|  |  |
| --- | --- |
| 学号：1191002001  **程序设计综合实践报告**  **——基于QT的简单图书借阅管理信息系统** | 姓名：白仕博 |
| 学号：1191002002 | 姓名：陈奥运 |
| 学号：1191002050 | 姓名：邹星宇 |
| 指导老师：徐敏 | |
| 专业：计算机科学与技术 | |
| 学院：信息工程学院 | |
| 学校：首都师范大学 | |
| 完成日期：2021年10月6日星期三 | |

# 目录

[目录 2](#_Toc16972)

[一、 系统需求分析 5](#_Toc7287)

[（一） 管理员需求 5](#_Toc15920)

[1. 用户管理 6](#_Toc14349)

[2. 图书管理 7](#_Toc7759)

[3. 统计 8](#_Toc23852)

[4. 设置 8](#_Toc17434)

[（二） 用户需求 9](#_Toc31599)

[1. 新书导读 9](#_Toc19716)

[2. 借书 9](#_Toc32181)

[3. 还书 10](#_Toc5826)

[4. 查阅借书记录 10](#_Toc14317)

[5. 设置 10](#_Toc1640)

[二、 总体设计 11](#_Toc24624)

[（一） 类的静态设计 11](#_Toc21943)

[（二） 类的动态设计 12](#_Toc16955)

[（三） 功能事务的实现过程 13](#_Toc19564)

[三、 系统详细设计 17](#_Toc28688)

[（一） 类的详细设计 17](#_Toc27181)

[（二） 主要功能函数的实现 19](#_Toc2470)

[（三） 程序主函数的算法流程 34](#_Toc2810)

[四、 系统测试 35](#_Toc3123)

[（一） 操作界面测试 35](#_Toc19354)

[（二） 功能测试 39](#_Toc3105)

[（三） CPU占用测试 52](#_Toc13959)

图目录

[图 1 管理员用例图 5](#_Toc12945)

[图 2 用户管理功能图 6](#_Toc2138)

[图 3 图书管理功能图 7](#_Toc10736)

[图 4 统计功能图 8](#_Toc19568)

[图 5 设置功能图 8](#_Toc22399)

[图 6 用户用例图 9](#_Toc26294)

[图 7 用户借书功能图 9](#_Toc16215)

[图 8 用户设置功能图 10](#_Toc9402)

[图 9 组织数据和操作数据的类图 11](#_Toc2324)

[图 10 装载人员信息文件顺序图 12](#_Toc17796)

[图 11 主要功能顺序图 12](#_Toc7058)

[图 12 简单图书借阅信息管理系统的总活动图 13](#_Toc4983)

[图 13 装载数据文件功能活动图 14](#_Toc9716)

[图 14 添加用户信息记录功能活动图 14](#_Toc21881)

[图 15 删除用户信息记录活动图 15](#_Toc14827)

[图 16 修改指定人员信息记录功能活动图 15](#_Toc23380)

[图 17 修改密码活动图 15](#_Toc23349)

[图 18 切换账号功能活动图 16](#_Toc10587)

[图 19 退出软件运行功能活动图 16](#_Toc10339)

[图 20 欢迎界面（左登录，右注册） 35](#_Toc1872)

[图 21 管理员界面——用户界面 36](#_Toc224)

[图 22 管理员界面——图书界面 36](#_Toc30341)

[图 23 管理员界面——统计界面 37](#_Toc19503)

[图 24 用户界面——借书界面 37](#_Toc21314)

[图 25 用户界面——新书导读 38](#_Toc18036)

[图 26 用户界面——预约书籍列表 38](#_Toc9575)

[图 27 用户界面——当前借书列表 38](#_Toc30262)

[图 28 用户界面——全部借书记录 39](#_Toc25682)

[图 29 注册-未填写完整信息 39](#_Toc12131)

[图 30 注册-用户已存在 40](#_Toc12803)

[图 31 注册-注册成功 40](#_Toc18324)

[图 32 登录-登录失败 41](#_Toc20862)

[图 33 登录-账号或密码错误 41](#_Toc24467)

[图 34 登录成功 42](#_Toc23665)

[图 35 借书-无可借库存 42](#_Toc26139)

[图 36 借书-借书成功 43](#_Toc31767)

[图 37 借书成功状态更改 43](#_Toc14302)

[图 38 预约成功 44](#_Toc10588)

[图 39 预约成功状态更改 44](#_Toc204)

[图 40 预约-有库存直接借阅 44](#_Toc14402)

[图 41 取消预约 45](#_Toc17043)

[图 42 还书成功 45](#_Toc12931)

[图 43 还书评价 45](#_Toc9314)

[图 44 借阅记录 45](#_Toc17784)

[图 45 查看书籍评价 46](#_Toc28659)

[图 46 修改密码成功 46](#_Toc18233)

[图 47 切换账号 46](#_Toc24520)

[图 48 管理员-修改用户信息 47](#_Toc5894)

[图 49 管理员-删除用户信息 47](#_Toc8885)

[图 50 管理员-增加一个用户 48](#_Toc23782)

[图 51 管理员-批量增加用户 48](#_Toc14494)

[图 52 查找用户界面 49](#_Toc27789)

[图 53 查找成功显示用户信息 49](#_Toc3202)

[图 54 统计-借阅总排榜 49](#_Toc20355)

[图 55 学院借阅排行榜 50](#_Toc28600)

[图 56 男女借阅量对比 50](#_Toc14979)

[图 57 图片导出-选择保存路径 51](#_Toc23161)

[图 58 图片导出-保存到本地桌面 51](#_Toc29218)

# 系统需求分析

图书借阅信息管理程序是典型的**MIS**（**M**anagement **I**nformation **S**ystem）软件，是由在各类企业和事业单位中使用最为普遍的软件之一人事信息管理程序拓展而来。设计、编写一个简单的图书借阅信息管理程序对于学习软件设计的一般方法（如何从分析程序的功能需求入手；如何进行以满足需求的类结构、类对象之间的动态协作和实现功能事务的活动为主要内容的总体设计；如何以总体设计为依据，完成以各类算法设计和工程化编码实现为主要内容的详细设计；以及如何对可以运行的程序进行必要测试。）和了解**MIS**类软件的特点是十分必要的。我们所要设计的《图书借阅信息管理程序》的功能需求如下所示：

## 管理员需求

管理员具有最高权限，管理整个系统，功能概览如下：

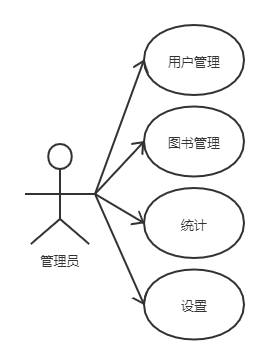


图 1 管理员用例图

### 用户管理

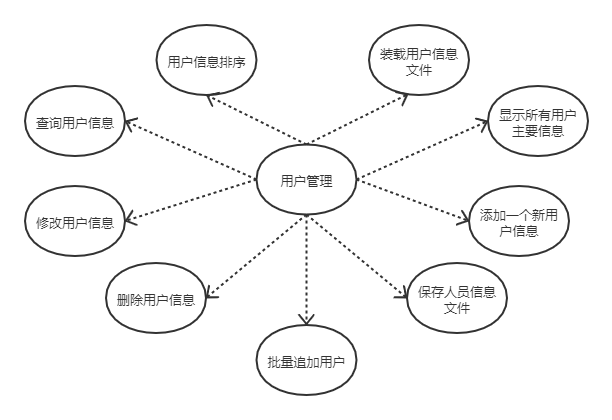


图 2 用户管理功能图

1. **装载用户信息文件：**所有用户信息应能永久保存在一个特定的磁盘文件中。在系统开始运行时应首先打开文件，将文件中的全部信息装载到内存中存放这些信息的对应用户信息集合中。
2. **显示所有用户主要信息：**根据读入文件顺序显示当前用户信息集合中所有用户的主要信息（学号、密码、性别、姓名、院系、班级编号、手机号码、当前借阅书籍、历史借阅记录）。
3. **添加一个新用户信息：**通过交互界面输入一个新用户的主要信息，并将该用户信息记录添加到用户信息集合中。【注意集合中不允许有重复的信息记录。】
4. **查询用户信息：**通过交互界面确定待查询人员的信息，在信息集合中搜索待查询的人员，并根据搜索结果显示不同的信息：如果指定的人员信息记录存在，则显示该人员的详细信息；若指定的人员信息记录不存在，则提示被查询人员不存在。【支持输入条件不完全查找。】
5. **批量追加用户：**通过系统交互界面选择磁盘文件，将文件中的全部信息装载到内存中存放这些信息的对应用户信息集合中。
6. **用户信息排序：**可在用户信息界面选择任意项进行上升或下降排序，并在排序操作后显示全部用户的主要信息。
7. **修改用户信息：**可在用户信息界面选择任意记录点击修改按钮进行修改，直接修改对应项（学号、密码、性别、姓名、院系、班级编号、手机号码），并在编辑修改操作结束后确认修改提醒修改成功后显示修改后的信息。
8. **删除用户信息：**可在用户信息界面选择任意记录点击删除按钮进行删除。
9. **保存人员信息文件：**在系统退出运行时，会自动更新用户信息文件到磁盘中。

### 图书管理

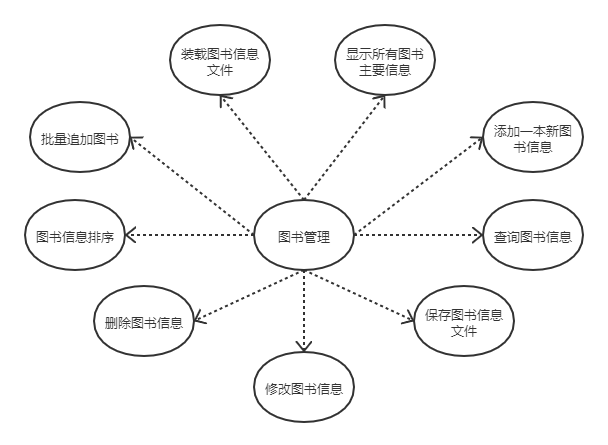


图 3 图书管理功能图

1. **装载图书信息文件：**所有图书信息应能永久保存在一个特定的磁盘文件中。在系统开始运行时应首先打开文件，将文件中的全部信息装载到内存中存放这些信息的对应图书信息集合中。
2. **显示所有图书主要信息：**根据读入文件顺序显示当前图书信息集合中所有图书的主要信息（书编号、书名、作者、出版社、书总数、已借出的数量、出版日期）。
3. **添加一本新图书信息：**通过交互界面输入一本新图书的主要信息，并将该图书信息记录添加到图书信息集合中。【注意集合中不允许有重复的信息记录。】
4. **查询图书信息：**通过交互界面确定待查询图书的信息，在信息集合中搜索待查询的图书，并根据搜索结果显示不同的信息：如果指定的图书信息记录存在，则显示该图书的详细信息；若指定的图书信息记录不存在，则提示被查询图书不存在。【支持输入条件不完全查找。】
5. **批量追加图书：**通过系统交互界面选择磁盘文件，将文件中的全部信息装载到内存中存放这些信息的对应图书信息集合中。
6. **图书信息排序：**可在图书信息界面选择任意项进行上升或下降排序，并在排序操作后显示全部图书的主要信息。
7. **修改图书信息：**可在图书信息界面选择任意记录点击修改按钮进行修改，直接修改对应项，并在编辑修改操作结束后确认修改提醒修改成功后显示修改后的信息。
8. **删除图书信息：**可在图书信息界面选择任意记录点击删除按钮进行删除。
9. **保存图书信息文件**：在系统退出运行时，会自动更新图书信息文件到磁盘中。

