**自行车租赁监管系统**

**开发记录文档**

**计算机科学与技术学院**

**2020.6**

#### 目录

[1. 用户界面设计 3](#_Toc10429)

[1.1. 设计 3](#_Toc28250)

[1.1.1. 界面设计图 3](#_Toc13670)

[1.1.2. 相关代码 3](#_Toc5704)

[1.1.3. 操作逻辑流程图 5](#_Toc6273)

[1.1.4. 详细设计 5](#_Toc3684)

[1.2. 调试 6](#_Toc13701)

[1.2.1. 实际运行图 6](#_Toc27254)

[1.2.2. 问题及解决 7](#_Toc16022)

[2. 登录界面 8](#_Toc21385)

[2.1. 设计 8](#_Toc25131)

[2.1.1. 界面设计图 8](#_Toc2435)

[2.1.2. 相关代码 9](#_Toc9905)

[2.1.4. 逻辑流程图 11](#_Toc11960)

[2.2. 调试 12](#_Toc27741)

[2.2.1. 程序运行截图 12](#_Toc5361)

[2.2.2. 出现问题和解决 14](#_Toc3734)

[3. 车辆信息管理页面 15](#_Toc16311)

[3.1. 设计 15](#_Toc10646)

[3.1.1. 界面设计 15](#_Toc11745)

[3.1.2. 相关代码 15](#_Toc28216)

[3.1.3. 插入数据流程图 19](#_Toc19643)

[3.1.4. 显示所有的数据流程图 20](#_Toc32012)

[3.1.5. 查询车辆信息流程图 21](#_Toc18751)

[3.2. 调试 22](#_Toc19234)

[3.2.1 运行截图 22](#_Toc22582)

[3.2.2. 问题 25](#_Toc26164)

