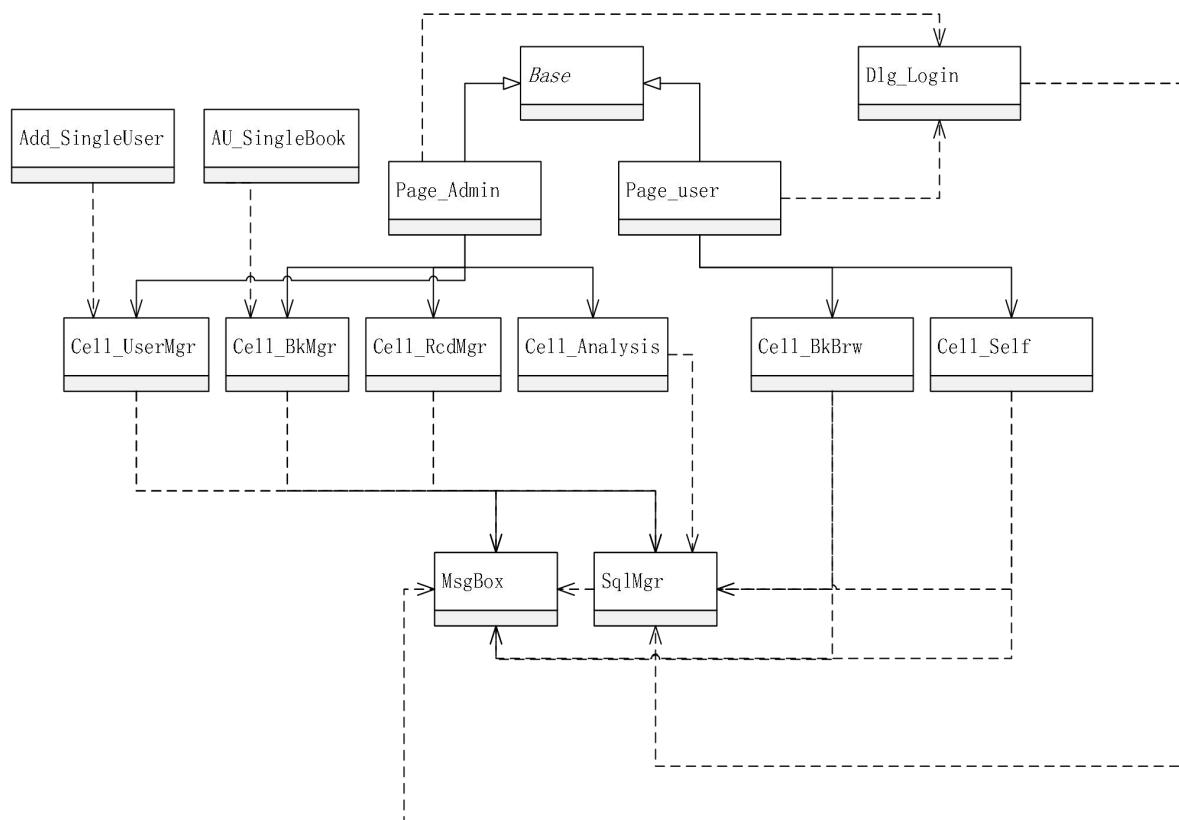
对数据库的基本操作：增删查改，通过下列语句实现

```

1 SELECT * FROM "user" WHERE admin=NULL;
2
3 INSERT INTO user VALUES(NULL,'科比','123456','1','');
4
5 DELETE FROM sqlite_sequence WHERE name='user';
6
7 UPDATE record SET start='2',request='未借阅' WHERE recordid =1;
```

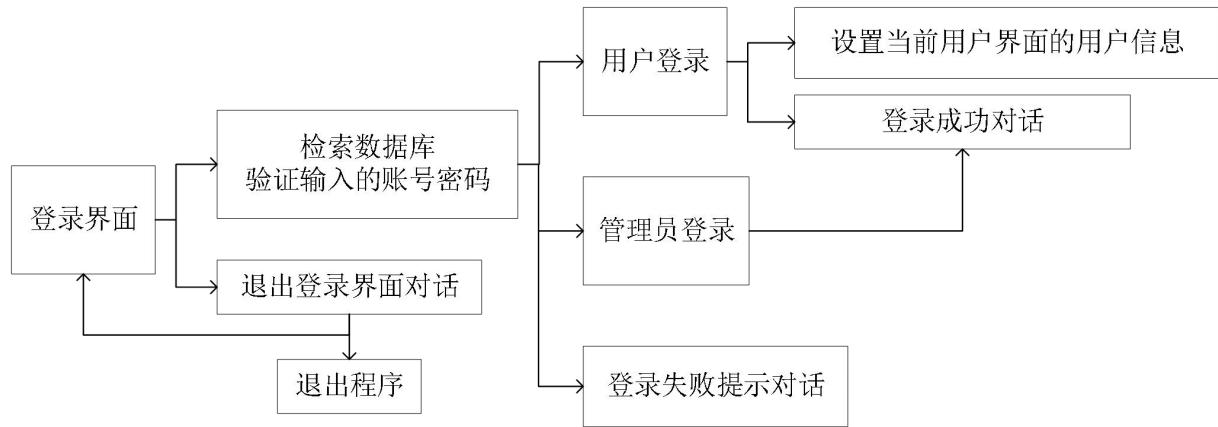
(1) 类的设计概览

图书管理系统设计了个14类， Dlg_Login实现登录对话， Page_Admin和Page_user对应管理员端和用户端， 均继承自抽象类 Base（Base类继承自Qt库中的 QMainWindow类， 由Base类派生实现了多重继承）， Sql_Mgr类用于建立数据库连接以及对数据库的操作， 其余子页面集成在对应的端口。 Cell_UserMgr为用户管理界面， Cell_BkMgr为图书管理界面， Cell_RcdMgr为借阅记录管理界面， Cell_Analysis为统计分析界面， Cell_BkBrw为图书借阅界面， Cell_Self为个人信息管理界面。此外， Add_SingleUser为添加单个用户对话， AU_SingleBook为添加/修改单本图书对话， MsgBox为消息对话框。 对应的UML类图（由于类的总数、 属性和方法均较多， 完整的UML图绘制之后阅读的清晰度较差， 这里省略了每个类的具体属性和方法）如下：