### 统计

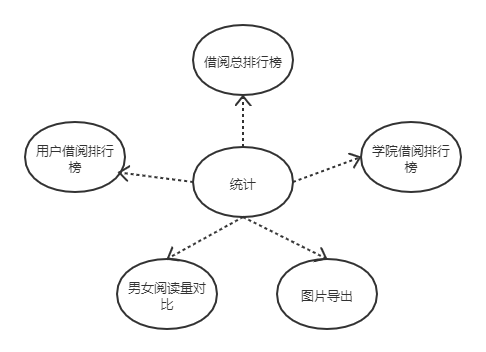


图 4 统计功能图

1. **借阅总排行榜：**统计当前图书信息集合中书籍被借出数量前十名。
2. **用户借阅排行榜：**统计当前用户信息集合中用户借书数量前十名。
3. **学院借阅排行榜：**统计当前各个院系借书数量前十名。
4. **男女阅读量对比：**统计男女生借阅数量。
5. **图片导出：**可以图片格式导出统计界面当前统计量至指定路径。

### 设置

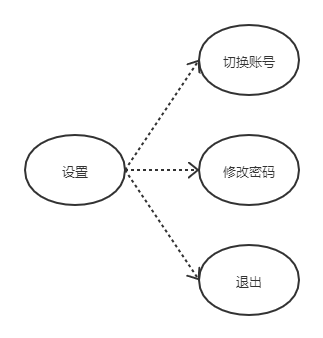


图 5 设置功能图

1. **切换账号：**返回登录欢迎界面切换账号重新登录。
2. **修改密码：**通过交互界面输入账号和新密码，若账号输入不正确则提示账号错误，若账号输入正确且新密码输入不为空则提示修改完成。
3. **退出：**点击退出后，系统退出，交互界面关闭

## 用户需求

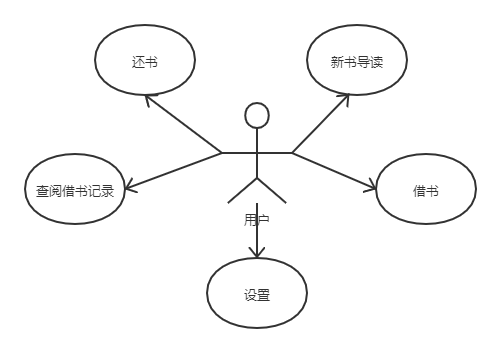


图 6 用户用例图

用户分为学生和老师群体，两者所能做的操作都是一样的，唯一不同点在于借书的上限不同，学生12本，老师20本。

### 新书导读

新书导读栏目用户可以查看到图书信息库里当前新出版的十本书的主要信息（书编号、书名、作者、出版社、书总数、已借出的数量、出版日期），在新书导读的界面上选择借书、预约、查看评价进行操作。

### 借书

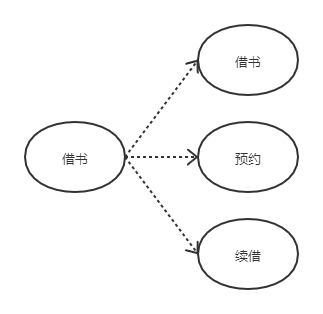


图 7 用户借书功能图

1. **借书：**查看图书信息库里当前所有图书的主要信息（书编号、书名、作者、出版社、书总数、已借出的数量、出版日期），在借书的界面上选择借书、预约、查看评价进行操作，借书期限是30天。
2. **预约：**预约界面可以查看到当前预约书籍的主要信息（书编号、书名、作者、出版社、书总数、已借出的数量、出版日期），可在预约的界面上选择借书、取消预约。如果预约的书籍有库存了，在用户登录时会自动检索并弹窗提醒用户。
3. **续借：**用户可在当前借阅列表中点击续借按钮选择续借，续借期限增加30天。

### 还书

1. **还书：**还书操作可在新书导读、借书界面、当前借书记录点击按钮执行操作；如果超过借阅期限未还书，当用户登录系统时就会自动弹窗提醒，如有未还书籍，用户不可借书、预约；还书时可以对该书进行匿名评价，评价会记录到图书信息中。

### 查阅借书记录

1. **查看当前已借列表：**用户通过交互界面可点击当前已借查看用户现在正在借阅的书籍主要信息（书编号、书名、作者、出版社、书总数、已借出的数量、出版日期、借书时间、应还时间），并且可以在该栏目下进行续借、还书操作。
2. **查看全部借书记录：**用户通过交互界面可点击查看用户的借阅记录，显示借阅过的书籍的主要信息（书编号、书名、作者、出版社、书总数、已借出的数量、出版日期、借书时间、还书时间）。

### 设置

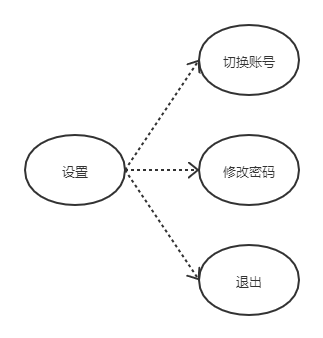


图 8 用户设置功能图

1. **切换账号：**返回登录欢迎界面切换账号重新登录。
2. **修改密码：**通过交互界面输入账号和新密码，若账号输入不正确则提示账号错误，若账号输入正确且新密码输入不为空则提示修改完成。
3. **退出：**点击退出后，系统退出，交互界面关闭

# 总体设计

从软件的需求分析中不难看出：本软件要处理的数据就是描述用户信息和图书信息的各种数据，因此如何组织这些数据、如何操作这些数据、如何在内存和磁盘中存储这些数据、如何为操作这些数据提供交互界面等就是系统总体设计的主要任务。

## 类的静态设计

组成一个《简单图书借阅管理信息系统》的类可以分为两种：一种是组织数据和完成对数据的各项操作的类，另一种是辅助各项数据操作实现的交互界面类。

1. **组织数据和操作数据的类**

描述用户的各种信息是本软件中最主要的数据，因此，应该将描述一个用户的所有信息数据组织在一起，并将对这些数据的基本操作一起封装在一个类中，该信息类可以命名为Person。如何将所有人员（Person对象）信息数据组织在一起对于安全、高效、方便地管理和访问这些数据非常重要的，由于使用者分成管理员、老师、学生，这三个类的属性和基本操作以及它们之间的静态关系如下：

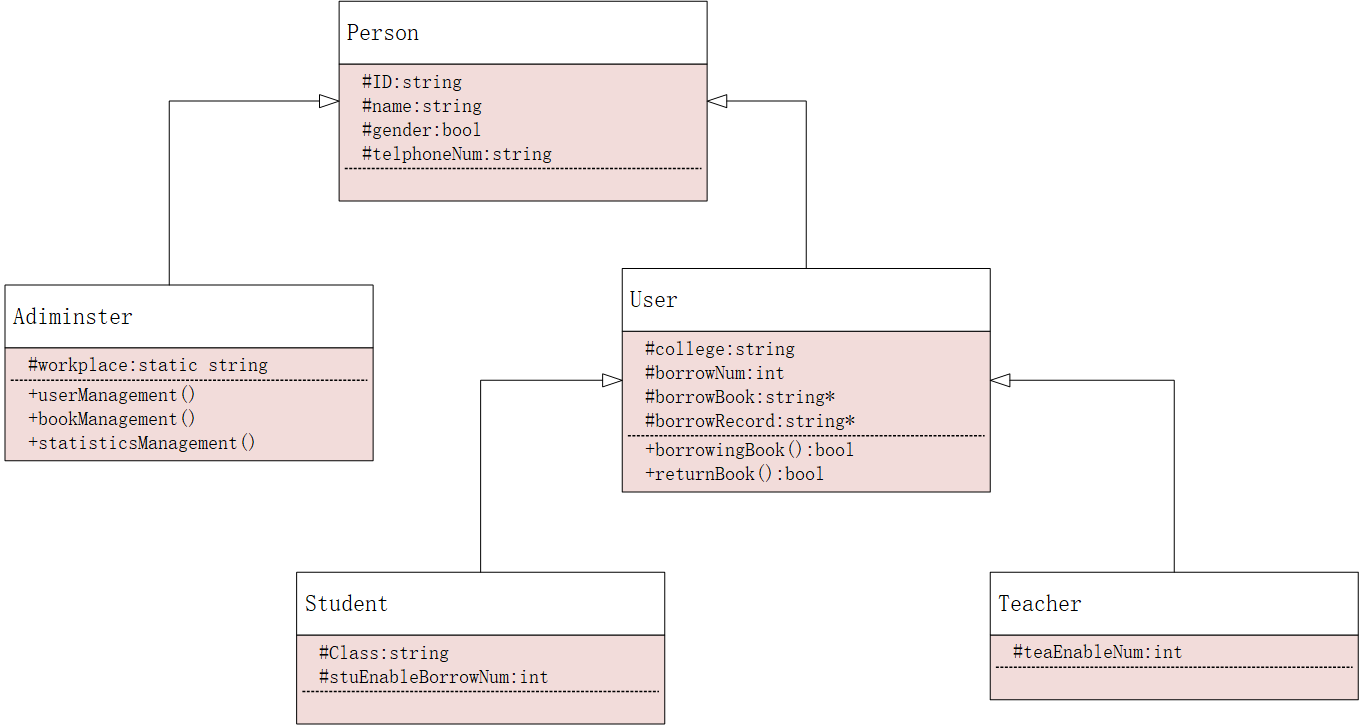


图 9 组织数据和操作数据的类图

1. **提供交互界面的类**

设计这些类的目的是为用户使用《简单图书借阅信息管理系统》的各种功能提供友好、美观的交互界面。归纳功能的操作要求和用户的使用习惯，本系统中的交互界面类应该包括：提供单个用户信息输入/输出/编辑的界面**adminster**、查询用户界面**searchUser**、查询图书界面**searchBook**、增加图书界面**addbook**、增加用户界面**adduser**、管理员修改密码界面**adminchangepassword**、用户修改密码界面**userchangepassword**、用户界面user。如果使用**GUI**(**G**raphics **U**ser **I**nterface)，这些交互界面类一般都从提供了基本交互功能的系统基类中派生定义的。因此这些类的设计与系统开发环境和使用什么基类库有着直接关系。在本系统的设计中我们选择了**C++**开发环境，并使用QT作为基类支持。

## 类的动态设计

依据系统需求分析中的用例功能描述，使用顺序图分析上述各类对象在每个用例功能中的相互作用是进一步确定各类的属性和操作，特别是确定类对使用者提供的接口协议是十分关键的。同时为各种系统功能的算法设计和代码实现提供最重要的依据。