# 用户界面设计

## 设计

### 界面设计图

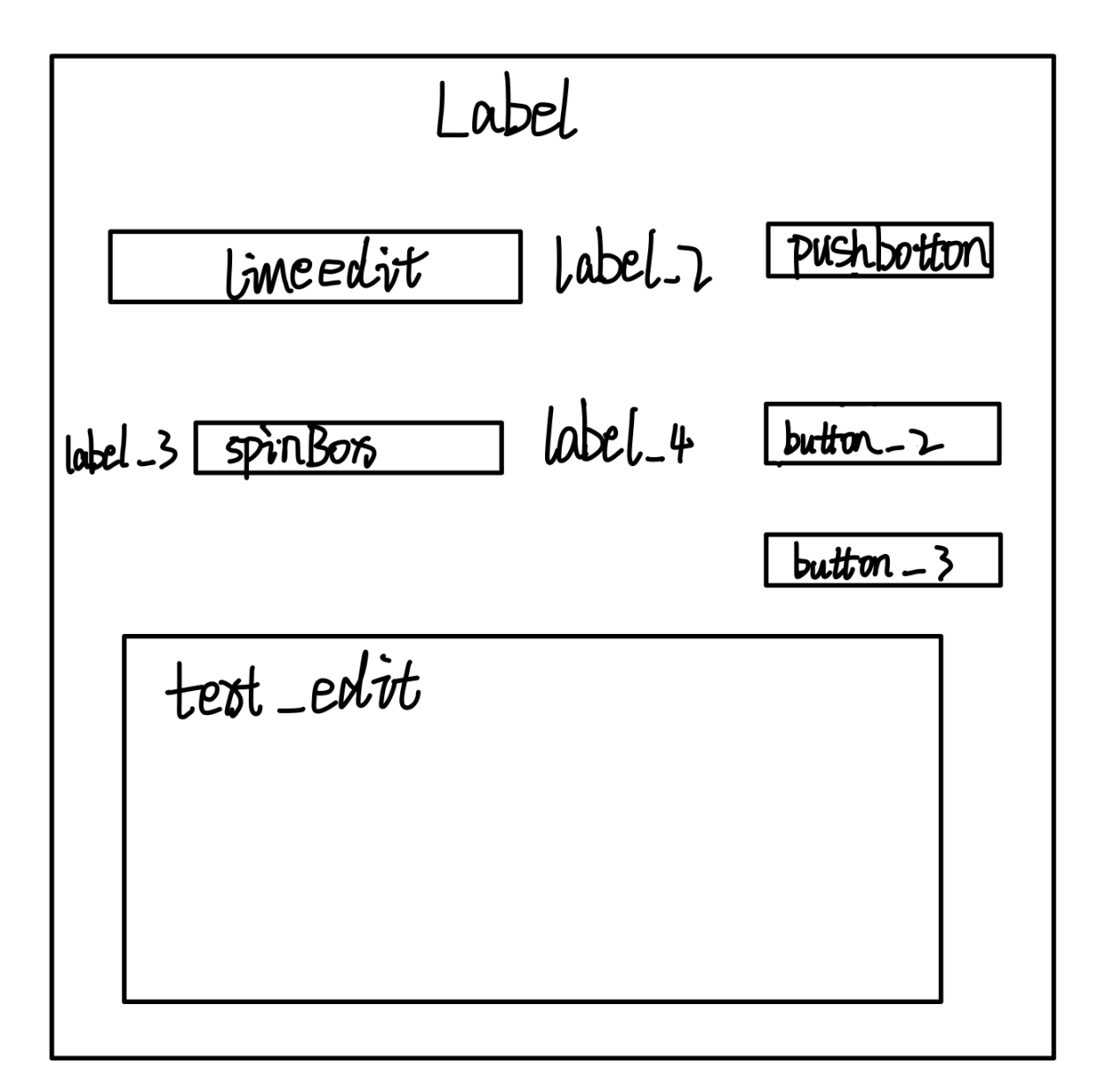


图1

### 相关代码

**串口初始化：**

serial=new QSerialPort;

//串口初始化

serial->setPortName("COM10");

serial->open(QIODevice::ReadWrite);

serial->setBaudRate(9600);

serial->setDataBits(QSerialPort::Data8);

serial->setParity(QSerialPort::NoParity);

serial->setStopBits(QSerialPort::OneStop);

serial->setFlowControl(QSerialPort::NoFlowControl);

connect(serial,SIGNAL(readyRead()),this,SLOT(dataReceive()));

**串口发送槽函数：**

void Widget::on\_pushButton\_clicked()

{

QString sendme="1"+ui->lineEdit->text()+"#";

QByteArray sendbytes=(sendme).toLatin1();

//QByteArray sendbytes=ui->lineEdit->text().toLatin1();

//qDebug()<<sendbytes;

serial->write(sendbytes);

}

void Widget::on\_pushButton\_2\_clicked()

{

//QString sendme="0"+ui->lineEdit\_2->text()+"#";

QString sendme="0"+ui->spinBox->text()+"#";

QByteArray sendbytes=(sendme).toLatin1();

//QByteArray sendbytes=ui->lineEdit->text().toLatin1();

serial->write(sendbytes);

}

void Widget::on\_pushButton\_3\_clicked()

{

QString sendme="c#";

QByteArray sendbytes=sendme.toLatin1();

serial->write(sendbytes);

}

**串口接收槽函数：**

void Widget::dataReceive()

{

QString buf;

buf=serial->readAll();

qDebug()<<buf;

ui->textEdit->append(buf);

}

### 操作逻辑流程图

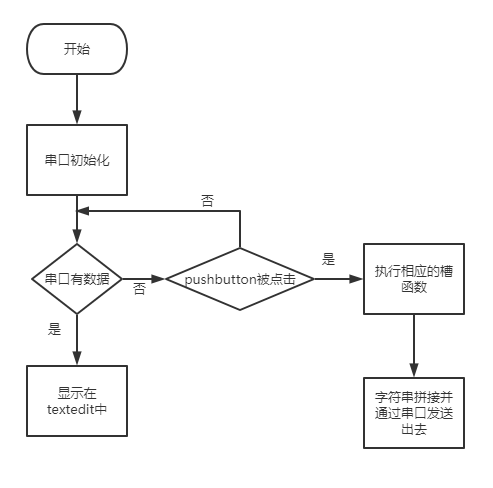


图2 操作逻辑流程图

### 详细设计

|  |  |
| --- | --- |
| 所用对象 | 所属类 |
| Label | QLable |
| Label\_2 | QLable |
| Label\_3 | QLable |
| Label\_4 | QLable |
| pushbutton | QPushButton |
| Pushbutton\_2 | QPushButton |
| Pushbutton\_3 | QPushButton |
| spinbox | QSpinBox |
| textEdit | QTextEdit |
| lineEdit | QLineEdit |

