考试管理系统

qt程序设计报告

JS321141马浩轩

8.20

目录

[一、需求分析, 3](#_Toc112845923)

[二、概要设计： 3](#_Toc112845924)

[1. 主界面设计 3](#_Toc112845925)

[2. 学生端设计： 3](#_Toc112845926)

[3. 教师端设计: 4](#_Toc112845927)

[4. 管理端设计 : 4](#_Toc112845928)

[三、详细设计： 5](#_Toc112845929)

[1. 三个基本类： 5](#_Toc112845930)

[a. Student类： 5](#_Toc112845931)

[b. Teacher类： 5](#_Toc112845932)

[c. Admin类： 6](#_Toc112845933)

[2. 主界面： : 7](#_Toc112845934)

[a. UI制作： 7](#_Toc112845935)

[b. 利用代码来设计界面的美观性与实用性： 8](#_Toc112845936)

[c. 登录的实现： 9](#_Toc112845937)

[3. 学生界面 10](#_Toc112845938)

[4. 教师界面 11](#_Toc112845939)

[a.学号栏： 11](#_Toc112845940)

[b.改错/编程栏： 11](#_Toc112845941)

[c.显示卷面 11](#_Toc112845942)

[d.计算总分: 11](#_Toc112845943)

[e.本卷判毕: 12](#_Toc112845944)

[f.判新卷: 12](#_Toc112845945)

[g.学生问题反馈: 12](#_Toc112845946)

[h.成绩管理界面: 12](#_Toc112845947)

[i.成绩保存目录栏 13](#_Toc112845948)

[5. 管理界面 14](#_Toc112845949)

[a.密码修改： 14](#_Toc112845950)

[b.全校成绩打印： 14](#_Toc112845951)

[c.管理员注册： 15](#_Toc112845952)

[四、代码实现： 15](#_Toc112845953)

[1.主界面： 15](#_Toc112845954)

[a.登录的实现： 15](#_Toc112845955)

[b.鼠标事件改写： 19](#_Toc112845956)

[c.教师登录的IP查询与检测实现 19](#_Toc112845957)

[2.学生界面： 20](#_Toc112845958)

[a. 成绩查询： 20](#_Toc112845959)

[b.成绩有误反馈： 20](#_Toc112845960)

[3.教师界面 22](#_Toc112845961)

[a.本卷判毕： 22](#_Toc112845962)

[b.计算总分： 23](#_Toc112845963)

[c.显示卷面： 24](#_Toc112845964)

[d.问题反馈： 26](#_Toc112845965)

[e.成绩修改： 27](#_Toc112845966)

[f.成绩查询: 30](#_Toc112845967)

[4.管理界面： 30](#_Toc112845968)

[a.成绩修改： 30](#_Toc112845969)

[b.全校成绩打印： 32](#_Toc112845970)

[c.管理员注册： 34](#_Toc112845971)

[五、软件测试： 35](#_Toc112845972)

[六、总结： 46](#_Toc112845973)

# 一、需求分析,

打造学校考试系统：

主界面：登录账号密码，根据选择账户类型转入不同账户界面

学生端：登录账号，查询成绩，问题反馈，修改密码

教师端：登录账号，批阅试卷(查看试卷(编程与改错)，计算总分，导入系统，阅览新卷，修改成绩，查询成绩，查询问题反馈，查看成绩保存文档

管理员端：修改学生，教师密码，打印全校成绩排名，注册管理员

# 二、概要设计：

## 1. 主界面设计

创建MainWindow类添加登录端及各部件,主登录界面可前往学生端，教师端，管理端.

.首先创建一个lineEdit并装饰，在里面输入账号，再创建第二个lineEdit并装饰，在里面输入密码（可实现加密密码技术，通过创建一个comboBox，通过其选择学生，教师或管理员，并识别密码是否正确，若正确则前往对应的端口界面。

.利用QtableWidget，Qframe，QlineEdit，Qlabel，Qwidget模块进行相互组合，并结合颜色（rgba），图片（Qsource）等调节透明度，颜色拼接，打造一个完美的登录界面

并创建了数据库，储存管理员的账号密码

并查询了IPV4地址，防止非内网窃取教师密码登录

## 2. 学生端设计：

学生登入到学生界面后，ID栏锁定学生的ID，lineEdit\_User设置为只读,学生可查询成绩，反馈问题

.查询成绩的实现：通过对成绩文档进行文件流读入，锁定到查询学生的ID，进而查询到该学生的成绩

.反馈问题的实现：通过文档写入，将该学生ID写入问题文档，并实现不可重复操作

a.创建Student类管理学生基本信息：ID,password,score

b.创建Stuscoremenu类显示学生成绩操作界面 包括成绩查询，问题反馈

c.创建 Stuscore 类显示学生查询成绩界面

## 3. 教师端设计:

教师登录教师界面后,可以进行批阅试卷(查看试卷(编程与改错)，计算总分，导入系统，阅览新卷，修改成绩，查询成绩，查询问题反馈，查看成绩保存文档

.查看试卷实现：通过spinDoubleBox选中编程或改错，进而通过卷面查看按钮的选择判断，利用文件读入将不同文本读入到text栏

.计算总分实现：首先将QString类转为int类，再计算总分导入总分栏

.本卷判毕实现: 将学号和总分按顺序导入到score.txt总分文档，并实现查看文档中是否有该学生成绩，防止重复批改

.阅览新卷实现: 利用clear函数清屏

.查询/修改成绩实现：利用checkBox判断进入不同界面，然后利用文本读写操作，以及QStinglist链表操作按顺序查询，修改成绩, 修改成绩后，自动删除doubt,.txt学生反馈问题的ID

.查询问题反馈: 通过读doubt.txt文档读取学生反馈信息

.查看成绩保存文档，调用score.txt实现

a.创建Teacher类管理教师基本信息: ID.password

b.创建Teamenu类显示教师批卷界面操作面板

c.创建Tmodify 类管理教师批改界面

d.创建Tsearch 类管理教师查询界面

## 4. 管理端设计 :

修改学生，教师密码，打印全校成绩排名

.成绩修改实现：通过文本读取，QSTringlist链表读写来修改密码

.全校成绩打印; 利用QStringlistr链表以及Student类的插入排序算法实现

.管理员注册: 创建db数据库，并添加admin表格，通过数据库操作来管理管理员

a.创建Admin类管理管理员基本信息:ID.password

b.创建admenu 类管理管理员操作面板

c.创建Stupassmod创建学生密码修改界面

d.创建Teapassmod创建教师密码修改界面

e.创建Rank类创建全校成绩排行打印界面

f .创建db数据库

# 三、详细设计：

## 1. 三个基本类：

### a. Student类：

学习信息管理类

public:

explicit **Student**(QWidget \*parent = nullptr); //构造/析构函数

~***Student***();