1. **装载人员信息文件**

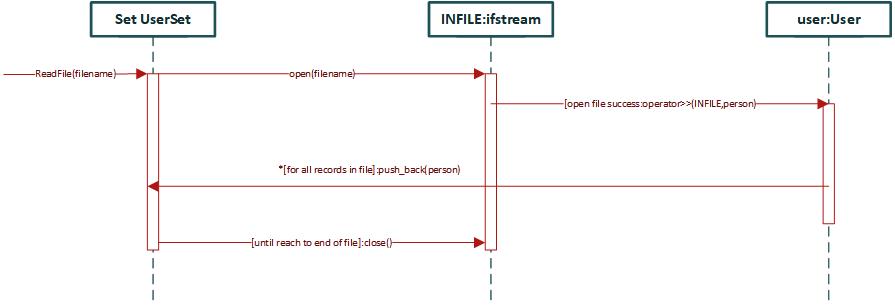


图 10 装载人员信息文件顺序图

\* **ifstream**是标准输入文件流类，**open**和**close**是该类的成员函数。

1. **主要功能顺序图**

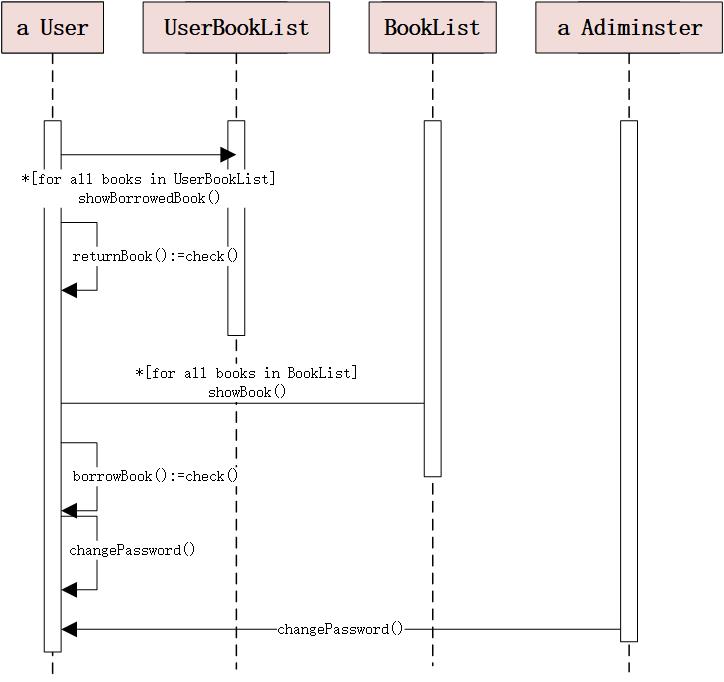


图 11 主要功能顺序图

## 功能事务的实现过程

在类的设计完成后，就可以按照需求模型中的功能事务描述，通过绘制整个软件的活动图和各个功能事务执行过程的活动图，设计软件的运行模型。

**1. 简单图书借阅信息管理系统的总活动图**

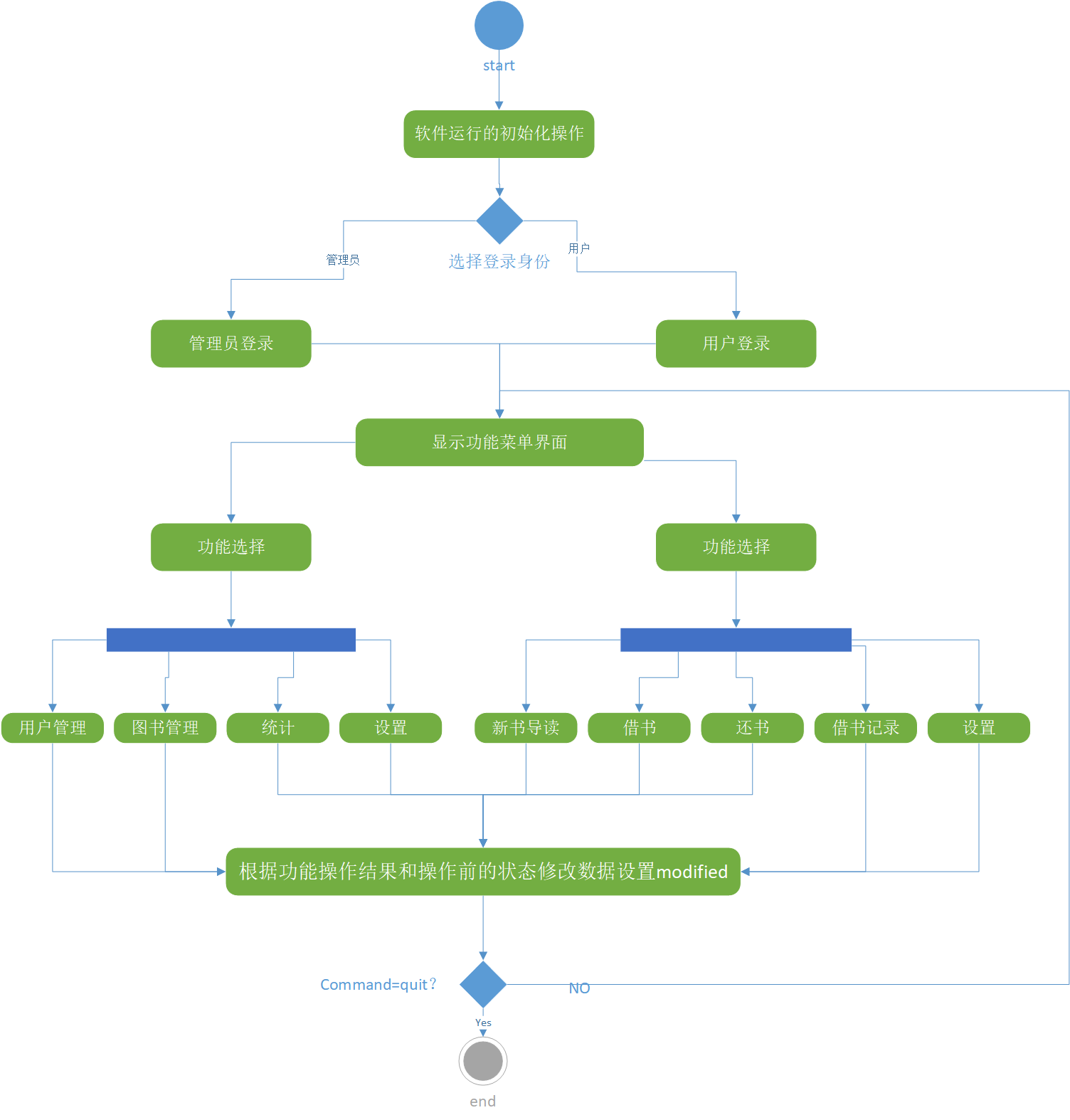


图 12 简单图书借阅信息管理系统的总活动图

1. **装载人员信息文件功能活动图**

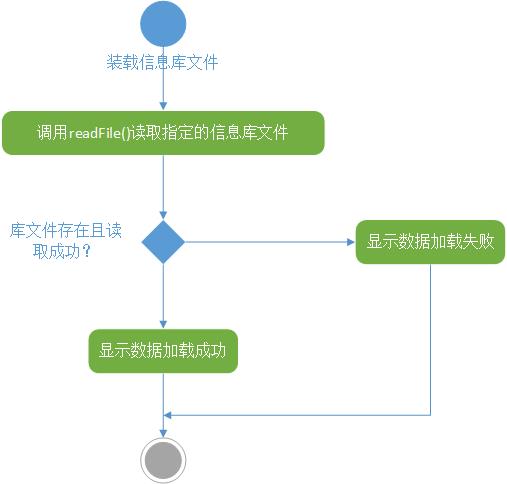


图 13 装载数据文件功能活动图

1. **添加用户信息记录功能活动图**

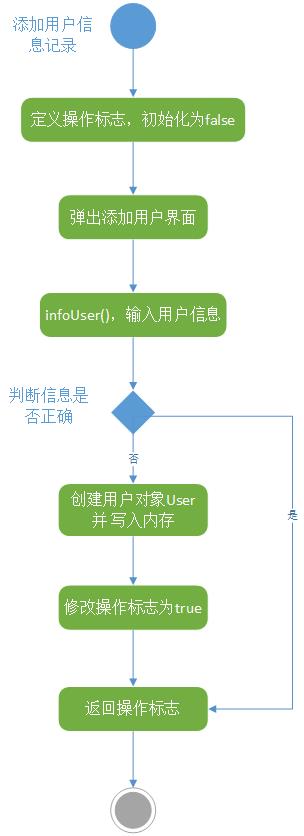


图 14 添加用户信息记录功能活动图

1. **删除用户信息记录功能活动图**



图 15 删除用户信息记录活动图

1. **修改指定人员信息记录功能活动图**

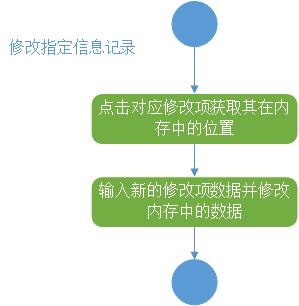
****

图 16 修改指定人员信息记录功能活动图

1. **修改密码功能活动**

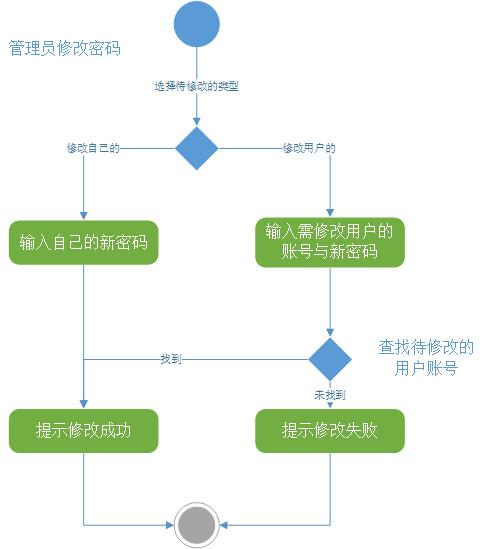


图 17 修改密码活动图

1. **切换账号功能活动**

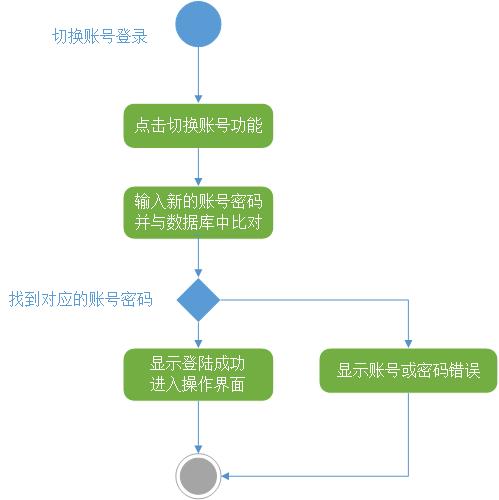
****

图 18 切换账号功能活动图

1. **退出软件运行功能活动图**

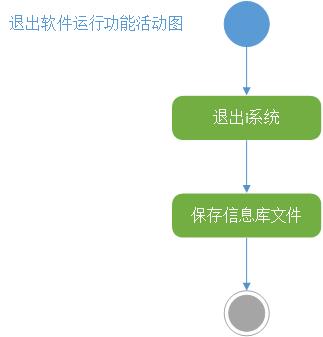
****

图 19 退出软件运行功能活动图

# 系统详细设计

## 类的详细设计

根据模型中的类图和相应的类成员描述设计软件中各个类的定义代码和主要成员函数的实现算法。

1. **UserPerson类**

class **UserPerson** : public Person

{

protected:

string college; //学院

string classID; //班级

vector<string> reservation; //预约书籍

vector<BookAndBorrowTime> borrowInfor; //当前借阅

vector<BookAndBorrowTime> borrowRecord; //借阅记录

public:

bool **addReservation**(string isbn) // 增加预约

void **delReservation**(string isbn) // 取消预约

bool **checkReservation**(BookTable bookTable) // 检查预约书籍有无库存

bool **modifyReturnTime**(string bookID,int flag,int day = 0) //flag==1为还书，flag==2为续借

bool **ifDueSoon**() // 判断是否有书籍即将到期未还，如有，在用户登录时弹窗提醒

……

}

1. **Account\_Password账号密码类**

class **Account\_Password**

{

protected:

string account;

string password;

public:

**Account\_Password**() {}

**Account\_Password**(string account\_in, string password\_in)

**Account\_Password**(const Account\_Password &in)

bool **Judge\_exist**(Account\_Password account\_exist) //判断账号是否存在，存在返回true，不存在返回false

int **Modify**(string password\_in)

friend class Account\_group;

Set & Get方法……

};

1. **Account\_group账号集合类**

class **Account\_group**

{

private:

vector<Account\_Password> groups;

public:

**Account\_group**() {}

int **SearchAccountFromAccount**(string Account)

bool **Modify**(string IDIn, string newPassword) //

int **write\_file**(const string file\_name)

int **read\_file**(const string file\_name)

bool **del**(string ID)

bool **add**(string account, string password)

bool **logIn**(string account\_in, string password\_in)

string **getPassword**(string id)

vector<Account\_Password> **getGroups**()

};

1. **Book图书类**

class **Book**

{

protected:

string ISBN; //书编号

string bookName; //书名

string bookWriter; //作者

string bookPress; //出版社

int bookTotalNum; //书总数

int bookBorrowOutNum; //已借出的数量

string bookPubYear; //出版年

string bookPubMonth; //出版月

string bookPubDay; //出版日

vector<string> bookComment;//评论

public:

**Book**() {}

**Book**(string ISBNIn, string bookNameIn, string bookWriterIn, string bookPressIn, int bookTotalNumIn, int bookBorrowOutNumIn, string bookPubYearIn, string bookPubMonthIn, string bookPubDayIn)

friend ostream &operator<<(ostream &os, Book &book)

friend class BookTable;

Set & Get方法……

};

1. **BookTable图书集合类**

class **BookTable**

{

private:

vector<Book \*> bookTable;

public:

**BookTable**() {}

~**BookTable**(){

bool **addBook**(Book bookIn)

bool **deleteBook**(string IDIn)

bool **modifyBook**(Book bookIn) //修改

vector<Book \*> **searchBook**(Book bookIn)

vector<Book \*> **getBookTable**()

bool **readFile**(const string fileName)

bool **writeFile**(const string fileName){

void **testDis**(){

bool **replaceTable**(vector<Book \*> newTable)

QVector <Book\*> **sortBorrowTimes**(){ // 根据书籍被借出数量排序返回前十名指针数组

vector <Book\*> **sortPublishTime**(){ // 根据出版日期按大到小排序

}

## 主要功能函数的实现

1. **用户数据展示**

void User::**layData**(vector <Book\*>booklist,QTableView \*view,int flag){

vector<Book\*>currentBorrowBook=userperson->getBorrowBook(\*bookTable);

vector<int> currentBookIndex;

vector <string> reservationIsbn = userperson->getReservation();//预约书籍的编号

vector <int> reservationBookIndex;

for(unsigned int i = 0;i < reservationIsbn.size(); i ++){

reservationBookIndex.push\_back(bookTable->searchBookFromID(reservationIsbn[i]));

}

for(unsigned int i=0;i<currentBorrowBook.size();i++){

currentBookIndex.push\_back(bookTable->searchBookFromID(currentBorrowBook[i]->getISBN().data()));

}

qDebug()<<currentBookIndex;

view->setEditTriggers(QAbstractItemView::NoEditTriggers);

QStandardItemModel \*imodel=new QStandardItemModel(this);

view->*setModel*(imodel);

view->verticalHeader()->hide();

view->horizontalHeader()->setStyleSheet("QHeaderView::section {""color: black;padding-left: 4px;border: 1px solid #6c6c6c;}");

QStringList headercontent;//设置表头

if(flag==1){ // 借书记录

headercontent<<"书编号"<<"书名"<<"作者"<<"出版社"<<"书总数"<<"已借出的数量"<<"出版日期"<<"续借"<<"还书";

}

else if(flag==4) // 当前借书记录

{

headercontent<<"书编号"<<"书名"<<"作者"<<"出版社"<<"书总数"<<"已借出的数量"<<"出版日期"<<"续借"<<"还书"<<"借书时间"<<"应还时间";

}

else if(flag==3){ // 总的借书记录

headercontent<<"书编号"<<"书名"<<"作者"<<"出版社"<<"书总数"<<"已借出的数量"<<"出版日期"<<"借书时间"<<"还书时间";

}

else{ // 新书导读 & 所有书单

headercontent<<"书编号"<<"书名"<<"作者"<<"出版社"<<"书总数"<<"已借出的数量"<<"出版日期"<<"借书"<<"还书"<<"预约"<<"查看评价";

}

imodel->setHorizontalHeaderLabels(headercontent);

if(flag==1||flag==3){

imodel->setColumnCount(9);

}

else if(flag==4){

imodel->setColumnCount(11);

}

else{

imodel->setColumnCount(11);

}

imodel->setRowCount(booklist.size());

int tempCountNum = 0;

int reservationCountNum=0;

for(unsigned int i =0;i<booklist.size();i++){//依次在相应位置放入数据

imodel->setItem(i, 0, new QStandardItem(QString::fromUtf8(booklist[i]->getISBN().data())));

imodel->setItem(i, 1, new QStandardItem(QString::fromUtf8(booklist[i]->getBookName().data())));

imodel->setItem(i, 2, new QStandardItem(QString::fromUtf8(booklist[i]->getbookWriterr().data())));

imodel->setItem(i, 3, new QStandardItem(QString::fromUtf8(booklist[i]->getBookPress().data())));

imodel->setItem(i, 4, new QStandardItem(QString::number(booklist[i]->getBookTotalNum())));

imodel->setItem(i, 5, new QStandardItem(QString::number(booklist[i]->getBookBorrowOutNum())));

imodel->setItem(i, 6, new QStandardItem(QString::fromUtf8(booklist[i]->getBookYear().data())

+"-"+QString::fromUtf8(booklist[i]->getBookMonth().data())

+"-"+QString::fromUtf8(booklist[i]->getBookDay().data())));

QPushButton \*borrowBookBtn=new QPushButton(this);

if(flag==0&&currentBookIndex.size()!=0){

if(i != unsigned(currentBookIndex[tempCountNum]))borrowBookBtn->setText("借书");

else {

borrowBookBtn->setText("已借");

tempCountNum++;

}

}

else if(flag==1||flag==4){

borrowBookBtn->setText("续借");

}

else

borrowBookBtn->setText("借书");

QPushButton \*reserveBookBtn=new QPushButton(this);

if(flag==0&&reservationBookIndex.size()!=0){

if(i != unsigned(reservationBookIndex[reservationCountNum]))reserveBookBtn->setText("预约");

else {

reserveBookBtn->setText("已预约");

reservationCountNum++;

}

}

else

reserveBookBtn->setText("预约");

connect(borrowBookBtn,&QPushButton::clicked,this,[=](){

if(borrowBookBtn->text()=="借书"){

borrowBook(view);

if(flag==4||flag==2)

view->close();

int index=view->currentIndex().row();

imodel->setItem(i, 5, new QStandardItem(QString::number(booklist[index]->getBookBorrowOutNum())));

borrowBookBtn->setText("已借");

}

else{

QMessageBox::information(this,"借书","已借");

}

});

connect(reserveBookBtn,&QPushButton::clicked,this,[=](){

if(reserveBookBtn->text()=="预约") {

reserveBook(view);

if(GetReserveStatus(view)==6){

borrowBookBtn->setText("已借");

}

else reserveBookBtn->setText("已预约");

}

else{

QMessageBox::information(this,"预约" ,"已预约");

}

});

QPushButton \*giveBackBookBtn=new QPushButton(this);

giveBackBookBtn->setText("还书");

connect(giveBackBookBtn,&QPushButton::clicked,this,[=](){

giveBackBook(view);

if(flag==4)

view->close();

int index=view->currentIndex().row();

imodel->setItem(i, 5, new QStandardItem(QString::number(booklist[index]->getBookBorrowOutNum())));

borrowBookBtn->setText("借书");

});

QPushButton \*showCommentBtn=new QPushButton(this);

showCommentBtn->setText("查看评价");

connect(showCommentBtn,&QPushButton::clicked,this,[=](){

showComment();

});

if(flag==1||flag==4){

view->setIndexWidget(imodel->*index*(i, 7), borrowBookBtn);

view->setIndexWidget(imodel->*index*(i, 8), giveBackBookBtn);

}

else if(flag==0||flag==2){

view->setIndexWidget(imodel->*index*(i, 7), borrowBookBtn);

view->setIndexWidget(imodel->*index*(i, 8), giveBackBookBtn);

view->setIndexWidget(imodel->*index*(i, 9), reserveBookBtn);

view->setIndexWidget(imodel->*index*(i, 10), showCommentBtn);

}

else {

imodel->setItem(i, 7, new QStandardItem(QString(userperson->getBorrowInfor()[i].getBorrowTime().toString("yyyy-MM-dd"))));

imodel->setItem(i, 8, new QStandardItem(QString(userperson->getBorrowInfor()[i].getReturnTime().toString("yyyy-MM-dd"))));

}

if(flag==4){

imodel->setItem(i, 9, new QStandardItem(QString(userperson->getBorrowInfor()[i].getBorrowTime().toString("yyyy-MM-dd"))));

imodel->setItem(i, 10, new QStandardItem(QString(userperson->getBorrowInfor()[i].getReturnTime().toString("yyyy-MM-dd"))));

}

}

view->setColumnWidth(1,145);

view->setColumnWidth(3,145);

view->setSortingEnabled(true);

}

1. 借阅排行榜

QVector <Book\*> **sortBorrowTimes**(){ // 根据书籍被借出数量排序返回前十名指针数组

QVector<Book \*> pBook;

vector<Book \*> tmp = bookTable;

for(unsigned int i = 0;i < tmp.size(); i++){

for(unsigned int j = i;j < tmp.size(); j++){

if(tmp[i]->getBookBorrowOutNum() < tmp[j]->getBookBorrowOutNum()){

Book \*temp;

temp = tmp[j];

tmp[j] = tmp[i];

tmp[i] = temp;

}

}

}

for(unsigned int i = 0;i < 10; i++){

pBook.push\_back(tmp[i]);

}

return pBook;

}

1. **用户修改密码**

void UserChangePassword::**on\_pushButton\_2\_clicked**()

{

if (ui->newPassword->text() == ui->renewPassword->text())

{ //设置新密码

usertable->modifyPassword(QString(ui->newPassword->text()).toUtf8().toStdString());

QMessageBox message(QMessageBox::Information, "提示", "修改成功！");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

usertable->getStudentAccount()->write\_file("stuAccount.txt");

usertable->getTeacherAccount()->write\_file("teaAccount.txt");

}

else if (QString(ui->newPassword->text()).toUtf8().toStdString() == "")

{ // 未输入新密码

QMessageBox message(QMessageBox::Critical, "提示", "未输入新密码！");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

}

else if (QString(ui->renewPassword->text()).toUtf8().toStdString() == "")

{ // 未输入确认密码

QMessageBox message(QMessageBox::Critical, "提示", "未再次输入密码确认！");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

}

else

{ // 新密码与确认密码不匹配

QMessageBox message(QMessageBox::Critical, "提示", "两次输入密码不匹配！");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

}

}

1. **新用户注册**

void MainWindow::**on\_pushButton\_2\_clicked**() //注册

{

initNewpeopleValue();

qDebug() << people->getAccount().data();

qDebug() << people->getPassword().data();

qDebug() << newpeople->getID().data();

qDebug() << newpeople->getTel().data();

qDebug() << newpeople->getName().data();

qDebug() << newpeople->getGender().data();

qDebug() << newpeople->getCollege().data();

qDebug() << newpeople->getClassID().data();

if (newpeople->getID() != "" && people->getPassword() != "" && newpeople->getTel() != "" && newpeople->getName() != "" && newpeople->getGender() != "" && newpeople->getCollege() != "" && newpeople->getClassID() != "")

{

// if(ui->checkBoxAdmin\_2->isChecked()==true){

// adminGroup.add(people->getAccount(),people->getPassword());

// AdiminsterPerson admin(newpeople->getID(), newpeople->getName(), newpeople->getGender(), newpeople->getTel());

// adminlist.addAdmin(admin);

// QMessageBox message(QMessageBox::Information, "提示","<font color=' black'>注册成功！</font>");

// QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

// message.setWindowIcon(\*icon);

// message.exec();

// adminlist.writeFile("admin.txt");

// }

if (ui->checkBoxTea\_2->isChecked() == true)

{

Teacher t;

t.setID(newpeople->getID());

t.setName(newpeople->getName());

t.setGender(newpeople->getGender());

t.setCollege(newpeople->getCollege());

t.setClassID(newpeople->getClassID());

t.setTelphoneNum(newpeople->getTel());

if (userlist.addTeacher(t))

{

// QMessageBox::information(this,"注册","注册成功");

qDebug() << "teacherTable[167] addr: ";

qDebug() << userlist.getStudentTable()[167];

userlist.getTeacherAccount()->add(people->getAccount(), people->getPassword());

QMessageBox message(QMessageBox::Information, "提示", "<font color=' black'>注册成功！</font>");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

userlist.writeFile("student.txt", "teacher.txt");

}

else

{

QMessageBox message(QMessageBox::Critical, "提示", "<font color=' black'>该用户已存在！</font>");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

