东软教育科技集团

项目实训总结报告

项 目 名 称:智慧医疗管理系统

学 校 名 称:北京理工大学

项 目 讲 师:

姓 名:汤深文

学 号:1120212493

实 训 日 期: 2023年8月21日-8月31日

**1 引言**

针对智能医疗管理系统，我们团队设计多个功能，包括注册登陆功能、医生方面包括了查看病人挂号信息、编辑病历、编辑医生个人信息、编辑处方和缴费单功能，患者方面包括挂号、查看病历、查看处方和缴费单、编辑个人信息、查看医生信息、健康评估功能，以及基于TCP通信协议的医患沟通平台。实训中所设计的智能医疗管理系统虽然和实际应用的大型系统之间仍存在较大差距，但是已经结合数据库实现基本功能，能够完成基本实际医疗操作步骤。

1.1项目概述

项目的目的是体会从需求理解出发，到软件整体设计，详细设计，开发，测试，发布的整体流程。熟悉软件开发整体过程，了解Windows操作系统下的应用程序开发方法，掌握Qt Creator以及连接数据库的使用，培养快速学习新知识，并在项目中使用的能力及独立解决问题的能力，掌握开源软件技术Qt图形库的使用，提高初级ui设计能力，掌握程序并发的开发方法包括多进程(process)编程及多线程(pthread)编程，掌握socket通信编程方法。

项目的背景和意义是随着人们生活水平的提高和健康意识的增强，医疗服务需求得到持续释放。但因为传统医疗行业的特性，产生了一系列的问题，包括：患者需要在家里和医院之间来回跑、优质医疗资源过度集中、排队难、挂号难、分级诊疗推行效率低、医院医生病患三者关系痛点多等问题，导致了人民群众“看病难、看病贵”。另外，随着计算机技术的发展，未来几年移动医疗健康市场的发展步伐将会明显提升，为了助力我国卫生医疗资源增长，进一步健全医疗服务体系，通过借助计算机相关技术建立新的医患沟通渠道、患患交流社区成为了一大热点，如果能基于安卓和iOS等移动终端系统进行开发出一款符合移动医疗健康市场的智能医疗管理系统，实现核心医疗服务内容的移动化，为医疗卫生服务提供了一种有效方法，解决因医疗人力资源短缺，导致的“看病难、看病贵”等医疗问题。

项目的参考资料是Linux系统编程课件以及Qt Creator使用教程课件。

1.2 运行环境

硬件要求：256 MB of RAM;500 MHz CPU (1 GHz preferred);OpenGL ES 2.0 support

软件环境：Windows系统，Windows下的qt-opensource；如果想看数据库端的具体内容需要有MySQL及其驱动、Windows系统自带的ODBC数据源。

**2 需求分析**

主要有三个方面：患者端，数据库端，医生端的功能。

患者端功能模块如下：（1）账户注册；（2）登录账号；（3）个人信息编辑；（4）挂号；（5）查看医生信息；（6）查看病例；（7）查看医嘱；（9）查看处方；（10）医患沟通；（11）健康评估。

数据库显示终端功能模块如下：（1）医生、患者的账号及密码；（2）病人基本信息、医生基本信息；（3）病人病历详情；（4）挂号信息；（5）医生诊断详情；（6）医嘱；（7）处方等。

医生端功能模块如下：（1）账户注册；（2）登录账号；（3）编辑个人信息；（4）查看预约患者信息，病历详情；（5）作出诊断，留下医嘱

**3 概要设计**

我们组主要将需求分成了几个部分进行实现：登录系统、编辑系统、处方系统、数据库查看系统、通信系统、挂号系统

**4 实现方案**（以个人主要负责内容为主）

个人负责内容如下：



挂号系统的实现：首先有多个挂号诊室，每个诊室对应着3个医生，然后选择挂号诊室后可以自动跳转到相对应的医生，然后会自动显示剩余预约位，调整预约日期以及预约时间，也会自动显示相对应时间端的剩余预约位，默认剩余预约位是20；

核心代码：

QString id1 = ui->lineEdit\_4->text();//插入数据

QString name1 = ui->lineEdit->text();

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

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

QString location = ui->comboBox\_2->currentText();

QString doctor = ui->comboBox\_3->currentText();

QString room = ui->comboBox->currentText();

QString time = ui->comboBox\_4->currentText();

QDate date = ui->dateEdit->date(); // 日期

QString year = QString::number(date.year()); // 年

QString month = QString::number(date.month()); // 月

QString day = QString::number(date.day()); // 日

QString date\_ = year + "." + month + "." + day + ".";

QSqlQuery sq;

QString sql\_exist="select \* from registration WHERE `时间`=? AND `日期`= ? AND `医生` = ?";

sq.prepare(sql\_exist);

sq.addBindValue(time);

sq.addBindValue(date\_);

sq.addBindValue(doctor);

bool isExist = sq.exec();//看是否存在已有预约

sq.next();

if(isExist){//如果存在，先判断是否已满，不满就将剩余预约人数减一

int available = sq.value(9).toString().toInt()-1;

qDebug()<<available<<"asdasdas";

if(available<=0){

QMessageBox::information(this, "Tips", "该时段预约已满! ", QMessageBox::Ok);