//get返回函数

QString **getID**() const;

QString **getscore**() const;

QString **getpss**();

//set获取函数

void **setID**(const QString &value);

void **setscore**(const QString &value);

void **setpss**(const QString &value);

private:

Ui::Student \*ui; //ui界面

QString m\_strID; //学生账号

QString m\_strScore; //学生成绩

QString m\_strPss; //学生密码

Stuscore m\_Stusc; //学生成绩查询界面

};

### b. Teacher类：

教师信息管理类：

public:

explicit **Teacher**(QWidget \*parent = nullptr); //构造析构函数

~***Teacher***();

//获取ID,Pss

QString **getID**() const;

QString **getpss**() const;

//设置ID，Pss

void **setID**(const QString &value);

void **setpss**(const QString &value);

private:

Ui::Teacher \*ui;

QString m\_strID;

QString m\_strPss;

TeaMenu m\_Tm;

};

### c. Admin类：

管理员信息管理类：

public:

explicit **Student**(QWidget \*parent = nullptr); //构造/析构函数

~***Student***();

//get返回函数

QString **getID**() const;

QString **getscore**() const;

QString **getpss**();

//set获取函数

void **setID**(const QString &value);

void **setscore**(const QString &value);

void **setpss**(const QString &value);

private:

Ui::Student \*ui; //ui界面

QString m\_strID; //学生账号

QString m\_strScore; //学生成绩

QString m\_strPss; //学生密码

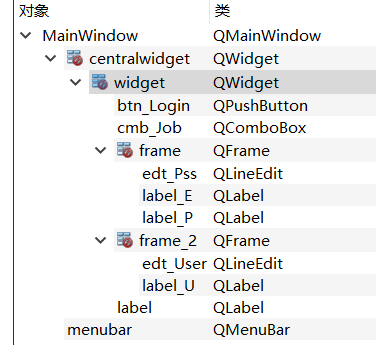
Stuscore m\_Stusc; //学生成绩查询界面

};

## 2. 主界面： :



### UI制作：



利用Qframe widget label 嵌套组合使界面美观：

Frame\_2:

edt\_User 登录账户输入口

Label\_U 登陆小人设置背景图片

Frame\_;

edt\_Pss 登录密码输入口

Label\_U 登陆密码锁设置背景图片

Label\_p 登录右侧眼睛触碰显示密码

将两个frame以及东大校徽label一起放入widget，并设置widget白色半透明度，最后将widget放入为已设置后背景样式的mainwindow里最终组成登陆界面

### b. 利用代码来设计界面的美观性与实用性：

this->setWindowTitle(QStringLiteral("东大考试系统"));//设置窗口名称

//设置背景颜色

ui->widget->setStyleSheet("background-color:rgba(244,244,244,200)");

//设置edit栏字体，字号，提示信息，输入限制

ui->btn\_Login->setFont(QFont("宋体", 18));

ui->edt\_User->setFont(QFont("宋体", 18));

ui->edt\_Pss->setFont(QFont("宋体", 18));

ui->cmb\_Job->setFont(QFont("宋体", 12));

ui->edt\_User->setPlaceholderText("账号");

ui->edt\_Pss->setPlaceholderText("密码");

QRegExp regx("[0-9]+$");

QValidator \*validator = new QRegExpValidator(regx, ui->edt\_Pss );

ui->edt\_Pss->setValidator(validator);

ui->edt\_Pss->setMaxLength(6);

//设置确定按钮点击样式

QString btnStyle2 =

"QPushButton{\

color: rgb(255, 255, 255);\

background: qlineargradient(x1:0, y1:0, x2:0, y2:1, stop:0 rgb(166,164,208), stop:0.3 rgb(171,152,230), stop:1 rgb(152,140,220));\

border:1px;\

border-radius:5px; /\*border-radius控制圆角大小\*/\

padding:2px 4px; \

}\

QPushButton:hover{\

color: rgb(255, 255, 255); \

background: qlineargradient(x1:0, y1:0, x2:0, y2:1, stop:0 rgb(130,120,226), stop:0.3 rgb(120,130,230), stop:1 rgb(125,140,226));\

border:1px; \

border-radius:5px; /\*border-radius控制圆角大小\*/\

padding:2px 4px; \

}\

QPushButton:pressed{ \

color: rgb(255, 255, 255); \

background: qlineargradient(x1:0, y1:0, x2:0, y2:1, stop:0 rgb(240,156,121), stop:0.3 rgb(220,160,140), stop:1 rgb(230,140,120)); \

border:1px; \

border-radius:5px; /\*border-radius控制圆角大小\*/\

padding:2px 4px; \

}";

ui->btn\_Login->setStyleSheet(btnStyle2);

//设置账号，密码前前置，后置图案

ui->label\_E->setAlignment(Qt::AlignCenter);

ui->label\_E->setPixmap(QPixmap(":/image.qrc/EYE.JPG").scaled(20, 20));

ui->label\_U->setAlignment(Qt::AlignCenter);

ui->label\_U->setPixmap(QPixmap(":/image.qrc/pic/user\_3.png").scaled(20, 20));

ui->label->setAlignment(Qt::AlignCenter);

ui->label->setPixmap(QPixmap(":/image.qrc/pic/seu.png").scaled(20, 20));

//设置密码隐藏功能

ui->edt\_Pss->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);

ui->edt\_Pss->setEchoMode(QLineEdit::Password);

ui->label\_E->installEventFilter(this);//安装事件触发器

### c. 登录的实现：

首先，利用comboBox设计登录用户的身份选择（学生，教师，管理员）

其次，学生和教师利用文件读写检验的操作验证登录密码，其中，教师登录需要校园网，非校园网教师账户无法登陆。

最后，管理员利用已经创建好的数据库来检验登录密码

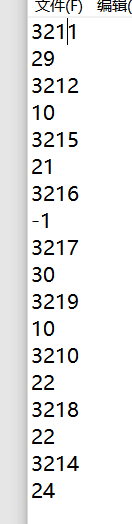
值得注意的是所有密码为只为数字，最多六位，这在本小节中的b中已详细给出，详细登录代码的实现请见四、代码实现