}

}

else if (ui->checkBoxStu\_2->isChecked() == true)

{

Student s;

s.setID(newpeople->getID());

s.setName(newpeople->getName());

s.setGender(newpeople->getGender());

s.setCollege(newpeople->getCollege());

s.setClassID(newpeople->getClassID());

s.setTelphoneNum(newpeople->getTel());

// qDebug()<<userlist.addStudent(s);

if (userlist.addStudent(s))

{

userlist.getStudentAccount()->add(people->getAccount(), people->getPassword());

QMessageBox message(QMessageBox::Information, "提示", "<font color=' black'>注册成功！</font>");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

userlist.writeFile("student.txt", "teacher.txt");

}

else

{

QMessageBox message(QMessageBox::Critical, "提示", "<font color=' black'>该用户已存在！</font>");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

}

}

}

else if (people->getAccount() == "" || people->getPassword() == "" || newpeople->getTel() == "" || newpeople->getName() == "" || newpeople->getGender() == "" || newpeople->getCollege() == "" || newpeople->getClassID() == "")

{

QMessageBox message(QMessageBox::Critical, "提示", "<font color=' black'>未完整填写信息！</font>");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

}

else

{

QMessageBox message(QMessageBox::Critical, "提示", "<font color=' black'>未勾选身份！</font>");

QIcon \*icon = new QIcon(":/image/CNUlibrary.jpg");

message.setWindowIcon(\*icon);

message.*exec*();

}

}

1. **检查预约**

bool **checkReservation**(BookTable bookTable) // 检查预约书籍有无库存

{

vector<Book \*> temp;

for(unsigned int i = 0;i < reservation.size(); i ++){

int index = bookTable.searchBookFromID(reservation[i]); // 获取每个预约书的序号

if(index != -1){ // 如果该书有库存则消息提醒用户

if(bookTable.getBookTable()[i]->getBookTotalNum() > bookTable.getBookTable()[i]->getBookBorrowOutNum()){

temp.push\_back(bookTable.getBookTable()[i]);

}

}

}

if(temp.size()!=0) return true;

else return false;

}

1. **借书**

void User::**borrowBook**(QTableView \*view){

int flag=0;

int index=view->currentIndex().row();

QAbstractItemModel \*model = ui->bookDataView->model ();

QString borrowBookID=model->*data*(model->*index*(index,0)).toString();

qDebug()<<borrowBookID;

vector<Book\*>currentBorrowBook=userperson->getBorrowBook(\*bookTable);

for(unsigned int i=0;i<currentBorrowBook.size();i++){

if(currentBorrowBook[i]->getISBN().data()==borrowBookID){

flag=1;

}

}

int borrowflag=userperson->borrowBook(borrowBookID.toStdString(),\*bookTable);

if(flag==1){

QMessageBox::information(this,"借书","已借该书");

}

else if(flag==0&&borrowflag==0){

QMessageBox::information(this,"借书","没有可借库存");

}

else if(flag==0&&borrowflag==1){

QMessageBox::information(this,"借书","借书已达上限");

}

else if(flag==0&&borrowflag==2){

int locate=bookTable->searchBookFromID(borrowBookID.toStdString());

int currentNum=bookTable->getBookTable()[locate]->getBookBorrowOutNum();

bookTable->getBookTable()[locate]->setBookBorrowOutNum(++currentNum);

QMessageBox::information(this,"借书","借书成功");

}

else if(flag==0&&borrowflag==3){

QMessageBox::information(this,"借书","该书不存在");

}

else if(flag==0&&borrowflag==4){

QMessageBox::information(this,"借书","您有逾期未还的书籍，不能借书！");

}

bookTable->writeFile("book.txt");

}

1. **还书**

void User::**giveBackBook**(QTableView \*view){

int flag=0;

int index=view->currentIndex().row();

QAbstractItemModel \*model = view->model ();

QString givebackBookID=model->*data*(model->*index*(index,0)).toString();

vector<Book\*>currentBorrowBook=userperson->getBorrowBook(\*bookTable);

for(unsigned int i=0;i<currentBorrowBook.size();i++){

if(currentBorrowBook[i]->getISBN().data()==givebackBookID){

flag=1;

break;

}

}

if(flag==0){

QMessageBox::information(this,"还书","请先借书");

}

else{

int locate=bookTable->searchBookFromID(givebackBookID.toStdString());

int currentNum=bookTable->getBookTable()[locate]->getBookBorrowOutNum();

bookTable->getBookTable()[locate]->setBookBorrowOutNum(--currentNum);

userperson->giveBackBook(givebackBookID.toStdString(),\*bookTable);

QMessageBox::information(this,"还书","还书成功");

commentBook();

}

bookTable->writeFile("book.txt");

}

1. **新书推荐**

vector <Book\*> **sortPublishTime**(){ // 根据出版日期按大到小排序

vector<Book \*> pBook; // 存放十本新书

vector<Book \*> tmp = bookTable;

for(unsigned int i = 0;i < tmp.size(); i++){

for(unsigned int j = i;j < tmp.size(); j++){

if(tmp[i]->getBookYear() < tmp[j]->getBookYear()){

Book \*temp;

temp = tmp[j];

tmp[j] = tmp[i];

tmp[i] = temp;

}

else if(tmp[i]->getBookYear() == tmp[j]->getBookYear()){

if(tmp[i]->getBookMonth() < tmp[j]->getBookMonth()){

Book \*temp;

temp = tmp[j];

tmp[j] = tmp[i];

tmp[i] = temp;

}

if(tmp[i]->getBookMonth() == tmp[j]->getBookMonth()){

if(tmp[i]->getBookDay() < tmp[j]->getBookDay()){

Book \*temp;

temp = tmp[j];

tmp[j] = tmp[i];

tmp[i] = temp;

}

}

}

}

}

for(unsigned int i = 0;i < 10; i++){

pBook.push\_back(tmp[i]);

}

return pBook;

}

1. **展示图书数据**

void Adiminster::**layData**(vector <T\*>booklist,QTableView \*view,int flag){

// string ISBN; //书编号

// string bookName; //书名

// string bookWriter; //作者

// string bookPress; //出版社

// int bookTotalNum; //书总数

// int bookBorrowOutNum; //已借出的数量

// string bookPubYear; //出版年

// string bookPubMonth; //出版月

// string bookPubDay; //出版日

QStandardItemModel \*imodel=new QStandardItemModel(this);

view->*setModel*(imodel);

view->verticalHeader()->hide();

view->horizontalHeader()->setStyleSheet("QHeaderView::section {" "color: black;padding-left: 4px;border: 1px solid #6c6c6c;}");

QStringList headercontent;//设置表头

if(flag!=1){

headercontent<<"书编号"<<"书名"<<"作者"<<"出版社"<<"书总数"<<"已借出的数量"<<"出版日期"<<"修改"<<"删除";

imodel->setColumnCount(9);

}

else{

headercontent<<"书编号"<<"书名"<<"作者"<<"出版社"<<"书总数"<<"已借出的数量"<<"出版日期";

imodel->setColumnCount(7);

}

imodel->setHorizontalHeaderLabels(headercontent);

imodel->setRowCount(booklist.size());

for(unsigned int i =0;i<booklist.size();i++){//依次在相应位置放入数据

imodel->setItem(i, 0, new QStandardItem(QString::fromUtf8(booklist[i]->getISBN().data())));

imodel->setItem(i, 1, new QStandardItem(QString::fromUtf8(booklist[i]->getBookName().data())));

imodel->setItem(i, 2, new QStandardItem(QString::fromUtf8(booklist[i]->getbookWriterr().data())));

imodel->setItem(i, 3, new QStandardItem(QString::fromUtf8(booklist[i]->getBookPress().data())));

imodel->setItem(i, 4, new QStandardItem(QString::number(booklist[i]->getBookTotalNum())));

imodel->setItem(i, 5, new QStandardItem(QString::number(booklist[i]->getBookBorrowOutNum())));

imodel->setItem(i, 6, new QStandardItem(QString::fromUtf8(booklist[i]->getBookYear().data())

+"-"+QString::fromUtf8(booklist[i]->getBookMonth().data())

+"-"+QString::fromUtf8(booklist[i]->getBookDay().data())));

if(flag!=1){

QPushButton \*modifyBtn=new QPushButton(this);

modifyBtn->setText("修改");

connect(modifyBtn,&QPushButton::clicked,this,[=](){

if(modifyBtn->text()=="修改"){

ui->bookDataTableView->setEditTriggers(QAbstractItemView::DoubleClicked);

ReadOnlyDelegate\* readOnlyDelegate = new ReadOnlyDelegate();

view->setItemDelegateForColumn(0, readOnlyDelegate); //设置某列只读

modifyBtn->setText("确认修改");

}

else{

modifyBook();

QMessageBox::information(this,"修改","修改成功，已为你更新当前数据");

showDateTable();

modifyBtn->setText("修改");

}

});

QPushButton \*delBtn=new QPushButton(this);

delBtn->setText("删除");

connect(delBtn,&QPushButton::clicked,this,[=](){

del(ui->tabWidget->currentIndex());

showDateTable();

});

view->setIndexWidget(imodel->*index*(i, 7), modifyBtn);

view->setIndexWidget(imodel->*index*(i, 8), delBtn);

}

}

view->setColumnWidth(1,180);

view->setColumnWidth(3,180);

view->setColumnWidth(6,148);

view->setSortingEnabled(true);

}

1. **修改用户信息**

void Adiminster::**modifyUser**(){

unsigned int row=ui->userDataTableView->currentIndex().row();//得到当前点击的行坐标

unsigned int colum=ui->userDataTableView->currentIndex().column();//得到当前点击的列坐标坐标

if(colum==11){

QAbstractItemModel \*model = ui->userDataTableView->model ();

QModelIndex index = model->*index*(row,0);//选中的行第一列的内容

QVariant data = model->*data*(index);//选中的第一列的值

stulist[row]->setID(QString(model->*data*(model->*index*(row,0)).toString()).toStdString());

usertable->getStudentAccount()->Modify(stulist[row]->getID(),(QString(model->*data*(model->*index*(row,1)).toString()).toStdString()));

stulist[row]->setGender(QString(model->*data*(model->*index*(row,2)).toString()).toStdString());

stulist[row]->setName(QString(model->*data*(model->*index*(row,3)).toString()).toStdString());

stulist[row]->setCollege(QString(model->*data*(model->*index*(row,4)).toString()).toStdString());

stulist[row]->setClassID(QString(model->*data*(model->*index*(row,5)).toString()).toStdString());

stulist[row]->setTelphoneNum(QString(model->*data*(model->*index*(row,6)).toString()).toStdString());

qDebug()<<index.data();

usertable->getStudentAccount()->write\_file("stuAccount.txt");

usertable->writeFile("student.txt","teacher.txt");

}

}

1. **添加新用户**

void Adiminster::**addNewUser**(){

AddUser \*adduser=new AddUser;

this->stu=new Student();

adduser->show();

adduser->*exec*();

stu=adduser->getStu();

if(stu->getID()=="\*"){

return;

}

//qDebug()<<book->getBookName().data();

stulist.push\_back(new Student(stu->getID(),stu->getName(),

stu->getGender(), stu->getTel(),

stu->getCollege(), stu->getClassID()));

}

## 程序主函数的算法流程

int main(int argc, char \*argv[])

{

QApplication a(*argc*, argv);

// //主登录界面

MainWindow w;

w.show();

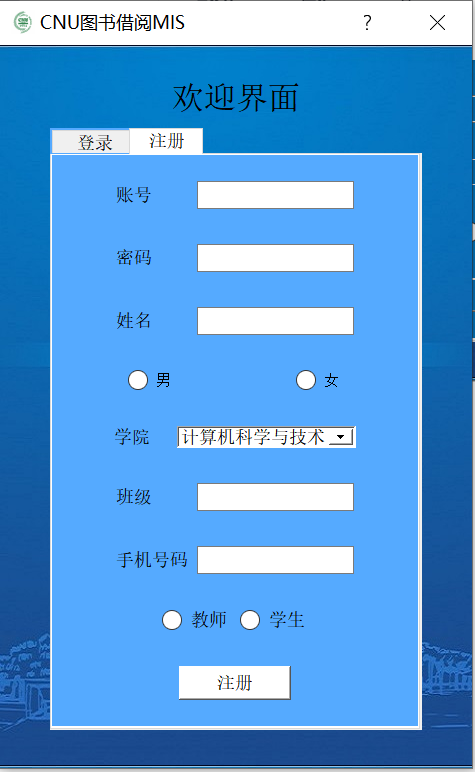
return a.exec();