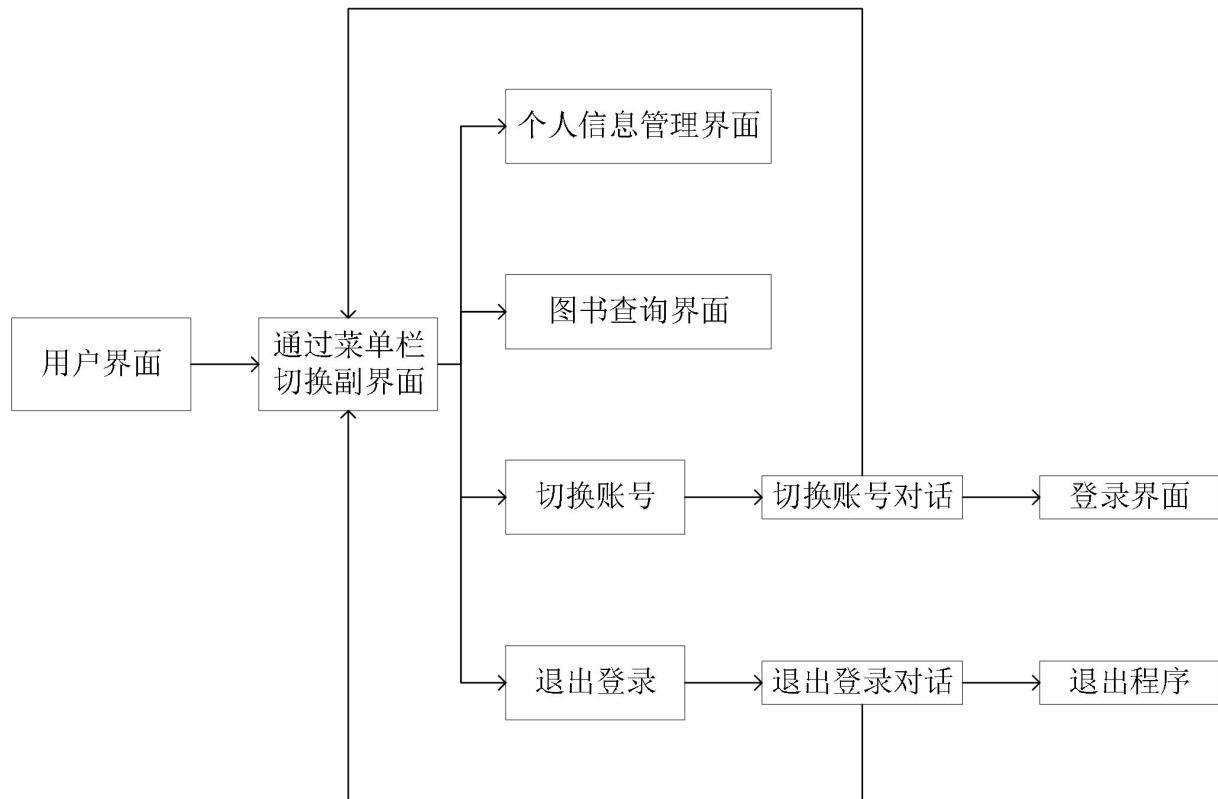


(2) 功能模块

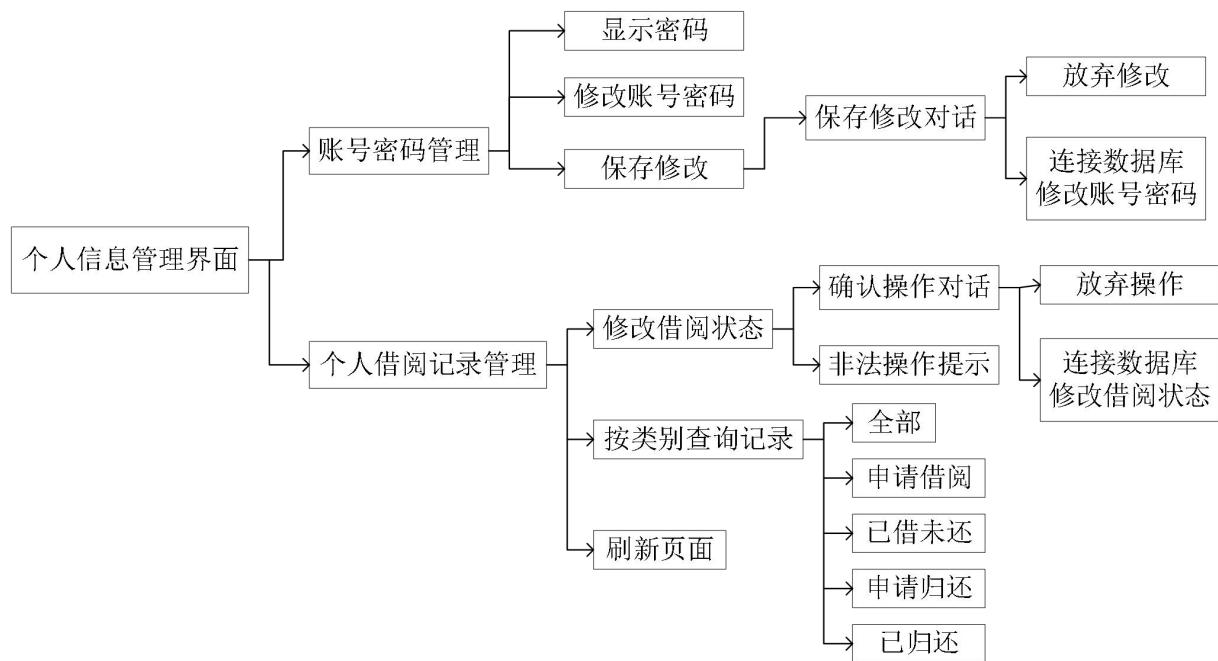
登录界面功能如下：



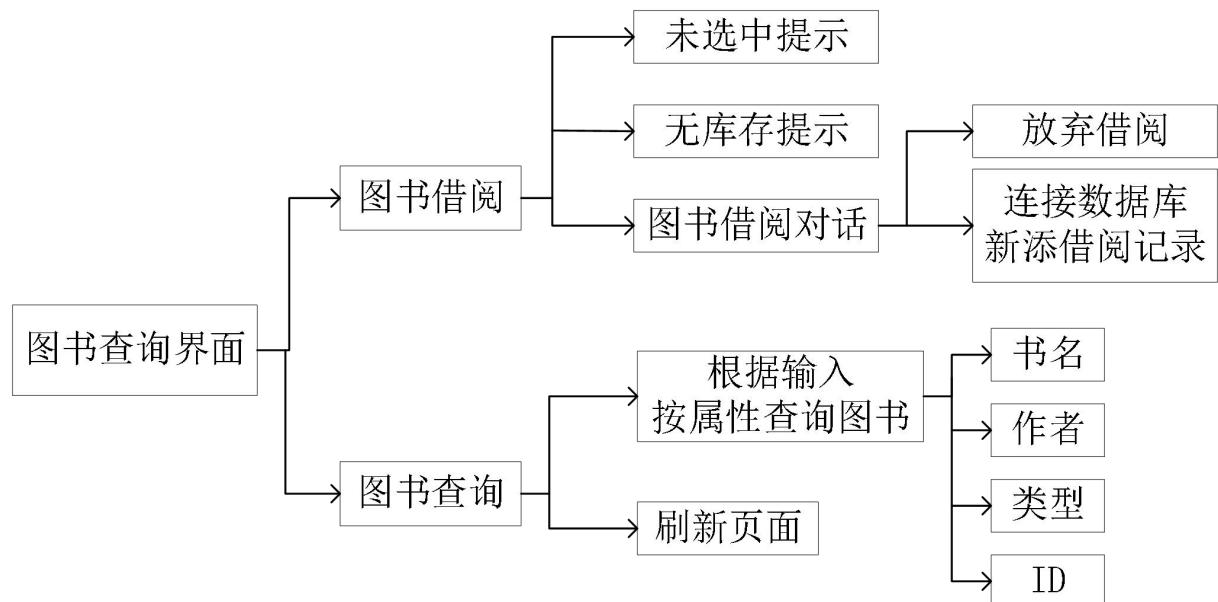
用户端主界面功能如下：



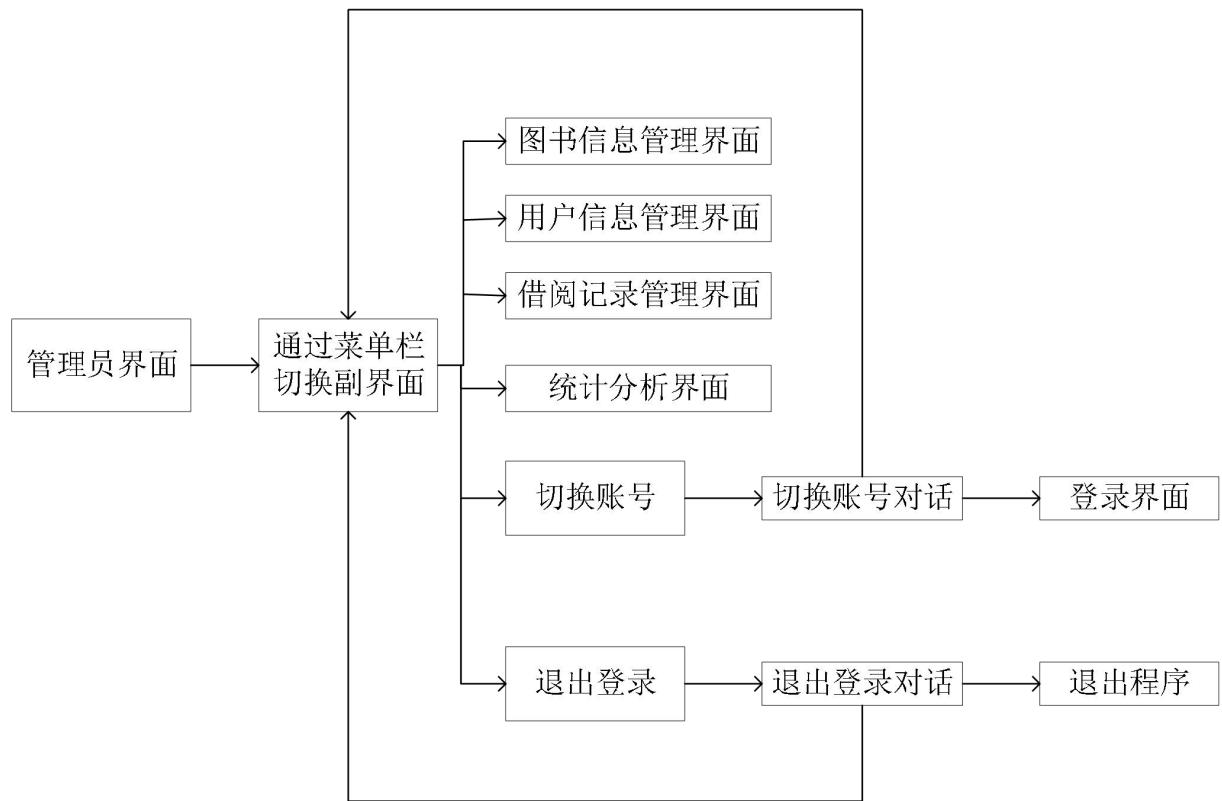
个人信息管理界面功能如下：



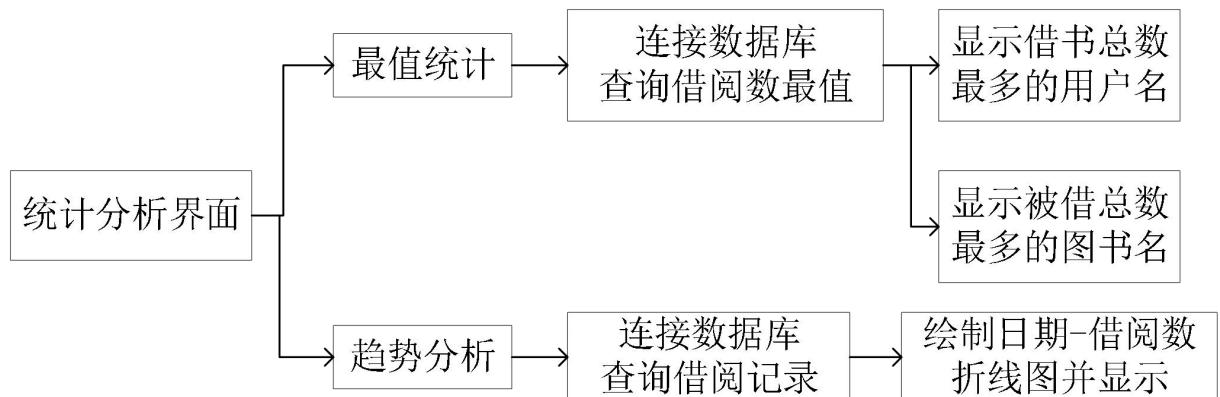
图书查询界面功能如下：



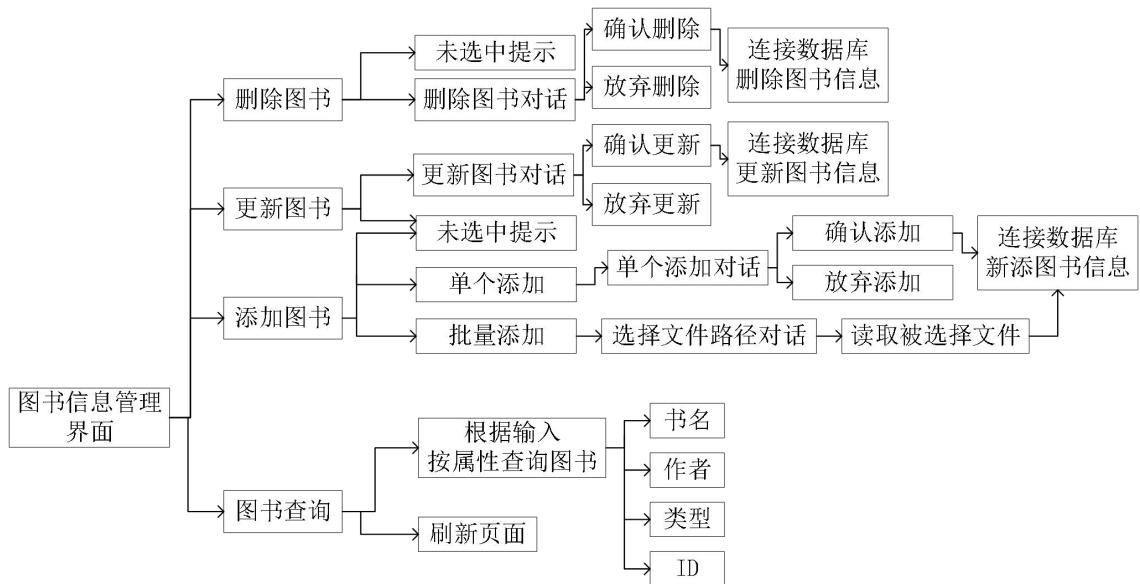
管理员端主界面功能如下：



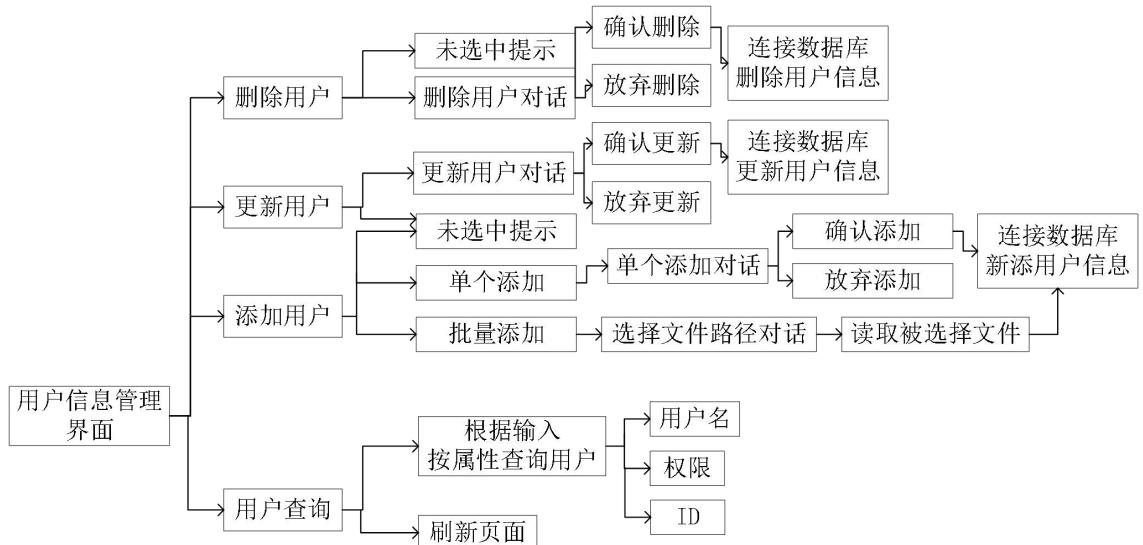
统计分析界面功能如下：



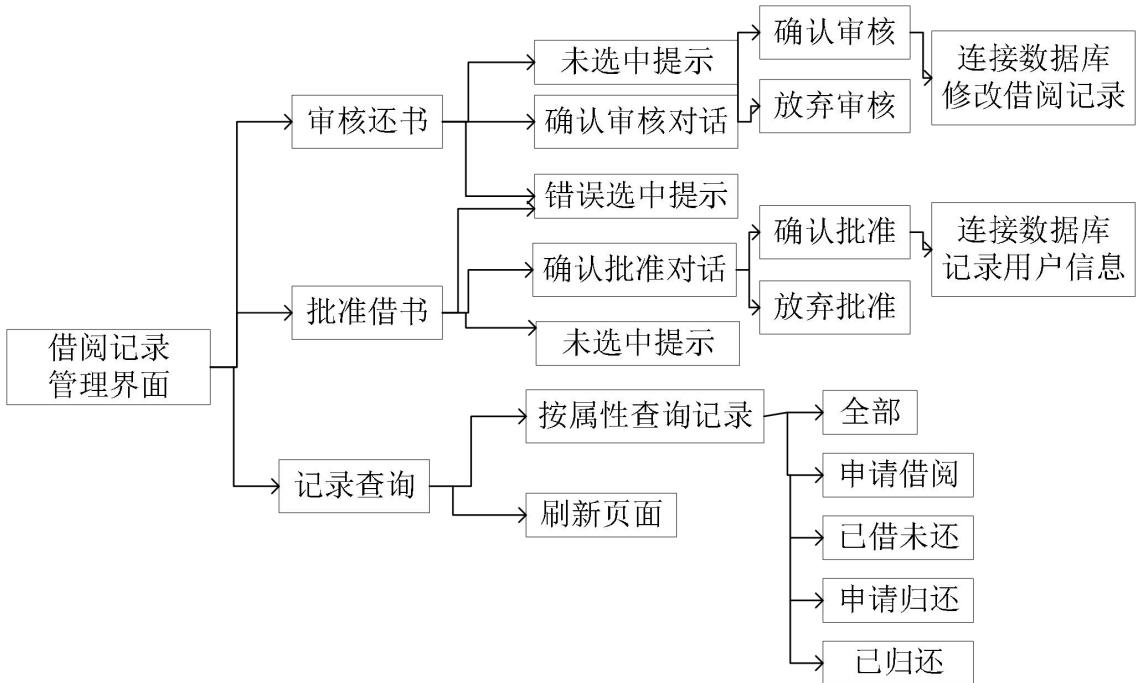
图书信息管理功能如下：



用户信息管理功能如下：



借阅记录管理界面功能如下：



程序设计要求的达成：

虚函数多态性机制：在抽象类Base中定义了纯虚函数void pop_message()用于弹出登录后的提示对话。在派生类Page_Admin和Page_user中分别实现pop_message()。其中对于管理员主界面类，登录提示信息为提醒管理员尽快完成借还书记录的批准。对于用户主界面类，登录提示信息为提醒用户尽快归还已借未还的图书。

运算符重载机制：在自定义消息对话框类MsgBox中，定义了“!”运算符的重载，用于隐藏消息对话框的按钮，只展示信息。具体实现方法请见源程序清单的对应代码。

程序功能的达成：

本图书管理系统除了实现了用户信息管理，图书信息管理，借还书记录管理，统计分析功能和特殊情况处理功能这些要求的基本功能之外，还实现了数据库处理数据，登录界面设计，管理员端和用户端的集成，以及利用Qt完成的图形界面设计。

4. 系统调试

在编写程序以及测试功能的过程中遇到的部分具有代表性的问题如下：

- (1) 程序中使用的SQLite数据库对应的MySQL语言不熟练，出现了忘加空格，忘加单引号等错误。
- (2) 由于Qt库的封装性非常完善，功能专门性程度很强。在使用已有的较完善的 QMainWindow 类派生抽象类 Base，再由 Base 类派生两类主界面类的过程中遇到了未在派生类中实现虚函数导致编译不通过的问题。
- (3) 由于Qt特有的信号与槽机制，在手动连接一些具有时序逻辑的信号与槽函数时出现了逻辑错误。比如在用户成功登录之后应该向用户界面类发送信号，通过当前用户的账号查找到当前用户的ID从而在用户主界面的个人信息子界面中显示用户信息，但是在进行编写时出现了槽函数对象连接的错误。
- (4) 由于Qt编程可以通过编辑.ui文件直接进行界面设计，但是在进行设计时对于控件和布局的属性设置不熟悉。常常出现某界面的某控件没有成功显示预设的属性或者样式表编辑时CSS语言编写错误等问题。

总体而言系统的调试过程比较成功，出现的错误大多是由不熟悉Qt编程的一些细节以及在逻辑设计上思考的不够全面。因为整体程序的模块化设计，出现的错误能够很快的定位和改正。

5. 使用说明/功能介绍

(1) 登录界面

数据库中user表部分内容截图如下：

userid	username	password	brwcnt	admin
1	kobe	0824	3	
2			(Null)	
3			0	管理员
4	ZhangWei	123456	10	
5	StarGazer2024	123456	16	
6	LunarEclipse9	123456	20	
7	MysticWave7	123456	5	
8	GalaxyHunter	Xyz#7890	8	
9	SolarFlareX	Sec!1234	12	
10	QuantumLeaf	Key@5678	22	
11	NebulaWatch	Pwd@3456	9	管理员
12	AstroVoyager	M@ster789	18	

程序登录界面：



读者登录需要输入已存在的用户名和密码：



登录成功后会显示登录提示对话，关闭后即进入主界面：



输入错误的账号/密码会显示登录失败信息：



管理员登录需要有管理员权限的用户输入正确的账号密码：



管理员登录成功之后会弹出登录提示信息：



在登录界面点击关闭按钮之后会弹出确认退出对话：



(2) 用户主界面

数据库中book表部分内容截图如下：

bookid	bookname	author	type1	type2	access	cnt
9	银河帝国	Isaac-Asimov	科幻	冒险	0	1
10	时间简史	Stephen-Hawking	科学	哲学	8	0
11	活着	余华	历史	社会	12	1
12	三体	刘慈欣	科幻	悬疑	10	2
13	神秘岛	Jules-Vern	冒险	推理	10	0
14	百年孤独	Gabriel-Garcia	爱情	历史	9	0
15	哈利·波特与魔法石	J.K.-Rowling	奇幻	冒险	25	0
16	基督山伯爵	Alexandre-Dumas	冒险	犯罪	18	0
17	麦田里的守望者	J.D.-Salinger	青春	文学	11	0

数据库中record表部分内容截图如下：

recordid	userid	bookid	bookname	start	end	request
9	2	12	三体	2024-06-15	2024-09-01	已归还
10	2	11	活着	2024-06-15	2024-09-01	已归还
11	5	24	霍乱时期的爱情	2024-06-15	2024-09-01	已借未还
12	2	12	三体	2024-06-15	2024-09-14	已归还
13	1	9	银河帝国	2024-06-15	2024-09-14	申请归还
14	2	21	飘	2024-06-15	2024-09-14	已归还
15	1	48	银河系漫游指南	2024-06-15	2024-09-14	已借未还
16	2	36	追忆似水年华	2024-06-15	2024-09-14	申请归还
17	2	34	刀锋	2024-06-20	2024-09-14	已借未还
18	2	17	麦田里的守望者			申请借阅

用户主界面总览：



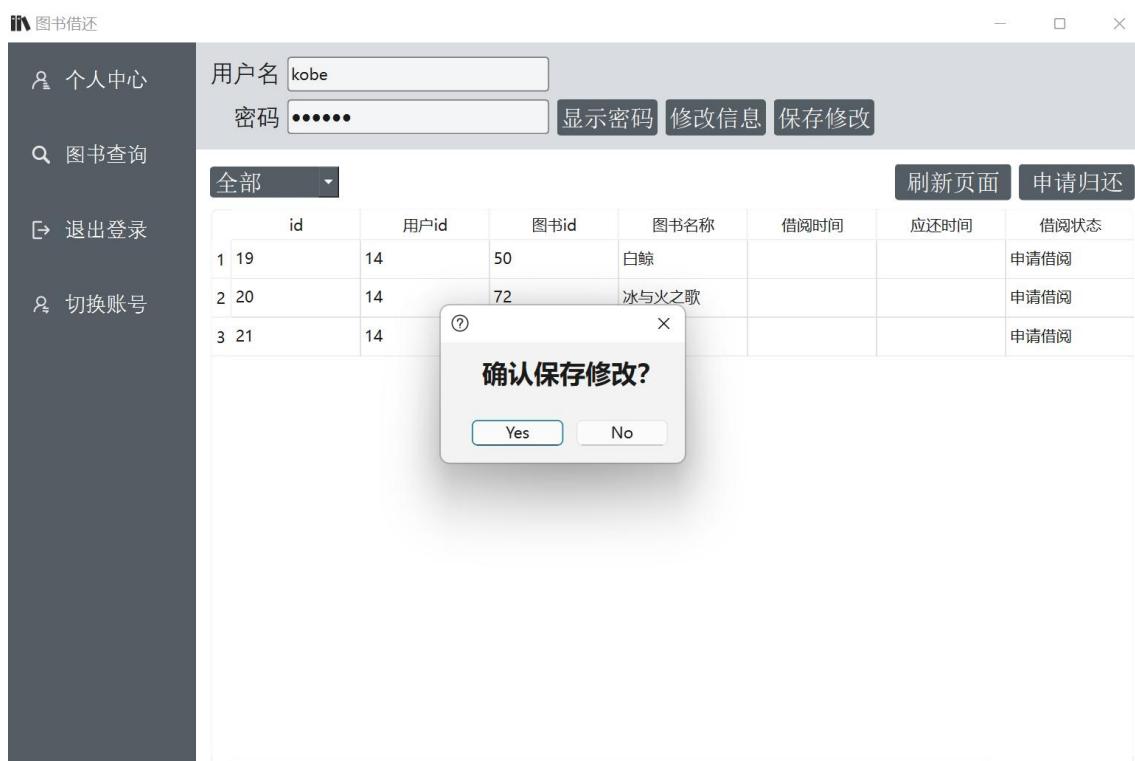
在个人中心点击显示密码，按钮文字变为“隐藏密码”，再次点击密码就会隐藏：



在个人中心点击修改信息，左侧的两个输入框会由不可编辑状态变为可编辑，此时可以进行账号密码的修改：



点击保存修改，此时会弹出确认操作对话，点击取消则取消本次保存操作，点击确定则进行下一步操作：



确认保存修改之后首先查询用户名是否已经被使用，若已被使用则弹出提示对话：



若用户名合法则弹出修改成功对话，此时程序已经通过Sql语句修改了userid为当前id的用户的用户名和密码：



点击申请归还按钮会首先检测操作的合法性，若鼠标当前并无选中，则弹出报错对话：



检测点击“申请归还”的合法性时，若鼠标当前选中的记录的借阅状态不是“已借未还”，则弹出报错对话：



若点击“申请归还”的操作合法，则弹出确认操作对话：



若点击确定，则会弹出成功申请的对话。此时程序已经通过Sql语句修改了recordid为当前id的借阅状态为“申请归还”：



点击左侧的组合框，切换当前的搜索关键词即显示对应借阅状态的借阅记录，此时程序通过Sql语句实现查询功能：



此时若选择组合框中“全部”或者点击右侧的刷新页面按钮会重新显示全部记录

图书查询界面显示如下：



点击左侧的组合框，切换当前的搜索属性即可通过输入框中的搜索关键词显示对应图书，此时程序通过Sql语句实现查询功能，对于书名、作者、类型均实现模糊查询：

The screenshots show a library management system interface with a sidebar and a main search area.

