

华中科技大学网络空间安全学院

程序设计综合课程设计

报告

题目： 校园卡-食堂管理系统

班 级: 信息安全 2004 班

学 号: U202011061

姓 名: 甘戈岑

成 绩:

指导教师: 郎量

完成日期: 2022 年 3 月 9 日



目录

一、系统需求分析.....	1
二、总体设计.....	2
三、数据结构设计.....	4
五、系统实现.....	15
六、运行测试与结果分析.....	16
七、总结.....	20
八、参考文献.....	21



一、系统需求分析

实现一个校园卡-食堂管理系统兼顾单项操作和批量操作，具体要求如下：

- 1) 校园卡操作：开户、发卡、挂失及解挂、补卡（含多次补卡）和充值；
- 2) 食堂消费：构建模拟的校园卡在食堂各窗口消费的业务操作，包含消费、当餐超额验证消费密码等处理；
- 3) 汇总操作：构建模拟定期将分别按各窗口记录的消费记录进行合并汇总、查询分析等操作；
- 4) 日志操作：构建各类操作日志——操作类型、操作时间、操作对象、操作结果、操作前后数据变化、操作成功与否等信息，便于回溯查询，也是用于检查当前操作是否成功和正确的依据。
- 5) 数据挖掘：依据汇总的消费数据来分析同学们的消费习惯等规律性信息；
- 6) 校验码：对消费记录增加校验数据项，用来验证本条数据和总日志文件没有被篡改，以及数据记录没有伪造增加或正常记录没有被删除。
- 7) 批量操作：实现批量操作的接口，可以合理处理字符串，激活各类操作，符合时间顺序，生成操作日志。



二、总体设计

- 1) 总体设计：基于 Qt 的前后端一体化特性，制作 GUI 和用户进行交互，接收用户的输入等操作载入后端数据域处理、判断、储存进而形成“前端-后端-前端…”的持续性交互模式。
- 2) 校园卡管理模块：
 1. 前端设计学生列表以及学生个人信息页面，提供按钮搜索框等交互控件，以及图标头像等美化控件；
 2. 后端设计学生类和学生列表管理类来处理校园卡管理模块，账户管理和饭卡管理集成于一个类中，并加以不同的权限限制，同时生成日志并将日志信息输送至表格模型可视化。
- 3) 食堂消费模块：
 1. 前端设计食堂信息列表以及显示食堂窗口编号和地址号，提供自定义消费对话框，消费日志陈列框，按钮搜索框等交互控件，以及图标头像等美化控件；
 2. 后端设计食堂窗口类以及食堂窗口列表管理类来处理食堂消费管理模块，对于食堂窗口记录当日消费总金额以及消费总次数，对于消费者消费交互窗口则设置卡号查询，余额查询，消费时间设置（模拟），消费金额，并辅助以 20 元限额和密码验证操作，同时生成日志显示在日志列表中。
- 4) 批量处理模块：
 1. 前端设置系列按钮，以及可视化 Gantt 图将卡片操作和消费模拟的时间轴表示出来，清晰明了。
 2. 后端调度快速排序，k-路归并排序，堆排序等排序算法和读取文件批量操作的算法；生成日志和相应的校验码，额外生成批量操作的记录日志和详细操作日志；生成总消费日志。
- 5) 数据挖掘模块：
 1. 前端输入学号，进行数据挖掘；同时有显示排列好的相关性得分情况的表单。



2. 后端通过窗口距离和相对距离赋以不同的分值权重，遍历总消费日志的全部内容并通过堆排序进行相应的打分输出。

6) 用户界面：

1. 功能：对于交互性的控件分别有静态控件和 Model-based Item View 模型对批量数据进行管理显示等，对于整体布局主要以局部布局方案（弹簧）和全局布局方案（Layouts）。
2. 美化：主要有内置 QSS 语句进行美化以及自定义控件（按钮，下拉条，logo 等）。