}

# 系统测试

## 操作界面测试

1. 《简单图书借阅信息管理系统》是一个菜单驱动的程序，程序一旦运行会首先显示一个登录和注册的欢迎界面。用户可以根据需要选择身份登录，执行相应的功能。每次重新运行，系统都会自动从信息库文件中装载信息记录。

图 20 欢迎界面（左登录，右注册）

1. 管理员界面——用户界面



图 21 管理员界面——用户界面

1. 管理员界面——图书界面



图 22 管理员界面——图书界面

1. 管理员界面——统计界面

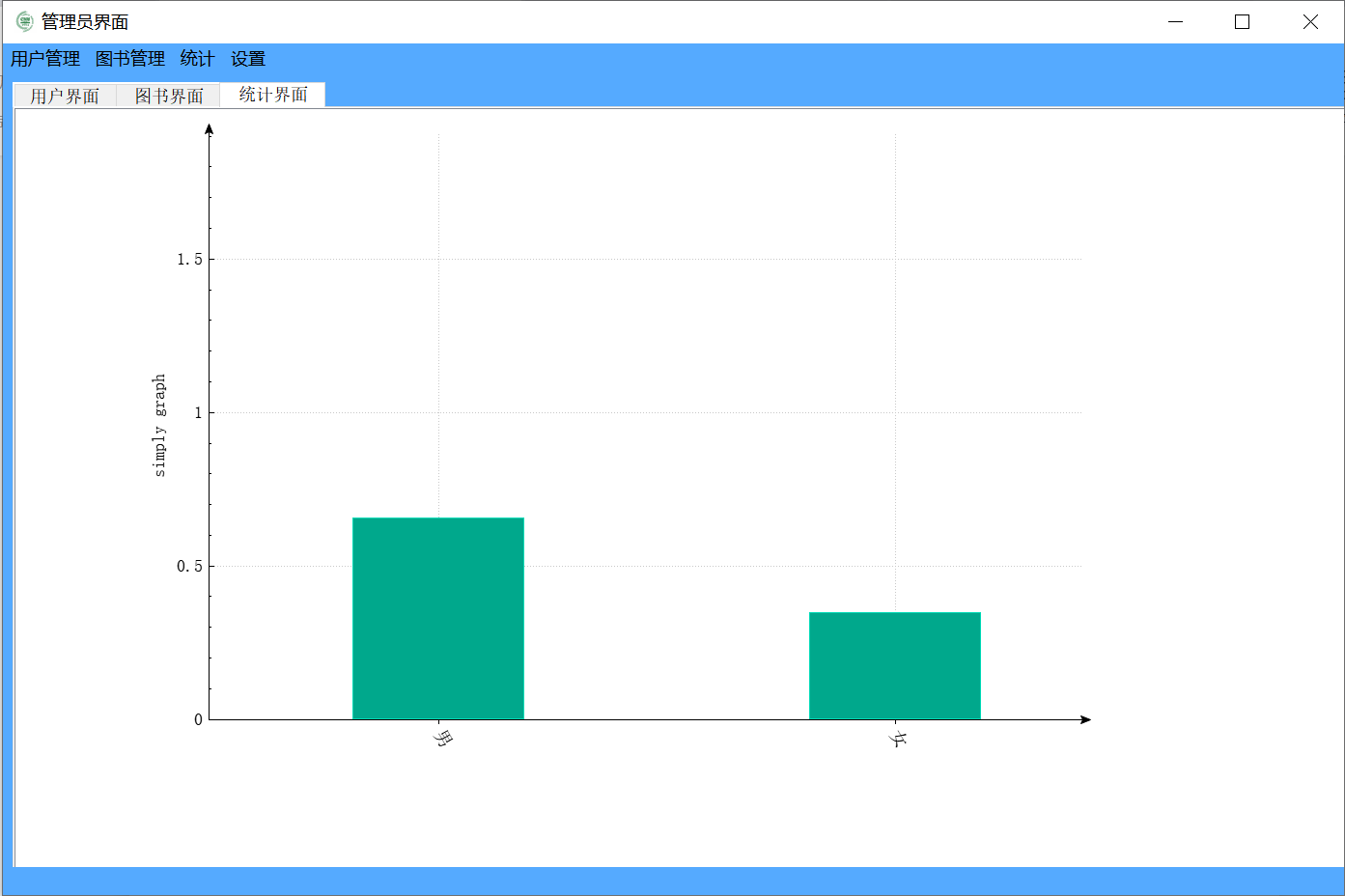


图 23 管理员界面——统计界面

1. 用户界面——借书



图 24 用户界面——借书界面

1. 用户界面——新书导读



图 25 用户界面——新书导读

1. 用户界面——预约书籍列表

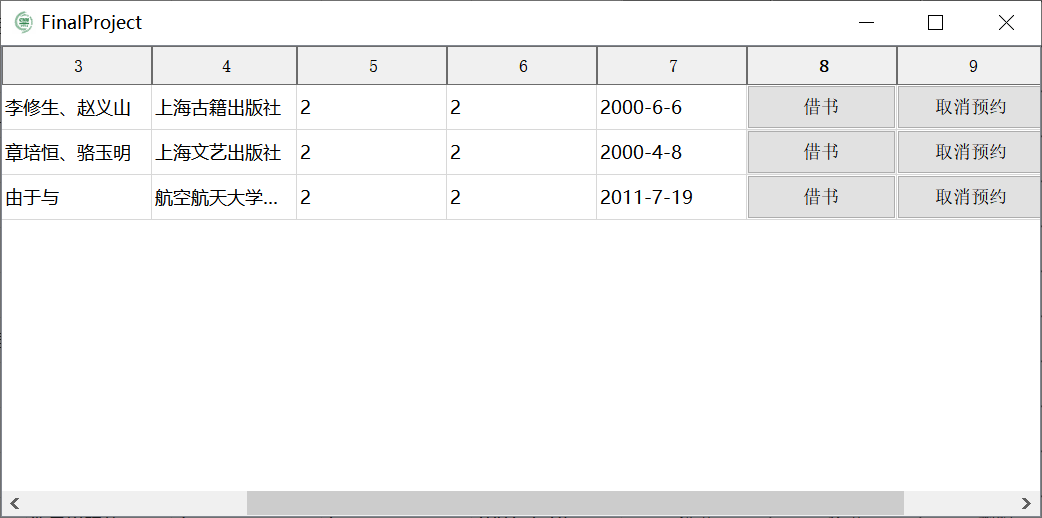


图 26 用户界面——预约书籍列表

1. 用户界面——当前借书列表



图 27 用户界面——当前借书列表

1. 用户界面——全部借书记录



图 28 用户界面——全部借书记录

## 功能测试

1. 注册功能
2. 检测未完整填入信息，则弹窗提示



图 29 注册-未填写完整信息

1. 输入一个已在用户集合中的信息，提示用户已存在

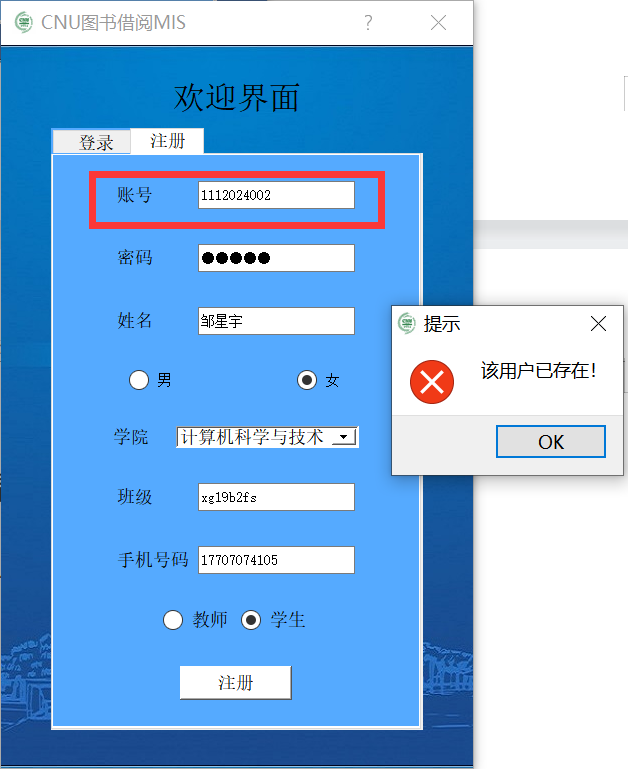


图 30 注册-用户已存在

1. 填写完整正确的信息，注册成功



图 31 注册-注册成功

1. 登录
2. 账号或密码为空时，弹窗提示未完整填写信息，登录失败



图 32 登录-登录失败

1. 填写错误账号密码



图 33 登录-账号或密码错误

1. 正确账号密码登录成功



图 34 登录成功

1. 借书
2. 当书总数=已借出数量时，弹窗提示没有可借的库存



图 35 借书-无可借库存