## 3. 学生界面



1. 成绩查询：

成绩查询主要利用了QSTringlist类链表



首先打开文档score.txt，将所有文件按行拆分传入指针，然后逐个遍历，当所查询的id等于链表中的指针所指的id时，指针往后移动一位，即为该学生的成绩，返回到成绩显示的文本栏中。

1. 成绩有误反馈：

创建了一个doubt.txt文档，将学生有误反馈的id写入到文档中，并检验是否有id重复,利用messagebox提示重复。

## 4. 教师界面



### a.学号栏：

利用spinDoubleBox选择在3210-3220学号中的学生

### b.改错/编程栏：

利用comboBox选择改错或编程

### c.显示卷面

读入学号栏中的学号，并且利用path路径凭借的技巧

QString path\_1="D:\\qtstudy\\summer\_school\\school\\SCHOOL\\paper\\";

QString path\_2=t\_nStu.getID();

QString path\_3="g.txt";

QString path=path\_1+path\_2+path\_3;

打开该学生编程或改错的试卷，并显示在右侧大的textEdit批卷栏里

其中，若paper文档里无该考生卷子，说明考生缺考，利用messageBox提示

### d.计算总分:

将各项小分写入到编程/改错得分的小分栏中，读入小分栏中的text文本，转为int类型（toInt）,最后相加再转为QString类型写入到总分栏中

### e.本卷判毕:

打开score.txt文本，首先利用QStringlist链表（原理同3.a成绩查询）判断该学生是否已批改完，若已批改完，则用messageBox提示信息，若未批改，则将该学生id，score逐行录入系统。

### f.判新卷:

主要利用edit文本的clear功能

### g.学生问题反馈:

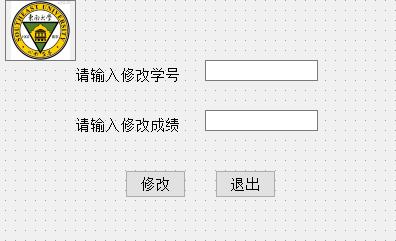
打开doubt.txt，将有问题的id逐行显示在右侧的text文本栏中，提示教师重新判该考生的卷子。

### h.成绩管理界面:

成绩管理界面打开后，为两个radioBox，

根据选择，进入不同的界面，

成绩修改界面：



利用QSTringlist链表原理（同3.a）

打开score.txt文档，将文档内容按行拆分逐次录入链表，并创建一个临时文档，将内容依次写入临时文档，

当检测到需修改学生的id后，将id写入到临时文档，再将修改后的成绩写入到临时文档，然后链表向后移动一位，跳过该学生的原成绩，最后再将后面的id以及score导入到临时文档，最后再将临时文档全部存入到score.txt中

（或许有些麻烦，但我也同时尝试了数据库的办法，相比数据库，文本的写法或许有些麻烦，但是相比数据库，文本可以在txt中直接操作，然而数据库无法可视化，无法在外部直接操作，因此我在这里选用了txt文本的方式）

成绩查询界面：



原理同3.a(一模一样)

### i.成绩保存目录栏

检索score.txt文档，超级简单。

## 5. 管理界面



### a.密码修改：

利用了与4.g 中成绩修改一模一样的技术，但值得一提的是，在这里我加入了两次密码相同确认的额外操作。

### b.全校成绩打印：

首先，创建了一个rank类，然后设计了Qtablewidget的列表控件放置成绩，

其次，打开score.txt 文本然后利用QSTringlist链表读入，并写入到一个student类数组中

再次，利用student类的插入排序，将student类按score进行排序

//利用插入排序排名成绩

for( int i =0;i<t\_num;i++)

{

for(int j=i;j>=1;j--)

{

if(t\_nStu[j].getscore().toInt()<t\_nStu[j-1].getscore().toInt())

{

Student t\_s;

t\_s.setID(t\_nStu[j].getID());

t\_s.setscore(t\_nStu[j].getscore());

t\_nStu[j].setID(t\_nStu[j-1].getID());

t\_nStu[j].setscore(t\_nStu[j-1].getscore());

t\_nStu[j-1].setID(t\_s.getID());

t\_nStu[j-1].setscore(t\_s.getscore());

}

else

break;

}

}

最后，将排好序的student类按照id，score依次写入tablewidget表格中，实现打印成绩。

### c.管理员注册：

创建数据库，利用数据库操作，相比txt文件异常简单，但无法在程序外操作，无法实现可视化。

# 四、代码实现：

## 1.主界面：

### a.登录的实现：

connect(ui->btn\_Login,&QPushButton::clicked,[=]()

{

if(ui->cmb\_Job->currentText()=="学生")//判断cmb\_Job选择为学生

{

Student t\_Stu;

t\_Stu.setID(ui->edt\_User->text());//创建临时学生类并读入ID

m\_Stusc.ui->edt\_User->setText(t\_Stu.getID()); //设置学生查询成绩界面ID

m\_Stusc.ui->edt\_User->setReadOnly(true); //将学生查询成绩界面ID设为只读

QString path\_1="D:\\qtstudy\\summer\_school\\school\\SCHOOL\\pass.txt"; //打开学生密码文件

QString path\_2="D:\\qtstudy\\summer\_school\\school\\SCHOOL\\score.txt"; //打开学生成绩文件

//文件读写操作验证密码是否正确

QFile file\_1(path\_1);

QFile file\_2(path\_2);

file\_1.*open*(QIODevice::ReadOnly | QIODevice::Text);

file\_2.*open*(QIODevice::ReadOnly | QIODevice::Text);

QString line\_1;

QString line\_2;

QTextStream in\_1(&file\_1); //用文件构造流

QTextStream in\_2(&file\_2);

line\_1 = in\_1.readLine(); //读取一行放到字符串

line\_2 = in\_2.readLine();

bool t\_nFlag=false; //定义flag用于判定是否找到该学生

while(!line\_2.isNull()) //字符串有内容

{

if(line\_2==t\_Stu.getID())//查询到该学生

{

line\_2=in\_2.readLine();//循环读取下行

t\_nFlag=true; //flag为true

break;