系统整体设计图和时间轴标识图如下所示。（如图 2-1，2-2 所示）

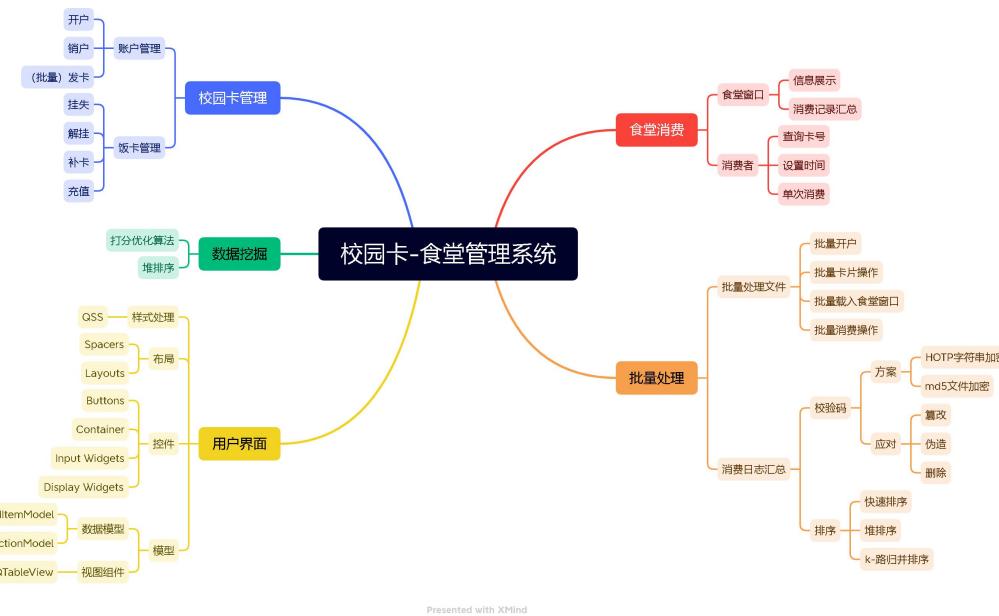


Figure 2.1 校园卡-食堂管理系统框架图

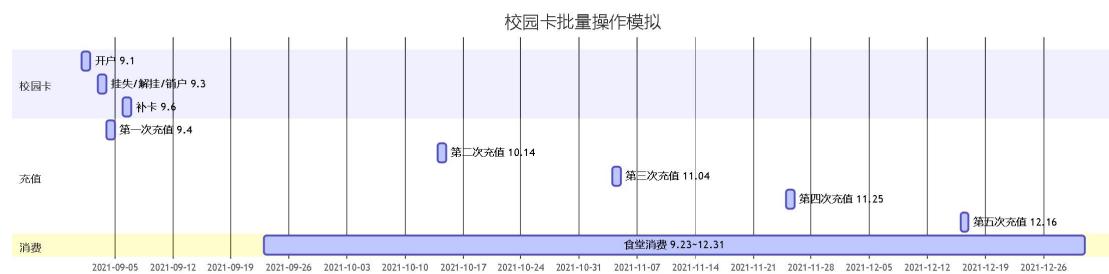


Figure 2.2 账户卡片操作-消费模拟时间图



三、数据结构设计

基于需求，我设计封装了以下若干类来控制管理数据之间的交互，关于页面我也设计了一些自定义的控件(比如自定义的圆形按钮，方形按钮等)，考虑到篇幅有限，此处不多罗列。

I. 单个校园卡

Figure3.1.1 Campus Card Properties

Name	Type	Description
studentName	QString	学生姓名
studentNumber	QString	学号
effectiveDate	QDateTime	有效日期
cardNumber	QStringList	卡号列表(尾元素为当前卡号)
accountState	Bool	账户状态
isDistributed	Bool	是否初始化发卡
cardState	Bool	当前卡号状态
balance	double	余额

Figure3.1.2 Campus Card Functions

Name	Input Parameter	Return Type	Description
CampusCard	stuNumber,Name...	CampusCard&	有参构造函数
Operator <	CampusCard&	Bool	重载运算符(按学号比较)
cardReportLoss	None	Bool	挂失
cardDistributed	serialNumber	QString	通过流水号发卡
cardUncouple	None	Bool	解挂
cardReissue	serialNumber	Bool	补卡
cardRecharge	money	Bool	充值

II. 校园卡管理

Figure3.2.1 Campus-Card Management Properties

Name	Type	Description
campusCardList	QVector<CampusCard>	储存校园卡的向量表
mapStuNumberToCardNumber	QMap<QString,int>	学号到表索引的映射
mapCanteenNumberToCardNumber	QMap<QString,QPair<int,bool>>	消费卡号到表索引,卡号有效性的映射
log	QVector<OperationLog>	日志表
stuIssuedNumber	int	已经发卡学生人数(决定流水号)
stuNumber	int	账户数目

Figure3.2.2 Campus-Card Management Functions



Name	Input Parameter	Return Type	Description
addNewAccount	stuNumber,stuName	Bool	添加新用户对外接口
deleteExistedAccount	stuNumber	Bool	销户
queryCampusCard	stuNumber	int	查询用户返回表索引
batch_distributeCard	None	void	批量发卡接口
reissueCard	stuNumber	Bool	补卡对外接口
rechargeCard	cardNumber,money	bool	充值

III. 单个食堂窗口

Figure3.3.1 Canteen Window Properties

Name	Type	Description
winNumber	QString	窗口号
locationNumber	QString	窗口位置
tempLog	QVector<OperationLog>	当前消费日志列表
CurrentDate	QDate	消费日期
CurrentTime	QTime	消费时间

Figure3.3.2 Canteen-Window Management Functions

Name	Input Parameter	Return Type	Description
consume	Result,time,cardNumber,money	void	消费
addConsumptionLog	Operation*	void	消费日志列表

IV. 食堂窗口管理

Figure3.4.1 Canteen-Window Management Properties

Name	Type	Description
windowNumber	int	窗口数目
CanteenWindowList	QVector<CanteenWindow>	食堂窗口列表
totalLog	QVector<OperationLog>	总操作日志

Figure3.4.2 Canteen-Window Management Functions

Name	Input Parameter	Return Type	Description
addNewCanteenWindow	winNumber,locNumber	void	添加新的食堂窗口
addNewCanteenLog	Operation*log	void	添加新日志

V. 堆排序所用到的自定义优先队列

Figure3.5.1 My Priority Queue Properties

Name	Type	Description
N	int	自定义优先队列长度
data;	QVector<QPair<QString,int>>	优先队列容器(index:1~N)

Figure3.5.2 My Priority Queue Functions



Name	Input Parameter	Return Type	Description
StringHeap	None	StringHeap&	默认构造函数
isEmpty	None	Bool	判空
top	None	QPair<QString,int>	获取最小值(小根堆)
push	QPair<QString,int> q	Void	插入数据
pop	None	Void	删除最小值
sink	int pos	Void	下沉(堆平衡)
swim	int pos	Void	上浮(堆平衡)
swap	int pos1,int pos2	Void	对内元素互换