1. 点击借书按钮，若有有可借库存则弹窗借书成功，书的状态变为已借，借出数量+1



图 36 借书-借书成功



图 37 借书成功状态更改

1. 预约
2. 预约成功

图 38 预约成功

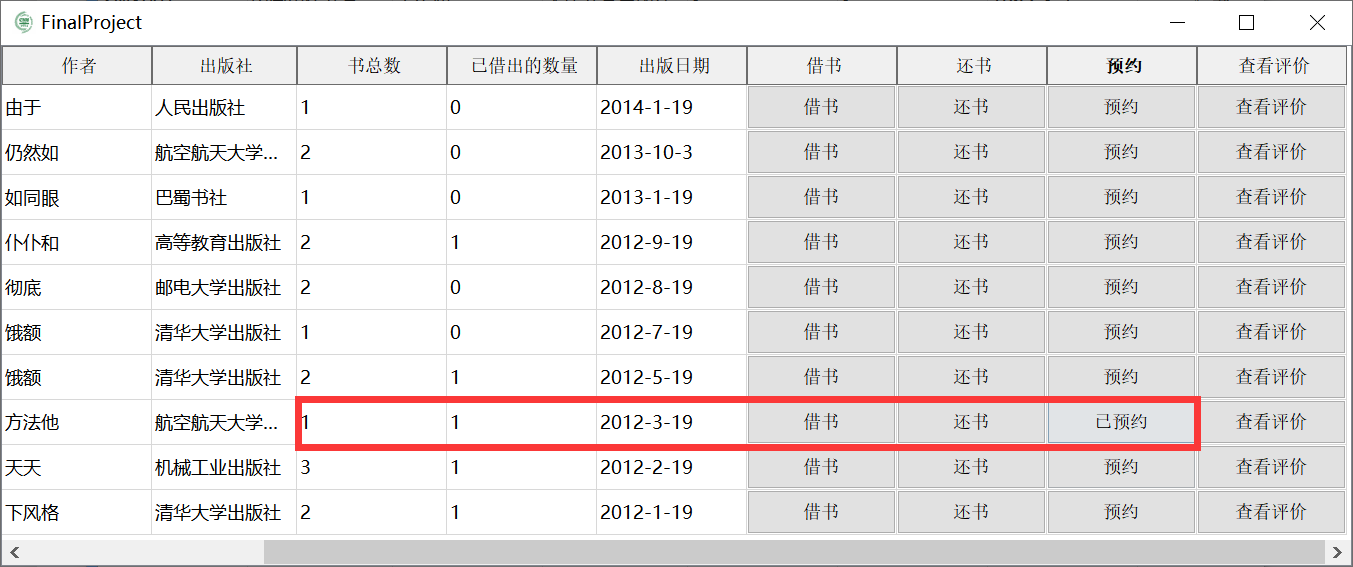


图 39 预约成功状态更改

1. 预约有库存的图书，直接借书

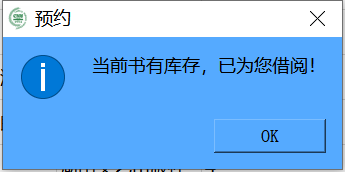


图 40 预约-有库存直接借阅

1. 查看预约记录，取消预约

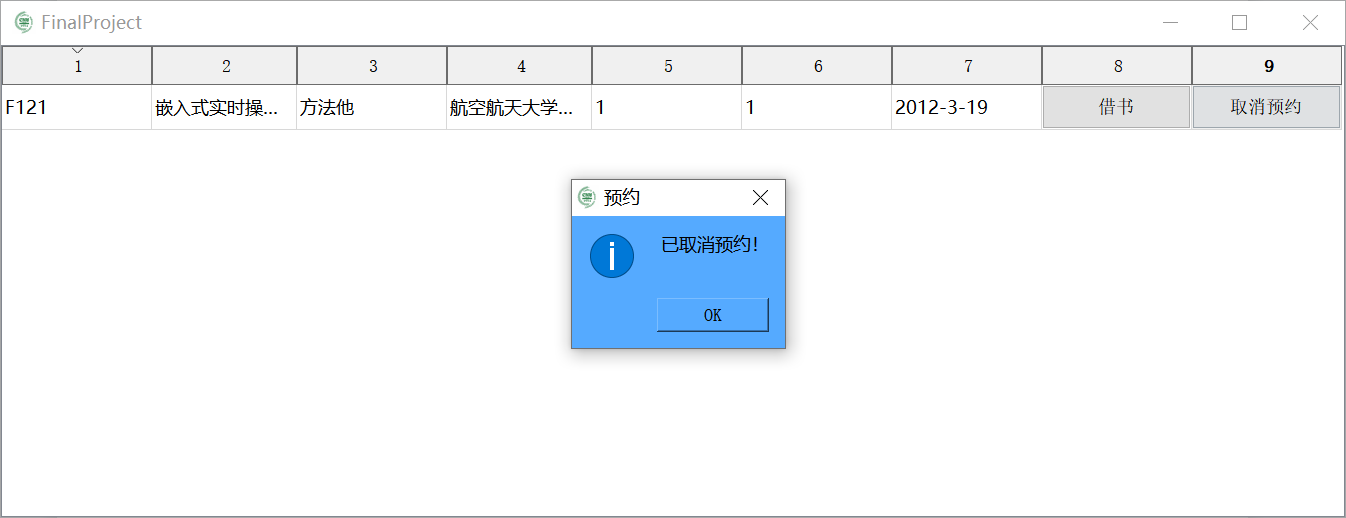


图 41 取消预约

1. 还书
2. 在当前已借列表点击还书直接还书

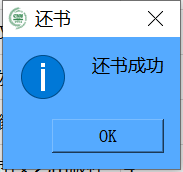


图 42 还书成功

1. 还书成功后出现评价窗口

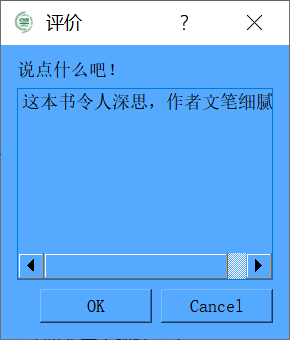


图 43 还书评价

1. 还书后当前已借记录删除，可在借阅记录查看

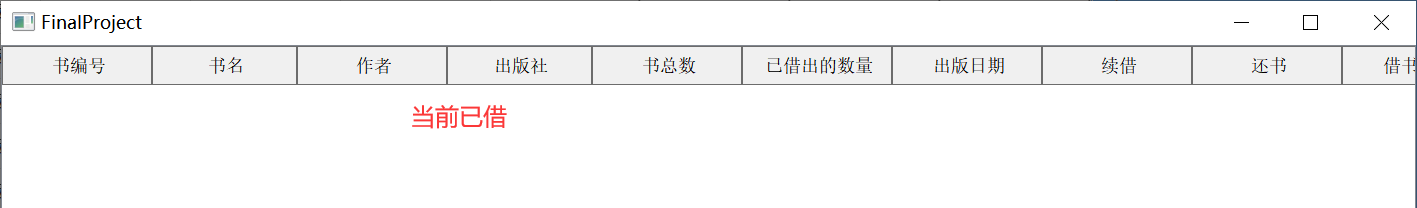




图 44 借阅记录

1. 查看评价



图 45 查看书籍评价

1. 修改密码

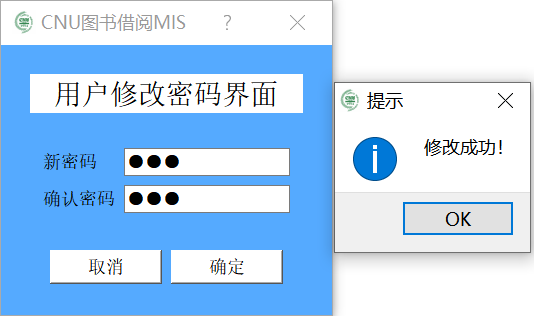


图 46 修改密码成功

1. 切换账号



图 47 切换账号

1. 管理员-修改用户信息



图 48 管理员-修改用户信息

1. 管理员-删除用户信息



图 49 管理员-删除用户信息

1. 管理员-增加一个用户

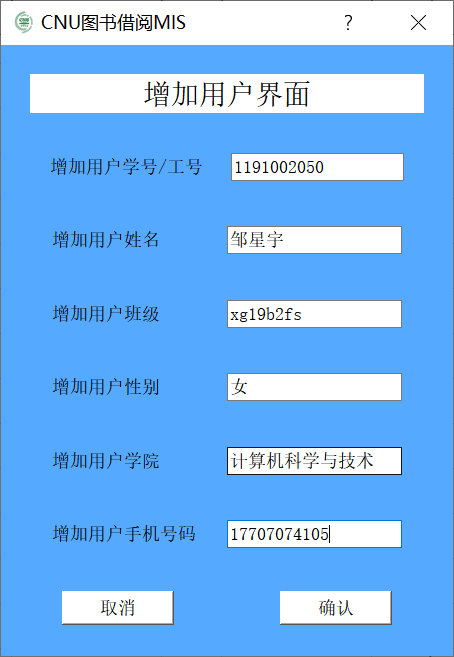


图 50 管理员-增加一个用户

1. 管理员-批量增加用户

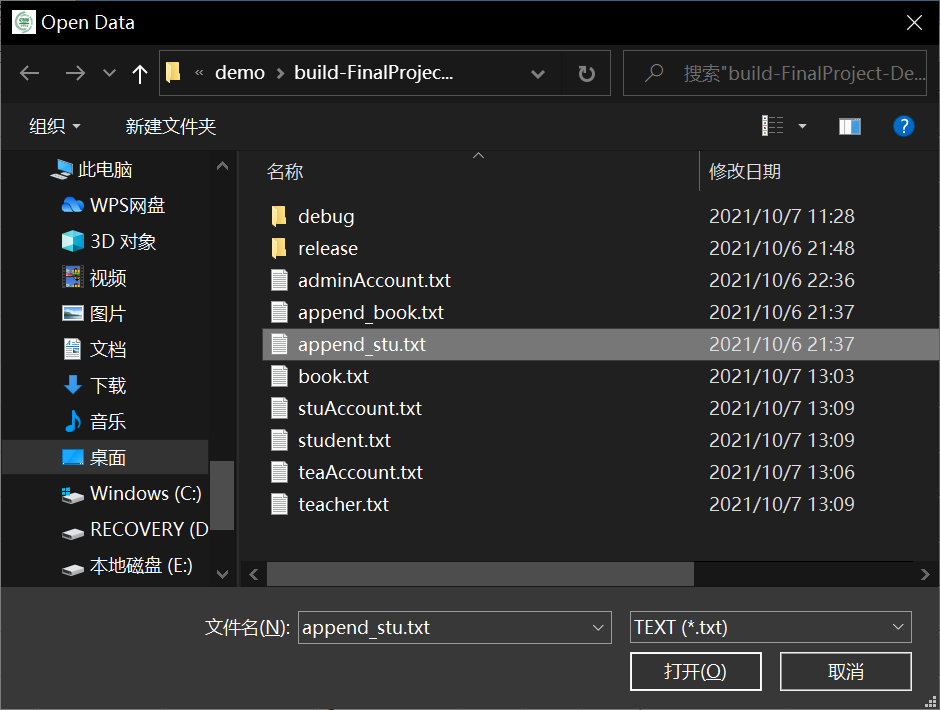


图 51 管理员-批量增加用户

1. 查找用户