- Screenshot 1 (Top Left):** Search by Book Name. The search term is "铜". Results include:

ID	书名	作者	类型一	类型二	在架数量	被借总数
1 9	银河帝国	Isaac-Asimov	科幻	冒险	0	1
2 48	银河系漫游指南	Douglas-Adams	科幻	冒险	16	1
3 94	银河	Isaac-Asimov	科幻	冒险	15	0
- Screenshot 2 (Top Right):** Search by Author. The search term is "mar". Results include:

书名	作者	类型一	类型二	在架数量	被借总数
百年孤独	Gabriel-Garcia...	爱情	历史	9	0
飘	Margaret-...	爱情	战争	6	1
霍乱时期的爱情	Gabriel-Garcia...	爱情	社会	4	1
教父	Mario-Puzo	犯罪	历史	8	0
追忆似水年华	Marcel-Proust	自传	社会	6	1
百年孤独	Gabriel-Garcia...	爱情	魔幻	12	0
冰与火之歌	George-R.R.-...	奇幻	冒险	19	0
- Screenshot 3 (Bottom Left):** Search by Type. The search term is "幻". Results include:

书名	作者	类型一	类型二	在架数量	被借总数
银河帝国	Isaac-Asimov	科幻	冒险	0	1
三体	刘慈欣	科幻	悬疑	8	2
哈利·波特与魔法石	J.K.-Rowling	奇幻	冒险	25	0
小王子	Antoine-de-...	哲学	奇幻	20	0
地心游记	Jules-Verne	冒险	科幻	12	0
银河系漫游指南	Douglas-Adams	科幻	冒险	16	1
百年孤独	Gabriel-Garcia...	爱情	魔幻	12	0
海底两万里	Jules-Verne	冒险	科幻	14	0
冰与火之歌	George-R.R.-...	奇幻	冒险	19	0
银河	Isaac-Asimov	科幻	冒险	15	0
- Screenshot 4 (Bottom Right):** Search by ID. The search term is "16". Results include:

ID	书名	作者	类型一	类型二	在架数量	被借总数
1 16	基督山伯爵	Alexandre-...	冒险	犯罪	18	0

此时若点击右侧的刷新页面按钮会重新显示全部记录

点击申请借阅按钮会首先检测点击操作的合法性，若鼠标当前并无选中，则弹出报错对话：



检测点击“申请借阅”的合法性时，若鼠标当前选中的记录的图书在架数量小于1，则弹出提示对话：



若点击“申请借阅”的操作合法，则弹出确认操作对话：



若点击确定，则会弹出成功申请的对话。此时程序已经通过Sql语句插入了一条借阅状态为“申请借阅”的record：



点击导航栏中的“退出登录”或程序右上角的关闭按钮，则弹出确认操作对话，若点击确定则退出程序：



点击导航栏中的“切换账号”会弹出确认操作对话，若点击确定则回到登录界面：



(3) 管理员主页面

“图书管理”子页面中的搜索，刷新功能与用户页面的“图书查询”子页面中的对应功能完全相同，不再赘述：



The screenshot shows the Admin Main Page with a sidebar on the left containing links: '用户管理', '借阅记录', '统计分析', '退出登录', and '切换账号'. The main area is a table titled '图书管理' with columns: id, 书名 (Book Name), 作者 (Author), 类型一 (Type 1), 类型二 (Type 2), 在架数量 (On-shelf Quantity), and 被借总数 (Total Loans). The table contains 13 rows of book data.

		书名	搜索	刷新页面	添加图书	修改信息	删除图书	
		id	书名	作者	类型一	类型二	在架数量	被借总数
1	9	银河帝国	Isaac...	科幻	冒险	0	1	
2	10	时间简史	Stephen...	科学	哲学	8	0	
3	11	活着	余华	历史	社会	12	1	
4	12	三体	刘慈欣	科幻	悬疑	8	2	
5	13	神秘岛	Jules-Verne	冒险	推理	10	0	
6	14	百年孤独	Gabriel...	爱情	历史	9	0	
7	15	哈利·波特与...	J.K.-Rowling	奇幻	冒险	25	0	
8	16	基督山伯爵	Alexandre...	冒险	犯罪	18	0	
9	17	麦田里的守...	J.D.-Salinger	青春	文学	11	0	
10	18	1984	George-...	政治	历史	9	0	
11	19	红楼梦	曹雪芹	爱情	社会	14	0	
12	20	追风筝的人	Khaled...	自传	战争	12	0	
13	21	飘	Margaret...	爱情	战争	6	1	

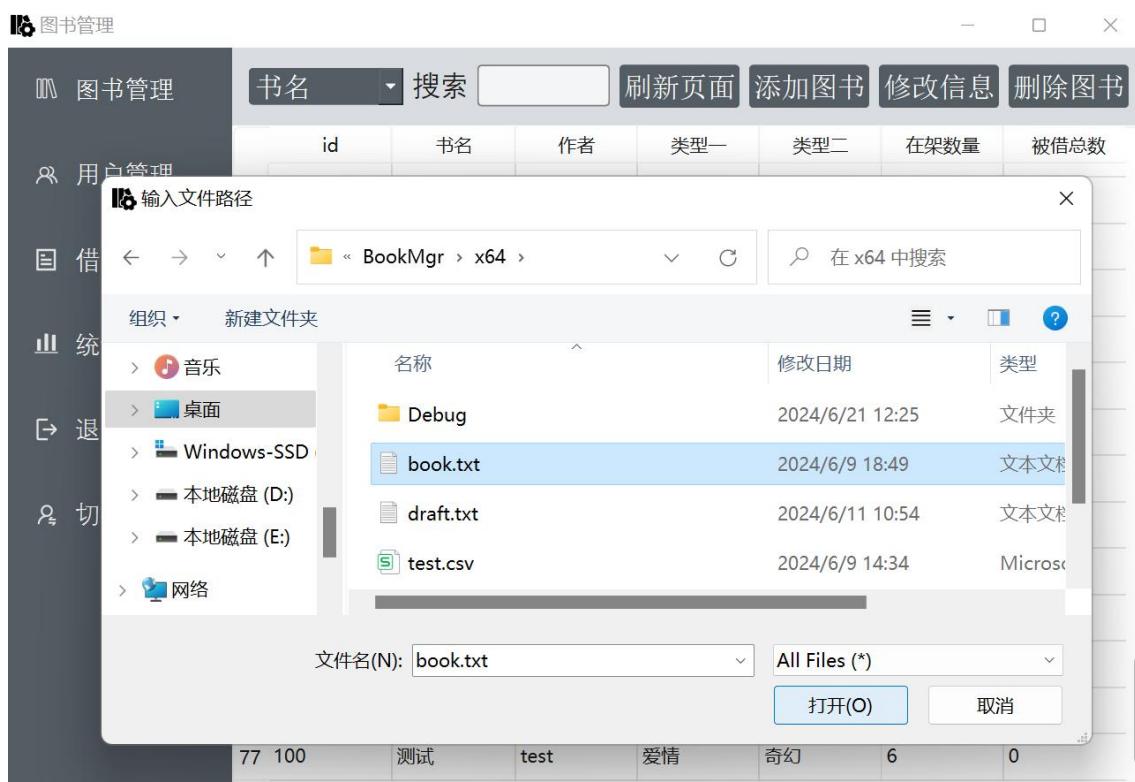
点击“添加图书”按钮之后，会弹出选择对话确认添加图书的方式为单个添加还是批量导入：



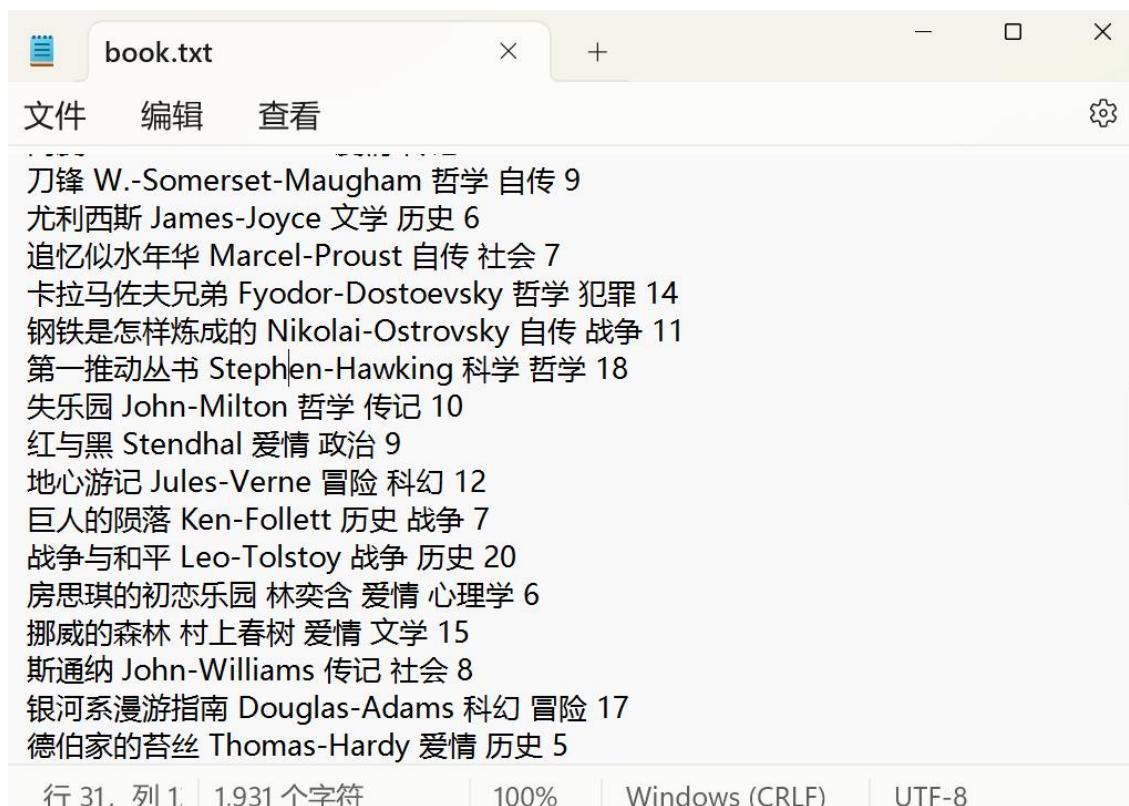
选择“单个添加”之后会弹出添加对话，输入相关属性之后点击确定则成功录入图书，点击取消则取消本次操作。此时程序通过Sql语句实现插入一条book数据：



选择“批量导入”之后会弹出选择文件路径对话，输入目标文件路径之后点击打开则会自动读取文件内容，新添图书。此时程序通过Sql语句实现多线程插入若干条book数据：



对于“批量导入”操作，要求文件格式为.txt或.csv。每条图书数据占一行。若一行中有多条图书数据则只读取第一条。每条图书数据的不同属性字段之间可以使用全角或半角的逗号或分号，或者若干空格来分隔。程序通过正则表达式实现分隔符号的设置：



The screenshot shows an Excel spreadsheet with the title bar 'book - 副本.csv'. The data is organized into columns A through E:

	A	B	C	D	E
1	世界是平的Thomas-Fr	政治	经济		17
2	思考快与慢Daniel-Kal	心理学	科学		6
3	史蒂夫·乔 Walter-I	传记	商业		13

点击“修改信息”按钮会首先检测点击操作的合法性，若鼠标当前并无选中，则弹出报错对话：



若“修改信息”操作合法，则弹出修改信息对话。此时输入框中显示当前的图书信息，修改之后点击确定即完成修改。程序通过Sql语句完成对book表中当前bookid的数据的更新：



点击“删除图书”按钮会首先检测点击操作的合法性，若鼠标当前并无选中，则弹出报错对话：



选中图书且在确认操作对话中选择确认删除操作之后。若被选中图书当前有外借，即在record表中有bookid为当前图书id且借阅状态request不为‘已归还’的借阅记录，则判定删除操作非法并弹出报错对话：



若删除操作合法，则弹出删除成功提示。程序通过Sql语句在book表中删除了bookid为选中图书id的数据：



The screenshot shows a book management interface with a sidebar containing links like '用户管理', '借阅记录', '统计分析', '退出登录', and '切换账号'. The main area has a search bar and a table with columns: id, 书名 (Book Name), 作者 (Author), 类型一 (Type 1), 类型二 (Type 2), 在架数量 (On-shelf Quantity), and 被借总数 (Total Loans). A row for book ID 72 is selected. A modal dialog box in the center says '删除成功!' (Delete Success!).

id	书名	作者	类型一	类型二	在架数量	被借总数
62 71	论摄影	Susan...	艺术	哲学	7	0
63 72	冰与火之歌	George...	奇幻	冒险	19	0
64 73	查令十字街...	Helene...	自传	文学	11	0
65 74	鬼谷子		政治		13	0
66 76	从零到一		创业		12	0
67 77	重新发现		经济		9	0
68 78	道德情操论	Adam...	哲学	经济	15	0
69 79	西窗法雨	刘星	社会		2	0
70 84	小说		悬疑	爱情	1	0
71 86	孟子	孟轲	哲学	政治	4	0
72 94	银河	Isaac...	科幻	冒险	15	0
73 95	时间	Stephen...	科学	哲学	8	0
74 96	活	余华	历史	社会	12	0

在用户管理界面，根据组合框中选择的搜索属性和文本输入框中的搜索词显示用户信息。程序通过Sql语句在user表中根据搜索词对相应数据进行查询。用户名实现模糊搜索：



The screenshots show three separate user management interfaces. Each has a sidebar with '用户管理' and '借阅记录'. The top one has a search bar for 'ID' and finds '17'. The middle one has a search bar for '用户名' and finds 'com'. The bottom one has a search bar for '权限' and finds '管理员'.

ID	用户名	密码	借阅总数	权限
1 17	PulsarSeeker	*****	14	

用户名	密码	借阅总数	权限
CometHunter	*****	21	
CometVoyager	*****	20	
CometDreamer	*****	22	

ID	用户名	密码	借阅总数	权限
2 4	ZhangWei	123456	10	管理员
3 11	NebulaWatcher	*****	9	管理员
4 20	SkyStrider	*****	17	管理员

在用户管理界面，根据组合框中选择的搜索属性和文本输入框中的搜索词显示用户信息。程序通过Sql语句在user表中根据搜索词对相应数据进行查询。用户名实现模糊搜索：

The figure consists of three vertically stacked screenshots of a user management application. Each screenshot shows a table with columns: id, 用户名 (Username), 密码 (Password), 借阅总数 (Loan Total), and 权限 (Permission). The first screenshot shows a search for '17' under 'ID'. The second shows a search for 'com' under '用户名'. The third shows a search for '管理员' under '权限'.

	id	用户名	密码	借阅总数	权限
1	17	PulsarSeeker	*****	14	
1	21	CometHunter	*****	21	
2	28	CometVoyager	*****	20	
3	44	CometDreamer	*****	22	

点击“导入用户”按钮之后，会弹出选择对话确认导入用户的方式为单个添加还是批量导入：

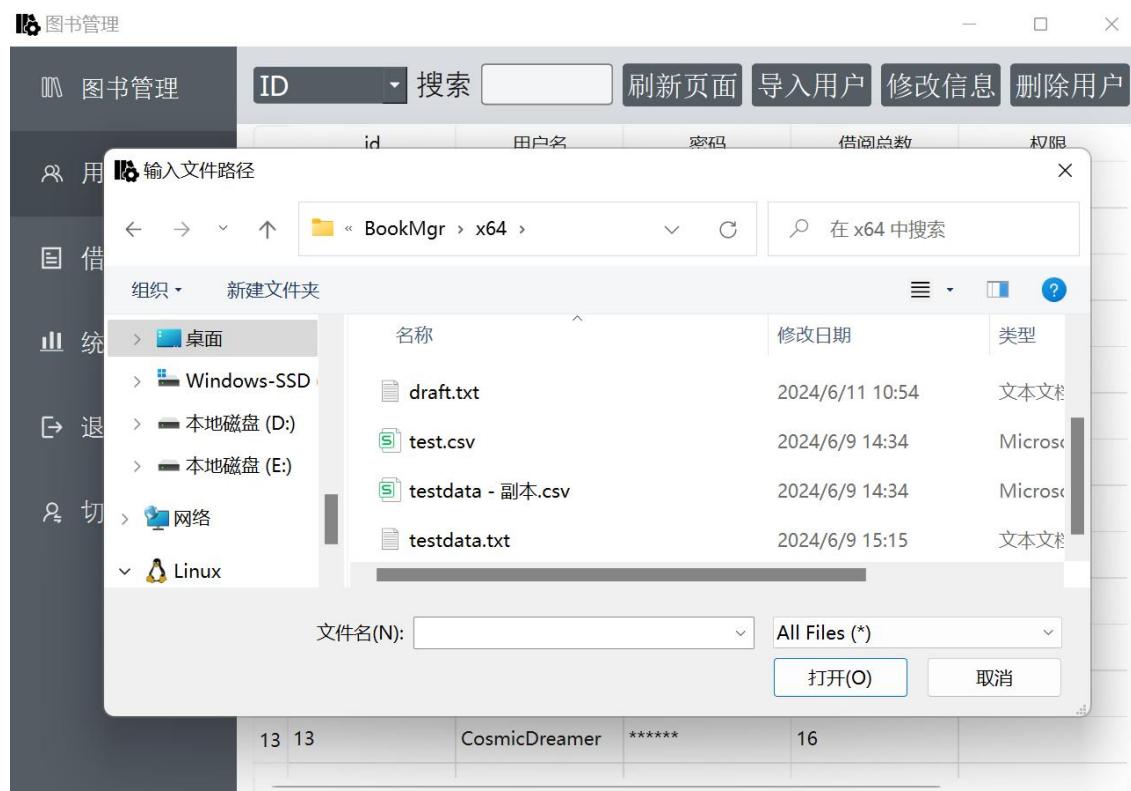
The screenshot shows the user management interface with a modal dialog titled "添加方式?" (Addition Method?). The dialog contains two buttons: "单个添加" (Add One) and "批量导入" (Batch Import). The main table lists 13 users with columns: id, 用户名 (Username), 密码 (Password), 借阅总数 (Loan Total), and 权限 (Permission).

	id	用户名	密码	借阅总数	权限
1	1	kobe	*****	3	
2	2		*****		
3	3		*****	0	管理员
4	4			10	管理员
5	5			16	
6	6			20	
7	7			5	
8	8	GalaxyHunter	*****	8	
9	9	SolarFlareX	*****	12	
10	10	QuantumLeapZ	*****	22	
11	11	NebulaWatcher	*****	9	管理员
12	12	AstroVoyager	*****	18	
13	13	CosmicDreamer	*****	16	