}

line\_2=in\_2.readLine();

}

if(t\_nFlag==false)

QMessageBox::warning(this,"提示","没有您的成绩",QMessageBox::Ok); //利用 QMessageBox 提示

else

{

while(!line\_1.isNull()) //字符串有内容

{

if(line\_1==t\_Stu.getID())

{

line\_1=in\_1.readLine(); //循环读取下行

break;

}

line\_1=in\_1.readLine();

}

if(line\_1==ui->edt\_Pss->text()) //密码正确

{

m\_Stusc.*exec*();

}

else

{ //利用 QMessageBox 提示

QMessageBox::warning(this,"提示","密码错误",QMessageBox::Ok);

}

}

file\_1.*close*(); //及时关闭文件防止误操作

file\_2.*close*();

}//学生界面结束

if(ui->cmb\_Job->currentText()=="教师") //判断cmb\_Job选择为学生

{

if(ui->edt\_User->text()==""||ui->edt\_Pss->text()=="") //未输入账号或密码

QMessageBox::warning(this,"提示","请填写账号或密码",QMessageBox::Ok); //利用 QMessageBox提示

else

{

Teacher t\_Tea;

t\_Tea.setID(ui->edt\_User->text());//创建临时学生类并读入ID

//打开教师密码文件

QString path="D:\\qtstudy\\summer\_school\\school\\SCHOOL\\tpass.txt";

//文件读写操作验证密码是否正确

QFile file(path);

file.*open*(QIODevice::ReadOnly | QIODevice::Text);

QString line;

QTextStream in(&file); //用文件构造流

line = in.readLine(); //读取一行放到字符串

while(!line.isNull()) //字符串有内容

{

if(line==t\_Tea.getID())

{

line=in.readLine(); //循环读取下行

break;

}

line=in.readLine();

}

if(line==ui->edt\_Pss->text()) //密码正确

m\_Tm.*exec*(); //转至教师界面

else

//未输入账号或密码

QMessageBox::warning(this,"提示","密码错误",QMessageBox::Ok);

file.*close*(); //及时关闭文件防止误操作

}

}//教师界面结束

if(ui->cmb\_Job->currentText()=="管理员") //判断cmb\_Job选择为学生

{

QString id=ui->edt\_User->text();

QString psswd=ui->edt\_Pss->text();

QSqlDatabase db = QSqlDatabase::database();

db.open();

QString t\_strCmd="create table if not exists admin (id varchar(64),psswd varchar(64))";

QSqlQuery query;

query.exec(t\_strCmd);

QString id\_1="5811";

QString psswd\_2="123456";

QString str = QString("insert into admin values('%1', '%2')").arg(id\_1).arg(psswd\_2);

query.exec(str);

QSqlTableModel \*model=new QSqlTableModel;

model->*setTable*("admin");

model->*setFilter*(QString("id='%1' and psswd='%2'").arg(id).arg(psswd));

model->*select*();

int row=model->*rowCount*();

if(row>0)

{

m\_Am.*exec*();

}

else

{

QMessageBox::warning(this,"提示","密码错误",QMessageBox::Ok);

}

}

});

### b.鼠标事件改写：

bool MainWindow::***eventFilter***(QObject \*watched, QEvent \*event)

{

switch (event->type()) {

case QEvent::Enter:

ui->edt\_Pss->setEchoMode(QLineEdit::Normal);

break;

case QEvent::Leave:

ui->edt\_Pss->setEchoMode(QLineEdit::Password);

break;

default:

break;

}

return QWidget::eventFilter(watched, event);

}

### c.教师登录的IP查询与检测实现

QString strIpAddress;

QList<QHostAddress> ipAddressesList = QNetworkInterface::allAddresses();

// 获取第一个本主机的IPv4地址

int nListSize = ipAddressesList.size();

for (int i = 0; i < nListSize; ++i)

{

if (ipAddressesList.at(i) != QHostAddress::LocalHost &&

ipAddressesList.at(i).toIPv4Address())

{

strIpAddress = ipAddressesList.at(i).toString();

break;

}

}

// 如果没有找到，则以本地IP地址为IP

if (strIpAddress.isEmpty())

{

strIpAddress = QHostAddress(QHostAddress::LocalHost).toString();

}

if(strIpAddress=="10.208.67.113")

## 2.学生界面：

### a. 成绩查询：

QString path="D:\\qtstudy\\summer\_school\\school\\SCHOOL\\score.txt"; //打开文件

//利用文件流验证

QFile file(path);

file.*open*(QIODevice::ReadOnly | QIODevice::Text);

QString line;

QTextStream in(&file); //用文件构造流

line = in.readLine();//读取一行放到字符串

while(!line.isNull())//字符串有内容

{

if(line==ui->edt\_User->text())

{

line=in.readLine();//循环读取下行

break;

}

line=in.readLine();

}

ui->edt\_Pss->setText(line); //导入成绩

file.*close*();

### b.成绩有误反馈：

connect(ui->btn\_Doubt,&QPushButton::clicked,[=]()//成绩有误按钮lamda表达式

{

QFile file("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\doubt.txt");

file.*open*(QIODevice::ReadOnly | QIODevice::Text); //打开学生成绩文件

//利用文件流操作

QString line;

QTextStream in(&file); //用文件构造流

line = in.readLine(); //读取一行放到字符串

bool t\_flag=false; //利用t\_flag遍历判断是否有该学生，默认为false

while(!line.isNull()) //字符串有内容

{

if(line==ui->edt\_User->text()) //学生存在

{

line=in.readLine(); //循环读取下行

t\_flag=true;

break; //判断成功，跳出循环

}

line=in.readLine();

}

file.*close*(); //用完后关闭文件防止误操作

if(t\_flag==false) //成绩单无该学生，录入该学生成绩

{

file.*open*(QIODevice::WriteOnly|QIODevice::Append); //在尾部打开学生成绩单

QString data\_1=ui->edt\_User->text()+"\n"; //学生学号

QTextStream dataStr(&file); //用文件输出流

dataStr << data\_1; //录入成绩

file.*close*(); //用完后关闭文件防止误操作

}

else

{

QMessageBox::warning(this,"提示","此学生已录入系统",QMessageBox::Ok);