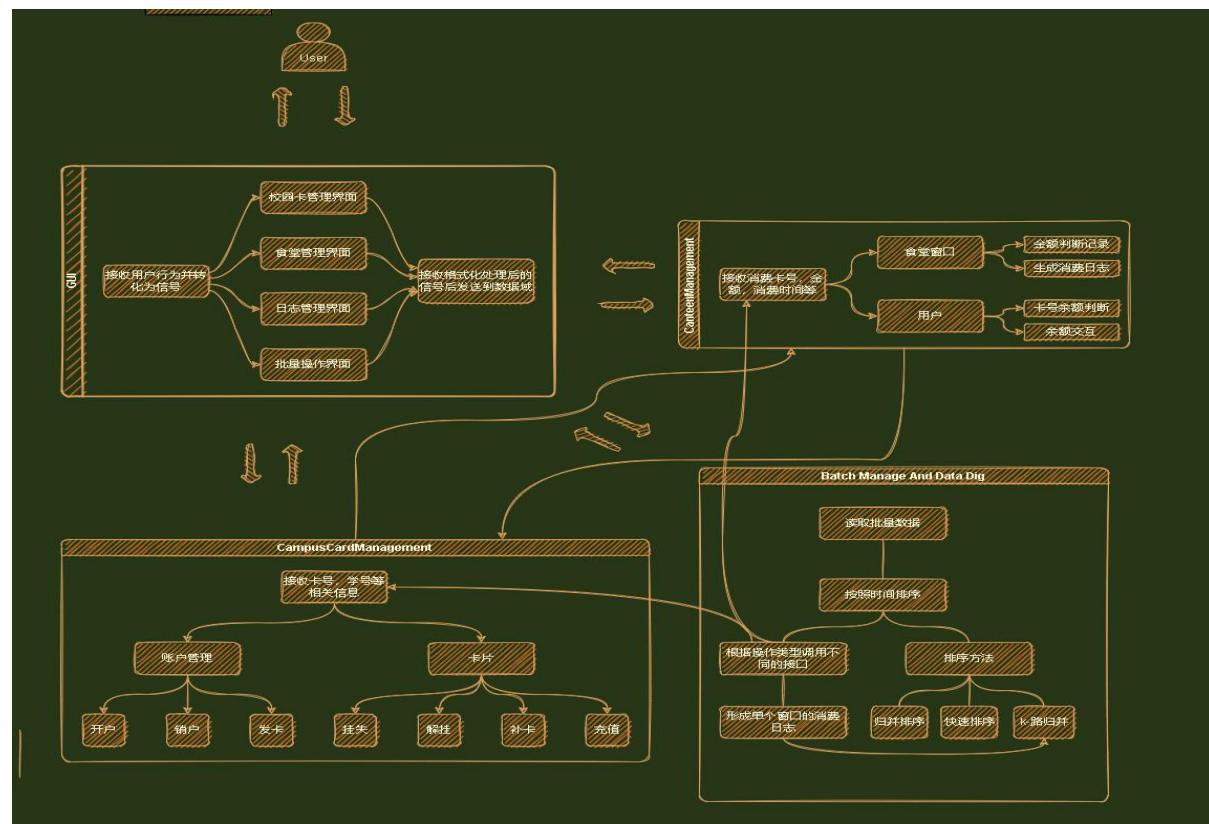
VI. 便于陈列可视化而设计的日志类

Figure3.6 Operation Log Properties

Name	Type	Description
nameLocation	int	0~8->定位到相应的操作名
stuName	QString	学生姓名
stuNumber	QString	学生学号
opTime	QDateTime	操作时间
opResult	bool	操作结果
opName	QStringList	操作名(常量列表, 与 nameLoc 对应)
canteenNumber	QString	消费卡号
windowNumber	QString	消费窗口号