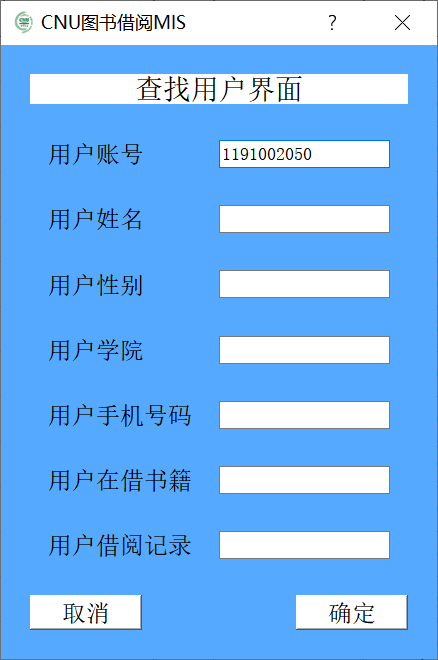


图 52 查找用户界面



图 53 查找成功显示用户信息

1. 统计
2. 借阅总排行榜

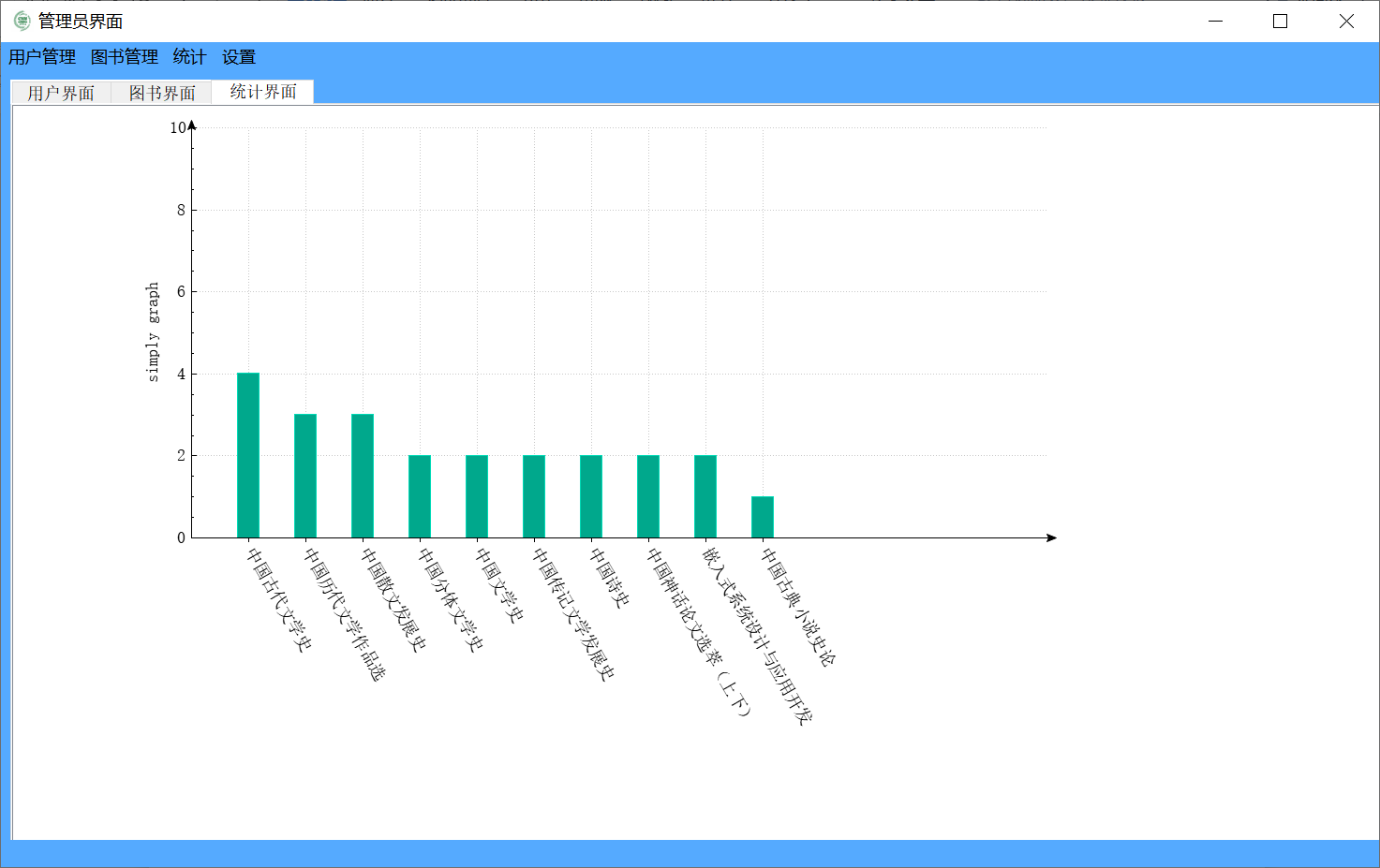


图 54 统计-借阅总排榜

1. 学院借阅排行榜

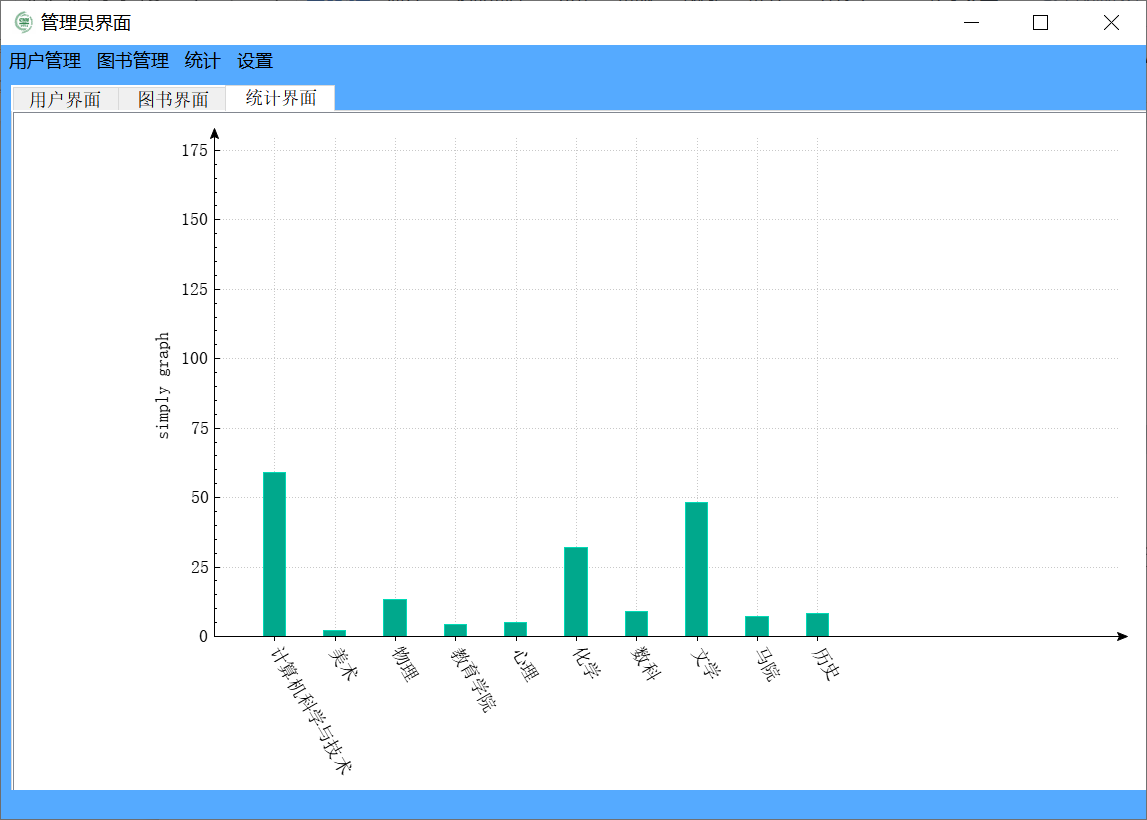


图 55 学院借阅排行榜

1. 男女借阅量对比

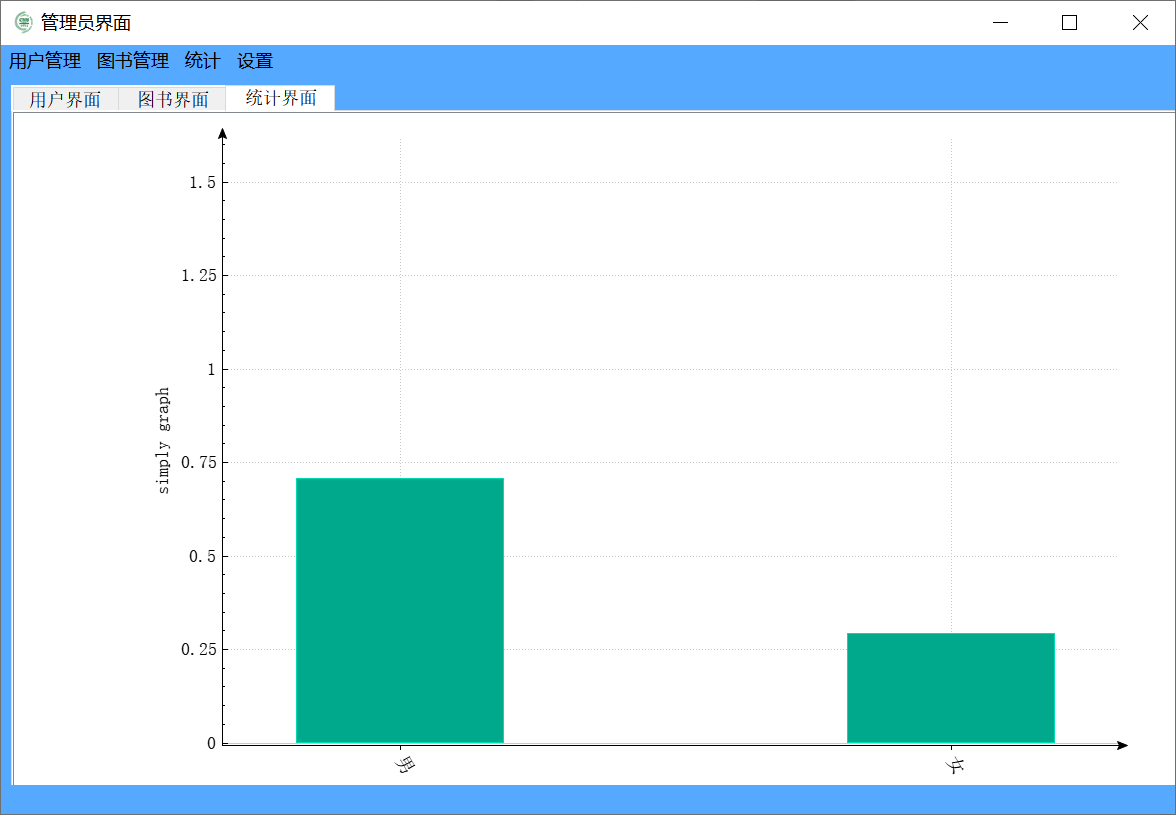


图 56 男女借阅量对比

1. 图片导出

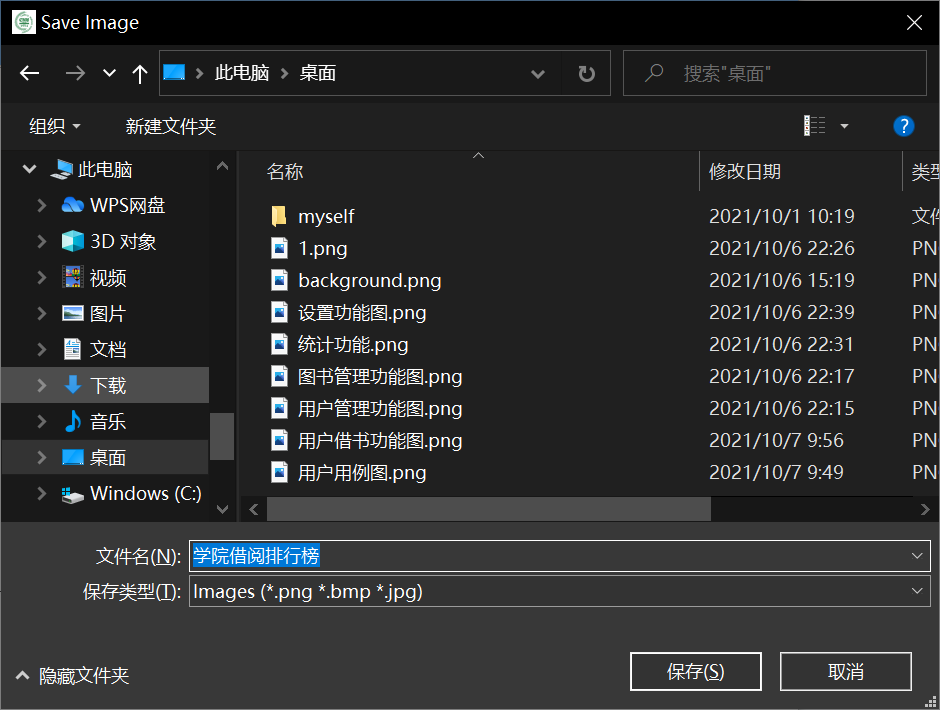


图 57 图片导出-选择保存路径

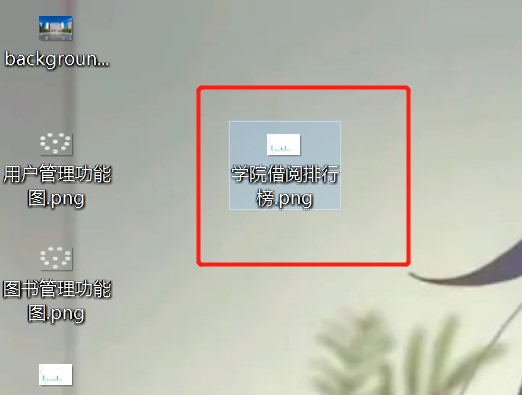
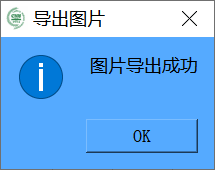


图 58 图片导出-保存到本地桌面

## CPU占用测试

在使用英特尔酷睿i7-9750H（基本频率2.60GHz，睿频4.5GHz，线程数12，内核数6，详情[英特尔® 酷睿™ i7-9750H 处理器（12M 高速缓存，高达 4.50 GHz） 产品规范 (intel.com)](https://ark.intel.com/content/www/cn/zh/ark/products/191045/intel-core-i7-9750h-processor-12m-cache-up-to-4-50-ghz.html)）同时使用系统“任务管理器”监控运行，《简单图书借阅信息管理系统》，在执行各项操作时，查看“任务管理器”所检测到的CPU占用状态的变化和瞬间峰值：检测结果《简单图书借阅信息管理系统》的各项功能运行时，CPU占用的平均值在2.5%上下波动，基本处于2.5%以下，最大峰值不超过6%，结果表明该软件在运行过程中并未产生不合理CPU占用。