}

});

## 3.教师界面

### a.本卷判毕：

connect(ui->btn\_Over,&QPushButton::clicked,[=]()//判卷完毕按钮lamda表达式

{

QString t\_strID = ui->spb\_StuID->text();

QString t\_strSc = ui->edt\_Total->text();

QString str = QString("insert into student(t\_strID, t\_strSc) values('%1', '%2')").arg(t\_strID).arg(t\_strSc);

QSqlQuery query;

query.exec(str); //执行插入操作

QFile file("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\score.txt"); //定义学生成绩文件

file.*open*(QIODevice::ReadOnly | QIODevice::Text); //打开学生成绩文件

//利用文件流操作

QString line;

QTextStream in(&file); //用文件构造流

line = in.readLine(); //读取一行放到字符串

bool t\_flag=false; //利用t\_flag遍历判断是否有该学生，默认为false

while(!line.isNull()) //字符串有内容

{

if(line==ui->spb\_StuID->text()) //学生存在

{

line=in.readLine(); //循环读取下行

t\_flag=true;

break; //判断成功，跳出循环

}

line=in.readLine();

}

file.*close*(); //用完后关闭文件防止误操作

if(t\_flag==false) //成绩单无该学生，录入该学生成绩

{

file.*open*(QIODevice::WriteOnly|QIODevice::Append); //在尾部打开学生成绩单

QString data\_1=ui->spb\_StuID->text()+"\n"; //学生学号

QString data\_2=ui->edt\_Total->text()+"\n"; //学生成绩

QTextStream dataStr(&file); //用文件输出流

dataStr << data\_1<<data\_2; //录入成绩

file.*close*(); //用完后关闭文件防止误操作

}

else

{

QMessageBox::warning(this,"提示","此学生已录入系统",QMessageBox::Ok);

}

});

### b.计算总分：

connect(ui->btn\_Cal,&QPushButton::clicked,[=]() //计算总分lamda表达式

{

if(ui->edt\_S1->text()==""||ui->edt\_S2->text()==""||

ui->edt\_S3->text()==""||ui->edt\_S4->text()==""||

ui->edt\_S5->text()==""||ui->edt\_S6->text()=="") //判断信息是否填写完全

{

QMessageBox::warning(this,"提示","请将信息填写完整",QMessageBox::Ok); //利用QMessageBox提示未填写完全

}

else

{

int t\_nSum=ui->edt\_S1->text().toInt()+ui->edt\_S2->text().toInt()

+ui->edt\_S3->text().toInt()+ui->edt\_S4->text().toInt()

+ui->edt\_S5->text().toInt()+ui->edt\_S6->text().toInt(); //将QString转为int计算总成绩

if(t\_nSum>30)//总分超过30提示重新批卷

{

QMessageBox::warning(this,"提示","总分超过30，分数填写有误",QMessageBox::Ok);//利用QMessageBox提示批错了

}

else

{

QString t\_strSum=QString::number(t\_nSum); //将int类型的总分转为QString类型

ui->edt\_Total->setText(t\_strSum); //将总分显示在总分栏

}

}

});

### c.显示卷面：

connect(ui->btn\_Paper,&QPushButton::clicked,[=]()//显示卷面的lamda表达式

{

if(ui->cmb\_Paper->currentText()=="改错") //进入改错界面

{

Student t\_nStu;

t\_nStu.setID(ui->spb\_StuID->text()); //创建临时学生类记录学生学号

//创建学生试卷文件路径

QString path\_1="D:\\qtstudy\\summer\_school\\school\\SCHOOL\\paper\\";

QString path\_2=t\_nStu.getID();

QString path\_3="g.txt";

QString path=path\_1+path\_2+path\_3;

//利用文件流操作

QFileInfo fi(path);

if(fi.isFile()) //判断文件是否存在

{

QFile file(path);

file.*open*(QIODevice::ReadOnly);

QByteArray array=file.readAll(); //将试卷信息读入QBytearray

ui->edt\_Paper->setText(array); //将学生试卷显示到textEdit上

file.*close*(); //用完及时关掉，防止误用

}

else

{

QMessageBox::warning(this,"提示","学生缺考，手动设置成绩-1",QMessageBox::Ok);//利用QMessageBox提示学生缺号

}

}//改错界面结束

if(ui->cmb\_Paper->currentText()=="编程") //进入编程界面

{

Student t\_nStu;

t\_nStu.setID(ui->spb\_StuID->text()); //创建临时学生类记录学生学号

//创建学生试卷文件路径

QString path\_1="D:\\qtstudy\\summer\_school\\school\\SCHOOL\\paper\\";

QString path\_2=t\_nStu.getID();

QString path\_3="b.txt";

QString path=path\_1+path\_2+path\_3;

//利用文件流操作

QFileInfo fi(path);

if(fi.isFile()) //判断文件是否存在

{

QFile file(path);

file.*open*(QIODevice::ReadOnly);

QByteArray array=file.readAll(); //将试卷信息读入QBytearray

ui->edt\_Paper->setText(array); //将学生试卷显示到textEdit上

file.*close*(); //用完及时关掉，防止误用

}

else

{

QMessageBox::warning(this,"提示","无此学号的学生",QMessageBox::Ok);

}

}//编程界面结束

});

### d.问题反馈：

connect(ui->btn\_Doubt,&QPushButton::clicked,[=]()//显示卷面的lamda表达式

{

QString path="D:\\qtstudy\\summer\_school\\school\\SCHOOL\\doubt.txt";

//利用文件流操作

QFileInfo fi(path);

if(fi.isFile()) //判断文件是否存在

{

QFile file(path);

file.*open*(QIODevice::ReadOnly);

QByteArray array=file.readAll(); //将试卷信息读入QBytearray

ui->edt\_Paper->setText(array); //将学生试卷显示到textEdit上

file.*close*(); //用完及时关掉，防止误用

}

else

{

QMessageBox::warning(this,"提示","无问题",QMessageBox::Ok);//利用QMessageBox提示

}

});