VII. 数据结构交互应用流程图如下所示。

Figure3.7 Flow Chart on Interaction among Data Structures





四、详细设计

I. 校园卡操作

a) 开户：简单的顺序操作。

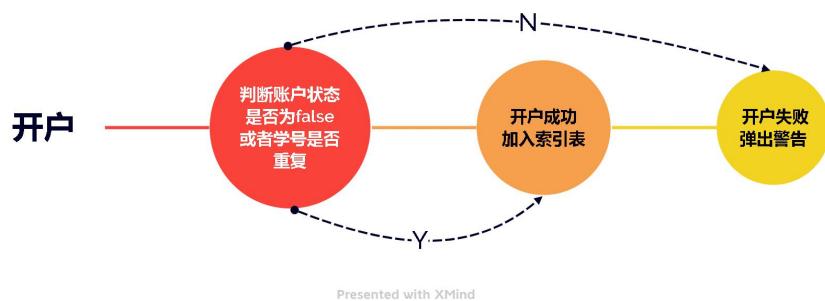


Figure4.1 Flow Chart of Opening Account

b) 销户：同理开户，此处不多赘述。

c) 发卡：根据流水号生成校验码

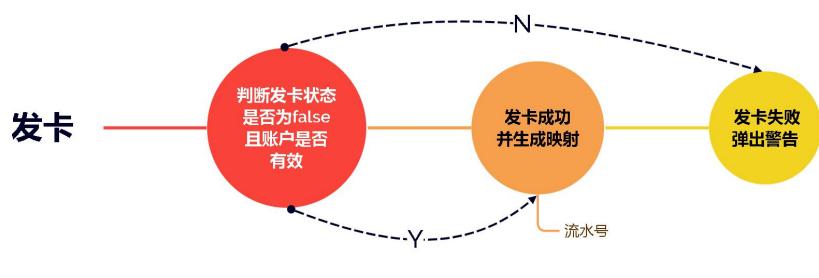


Figure4.2 Flow Chart of Distributing Card Number

d) 挂失：如下所示

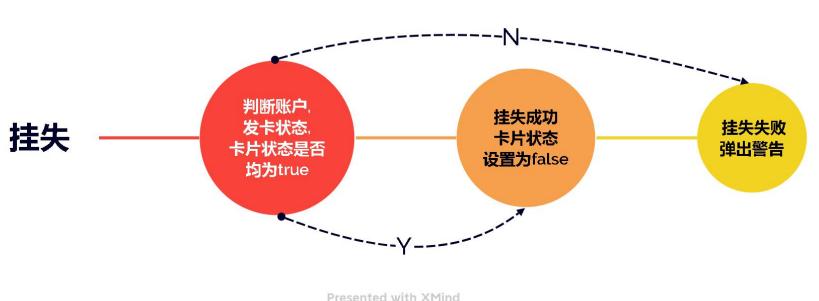


Figure4.3 Flow Chart of Reporting Loss

e) 解挂：同理挂失

f) 补卡：判断账户有效，已经初始化发过卡并且卡片状态为 false 的前提下同理发卡。



g) 充值：需要判断账户状态以及充值金额是否超出限额。

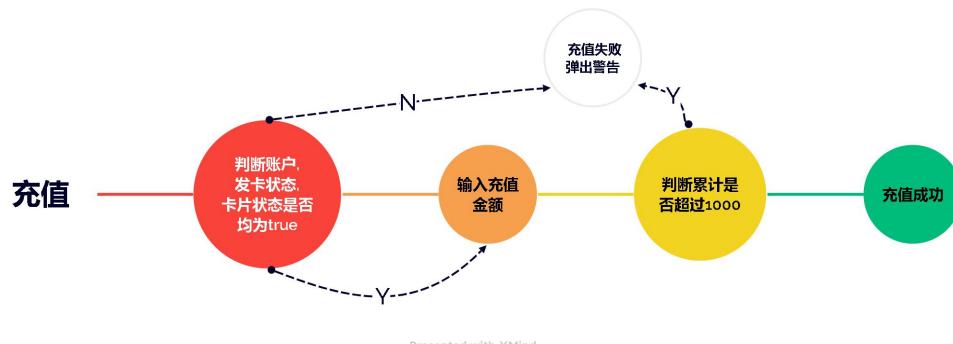
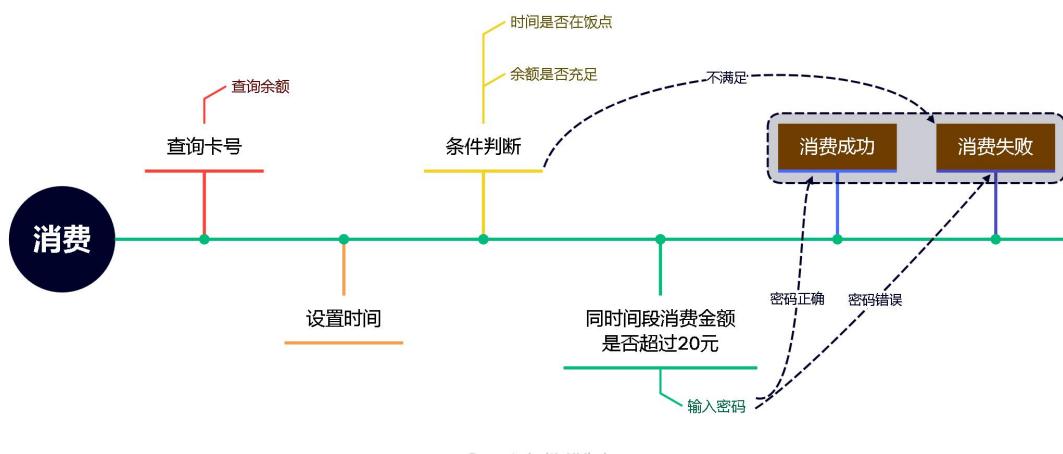