//这里应该弹出该时段预约已满的提示框

return;

}

QString str = QString("insert into registration(`身份证号`,`姓名`,`性别`,`年龄`,`院区`,`诊室`,`医生`,`日期`,`时间`)values('%1','%2','%3','%4','%5','%6','%7','%8','%9')").arg(id1).arg(name1).arg(sex).arg(age).arg(location).arg(room).arg(doctor).arg(date\_).arg(time);

QSqlQuery query;

bool isAddData = query.exec(str);

if(isAddData)

{

qDebug() << "插入数据成功";

}

else {

qDebug() << "插入数据失败";

qDebug() << query.lastError().text(); // 输出错误原因方便纠错

}

QSqlQuery sq;

QString sql="UPDATE registration SET `剩余预约位`=? WHERE `时间`=? AND `日期`= ? AND `医生` = ?";

sq.prepare(sql);

sq.addBindValue(available);

sq.addBindValue(time);

sq.addBindValue(date\_);

sq.addBindValue(doctor);

if(sq.exec())

{

QMessageBox::information(this,"通知","成功预约");

}else

{

QString text=sq.lastError().text();

QMessageBox::warning(this,"提示",text);

}

}

else//如果不存在，就将剩余预约人数设置为20

{

QString str = QString("insert into registration(`身份证号`,`姓名`,`性别`,`年龄`,`院区`,`诊室`,`医生`,`日期`,`时间`)values('%1','%2','%3','%4','%5','%6','%7','%8','%9')").arg(id1).arg(name1).arg(sex).arg(age).arg(location).arg(room).arg(doctor).arg(date\_).arg(time);

QSqlQuery query;

bool isAddData = query.exec(str);

if(isAddData)

{

qDebug() << "插入数据成功";

}

else {

qDebug() << "插入数据失败";

qDebug() << query.lastError().text(); // 输出错误原因方便纠错

}

QSqlQuery sq;

QString sql="UPDATE registration SET `剩余预约位`=? WHERE `时间`=?";

//预处理

sq.prepare(sql);

//绑定

QString time = ui->comboBox\_4->currentText();

sq.addBindValue(20);

sq.addBindValue(time);

//执行

if(sq.exec())

{

QMessageBox::information(this,"通知","成功预约，你是第一个预约者");

}else

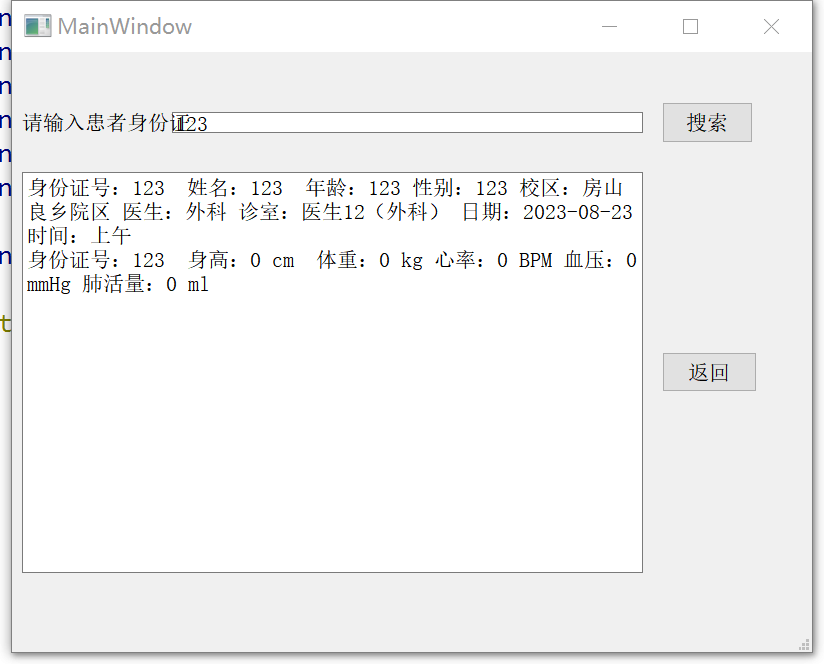
{

QString text=sq.lastError().text();

QMessageBox::warning(this,"提示",text);

}

}



医生查看患者挂号信息以及患者相关信息的实现： 因为医生查看患者挂号信息以及患者相关信息的操作跟代码类似，因此我直接放在同一个窗口下进行实现，医生可以通过输入患者的身份证号，来进行查看患者的挂号信息和患者的健康信息。

相关代码实现：QString id[100]; //用来存储从数据库中找出来的信息

QString name[100];

QString age[100];

QString sex[100];

QString location[100];