|  |  |
| --- | --- |
| 槽函数 | 功能 |
| **dataReceive**() | 接受串口发送过来的数据并显示在textedit上 |
| **on\_pushButton\_clicked**() | 充款按钮按下后获取lineedit的数据并拼接上“1”表示是充款，“#”表示数据结束符，拼接完成后通过串口发送出去 |
| **on\_pushButton\_2\_clicked**() | 扣款按钮按下后获取spinBox的数据并拼接上“0”表示是扣款，“#”表示数据结束符，拼接完成后通过串口发送出去 |
| **on\_pushButton\_3\_clicked**() | 通过串口发送“c#”字符串，表示清零指令 |

## 调试

### 实际运行图



图3 运行图1

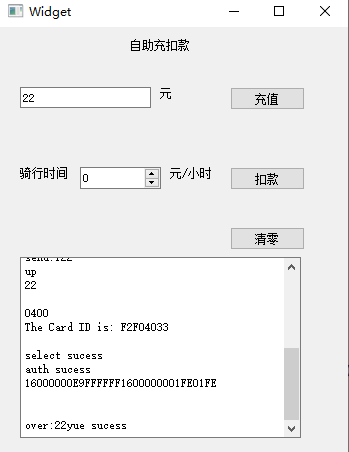


图4 运行图2

### 问题及解决

串口发送的数据必须通过比特流的方式进行发送，同时串口的初始化，比如波特率等数据一定要相互对应，否则容易出错。

# 登录界面

## 设计

### 界面设计图

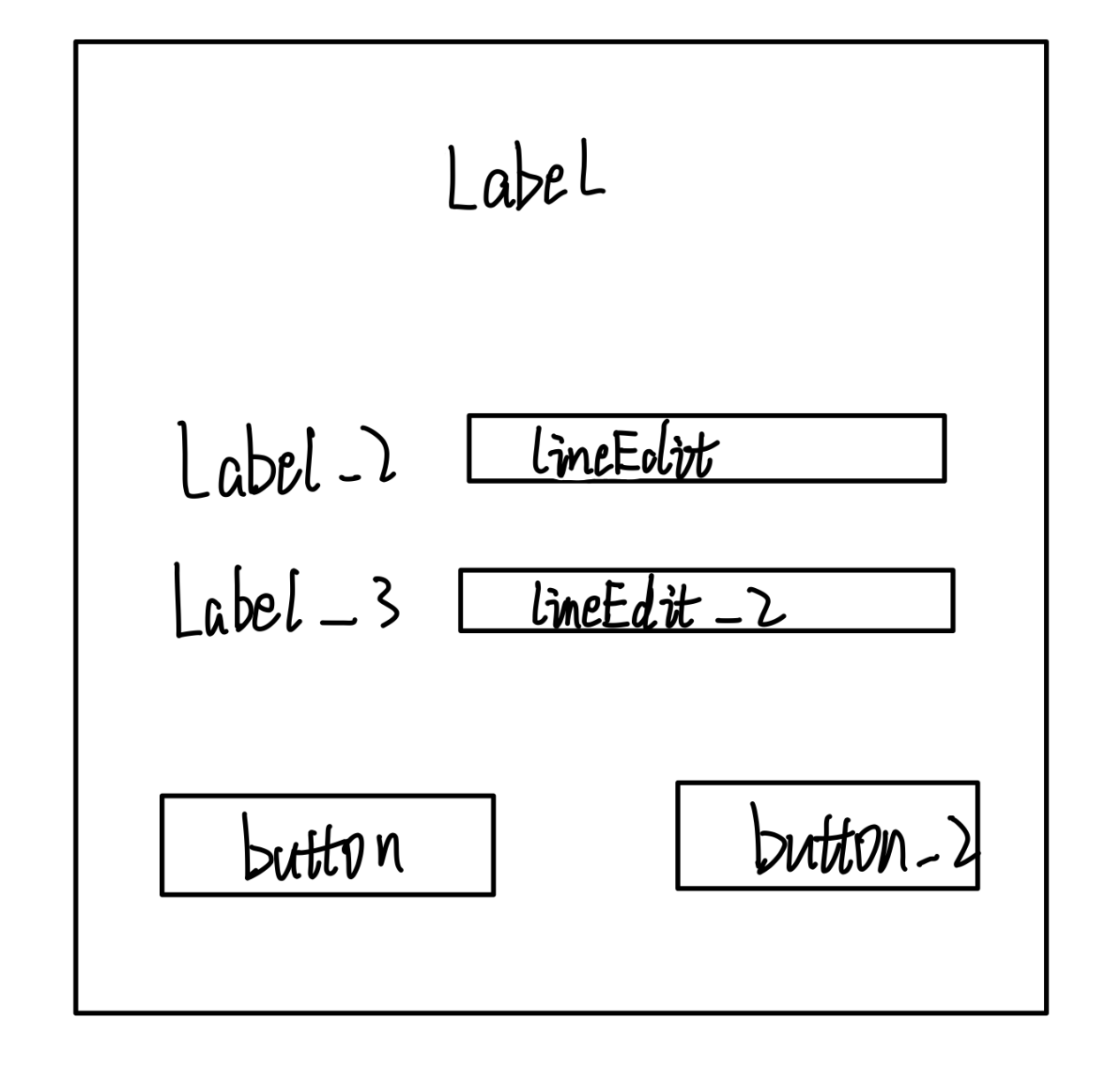


图5 界面设计图

## 相关代码

##### 数据库头文件：

#include <QSqlDatabase>

#include <QMessageBox>

#include <QSqlError>

#include <QSqlQuery>

#include <QVariantList>

##### 数据库初始化：

//数据库

qDebug()<<QSqlDatabase::drivers();

//打开数据库

QSqlDatabase db;

if(QSqlDatabase::contains("qt\_sql\_default\_connection")){

db = QSqlDatabase::database("qt\_sql\_default\_connection");

}

else{

db=QSqlDatabase::addDatabase("QSQLITE");

}

db.setDatabaseName("user.db");

if(!db.open()){

QMessageBox::warning(this,"错误",db.lastError().text());

return;

}

QSqlQuery deleteall;

deleteall.exec("delete from user;");

//创建表

QSqlQuery query;

query.exec("create table user(id varchar(255),name varchar(255),passwd varchar(255));");

query.prepare("insert into user(id,name,passwd) values(?,?,?);");

QVariantList idlist;

idlist<<"11"<<"22"<<"33";

QVariantList namelist;

namelist<<"11"<<"ming"<<"li";

QVariantList passwd;

passwd<<"11"<<"22"<<"33";

query.addBindValue(idlist);

query.addBindValue(namelist);

query.addBindValue(passwd);

query.execBatch();

##### 用户名密码验证槽函数代码：

void MyWidget::on\_b1\_clicked()

{

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

if(id==NULL){

QMessageBox::warning(this,"警告","请输入用户ID");

}

else{

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

if(passwd==NULL){

QMessageBox::warning(this,"警告","请输入密码");

}

else{

QSqlQuery check;

QString temp=QString("select \* from user where id='%1' and passwd ='%2'").arg(id).arg(passwd);

check.exec(temp);

QString checkin;

while(check.next()){

checkin=check.value(1).toString();

}

if(checkin==NULL){

QMessageBox::warning(this,"警告","用户名或密码输入错误");

}

else{

this->hide();

c.show();