### e.成绩修改：

connect(ui->btn\_Modify,&QPushButton::clicked,[=]()//修改按钮lamda表达式

{

QString strALL; //用于储存文件全部信息

QStringList strList; //利用QString链表用于后续文件操作

//读文件流操作

QFile readFile("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\score.txt");

if(readFile.*open*((QIODevice::ReadOnly|QIODevice::Text))) //判断文件是否存在标准操作

{

QTextStream stream(&readFile); //构造输入流

strALL=stream.readAll(); //将strALL变为文本全部信息

}

readFile.*close*(); //及时关闭文件方式误操作

//写文件流操作

QFile writeFile("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\score.txt");

if(writeFile.*open*(QIODevice::WriteOnly|QIODevice::Text)) //判断文件是否存在标准操作

{

QTextStream stream(&writeFile); //构造输出流

strList=strALL.split("\n"); //通过换行符将文件每一行信息读如QString链表

for(int i=0;i<strList.count();i++) //遍历链表

{

if(strList.at(i).contains(ui->edt\_ID->text())) //当读到该学生学号时

{

stream<<ui->edt\_ID->text()<<"\n"; //通过输出流将学生ID写入文件

QString t\_Str=strList.at(i+1); //链表向后移动到该学生成绩

//将该学生成绩替换为修改后的成绩

t\_Str.replace(0,t\_Str.length(),ui->edt\_Score->text());

stream<<t\_Str<<"\n"; //写入

i++; //链表后移到下一位同学

//利用提示信息通知修改成功

QMessageBox::warning(this,"恭喜","修改成功",QMessageBox::Ok);

}

else //最后一行特殊处理

{

if(i==strList.count()-1)

{

stream<<strList.at(i);

}

else

{

stream<<strList.at(i)<<"\n";

}

}

}

}

//及时关闭防止误操作

writeFile.*close*();

//及时将学生反馈文件中的学号清理掉

QString strALL\_2; //用于储存文件全部信息

QStringList strList\_2; //利用QString链表用于后续文件操作

//读文件流操作

QFile readFile\_2("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\doubt.txt");

if(readFile\_2.*open*((QIODevice::ReadOnly|QIODevice::Text))) //判断文件是否存在标准操作

{

QTextStream stream\_2(&readFile\_2); //构造输入流

strALL\_2=stream\_2.readAll(); //将strALL变为文本全部信息

}

readFile\_2.*close*(); //及时关闭文件方式误操作

//写文件流操作

QFile writeFile\_2("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\doubt.txt");

if(writeFile\_2.*open*(QIODevice::WriteOnly|QIODevice::Text)) //判断文件是否存在标准操作

{

QTextStream stream\_2(&writeFile\_2); //构造输出流

strList\_2=strALL\_2.split("\n"); //通过换行符将文件每一行信息读如QString链表

for(int i=0;i<strList\_2.count();i++) //遍历链表

{

if(strList\_2.at(i).contains(ui->edt\_ID->text())) //当读到该学生学号时

{

/\* i++; //链表后移到下一位同学

stream<<strList.at(i)<<"\n";\*/ //通过输出流将学生ID写入文件

}

else //最后一行特殊处理

{

if(i==strList\_2.count()-1)

{

stream\_2<<strList\_2.at(i);

}

else

{

stream\_2<<strList\_2.at(i)<<"\n";

}

}

}

}

});

### f.成绩查询:

connect(ui->btn\_Search,&QPushButton::clicked,[=]()//查询按钮lamda表达式

{

Student t\_stu;

t\_stu.setID(ui->edt\_ID->text());//创建临时学生类读入该学生ID

//文件流操作

QFile file("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\score.txt");

file.*open*(QIODevice::ReadOnly | QIODevice::Text); //打开学生成绩文件

QString line;

QTextStream in(&file); //用文件构造流

line = in.readLine(); //读取一行放到字符串

qDebug() << line;

while(!line.isNull()) //字符串有内容

{

if(line==t\_stu.getID()) //查询有该学生ID

{

line=in.readLine(); //循环读取下行

break;

}

line=in.readLine();

}

ui->edt\_Score->setText(line);

file.*close*();

});

## 4.管理界面：

### a.成绩修改：

connect(ui->btn\_Modify,&QPushButton::clicked,[=]()

{

if(ui->edt\_Pss->text()!=ui->edt\_Pss\_2->text())

{

QMessageBox::warning(this,"提示","密码不一致",QMessageBox::Ok);

}

else

{

Teacher t\_tea;

t\_tea.setID(ui->edt\_User->text());

QString strALL;

QStringList strList;

QFile readFile("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\tpass.txt");

if(readFile.*open*((QIODevice::ReadOnly|QIODevice::Text)))

{

QTextStream stream(&readFile);

strALL=stream.readAll();

}

readFile.*close*();

QFile writeFile("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\tpass.txt");

if(writeFile.*open*(QIODevice::WriteOnly|QIODevice::Text))

{

QTextStream stream(&writeFile);

strList=strALL.split("\n");

for(int i=0;i<strList.count();i++)

{

if(strList.at(i).contains(ui->edt\_User->text()))

{

stream<<ui->edt\_User->text()<<"\n";

QString t\_Str=strList.at(i+1);

t\_Str.replace(0,t\_Str.length(),ui->edt\_Pss->text());

stream<<t\_Str<<"\n";

i++;

QMessageBox::warning(this,"恭喜","修改成功",QMessageBox::Ok);

}

else

{

if(i==strList.count()-1)

{

stream<<strList.at(i);

}

else

{

stream<<strList.at(i)<<"\n";

}

}

}

}

writeFile.*close*();

}

});

### b.全校成绩打印：

connect(ui->btn\_Pin,&QPushButton::clicked,[=]() //btn\_pin的lamda表达式

{

QString ID[100];//用来存储从数据库中找出来的信息

QString score[100];//用来存储从数据库中找出来的信息

int i = 0;

QSqlQuery query;

query.exec("select \* from student");//查询所有的信息

while(query.next())

{

ID[i] = query.value(0).toString();

score[i] = query.value(1).toString();

i++;

}

ui->textEdit->clear();

int j = 0;

for(j = 0; j < i; j++)//将这些信息都显示在下方的文本编辑框中

{

QString str = QString("学号：%1 成绩：%2").arg(ID[j]).arg(score[j]);

ui->textEdit->append(str);

}

//初始化

ui->tb\_Rank->setHorizontalHeaderLabels(QStringList() << "学号" << "成绩");