QString doctor[100];

QString room[100];

QString date[100];

QString time[100];

int i = 0;

QString query="select \* from registration WHERE `身份证号`=?";

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

QSqlQuery sq;

sq.prepare(query);

sq.addBindValue(id\_number);

sq.exec();//查询所有的信息

while(sq.next())

{

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

name[i] = sq.value(1).toString();

age[i] = sq.value(2).toString();

sex[i] = sq.value(3).toString();

location[i] = sq.value(4).toString();

doctor[i] = sq.value(5).toString();

room[i] = sq.value(6).toString();

date[i] = sq.value(7).toString();

time[i] = sq.value(8).toString();

i++;

}

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

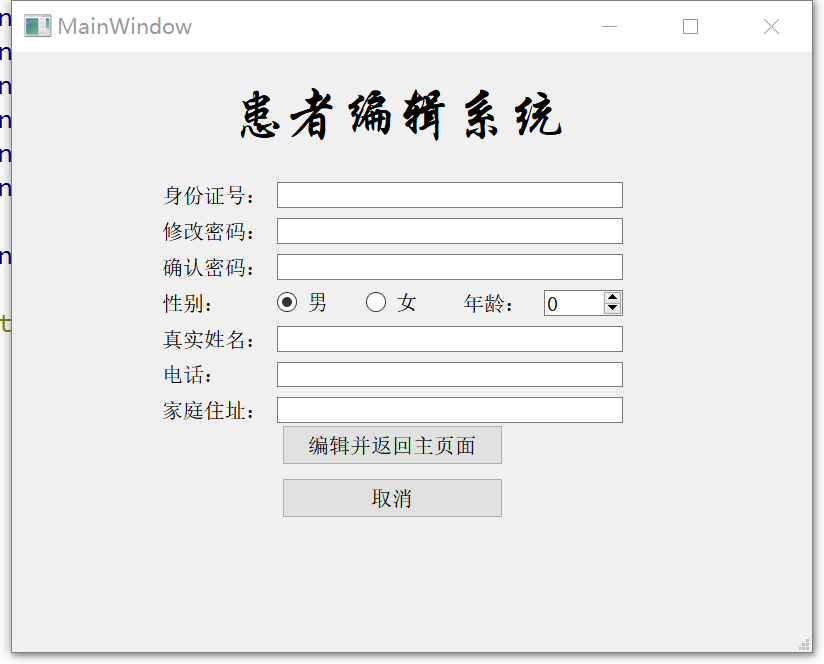
{

QString str = QString("身份证号：%1 姓名：%2 年龄：%3 性别：%4 校区：%5 医生：%6 诊室：%7 日期：%8 时间：%9").arg(id[j]).arg(name[j]).arg(age[j]).arg(sex[j]).arg(location[j]).arg(doctor[j]).arg(room[j]).arg(date[j]).arg(time[j]);

printf("/n");

ui->textEdit->append(str);

}



患者编辑系统：将患者的编辑系统跟数据库连接起来，然后在患者编辑的时候，需要先输入身份证号，这个地方身份证号码需要跟登录的时候身份证号码对照起来，如果此次输入的身份证号码与你登录的时候输入的身份证号码不一致，将会出现编辑失败——身份证号码填写错误提示。

相关代码：QString t1 = ui->lineEdit->text();//sfzh

QString t2 = ui->lineEdit\_2->text();//password

QString t4 = "0";

int temp2 = qb\_2->checkedId();

if(temp2==4)

{

t4 = "1";

}

QString t5 = ui->spinBox->text();//age

QString t6 = ui->lineEdit\_6->text();//real\_name

QString t7 = ui->lineEdit\_3->text();//tel

QString t8 = ui->lineEdit\_4->text();//address

QSqlQuery result = db.exec(" select \* from message");

result.clear();

QSqlQuery sq;

QString sql=QString("UPDATE message SET password='%1',gender='%2', age='%3' , real\_name='%4' , tel='%5' ,address='%6' WHERE user\_id='%7' ").arg(passwd).arg(t4).arg(t5).arg(t6).arg(t7).arg(t8).arg(name);

bool result\_successed = result.exec(sql);

if(result\_successed){

qDebug()<<"编辑成功！";

}else {

qDebug()<<"编辑失败！";

qDebug() << sq.lastError().text();

}

**5 系统测试**

通过多个测试跟操作进行debug，在挂号系统的实现中存在多个bug，特别是在自动显示剩余预约位的位置，很多错误都是由于数据库的相关代码引起的，很多插入查询语句都在最开始编写的时候出现了bug。

**6 项目经验及心得体会**

这一次开发收获颇多，不仅仅学会了MySQL的安装使用，页面化实现，收获最多的属于是在数据库连接qt上，对于相关代码的实现和使用，使得我对于数据库的连接更加熟悉，同时qt中的ui页面开发也很有意思，让我感受到了开发项目的乐趣。

（正文，五号宋体，1.25倍行距）