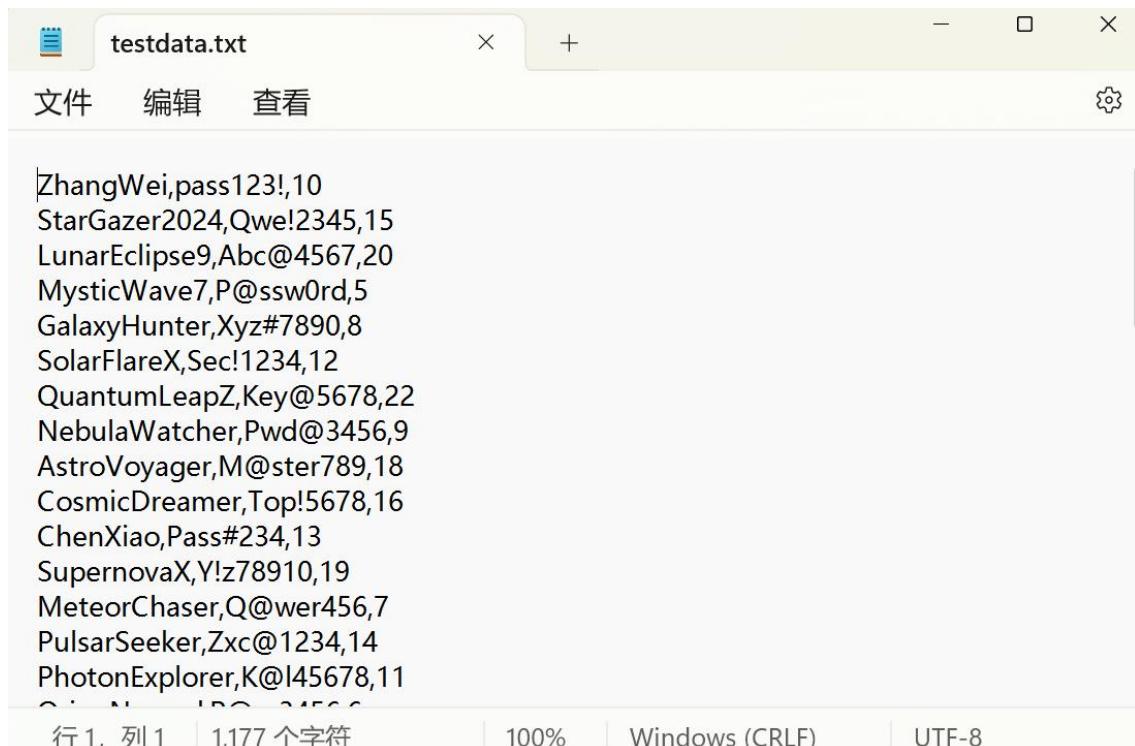
选择“单个添加”之后会弹出添加对话，输入相关属性之后点击添加则成功录入用户，点击取消则取消本次操作。此时程序通过Sql语句实现在user表中插入一条数据，初始密码为“123456”，借阅总数为0，权限为普通用户：



选择“批量导入”之后会弹出选择文件路径对话，输入目标文件路径之后点击打开则会自动读取文件内容，新添图书。此时程序通过Sql语句实现多线程插入若干条book数据：



“批量导入”用户的操作所需要的文件格式与“批量导入”图书所需要的文件格式基本相同：要求文件格式为.txt或.csv。每条用户数据占一行。若一行中有多条用户数据则只读取第一条。每条用户数据的不同属性字段之间可以使用全角或半角的逗号或分号，或者若干空格来分隔。



The screenshot shows a text editor window titled "testdata.txt". The content of the file is as follows:

```
ZhangWei,pass123!,10
StarGazer2024,Qwe!2345,15
LunarEclipse9,Abc@4567,20
MysticWave7,P@ssw0rd,5
GalaxyHunter,Xyz#7890,8
SolarFlareX,Sec!1234,12
QuantumLeapZ,Key@5678,22
NebulaWatcher,Pwd@3456,9
AstroVoyager,M@ster789,18
CosmicDreamer,Top!5678,16
ChenXiao,Pass#234,13
SupernovaX,Y!z78910,19
MeteorChaser,Q@wer456,7
PulsarSeeker,Zxc@1234,14
PhotonExplorer,K@l45678,11
```

At the bottom of the editor, it shows "行 1, 列 1 | 1,177 个字符" (Line 1, Column 1 | 1,177 characters), "100%", "Windows (CRLF)", and "UTF-8".

如果用户的密码为初始密码，在用户登录时会弹出提示信息，提醒尽快修改密码：



点击“修改信息”按钮会首先检测点击操作的合法性，若鼠标当前并无选中，则弹出报错对话：



若“修改信息”操作合法，则弹出确认修改信息操作对话。共“激活权限”和“重置密码”两种修改操作：



点击“激活权限”之后，若当前选中用户已经拥有管理员权限，则判定操作非法，弹出报错对话：

ID	用户名	密码	借阅总数	权限
1	kobe	*****	3	
2		*****		
3		*****	0	管理员
4			10	管理员
5			16	
6			20	
7			5	
8	GalaxyHunter	*****	8	
9	SolarFlareX	*****	12	
10	QuantumLeapZ	*****	22	
11	NebulaWatcher	*****	9	管理员
12	AstroVoyager	*****	18	
13	CosmicDreamer	*****	16	

若“激活权限”合法，则在确认操作对话中选择确定之后成功激活所选用户的管理员权限，页面自动刷新之后即可显示，此后该用户即可登录管理员端。程序通过Sql语句更新user表中userid为当前选中用户id的权限为“管理员”：

ID	用户名	密码	借阅总数	权限
1	kobe	*****	3	
2		*****		
3		*****	0	管理员
4			10	管理员
5			16	
6			20	
7			5	
8	GalaxyHunter	*****	8	
9	SolarFlareX	*****	12	管理员
10	QuantumLeapZ	*****	22	
11	NebulaWatcher	*****	9	管理员
12	AstroVoyager	*****	18	
13	CosmicDreamer	*****	16	

点击“重置密码”之后，若当前选中用户的密码为初始密码，则判定操作非法，弹出报错对话：

The screenshot shows a user interface for managing library users. On the left is a sidebar with options like '图书管理', '用户管理' (selected), '借阅记录', '统计分析', '退出登录', and '切换账号'. The main area has a table with columns: ID, 用户名 (Username), 密码 (Password), 借阅总数 (Loan Count), and 权限 (Permissions). A modal dialog box in the center says '密码已重置!' (Password Reset!). The table data is as follows:

ID	用户名	密码	借阅总数	权限
1 1	kobe	*****	3	
2 2		*****		
3 3		*****	0	管理员
4 4		*****	10	管理员
5 5			16	
6 6			20	
7 7	lmysticwave7	123456	5	
8 8	GalaxyHunter	*****	8	
9 9	SolarFlareX	*****	12	管理员
10 10	QuantumLeapZ	*****	22	
11 11	NebulaWatcher	*****	9	管理员
12 12	AstroVoyager	*****	18	
13 13	CosmicDreamer	*****	16	

若“重置密码”合法，则在确认操作对话中选择确定之后成功重置所选用户的密码，页面自动刷新之后该用户的密码由不可见状态变为可见。程序通过Sql语句更新user表中userid为当前选中用户id的密码为初始密码“123456”：

The screenshot shows the same user interface as the previous one, but with a different modal dialog. It asks '确认重置?' (Confirm Reset?) with 'Yes' and 'No' buttons. The table data is identical to the first screenshot. The 'Yes' button is highlighted.

ID	用户名	密码	借阅总数	权限
1 1	kobe	*****	3	
2 2		*****		
3 3		*****	0	管理员
4 4		*****	10	管理员
5 5			16	
6 6			20	
7 7	lmysticwave7	123456	5	
8 8	GalaxyHunter	*****	8	
9 9	SolarFlareX	*****	12	管理员
10 10	QuantumLeapZ	*****	22	
11 11	NebulaWatcher	*****	9	管理员
12 12	AstroVoyager	*****	18	
13 13	CosmicDreamer	*****	16	

点击“删除用户”按钮会首先检测点击操作的合法性，若鼠标当前并无选中，则弹出报错对话：



若选中的用户有已借未还或者未被管理员审核归还的图书，则判定删除操作非法，弹出报错信息：



若删除操作合法，则在确认操作对话中选择确定即完成删除操作。程序通过Sql语句实现删除user表中userid为当前选中用户id的数据信息：



借阅记录管理界面的搜索功能与用户端个人信息页面的借阅记录搜索功能完全相同，根据选择的借阅状态展示借阅记录：



此时若选择组合框中“全部”或者点击右侧的刷新页面按钮会重新显示全部记录。

在借阅记录管理页面点击“审核还书”按钮会首先检测点击操作的合法性，若鼠标当前并无选中，则弹出报错对话：



若选中的记录对应的借阅状态不为“申请归还”，则判定此次操作非法，弹出报错对话：



若判定操作合法，则在确认操作对话中选择确定之后会弹出操作成功信息。程序通过Sql语句更新record表中recordid为当前id的数据的借阅状态属性为“已归还”：



The screenshot shows a library management application window. On the left is a sidebar with navigation links: 图书管理 (selected), 用户管理, 借阅记录, 统计分析, 退出登录, and 切换账号. The main area has tabs: 全部 (selected), 刷新页面, 审核还书, and 批准借书. A table lists borrow records with columns: id, 用户id, 图书id, 图书名称, 借阅时间, 应还时间, and 借阅状态. Row 4 (id 12) has its status changed to '已归还'. A modal dialog box in the center says '批准成功!' (Approved successfully!).

	id	用户id	图书id	图书名称	借阅时间	应还时间	借阅状态
1	9	2	12	三体	2024-06-11	2024-09-0...	已归还
2	10	2	11	活着	2024-06-11	2024-09-0...	已归还
3	11	5	24	霍乱时期的...	2024-06-11	2024-09-0...	已借未还
4	12	2	40	本	2024-06-13	2024-09-1...	申请归还
5	13	1	40	可帝国	2024-06-13	2024-09-1...	申请归还
6	14	2	40		2024-06-13	2024-09-1...	已归还
7	15	1	40	银河系漫游...	2024-06-16	2024-09-1...	已借未还
8	16	2	36	追忆似水年...	2024-06-16	2024-09-1...	申请归还

点击“批准借书”按钮会首先检测点击操作的合法性，若鼠标当前并无选中，则弹出报错对话：



The screenshot shows the same library management application window. The sidebar and tabs are identical. The table lists borrow records. Row 4 (id 16) is selected. A modal dialog box in the center says '没有选中记录!' (No selected records!).

	id	用户id	图书id	图书名称	借阅时间	应还时间	借阅状态
1	9	2	12	三体	2024-06-11	2024-09-0...	申请归还
2	12	2	12	三体	2024-06-13	2024-09-1...	申请归还
3	13	1	9	银河帝国	2024-06-13	2024-09-1...	申请归还
4	16	2	40	冰年华	2024-06-16	2024-09-1...	申请归还
5	20	1	40	之歌	2024-06-21	2024-09-1...	申请归还

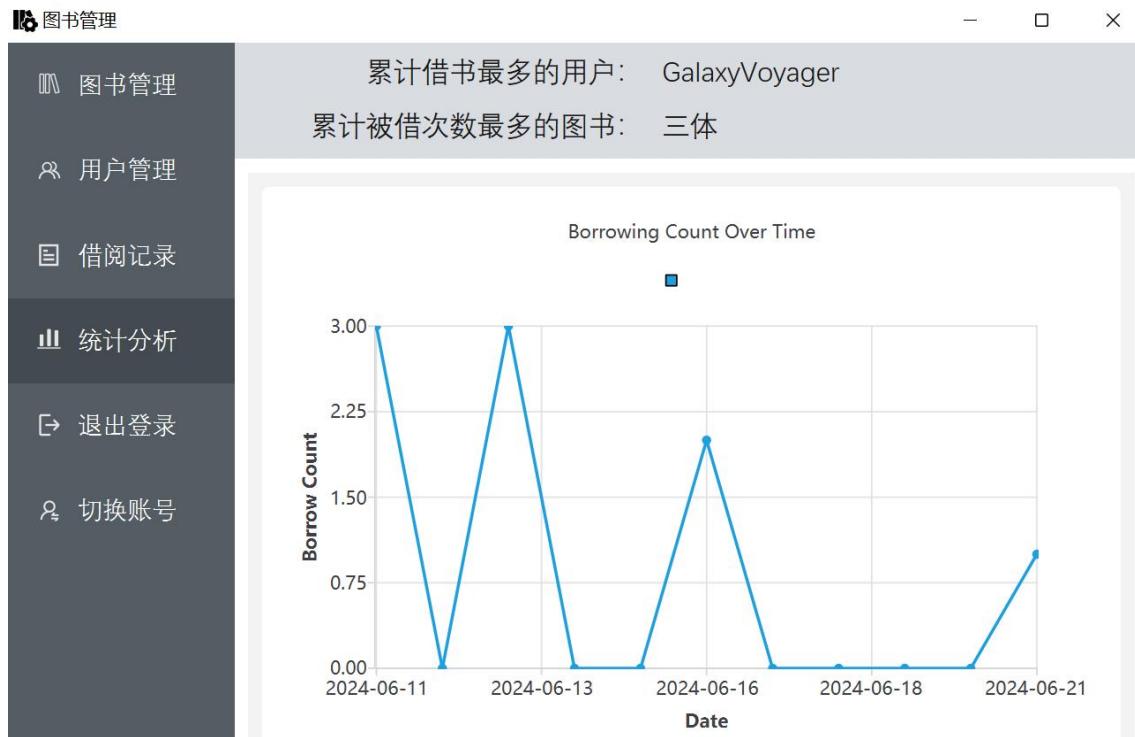
若选中的记录对应的借阅状态不为“申请借阅”，则判定此次操作非法，弹出报错对话：



若判定操作合法，则在确认操作对话中选择确定后会弹出操作成功信息。程序通过Sql语句更新record表中recordid为当前id的数据的借阅状态属性为“已借未还”，并更新记录的节约时间为当前日期（精确到天），应还时间为90天后的23:59分：



统计分析界面随机显示一个累计结束最多的用户，一本累计被借次数最多的图书。并且在下方显示用户借阅量随日期的变化趋势。程序通过Sql语句查询到全部的具有起始日期的借阅记录并按照日期统计借阅量，绘制成折线图：



管理员主界面的退出登录和切换账号功能与用户主界面的对应功能完全相同，这里不再赘述。

6. 总结

在此次大作业的编写中，我收获颇多。其中最大的收获是学习了Qt编程的基本范式，了解了Qt开发的基础方法。对于Qt涉及到的ui设计，信号与槽机制和Qt标准类和对象形成了初步认识。Qt编程在页面上开发具体功能的思路和方式大大简化了编写一个具有图形界面且交互较便利的图书管理系统的工作量和进程。

此外，我也进一步巩固了SQLite数据库和SQL语言的基础知识。通过数据库和数据库语言，程序处理数据的进程实现标准化，便捷化。利用Qt内置的 QSqlDatabase 类派生出与程序的源数据库进行实时连接的类 SqlMgr，并在程序中使用C++编程的单例模式，这使得所有与数据查询和修改的操作全部都通过 SqlMgr 类的唯一对象接口来调用函数实现。这样做消除了不同层次的类之间不必要的耦合。这一点与程序中采用的 Qt 独有的信号与槽机制作用类似，通过传递信号来触发槽函数在不破坏类的封装性的基础上便捷高效的实现了类之间的信息交流。

通过在VS中配置Qt开发环境，编写这次大作业既能够使用Qt编程的基本工具，又能够利用VS强大的调试能力来对程序进行调试。调试过程中使用QDebug类在控制台进行输出，尽管没有使用基本的断点调试，但通过在程序的相应位置添加 qDebug 输出语句也能快速的定位和修改错误。

此图书管理系统也有一些不足之处，比如界面设计可以进行进一步的美化，管理员端口的统计分析界面绘制的折线图可以添加更多显示属性和功能，代码层面的复用和整合还能够进一步增强等等。

总之，这次大作业锻炼了我面向对象编程的基本能力，拓展了C++编程的一些常用衍生工具的使用，也使我对编写小型应用程序的整体流程有了一个初步的认识。

二、源程序清单

1. 说明

在运行项目之后Qt的MOC工具，UIC工具和RCC工具会自动生成处理信号与槽和动态属性系统，创建和操作Qt Designer设计的界面以及在程序中嵌入和访问资源文件的代码，这些代码根据开发者设计的. ui, . h, . cpp文件生成。故在此不展示这些自动生成文件和无法以代码形式查看的. ui文件。

2. 头文件

(1) add_singleuser.h

```
#pragma once
////////////////////////////////////////////////////////////////
//add_singleuser.h : 添加单个用户的对话

#include <QDialog>
#include "ui_add_singleuser.h"           //UIC工具自动生成的头文件

QT_BEGIN_NAMESPACE
namespace Ui { class Add_SingleUserClass; }
QT_END_NAMESPACE

class Add_SingleUser : public QDialog
{
    Q_OBJECT

public:
    Add_SingleUser(QWidget *parent = nullptr);
    ~Add_SingleUser();

public slots:
    void on_btn_add_clicked();           //添加按钮功能实现
    void on_btn_cancel_clicked();        //取消按钮功能实现

private:
    Ui::Add_SingleUserClass *ui;        //界面ui作为指针类型成员
};
```

(2) au_singlebook.h

```
#pragma once

////////////////////////////////////////////////////////////////
//au_singlebook.h : 添加/修改单个图书的对话

#include <QDialog>
#include "ui_au_singlebook.h"

QT_BEGIN_NAMESPACE
namespace Ui { class AU_SingleBookClass; }
QT_END_NAMESPACE

class AU_SingleBook : public QDialog
{
    Q_OBJECT

public:
    AU_SingleBook(QWidget *parent = nullptr);
    ~AU_SingleBook();

    void set_type(QString,QStringList);           //更新信息时将原始信息显示到对应控件

public slots:
    void on_btn_ok_clicked();                     //确认按钮功能实现
    void on_btn_cancel_clicked();                 //取消按钮功能实现

private:
    Ui::AU_SingleBookClass *ui;

    QString m_id="-1";                          //值为-1时实现添加, 否则实现更新
};
```