}

}

}

}

### 2.1.4. 逻辑流程图

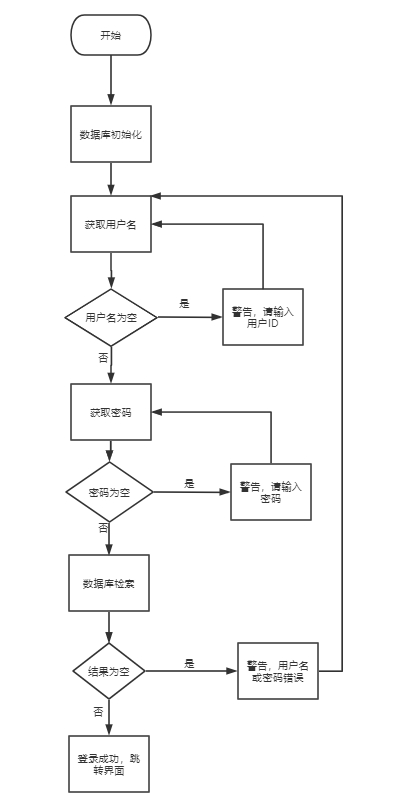


图6 逻辑流程图

## 调试

### 程序运行截图



图7 界面运行截图

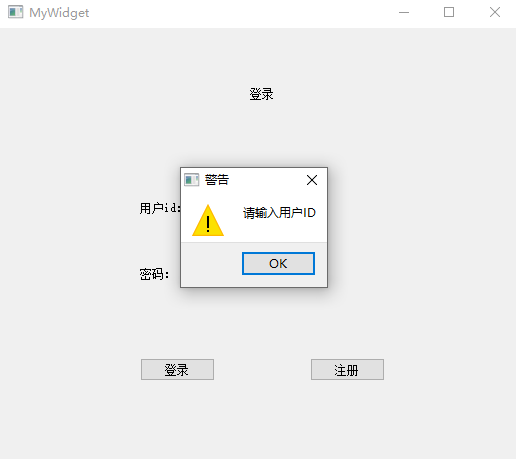


图8 用户名为空报错

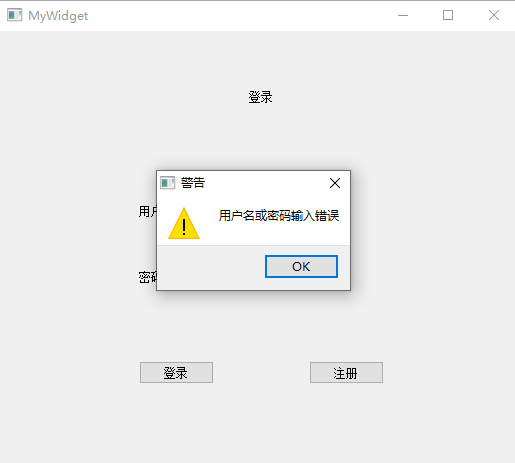


图9 数据库检索结果为空报错

### 出现问题和解决

页面相互切换的问题：从一个页面切换到另一个页面简单，但是从另一个页面再切换回来就很难了，最后解决办法是，将要跳转的页面作为主界面的一个类，通过子页面发送一个自定义的信号，让主页面捕捉到，在运行相应的槽函数就可以实现了。