Figure4.4 Flow Chart of Recharging

II. 食堂消费

- a) 消费：主要是关注各个变量的限制以及密码输入的判断和窗口通信的交互。
b) 流程图如下

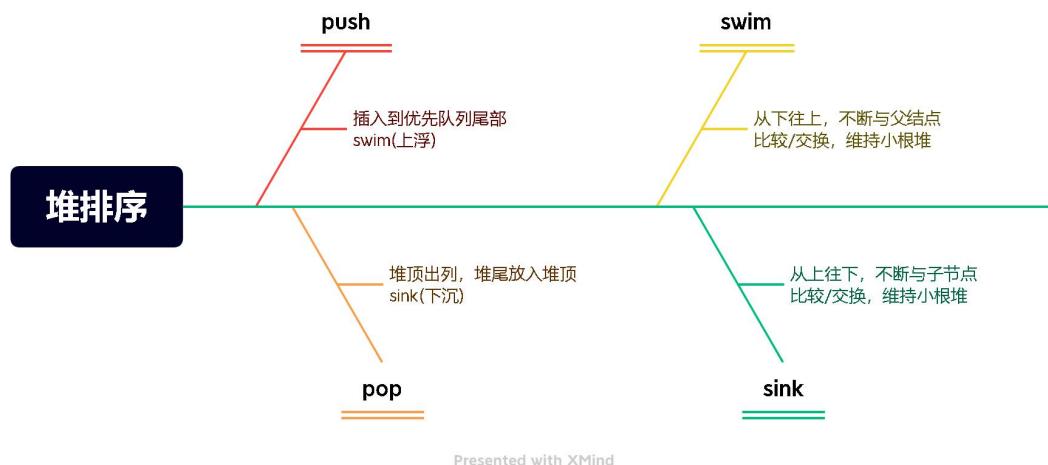


III. 批量处理

- a) 批量开户：此功能较易实现，只需读入并调用开户接口并生成日志即可。
b) 批量载入食堂窗口：同理(a)。
c) 批量卡片操作：按时间读取后调用挂失，解挂，补卡，充值等接口，接口中自有一些前提判断的流程来决定该操作的成败。
d) 批量充值消费：编写两个参数为起止时间的函数，在该时间区间内读取文件触发充值或者消费操作，并对应每个食堂窗口生成单个日志。对于文件读写采取遍历的方式。因此当总操作触发时系统一次调用不同时间段的充值和消费函数(附带有文件读写操作)。



- e) 消费数据合并：57路合并，合并时采取归并排序，由于Qt不支持优先队列，因此我自己写了一个优先队列(动态小根堆)。如此一来只需要初次将57个文件首行入列，然后每次出列得到该条消费日志所在文件，其出列的同时如果相应文件未读完则读取其下一条入列。



IV. 搜索

a) 正则表达式

- i. 策略：输入框每变化一次都遍历表全部信息并只显示和正则表达式匹配的项。
- ii. “.”：单个不确定字符
- iii. “.{1,}”至少一个不确定字符
- iv. “.{2,}”多个不确定字符
- v. 支持搜索方式：
 1. 顺序输入过程中不断变化：202014...、曹...
 2. 正则表达式化输入：2*5、刘?华
 3. 完全输入匹配

b) 参考代码

```
1. ****  
2. * @Name : 查询槽函数  
3. * @Function : 根据搜索框内容动态变化列表  
4. * @Input : arg1(搜索框内字符串)  
5. * @return : void
```



```
6. * @version : 1.3
7. ****
8. void Widget::on_ldt_searchTextChanged(const QString &arg1)
9. {
10.     if(arg1=="")
11.     {
12.         for(int i=0;i<ui->tableViewStudents->model()->rowCount();i++)
13.             ui->tableViewStudents->setRowHidden(i,false); //设置为均可
见
14.     }
15.     else
16.     {
17.         //正则表达式格式
18.         QString stdFormat = "";
19.         QString stdFormat2 = "";
20.         //标志变量
21.         bool flag = true;
22.         //全局匹配
23.         for(int i=0;i<arg1.size();i++)
24.         {
25.             if(arg1[i]=='?')
26.             {
27.                 stdFormat += "."; //单个不确定字符
28.                 flag=false;
29.             }
30.             else if(arg1[i]=='*')
31.             {
32.                 stdFormat += ".{2,}"; //多个不确定字符
33.                 flag=false;
34.             }
35.         }
36.     }
37. }
```



```
36.             stdFormat += arg1[i];//默认匹配
37.         }
38.         if(flag)
39.             stdFormat2 = arg1 + ".{1,}";//后面至少一个
40.
41.         //姓名和学号同时匹配
42.         for(int i=0;i<ui->tableViewStudents->model()->rowCount();i++)
43.
44.     {
45.         //设置不可见
46.         ui->tableViewStudents->setRowHidden(i,true);
47.         //学号
48.         QString stuNumber="";
49.         //姓名
50.         QString stuName="";
51.         //提取学生信息
52.         QAbstractItemModel *model=ui->tableViewStudents->mode
53.         l();
54.         //坐标
55.         QModelIndex index1,index2;
56.         //获取学号和姓名的坐标
57.         index1=model->index(i,0),index2=model->index(i,1);
58.         //学号
59.         stuNumber+=model->data(index1).toString();
60.         //姓名
61.         stuName+=model->data(index2).toString();
62.         //全局匹配
63.         QRegExp res(stdFormat);
64.         //部分匹配
65.         QRegExp resPart(stdFormat2);
66.         //学号匹配
```



```
66.         bool match_stuNumber=res.exactMatch(stuNumber);
67.         bool match1=resPart.exactMatch(stuNumber);
68.         //姓名匹配
69.         bool match_stuName=res.exactMatch(stuName);
70.         //姓名匹配
71.         bool match2=resPart.exactMatch(stuName);
72.         //匹配
73.         if(match_stuName||match_stuNumber||match1||match2)
74.             ui->tableViewStudents->setRowHidden(i, false);
75.     }
76. }
77. }
```

c)

V. 数据挖掘:根据总消费日志, 在 8120*8120 的二维数组中进行打分积累

a) 横向打分

- i. 同窗口: 3',
- ii. 相邻窗口: 2',
- iii. 相间窗口: 1',
- iv. 其余: 0'

b) 纵向打分

- i. 距离 d 得分:(10-d)'(d>=0&&d<=10)
- ii. 打分时只考虑其后的同学不考虑其前的同学, 因为其前的同学打分时已考虑过自身。

c) 边界处理

- i. 时间段看小时数相差超过 2 即停止统计。

d) 最终排序

- i. 根据快速排序(利用 Qt 内置的 qSort 进行快排)。

e) 打分函数代码:

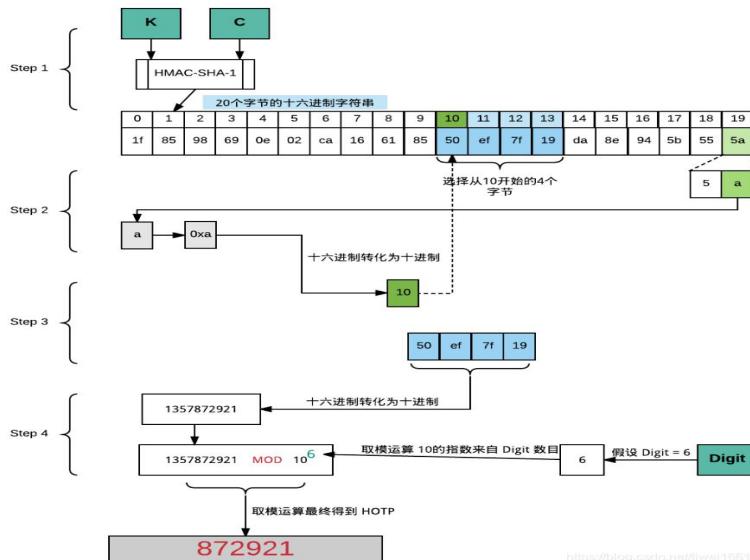
```
1. ****
2. * @Name: 打分函数
3. * @Input: 距离, 两个消费记录的窗口号
```



```
4. * @return: 所得分数
5. * @version: 1.3
6. ****
7. int Widget::Score(int distance, int winId_1, int winId_2)
8. {
9.     int s1=10 - distance; //日志记录距离分
10.    int s2;//窗口距离分
11.    int dis2 = qAbs(winId_1 - winId_2); //求绝对值
12.    switch (dis2) {
13.        case 0:
14.            s2=3; break;
15.        case 1:
16.            s2=2; break;
17.        case 2:
18.            s2=1; break;
19.        default:
20.            s2=0; break;
21.    }
22.    return s1*s2; //返回打分
23. }
```

VI. 日志校验

a) HOTP 算法





代码：

```
1. ****
2. * @Function : 生成 HOTP 校验码
3. * @Input : key(关键字), baseString(流水号)
4. * @return : QByteArray
5. * @version : 1.3
6. ****
7. QByteArray Widget::hmacSha1(QByteArray key, QByteArray baseString)
8. {
9.     int blockSize = 64;
10.    if (key.length() > blockSize) {
11.        key = QCryptographicHash::hash(key, QCryptographicHash::Sha1)
12.        ;
13.        QByteArray innerPadding(blockSize, char(0x36));
14.        QByteArray outerPadding(blockSize, char(0x5c));
15.        for (int i = 0; i < key.length(); i++) {
16.            innerPadding[i] = innerPadding[i] ^ key.at(i);
17.            outerPadding[i] = outerPadding[i] ^ key.at(i);
18.        }
19.        QByteArray total = outerPadding;
20.        QByteArray part = innerPadding;
21.        part.append(baseString);
22.        total.append(QCryptographicHash::hash(part, QCryptographicHash::
23.            Sha1));
23.        QByteArray hashed = QCryptographicHash::hash(total, QCryptograph
24.            icHash::Sha1);
24.        QByteArray arrayFromHexString = QByteArray::fromHex(hashed.toHex
25.            ());
25.        return arrayFromHexString;
26. }
```



五、系统实现

(一) 开发环境 IDE : Qt Creator 4.3.1(based on Qt 5.9.1)

(二) 支持包 : MSVC 2015, 32 bit

(三) 注释格式 :

a) 函数注释 :

```
1. /*****  
2. * @function : explanation on this function  
3. * @Input : ...  
4. * @return : ...  
5. * @version : About the version of APP  
6. *****/
```

b) 段注释 :

```
1. /*****设置 TableView 的样式*****/  
2. //Where lies the functions that setting the style of tableView  
3. *****/
```

(四) 具体实现 :

参见 GitHub repository :

<https://github.com/Ganliber/Campus-Card---Canteen-Management-System>



六、运行测试与结果分析

1) 程序初始化界面



Figure 6.1.1 Account



Figure 6.1.2 Canteen



Figure 6.1.3 Log



Figure 6.1.4 Batch Process

2) 账户管理功能:挂失、解挂、补卡、充值以及条件限制



Figure 6.2.1 Report Loss



Figure 6.2.2 Uncouple



Figure 6.2.3 Report Loss



Figure 6.2.4 ReCharge



3) 搜索功能

基于动态输入的搜索功能见 Figure 6.3.1 Student Number(I) 和 Figure 6.3.2 Student Name(I), 基于正则表达式的搜索功能见 Figure 6.3.3 Student Number(II) 与 Figure 6.3.4 Student Name(II)。

学生列表				
学号	姓名	账户	校园卡	
200	邱半雪	正常	已发卡	
201	吕半雪	正常	已发卡	
202	曹梦蓝	正常	已发卡	
203	唐歌	正常	已发卡	
204	邵梦安	正常	已发卡	
205	赵宛凝	正常	已发卡	
206	叶婉	正常	已发卡	
207	邵夕	正常	已发卡	
208	丁南风	正常	已发卡	
209	贾翠曼	正常	已发卡	
210	吴瀛	正常	已发卡	

Figure 6.3.1 Student Number(I)

学生列表				
学号	姓名	账户	校园卡	
9	宋曼	正常	已发卡	
247	宋如	正常	已发卡	
418	宋妙彤	正常	已发卡	
440	宋红	正常	已发卡	
679	宋媛	正常	已发卡	
765	宋南风	正常	已发卡	
879	宋巧荷	正常	已发卡	
999	宋思雁	正常	已发卡	
1109	宋灵蕊	正常	已发卡	
1255	宋沁媛	正常	已发卡	
1315	宋诗蕊	正常	已发卡	
1342	宋煊	正常	已发卡	
1840	宋元枫	正常	已发卡	
2015	宋紫山	正常	已发卡	