(3) base.h

```
#pragma once

///////////////////////////////
//base.h : 继承自 QMainWindow类的抽象类, 用于派生主界面类

#include <QMainWindow>

class Base : public QMainWindow
{
    Q_OBJECT

public:
    Base(QWidget* parent = nullptr);
    virtual ~Base();

    virtual void initpage()=0;
    virtual void pop_message() = 0;      //将初始化界面和登录提示设置为纯虚函数

    void closeEvent(QCloseEvent* e);     //鼠标点击关闭事件的对话

signals:
    void switch_acnt();               //将鼠标点击切换账号设置成信号

public slots:
    void on_btn_switch_clicked();     //切换账号按钮功能实现
    void on_btn_exit_clicked();       //退出登录按钮功能实现
    //抽象类不需要添加ui界面
};
```

(4) cell_analysis.h

```
#pragma once

///////////////////////////////
//cell_analysis.h : 管理员主界面的统计分析页面

#include <QWidget>
#include "ui_cell_analysis.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Cell_AnalysisClass; }
QT_END_NAMESPACE

class Cell_Analysis : public QWidget
{
    Q_OBJECT

public:
    Cell_Analysis(QWidget *parent = nullptr);
    ~Cell_Analysis();

    void init();           //初始化整个界面
    void init_label();     //初始化文字标签
    void init_chart();     //初始化图表
    void clear_container(); //在初始化图表之前清除之前存在的图表
private:
    Ui::Cell_AnalysisClass *ui;
};
```

(5) cell_bkbrw.h

```
#pragma once

///////////////////////////////
//cell_bkbrw.h : 用户主界面的图书查询/借阅页面

#include <QWidget>
#include <QStandardItemModel>
#include "ui_cell_bkbrw.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Cell_BkBrwClass; }
QT_END_NAMESPACE

class Cell_BkBrw : public QWidget
{
    Q_OBJECT

public:
    Cell_BkBrw(QWidget *parent = nullptr);
    ~Cell_BkBrw();

    void init(QString str_cond=""); //初始化个人借阅记录
    void getid(QString self_name); //根据登录时的用户名查询用户信息,
                                    //并显示到对应控件上

public slots:
    void on_le_qryTextChanged(); //输入文本框文字变化后搜索功能实现
    void on_btn_brw_clicked(); //借阅按钮功能实现
    void on_btn_ref_clicked(); //刷新按钮功能实现

private:
    Ui::Cell_BkBrwClass *ui;
    QString self_id=""; //用户id
    QStandardItemModel model; //使用标准模型在界面控件上显示记录
};
```

(6) cell_bkmgr.h

```
#pragma once

///////////////////////////////
//cell_bkmgr.h : 管理员主界面的图书管理页面

#include <QWidget>
#include <qstandarditemmodel.h>
#include "ui_cell_bkmgr.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Cell_BkMgrClass; }
QT_END_NAMESPACE

class Cell_BkMgr : public QWidget
{
    Q_OBJECT

public:
    Cell_BkMgr(QWidget *parent = nullptr);
    ~Cell_BkMgr();

    void init(QString str_cond = ""); //初始化页面

public slots:
    void on_le_qryTextChanged();           //输入文本框文字变化后搜索功能实现
    void on_btn_imp_clicked();            //导入按钮功能实现
    void on_btn_dlt_clicked();            //删除按钮功能实现
    void on_btn_set_clicked();            //更新按钮功能实现
    void on_btn_ref_clicked();            //刷新按钮功能实现

private:
    Ui::Cell_BkMgrClass *ui;
    QStandardItemModel model;
};
```

(7) cell_rcdmgr.h

```
#pragma once

///////////////////////////////
//cell_rcdmgr.h : 管理员主界面的借阅记录管理页面

#include <QWidget>
#include <qstandarditemmodel.h>
#include "ui_cell_rcdmgr.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Cell_RcdMgrClass; }
QT_END_NAMESPACE

class Cell_RcdMgr : public QWidget
{
    Q_OBJECT

public:
    Cell_RcdMgr(QWidget *parent = nullptr);
    ~Cell_RcdMgr();

    void init(QString str_cond = "");           //初始化页面

public slots:
    void on_comboBox_currentIndexChanged();      //组合框索引改变后搜索功能实现
    void on_btn_vrf_clicked();                  //审核还书按钮功能实现
    void on_btn_ref_clicked();                 //刷新页面按钮功能实现
    void on_btn_apr_clicked();                 //批准借书按钮功能实现

private:
    Ui::Cell_RcdMgrClass *ui;
    QStandardItemModel model;
};
```

(8) cell_self.h

```

#pragma once

///////////////////////////////
//cell_rcdmgr.h : 管理员主界面的借阅记录管理页面

#include <QWidget>
#include <qstandarditemmodel.h>
#include "ui_cell_self.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Cell_SelfClass; }
QT_END_NAMESPACE

class Cell_Self : public QWidget
{
    Q_OBJECT

public:
    Cell_Self(QWidget *parent = nullptr);
    ~Cell_Self();

    void getid(QString self_name);           //根据登录输入的账号获取用户信息并显示在对应控件上
    void init(QString str_cond = "");        //初始化页面

signals:
    void show_pop();                        //登录成功后发送信号提示弹出登录信息

public slots:
    void on_comboBox_currentIndexChanged();   //组合框索引改变后搜索功能实现
    void on_btn_req_clicked();               //申请还书按钮功能实现
    void on_btn_ref_clicked();              //刷新页面按钮功能实现
    void on_btn_showpwd_clicked();          //显示密码按钮功能实现
    void on_btn_set_clicked();              //修改信息按钮功能实现
    void on_btn_save_clicked();             //保存修改按钮功能实现

    void rmd_rstpwd();                    //提示修改初始密码功能实现

private:
    Ui::Cell_SelfClass *ui;
    QString self_id="";                   //当前用户的id
    QString name = "";                   //当前用户的用户名
    QStandardItemModel model;            //用于显示个人借阅信息的模型
};

```

(9) cell_usermgr.h

```
#pragma once

///////////////////////////////
//cell_usermgr.h : 管理员主界面的用户信息管理页面

#include <QWidget>
#include "ui_cell_usermgr.h"
#include <QStandardItemModel>

QT_BEGIN_NAMESPACE
namespace Ui { class Cell_UserMgrClass; }
QT_END_NAMESPACE

class Cell_UserMgr : public QWidget
{
    Q_OBJECT

public:
    Cell_UserMgr(QWidget *parent = nullptr);
    ~Cell_UserMgr();

    void init(QString str_cond=""); //初始化页面

public slots:
    void on_le_qryTextChanged(); //输入文本框文本改变后搜索功能实现
    void on_btn_imp_clicked(); //导入按钮功能实现
    void on_btn_dlt_clicked(); //删除按钮功能实现
    void on_btn_set_clicked(); //修改按钮功能实现
    void on_btn_ref_clicked(); //刷新按钮功能实现

private:
    Ui::Cell_UserMgrClass *ui;
    QStandardItemModel model;
};
```

(10) dlg_login.h

```
#pragma once

////////////////////////////////////////////////////////////////
//dlg_login.h : 登录对话

#include <QDialog>
#include "ui_dlg_login.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Dlg_LoginClass; }
QT_END_NAMESPACE

class Dlg_Login : public QDialog
{
    Q_OBJECT

public:
    Dlg_Login(QWidget* parent = nullptr);
    ~Dlg_Login();

    void closeEvent(QCloseEvent* e); //鼠标点击关闭事件

signals:
    void admin_lgn(); //管理员成功登录信号
    void user_lgn(); //用户成功登录信号
    void log_success(const QString& username); //用户成功登录后传递用户名信号

public slots:
    void on_btn_userlgn_clicked(); //用户登录按钮功能实现
    void on_btn_adminlgn_clicked(); //管理员登录按钮功能实现

private:
    Ui::Dlg_LoginClass *ui;
};
```

(11) msgbox.h

```
#pragma once

///////////////////////////////
//msgbox.h : 自定义消息对话框类

#include <QMessageBox>

class MsgBox :public QMessageBox
{
public:
    //默认消息对话框：无父指针，无文本，无标题，按钮为Yes|No，图标为默认图标
    MsgBox(QWidget* parent = nullptr, const QString& text = " ",
            QIcon icon = QIcon(":/x64/Debug/resource/pic/default.png"),
            StandardButtons btns=Yes|No,
            const QString& title = " ", StandardButton dftbtn=NoButton);

    //修改按钮文本
    void set_btn_text(StandardButton btn, const QString& btn_txt);

    //'!'运算符重载
    MsgBox& operator !();

    //鼠标点击关闭事件重载
    void closeEvent(QCloseEvent* event) override;
};
```

(12) page_admin.h

```
#pragma once

///////////////////////////////
//page_admin.h : 管理员主界面

#include "base.h"
#include "ui_page_admin.h"
#include "cell_analysis.h"
#include "cell_bkmgr.h"
#include "cell_rcdmgr.h"
#include "cell_usermgr.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Page_AdminClass; }
QT_END_NAMESPACE

class Page_Admin : public Base
{
    Q_OBJECT

public:
    Page_Admin(QWidget *parent = nullptr);
    ~Page_Admin();

    void initpage();           //初始化界面
    void dealmenu();           //实现导航栏切换当前页面
    void pop_message();        //基类Base的对应抽象函数的实现，为管理员登录提示信息

private:
    Ui::Page_AdminClass *ui;

    Cell_Analysis* analy;     //统计分析页面类指针
    Cell_BkMgr* bkmgr;        //图书管理页面类指针
    Cell_RcdMgr* rcdmgr;      //借阅记录页面类指针
    Cell_UserMgr* usermgr;    //用户管理页面类指针

    int msg_cnt = 0;           //统计登录提示信息弹出次数
};
```

(13) page_user.h

```
#pragma once

///////////////////////////////
//page_user.h : 用户主界面

#include "ui_Page_user.h"
#include "base.h"
#include "cell_self.h"
#include "cell_bkbrw.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Page_userClass; }
QT_END_NAMESPACE

class Page_user : public Base
{
    Q_OBJECT

public:
    Page_user(QWidget *parent = nullptr);
    ~Page_user();

    void dealmenu();           //实现导航栏切换当前页面
    void pop_message();        //基类Base的对应抽象函数的实现，为用户登录提示信息

public slots:
    void initpage();           //初始化页面

    //根据登录成功之后登录对话发送的用户名，设置两个子页面需要的当前用户信息
    void set_username(const QString& username);

private:
    Ui::Page_userClass *ui;

    Cell_BkBrw* bkbrw;        //图书查询/借阅页面类指针
    Cell_Self* self;           //个人信息页面类指针
};
```

(14) sqlmgr.h

```

#pragma once

///////////////////////////////
//sqlmgr.h : Sqlite数据库处理类

#include <QSqlDataBase>

class SqlMgr
{
public:
    SqlMgr();
    ~SqlMgr();

    //采用单例模式，全局只创建唯一对象
    static SqlMgr* instance;
    static SqlMgr* getinstance();

    //初始化
    void init();

    //登录逻辑实现
    bool login(QString str_name, QString str_pwd, int admin);

    //获取用户
    QVector<QStringList> get_users(QString str_cond = "");

    //单个添加用户
    void add_user(QString str_name, QString& str_admin);

    //批量导入用户
    void add_user(QVector<QStringList>);

    //激活用户权限
    void update_users(QString str_id="1",int bhv=0);

    //修改用户账号密码
    void set_userinfo(QString id,QString username,QString password);

    //删除用户
    bool dlt_user(QString str_id);

    //获取图书
    QVector<QStringList> get_books(QString str_cond = "");

    //添加图书
    void add_books(QVector<QStringList>);

    //修改图书信息
    void update_books(QStringList, QString);

    //删除图书
    bool dlt_books(QString str_id);

    //归还图书
    void return_books(QString recordid);
}

```

```
//申请借阅图书
void borrow_books(QString userid, QString bookid,QString bookname);

//批准借阅
void verify_borrow(QString recordid,QString userid, QString bookid);

//审核归还
void verify_return(QString recordid, QString bookid);

//获取借阅记录
QVector<QStringList> get_records(QString str_cond = "");

private:
    //数据库对象
    QSqlDatabase db;
};
```

3. 源文件

(1) add_singleuser.cpp

```
#include "add_singleuser.h"
#include "msgbox.h"
#include "sqlmgr.h"

Add_SingleUser::Add_SingleUser(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::Add_SingleUserClass())
{
    ui->setupUi(this);
    setWindowIcon(QIcon(":/x64/Debug/resource/pic/add.png"));
    //设置窗口图标
}

Add_SingleUser::~Add_SingleUser()
{
    delete ui;
}

void Add_SingleUser::on_btn_add_clicked()
{
    MsgBox msgbox = MsgBox(nullptr,
        "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 确认添加? < / div>",
        QIcon(":/x64/Debug/resource/pic/question.png"));
    //确认操作对话

    int ret = msgbox.exec();

    if (ret == QMessageBox::Yes)
    {
        int idx = ui->comboBox->currentIndex();
        QString str_admin = idx ? "管理员" : "";
        QString str_name = ui->le_name->text();

        SqlMgr::getInstance()->add_user(str_name, str_admin);
        //新添一条用户名为文本框le_name中输入的文本，身份为str_admin的用户数据
    }
    this->close();
    //完成添加操作后关闭窗口
}

void Add_SingleUser::on_btn_cancel_clicked()
{
    this->close();
}
```

(2) au_singlebook.cpp

```
#include "au_singlebook.h"
#include "sqlmgr.h"

AU_SingleBook::AU_SingleBook(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::AU_SingleBookClass())
{
    ui->setupUi(this);
    setWindowIcon(QIcon(":/x64/Debug/resource/pic/admin.png"));
    //设置窗口图标
}

AU_SingleBook::~AU_SingleBook()
{
    delete ui;
}

void AU_SingleBook::set_type(QString id, QStringList info)
{
    //当更新信息时，将传入的原始信息显示到页面的对应控件上
    m_id = id;
    ui->le_name->setText(info[0]);
    ui->le_author->setText(info[1]);
    ui->cb1->setCurrentText(info[2]);
    ui->cb2->setCurrentText(info[3]);
    ui->spinBox->setValue(info[4].toInt());
}

void AU_SingleBook::on_btn_ok_clicked()
{
    //将修改后的信息存入字符串链表中
    QStringList l;
    l << ui->le_name->text();
    l << ui->le_author->text();
    l << ui->cb1->currentText();
    l << ui->cb2->currentText();
    l << QString::number(ui->spinBox->value());
    l << QString::number(0);

    if (m_id != QString("-1"))
    {
        //更新信息分支
        SqlMgr::getInstance()->update_books(l, m_id);
    }
    else
    {
        //插入信息分支
        QVector<QStringList> v;
        v.push_back(l);
        SqlMgr::getInstance()->add_books(v);
    }
    //完成操作后关闭对话
    this->close();
}

void AU_SingleBook::on_btn_cancel_clicked()
{
    this->close();
}
```

(3) base.cpp

```

#include "base.h"
#include "msgbox.h"
#include "dlg_login.h"
#include <QCloseEvent>

Base::Base(QWidget* parent) : QMainWindow(parent)
{};

Base::~Base()
{}

void Base::on_btn_exit_clicked()
{
    //退出程序逻辑的实现
    int ret = MsgBox(this,
                      "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 退出系统? < / div>",
                      QIcon(":/x64/Debug/resource/pic/exit.png")).exec();

    if (ret == QMessageBox::Yes) {
        exit(0);
    }
    else {
        return;
    }
}

void Base::on_btn_switch_clicked()
{
    //切换账号逻辑的实现
    int ret = MsgBox(this,
                      "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 切换账号? < / div>",
                      QIcon(":/x64/Debug/resource/pic/switch.png")).exec();

    //若选择切换账号，发送对应信号并隐藏当前窗口
    if (ret == QMessageBox::Yes) {
        emit switch_acnt();
        hide();
    }
    else {
        return;
    }
}

void Base::closeEvent(QCloseEvent* event)
{
    //鼠标点击关闭窗口事件
    int ret = MsgBox(this,
                      "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 退出系统? < / div>",
                      QIcon(":/x64/Debug/resource/pic/exit.png")).exec();

    if (ret == QMessageBox::Yes) {
        event->accept(); // 接受关闭事件，关闭窗口
    }
    else {
        event->ignore(); // 忽略关闭事件，保持窗口不变
    }
}

```

(4) cell_analysis.cpp

```

#include <QtWidgets/qboxlayout.h>
#include <QtCharts/qchart.h>
#include <QtCharts/qchartview.h>
#include <QtCharts/QBarSeries>
#include <QtCharts/QBarSet>
#include <QtCharts/QLineSeries>
#include <QtCharts/QDateTimeAxis>
#include <QtCharts/QValueAxis>
#include <QDate>
#include <QDateTime>
#include <QMap>
#include <QDebug>
#include <qrandom.h>
#include "sqlmgr.h"
#include "cell_analysis.h"

Cell_Analysis::Cell_Analysis(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Cell_AnalysisClass())
{
    ui->setupUi(this);
}

Cell_Analysis::~Cell_Analysis()
{
    delete ui;
}

void Cell_Analysis::init()
{
    init_label();
    init_chart();
}

void Cell_Analysis::init_label()
{
    //在数据库中查询借阅总数最多的用户和被借总数最多的图书，并在页面对应控件上各随机展示一个
    QVector<QStringList> l = SqlMgr::getInstance()->get_users("where brwcnt
= (select MAX(brwcnt) from user)");
    if (l.isEmpty())
        return;
    int rdm_useridx = QRandomGenerator::global()->bounded(l.size());
    QStringList user = l[rdm_useridx];
    qDebug() << user;

    QVector<QStringList> b = SqlMgr::getInstance()->get_books("where cnt =
(select MAX(cnt) from book)");
    if (b.isEmpty())
        return;
    int rdm_bkidx = QRandomGenerator::global()->bounded(b.size());
    QStringList book = b[rdm_bkidx];
    qDebug() << book;

    ui->lb_bookname->setText(book[1]);
    ui->lb_username->setText(user[1]);
}