# 车辆信息管理页面

## 设计

### 界面设计

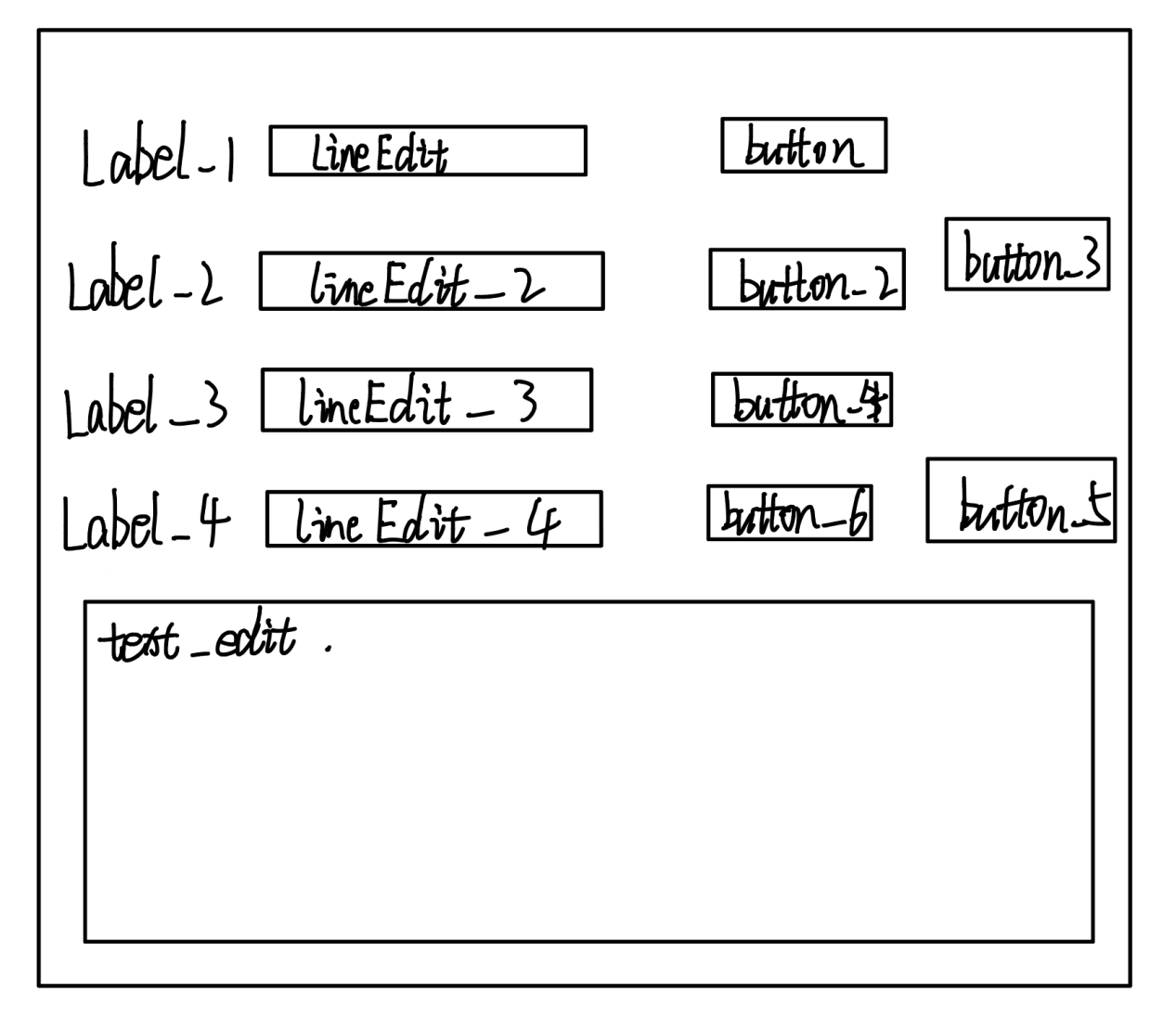


图10 界面设计图示

### 相关代码

数据库初始化：

qDebug() << QSqlDatabase::drivers();//打印qt支持的数据库类型

//打开数据库

QSqlDatabase db;

if(QSqlDatabase::contains("qt\_sql\_default\_connection")){

db = QSqlDatabase::database("qt\_sql\_default\_connection");

}

else{

db=QSqlDatabase::addDatabase("QSQLITE");

}

db.setDatabaseName("user.db");

if(!db.open()){

QMessageBox::warning(this,"错误",db.lastError().text());

return;

}

QSqlQuery query;

query.exec("create table carinfo(id varchar(255),location varchar(255),bx varchar(255),bro\_info varchar(255));");

query.exec("delete from carinfo;");

槽函数---插入一条信息：

void carinfo::on\_insert\_button\_clicked()

{

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

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

QString bx=ui->lineEdit\_3->text();