Figure 6.3.2 Student Name(I)

学生列表				
学号	姓名	账户	校园卡	
1	秦晓烟	正常	已发卡	
18	卢翼青	正常	已发卡	
25	蔡碧曼	正常	已发卡	
32	方楠	正常	已发卡	
49	顾桔	正常	已发卡	
56	钱映寒	正常	已发卡	
63	齐冷梅	正常	已发卡	
70	范绿春	正常	已发卡	
87	陈傲晴	正常	已发卡	
94	冯凝梅	正常	已发卡	
102	戴璐丝	正常	已发卡	
115	廖青筠	正常	已发卡	
122	杜从梦	正常	已发卡	
139	彭辰	正常	已发卡	

Figure 6.3.3 Student Number(II)

学生列表				
学号	姓名	账户	校园卡	
765	宋南风	正常	已发卡	
5031	宋访风	正常	已发卡	

Figure 6.3.4 Student Name(II)

4) 食堂消费功能

食堂消费功能界面可以自定义消费模拟场景，对用户而言具有较强的灵活度，同时具有单个时间段消费金额超过 20 输入密码的功能以及单个窗口日志功能。

Dialog

单次窗口消费

当前消费时间: 15时59分3秒

查询卡号: 3123659

当前消费时间: 0:00

消费金额: 0.00

消费卡号: 3123659 当前余额: 417.15

当前时段累计消费余额: 0

Figure 6.4.1 Consume Page

Dialog

单次窗口消费

当前消费时间: 12时10分0秒

查询卡号: 3123659

当前消费时间: 12:10

消费金额: 7.00

消费卡号: 3123659 当前余额: 410.15

当前时段累计消费余额: 7

Figure 6.4.2 Consume Result(I)



Figure 6.4.2 Consume Page



Figure 6.4.3 Single Window Log

5) 归并排序和堆排序功能

图 Figure 6.5.2 展示了经过基于堆排序实现的 57 路归并排序所得总日志文件，时间复杂度为 $O(n \log n)$ ，图 Figure 6.5.3 展示了处理两百万条消费记录所耗时间，还是很快的。

W1.txt	2022/3/18 17:49	中文-中文档
W2.txt	2022/3/18 17:49	文本文档
W3.txt	2022/3/18 17:49	文本文档
W4.txt	2022/3/18 17:49	文本文档
W5.txt	2022/3/18 17:49	文本文档
W6.txt	2022/3/18 17:49	文本文档
W7.txt	2022/3/18 17:49	文本文档
W8.txt	2022/3/18 17:49	文本文档
W9.txt	2022/3/18 17:49	文本文档
W10.txt	2022/3/18 17:49	文本文档
W11.txt	2022/3/18 17:49	文本文档
W12.txt	2022/3/18 17:49	文本文档
W13.txt	2022/3/18 17:49	文本文档
W14.txt	2022/3/18 17:49	文本文档
W15.txt	2022/3/18 17:49	文本文档
W16.txt	2022/3/18 17:49	文本文档
W17.txt	2022/3/18 17:49	文本文档
W18.txt	2022/3/18 17:49	文本文档
W19.txt	2022/3/18 17:49	文本文档
W20.txt	2022/3/18 17:49	文本文档

Figure 6.5.1 Single Log

归并排序(堆)用时 0.630383 min

Figure 6.5.3 Time for Merge Sort

6) 操作日志

操作日志展示了卡片操作，单个单日窗口日志，专门针对批量操作的分析日志，全部日志等多种形式的日志并佐以校验码。

交园卡						
	时间	操作名	姓名	学号	涉及卡号	结果
23021	2021-10-14	充值	乔蔚兰	2020672065	3180467	成功
23022	2021-10-14	充值	曹殊	2020672071	3180476	失败
23023	2021-10-14	充值	何青香	2020672084	3180485	成功
23024	2021-10-14	充值	孟物	2020672097	3180494	失败
23025	2021-10-14	充值	黎书怡	2020672109	3180502	成功
23026	2021-10-14	充值	崔琳滔	2020680014	3180511	成功
23027	2021-10-14	充值	邵典	2020680027	3180520	失败

Figure 6.6.1 Card Log

当日窗口消费日志			
消费卡号	余额变化	当前余额	消费结果
1 3123498	-2	163	成功
2 3123488	-2	161	成功
3 3123488	-19	165	成功
4 3123488	-8	192	成功
5 3123488	-8	184	成功



Figure 6.6.2 Single Window Log

批量操作日志

批量操作

	操作类型	数据项数	成功项数	异常项数	
1	开户	8120	8120	0	2021-09-24 07:00:53:700,1.6324,09,3186704,754.15(-3.35),消费成功,146925
2	卡片操作	9218	9208	10	2021-09-24 07:01:04:440,1.5525,09,3178712,605.00(-8.95),消费成功,524959
3	消费	434398	406935	27463	2021-09-24 07:01:20:210,1.7936,09,3202822,803.55(-6.50),消费成功,679334
4	充值	8120	5752	2368	2021-09-24 07:01:33:30,1.3221,09,315156,72,800.55(-3.35),消费成功,602728
5	消费	434397	349002	85395	2021-09-24 07:02:00:670,1.6098,09,3184445,532.70(-5.80),消费成功,123172
6	充值	8120	5060	3060	2021-09-24 07:02:23:330,1.7108,09,3174545,798.25(-4.75),消费成功,259642
7	消费	434395	345822	88573	2021-09-24 07:02:35:590,1.1067,09,3134134,788.75(-8.25),消费成功,76834
8	充值	8120	5737	2383	2021-09-24 07:02:55:320,1.4045,09,3163916,856.85(-3.35),消费成功,408136
9	消费	434400	359196	75204	2021-09-24 07:03:06:490,1.4523,09,3168696,627.20(-9.30),消费成功,129670
10	充值	8120	5659	2461	2021-09-24 07:03:19:120,1.397,09,3127439,872.70(-4.40),消费成功,799704
11	消费	330967	284301	46666	2021-09-24 07:03:32:460,1.364,09,3159863,807.80(-7.20),消费成功,473597