```

```

void Cell_Analysis::init_chart()
{
    //初始化图表之前首先清空显示控件中已存在的其他内容
    clear_container();

    //获取所有具有起始借阅时间的记录
    QVector<QStringList> l = SqlMgr::getInstance()->get_records();
    QStringList brw_time;
    for (int i = 0; i < l.size(); i++)
    {
        QString item = l[i][4];
        if (QString("") != item)
            brw_time << item;
    }
    qDebug() << brw_time;

    QDate baseDate = QDate::fromString("2024-06-11", "yyyy-MM-dd"); // 基准日期

    //使用 QMap 统计每个日期的借阅记录数量
    QMap<QString, int> dateCounts;

    // 使用天数差作为索引
    for (int i = 0; i < brw_time.size(); i++) {
        dateCounts[brw_time[i]]++;
    }

    qDebug() << dateCounts;

    // 创建折线图数据
    QLineSeries* lineSeries = new QLineSeries();

    // 设置最小和最大日期
    QDateTime minDate = QDateTime::fromString("2024-06-11", "yyyy-MM-dd");
    QDateTime maxDate = QDateTime::currentDateTime();

    //获取范围内的所有日期
    QDate startDate = minDate.date();
    QDate endDate = maxDate.date();
    for (QDate date = startDate; date <= endDate; date = date.addDays(1)) {
        QString dateStr = date.toString("yyyy-MM-dd");
        int borrowCount = dateCounts.value(dateStr, 0);
        QDateTime dateTime = QDateTime::fromString(dateStr, "yyyy-MM-dd");
        qint64 timestamp = dateTime.toMsecsSinceEpoch();
        lineSeries->append(timestamp, borrowCount);
    }
    qDebug() << dateCounts;

    lineSeries->setPointsVisible(true);

    //创建图表对象并将数据添加到图表
    QChart* chart = new QChart();
    chart->setTitle("Borrowing Count Over Time");
    chart->addSeries(lineSeries);

    //设置轴
    QDateTimeAxis* axisX = new QDateTimeAxis;
    axisX->setFormat("yyyy-MM-dd");
    axisX->setTitleText("Date");

    QValueAxis* axisY = new QValueAxis;
    axisY->setTitleText("Borrow Count");
}

```

```
chart->addAxis(axisX, Qt::AlignBottom);
chart->addAxis(axisY, Qt::AlignLeft);
lineSeries->attachAxis(axisX);
lineSeries->attachAxis(axisY);

//创建一个ChartView来显示图表
QChartView* chartView = new QChartView(chart);
chartView->setRenderHint(QPainter::Antialiasing);

//创建一个布局并添加ChartView到chartContainer中
QLayout* existingLayout = ui->container->layout();
if (existingLayout) {
    existingLayout->addWidget(chartView);
}
else {
    // 如果没有布局，创建一个新的布局并设置
    QVBoxLayout* layout = new QVBoxLayout();
    layout->addWidget(chartView);
    ui->container->setLayout(layout);
}

void Cell_Analysis::clear_container()
{
    if (ui->container->layout() != nullptr) {
        QLayoutItem* item = ui->container->layout()->takeAt(0);
        if (item != nullptr) {
            QWidget* widget = item->widget();
            if (widget != nullptr) {
                delete widget; // 删除该widget
            }
            delete item; // 删除布局项
        }
    }
}
```

(5) cell_bkbrw.cpp

```

#include "cell_bkbrw.h"
#include "sqlmgr.h"
#include "msgbox.h"
#include <QDebug>

Cell_BkBrw::Cell_BkBrw(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Cell_BkBrwClass())
{
    ui->setupUi(this);
    //在tableView控件上使用标准模型来显示图书信息数据
    //设置信息为不可编辑，且只能选中单行
    ui->tableView->setModel(&model);
    ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
    ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);
}

Cell_BkBrw::~Cell_BkBrw()
{
    delete ui;
}

void Cell_BkBrw::init(QString str_cond)
{
    //若当前用户的id未能成功获得，无法初始化
    if (QString("") == self_id) return;

    //获取所有图书信息数据
    auto l = SqlMgr::getInstance()->get_books(str_cond);

    //使用模型显示数据
    model.clear();
    model.setHorizontalHeaderLabels(QStringList{ "id", "书名", "作者", "类型一",
    "类型二", "在架数量", "被借总数" });
    //适应页面变化
    ui->tableView->horizontalHeader()->setStretchLastSection(true);
    ui->tableView->horizontalHeader()-
    >setSectionResizeMode(QHeaderView::Stretch);
    for (int i = 0; i < l.size(); i++)
    {
        QList<QStandardItem*> items;
        for (int j = 0; j < l[i].size(); j++)
            items.append(new QStandardItem(l[i][j]));

        model.appendRow(items);
    }
}

void Cell_BkBrw::getid(QString self_name)
{
    //通过传入的用户名获取当前用户的id
    QString str_sql = QString("where username='%1' ")
        .arg(self_name);
    auto l = SqlMgr::getInstance()->get_users(str_sql);
    self_id = l[0][0];
    qDebug() << "BkBrw: " << l[0][0];
}

```

```

void Cell_BkBew::on_le_qryTextChanged()
{
    //结合组合框控件当前的搜索属性与文本输入框的已输入文本对图书信息进行查询
    int idx = ui->comboBox->currentIndex();
    QString str_cond;
    switch (idx)
    {
    case 0:
    {
        //Sql语句实现模糊查询
        str_cond = QString("where bookname like '%%1%'")
            .arg(ui->le_qry->text());
        break;
    }
    case 1:
    {
        str_cond = QString("where author like '%%1%'")
            .arg(ui->le_qry->text());
        break;
    }
    case 2:
    {
        str_cond = QString("where type1 like '%%1%' or type2 like '%%1%'")
            .arg(ui->le_qry->text());
        break;
    }
    case 3:
    {
        str_cond = QString("where bookid=%1")
            .arg(ui->le_qry->text());
        break;
    }
    default:
        return; break;
    }
    //根据查询结果重新初始化页面
    init(str_cond);
}

void Cell_BkBew::on_btn_brw_clicked()
{
    //借阅图书按钮功能实现
    int r = ui->tableView->currentIndex().row();

    //无选中判定
    if (r < 0)
    {
        (!MsgBox(this,
                  "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 没有选中图书! < / div>",
                  QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
        return;
    }

    int access = model.item(r, 5)->text().toInt();
    //无库存判定
    if (access < 1)
    {
        (!MsgBox(this,
                  "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 没有库存! < / div>",
                  QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
        return;
    }
}

```

```
//确认操作对话
int ret=MsgBox(this,
    "< div style = 'text-align: center; font-weight: bold; font-size:
20px;' > 确认申请? < / div>",
    QIcon(":/x64/Debug/resource/pic/question.png")).exec();

if (ret==QMessageBox::Yes)
{
    //新添借阅记录数据
    SqlMgr::getInstance()->borrow_books(self_id, model.item(r, 0)-
>text(), model.item(r, 1)->text());
    (!MsgBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-
size: 20px;' > 申请成功! < / div>",
        QIcon(":/x64/Debug/resource/pic/default.png))).exec();
}

void Cell_BkBrw::on_btn_ref_clicked()
{
    //刷新页面按钮功能实现
    init();
}
```

(6) cell_bkmgr.cpp

```

#include "cell_bkmgr.h"
#include "sqlmgr.h"
#include "msgbox.h"
#include "au_singlebook.h"
#include "qfiledialog.h"
#include <qregularexpression.h>
#include <QDeBug>

Cell_BkMgr::Cell_BkMgr(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Cell_BkMgrClass())
{
    ui->setupUi(this);
    //使用标准模型在tableview控件上显示图书信息
    ui->tableView->setModel(&model);
    ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
    ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);
}

Cell_BkMgr::~Cell_BkMgr()
{
    delete ui;
}

void Cell_BkMgr::init(QString str_cond)
{
    //根据查询关键词初始化页面
    auto l = SqlMgr::getInstance()->get_books(str_cond);
    model.clear();
    model.setHorizontalHeaderLabels(QStringList{ "id", "书名", "作者", "类型一",
    "类型二", "在架数量", "被借总数" });
    ui->tableView->horizontalHeader()->setStretchLastSection(true);
    ui->tableView->horizontalHeader()->
    setSectionResizeMode(QHeaderView::Stretch);
    for (int i = 0; i < l.size(); i++)
    {
        QList<QStandardItem*> items;
        for (int j = 0; j < l[i].size(); j++)
            items.append(new QStandardItem(l[i][j]));

        model.appendRow(items);
    }
}

void Cell_BkMgr::on_le_qryTextChanged()
{
    //结合组合框控件当前的搜索属性与文本输入框的已输入文本对图书信息进行查询
    int idx = ui->comboBox->currentIndex();
    QString str_cond;
    switch (idx)
    {
    case 0:
    {
        //Sql语句实现模糊查询
        str_cond = QString("where bookname like '%%1%'")
            .arg(ui->le_qry->text());
        break;
    }
}

```

```

case 1:
{
    str_cond = QString("where author like '%%1%'")
        .arg(ui->le_qry->text());
    break;
}
case 2:
{
    str_cond = QString("where type1 like '%%1%' or type2 like '%%1%'")
        .arg(ui->le_qry->text());
    break;
}
case 3:
{
    str_cond = QString("where bookid=%1")
        .arg(ui->le_qry->text());
    break;
}
default:
    return; break;
}
//根据查询结果重新初始化页面
init(str_cond);
}

void Cell_BkMgr::on_btn_imp_clicked()
{
    //导入按钮功能实现
    MsgBox msgbox = MsgBox(nullptr,
        "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 添加方式? < / div>",
        QIcon(":/x64/Debug/resource/pic/question.png"));

    msgbox.set_btn_text(QMessageBox::Yes, "单个添加");
    msgbox.set_btn_text(QMessageBox::No, "批量导入");
    int ret = msgbox.exec();

    if (ret == QMessageBox::Yes)
    {
        qDebug() << QString("单个添加");
        AU_SingleBook au_singlebook;
        au_singlebook.exec();
    }
    else if (ret == QMessageBox::No)
    {
        //批量导入，参考资料
        auto str_path = QFileDialog::getOpenFileName(nullptr, "输入文件路径");

        if (!str_path.isEmpty())
        {
            QFile f(str_path);
            f.open(QFile::ReadOnly);
            QVector<QStringList> vec_data;

            //使用正则表达式设置读取文件时的分隔符
            QRegularExpression re("[, , ; , \\\s]+");

            while (!f.atEnd())
            {
                //一次读取一行内容
                QString str = f.readLine().trimmed();

                //清除无关字符
                auto l = str.split(re, Qt::SkipEmptyParts);

```

```

//只读取当前行的第一条数据，若数据不完整，用""补全
    while (l.size() < 5) {
        l.append("");
    }
    l.append(QString::number(0));

    if (l.size() > 6) {
        l = l.mid(0, 6);
    }

    vec_data.push_back(l);
}
qDebug() << "\n\n";
qDebug() << vec_data;

SqlMgr::getInstance()->add_books(vec_data);
}
}
else { return; }
init();
}

void Cell_BkMgr::on_btn_set_clicked()
{
//更新图书按钮功能实现
int r = ui->tableView->currentIndex().row();
if (r < 0)
{
    (!MsgBox(this,
              "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 没有选中图书! < / div>",
              QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
    return;
}
else
{
    auto id = model.item(r, 0)->text();
    AU_SingleBook au_singlebook;

//将当前选中行的图书信息传到添加/更新图书信息对话
QStringList info;
info << model.item(r, 1)->text();
info << model.item(r, 2)->text();
info << model.item(r, 3)->text();
info << model.item(r, 4)->text();
info << model.item(r, 5)->text();

au_singlebook.set_type(QString(id),info);
au_singlebook.exec();
init();
}
}

void Cell_BkMgr::on_btn_dlt_clicked()
{
//删除按钮功能实现
int r = ui->tableView->currentIndex().row();

//无选中判定
if (r < 0)
{
    (!MsgBox(this,
              "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 没有选中图书! < / div>",
              QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
    return;
}
}

```

```
int ret = MsgBox(this,
    "< div style = 'text-align: center; font-weight: bold; font-size:
20px;' > 确认删除? < / div>",
    QIcon(":/x64/Debug/resource/pic/question.png")).exec();

if (ret == QMessageBox::Yes)
{
    QString id = model.item(r, 0)->text();
    QString str_sql = QString("where bookid = %1 and request != '已归还
'").arg(id);

    //非法删除判定：若当前图书有外借库存，判定删除操作非法
    if (SqlMgr::getInstance()->get_records(str_sql).size())
    {
        (!MsgBox(this,
            "< div style = 'text-align: center; font-weight: bold;
font-size: 20px;' > 删除失败！有外借库存！ < / div>",
            QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
        return;
    }
    SqlMgr::getInstance()->dlt_books(id);
    init();
}
}

void Cell_BkMgr::on_btn_ref_clicked()
{
    init();
}
```

(7) cell_rcdmgr.cpp

```
#include "cell_rcdmgr.h"
#include "sqlmgr.h"
#include "msgbox.h"
#include <qdebug.h>

//使用标准模型在tableView控件上显示借阅记录信息的功能与前面页面基本相同，不再赘述
Cell_RcdMgr::Cell_RcdMgr(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Cell_RcdMgrClass())
{
    ui->setupUi(this);
    ui->tableView->setModel(&model);
    ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
    ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);
}

Cell_RcdMgr::~Cell_RcdMgr()
{
    delete ui;
}

void Cell_RcdMgr::init(QString str_cond)
{
    auto l = SqlMgr::getInstance()->get_records(str_cond);
    model.clear();
    model.setHorizontalHeaderLabels(QStringList{ "id", "用户id", "图书id", "图书名称", "借阅时间", "应还时间", "借阅状态" });
    ui->tableView->horizontalHeader()->setStretchLastSection(true);
    ui->tableView->horizontalHeader()->setSectionResizeMode(QHeaderView::Stretch);
    for (int i = 0; i < l.size(); i++)
    {
        QList<QStandardItem*> items;
        for (int j = 0; j < l[i].size(); j++)
            items.append(new QStandardItem(l[i][j]));

        model.appendRow(items);
    }
}

void Cell_RcdMgr::on_comboBox_currentIndexChanged()
{
    //根据组合框控件当前的索引实现搜索功能
    int idx = ui->comboBox->currentIndex();
    QString str_cond;
    switch (idx)
    {
    case 1:
    {
        str_cond = QString("where request='申请借阅'");
        break;
    }
    case 2:
    {
        str_cond = QString("where request='已借未还'");
        break;
    }
    case 3:
    {
        str_cond = QString("where request='申请归还'");
        break;
    }
}
```

```

case 4:
{
    str_cond = QString("where request='已归还'");
    break;
}
default:
    break;
}
init(str_cond);
}

void Cell_RcdMgr::on_btn_vrf_clicked()
{
    //审核还书按钮功能实现
    int r = ui->tableView->currentIndex().row();

    //无选中判定
    if (r < 0)
    {
        (!MsgBox(this,
                  "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 没有选中记录! < / div>",
                  QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
        return;
    }

    //非法选中判定
    QString request = model.item(r, 6)->text();
    if (request!=QString("申请归还"))
    {
        (!MsgBox(this,
                  "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 无法批准还书! < / div>",
                  QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
        return;
    }
    int ret = MsgBox(this,
                      "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 确认批准还书? < / div>",
                      QIcon(":/x64/Debug/resource/pic/default.png")).exec();
    if (ret == QMessageBox::Yes)
    {
        SqlMgr::getInstance()->verify_return(model.item(r, 0)-
>text(),model.item(r, 2)->text());
        (!MsgBox(this,
                  "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 批准成功! < / div>",
                  QIcon(":/x64/Debug/resource/pic/default.png"))).exec();
    }
    init();
}

void Cell_RcdMgr::on_btn_apr_clicked()
{
    //批准借书按钮功能实现
    int r = ui->tableView->currentIndex().row();

    //无选中判定
    if (r < 0)
    {
        (!MsgBox(this,
                  "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 没有选中记录! < / div>",
                  QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
        return;
    }
}

```

```
//非法选中判定
QString request = model.item(r, 6)->text();
if (request != QString("申请借阅"))
{
    (!MsgBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 已批准借阅! < / div>",
        QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
    return;
}
int ret= MsgBox(this,
    "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 确认批准借阅? < / div>",
    QIcon(":/x64/Debug/resource/pic/default.png")).exec();
if (ret==QMessageBox::Yes)
{
    SqlMgr::getInstance()->verify_borrow(model.item(r, 0)->text(),
        model.item(r, 1)->text(), model.item(r, 2)->text());
    (!MsgBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 批准成功! < / div>",
        QIcon(":/x64/Debug/resource/pic/default.png"))).exec();
}
init();
}

void Cell_RcdMgr::on_btn_ref_clicked()
{
    init();
}
```

(8) cell_self.cpp

```

#include "cell_self.h"
#include "sqlmgr.h"
#include "msgbox.h"
#include <qdebug.h>

Cell_Self::Cell_Self(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Cell_SelfClass())
{
    ui->setupUi(this);
    ui->tableView->setModel(&model);
    ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
    ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);

    //将提醒修改初始密码的信号和对应的槽函数连接起来
    connect(this, &Cell_Self::show_pop, this, &Cell_Self::rmd_rstpwd);
}

Cell_Self::~Cell_Self()
{
    delete ui;
}

void Cell_Self::getid(QString self_name)
{
    QString str_sql = QString("where username='%1' ")
        .arg(self_name);

    //通过当前的用户名查找到用户信息
    auto info = SqlMgr::getInstance()->get_users(str_sql)[0];
    self_id = info[0];
    name = info[1];
    ui->le_name->setText(info[1]);
    ui->le_pwd->setText(info[2]);
    ui->le_pwd->setEchoMode(LineEdit::Password);
    ui->le_name->setReadOnly(true);
    ui->le_pwd->setReadOnly(true);

    //若当前用户密码为初始密码，发送提示修改密码的信号
    if (info[2] == QString("123456"))
        emit show_pop();

    qDebug() << "Self: " << self_id;
    qDebug() << "Name: " << info[1];
}

void Cell_Self::init(QString str_cond)
{
    ui->le_pwd->setReadOnly(true);
    ui->le_name->setReadOnly(true);
    if (QString("") == self_id) return;

    //若用户修改了账号密码，显示修改后的账号密码
    QString str_sql = QString("where userid=%1 ")
        .arg(self_id);
    auto info = SqlMgr::getInstance()->get_users(str_sql)[0];
    ui->le_name->setText(info[1]);
    ui->le_pwd->setText(info[2]);
}