QString bro\_info=ui->lineEdit\_4->text();

if(id==NULL||location==NULL||bx==NULL||bro\_info==NULL){

QMessageBox::warning(this,"警告","请输入完整信息");

}

else{

QSqlQuery query;

QString str=QString("select \* from carinfo where id = '%1'").arg(id);

query.exec(str);

QString checkin;

while(query.next()){

checkin=query.value(1).toString();

}

if(checkin==NULL){

query.prepare("insert into carinfo(id,location,bx,bro\_info) values(?,?,?,?);");

QVariantList idlist;

idlist<<id;

QVariantList locatlist;

locatlist<<location;

QVariantList bxlist;

bxlist<<bx;

QVariantList brolist;

brolist<<bro\_info;

query.addBindValue(idlist);

query.addBindValue(locatlist);

query.addBindValue(bxlist);

query.addBindValue(brolist);

query.execBatch();

QMessageBox::warning(this,"SUCESS","插入成功");

ui->lineEdit->clear();

ui->lineEdit\_2->clear();

ui->lineEdit\_3->clear();

ui->lineEdit\_4->clear();

}

else{

QMessageBox::warning(this,"警告","已存在相同信息");

}

}

}

槽函数----显示所有的信息：

void carinfo::on\_query\_a\_button\_clicked()

{

QString id[100];

QString location[100];

QString bx[100];

QString bro\_info[100];

int i=0;

QSqlQuery query;

query.exec("select \* from carinfo;");

while(query.next()){

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

//qDebug()<<query.value(0).toString();

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

bx[i]=query.value(2).toString();

bro\_info[i]=query.value(3).toString();

i++;

}

ui->textEdit->clear();

int j=0;

for(j=0;j<i;j++){

QString str=QString("车辆ID:%1 位置:%2 保修情况:%3 借出情况:%4").arg(id[j]).arg(location[j]).arg(bx[j]).arg(bro\_info[j]);

ui->textEdit->append(str);

}

}

根据车辆ID查询车辆信息：

void carinfo::on\_query\_button\_clicked()

{

QString location,bx,bor;

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

if(id==NULL){

QMessageBox::warning(this,"警告","请输入车辆ID");

}

else{

QSqlQuery query;

QString str=QString("select \* from carinfo where id=%1").arg(id);

query.exec(str);

QString checkin;

while(query.next()){

checkin=query.value(1).toString();

}

if(checkin==NULL){

QMessageBox::warning(this,"警告","没有此车信息，请检查输入的信息");

}

else{

str=QString("select \* from carinfo where id=%1").arg(id);

query.exec(str);

while(query.next()){

location=query.value(1).toString();

bx=query.value(2).toString();

bor=query.value(3).toString();

str=QString("车辆ID:%1 位置:%2 保修情况:%3 借出情况:%4").arg(id).arg(location).arg(bx).arg(bor);

ui->textEdit->setText(str);