//文件流操作

QString path="D:\\qtstudy\\summer\_school\\school\\SCHOOL\\score.txt";;

QFile file(path);

file.*open*(QIODevice::ReadOnly | QIODevice::Text);

QString line;

QTextStream in(&file); //用文件构造流

line = in.readLine(); //读取一行放到字符

QString strALL; //全部读出成绩

QStringList strList; //调用QString链表

//文件读入

QFile readFile("D:\\qtstudy\\summer\_school\\school\\SCHOOL\\score.txt");

if(readFile.*open*((QIODevice::ReadOnly|QIODevice::Text)))

{

QTextStream stream(&readFile);

strALL=stream.readAll();

}

readFile.*close*(); //即使关闭文件

strList=strALL.split("\n"); //链表导入学号与成绩

int t\_num=strList.count()/2; //学生总数

Student t\_nStu[t\_num]; //创建学生临时变量

int t\_d=0; //计数器

for(int i=0;i<strList.count()-1;i++) //遍历

{

t\_nStu[t\_d].setID(strList.at(i)); //设置学号

t\_nStu[t\_d].setscore(strList.at(i+1)); //设置成绩

t\_d++; //学号加一

i++;

}

//利用插入排序排名成绩

for( int i =0;i<t\_num;i++)

{

for(int j=i;j>=1;j--)

{

if(t\_nStu[j].getscore().toInt()<t\_nStu[j-1].getscore().toInt())

{

Student t\_s;

t\_s.setID(t\_nStu[j].getID());

t\_s.setscore(t\_nStu[j].getscore());

t\_nStu[j].setID(t\_nStu[j-1].getID());

t\_nStu[j].setscore(t\_nStu[j-1].getscore());

t\_nStu[j-1].setID(t\_s.getID());

t\_nStu[j-1].setscore(t\_s.getscore());

}

else

break;

}

}

//将成绩写入tablewidget

for(int i=0;i<t\_num;i++)

{

for(int j=0;j<2;j++)

{

if(j==0)

ui->tb\_Rank->setItem(i,j,new QTableWidgetItem(t\_nStu[t\_num-1-i].getID()));

if(j==1)

ui->tb\_Rank->setItem(i,j,new QTableWidgetItem(t\_nStu[t\_num-1-i].getscore()));

}

}

### c.管理员注册：

QSqlDatabase db = QSqlDatabase::database();

db.open();

QString t\_strCmd="create table if not exists admin (id varchar(64),psswd varchar(64))";

QSqlQuery query;

query.exec(t\_strCmd);

connect(ui->pushButton,&QPushButton::clicked,[=]()

{

QString id = ui->lineEdit\_1->text();

if(ui->lineEdit\_2->text()==ui->lineEdit\_3->text())

{

QString psswd=ui->lineEdit\_2->text();

QString str = QString("insert into admin values('%1', '%2')").arg(id).arg(psswd);

QSqlQuery query;

query.exec(str);

QMessageBox::warning(this,"提示","注册成功",QMessageBox::Ok);//利用QMessageBox提示批错了

}

else

{

QMessageBox::warning(this,"提示","密码不一致",QMessageBox::Ok);//利用QMessageBox提示批错了

}

});