```

```

QString user = QString("where userid= %1 ").arg(self_id);
auto l = SqlMgr::getInstance()->get_records(user + str_cond);
model.clear();
model.setHorizontalHeaderLabels(QStringList{ "id", "用户id", "图书id", "图书
名称", "借阅时间", "应还时间", "借阅状态" });
ui->tableView->horizontalHeader()->setStretchLastSection(true);
ui->tableView->horizontalHeader()->
>setSectionResizeMode(QHeaderView::Stretch);
for (int i = 0; i < l.size(); i++)
{
    QList<QStandardItem*> items;
    for (int j = 0; j < l[i].size(); j++)
        items.append(new QStandardItem(l[i][j]));

    model.appendRow(items);
}
}

//由于代码复用性较好，搜索功能和申请功能与之前页面中的对应功能的编写相近，不再注释
void Cell_Self::on_comboBox_currentIndexChanged()
{
    int idx = ui->comboBox->currentIndex();
    QString str_cond="";
    switch (idx)
    {
    case 1:
    {
        str_cond = QString("and request='申请借阅'");
        break;
    }
    case 2:
    {
        str_cond = QString("and request='已借未还'");
        break;
    }
    case 3:
    {
        str_cond = QString("and request='申请归还'");
        break;
    }
    case 4:
    {
        str_cond = QString("and request='已归还'");
        break;
    }
    default:
        break;
    }
    init(str_cond);
}

void Cell_Self::on_btn_req_clicked()
{
    int r = ui->tableView->currentIndex().row();
    if (r < 0)
    {
        (!MsgBox(this,
                  "< div style = 'text-align: center; font-weight: bold; font-
size: 20px;' > 没有选中记录! < / div>",
                  QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
        return;
    }

    QString access = model.item(r, 6)->text();

```

```

if (access !=QString("已借未还"))
{
    (!MsgBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 当前不能申请归还! < / div>",
        QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
    return;
}
int ret = MsgBox(this,
    "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 确认申请? < / div>",
    QIcon(":/x64/Debug/resource/pic/question.png")).exec();

if (ret == QMessageBox::Yes)
{
    SqlMgr::getInstance()->return_books(model.item(r, 0)->text());
    (!MsgBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 申请成功! < / div>",
        QIcon(":/x64/Debug/resource/pic/default.png"))).exec();
}
init();
}

void Cell_Self::on_btn_showpwd_clicked()
{
    //显示/隐藏密码按钮功能实现
    if (ui->btn_showpwd->text() == QString("显示密码"))
    {
        ui->btn_showpwd->setText("隐藏密码");
        ui->le_pwd->setEchoMode(QLineEdit::Normal);
    }
    else
    {
        ui->btn_showpwd->setText("显示密码");
        ui->le_pwd->setEchoMode(QLineEdit::Password);
    }
}

void Cell_Self::on_btn_set_clicked()
{
    //修改信息按钮功能实现
    ui->le_pwd->setReadOnly(false);
    ui->le_name->setReadOnly(false);
}

void Cell_Self::on_btn_save_clicked()
{
    //保存修改按钮功能实现
    int ret = MsgBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 确认保存修改? < / div>",
        QIcon(":/x64/Debug/resource/pic/question.png")).exec();

    if (ret == QMessageBox::Yes)
    {
        QString duplicate = QString("where username = '%1'").arg(ui->le_name->text());
        if (!SqlMgr::getInstance()->get_users(duplicate).size() || name == ui->le_name->text()) {
            SqlMgr::getInstance()->set_userinfo(self_id, ui->le_name->text(), ui->le_pwd->text());
            (!MsgBox(nullptr,
                "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 修改成功! < / div>",
                QIcon(":/x64/Debug/resource/pic/default.png"))).exec();
        }
    }
}

```

```
        else
    {
        (!MsgBox(this,
            "< div style = 'text-align: center; font-weight: bold;
font-size: 20px;' > 用户名已被使用, 请更换! < / div>",
            QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
    }
    else { return; }
    init();
}

void Cell_Self::rmd_rstpwd()
{
    //提示修改初始密码对话框
    (!MsgBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-size:
20px;' > 您的密码仍为初始密码, 请更换! < / div>",
        QIcon(":/x64/Debug/resource/pic/default.png"))).exec();
}

void Cell_Self::on_btn_ref_clicked()
{
    init();
}
```

(9) cell_usermgr.cpp

```

#include "cell_usermgr.h"
#include "sqlmgr.h"
#include "msgbox.h"
#include "add_singleuser.h"
#include "qfiledialog.h"
#include <qregularexpression.h>
#include <QDebug>

Cell_UserMgr::Cell_UserMgr(QWidget* parent)
    : QWidget(parent)
    , ui(new Ui::Cell_UserMgrClass()){
    ui->setupUi(this);
    ui->tableView->setModel(&model);
    ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
    ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);
}

Cell_UserMgr::~Cell_UserMgr()
{
    delete ui;
}

void Cell_UserMgr::init(QString str_cond)
{
    auto l = SqlMgr::getInstance()->get_users(str_cond);
    model.clear();
    model.setHorizontalHeaderLabels(QStringList{ "id", "用户名", "密码", "借阅总数", "权限" });
    ui->tableView->horizontalHeader()->setStretchLastSection(true);
    ui->tableView->horizontalHeader()-
    >setSectionResizeMode(QHeaderView::Stretch);
    for (int i = 0; i < l.size(); i++)
    {
        QList<QStandardItem*> items;
        for (int j = 0; j < l[i].size(); j++)
        {
            j==2&&l[i][j]!="123456"?items.append(new
QStandardItem(QString("*****"))):
            items.append(new QStandardItem(l[i][j]));
        }
        model.appendRow(items);
    }
}

void Cell_UserMgr::on_le_qryTextChanged()
{
    int idx = ui->comboBox->currentIndex();
    QString str_cond;
    switch (idx)
    {
    case 0:
    {
        str_cond = QString("where userid=%1")
            .arg(ui->le_qry->text());
        break;
    }
    case 1:
    {
        str_cond = QString("where username like '%%1%'")
            .arg(ui->le_qry->text());
        break;
    }
}

```

```

    case 2:
    {
        QString user_or_admin =
            ui->le_qry->text() == "" ? "" : "!";
        str_cond = QString("where admin %1=''");
        .arg(user_or_admin);
        break;
    }
    default:
        return; break;
    }
    init(str_cond);
}

void Cell_UserMgr::on_btn_imp_clicked()
{
    QMessageBox msgbox = QMessageBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 添加方式? < / div>",
        QIcon(":/x64/Debug/resource/pic/question.png"));

    msgbox.set_BTN_text(QMessageBox::Yes, "单个添加");
    msgbox.set_BTN_text(QMessageBox::No, "批量导入");
    int ret = msgbox.exec();

    if (ret == QMessageBox::Yes)
    {
        qDebug() << QString("单个添加");
        Add_SingleUser add_singleuser;
        add_singleuser.exec();
    }
    else if (ret == QMessageBox::No)
    {
        //批量导入
        auto str_path = QFileDialog::getOpenFileName(nullptr, "输入文件路径");

        if (!str_path.isEmpty())
        {
            QFile f(str_path);
            f.open(QFile::ReadOnly);
            QVector<QStringList> vec_data;
            QRegularExpression re("[, , ; , \\\s]+");

            while (!f.atEnd())
            {
                //清除无关字符的语句
                QString str = f.readLine().trimmed();
                auto l = str.split(re, Qt::SkipEmptyParts);

                while (l.size() < 4) {
                    l.append("");
                }
                if (l.size() > 4) {
                    l = l.mid(0, 4);
                }

                vec_data.push_back(l);
            }
            qDebug() << "\n\n";
            qDebug() << vec_data;

            SqlMgr::getInstance()->add_user(vec_data);
        }
    }
    else { return; }
    init();
}

```

```

void Cell_UserMgr::on_btn_set_clicked()
{
    int r = ui->tableView->currentIndex().row();
    if (r < 0)
    {
        (!MsgBox(this,
                  "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 没有选中用户! < / div>",
                  QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
        return;
    }

    QMessageBox msgbox = QMessageBox(this,
                                    "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 修改操作? < / div>",
                                    QIcon(":/x64/Debug/resource/pic/question.png"));

    msgbox.setBtnText(QMessageBox::Yes, "激活权限");
    msgbox.setBtnText(QMessageBox::No, "重置密码");

    int ret_choice = msgbox.exec();

    //bhv=0, 操作为重置密码, bhv=1, 操作为激活权限
    int bhv = 0;
    if (ret_choice == QMessageBox::Yes)
    {
        if (model.item(r, 4)->text() == "管理员")
        {
            (!MsgBox(this,
                      "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 权限已激活! < / div>",
                      QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
            return;
        }
        int ret_act = QMessageBox(this,
                                  "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 确认激活? < / div>",
                                  QIcon(":/x64/Debug/resource/pic/question.png")).exec();
        if (ret_act == QMessageBox::Yes)
        {
            qDebug() << "激活请求发出";
            auto id = model.item(r, 0)->text();
            qDebug() << id;
            SqlMgr::getInstance()->update_users(id,++bhv);
            init();
        }
    }
    else
    {
        if (model.item(r, 2)->text() == "123456")
        {
            (!MsgBox(this,
                      "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 密码已重置! < / div>",
                      QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
            return;
        }

        int ret_rst = QMessageBox(this,
                                  "< div style = 'text-align: center; font-weight: bold; font-size: 20px;' > 确认重置? < / div>",
                                  QIcon(":/x64/Debug/resource/pic/question.png")).exec();
        if (ret_rst == QMessageBox::Yes)
        {
            auto id = model.item(r, 0)->text();
            SqlMgr::getInstance()->update_users(id,bhv);
        }
    }
}

```

```

        init();
    }
}

void Cell_UserMgr::on_btn_dlt_clicked()
{
    int r = ui->tableView->currentIndex().row();
    if (r < 0)
    {
        (!MsgBox(this,
                  "< div style = 'text-align: center; font-weight: bold; font-
size: 20px;' > 没有选中用户! < / div>",
                  QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
        return;
    }

    int ret = MsgBox(this,
                      "< div style = 'text-align: center; font-weight: bold; font-size:
20px;' > 确认删除? < / div>",
                      QIcon(":/x64/Debug/resource/pic/question.png")).exec();

    if (ret == QMessageBox::Yes)
    {
        QString id = model.item(r, 0)->text();
        QString str_sql = QString("where userid = %1 and (request = '已借未
还' or request = '申请归还')").arg(id);
        if (SqlMgr::getInstance()->get_records(str_sql).size())
        {
            (!MsgBox(this,
                      "< div style = 'text-align: center; font-weight: bold;
font-size: 20px;' > 删除失败! 有未还图书! < / div>",
                      QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
            return;
        }
        SqlMgr::getInstance()->dlt_user(id);
        init();
    }
}

void Cell_UserMgr::on_btn_ref_clicked()
{
    init();
}

```

(10) dlg_login.cpp

```

#include "dlg_login.h"
#include "msgbox.h"
#include "sqlmgr.h"
#include <QCloseEvent>

Dlg_Login::Dlg_Login(QWidget *parent)
    : QDialog(parent)
    , ui(new Ui::Dlg_LoginClass())
{
    ui->setupUi(this);
    this->setWindowIcon(QIcon(":/x64/Debug/resource/pic/login.png"));
    //设置窗口图标
}

Dlg_Login::~Dlg_Login()
{
    delete ui;
}

void Dlg_Login::on_btn_adminlgn_clicked()
{
    //管理员登录按钮功能实现，若登录成功，发送对应信号
    if (SqlMgr::getInstance()->login(ui->le_name->text(), ui->le_pwd-
>text(),1))
    {
        emit admin_lgn();
        hide();
        ui->le_name->setText("");
        ui->le_pwd->setText("");
    }
}

void Dlg_Login::on_btn_userlgn_clicked()
{
    //用户登录按钮功能实现，若登录成功，发送对应信号
    if (SqlMgr::getInstance()->login(ui->le_name->text(), ui->le_pwd-
>text(),0))
    {
        emit log_success(ui->le_name->text());
        emit user_lgn();
        hide();
        ui->le_name->setText("");
        ui->le_pwd->setText("");
    }
}

void Dlg_Login::closeEvent(QCloseEvent* event)
{
    //鼠标点击关闭事件
    int ret = MsgBox(nullptr,
        "< div style = 'text-align: center; font-weight: bold; font-size:
20px;' > 是否退出？ < / div>",
        QIcon(":/x64/Debug/resource/pic/exit_b.png")).exec();

    if (ret == QMessageBox::Yes) {
        event->accept(); //接受关闭事件，关闭窗口
    }
    else {
        event->ignore(); //忽略关闭事件，保持窗口不变
    }
}

```

(11) msgbox.cpp

```
#include "msgbox.h"
#include <QAbstractButton>

MsgBox::MsgBox(QWidget* parent, const QString& text, QIcon icon,
    StandardButtons btns, const QString& title, StandardButton dftbtn):
    QMessageBox(parent)
{
    setWindowTitle(title);
    setText(text);
    setStandardButtons(btns);
    setDefaultButton(dftbtn);
    setWindowIcon(icon);
    //将文本设置为接受html语言
    setTextFormat(Qt::RichText);
}

void MsgBox::set_btn_text(StandardButton btn, const QString& btn_txt)
{
    //修改按钮文本
    QAbstractButton* diy_btn = button(btn);
    if (diy_btn)
        diy_btn->setText(btn_txt);
}

MsgBox& MsgBox::operator!()
{
    //'!'运算符重载: 隐藏按钮, 只显示信息
    QList<QAbstractButton*> buttons = this->buttons();
    for (QAbstractButton* button : buttons)
    {
        button->hide();
    }
    return *this;
}

void MsgBox::closeEvent(QCloseEvent* event)
{
    //将鼠标点击关闭事件和点击“No/Cancel”按钮事件区分开
    QDialog::done(Rejected - 1);
}
```

(12) page_admin.cpp

```

#include "page_admin.h"
#include "msgbox.h"

Page_Admin::Page_Admin(QWidget *parent)
    : Base(parent)
    , ui(new Ui::Page_AdminClass())
    , analy(nullptr)
    , bkmgr(nullptr)
    , rcdmgr(nullptr)
    , usermgr(nullptr)
//初始化各成员指针
{
    ui->setupUi(this);
    this->setWindowIcon(QIcon(":/x64/Debug/resource/pic/admin.png"));
    ui->btn_bkmgr->setIcon(QIcon(":/x64/Debug/resource/pic/book_mg.png"));
    ui->btn_usermgr->setIcon(QIcon(":/x64/Debug/resource/pic/user_mg.png"));
    ui->btn_analy->setIcon(QIcon(":/x64/Debug/resource/pic/analysis.png"));
    ui->btn_rcdmgr->setIcon(QIcon(":/x64/Debug/resource/pic/record_mg.png"));
    ui->btn_exit->setIcon(QIcon(":/x64/Debug/resource/pic/exit_w.png"));
    ui->btn_switch->setIcon(QIcon(":/x64/Debug/resource/pic/switch_w.png"));
//设置主界面各按钮图标
    initpage();
}

Page_Admin::~Page_Admin()
{
    delete ui;
}

void Page_Admin::pop_message()
{
    //管理员页面提示信息
    (!MsgBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-size: 10px;' > 尊敬的管理员, 请您尽快批准未批准记录! < / div>",
        QIcon(":/x64/Debug/resource/pic/default.png"))).exec();
}

void Page_Admin::initpage()
{
    analy = new Cell_Analysis(this);
    bkmgr = new Cell_BkMgr(this);
    rcdmgr = new Cell_RcdMgr(this);
    usermgr = new Cell_UserMgr(this);

    //实现堆栈窗口逻辑
    ui->stackedWidget->addWidget(bkmgr);
    ui->stackedWidget->addWidget(usermgr);
    ui->stackedWidget->addWidget(rcdmgr);
    ui->stackedWidget->addWidget(analy);

    ui->stackedWidget->setCurrentIndex(0);

    auto l = ui->menu->children();
    for (auto it : l)
    {
        if (it->objectName().contains("btn"))
        {
            //将点击导航栏按钮信号和显示对应页面函数连接起来
            connect(static_cast<QToolButton*>(it), &QToolButton::clicked,
this, &Page_Admin::dealmenu);
        }
    }
}

```

```
bkmgr->init();
usermgr->init();
rcdmgr->init();
analy->init();
}

void Page_Admin::dealmenu()
{
    //实现导航栏逻辑
    auto str = sender()->objectName();
    do
    {
        if ("btn_bkmgr"==str)
        {
            ui->stackedWidget->setcurrentIndex(0);
            if (!msg_cnt++) pop_message();
            break;
        }
        if ("btn_usermgr" == str)
        {
            ui->stackedWidget->setcurrentIndex(1);
            if (!msg_cnt++) pop_message();
            break;
        }
        if ("btn_rcdmgr" == str)
        {
            ui->stackedWidget->setcurrentIndex(2);
            if (!msg_cnt++) pop_message();
            break;
        }
        if ("btn_analy" == str)
        {
            ui->stackedWidget->setcurrentIndex(3);
            if (!msg_cnt++) pop_message();
            break;
        }
    } while (false);
}
```