}

}

}

}

### 插入数据流程图

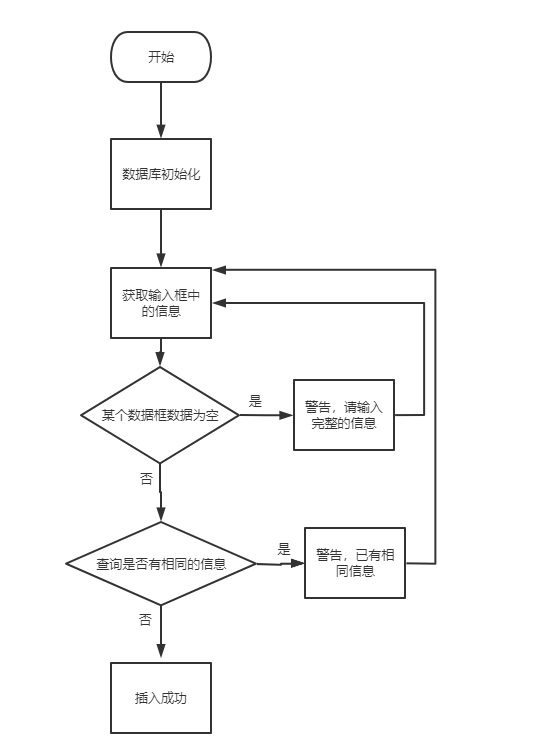


图11 插入数据流程图

### 显示所有的数据流程图

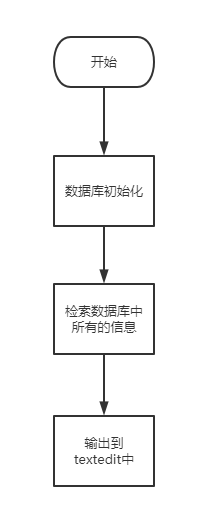


图12 显示所有的数据流程图

### 查询车辆信息流程图

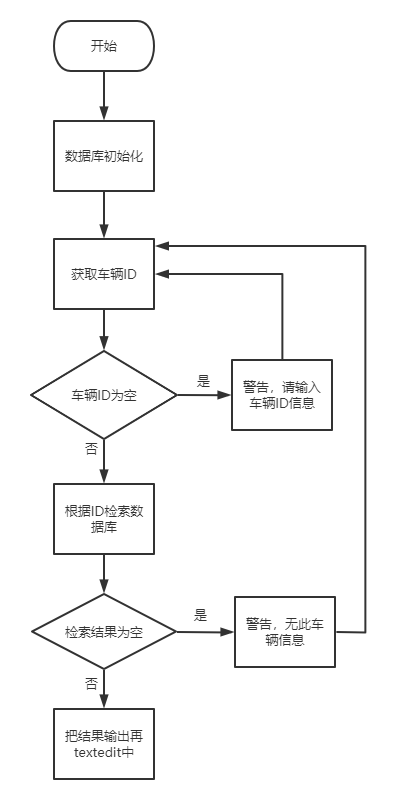


图13 查询车辆信息流程图

## 调试

### 3.2.1 运行截图



图13 程序界面

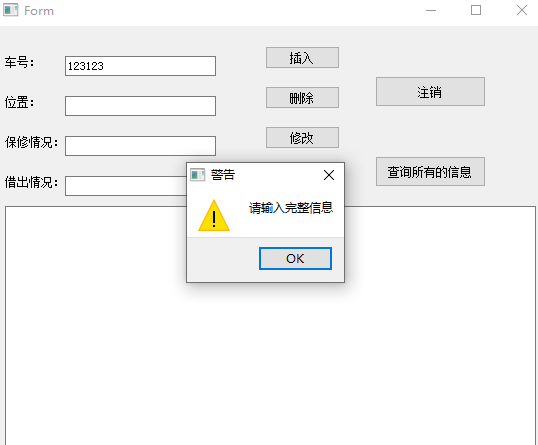


图14 信息不完整报错

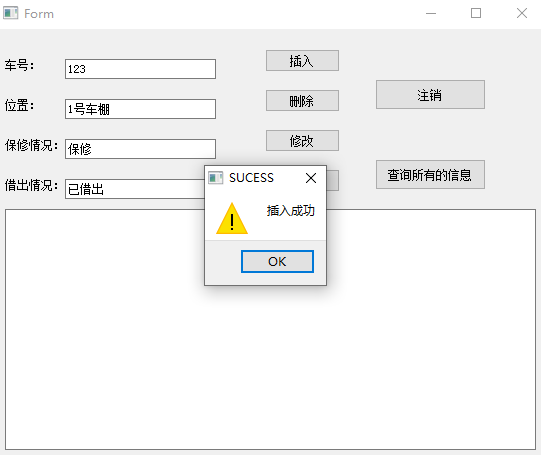


图15 插入成功



图16 查询所有的信息

### 3.2.2. 问题

整个程序是不可以同时开启两个数据库的，一开始我同时设计了两个数据库，一个是注册界面的数据库，保存用户的用户名和密码，另一个数据库是车辆信息的数据库，保存了车辆的信息。我同时打开两个数据库的时候，出现了第二个打开的数据库无法对相关的命令相应的问题。最后我通过一个数据库，两个不同的表解决了这个问题。