# 五、软件测试：

学生登录：



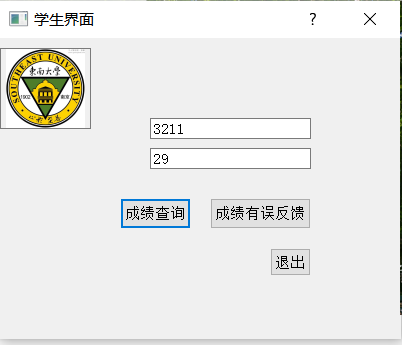
触碰眼睛显示密码：



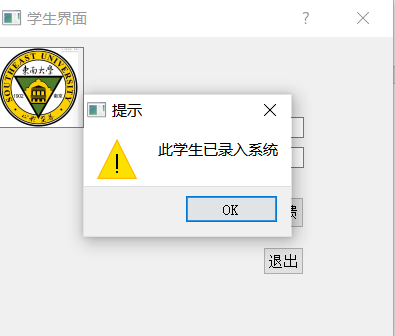
密码错误提示：



成绩查询：

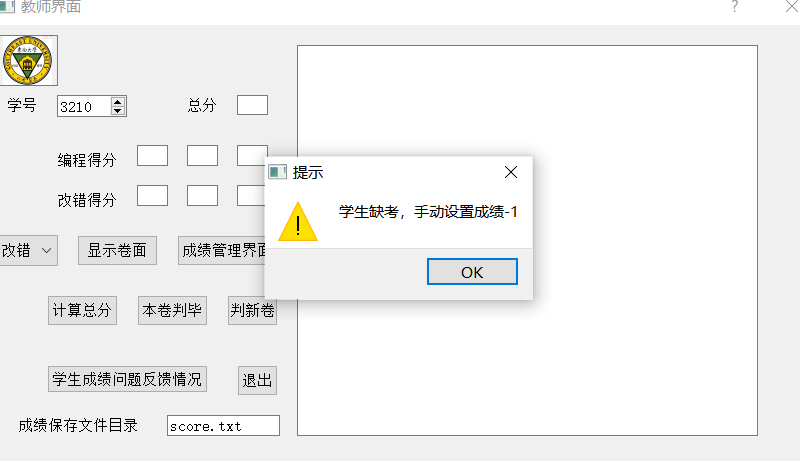


成绩有误反馈：

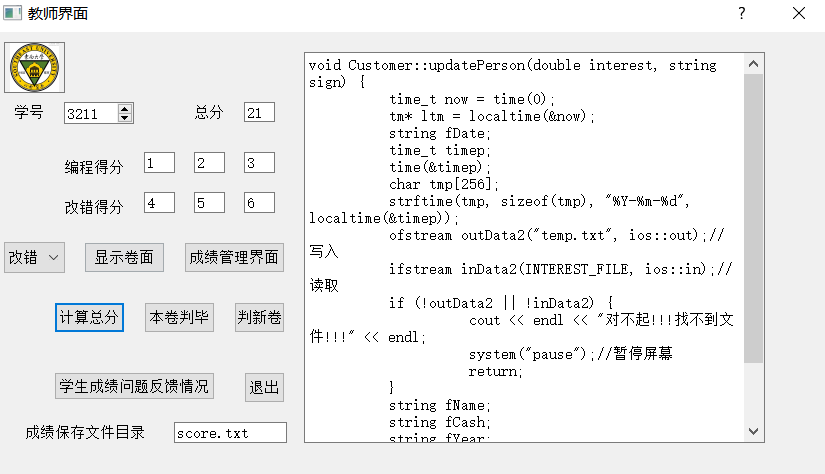


教师界面：

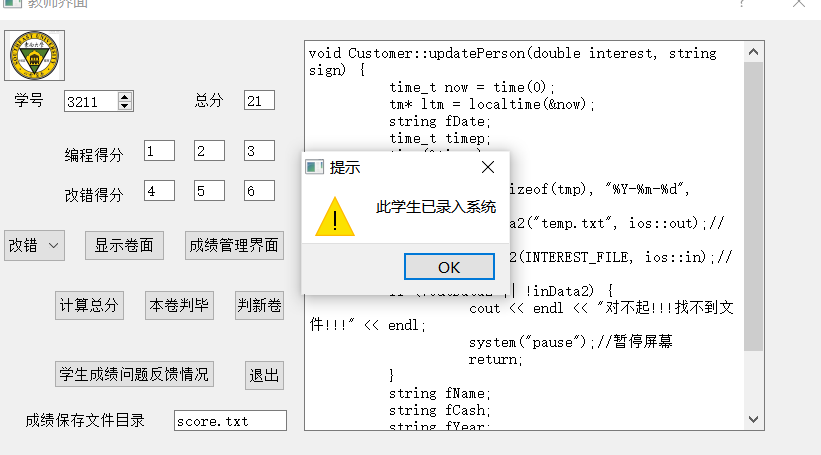
缺考：



显示卷面并计算总分



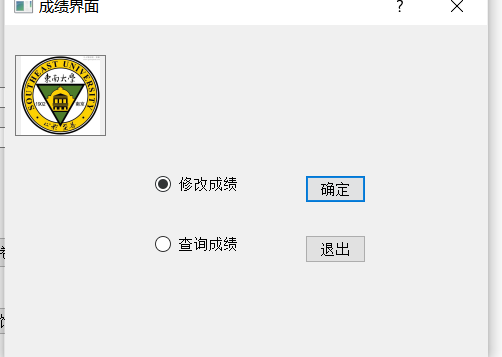
本卷判毕：



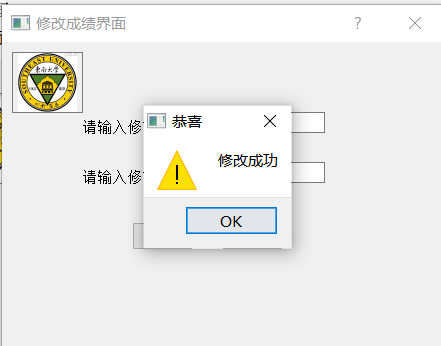
判新卷：



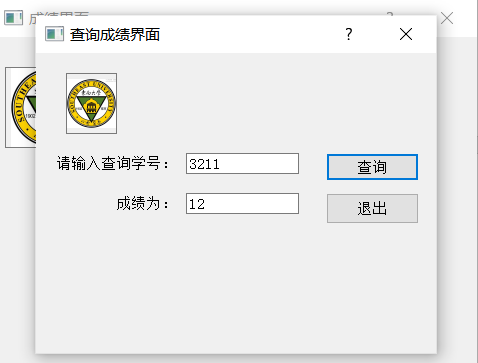
成绩管理界面：



修改成绩：



查询成绩：



成绩有误反馈：



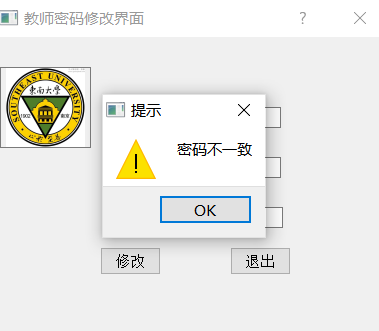
修改3211后3211消失：



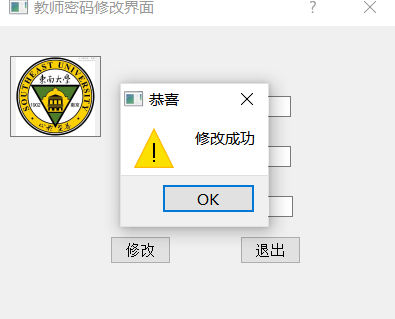
管理界面：

密码修改：





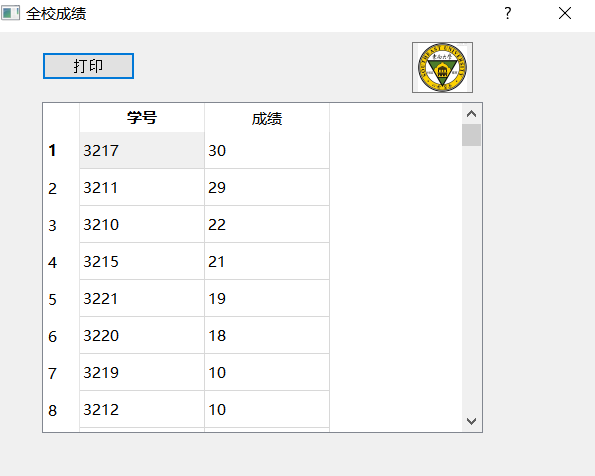




管理员注册：

全校成绩打印：



# 六、总结：

通过本次对qt的学习，我进一步掌握了对c++语言可视化技术的掌握，学会了包括ui界面设计，控件连接，信号槽等技术。

在本项目中，以下几点让我无比欣慰：

1. 完全自主操作实现了与东大综合服务大厅一样的ui界面
2. 利用qstringlist链表对文件进行修改查询
3. 学会了qt中的数据库，也对数据库有了基本的了解
4. 利用了插入排序等算法，将算法与软件结合
5. 对计算机网络有了初步的了解，其中对IPV4网址通过代码的实现也有初步掌握
6. 老师的代码规范指导对我影响深远