(13) page_user.cpp

```

#include "Page_user.h"
#include "msgbox.h"
#include <QDebug>

Page_user::Page_user(QWidget *parent)
    : Base(parent)
    , ui(new Ui::Page_userClass())
    , bkbrw(new Cell_BkBrw(this))
    , self(new Cell_Self(this))
    //初始化各成员指针
{
    ui->setupUi(this);
    setWindowIcon(QIcon(":/x64/Debug/resource/pic/books.png"));
    ui->btn_info->setIcon(QIcon(":/x64/Debug/resource/pic/info.png"));
    ui->btn_qry->setIcon(QIcon(":/x64/Debug/resource/pic/qry.png"));
    ui->btn_exit->setIcon(QIcon(":/x64/Debug/resource/pic/exit_w.png"));
    ui->btn_switch->setIcon(QIcon(":/x64/Debug/resource/pic/switch_w.png"));
    //设置主界面各按钮图标

    initpage();
}

Page_user::~Page_user()
{
    delete ui;
}

void Page_user::pop_message()
{
    //用户页面提示信息
    (!MsgBox(this,
        "< div style = 'text-align: center; font-weight: bold; font-size: 10px;' > 尊敬的用户，如有将逾期图书请尽快归还！ < / div>",
        QIcon(":/x64/Debug/resource/pic/default.png"))).exec();
}

void Page_user::initpage()
{
    //实现堆栈窗口逻辑
    ui->stackedWidget->addWidget(self);
    ui->stackedWidget->addWidget(bkbrw);

    ui->stackedWidget->setCurrentIndex(0);

    auto l = ui->menu->children();
    for (auto it : l)
    {
        if (it->objectName().contains("btn"))
        {
            //将点击导航栏按钮信号和显示对应页面函数连接起来
            connect(static_cast<QToolButton*>(it), &QToolButton::clicked,
this, &Page_user::dealmenu);
        }
    }
    self->init();
    bkbrw->init();
}

```

```
void Page_user::dealmenu()
{
    //实现导航栏逻辑
    auto str = sender()->objectName();
    do
    {
        if ("btn_info" == str)
        {
            ui->stackedWidget->setcurrentIndex(0);
            break;
        }
        if ("btn_qry" == str)
        {
            ui->stackedWidget->setcurrentIndex(1);
            break;
        }
    } while (false);
}

void Page_user::set_username(const QString& username)
{
    //用户登录成功后将登录对话中输入的用户名传递到用户信息和图书借阅的页面中
    bkbrw->getid(username);
    self->getid(username);

    pop_message();
}
```

(14) sqlmgr.cpp

```

#include "sqlmgr.h"
#include <msgbox.h>
#include <QSqlError>
#include <QSqlQuery>
#include <QSqlRecord>
#include <QCoreApplication>
#include <QDebug>
#include <QDateTime>

//初始化静态成员
SqlMgr* SqlMgr::instance = nullptr;

SqlMgr::SqlMgr()
{
}

SqlMgr::~SqlMgr()
{
}

SqlMgr* SqlMgr::getInstance()
{
    //在单例模式中获取全局唯一对象
    if (nullptr == instance) {
        instance = new SqlMgr();
    }
    return instance;
}

void SqlMgr::init()
{
    //导入程序内置数据库
    QString absolutePath = QCoreApplication::applicationDirPath()+
    "/resource/db/bk_mgr.db";
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(absolutePath);

    if (!db.open()) {
        (!MsgBox(nullptr,
            "< div style = 'text-align: center; font-weight: bold; font-
size: 20px;' > 数据库导入失败! < / div>",
            QIcon(":/x64/Debug/resource/pic/error.png"))).exec();
    }
}

bool SqlMgr::login(QString str_name, QString str_pwd, int admin)
{
    //实现登录逻辑：在数据库中查找用户名、密码以及权限与登录对话的对应输入相符的用户
    //数据，如果有匹配项则登录成功，否则失败
    QSqlQuery q(db);
    QString str_sql;
    if (admin) {
        //管理员登录
        str_sql = QString("select * from user where username='%1' and
password='%2' and admin!=''");
        .arg(str_name).arg(str_pwd);
    }
}

```

```

    else{
        //用户登录
        str_sql = QString("select * from user where username='%1' and
password='%2'");
        .arg(str_name).arg(str_pwd);
    }
    if (q.exec(str_sql))
    {
        //登录成功
        while (q.next()) {
            return true;
        }
    }
    else
    {
        qDebug() << q.lastError().text();
    }

    //登录失败提示
    (!MsgBox(nullptr,
    "< div style = 'text-align: center; font-weight: bold; font-size:
20px;' > 用户名/密码错误! < / div>",
    QIcon(":/x64/Debug/resource/pic/error.png"))).exec();

    return false;
}

QVector<QStringList> SqlMgr::get_users(QString str_cond)
{
    //实现获取用户：在数据库中使用传入的搜索语句搜索满足对应条件的用户数据
    QSqlQuery q(db);
    QString str_sql;

    str_sql = QString("select * from user %1").arg(str_cond);
    qDebug() << str_sql;

    QVector<QStringList> vec;
    if (q.exec(str_sql))
    {
        int cols = q.record().count();
        QStringList l;
        while (q.next())
        {
            l.clear();
            for (int i = 0; i < cols; i++)
            {
                l << q.value(i).toString();
            }
            vec.push_back(l);
        }
    }
    else { qDebug() << q.lastError().text(); }
    return vec;
}

void SqlMgr::add_user(QString str_name,QString& str_admin)
{
    //实现添加用户：由输入的文本插入一条新的用户数据
    QSqlQuery q(db);
    QString str_sql = QString("insert into user
values(null,'%1','123456',0,'%2')")
    .arg(str_name).arg(str_admin);
    if (!q.exec(str_sql))
    {
        qDebug() << q.lastError().text();
    }
}

```

```

void SqlMgr::add_user(QVector<QStringList> v)
{
    //实现批量导入：根据输入的文件路径进行批量插入
    db.transaction();
    QSqlQuery q(db);
    for (auto it : v)
    {
        QString str_sql = QString("insert into user
VALUES(NULL, '%1', '%2', '%3', '%4');");
        .arg(it[0])
        .arg(it[1])
        .arg(it[2])
        .arg(it[3]);
        bool ret = q.exec(str_sql);
        if (!ret)
            qDebug() << q.lastError().text();
    }

    db.commit();
}

void SqlMgr::update_users(QString str_id,int bhv)
{
    //实现更新用户：根据选择的具体更新操作，激活当前选中用户的管理员权限或者将密码重置为初始密码
    QSqlQuery q(db);
    QString str_sql;
    qDebug() << "激活操作";
    if (bhv)
    {
        str_sql = QString("update user set admin='管理员' where userid=%1")
            .arg(str_id);
    }
    else
    {
        str_sql = QString("update user set password='123456' where
userid=%1")
            .arg(str_id);
    }

    if (!q.exec(str_sql))
    {
        qDebug() << q.lastError().text();
    }
}

void SqlMgr::set userinfo(QString id, QString username, QString password)
{
    //实现修改账号密码
    QSqlQuery q(db);
    QString str_sql = QString("update user set username='%1', password='%2'
where userid=%3")
        .arg(username)
        .arg(password)
        .arg(id);
    bool ret = q.exec(str_sql);
    if (!ret)
    {
        qDebug() << q.lastError().text();
    }
}

```

```

bool SqlMgr::dlt_user(QString str_id)
{
    //实现删除用户
    QSqlQuery q(db);
    QString str_sql = QString("delete from user where userid=%1")
        .arg(str_id);
    bool ret = q.exec(str_sql);
    if (!ret)
    {
        qDebug() << q.lastError().text();
    }
    return ret;
}

//图书管理部分
//QVector<QStringList> SqlMgr::get_books(QString str_cond)
{
    //实现获取图书
    QSqlQuery q(db);
    QString str_sql;

    str_sql = QString("select * from book %1").arg(str_cond);

    QVector<QStringList> vec;
    if (q.exec(str_sql))
    {
        int cols = q.record().count();
        QStringList l;
        while (q.next())
        {
            l.clear();
            for (int i = 0; i < cols; i++)
            {
                l << q.value(i).toString();
            }
            vec.push_back(l);
        }
    }
    else { qDebug() << q.lastError().text(); }
    return vec;
}

void SqlMgr::add_books(QVector<QStringList> v)
{
    //实现批量导入图书
    db.transaction();
    QSqlQuery q(db);
    for (auto it : v)
    {
        QString str_sql = QString("insert into book
VALUES(NULL, '%1', '%2', '%3', '%4', '%5', '%6');");
        .arg(it[0])
        .arg(it[1])
        .arg(it[2])
        .arg(it[3])
        .arg(it[4])
        .arg(it[5]);
        bool ret = q.exec(str_sql);
        if (!ret)
            qDebug() << q.lastError().text();
    }
    db.commit();
}

```

```

void SqlMgr::update_books(QStringList l,QString id)
{
    //实现更新图书信息
    QSqlQuery q(db);
    QString str_sql = QString("update book "
                               "set
bookname='%1',author='%2',type1='%3',type2='%4',access=%5 "
                               "where bookid=%7 ;")
    .arg(l[0])
    .arg(l[1])
    .arg(l[2])
    .arg(l[3])
    .arg(l[4])
    .arg(id);

    bool ret = q.exec(str_sql);
    if (!ret)
        qDebug() << q.lastError().text();

}

bool SqlMgr::dlt_books(QString str_id)
{
    //实现删除图书
    QSqlQuery q(db);
    QString str_sql = QString("delete from book where bookid=%1")
    .arg(str_id);
    bool ret= q.exec(str_sql);
    if (!ret)
    {
        qDebug() << q.lastError().text();
        //最终可改为报错弹窗
        return false;
    }
    (!MsgBox(nullptr,
              "< div style = 'text-align: center; font-weight: bold; font-size:
20px;' > 删除成功! < / div>",
              QIcon(":/x64/Debug/resource/pic/default.png"))).exec();
    return ret;
}

void SqlMgr::return_books(QString recordid)
{
    //实现申请归还操作：将当前选中的记录的借阅状态改为“申请归还”
    QSqlQuery q(db);
    QString str_sql = QString("UPDATE record SET request='申请归还' WHERE
recordid=%1 ")
    .arg(recordid);
    bool ret = q.exec(str_sql);
    if (!ret)
    {
        qDebug() << q.lastError().text();
    }
}

void SqlMgr::borrow_books(QString userid, QString bookid,QString bookname)
{
    //实现申请借阅：新添一条当前用户申请借阅当前选中图书的借阅记录，借阅状态为“申请借
    //阅”
    QSqlQuery q(db);
    QString str_sql = QString("insert into record
VALUES(NULL,%1,%2,'%3','','','%4')")
    .arg(userid)
    .arg(bookid)
    .arg(bookname)
    .arg("申请借阅");
}

```

```

        bool ret = q.exec(str_sql);
        if (!ret)
        {
            qDebug() << q.lastError().text();
        }
    }

void SqlMgr::verify_borrow(QString recordid, QString userid, QString bookid)
{
    //实现批准借阅: 将当前选中的记录的借阅状态改为“已借未还”
    QSqlQuery q(db);
    QString str[3];
    QDateTime start = QDateTime::currentDateTime();
    QDateTime end =
    QDateTime::fromSecsSinceEpoch(QDateTime::currentSecsSinceEpoch() + 3600 * 24
    * 90);

    str[0] = QString("UPDATE record SET start='%1', end='%2' ,request='%3'
WHERE recordid =%4 ")
        .arg(start.toString("yyyy-MM-dd"))
        .arg(end.toString("yyyy-MM-dd,23:59"))
        .arg("已借未还")
        .arg(recordid);

    //用户借阅总数加1, 图书被借总数加1, 图书在架数量减1
    str[1] = QString("UPDATE user SET brwcnt=brwcnt+1 WHERE userid =%1
").arg(userid);
    str[2] = QString("UPDATE book SET cnt=cnt+1, access=access-1 WHERE
bookid =%1").arg(bookid);

    for (int i = 0; i < 3; i++)
    {
        if (!q.exec(str[i]))
        {
            qDebug() << q.lastError().text();
        }
    }
}

void SqlMgr::verify_return(QString recordid, QString bookid)
{
    //实现批准归还: 将当前选中的记录的借阅状态改为“已归还”
    QSqlQuery q(db);
    QString str[2];

    str[0] = QString("UPDATE record SET request='%1' WHERE recordid =%2 ")
        .arg("已归还")
        .arg(recordid);

    //图书在架数量加1
    str[1] = QString("UPDATE book SET access=access+1 WHERE bookid
=%1").arg(bookid);
    for (int i = 0; i < 2; i++)
    {
        if (!q.exec(str[i]))
        {
            qDebug() << q.lastError().text();
        }
    }
}

```

```
//  
//借阅记录部分  
//  
  
QVector<QStringList> SqlMgr::get_records(QString str_cond)  
{  
    //获取借阅记录  
    QSqlQuery q(db);  
    QString str_sql;  
  
    str_sql = QString("select * from record %1").arg(str_cond);  
    qDebug() << str_sql;  
  
    QVector<QStringList> vec;  
    if (q.exec(str_sql))  
    {  
        int cols = q.record().count();  
        QStringList l;  
        while (q.next())  
        {  
            l.clear();  
            for (int i = 0; i < cols; i++)  
            {  
                l << q.value(i).toString();  
            }  
            vec.push_back(l);  
        }  
    }  
    else { qDebug() << q.lastError().text(); }  
    return vec;  
}
```

(15) main.cpp

```
#include "dlg_login.h"
#include "Page_user.h"
#include "page_admin.h"
#include "sqlmgr.h"
#include <QtWidgets/QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    //创建并初始化数据库类的唯一对象
    SqlMgr::getInstance()->init();

    Dlg_Login dlg;
    Page_user user_page;
    Page_Admin admin_page;

    //用户登陆成功之后传输用户名，调用初始化页面函数
    QObject::connect(&dlg, &Dlg_Login::log_success, &user_page,
    &Page_user::set_username);
    QObject::connect(&dlg, &Dlg_Login::log_success, &user_page,
    &Page_user::initpage);

    //连接登陆成功信号和显示对应主界面槽函数
    QObject::connect(&dlg, &Dlg_Login::user_lgn, [&]() {
        user_page.show();
    });

    QObject::connect(&dlg, &Dlg_Login::admin_lgn, [&]() {
        admin_page.show();
    });

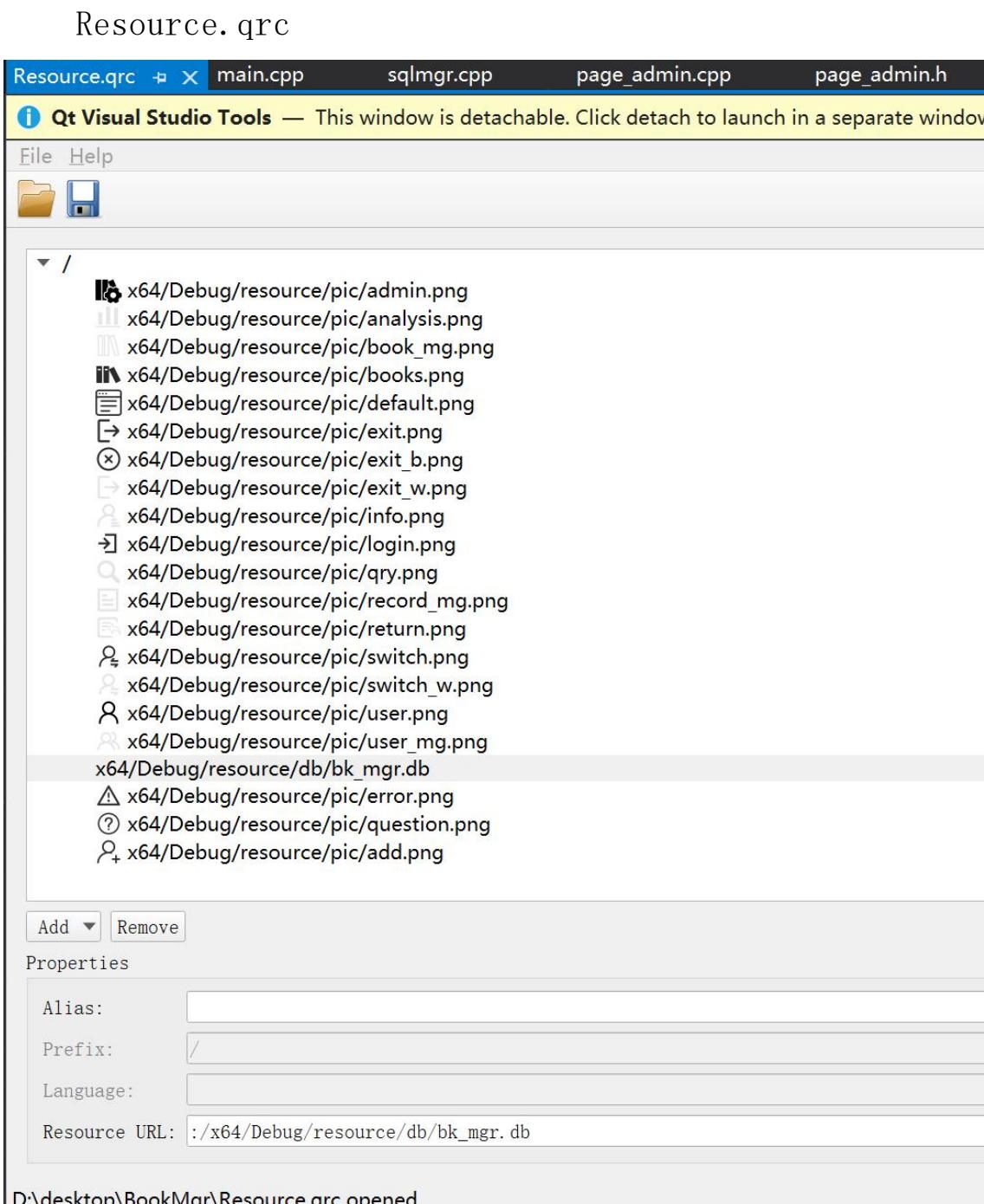
    //连接切换账号信号和返回登录对话槽函数
    QObject::connect(&user_page, &Page_user::switch_acnt, [&]() {
        dlg.show();
    });

    QObject::connect(&admin_page, &Page_Admin::switch_acnt, [&]() {
        dlg.show();
    });

    dlg.show();

    return a.exec();
}
```

4. 资源文件



三、评分表

项 目	评 价	
设计方案的合理性与创新性	3	
设计与调试结果	4	
设计说明书的质量	1	
程序基本要求涵盖情况	4	
程序代码编写素养情况	2	
课程设计周表现情况	1	
综合成绩	15	