Figure 6.6.3 Batch Log

Figure 6.6.4 Consuming Log

7) 校验码



Figure 6.7.1 tamper



Figure 6.7.2 delete



Figure 6.7.3 add

8) 数据挖掘

输入学号点击“找朋友”按钮即可进行打分式数据挖掘，此时对第一组数据进行验证：

2020240140, 2020331516, 2020331996, 2020610925, 2020281145, 2020470327, 202050277, 2020420971, 2020790674

结果证明完全正确，其余同理。

The screenshot shows a user interface for data mining. On the left, there are four main categories: '长户管理' (Long Household Management), '消费模拟' (Consumption Simulation), '总分析' (Overall Analysis), and '批量处理' (Batch Processing). In the center, a search bar is labeled '找朋友' (Find Friends) and contains the input '2020240140'. Below the search bar, a list of friend recommendations is displayed, each consisting of a name and a score. One specific recommendation is highlighted with a red box: '薛诗云 (2020281145) :128'. The entire list is as follows:

- 薛诗云 (2020281145) :128
- 叶醉文 (2020470327) :121
- 姚枫 (2020790674) :113
- 段珍 (2020331516) :93
- 曹梦丽 (2020331996) :82
- 于正妍 (2020610925) :64
- 丁笑天 (2020250277) :59
- 赖依 (2020420971) :46
- 钱念真 (2020700107) :30
- 邹曼荷 (2020851478) :30
- 黎怡诗 (2020561406) :29
- 周嫻 (2020800656) :27
- 黎孤 (2020770410) :26
- 朱平绿 (2020461299) :26
- 常亦寒 (2020900080) :25
- 熊以蕊 (2020170010) :25
- 蔡海萍 (2020700194) :24
- 江歌玲 (2020661635) :23
- 宋柳 (2020610251) :23
- 钱雅阳 (2020770085) :23
- 丁翠彤 (2020540837) :22
- 漕梦琪 (2020620742) :22



七、总结

通过本次工程项目我锻炼了自己的系统设计的思维并强迫自己学习并钻研了 C++ 的 GUI 编程。在此期间我撰写设计文档，需求文档，由易到难，又单次操作到批量操作，由线性编程到设计各种算法接口进行时间优化，结合任务书设计一系列针对官方给出的数据的 API，在近 4,000 行代码之间辗转反侧，最终以较好的完成度结束了本次工程经历。

我此期间由于 Qt 是一个专注于跨平台桌面软件的 IDE，我需要用类比的思想学习并熟练使用 Qt 内置的基本数据结构以及相应的可视化框架和前后端处理的思想，学到的知识包括但不限于：信号和槽这种松散耦合交互思想，基本 Widget 的使用，Layout 和 Spacer 搭配布局，重写事件和嵌入式 QSS 美化，基于数据前后端交互的可视化处理，QTableView 和 Model 的交互，优先队列的底层实现，快排，归并排序，运筹学打分模型，正则表达式……总之在类与对象这一层面的 coding 更加熟练，认识也更加深刻。

对于创新的功能，我借鉴了很多当前流行的桌面软件的前端布局，由于 C++ 过于底层，因此我最终决定自己设计，配色我借助了色卡这款 APP，图标我借助了阿里巴巴的 iconfont 开源网站的资源，期间大量查阅官方文档以及 QSS 和 CSS 的异同，不同按钮的效果借鉴了 WPS，VSCode，个人信息栏以及相关按钮借助了 Slack 等桌面软件，由于这些前端代码都没有开源，所以只能根据有限的 Qt Assistant 和官方的最新发布的 version 来自己试验。关于堆排序，由于 Qt 不支持 STL，所以我只好自己实现了一个小根堆，不过实现过好还是很有成就感的，对其理解也上了一个新的层次。

关于数据安全，我拜师室友，实现了 HOTP 动态字符串校验加密算法并加以实现，体现了信息安全专业的安全编程和安全设计的特色。

总之，此次工程项目在锻炼了我的编程功力的同时还加深了我对 OOP 的理解和应用，以及对 Software Security 的初步认识，可以说意义很大。



八、参考文献

- [1] 严蔚敏等. 数据结构(C 语言版). 清华大学出版社
- [2] Larry Nyhoff. ADTs, Data Structures, and Problem Solving with C++. Second Edition, Calvin College, 2005
- [3] 殷立峰. Qt C++ 跨平台图形界面程序设计基础. 清华大学出版社, 2014:192~197
- [4] 严蔚敏等. 数据结构题集(C 语言版). 清华大学出版社
- [6] Qt 实现 HMAC_SHA1 哈希算法 [EB/OL]. [2017-06-23].
<https://blog.csdn.net/WMX843230304WMX/article/details/73649294>.
- [8] 王维波, 栗宝鹃, 侯春望. Qt5.9 c++开发指南[M]. 第一版. 人民邮电出版社, 2018.
- [9] iconfont-阿里巴巴矢量图标库. <https://www.iconfont.cn/>